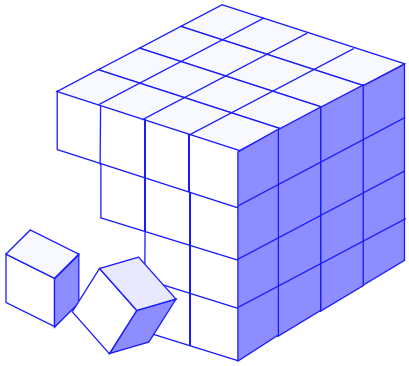
■ IODE ■

Version 6



Guide de l'utilisateur

Avril 2019

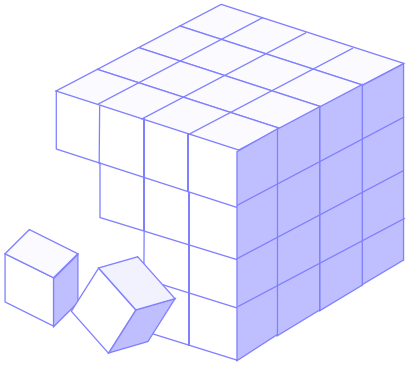


Table des matières

DISPONIBILITÉ DE IODE

IODE est actuellement disponible sous Windows. Des portages ont été réalisés sous différentes versions de Unix et de Linux mais ne sont plus aujourd'hui maintenues.

DOCUMENTATION

Le manuel de IODE est disponible sous la forme d'un fichier d'aide en ligne Windows (.chm), en format pdf et sur le site web.

Les modifications successives de IODE sont regroupées dans le manuel en ligne et sur le site du logiciel. Consultez-le pour obtenir les informations les plus récentes.

http://iode.plan.be

SUPPORT

Un support peut être assuré par courrier électronique via l'adresse **iode@plan.be**.

PRÉREQUIS

Ce logiciel s'adresse essentiellement à des économètres et ou à des statisticiens. Bien que décrites dans le texte, les notions essentielles relatives au développement de modèles économétriques et à la manipulation des séries statistiques sont indispensables pour pouvoir tirer profit des fonctionnalités économétriques de IODE.

Cependant, toute la partie de gestion de séries, de tableaux et de graphiques ne demande aucune connaissance particulière.

REMERCIEMENTS

Le logiciel IODE est le successeur de KaA, premier logiciel économétrique intégré développé dans les années 80 par le Bureau du Plan sur un mainframe IBM. Henri Bogaert a été présent à chaque étape des développements successifs. Il a notamment pris une part déterminante dans la mise au point du logiciel d'estimation. Son travail a contribué pour une large part à l'élaboration des concepts utilisés dans IODE.

Tanguy de Biolley et Joost Verlinden ont tous deux eu la patience d'étudier et de tester les premières versions de IODE. Les desiderata qu'ils ont exprimés ont conduit à affiner certaines fonctions du logiciel en les confrontant aux réalités du travail de l'économètre.

De nombreux autres utilisateurs de KaA puis de IODE ont apporté une aide précieuse au fil des années, que ce soit en signalant des erreurs ou en proposant des idées de développements. Citons, parmi d'autres, Francis Bossier, Cathy Streel, Jacques Floridor, Thérèse Père, Marie-Jeanne Festjens, Luc Masure, Albert Gilot, Henk Becquaert, Bernard Kahn, Michel Englert, Micheline Lambrechts, Thierry Bréchet, Koen Hendrickx, Igor Lebrun, Michel Saintrain, Bart Hertveld, Ludovic Dobbelaere, Antoine Melon, Gina Gentil, Vincent Frogneux, Francis Stallaert (Service d'Etudes des Finances), Arnaud Fougeyrollas (Erasme, Paris), Didier Baudewyns.

Enfin, la patience et le soin apportés par Herman Dekens dans la relecture et les corrections des manuels ont été particulièrement appréciés et ont largement contribué à la qualité de la présentation.

AUTEURS

IODE est développé par la Cellule Informatique du Bureau fédéral du Plan, plus particulièrement par Geert Bryon et Jean-Marc Paul.

DATE DE PUBLICATION

Cette version a été finalisée le 11 avril 2019.

CONTENU DU MANUEL

Le manuel de Iode est composé des chapitres suivants :

- Introduction au logiciel : présentation des notions de base (objets, workspaces, etc)
- Le programme IODE : gestion des objets, construction et exploitation des modèles économétriques
- Le langage LEC : syntaxe des formules dans IODE
- Les rapports : syntaxe et exemples de rapports
- Le programme IODECMD : utilisation de IODE sans interface
- Le programme A2M : interface de traduction des fichiers A2m
- Interfaces avec Excel, APL, ODBC : interfaces avec d'autres outils et langages
- Methods and algorithms : description des méthodes d'estimation et de simulation utilisées dans IODE
- Annexes : syntaxe des programmes et des fichiers ascii supportés par IODE, sample d'impression...

1. Introduction au logiciel

Le logiciel IODE a été mis au point avec l'appui scientifique d'une équipe d'économètres ayant en charge la construction et l'exploitation de grands modèles macroéconomiques. C'est en analysant le travail des membres de cette équipe et en tentant de répondre au mieux à leurs besoins que IODE, au départ confiné dans les aspects purement économétriques, s'est élargi à des fonctionnalités plus larges que celles nécessaires à la construction d'un modèle et aux simulations.

En effet, si la construction d'un modèle - rédaction des équations, estimation, tests et simulations - est la partie la plus lourde de l'élaboration d'un modèle, l'exploitation en aval des informations produites par le modèle représente une part considérable de l'emploi du temps des économètres. En amont, la construction des séries de base ou l'importation de celles-ci à partir d'informations externes représente également un travail non négligeable.

IODE s'attaque donc à toutes les étapes de la construction et de l'exploitation des modèles :

- l'importation et l'exportation des séries vers et à partir d'autres logiciels (TSP, Excel, KaA, LArray (Python), E-Views...),
- l'automatisation de l a construction des séries à partir de données provenant de plusieurs ensembles de données de base,
- la documentation des bases de données statistiques,
- la rédaction des équations,
- l'estimation des équations par différentes méthodes,
- les simulations de scénarios,
- la recherche d'objectifs (goal seeking),
- la production de tableaux et de graphiques,
- la génération automatisée de rapports intégrant tableaux, graphiques, texte libre, équations, etc,
- l'importation des outputs dans des logiciels externes comme Word, Excel, ou Frame Maker,
- l'automatisation via un langage de scripting propre.

Pour atteindre ces différents objectifs, l'utilisateur de IODE doit définir et utiliser différents "objets". Ces objets sont manipulés par IODE et stockés dans des espaces de travail ("workspaces").

La suite de ce chapitre aborde la définition des objets de IODE, indique l'organisation des programmes et fournit les notions de base de l'interface utilisateur. Elle se compose de trois parties :

- Les concepts et objets manipulés par IODE (notion d'objets et de workspaces),
- Organisation des fichiers qui introduit les différents éléments et programmes composant le logiciel,
- L'interface utilisateur de IODE qui décrit l'écran de base et ses différentes composantes.

1.1 Les concepts et objets manipulés

IODE opère sur des objets de différents types. Ceux-ci sont groupés dans des ensembles appelés workspaces (WS). Ces deux notions sont fondamentales pour la bonne compréhension du logiciel. Elles font l'objet des deux premières sections suivantes.

Différents types d'objets - par exemple les équations - utilisent des formules dans leur définition. Un langage standard, spécialement adapté aux expressions économétriques, est utilisé dans tous ces objets. Ce langage, le lec (pour "Langage Econométrique Condensé"), est rapidement présenté dans troisième section.

La dernière section est consacrée aux rapports, qui, via un langage de scripting, permettent d'automatiser les fonctions de IODE.

1.1.1 LES OBJETS

Avant de détailler le contenu de chaque type d'objet, commençons par analyser la justification de chacun d'eux. En d'autres termes, essayons de voir où interviennent, dans le cadre d'un modèle, les concepts auxquels ils sont associés.

Un modèle est un système d'EQUATIONS qui sont des formules faisant intervenir des VARIABLES, séries numériques temporelles définies sur une période de temps donnée, avec une fréquence déterminée (annuelles, trimestrielles, etc). Les équations peuvent contenir des coefficients - éventuellement estimés - qui sont des variables sans dimension, appelées dans IODE des SCALAIRES.

EQUATIONS, VARIABLES et SCALAIRES sont trois types d'objets gérés par IODE.

Les variables elles-mêmes ne sont pas toutes obtenues telles quelles, mais résultent le plus souvent de calculs basés sur d'autres variables, provenant éventuellement de plusieurs sources. Ces calculs peuvent être par exemple une agrégation des secteurs ou une dimension géographique ou encore des séries mises en base commune. Les formules utilisées pour effectuer le calcul de ces variables contruites sont appelées IDENTITES de construction.

Le nom donné à chaque VARIABLE ne permet en général pas d'en indiquer le contenu avec suffisamment de précision. IODE permet de créer des COMMENTAIRES dont le nom sera identique à celui des VARIABLES qu'ils définissent. Ces commentaires sont simlement des textes libres.

COMMENTAIRES ET IDENTITES sont les quatrième et cinquième objets gérés par IODE.

Lorsqu'on dispose de variables, il est souvent utile des les présenter - à l'écran ou sur papier - sous forme de tableaux ou de graphiques. IODE permet de construire à cette fin des TABLEAUX, non pas de valeurs, mais de formules et de texte.

Ces "tableaux" sont en quelque sorte des petits programmes qui permettent de construire des tables de nombres ou des graphiques sur base de VARIABLES contenues dans les fichiers de variables. Cette façon de procéder est très efficace : les mêmes tableaux sont réutilisés pour imprimer différentes versions des VARIABLES (après la simulation d'un scénario par exemple). Le même tableau peut aussi être imprimé en taux de croissance, en comparaison de fichiers, etc.

Le TABLEAU est le sixième type d'objet de IODE.

Il n'y a pas de notion de modèle en tant qu'objet dans IODE : un modèle est simplement une LISTE d'équations. Pour éviter le travail fastidieux de réencodage des listes, celles-ci sont gérées comme des objets à part entière et - comme on le verra plus tard - sauvées dans des fichiers de listes. L'utilisation des listes est par ailleurs omniprésente : on les trouve dans les formules pour en raccourcir l'écriture, en paramètre des fonctions de rapports pour simplifier l'encodage, etc.

La LISTE termine l'énumération des objets de IODE. Les objets gérés par IODE sont donc au nombre de sept.

Les objets des sept types sont identifiés par un nom de 20 caractères maximum (à partir de la version

6.01) commençant toujours par une lettre (ou le caractère '_'), en minuscule pour les SCALAIRES et en majuscule pour les autres objets (ceci afin de distinguer les scalaires et les variables dans les formules du langage LEC sans ambigüité).

Chaque type d'objet a donc trouvé sa justification. La suite du chapitre peut être consacrée à une définition plus détaillée de leur contenu.

Les commentaires

Les commentaires sont des textes libres. Un éditeur intégré à IODE permet de les encoder. Ils servent à documenter des objets de IODE.

Les équations

Une équation se présente comme une égalité qui fera partie d'un modèle. Chaque équation est composée des éléments suivants :

- la forme LEC (langage d'écriture des formules dans IODE)
- un commentaire libre (titre de l'équation)
- la méthode par laquelle elle a été estimée (s'il y a lieu)
- la période d'estimation éventuelle
- les noms des équations estimées simultanément
- les instruments utilisés pour l'estimation

Tous ces éléments de définition sont présents dans chaque équation, mais éventuellement laissés vides s'ils n'ont pas de justification.

Le nom d'une équation est celui de sa variable endogène.

Les identités

Une identité de construction est une expression en langage LEC qui permet de construire une nouvelle série statistique sur base de séries déjà définies. En général, les identités sont "exécutées" en groupe pour constituer ou mettre à jour un ensemble de variables.

Il ne faut par confondre identités et équations. Les identités au sens du logiciel TROLL sont des équations (ou parfois des listes) dans IODE.

Les listes

Les listes sont, comme les commentaires, des textes libres. Elles sont utilisées pour simplifier l'écriture en différentes circonstances :

- liste d'équations définissant un modèle
- liste de tableaux à imprimer
- argument quelconque d'une fonction (période d'impression par exemple)
- macro dans une équation, une identité ou un tableau
- etc

Les scalaires

Les scalaires sont pour l'essentiel des coefficients estimés d'équations économétriques. Pour cette

raison, chaque scalaire contient dans sa définition :

- sa valeur
- le paramètre de "relaxation", fixé à 0 pour bloquer le coefficient lors d'une estimation
- son t-test, résultat de la dernière estimation
- sa déviation standard, résultat de la dernière estimation

Seule la valeur du scalaire a de l'intérêt lors du calcul d'une expression LEC. Les trois autres valeurs n'ont de sens que pour l'estimation.

Comme les autres objets, les scalaires sont identifiés par un nom de 10 caractères maximum (20 à partir de la version 6.01). Ceux-ci doivent être en minuscules pour que les variables soient distinctes des scalaires dans les formules LEC.

Les tableaux

Une des opérations le plus souvent effectuées dans le cours d'un exercice de simulation est l'affichage ou l'impression de tableaux et de graphiques de résultats. Les tableaux ont été imaginés pour rendre cette opération la plus efficace possible.

Chaque tableau est un ensemble de lignes. Une ligne est composée de deux parties (en général) :

- une partie de texte qui sera le titre de la ligne
- une partie formule qui permettra de calculer les valeurs à placer dans le tableau

```
TITRE DU TABLEAU
----------------
Produit national brut       PNB
Chômage                     UL
Balance extérieure          X-I
```

Les lignes sont en fait de plusieurs types : titre centré sur la largeur de la page, lignes de valeurs, lignes de séparation, références, etc.

Un tableau peut être calculé sur différentes périodes de temps, décrites par un "sample" du type :

```
1980Y1:10               --> 10 observations à partir de 1980Y1
1980Y1, 1985Y1, 1990:5  --> 1980, 1985, puis 5 observations à partir
                            de 1990Y1
80/79:5                 --> 5 taux de croissance à partir de 1980
...
```

Il peut de plus contenir des valeurs en provenance de différents fichiers :

```
(1990:5)[1,2,1-2]    --> valeurs de 1990 à 1994 pour les fichiers
                         1, 2 et pour la différence entre les deux
                         fichiers.
```

Le résultat du calcul peut être :

- affiché à l'écran
- imprimé
- intégré dans un rapport

et ce, soit sous forme de tableau de nombres, soit sous forme de graphique.

Les tableaux peuvent très bien être utilisés dans le cadre d'un projet n'intégrant pas de modèle économétrique : les seules informations utilisées par les tableaux sont les variables et éventuellement les scalaires.

Les variables

Les variables sont simplement des séries de nombres. Comme les autres objets, elles sont organisées en workspaces (WS), notion qui est étudiée ci-dessous.

Toutes les variables d'un WS sont définies sur la même période (sample). Si des observations sont manquantes, elles prennent la valeur spéciale NA (Not Available) représentée par --.

Comme les autres objets, les variables sont identifiées par un nom de 10 caractères maximum (20 à partir de la version 6.01). Ceux-ci doivent être en majuscules pour que les variables soient distinctes des scalaires dans les formules LEC.

1.1.2 Les workspaces

On a vu précédemment qu'il y avait 7 types d'objets gérés dans IODE. Lors d'une session de IODE, l'espace mémoire est également divisé en 7 parties : chacune est réservée à un des types d'objets et est appelée workspace (WS = espace de travail). On a donc en permanence 7 WS "actifs" au cours d'une session de travail.

Au départ, tous les WS sont vides. Des fonctions, étudiées dans les chapitres suivants, permettent de modifier le contenu des WS, soit en agissant sur des WS entiers, soit sur des objets individuels.

Les WS sont stockés en mémoire pendant le durée d'une session IODE. En quittant le programme, toutes les données sont perdues. Pour les sauvegarder, il faut utiliser les fonctions de sauvetage de WS.

Parmi les fonctions agissant sur les WS entiers, on trouve notamment :

- Load : charge en mémoire le contenu d'un fichier
- Save : stocke l'actuel contenu de WS dans un fichier
- Copy : ajoute à un WS des objets copiés d'un fichier
- Clear : détruit tous les objets d'un WS
- Sample : change la période de définition du WS de variables

D'autres fonctions agissent sur la définition des objets : création, modification, destruction, etc.

Les fonctions économétriques (estimation, simulation) et la fonction de construction de séries sur base d'identités agissent indirectement sur les variables et les scalaires en modifiant leurs valeurs.

Résumé

- En début de session, les WS sont chargés en mémoire à partir de fichiers
- En cours de session, les objets sont modifiés, créés, détruits dans les WS
- En fin de session (ou à n'importe quel moment), il faut sauver les WS dans des fichiers sur disque de façon à pouvoir les récupérer dans des sessions ultérieures.

1.1.3 Le LEC, langage des formules de IODE

Qu'il s'agisse d'équations, d'identités de construction ou encore de graphiques, un langage d'écritu-

re de formules mathématiques adéquat est indispensable. Le langage LEC (Langage Econométrique Condensé) offre l'avantage d'être à la fois concis dans son écriture et particulièrement adapté aux formules faisant intervenir des séries chronologiques. Il est utilisé chaque fois qu'une formule est nécessaire dans le logiciel IODE.

Le LEC est également "naturel", en ce sens que sa syntaxe est proche de l'écriture de formules que l'on peut trouver dans la littérature.

```
Equation de consommation
------------------------
                 Yt
    Ct = a + b ---- + c.C                          (texte)
                 Pt      t-1

    C := a + b * Y / P + c * C[-1]                  (LEC)


Equation de production
----------------------
    ln Qt = a ln Kt + (1 - a) ln Lt + c.t + b       (texte)

    ln Q := a * ln K + (1 - a) * ln L + c * t + b   (LEC)
```

Le LEC est décrit en détail dans un chapitre séparé. Citons simplement ici quelques caractéristiques intéressantes dont certaines apparaissent dans les exemples ci-dessus :

- plus de 20 opérateurs mathématiques sont intégrés dans le langage : fonctions trigonométriques, hyperboliques, logarithmes, exponentielles, max, min, etc
- plus de 10 fonctions temporelles : maximum, minimum, somme et produit sur une période, lags, leads, différences et taux de croissance de degré quelconque (y compris calculé), moyennes mobiles, écarts type, etc
- les lags, leads et périodes sont écrits très simplement, y compris sur des expressions et peuvent être combinés :

```
(A + B)[-1]
UL[+1]
(X + dln Y)[1985Y1]
```

- des parties de formules peuvent être temporairement annulées en les mettant en commentaire :

```
ln Q := a * ln K /* + (1 - a) ln (L + Y + Z) */
```

- des listes (macros) peuvent être utilisées dans des formules :

```
A + B := c1 + c2 * $LL + c3 * $ZZ
```

1.1.4 Les rapports : scripter IODE

Un rapport est un fichier ASCII (extension .rep) contenant deux types d'éléments :

- des instructions (les lignes qui commencent par $ ou #)
- du texte libre (les autres lignes)

L'exécution d'un rapport se traduit d'une part par l'enchaînement d'opérations (charger un fichier, estimer des équations, imprimer un tableau, etc) et d'autre part par l'impression d'un document "fi-

ni".

Les instructions commençant par $ sont des instructions qui n'affichent rien à l'écran. Celles commençant par # font intervenir l'écran, soit pour afficher des informations, soit pour poser des questions.

On trouve plusieurs groupes d'instructions :

- celles permettant de contrôler le déroulement d'un rapport (Label, GoTo, Ask, Return, Quit, OnError, Foreach, ProcExec, etc)
- celles exécutant des fonctions de IODE (WsLoadEqs, WsSaveVar, FileDeleteA2m, DataEditVar, PrintTbl, ModelSimulate, ReportExec, etc)

Les instructions du second type permettent en fait d'effectuer pratiquement toutes les opérations effectuées normalement à partir des menus de IODE.

Avant d'être exécutée, chaque ligne de rapport est analysée et, si nécessaire, modifiée : on peut ainsi utiliser des macros, remplacer des listes par leur valeur, effectuer des calculs...

Exemples de rapports

L'exemple qui suit charge des Workspace IODE de différents types. Ensuite, il fixe la destination des impressions (ici dans un fichier de format intermédiaire, étudié plus tard). Enfin, un beep sonore est produit et un message indique que l'environnement est chargé.

Typiquement, l'exécution de ce rapport débute une session IODE en fixant un environnement initial.

```
$WsLoadVar fun\fun.var
$WsLoadEqs fun\fun.eqs
$WsLoadScl fun\fun.scl

$PrintDest tmp.a2m A2M
#Beep
#Show Environnement fun chargé
```

Le rapport suivant définit le fichier output de l'exécution (bist92\bistelf1.rtf), affiche ensuite un message, construit les tableaux et les imprime. Du texte libre est intégré dans ce rapport ("TABLEAUX DES HYPOTHESES" par exemple).

```
$PrintDest bist92\bistelf1.rtf RTF
#Show processing french tables
$PrintNbdec 1
    TABLEAUX DES HYPOTHESES
    -----------------------

$PrintTbl 89:8 HYPEIR
$PrintTbl 89/88:8 HYPEIIR

$PrintNbdec 0
.page
    TABLEAUX DES RESULTATS
    ----------------------

$PrintTbl 89:8 RESL00

$PrintNbdec 1
$PrintTbl 89/88:8 RESL00R
$PrintTbl 89:8 RESL03
#Beep
```

Le format A2M

Le format A2M est décrit en détail dans un chapitre séparé.

En résumé, un fichier A2M est un texte ASCII entrecoupé d'instructions d'enrichissement typographiques et de structuration d'un document en paragraphes, tableaux et graphiques. Ce langage est utilisé en interne par IODE comme langage intermédiaire. Des traducteurs internes à IODE permettent ensuite d'imprimer un fichier A2M ou de le traduire vers des formats de fichiers comme Word ou Frame.

Il est également loisible à l'utilisateur d'enrichir ses impressions en plaçant dans ses rapports des commandes de formattage A2M. Celles-ci seront interprétées par IODE lors de l'envoi vers une imprimante ou vers un document MIF, HTML, RTF ou CSV.

L'utilisateur peut également décider de sauvegarder ce format A2M pour une impression ultérieure ou pour produire plusieurs formats différents.

Les programmes IODE et A2M interprètent un fichier au format A2M et permettent de générer :

- une impression sur une des imprimantes définies sous Windows
- un fichier Rtf destiné à l'importation par exemple dans MS-Word
- un fichier Html destiné à la publication sur Internet
- un fichier Csv destiné à l'importation dans MS-Excel ou Lotus
- un fichier Mif destiné à l'importation dans Frame Maker

Grâce à cette fonctionnalité, le choix de la destination des impressions permet d'importer aisément les états (tableaux, graphiques, etc) dans un programme de traitement de texte ou dans un tableur.

Notons que les fichiers A2M sont exploitables en dehors de IODE, à l'aide du programme **`scr4w_at.`**

1.2 ORGANISATION DES FICHIERS

Les fichiers suivants font partie du package :

- Programmes
  - iode.exe : le programme IODE avec interface
  - iodecmd.exe : le programme IODE sans interface (lancement de rapports)
  - iodecom.exe : la version de IODE pouvant être utilisée depuis un programme externe (Excel...)
- Manuels
  - iode.hlp, iode.cnt : manuel en ligne de IODE dans l'ancien format d'aide de Windows (nécessite une installation particulière à partir de Windows Vista)
  - iode.chm : manuel de IODE en format d'aide Windows
  - iode.pdf : manuel de IODE en format PDF
  - readme.htm : dernières modifications de IODE en format HTML

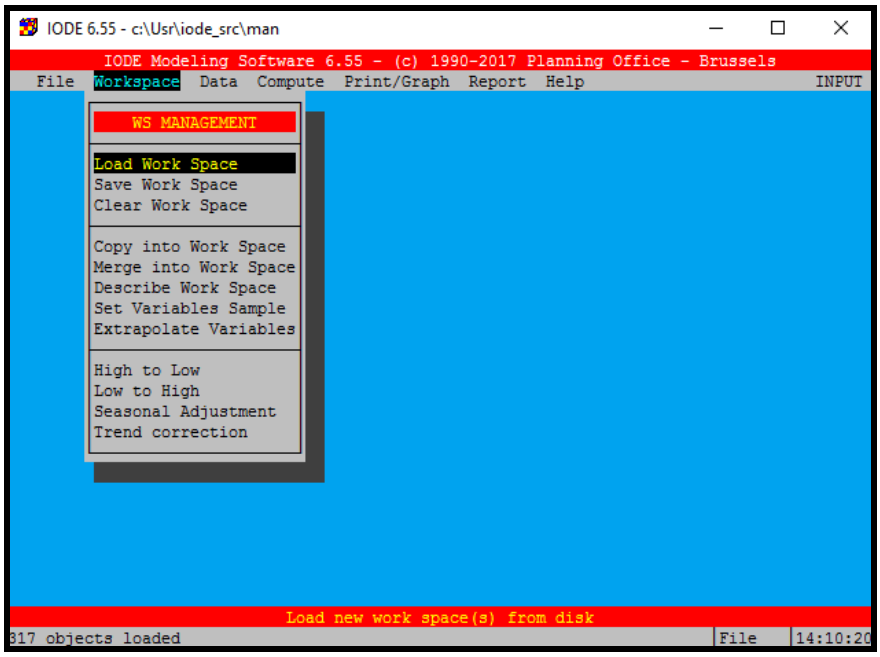
1.3 L'INTERFACE UTILISATEUR DE IODE

Cette section décrit les différents éléments qui composent l'écran de base de IODE et les moyens dont dispose l'utilisateur pour interagir avec le logiciel.

Le lexique qui se trouve en fin de manuel fournit également certaines informations sur l'interface.

L'ÉCRAN DE BASE

FIGURE 1



En allant de la première à la dernière ligne de l'écran, on recontre successivement: les éléments suivants :

Première ligne : version du logiciel

La première ligne indique le numéro de version et le nom du logiciel.

Deuxième ligne : menu déroulant

La seconde ligne contient un menu. Chaque option concerne un groupe de fonctions : "File" donne accès aux actions concernant les fichiers, "WS" à celles concernant les workspaces, etc. Pour accéder à une de ces options, il suffit de presser la touche ALT associée à la première lettre de l'option (ALT+F donne accès au menu FILE), ou de presser la souris sur l'option choisie.

Dans le coin supérieur droit de l'écran est affiché un mot qui peut prendre deux valeurs:

- INPUT qui indique que le programme attent
- WAIT qui indique que le programme est en train d'exécuter une fonction

Centre de l'écran : les données et les écrans de saisie

En fonction de l'option sélectionnée dans les menus, s'afficheront dans le centre de l'écran les don-

nées demandées ou un écran de saisie. Dans l'exemple ci-dessus, l'option "Load Workspace" du menu "Workspace" a été sélectionnée et l'écran permet de spécifier les fichiers à charger en mémoire.

Avant-dernière ligne de l'écran : aide rapide

Dans l'avant-dernière ligne de l'écran est affiché un message. Celui-ci est un commentaire indiquant les différentes touches utilisables et leur fonction ou simplement l'effet de la commande actuellement sélectionnée.

Dernière ligne de l'écran : message, printer/file, heure

La dernière ligne de l'écran contient, de gauche à droite, les informations suivantes :

- un message indiquant soit une erreur, soit l'état d'avancement du déroulement d'une fonction (simulation, estimation, $show dans un rapport, etc)
- la destination de l'impression qui peut être soit FILE pour rediriger toute impression vers un fichier, soit PRINTER.
- l'heure

LES ÉCRANS DE SAISIE

Le fonctionnement des écrans de saisie est assez intuitif. Quelques informations utiles sont reprises ci-dessous.

Touches spéciales

- `Ctrl+U` : vide le champ de son contenu
- `Ctrl+C` : copie la partie sélectionnée du champ
- `Ctrl+V` : colle la sélection précédemment copiée (dans n'importe quelle application)
- ENTER : selon le type de champ, le comportement varie :
  - : champ bouton : exécute l'action associée au bouton
  - : champ DIR : affiche les fichiers répondant au masque du champ
  - : champ EDITOR : entre en édition du texte du champ.
  - : champ MENU : ouvre le menu pour effectuer une sélection
- F10 : équivaut à presser le bouton Ok de la fenêtre.
- ESC ou F3 : équivaut à presser le bouton Escape de la fenêtre.

Champ menu

Le menu s'ouvre en pressant la touche ENTER ou en cliquant sur la souris. On s'y déplace avec les touches fléchées et de sauts de page haut et bas. L'option sélectionnée est validée avec la touche ENTER et le texte correspondant s'inscrit dans la page de saisie. On peut également sélectionner une option en tapant sa première lettre ou un espace pour passer d'option en option.

Champ DIR

Un champ de type DIR est un champ de saisie permettant la recherche d'un nom de fichier en fonction d'un masque de saisie. On commence par encoder dans le champ le masque de fichier exactement comme dans un champ normal de texte (p.ex. "*.var").

La touche ENTER affiche alors la liste des fichiers correspondant au masque de saisie. Les sous-répertoires sont également mentionnés. Le nom du fichier désiré peut alors être sélectionné à l'aide des touches fléchées et de sauts de pages haut et bas. Le nom choisi est validé par la touche ENTER. Ce nom s'inscrit dans la page de saisie à la place du masque de recherche. Si le nom sélectionné est un nom de répertoire, la recherche se poursuit dans ce répertoire. Ces champs fonctionnent par ailleurs comme un champ de texte normal.

Si aucun nom affiché dans le menu ne doit être sélectionné, la touche ESCAPE permet de quitter le menu sans exporter de nom dans le champ.

Dans le menu, on trouve pour chaque fichier plusieurs informations comme sa taille, la date de dernière modification, ses propriétés d'accès, ...

Champ EDITOR

Les champs EDITOR sont reconnaissables au "scrollbar" qui apparaît en général à leur gauche. Il sont le plus souvent encadrés.

Pour entrer en édition d'un champ EDITOR, il faut presser ENTER. On dispose alors d'un éditeur de texte complet, avec fonctions de copie/collage, formattage de texte, etc. Ce type de champ n'a pas de limite de taille. Il est en particulier utile pour la saisie de formules. Les principales fonctions sont citées ici :

- Pour quitter l'éditeur (et revenir au niveau de la PAGE de saisie) il suffit de presser ESCAPE ou Alt-Q.
- Pour marquer un bloc de texte (à copier), on utilisera les touches Alt-B, Alt-L ou Alt-T selon que le bloc à marquer est un rectangle (B), un groupe de lignes (L) ou une partie de texte (T).
- Pour copier le bloc marqué (en vue d'une copie), on utilisera `Ctrl+C`. Pour coller le contenu du clipboard à l'endroit du curseur, on pressera `Ctrl+V`. Cette copie peut se faire dans un autre texte que le texte courant (copie d'une équation à une autre par exemple).
- Pour détruire le bloc marqué et le placer dans le tampon, on utilisera `Ctrl+D`.

Si on dispose d'une souris, celle-ci offre des facilités pour le marquage (bouton de gauche) et la fonction copie/collage (bouton de droite).

LES TOUCHES FONCTIONS

Pour spécifier une action à effectuer, il y a plusieurs façons de procéder :

- une première est de presser la touche fonction (F<n>, ...) adéquate,
- une deuxième est de presser ENTER ou de cliquer sur le bouton d'un écran de saisie (qui correspond toujours à une touche fonction),
- une troisième enfin est de cliquer avec la souris dans la zone de commentaire (avant-dernière ligne de l'écran) sur le texte de la touche correspondant à la fonction désirée. Le texte de la touche se distingue du reste du commentaire par sa couleur.

Les touches fonctions (F1 à F10, Alt+F1 à Alt+F10, etc) sont de trois types : les touches locales à l'endroit de l'application où on se trouve, les touches globales qui s'exécutent en interrompant temporairement l'exécution du processus en cours et les touches permettant un accès rapide au menu déroulant.

Touches fonctions locales

Un cas évident où les touches fonctions locales sont indispensables est celui des écrans de saisie: une fois les données encodées, l'utilisateur doit pouvoir soit confirmer le traitement à effectuer, soit

décider d'abandonner la page en cours sans exécution. Ce choix est indiqué au programme par une touche fonction (en général F10 confirme et ESCAPE abandonne).

Toujours dans un écran de saisie, une touche peut permettre d'introduire des informations complémentaires ou de visualiser des données utiles. C'est le cas dans l'écran d'impression de tableaux : F8 permet d'imprimer, tandis que F10 affiche à l'écran les valeurs du tableau.

En toutes circonstances, les touches fonctions locales sont indiquées avec l'opération qu'elles déclenchent dans la ligne de commentaire de l'écran (avant-dernière ligne).

Touches fonctions globales

D'autres touches de fonctions ont un effet global : où que l'on se trouve dans l'application, ces touches interrompent le programme et exécutent une action déterminée. Le processus en cours à l'origine reprend après la fin de l'action associée à la touche fonction globale.

Dans le module IODE, la touche fonction globale F9 permet à tout moment, de voir l'état d'utilisation de la mémoire sans avoir à quitter l'endroit actuel de l'application.

Toutes les touches fonctions définies sont exposées en détail en fonction du contexte.

Touches fonctions du menu déroulant

D'autres touches globales sont celles permettant de se positionner directement dans une fonction du menu déroulant. La touche Alt associée à la première lettre d'une option du menu déroulant a pour effet de quitter la fonction en cours et de positionner l'application sur le point du menu correspondant (Alt + F = menu File).

Ces touches abandonnent par conséquent l'action en cours s'il y en a une.

Dans les sous-menus du menu déroulant, on trouve à droite de certaines options l'indication d'une touche fonction. Le fait de presser la touche en question abandonne l'action en cours et exécute directement la fonction indiquée dans le menu.

Par exemple, dans le menu "Misc", la touche Alt-X est associée à l'option "Quit". Presser Alt + X à n'importe quel moment permet de quitter le programme.

LA SOURIS

La souris facilite un certain nombre d'opérations dans l'utilisation de IODE :

- dans les écrans de saisie, un simple clic permet de positionner le curseur dans le champ voulu, et à la position souhaitée.
- sur les contours des fenêtres de saisie, d'aide ou de scroll, des petites icônes permettent d'effectuer certaines opérations :
  - [o] : fermer la fenêtre (= ESCAPE)
  - [=] : accès au menu de la fenêtre reprenant toutes les touches fonctions définies localement
  - [?] : accès au manuel en ligne
  - d'autres icônes permettent d'effectuer une rotation dans les scrolls, un déplacement sur les sujets précédents ou suivants dans les écrans d'aide, de maximiser et minimi-

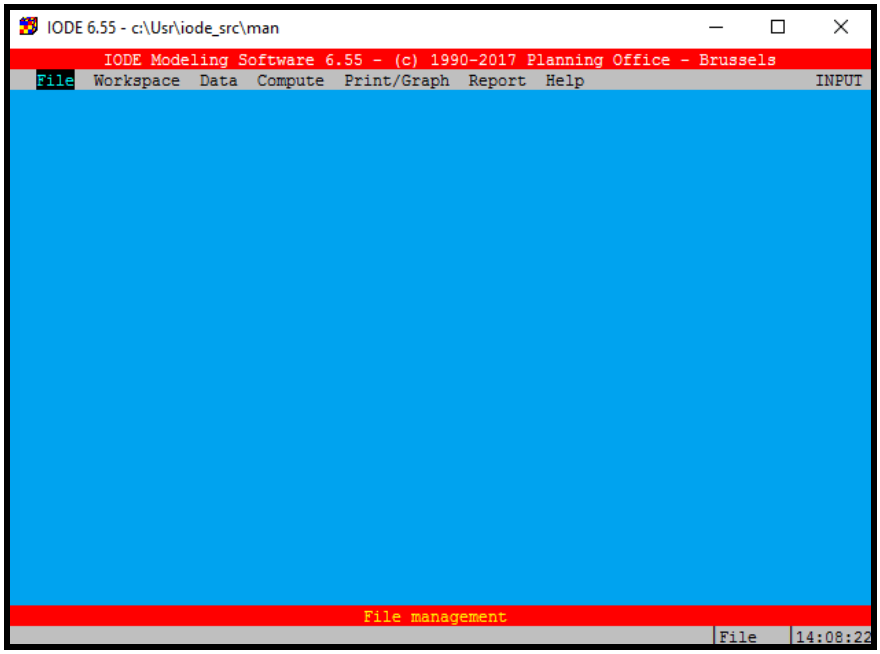
ser la taille de la fenêtre, etc.

- dans l'éditeur MMT (rapports, champs EDITOR), elle permet une sélection de bloc ou un accès à un menu simplifié pour les opérations de copie et collage (bouton de droite).
- lorsque certaines touches fonctions sont mises en exergue dans la zone de commentaire, la souris permet de "presser" ces touches sans avoir recours au clavier
- les fenêtres (pages, scroll, help, mmt) peuvent être déplacées en plaçant la souris sur le contour et en gardant le bouton appuyé pendant le déplacement
- la taille des mêmes fenêtres peut être modifiée en plaçant la souris sur le coin inférieur droit et en "tirant" tout en gardant enfoncé le bouton de la souris.
- les options du menu déroulant sont directement accessibles à l'aide de la souris, comme à l'aide des touches Alt-lettre.
- dans les objets ayant un scrollbar (MMT, champs EDITOR, Help, Menus, Scrolls) la souris permet de placer le curseur à l'endroit souhaité par un simple clic.

2. Le programme IODE

Ce chapitre décrit les principales options de la barre d'action du programme IODE et des sous-menus correspondants :

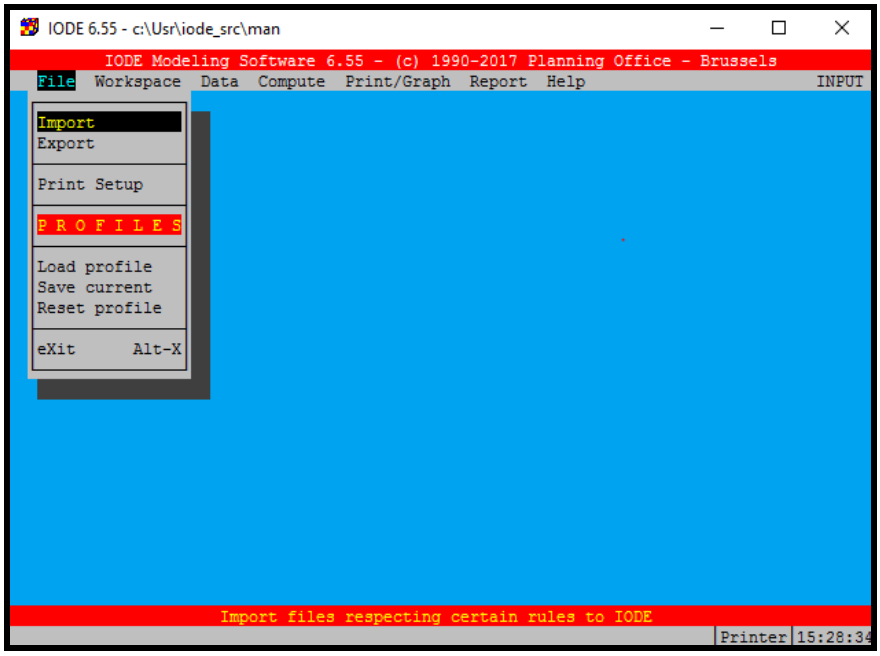
FIGURE 2



- File : Importation et exportation de fichiers, préférences, gestion de profils et fin de travail
- Workspace : Manipulation des workspaces
- Data : Manipulation des données
- Compute : Calcul d'identités, estimation et simulation de modèles
- Print/Graph : Impression des objets, tables et graphiques
- Report : Edition et exécution des rapports
- Help : Diverses fonctions d'aide

2.1 FILE

FIGURE 3



Les fonctions d'importation et d'exportation de données en format externe à IODE sont regroupées dans ce menu.

Ce menu permet également de gérer les fichiers de "profils" qui retiennent les dernières valeurs introduites dans les champs des pages de saisie.

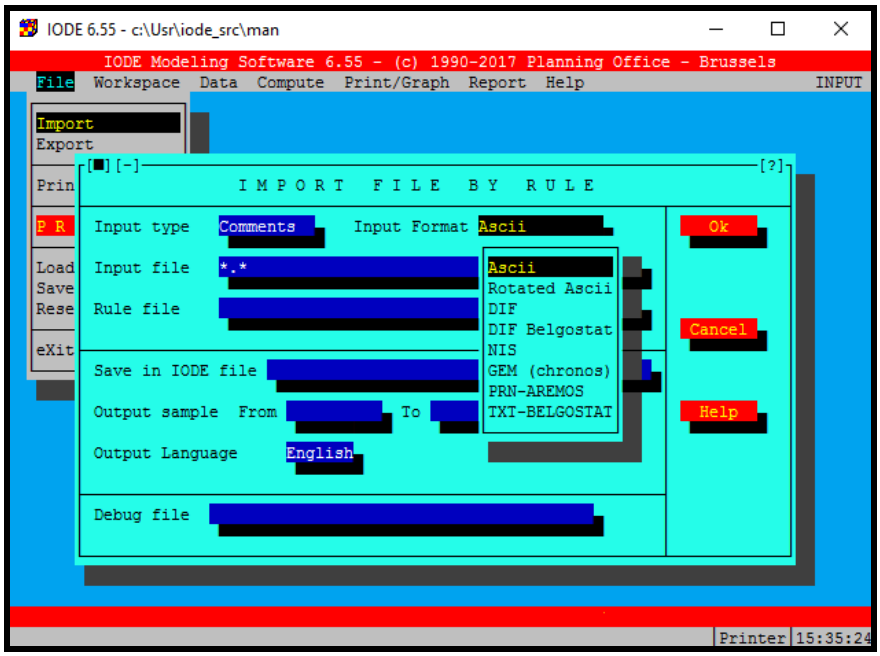
Ce paragraphe décrit les différentes manipulations de fichiers accessibles au départ du menu FILE de IODE:

- Import : Importation de données en format externe
- Export : Exportation de données en format externe
- Print Setup : Sélection des options d'impression
- Load profile : Charger un fichier "profile"
- Save current : Sauver le "profile" courant
- Reset profile : Réinitialiser les valeurs par défaut du "profile" courant
- Exit (Alt-X) : Quitter IODE

D'autres manipulations sur les fichiers de données sont également possibles à partir du menu *Workspace*.

2.1.1 IMPORT

FIGURE 4



Cette fonction concerne les importations d'objets, plus particulièrement des VARIABLES en différents formats actuellement moins utilisés comme DIF, ASCII, rotated ASCII, DIF Belgostat, TXT Belgostat et PRN Aremos.

La description des formats qui peuvent être interprétés par cette fonction peut être trouvée dans les annexes.

Le processus n'a pas d'effet sur les objets actuellement en WS : le résultat d'une importation se trouve dans un fichier IODE.

TRANSCODAGE DE NOMS DE SÉRIES

Un usage particulier de cette fonction d'importation est de définir des règles de transcodage des noms de séries. Ainsi, une série dont le code est par exemple `123456789` dans le fichier ASCII à importer peut être remplacée par n'importe quel nom au choix de l'utilisateur

Les champs de la page sont décrits ci-dessous.

Input type

Ce champ permet de spécifier le type d'objet à importer.

Input format

Les formats d'importation suivants peuvent être interprétés par IODE:

- Ascii : format Ascii des objets spécifique à IODE
- Rotated Ascii : format ascii de variables avec les séries en colonnes
- DIF : format DIF (Data Interchange Format)
- DIF Belgostat : format (ancien) d'échange spécifique à Belgostat
- NIS : format Ascii (ancien) de l'Institut National de Statistiques
- GEM : format Ascii du logiciel Chronos
- PRN-Aremos : format Ascii du logiciel Aremos
- TXT Belgostat : format (ancien) d'échange spécifique à Belgostat

Input file

Ce champ est du type DIR, il sert à préciser le nom du fichier de variables à importer.

Rule file

Champ DIR optionnel qui sert à préciser le nom du fichier de règles de transcodage à utiliser.

Save in IODE file

Ce champ DIR sert à préciser le nom du fichier de destination des objets importés.

Output Sample

Ces deux champs permettent de limiter la période des variables à importer (optionnel). Le format est celui d'un sample de IODE. Par exemple:

```
1980Y1
1990M11
```

Output Language

Ce champ ne sert que dans le cas où un texte apparaît en plusieurs langues dans un fichier de commentaires. Actuellement, seul le format DIF Belgostat exploite cette valeur et permet de choisir la langue des commentaires extraits.

Debug File

Ce dernier champ permet d'indiquer un fichier de trace des opérations de transcodage effectuées (optionnel).

DÉFINITION DES RULES

Les règles ("rules") sont définies dans un fichier externes et permettent essentiellement deux choses :

- limiter les variables à importer
- déterminer un nom d'objet à partir de l'information disponible dans le format Ascii.

Syntaxe des "rules"

Les règles sont définies dans un fichier ascii. Chaque règle est composée de deux champs :

- le masque (pattern) de sélection contenant une description des noms concernés par la règle. Ce masque se définit à la manière du search dans IODE (menu Data), c'est-à-dire en incluant des * et des ? pour spécifier des ensembles.
- l'algorithme de transcodage du nom pour cette règle qui peut contenir :
  - \+ : indique que le caractère doit être repris dans le nom
  - \- : indique que le caractère doit être sauté
  - n'importe quel autre caractère : repris dans le nom

Exemple

```
B*    C+-+        transforme B1234 en CB2, BCDEF en CBE, etc
*X    ++++++++++  garde les noms se terminant par X inchangés
```

Sélection via rules

Les règles définies permettent de ne conserver que les séries dont le nom répond à au moins une des règles. Toutes les autres sont éliminées.

En particulier la règle `* ++++++++++` sélectionne toutes les séries en gardant le nom (tronqué à 10 caractères, 20 à partir de la version 6.01). Si le fichier de règles est vide ou non défini, la règle

```
*  ++++++++++
```

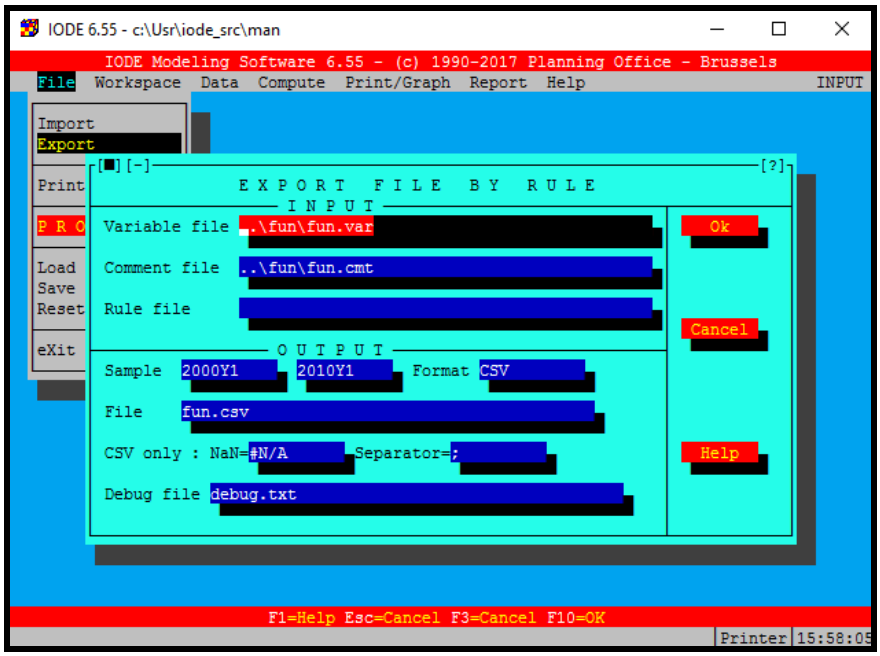
est utilisée.

Ordre d'exécution

Les règles s'exécutent dans l'ordre de définition. Dès qu'une règle s'applique à un nom, c'est elle qui est utilisée. On prendra donc soin de placer en début de fichier les exceptions et de terminer par les cas généraux.

2.1.2 EXPORT

FIGURE 5



Cette fonction permet de générer des fichiers de données en formats externes : Comma separated values (CSV), CSV inversé, TSP, WKS (Lotus/Excel), DIF.

Le processus charge temporairement un fichier IODE (Variable File). Il n'y a donc pas d'effet sur les objets actuellement en WS.

Les champs de l'écran de saisie son définis ci-dessous.

Variable file

Ce champ indique le nom du fichier contenant les variables à exporter. Il est requis.

Comment file

Ce champ permet de spécifier un fichier de commentaires. Les commentaires seront associés aux variables du même nom dans le fichier résultat lorsque le format le permet.

Rule file

Le fichier des "rules" permet de limiter les variables à exporter et éventuellement d'en changer les noms.

Sample

Si le sample n'est pas précisé, toutes les observations sont fournies dans le fichier résultat. Sinon, le fichier résultat sera limité au sample indiqué.

Format

Ce champ MENU permet de choisir le format du fichier output.

Dans le cas des fichiers WKS, un "Label" est associé à chaque range de valeurs, ce label étant identique au code de la série.

File

Ce champ DIR indique le nom du fichier résultat. Il est requis.

NaN et Separator

Dans le cas du format CSV, deux éléments sont libres : le séparateur de colonnes (par défaut `;`) et la représentation de la valeur inconnue (par défaut `N/A`).

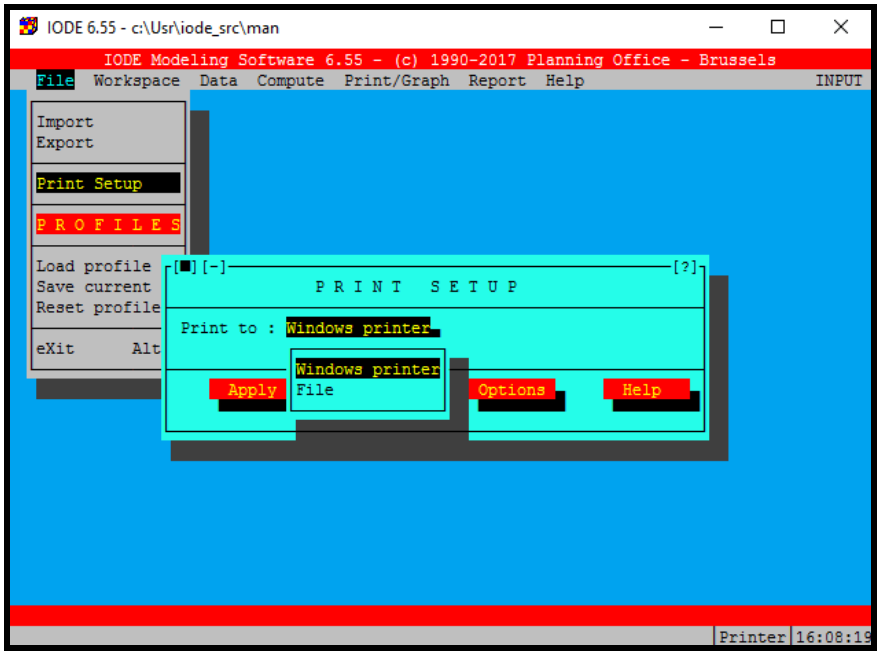
Debug File

Ce champ DIR permet de spécifier le nom d'un fichier qui contiendra une trace de l'exportation, en particulier de l'application des "rules".

Notons qu'il est également possible de générer des fichiers CSV (pour un tableur) à partir d'un fichier a2m ou lors d'une impression dans un fichier.

2.1.3 PRINT SETUP

FIGURE 6



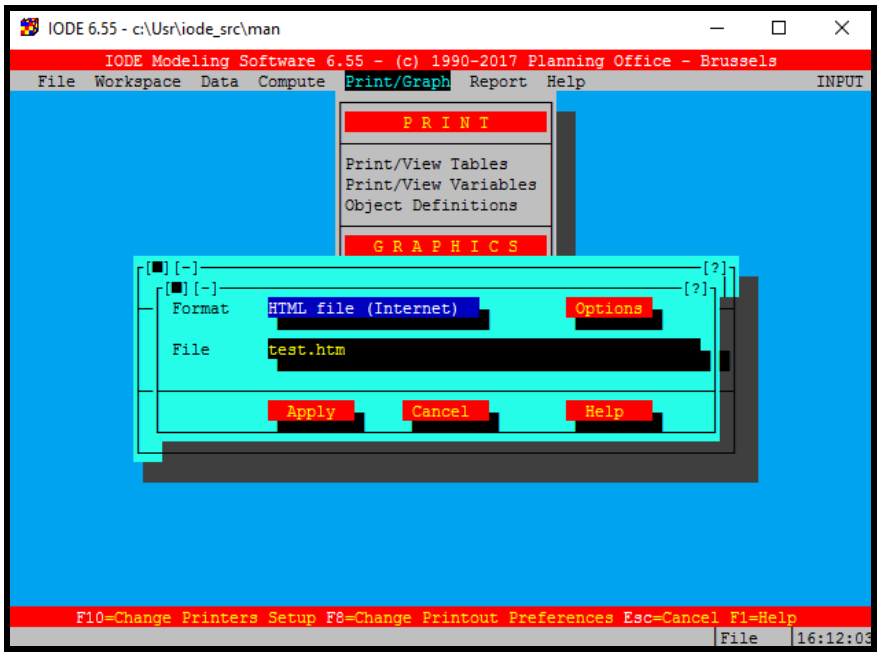
Cette fonction permet de préciser le type d'impression que l'on souhaite obtenir : dans un fichier ou sur une imprimante. Le choix effectué dans cet écran a un effet sur toutes les impressions.

Print To

Ce champ offre le choix entre une impression dans un fichier ou sur l'imprimante courante définie dans Windows.

Dans le second cas, toute impression est simplement dirigée vers l'imprimante. Dans le premier cas, à chaque nouvelle impression, un écran de saisie demandera le nom et le format du fichier destiné à recevoir le produit de l'impression.

FIGURE 7



Apply

Ce bouton permet d'enregistrer les choix pour la suite du programme.

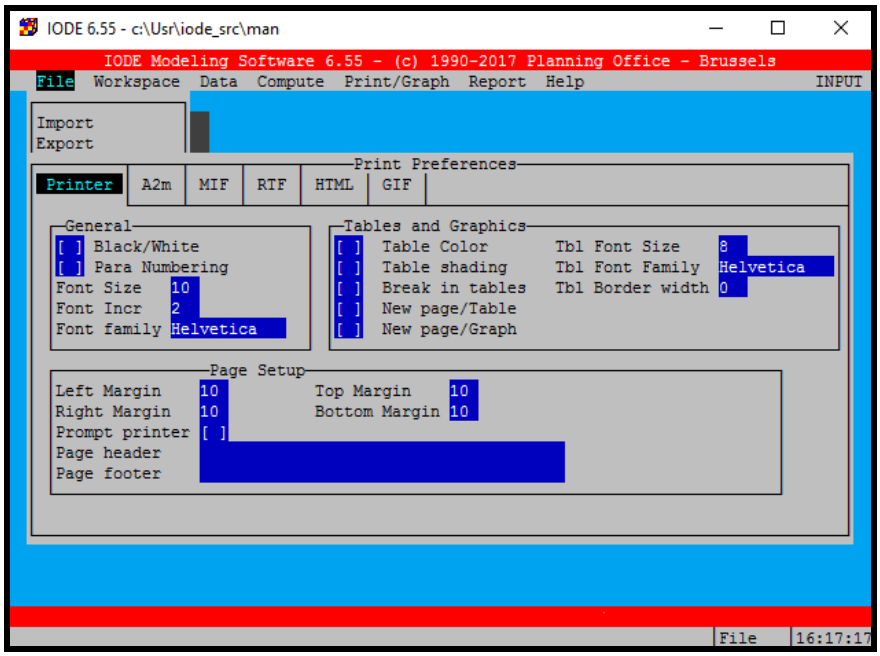
Options

Ce bouton donne accès à toutes les options de traduction et d'impression. Pour chaque destination d'impression ou de traduction, une série d'options sont disponibles: elles sont illustrées ci-dessous.

Ces options sont semblables à celles définies dans le programme A2M qui fait l'objet d'un chapitre séparé de ce manuel.

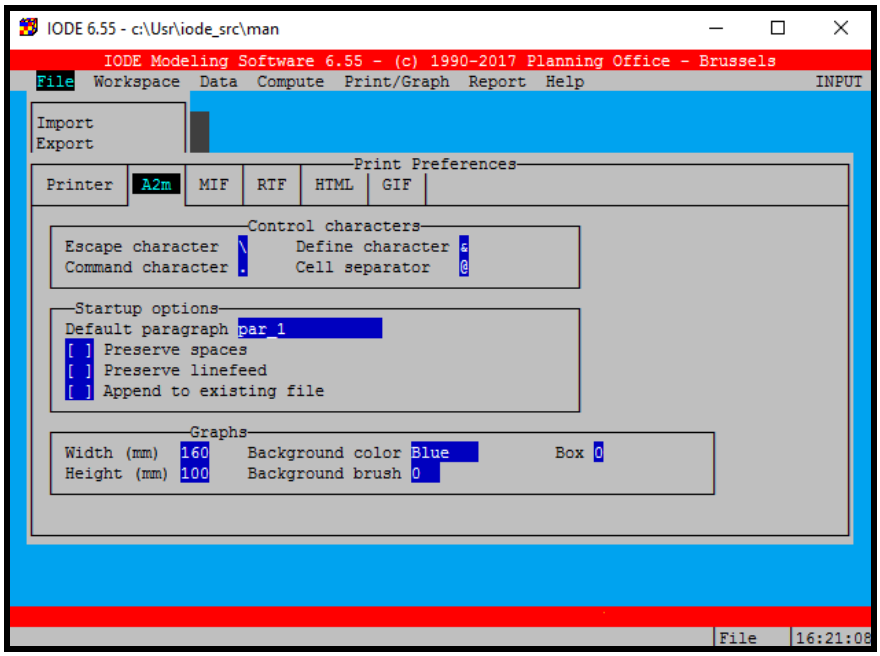
Options impression directe (Printer)

FIGURE 8



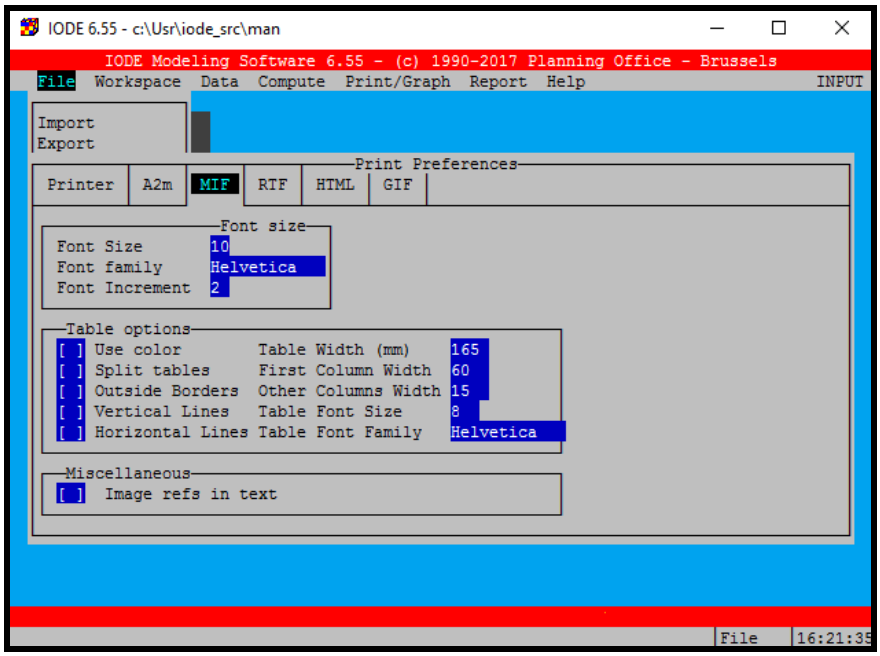
Options génération fichier A2m

FIGURE 9



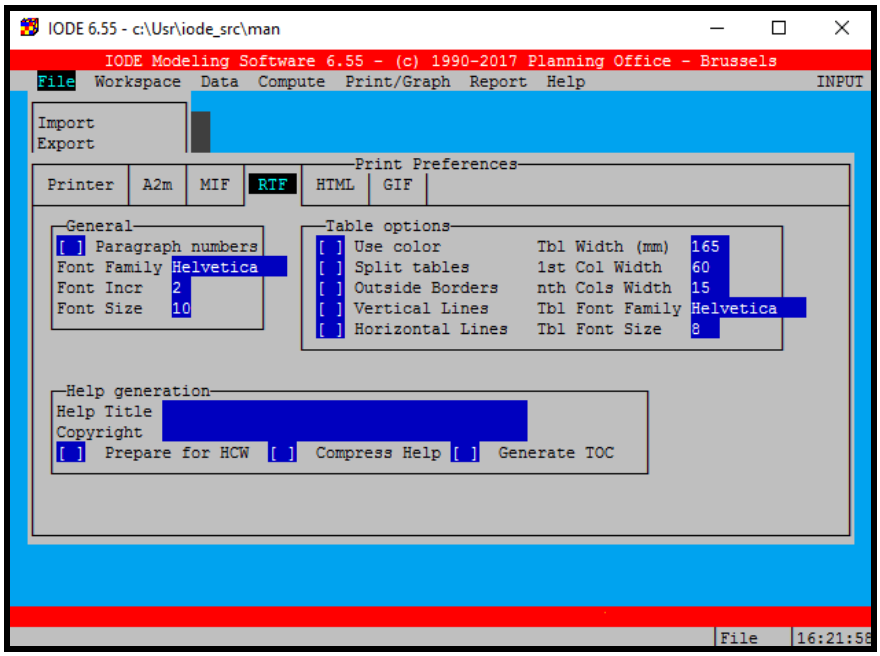
Options génération fichier MIF

FIGURE 10



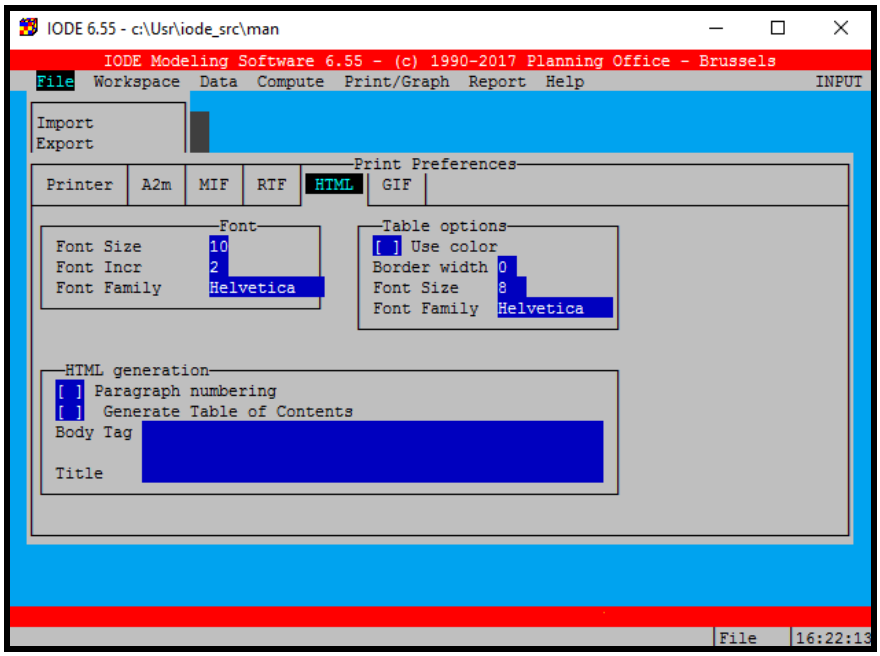
Options génération fichier RTF

FIGURE 11



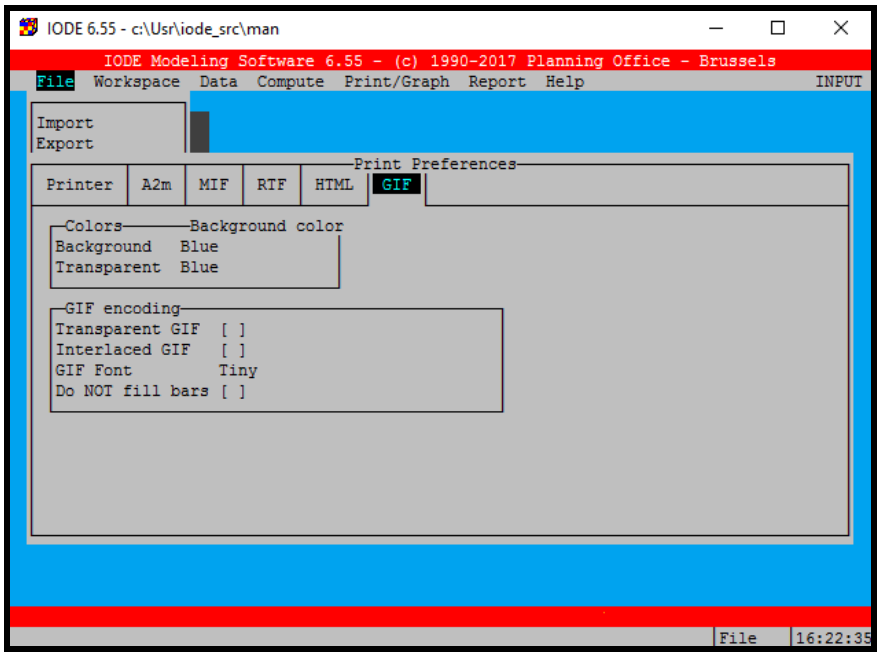
Options génération fichier HTML

FIGURE 12



Options génération fichier GIF

FIGURE 13

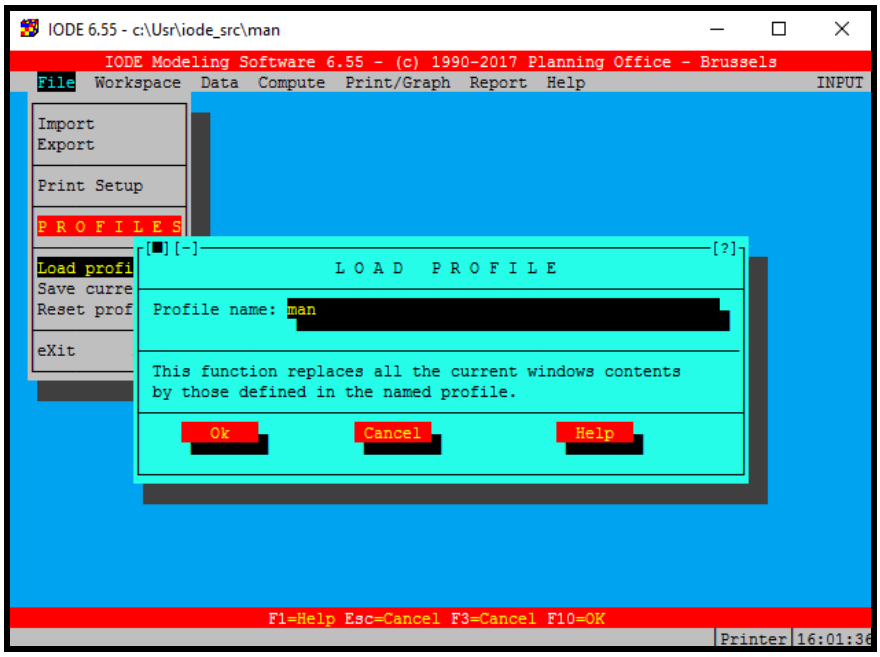


2.1.4 LOAD PROFILE

Ce point du menu charge un fichier "profile" à partir du disque et réinitialise les pages de l'applica-

tion avec les valeurs de ce profil.

FIGURE 14



Le fichier profile est une image du contenu des champs des pages de saisie au moment où l'on quitte IODE ou au moment où l'on sauve le profile (voir Save Current Profile). L'utilisateur peut donc sauver la configuration et le contenu courant des pages de façon à les retrouver ultérieurement au cours d'une autre session de travail. Il peut aussi changer de configuration en cours de session.

Le profile courant est sauvé automatiquement dans le fichier IODE.PRF lorsque l'on quitte IODE et chargé automatiquement lorsque le programme est lancé.

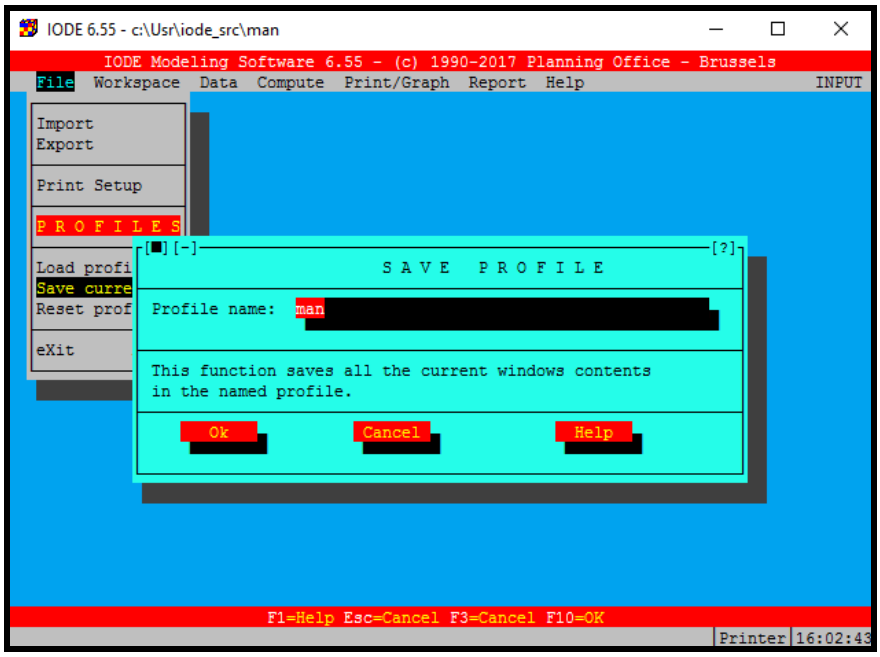
La page de saisie permet de sélectionner le profile à charger. Le champ de saisie du nom du fichier à charger est du type DIR. La touche F10 valide l'opération lorsque la page est complétée.

Le profile chargé remplace le profile courant.

2.1.5 SAVE CURRENT PROFILE

Ce point du menu permet de sauver un fichier profile sur disque.

FIGURE 15



Voir Load Profile pour la description des "profils".

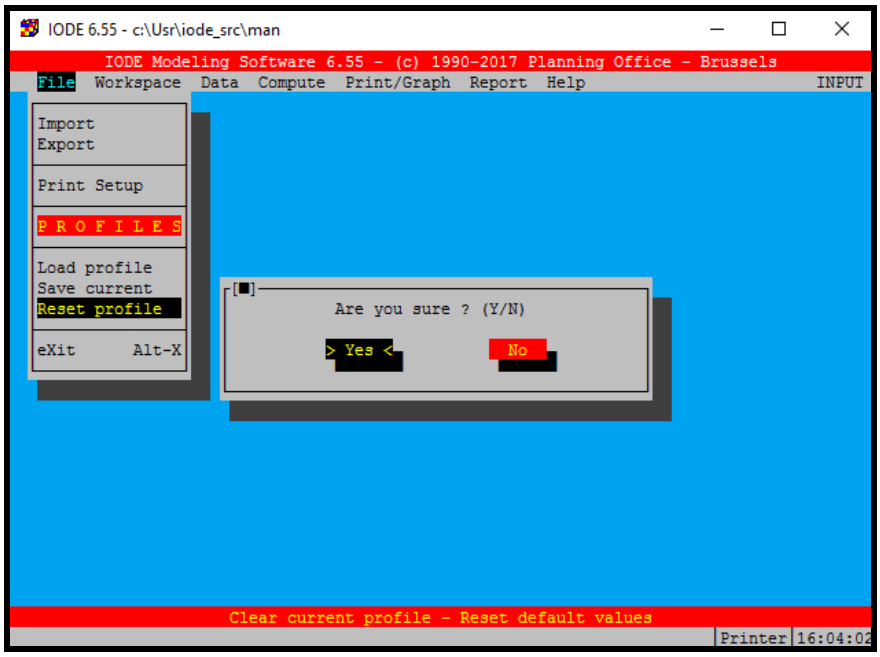
Le profil courant est sauvé automatiquement dans le fichier **IODE.PRF** lorsque l'on quitte IODE et chargé automatiquement lorsque le programme est lancé. Le profil est sauvegardé dans le directory courant.

La page de saisie permet de donner un nom au profile à sauver. Le champ de saisie du nom du fichier à charger est du type DIR. La touche F10 valide l'opération lorsque la page est complétée.

2.1.6 RESET PROFILE

Ce point du menu permet de vider tous les champs de saisie des pages ou de les remettre à leur valeur par défaut.

FIGURE 16

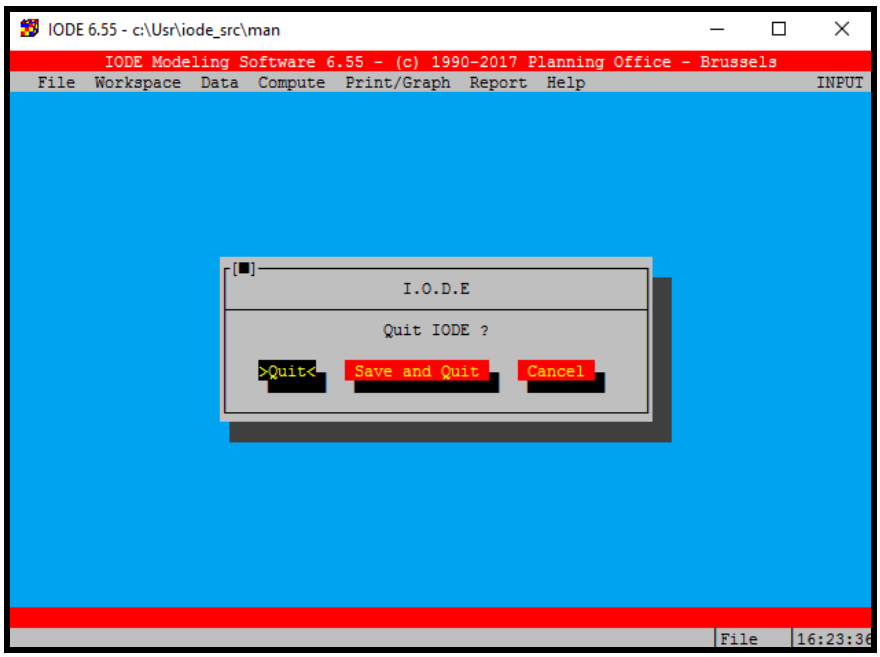


Pour éviter une opération involontaire, cette opération - qui ne demande pas de paramètre - doit être confirmée.

2.1.7 EXIT : QUITTER LE PROGRAMME

Cette option permet de quitter le programme IODE. Confirmation est d'abord demandée pour éviter de quitter brutalement l'application.

FIGURE 17

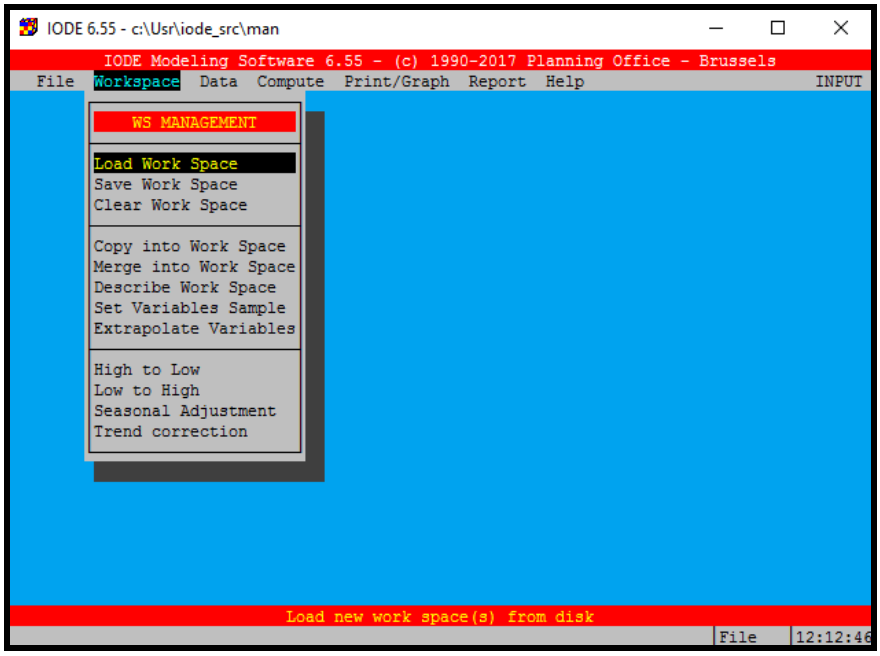


Elle permet de faire un ultime sauvetage des WS de données avant de quitter définitivement IODE.

Cette fonction peut être déclenchée n'importe où dans le logiciel en pressant la touche Alt-X.

2.2 WORKSPACE

FIGURE 18



La notion de workspace (WS) occupe une place centrale dans le déroulement d'une session IODE. Rappelons que le logiciel opère sur les objets contenus dans les WS courants, c'est-à-dire chargés en mémoire.

Il y a un WS courant par type d'objet. Le menu WORKSPACE offre toutes les fonctions permettant de charger à partir du disque et de sauver sur disque les fichiers associés aux workspaces. Il permet de traiter les WS dans leur ensemble. Le menu DATA permettra quant à lui de travailler au niveau des objets individuels contenus dans les workspaces.

Un fichier de workspace sur disque porte un nom dont l'extension indique le type d'objet qu'il contient.

On distingue sept types d'objets dans IODE et donc sept extensions différentes aux noms de fichiers associés :

- les commentaires : .cmt
- les équations : .eqs
- les identités : .idt
- les listes : .lst
- les scalaires : .scl
- les tableaux : .tbl
- les variables : .var

En plus de ces extensions qui correspondent aux fichiers d'objets en format interne, on a des extensions associées aux fichiers en format Ascii :

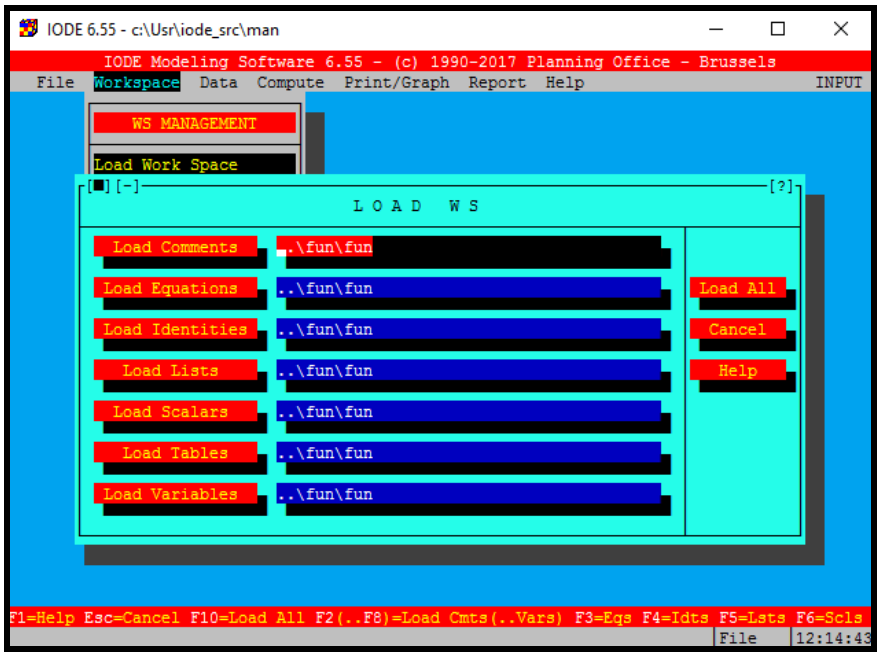
- les commentaires : .ac
- les équations : .ae
- les identités : .ai
- les listes : .al
- les scalaires : .as
- les tableaux : .at
- les variables : .av

Les différentes options du menu WORKSPACE sont :

- Load Work Space : charge des WS depuis le disque
- Save Work Space : sauve les WS courants sur disque
- Clear Work Space : vide les WS courants
- Copy into Work Space : copie les données sélectionnées des WS sur disque vers les WS courants
- Merge into Work Space : fusionne les données des WS sur disque avec les WS courants
- Describe Work Space : change les descriptions des WS courants
- Set Variables Sample : change la période de définition des variables
- Extrapolate Variables : prolonge les séries par extrapolation
- High to Low : modifie la périodicité de séries
- Low to High : modifie la périodicité de séries
- Seasonal Adjustment : Census XI - method for Seasonal Adjustment
- Trend Correction : Filtre de Hodrick-Prescott

2.2.1 LOAD WORKSPACE

FIGURE 19



L'opération de chargement (LOAD) d'un workspace remplace le contenu du WS courant par le contenu d'un fichier disque : tous les objets actuellement présents en WS sont détruits et remplacés par ceux définis dans le fichier à charger. Dans le cas des variables, le SAMPLE du fichier chargé remplace le SAMPLE courant.

Cette fonction ouvre un écran dans lequel on peut entrer un nom de fichier par type d'objet. Au départ, les noms sont ceux des WS courants. Tous les champs sont des champs DIR : la touche ENTER permet donc de sélectionner le fichier désiré au travers d'un menu reprenant tous les fichiers correspondants au masque introduit dans le champ.

Les touches F2 à F8 permettent de charger un seul WS, selon la touche pressée : F2 pour comments, F3 pour equations, etc. La touche F10 permet de charger tous les WS dont les noms sont spécifiés.

Des boutons correspondant à ces touches sont également proposés.

La fonction n'essayera de charger que les fichiers de WS dont le nom n'est pas vide. Il suffit donc de laisser blanc les noms correspondant aux types de WS qu'on ne veut pas modifier et de presser F10 pour charger un nombre limité de WS.

Si un fichier n'est pas trouvé, la fonction ne tente pas de charger les WS suivants. Le contenu du WS correspondant au type de fichier n'est pas détruit.

EXTENSIONS DES NOMS DE FICHIERS

Le programme de chargement de WS est capable non seulement de charger des fichiers en format interne, mais également des fichiers en format Ascii.

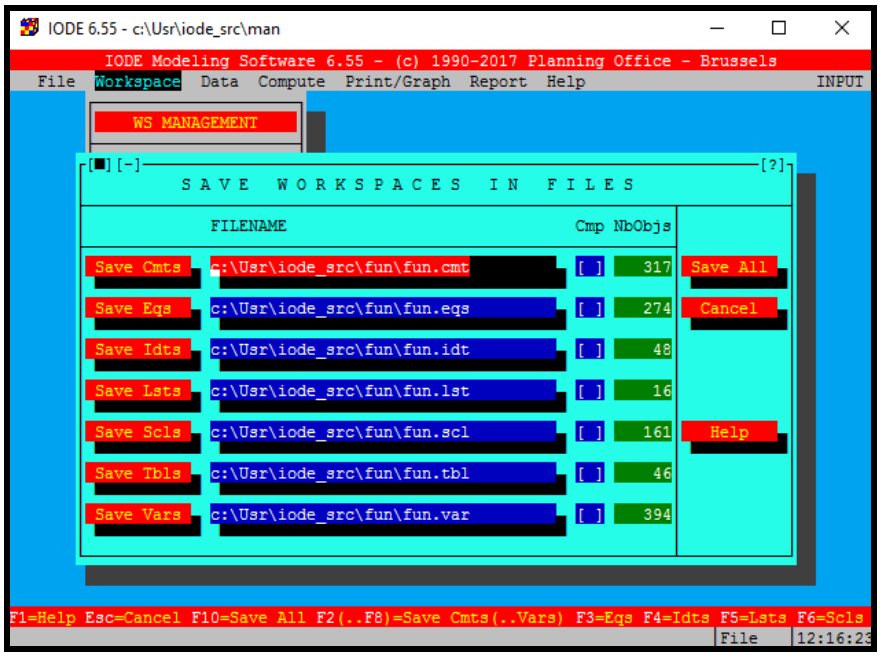
Il se base pour faire le choix entre les deux formats sur l'extension du fichier. Si cette extension est une des extensions standardisées (`myws.cmt`) ou s'il n'y a pas d'extension au nom du fichier (`myws`),

le chargement d'un fichier en format interne est effectué. Si l'extension est autre (**myws.aaa** par exemple), la fonction de chargement d'un fichier en format Ascii est utilisée.

2.2.2 SAVE WORKSPACE

Cette fonction réalise l'opération inverse de la fonction LOAD : elle copie sur disque les WS dans leur état actuel, en remplaçant éventuellement les fichiers existants. Accessoirement, elle permet de connaître le contenu actuel des différents WS.

FIGURE 20



La fenêtre présente pour chaque type de WS :

- le nom du fichier associé au WS. Par défaut, ce nom est ws.type où type indique le type d'objet. Ce nom peut évidemment être modifié. Si le WS a été chargé à partir du disque, le nom du WS est le nom du fichier d'origine.
- le nombre d'objets actuellement définis dans le WS

Un nom vide ne donnera pas lieu à un sauvetage. Les WS courants (en mémoire) ne sont pas modifiés par cette fonction.

Les touches F2 à F8 permettent de sauver un seul WS, selon la touche pressée : F2 pour comments, F3 pour equations, etc. La touche F10 permet de sauver tous les WS dont les noms sont spécifiés.

Des boutons correspondant à ces touches sont également proposés.

La fonction n'essayera de sauver que les WS dont le nom n'est pas vide. Il suffit donc de laisser blanc les noms correspondant aux types de WS qu'on ne veut pas sauver et de presser F10 pour sauver un nombre limité de WS.

La fonction s'arrête dès qu'un WS ne peut être sauvé (nom incorrect, manque de place sur disque, problème de mémoire, etc). Les WS suivants ne sont donc pas sauvés dans ce cas.

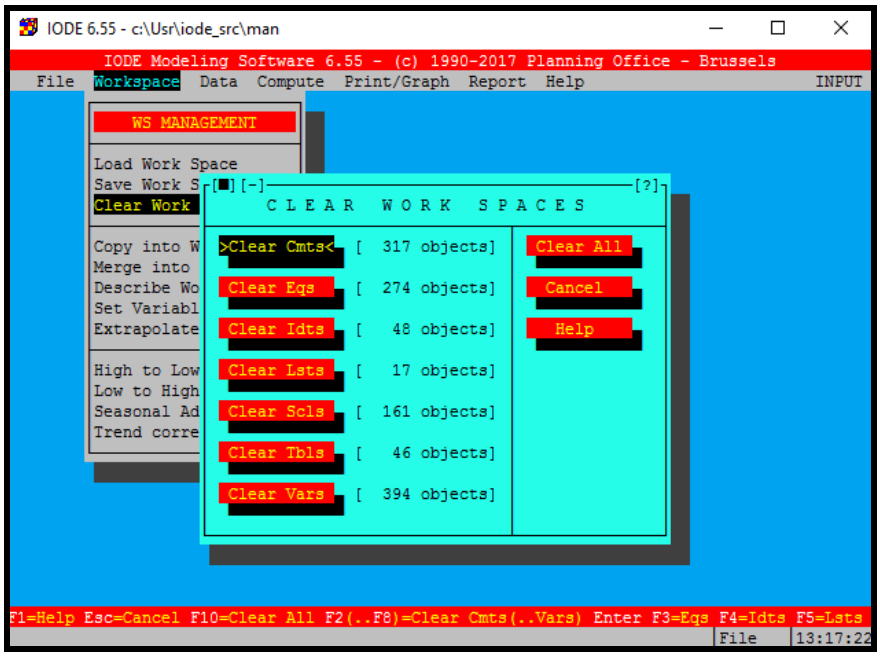
BACKUP

Avant de sauver un WS, une copie de l'ancienne version du fichier (s'il en existe une) est prise par sécurité. Ce fichier porte le même nom que l'ancien, avec la même extension sauf la dernière lettre qui est remplacée par un $. Ainsi le fichier `mytest.cmt` est copié sous le nom `mytest.cm$`

Pour restaurer un fichier, il suffit de copier le backup sous un nom accepté par IODE.

2.2.3 CLEAR WORKSPACE

FIGURE 21



Lorsqu'on souhaite détruire tous les objets contenus dans un WS courant (en mémoire), au moins deux méthodes globales sont disponibles : charger un nouveau fichier de WS ou vider à l'aide de la fonction CLEAR le contenu du WS.

L'écran présente pour chaque type de WS le nombre d'objets actuellement définis en mémoire.

Les touches F2 à F8 permettent de vider un seul WS, selon la touche pressée : F2 pour comments, F3 pour equations, etc. La touche F10 permet de vider tous les WS.

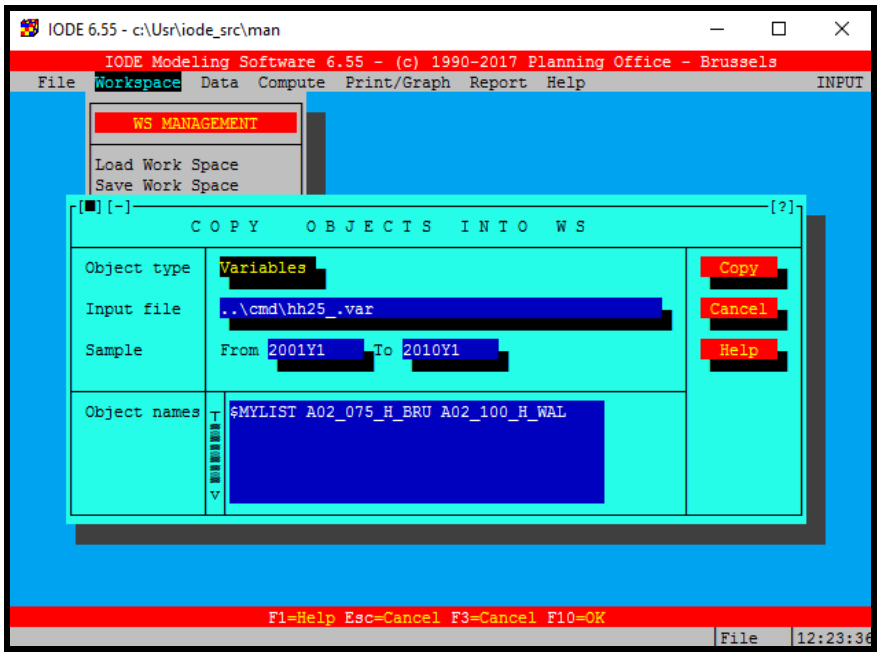
Des boutons correspondant à ces touches sont également proposés.

Cas particulier : les variables

Lorsqu'un WS de variables est "vidé", la période de définition des séries n'est plus définie pour permettre de changer la périodicité du WS.

2.2.4 COPY INTO WORKSPACE

FIGURE 22



Il est possible d'ajouter ou de remplacer des objets en mémoire à partir de WS sauvés sur disque. La fonction COPY demande le type d'objet à copier, le nom du fichier contenant le WS où se trouvent les objets à copier et enfin la liste des noms d'objets.

Dans le cas des variables, la période à copier peut également être spécifiée.

Une fois ces informations introduites, les objets indiqués sont copiés dans le WS courant et remplacent les valeurs actuelles. Le processus s'arrête dès qu'un objet ne peut être trouvé.

Le WS courant garde intacts tous les autres objets.

Un cas particulier : les variables

Dans le cas des variables, un problème supplémentaire se pose : le SAMPLE de définition du WS courant peut être différent de celui du fichier. Dans ce cas, les variables copiées s'adaptent au SAMPLE du WS courant, soit en perdant des valeurs, soit en ajoutant des valeurs NA pour compléter les séries.

Si une période est spécifiée dans l'écran de saisie, seules les observations de cette période sont copiées en WS.

Utilisation de listes

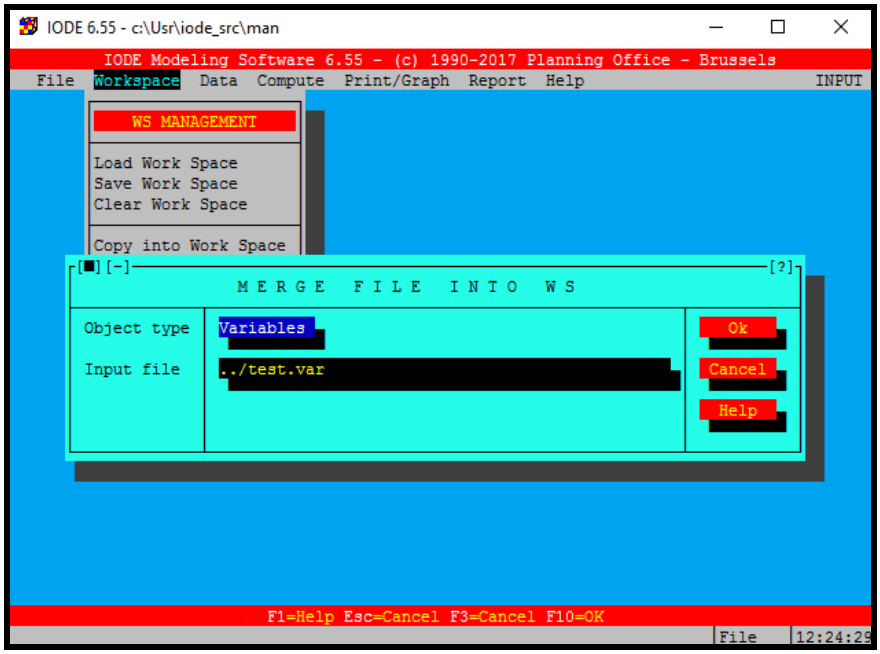
Un nom de liste précédé d'un dollar peut être introduit dans la zone correspondant aux noms d'objets :

```
$LISTE
```

La liste est alors remplacée par sa valeur et tous les noms définis dans la liste sont copiés. Une liste peut elle-même être définie par d'autres listes : celles-ci sont récursivement remplacées par leur valeur.

2.2.5 MERGE INTO WORKSPACE

FIGURE 23



Cette fonction permet d'ajouter toutes les données d'un workspace sur fichier au workspace actif (en mémoire). Dans le cas où un objet existe déjà dans le workspace actif, il est remplacé par la valeur du fichier.

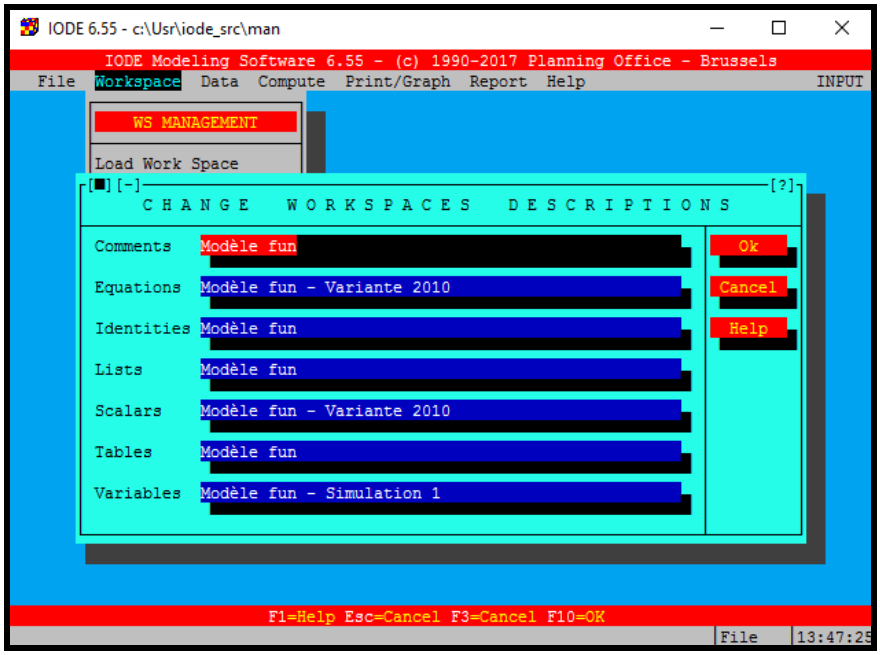
Dans le cas de variables, la période sur laquelle les séries sont copiées est celle définie par le SAMPLE du workspace actif. Si le workspace actif de variables n'a pas encore de SAMPLE, le SAMPLE du fichier est utilisé. Dans ce dernier cas, cette fonction est identique à un Load.

2.2.6 DESCRIBE WORKSPACE

Cette fonction permet de modifier la description associée à chaque workspace présent en mémoire.

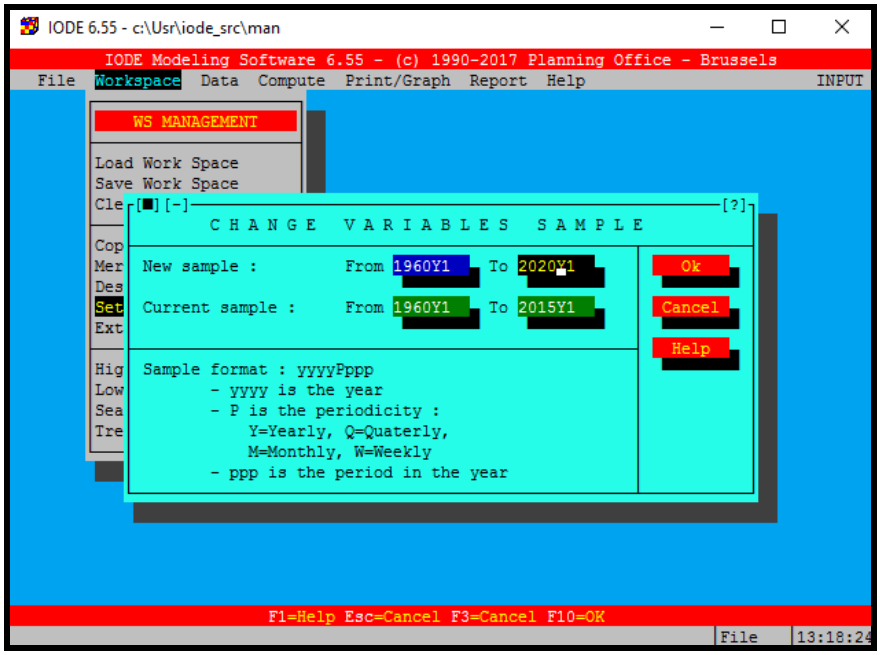
A ce niveau, l'effet n'est que local à la session : la modification de la description d'un WS n'a aucune influence sur le contenu des fichiers de WS sur disque. Par contre, au prochain sauvetage des workspaces (Save), les descriptions seront celles introduites lors du dernier appel à la fonction Describe.

FIGURE 24



2.2.7 SET VARIABLES SAMPLE

FIGURE 25



Le WS de variables est défini sur un "SAMPLE", indiquant les périodes extrêmes sur lesquelles les séries peuvent être définies. Toutes les séries d'un workspace sont définies sur la même période.

Il peut se produire que ce sample doive être modifié, par exemple pour ajouter des observations ou encore pour limiter l'espace requis par le stockage du WS.

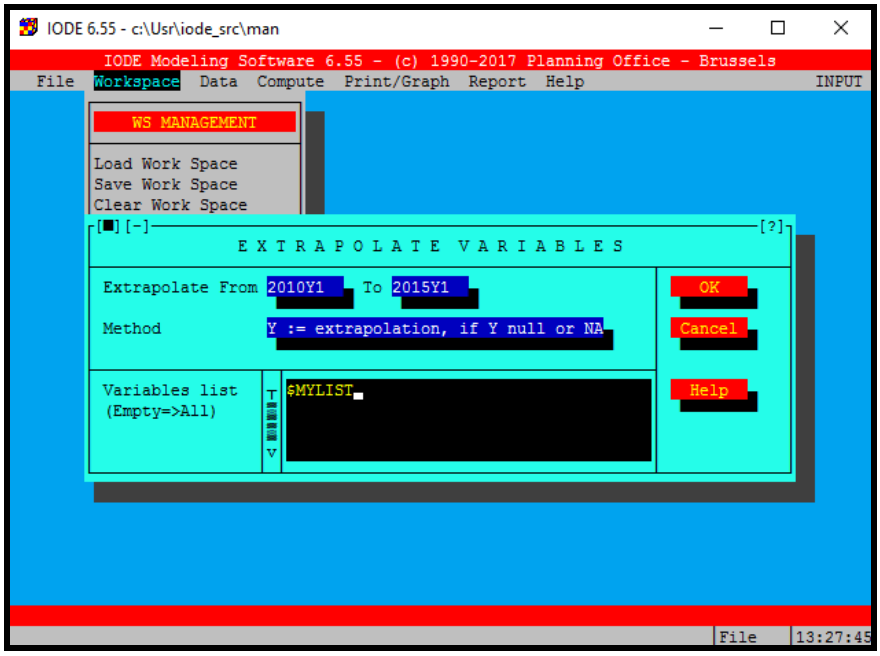
Les périodes ajoutées prennent comme valeur NA (not available). Les valeurs correspondantes aux

périodes supprimées sont détruites.

La périodicité ne peut être modifiée, sauf si le sample initial était non défini (nul). Pour changer de périodicité, il faut utiliser les fonctions High to Low ou Low To High.

2.2.8 EXTRAPOLATE VARIABLES

FIGURE 26



Cette fonction modifie les valeurs de séries sur une période choisie par extrapolation.

Les premiers champs permettent de spécifier la période à modifier.

Le deuxième champ offre le choix entre différentes méthodes (identiques à celles disponibles pour les valeurs de départ des simulations) :

- Y := Y[-1], if Y null or NA : chaque valeur nulle ou NA prend la valeur de la période précédente
- Y := Y[-1], always : chaque valeur prend la valeur de la période précédente
- Y := extrapolation, if Y null or NA : chaque valeur nulle ou NA prend comme valeur une extrapolation linéaire des deux périodes précédentes
- Y := extrapolation, always : chaque valeur prend comme valeur une extrapolation linéaire des deux périodes précédentes, qu'elle soit nulle ou non au départ
- Y unchanged : les valeurs ne sont pas modifiées.
- Y := Y[-1], if Y = NA : chaque valeur NA prend la valeur de la période précédente
- Y := extrapolation, if Y = NA : chaque valeur NA prend comme valeur une extrapolation linéaire des deux périodes précédentes

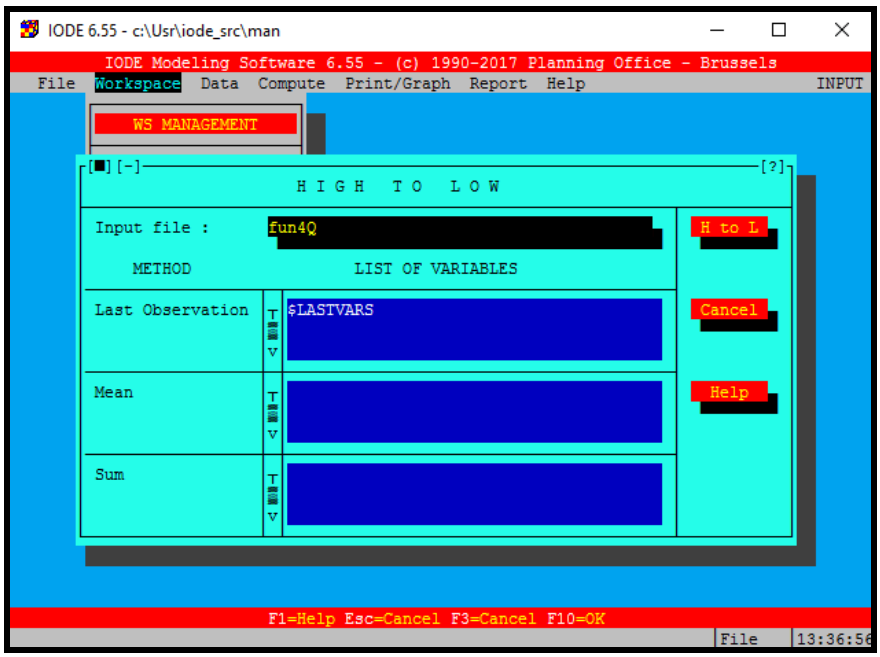
Le dernier champ contient la liste des noms de séries à extrapoler.

La fonction de rapport équivalente est :

```
$WsExtrapolate [method] from to [liste de variables]
```

2.2.9 HIGH TO LOW

FIGURE 27



Construit des séries de périodicité inférieure à partir de séries stockées dans un fichier.

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément la périodicité de ces séries. La nouvelle périodicité est celle actuellement définie dans le WS actif.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

Cette procédure existe pour les cas suivants :

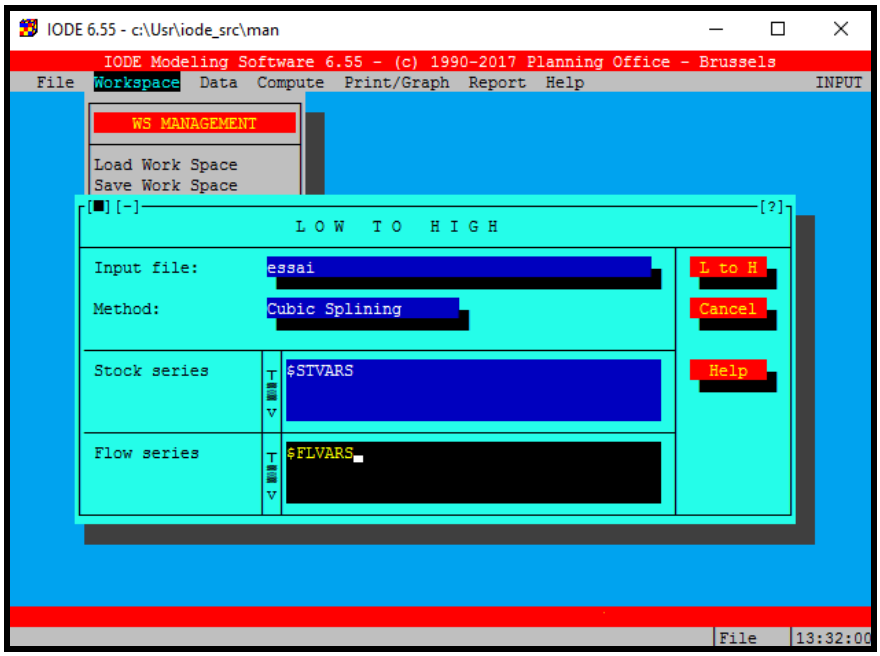
- mensuel vers annuel (observation annuelle = somme des 12 mois)
- trimestriel vers annuel (observation annuelle = somme des 4 trimestres)
- mensuel vers trimestriel (observation trimestrielle = somme des 3 mois)

Après avoir spécifié le nom du fichier input, trois listes doivent être fournies. La première liste est celles des séries pour lesquelles le résultat est la somme des sous-périodes, la deuxième la moyenne et la troisième, la dernière observation de la série input.

En cas de valeur inexistante (NA) pour l'une des sous-périodes, le résultat est NA.

2.2.10 LOW TO HIGH

FIGURE 28



Construit des séries de périodicité supérieure.

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément la périodicité de ces séries. La nouvelle périodicité est celle actuellement définie dans le WS actif.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

Cette procédure existe pour les cas suivants :

- annuel vers mensuel
- annuel vers trimestriel
- trimestriel vers mensuel

Deux méthodes sont disponibles, l'une pour les stock, l'autre pour les flux : dans le cas des stocks, toutes les valeurs assignées pour les sous-périodes sont celles de la période input.

A[1980Q{1,2,3,4}] = A[1980Y1]

Dans le cas des flux, la série est répartie sur les sous-périodes :

A[1980Q1] = 0.25 * A[1980Y1]

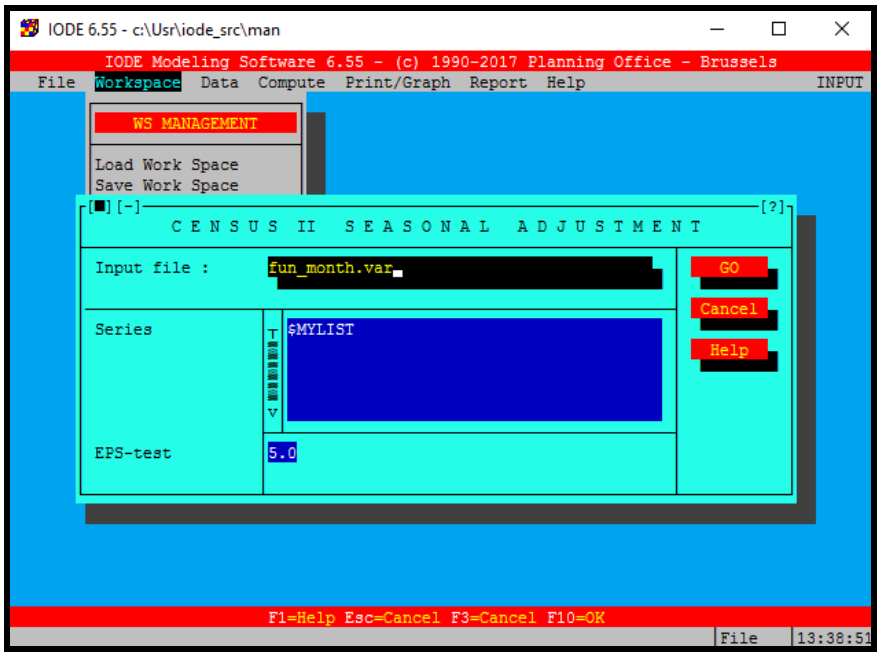
Après avoir spécifié le nom du fichier input, deux listes doivent être fournies. La première liste est celles des séries de type "stock", la deuxième de type "flux".

Enfin, il est possible de spécifier la méthode d'interpolation à utiliser pour calculer les valeurs intermédiaires :

- interpolation linéaire
- interpolation par "cubic splines"
- interpolation par escalier

2.2.11 SEASONAL ADJUSTMENT

FIGURE 29



Elimine les variations saisonnières des séries mensuelles. La méthode utilisée est celle de Census X11. La méthode ne fonctionne que pour les parties qui sont entièrement connues.

En plus de la série dessaisonnalisée (qui porte le nom de la série dans le fichier input), on obtient comme résultat deux autres séries :

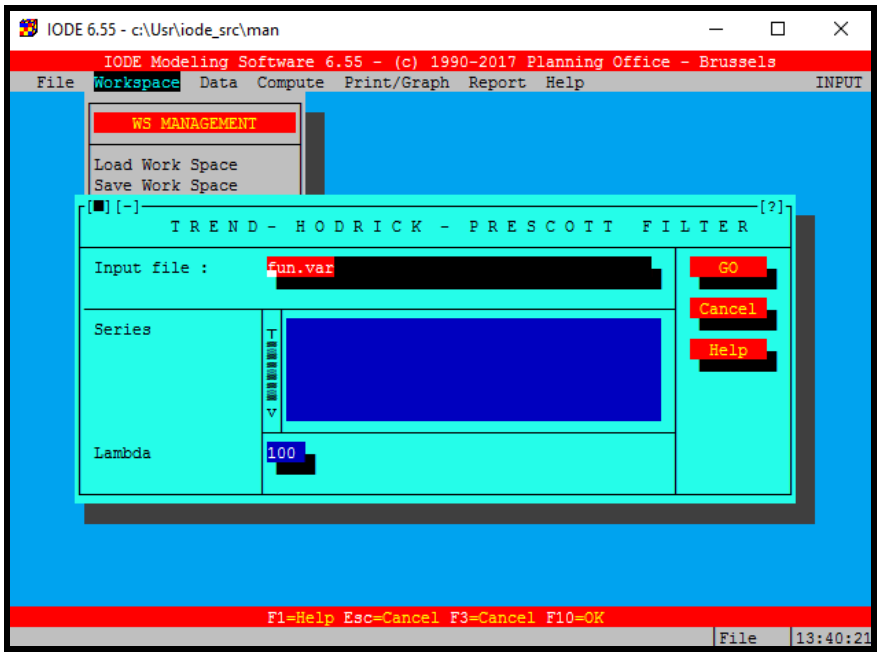
- _Cseries (où series est le nom de la série d'origine) qui contient la série de trends cycliques
- _Iseries qui contient la composante stochastique

Une fonction de rapport effectue la même opération :

```
$WsSeasonAdj
```

2.2.12 TREND CORRECTION

FIGURE 30



Implémentation de la méthode Hodrick-Prescott pour la construction de séries de trend. Le principe est le même que pour la dessaisonnalisation: les séries lues dans un fichier sont importées et transformées simultanément.

Une note technique [Bart Hertveld] décrit la méthode et son utilisation. Voir aussi Hodrick-Prescott Filter.

Les fonctions équivalentes de rapport ont la syntaxe suivante :

```
$WsTrend VarFilename Lambda series1 series2 ...
#WsTrend (interactive)
```

2.2.12.1 HODRICK-PRESCOTT FILTER

ABSTRACT

Technische beschrijving van de procedure voor het bepalen van een niet-lineaire trend in tijdreeksen.

1. INLEIDING

Met ingang van de versie 4.52 kan de IODE-gebruiker een beroep doen op de Hodrick-Prescott filter (in het vervolg: HP-filter). Deze filter ontbindt een seizoengezuiverde tijdreeks in een trendcomponent en de complementaire cyclische component. Voor conjunctuuranalyse is een dergelijke decompositie van de trendcyclische component onontbeerlijk. Enerzijds werd de incorporatie van deze tijdreekstechniek in IODE dus ingegeven door de toenemende belangstelling binnen het Planbureau

voor het domein van de korte termijn studies (zie o.a. het opstellen van het Economisch Budget). Anderzijds biedt de analyse van de trendbewegingen ruime toepassingsmogelijkheden voor studies die zich meer op lange termijn richten.

De HP-filter wordt tevens aangewend door het DG II van de Europese Commissie voor de berekening van de cyclische aanpassing van begrotingstekorten.

2. Opdeling van tijdreeksen

Klassiek in de tijdreeksanalyse is de opdeling van een basisreeks yt in vier componenten:

- de trendcomponent (gt);
- de cyclische component (ct);
- de seizoenscomponent (st);
- de volstrekt onvoorspelbare component of toevalscomponent (ut).

Per definitie is de seizoenscomponent onbestaande voor reeksen met een jaarfrequentie (stroomgrootheden over een periode van 12 maanden, voorraadgrootheden op een identiek tijdstip per jaar, groeivoet tegenover zelfde maand van het vorige jaar, groeivoet laatste twaalf maand tegenover vorige twaalf maand, cumulatieve groeivoet tegenover zelfde periode vorige jaar,...).

Deze decompositie kan additief of multiplicatief geformuleerd worden

$$y_t = g_t + c_t + s_t + u_t$$

$$y_t = g_t \cdot c_t \cdot s_t \cdot u_t$$

Deze opdeling van tijdreeksen kan via IODE als volgt bereikt worden.

- Stap 1: via de Census X-II methode (beschikbaar in IODE sedert de versie 4.38) wordt de basisreeks omgezet in een seizoengezuiverde reeks. Het verschil (additief of multiplicatief gedefinieerd) tussen de basisreeks en de seizoengezuiverde reeks is de seizoenscomponent st.
- Stap 2: toepassing van de HP-filter op de seizoengezuiverde reeks levert de trendwaarde gt.
- Stap 3: het residu (additief of multiplicatief gedefinieerd) uit stap 2 bevat in principe nog de cyclische en de toevalscomponent. Eventueel kan de toevalscomponent worden afgesplitst door afvlakking van de residureeks (b.v. via voortschrijdende gemiddelden).

3. De Hodrick-Prescott filter

FORMULERING

De Hodrick-Prescott trend gt van een seizoengezuiverde reeks yt wordt gedefinieerd als de oplossing van onderstaand minimaliseringsprobleem:

$$Min_{g_t}\left(\sum_{t=1}^{T}(y_t-g_t)^2\right) \quad (1)$$

o.v $\sum_{t=2}^{T-1}[(g_{t+1}-g_t)-(g_t-g_{t-1})]^2 \le \varepsilon$ met ε een arbitrair gekozen klein getal

- T:het aantal gegevens in de tijdreeks;
- yt:de logaritme van de basisreeks, dus gegeven;
- gt:de logaritme van de trendwaarde, de onbekende waarnaar de uitdrukking geminimaliseerd wordt.

Door *eps* te bepalen leggen we een voorwaarde op aan de groei van gt. Kiezen we *eps* = 0, dan kiezen we voor een constante groei van gt en komt de minimalisering neer op de schatting van een log-lineaire trend.

De oplossing van het probleem is afhankelijk van de waarde van *eps.* Voor gelijk welke e vindt men een oplossing als men de volgende vorm minimaliseert.

$$\sum_{t=1}^{T}(y_t-g_t)^2+\lambda\sum_{t=2}^{T-1}[(g_{t+1}-g_t)-(g_t-g_{t-1})]^2 \quad (2)$$

- *lambda* : multiplicator van Lagrange, in deze procedure vrij te kiezen constante positieve waarde.

Hoe groter *lambda* hoe gladder de resulterende gt. Laten we *lambda* naar oneindig gaan, dan krijgen we een lineaire trend.

UITWERKING

Berekenen we Min (gt, (2)) dan moeten we (2) T-keer partieel afleiden naar gt. Zo bekomen we een stelsel van T-vergelijkingen en T onbekenden.

$$\frac{\partial}{\partial g_t}\left(\sum_{t=1}^{T}(y_t-g_t)^2+\lambda\sum_{t=2}^{T-1}[(g_{t+1}-g_t)-(g_t-g_{t-1})]^2\right) = 0 \text{ met } t = 1, \ldots\ldots T \quad (3)$$

$$\text{Of} \quad \begin{cases} -y_1 + g_1 + \lambda(g_1 - 2g_2 + g_3) = 0 \\ -y_2 + g_2 + \lambda(-2g_1 + 5g_2 - 4g_3 + g_4) = 0 \\ -y_t + g_t + \lambda(g_{t-2} - 4g_{t-1} + 6g_t - 4g_{t+1} + g_{t+2}) = 0 \text{ met t = 3, ...,T-3} \\ -y_{T-1} + g_{T-1} + \lambda(g_{T-3} - 4g_{T-2} + 5g_{T-1} - 2g_T) = 0 \\ -y_T + g_T + \lambda(g_{T-2} - 2g_{T-1} + g_T) = 0 \end{cases}$$

$$\text{Of} \quad y_t = \begin{bmatrix} (1+\lambda) & -2\lambda & \lambda & 0 & \dots & \dots & 0 \\ -2\lambda & (1+5\lambda) & -4\lambda & \lambda & & & \\ \dots & & & & & & \dots \\ 0\dots & \lambda & -4\lambda & (1+6\lambda) & -4\lambda & \lambda & \dots 0 \\ \dots & & & & & & \dots \\ 0 & \dots & 0 & \lambda & -4\lambda & (1+5\lambda) & -2\lambda \\ 0 & \dots & \dots & 0 & \lambda & -2\lambda & (1+\lambda) \end{bmatrix} g_t$$

De matrix is een symmetrische T x T matrix en heeft als elementen de gewichten die toelaten de originele basisreeks terug te rekenen uit de trendreeks (elke rij van de matrix sommeert tot 1, wat de term gewichten wettigt). De trendreeks wordt uiteindelijk berekend door de inverse van de matrix te vermenigvuldigen met de vector van de basisreeks.

FILOSOFIE

Bij de raming van een trendbeweging stelt zich steeds een trade-off tussen de twee volgende eisen:

- de trend mag niet te ver afwijken van de originele reeks;
- de trend moet een vloeiend (”verloop hebben, m.a.w, de trendmatige groeivoet moet over de tijd heen vrij stabiel1 zijn.

De eerste eis wordt geformaliseerd in de eerste term van [2]: minimalisering van de som van de gekwadrateerde verschillen in logaritme tussen de originele waarden en de trendwaarden. Dit betekent concreet dat de verhouding2 tussen de originele waarde en de trendwaarde op geen enkel tijdstip ver van de eenheid mag afwijken.

De tweede eis wordt geformaliseerd in de tweede term van [2]: minimalisering van het verschil in opeenvolgende groeivoeten3 van de trendreeks. Dit betekent concreet dat de trendmatige groeivoet geen bokkesprongen mag vertonen en dat de trend dus een vloeiend verloop kent.

Dat tussen beide eisen een trade-off bestaat is duidelijk: hoe beter de originele reeks benaderd wordt (à la limite: trendwaarde = originele waarde), hoe minder vloeiend de trendreeks, en, hoe stabieler de trendgroei (à la limite: constante groeivoet), hoe groter de afwijking met de originele reeks.

Het belang dat de gebruiker in de hierboven geschetste trade-off wil toekennen aan de eerste, resp. de tweede eis, kan worden vertaald in de waarde van de parameter l. Hoe groter (kleiner) men l kiest, hoe sterker (minder sterk) de tweede eis, relatief tegenover de eerste, zal meespelen in de bepaling van de trendwaarden. De limietwaarden voor l maken een en ander duidelijk. Voor vervalt de eerste eis en bekomt men een lineaire trend (een constante groeivoet is immers de limietwaarde van een stabiele groeivoet). Voor l = 0 is de cyclische component gelijk aan nul (immers, de trendwaarde valt in dat geval samen met de originele waarde). In de literatuur wordt l = 1600 voor kwartaalreeksen, en l = 100 voor jaarreeksen aanbevolen. Deze waarden voor *lambda* werden standaard voorzien in IODE.1 Wegens het arbitraire karakter van deze keuze is het aangewezen geval per geval na te gaan in hoeverre de keuze van *lambda* de decompositie van de trendcyclische component beïnvloedt.

KANTTEKENINGEN

Keuze van lambda

Een eerste kanttekening bij de HP-filter betreft de parameter *lambda.* De keuze van *lambda* is arbitrair en kan dus de ontbinding van de trendcyclische component beïnvloeden. Meer bepaald kunnen, binnen het raam van de conjunctuuranalyse, alternatieve waarden voor *lambda* aanleiding geven tot tegenstrijdige bevindingen omtrent de datering van de cycli.

Deze kritiek op de HP-filter kan echter genuanceerd worden:

- "de" methode voor decompositie van de trendcyclische component bestaat niet; de onderzoeker wordt dus überhaupt geconfronteerd met een keuzeprobleem (nl. de keuze van de gehanteerde methode);
- de kritiek op de methode kan deels worden genuanceerd door de stabiliteit van de besluiten te testen voor verschillende (normaal geachte) waarden van *lambda;* dit pleit dus tegen een blind gebruik van deze techniek;
- de vrije keuze van *lambda* kan worden aangewend als instrument, met name bij het relateren van kwantitatieve aan kwalitatieve indicatoren: hierbij kan via de keuze van *lambda* op zoek worden gegaan naar de systematische verdeling van de totale trendcyclische beweging over trend en cyclus (onder de veronderstelling dat kwalitatieve indicatoren uitsluitend deze laatste capteren).

Stabiliteit aan de uiteinden van de reeks

Uit de constructie van de HP-filter (cf. supra) blijkt dat de begin- en eindwaarden van de trendreeks gevoelig zijn voor bijkomende observaties in de basisreeks. Voor historische studies kan dit probleem grotendeels ondervangen worden door de basisreeks zo lang mogelijk te kiezen en vervolgens aan beide uiteinden van de berekende trendreeks een aantal gegevens weg te laten. Voor conjunctuuronderzoek echter zijn precies de meest recente gegevens van groot belang. In dit geval is het aangewezen de basisreeks naar de toekomst toe te extrapoleren op basis van de meest waarschijnlijk geachte evolutie. Op die manier kan men herzieningen van de trendcyclische opsplitsing als gevolg van bijkomende waarnemingen in de basisreeks beperken in omvang.

VOORBEELD

Het gebruik van de HP-filter wordt hieronder geïllustreerd aan de hand van de reeks voor het Belgische BBP in constante prijzen (ESER-definitie). Voor de basisreeks beschikken we over observaties voor de periode 1953-1992. De betrokken reeks bevat uiteraard geen seizoenscomponent en de toevalscomponent wordt verondersteld gelijk aan nul te zijn.

Onderstaande figuren tonen, voor verschillende waarden van l, telkens drie trendreeksen voor de groeivoet van het BBP:

- BBP92: trend berekend via toepassing van de HP-filter op de basisreeks van de waarnemingen (1953-92);
- BBP95: trend berekend via toepassing van de HP-filter op de basisreeks, die voor de periode 1993-95 verlengd werd op basis van het Economisch Budget 1995 (juli 1994);
- BBP98: trend berekend via toepassing van de HP-filter op de basisreeks, die voor de periode 1993-98 verlengd werd op basis van de Economische Vooruitzichten 1994-98 (maart 1994).

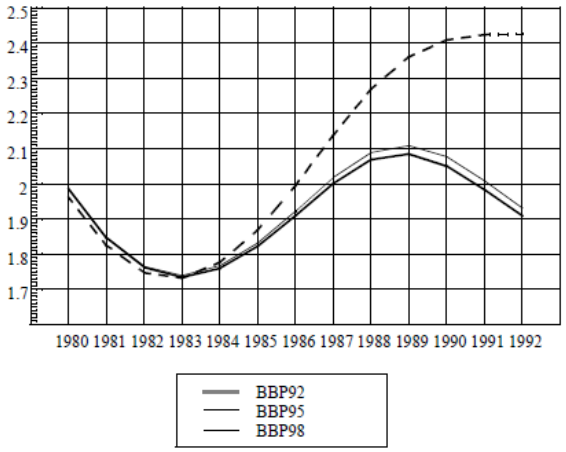
Opvallend is dat de variabele BBP92 aan het einde van de reeks een lichte stijging vertoont, die zich niet of in duidelijk mindere mate voordoet indien de basisreeks beredeneerd wordt doorgetrokken tot 1995, resp. 1998. Dit doet vermoeden dat de trendreeks BBP92 aan het einde (onterecht) een deel van de opgaande cyclische beweging uit het einde van de jaren 80 capteert.

Samengevat kunnen we dus stellen:

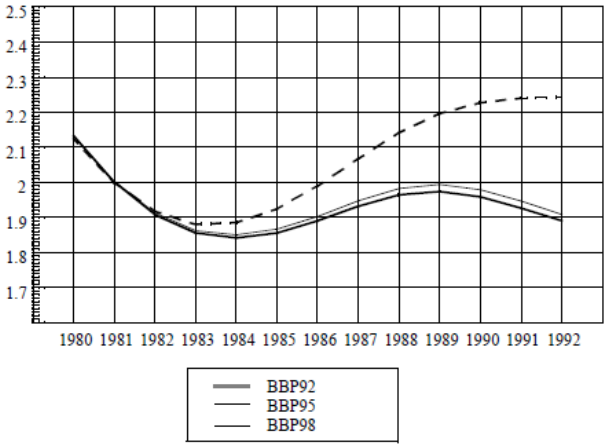
- bij toepassing van de HP-filter ontstaat een bias aan het begin en het einde van de basisreeks: de trendreeks capteert er, in vergelijking met de rest van de periode, een groter deel van de cyclische beweging;
- deze bias is verantwoordelijk voor de instabiliteit bij toevoeging van additionele observaties;
- deze bias kan worden beperkt door de basisreeks te verlengen op basis van de meest waarschijnlijk geachte evolutie;
- verschillen in deze extrapolaties (vergelijk BBP95 en BBP98) leiden tot verschillende trendwaarden, doch deze laatste verschillen vallen relatief kleiner uit dan de verschillen in de extrapolaties van de basisreeks.

Wat de keuze van *lambda* betreft, kan men zich laten leiden door de volgende overweging: de trendreeks wordt bepaald en moet dus kunnen worden verklaard door structurele factoren. Zo is het b.v. duidelijk dat zich sinds de jaren 60 een aantal structuurwijzigingen hebben voorgedaan die de daling van de trendmatige groei van het BBP wettigen. Voor `lambda=100` stelt men echter tijdens de jaren 80 opnieuw een, weliswaar lichte, stijging van de trendmatige groei vast, die zich niet manifesteert bij hogere waarden van *lambda.* Het antwoord op de vraag of deze stijging structureel te verklaren is, kan een leidraad vormen voor de keuze van de parameter `lambda`.

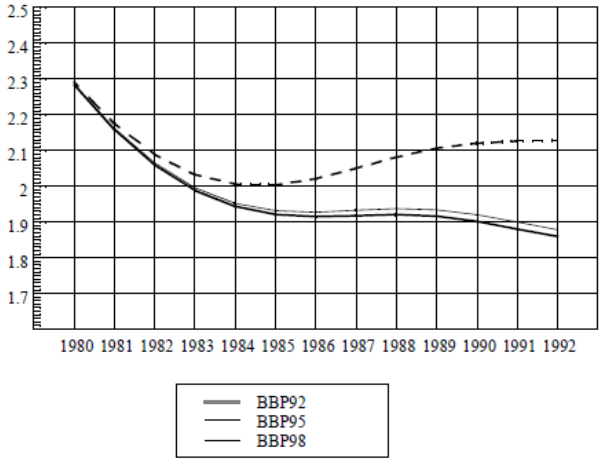
Trendgroei van het BBP tegen constante prijzen: 1980-92 (l=100)



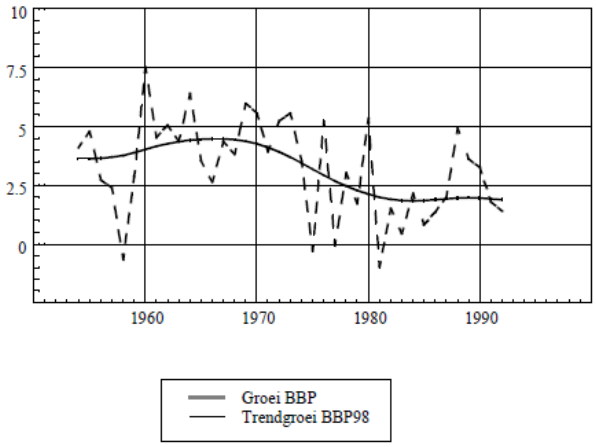
Trendgroei van het BBP tegen constante prijzen: 1980-92 (l=200)



Trendgroei van het BBP tegen constante prijzen: 1980-92 (l=400)



Groei en trendgroei van het BBP tegen constante prijzen: 1953-92 (I=200)



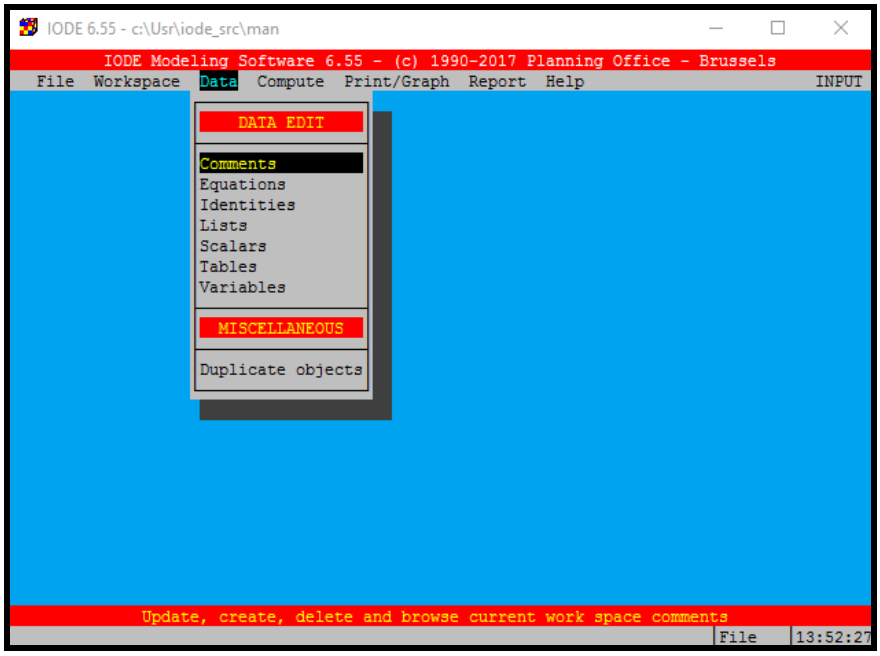
LITERATUUR

De Hodrick-Prescott filter wordt o.a. beschreven in:

- Hodrick R.J. and E.C. Prescott (1980). *U.S. business cycles: an empirical investigation"Carnegie-Mellon University Discussion Paper, nr. 451, Revised November 1980, 27 pp.
- Nicoletti G. and L. Reichlin (1993). Ìand cycles in labour productivity in the major OECD countries"OECD Economic Department Working Papers, nr. 129, 41-42.
- Nunes J. and H. Ongena (1994). Íof trend output in cyclical adjustment of budget balances: robustness of different methods"<String ` European Commission, Directorate-General for Economic and Financial Affairs, 10 pp.
- Prescott E.C. (1986). .ahead of business-cycle measurement"<String ` Carnegie-Rochester Conference Series on Public Policy, nr. 25, 11-44.

2.3 DATA

FIGURE 40



Le menu DATA permet de créer, détruire et modifier des objets dans les différents WS présents en mémoire. Ce menu propose également des fonctions utilitaires comme la recherche de textes dans les objets des WS ou la duplication d'objets.

La partie DATA EDIT se déroule de façon semblable pour tous les types d'objets :

- un premier écran permet d'introduire la liste des objets à éditer
- un deuxième écran présente les objets sélectionnés sous forme d'un tableau dans lequel on pourra se déplacer
- un troisième écran affiche la valeur et permet la modification et la création des objets

La première étape peut être court-circuitée en tapant la touche F10 : on se retrouve alors devant un tableau contenant tous les objets du WS.

Ci-dessous sont exposées les parties communes à l'édition de tous les objets : l'introduction des noms d'objets à éditer et l'utilisation du tableau ('grid' ou 'scroll') des objets sélectionnés.

Dans les chapitres correspondants à l'édition de chaque type d'objet, on trouvera les spécificités de chaque éditeur.

Les fonctions spécifiques aux calculs et aux constructions de listes se trouvent dans le chapitre sur l'édition des listes.

La fonction de duplication d'objets termine ce chapitre.

La suite du chapitre est donc composée des sujets suivants :

- Sélection des objets à éditer
- Edition d'un tableau d'objets
- Edition d'un objet particulier :
  - Comments : édition du WS de commentaires

- Equations : édition du WS d'équations
- Identities : édition du WS d'identités
- Lists : édition du WS de listes
- Scalars : édition du WS de scalaires
- Tables : édition du WS de tableaux
- Variables : édition du WS de variables

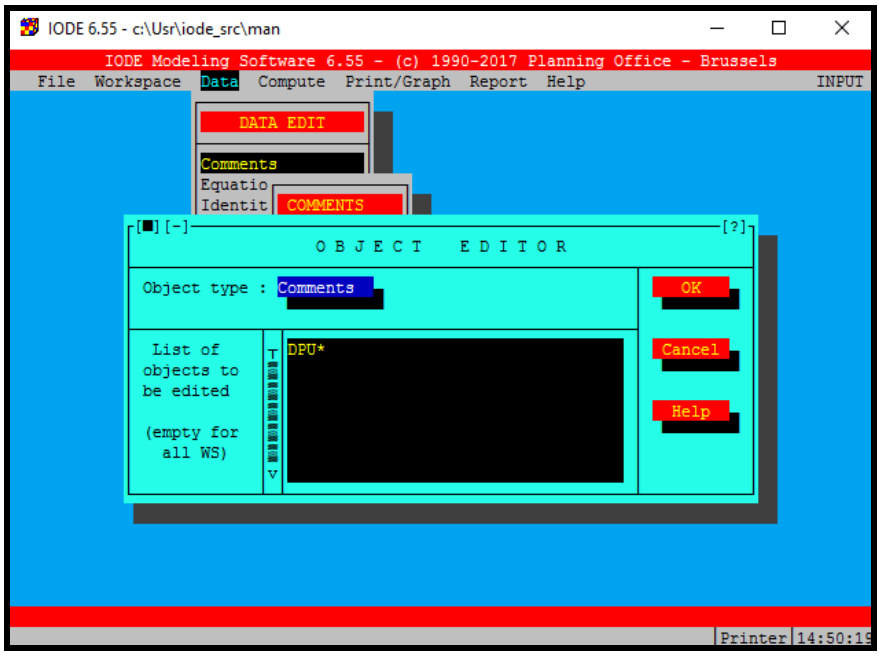
• Duplicate objects : copie d'objets à partir de WS sur disque vers les WS courants

Tous les objets gérés par IODE ont une taille limite de 32000 caractères, pour des raisons tenant à l'architecture du système. Ce n'est en général pas très restrictif, sinon parfois dans le cas des tableaux, qui ne doivent pas dépasser individuellement un nombre de lignes raisonnable (60 à 100). Il en va de même pour les équations et autres formes LEC : le bloc limite de 32000 caractères doit pouvoir contenir toutes les informations ayant trait à l'objet (forme LEC, forme compilée, etc).

2.3.1 Sélection des objets à éditer

Plutôt que d'afficher dans un tableau tous les objets d'un type donné actuellement présents en WS, ce qui rend la visualisation ardue, il est possible de sélectionner une liste d'objets. Un écran de saisie, identique pour tous les types d'objets, est prévu à cette fin.

FIGURE 41



Cet écran contient un champ de type EDITOR destiné à contenir des noms d'objets. Ceux-ci peuvent être entrés séparément ou sous forme de listes. Dans les deux cas, les noms seront séparés par un

des séparateurs suivants :

- le blanc
- la virgule
- le point-virgule
- le retour à la ligne

Si un nom de liste apparaît dans le champ, il doit être préfixé par le caractère $. Ainsi, en supposant que la liste EQS soit définie comme

```
EQ1, EQ2, EQ3, EQ4
```

les objets édités, si on introduit comme liste d'objets

```
EQ0, $EQS, EQ5
```

seront :

```
EQ0, EQ1, EQ2, EQ3, EQ4, EQ5
```

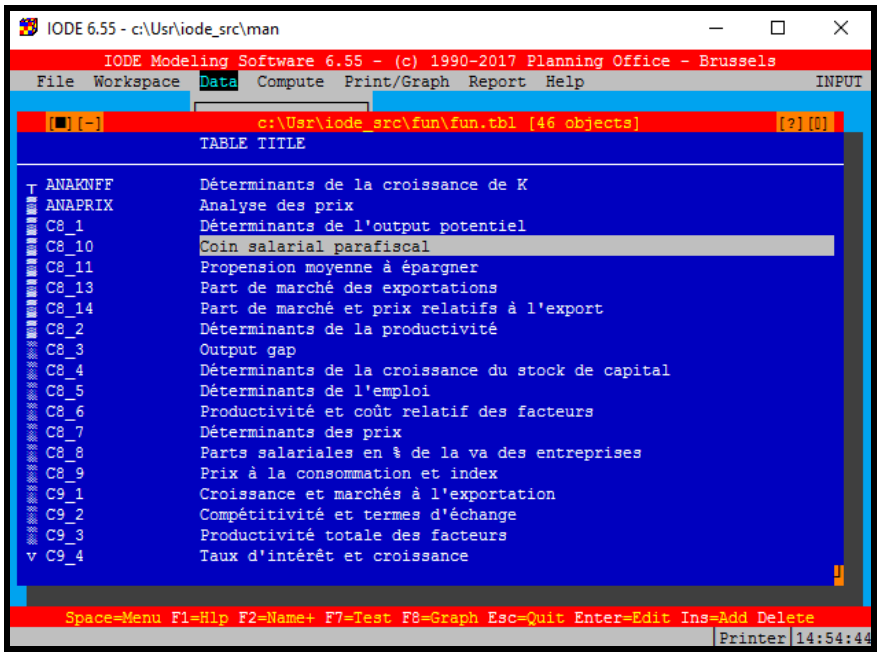
Cas particuliers

Si la liste est vide, tous les objets du WS sont sélectionnés pour l'édition.

Si la liste contient des noms d'objets inexistants, aucun message n'est produit. Les objets non définis ne seront simplement pas repris dans le tableau d'édition.

2.3.2 EDITION D'UN TABLEAU D'OBJETS

FIGURE 42



Le tableau d'édition (appelé aussi "grid") est composé des éléments suivants :

- le nom du WS et le nombre d'objets dans la ligne supérieure
- le nom des objets dans la colonne de gauche
- la définition (ou une partie de la définition) des objets en regard du nom dans la partie de droite
- un scrollbar indiquant la position courante dans le tableau et la proportion visible du tableau
- des icônes sur le contour du tableau

L'objet courant est indiqué par le fait que la ligne qui lui correspond est de couleur différente des autres lignes.

TOUCHES-FONCTIONS

Certaines touches vont déclencher des actions soit sur les objets eux-mêmes, soit sur l'affichage du tableau :

Sur les objets

- ENTER : édite l'objet courant (cliquer avec la souris a le même effet)
- INS : crée un nouvel objet
- DEL : détruit l'objet courant (après confirmation)
- Alt-F1 : ouvre un sous-tableaux avec les commentaires liés à la définition de l'objet courant. On peut de la sorte par exemple afficher les commentaires de toutes les séries utilisées dans la définition de l'équation courante.
- Alt-F2 : idem Alt-F1 avec affichage des Equations
- Alt-F3 : idem Alt-F1 avec affichage des Identités
- Alt-F4 : idem Alt-F1 avec affichage des Listes
- Alt-F5 : idem Alt-F1 avec affichage des Scalaires
- Alt-F6 : idem Alt-F1 avec affichage des Tableaux
- Alt-F7 : idem Alt-F1 avec affichage des Variables

Sur le choix de l'objet courant

- Flèche en haut : l'objet précédent devient l'objet courant
- Flèche en bas : l'objet suivant devient l'objet courant
- Page Up : le tableau défile d'un écran vers le haut. L'objet courant s'adapte en conséquence
- Page Down : le tableau défile d'un écran vers le bas. L'objet courant s'adapte en conséquence
- Ctrl-Home : positionnement sur le premier objet du tableau
- Ctrl-End : positionnement sur le dernier objet du tableau
- Lettre : déplacement sur l'objet suivant commençant par la lettre frappée.

Dans le cas des scalaires, des tableaux et des variables, les flèches à droite et à gauche permettent en plus de se déplacer sur les colonnes à éditer (périodes dans le cas des variables).

Sur l'affichage

- Ctrl-O (mOve) : permet de déplacer le tableau dans l'écran
- Ctrl-R (Rotate) : inverse lignes et colonnes (pour les variables)
- Ctrl-Z (resiZe) : permet de modifier la taille du tableau
- Ctrl-X (maXimize) : donne au tableau sa taille maximum

Effet global

- F1 Accès à ce manuel en ligne
- Escape : quitte le tableau
- Espace : ouvre le menu local à la fenêtre

Autres touches fonctions

Selon le type d'objet affiché, différentes touches fonctions peuvent être définies. Elles apparaissent dans la ligne de commentaire de l'écran (avant-dernière ligne) et dans le menu local ([=]).

SOURIS

La souris facilite un certain nombre d'opérations dans l'utilisation des grids :

- un simple clic permet de positionner le curseur sur l'objet voulu, et à la position souhaitée. Un second clic édite l'objet sélectionné.
- sur le contour de la fenêtre, des petites icônes permettent d'effectuer certaines opérations :
  - [o] : fermer la fenêtre (= ESCAPE)
  - [=] : accès au menu de la fenêtre reprenant toutes les touches fonctions définies localement
  - [?] : accès au manuel en ligne
  - [o/] : rotation de la fenêtre (variables)
  - [flèche] : maximiser et minimiser la taille de la fenêtre
- les touches fonctions sont mises en exergue dans la zone de commentaire. La souris permet de "presser" ces touches sans avoir recours au clavier.
- la fenêtre peut être déplacée en plaçant la souris sur son contour et en gardant le bouton appuyé pendant le déplacement
- la taille de la fenêtre peut être modifiée en plaçant la souris sur le coin inférieur droit et en "tirant" tout en gardant enfoncé le bouton de la souris.
- les options du menu déroulant sont directement accessibles à l'aide de la souris, comme à l'aide des touches Alt-lettre. Attention, cela quitte le tableau en cours d'édition!
- dans les scrollbars (vertical ou horizontal) la souris permet de placer la fenêtre à l'endroit souhaité par un simple clic.

Cas particulier

Si aucun objet n'est défini dans le WS courant, le tableau est affiché mais ne contient évidemment pas d'objet. Il n'y a donc pas d'objet courant, mais toutes les fonctions qui ne sont pas liées à l'objet courant sont disponibles (move, create, help, etc).

2.3.3 COMMENTAIRES

Les commentaires sont, avec les listes, les objets les plus simples : ce sont simplement des textes libres. Ceux-ci sont édités à l'aide d'un champ EDITOR, et peuvent donc être chacun un texte complet, éventuellement de plusieurs pages.

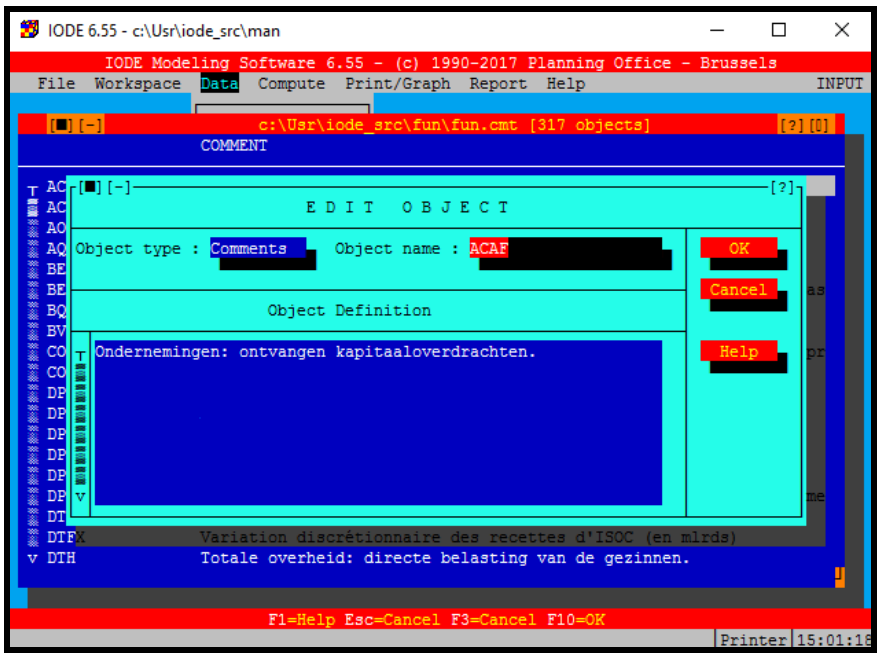
L'édition se fait en trois étapes:

- Sélection des objets à éditer (voir plus haut)
- Edition d'un tableau d'objets (voir plus haut)
- Edition des objets individuels

Les deux premières étapes ont été décrites plus haut.

Lors de l'édition des objets (création ou modification), l'écran de saisie se présente comme suit :

FIGURE 43



Nom

Le nom de l'objet apparaît dans le premier champ de l'écran de saisie. Si le nom d'un objet est modifié alors que celui-ci existe, un nouveau commentaire est créé portant le nouveau nom. L'ancien commentaire existe toujours, dans son état d'origine. Cette propriété offre une méthode très simple pour copier des commentaires.

Valeur

Le texte du commentaire s'édite dans le champ EDITOR de l'écran de saisie. Toutes les fonctions habituelles de ce type de champ sont disponibles.

Touches fonctions

Pour sauver l'objet après encodage ou modification, il suffit de presser F10. Si l'objet existe dans le

WS courant, il faut confirmer le remplacement de l'ancien objet par la nouvelle définition.

Pour quitter sans sauver, il suffit de presser ESCAPE.

2.3.4 Equations

Les équations sont, avec les tableaux, les objets les plus complexes gérés par le logiciel IODE. Si au départ il ne s'agit que d'expressions LEC, on ne peut les dissocier des informations qui ont permis l'estimation des coefficients qu'elles contiennent. Ce point du menu permet non seulement la définition des équations, mais également l'estimation des coefficients et l'affichage de graphiques.

Les équations de IODE contiennent donc les informations suivantes :

- une équation LEC
- un commentaire (ou titre)
- une méthode et une période d'estimation
- le bloc d'équations dont elle fait partie lors de l'estimation
- les éventuels instruments (formes LEC) si la méthode d'estimation retenue en requiert.

Bien sûr, toutes les équations ne nécessitent pas l'ensemble de ces informations. Certaines équations ne sont d'ailleurs que de simples raccourcis d'écriture dans les modèles qui ne font intervenir aucun coefficient et se ramènent donc à de simples formes LEC, accompagnées éventuellement d'un commentaire.

Nom des équations

Une contrainte importante sur le nom des équations est à souligner : le nom d'une équation est toujours le nom de sa variable endogène. Ceci implique entre autre qu'une seule équation peut définir une variable endogène dans un WS d'équations.

Par conséquent, les noms des équations répondent aux mêmes contraintes que ceux des variables : 10 caractères alphanumériques maximum (20 à partir de la version 6.01), commençant par une lettre ou le caractère de soulignement (_).

Cette façon de nommer les équations peut sembler par trop restrictive. En fait, il s'agit surtout de faciliter la tâche de l'utilisateur : lorsqu'il s'agira plus tard de simuler un modèle, la liste des équations permettra à elle seule de déterminer la liste des variables endogènes et exogènes.

Création et édition des équations

La création et l'édition des équations se fait en trois étapes:

- Sélection des objets à éditer (voir plus haut)
- Edition d'un tableau d'objets (voir plus haut)
- Création et édition des équations individuelles

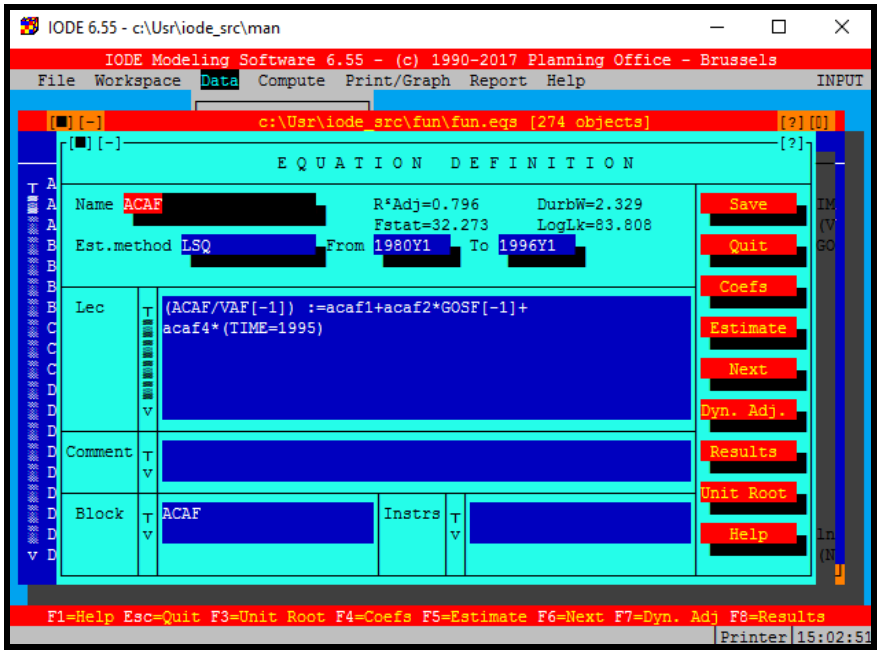
Les deux premières sont classiques et ont été détaillées en début de chapitre.

Pour créer une nouvelle équation, il suffit de presser la touche INS. Un écran vide permet l'encodage des informations utiles. Une alternative est d'éditer une équation existante proche de celle à créer et de modifier le nom de l'équation et sa définition dans l'écran de saisie. Après sauvetage, une nouvelle équation est construite.

L'écran de définition d'une équation doit au minimum contenir un nom et une forme LEC.

Pour modifier une équation, il faut d'abord se placer sur celle à éditer. Cela se fait à l'aide des touches fléchées. En pressant la touche ENTER, l'écran de définition suivant apparaît :

FIGURE 44



Name

Ce champ peut être modifié. Une nouvelle équation est alors créée au sortir de l'édition avec le nouveau nom. Si une équation portait déjà ce nom, elle est remplacée.

Estimation method

Ce champ permet de définir la méthode d'estimation à utiliser. Le choix est offert entre 4 méthodes:

- LSQ simple
- Zellner
- Instrumental
- GLS

From et To : période d'estimation

La période d'estimation éventuelle doit tenir compte des lags et leads, ainsi que des périodes fixées dans l'équation : si le WS de variables est défini sur la période 1960Y1 à 1990Y1, une équation contenant par exemple "d(X)+Y[+1]" ne peut être estimée que sur la période 1961Y1 à 1989Y1.

LEC

La forme LEC de l'équation est éditée à l'aide d'un champ EDITOR. Toutes les fonctions d'édition sont utilisables et notamment la copie de blocs, y compris en provenance d'autres équations. Une équation a toujours la forme :

```
lec_expression := lec_expression
```

Le nom de la variable endogène (et donc de l'équation) doit apparaître au moins une fois dans le texte de l'équation. Elle peut apparaître plusieurs fois et ne doit pas nécessairement se trouver dans le membre de gauche.

La description complète du langage LEC - syntaxe, opérateurs et fonctions - se trouve dans un chapitre séparé.

Comment

Ce champ EDITOR permet d'encoder un commentaire libre concernant l'équation, son estimation, etc.

Block

Ce champ permet de spécifier le bloc d'équations qui sont à estimer simultanément. Toutes les méthodes d'estimation acceptent d'estimer plusieurs équations simultanées.

Les noms des équations du bloc sont séparés par des blancs, virgules, points-virgules ou retour à la ligne.

Instrs

Les instruments utilisés dans les méthodes d'estimations utilisant une modification de métrique sur base d'instruments doivent être introduits dans ce champ. Les instruments sont de simples formes LEC. Ils sont séparés par points-virgules.

TOUCHES FONCTIONS

Une touche fonction est associée à chaque bouton défini dans l'écran :

- F10 : sauve l'équation et quitte son édition
- ESCAPE : quitte l'équation sans sauver. Une confirmation est demandée pour éviter la perte du travail suite à une fausse manoeuvre.
- F3 : tests de Dicky-Fuller.
- F4 : affiche les valeurs des coefficients présents dans les équations du bloc courant. Si ces coefficients n'existent pas dans le WS courant de coefficients, ceux-ci prennent comme valeur 0.9, avec 1.0 comme paramètre de relaxation.
- F5 : lance le processus d'estimation en prenant comme paramètres ceux affichés dans l'équation courante (période, méthode et instruments).
- F6 : affiche l'équation suivante du bloc, s'il y en a une.
- F7 : ajustements dynamiques automatiques. Deux méthodes sont disponibles :
  - Partial Adjustment (1 coefficient c1):

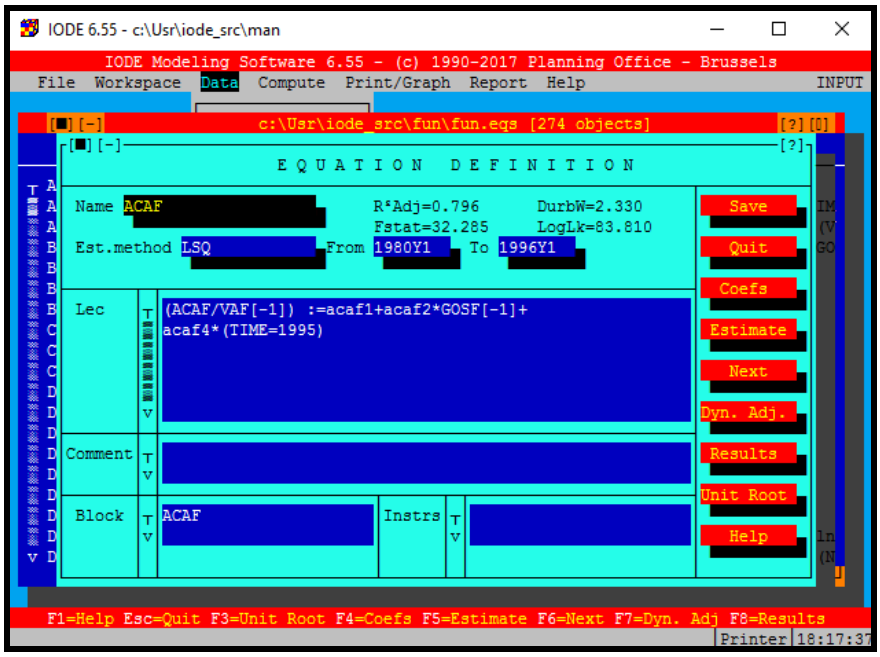
```
lhs := rhs -> d(lhs) := c1 * (rhs -(lhs)[-1])
```

  - Error Correction Model (deux coefficients c1 et c2):

```
lhs := rhs -> d(lhs) := c1 * d(rhs) + c2 * (rhs-lhs)[-1]
```

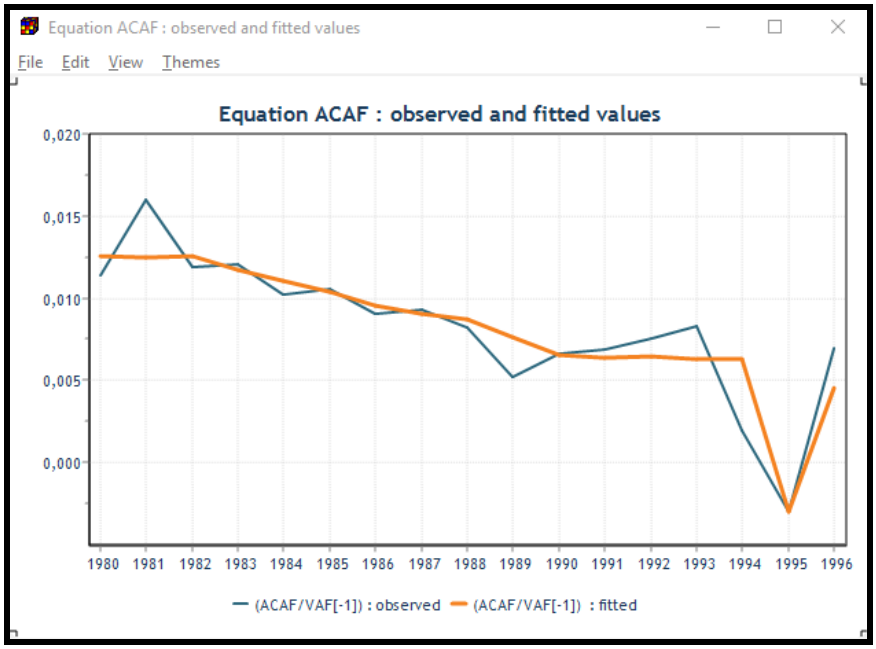
- F8 : affiche un menu permettant de visualiser et d'imprimer les résultats de l'estimation : coefficients, matrice de corrélation, tests statistiques, graphiques des résidus.

FIGURE 45



En pressant ENTER sur une des options, les valeurs correspondantes sont affichées.

FIGURE 46

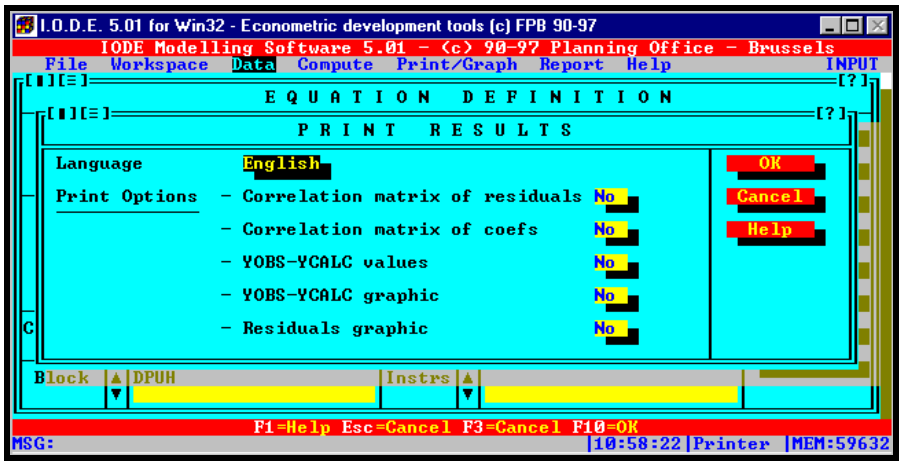


RÉSULTATS DE L'ESTIMATION

Les résultats de l'estimation (touche F8) ne sont disponibles que si une estimation vient d'avoir lieu. Les résultats disponibles sont:

- Coefficients : affiche dans un tableau les valeurs des coefficients estimés ainsi que les tests statistiques (standard error et t-stat).
- Correlation matrix : affiche dans un tableau la matrice de corrélation des coefficients.
- Results by equation : liste de résultats statistiques associés aux équations :
  - Correlation
  - St dev of residuals
  - Mean of YOBS
  - Sum of Squares of Residuals
  - Standard error
  - Standard error in %
  - F-stat
  - R square
  - R square adjusted
  - Durbin-Watson
  - Log Likelihood
- Graph of YOBS-YEST : affiche un graphique avec les courbes observées et estimées. Dans le cas d'un bloc d'équations, l'écran est divisé en autant de partie qu'il y a d'équations.
- Graph of Residuals : affiche un graphique avec la courbe résultant de la différence entre les observations et l'estimation. Dans le cas d'un bloc d'équations, il y a autant de graphiques que d'équations dans le bloc.
- Print graph : permet d'imprimer un des deux graphiques définis ci-dessus. Le langage dans lequel les libellés doivent être indiqués peut être sélectionné au niveau de cette page.
- Print output : l'écran de saisie suivant permet de sélectionner les différents paramètres nécessaires: langage pour les titres automatiques, et sélection des informations optionnelles à imprimer.

FIGURE 47



Un exemple d'impression est affiché ci-dessous :

```
ESTIMATION

1. INPUT

1.1 PARAMETERS

  - Estimation period : 1970Y1:1990Y1
  - Estimation method : LSQ
  - Number of observations : 21
  - Number of equations : 1
  - Number of coefficients : 2
  - Number of instruments : 0
  - Max number of iterations : 100
  - Convergence limit : 0.000010

1.2 EQUATIONS

  - B := c1 + c2 * A

2. RESULTS

2.1 COEFFICIENTS AND TESTS

 COEF    VALUE    ST ERR     T-STAT     RELAX
-----------------------------------------------
 c1    1.883107 0.037492 50.226407 1.000000
 c2    0.053122 0.001795 29.601900 1.000000

2.2 CORRELATION MATRIX OF COEFFICIENTS

         c1         c2
-----------------------
 c1   1.000000 -0.957095
 c2  -0.957095  1.000000

2.3 CORRELATION OF RESIDUALS

        B
-----------
 B 1.000000
```

```
2.4 RESULTS BY EQUATION

2.4.1 EQUATION : B

  - B := c1 + c2 * A

                                    VALUE
-----------------------------------------
 Number of coefficients                 2
 Number of observations                21
 Standard deviation on YOBS      0.333167
 Mean of YOBS                    2.945543
-----------------------------------------
             TESTS                  VALUE
-----------------------------------------
 Sum of square of residuals      0.047116
 Standard error                  0.049797
 Standard error in %             1.690597
 F-Stat                        876.245980
 R2                              0.978777
 R2 adjusted                     0.977660
 Durbin-Watson test              0.134944
 Log likelihood                 34.248870
-----------------------------------------

           ACTUAL AND FITTED VALUES
-------------------------------------------------
 PERIOD   YOBS      YCALC      RES        RES%
-------------------------------------------------
 1970Y1 2.302585 2.414325 -0.111740 -4.852798
 1971Y1 2.397895 2.467447 -0.069551 -2.900520
 1972Y1 2.484907 2.520569 -0.035662 -1.435142
 1973Y1 2.564949 2.573690 -0.008741 -0.340793
 1974Y1 2.639057 2.626812  0.012245  0.463996
 1975Y1 2.708050 2.679934  0.028116  1.038243
 1976Y1 2.772589 2.733056  0.039533  1.425844
 1977Y1 2.833213 2.786178  0.047036  1.660148
 1978Y1 2.890372 2.839300  0.051072  1.766976
 1979Y1 2.944439 2.892421  0.052017  1.766635
 1980Y1 2.995732 2.945543  0.050189  1.675351
 1981Y1 3.044523 2.998665  0.045857  1.506226
 1982Y1 3.091043 3.051787  0.039256  1.269978
 1983Y1 3.135494 3.104909  0.030585  0.975458
 1984Y1 3.178054 3.158031  0.020023  0.630047
 1985Y1 3.218876 3.211152  0.007723  0.239942
 1986Y1 3.258096 3.264274 -0.006178 -0.189615
 1987Y1 3.295837 3.317396 -0.021559 -0.654135
 1988Y1 3.332205 3.370518 -0.038313 -1.149791
 1989Y1 3.367296 3.423640 -0.056344 -1.673273
 1990Y1 3.401197 3.476762 -0.075564 -2.221694
```

MODE DE SAUVETAGE

Au cours de l'édition d'un bloc d'équations ou d'une équation seule, les valeurs introduites pour modifier les équations ne sont pas sauvées dans le WS d'équations. Le sauvetage n'a lieu qu'au moment de quitter l'équation ou le bloc.

Il en va de même pour les valeurs des coefficients qui ne sont sauvées dans le WS courant de coefficients qu'en fin d'estimation après confirmation du sauvetage par F10.

Ainsi, des coefficients qui n'existent pas au moment de l'entrée en édition d'une équation et qui sont créés automatiquement par l'estimation ne sont pas sauvés si l'équation ne l'est pas.

SCALAIRES ET VARIABLES RÉSULTANT DE L'ESTIMATION

Les tests résultant de la dernière estimation sont sauvés dans des scalaires de façon à permettre différents calcul par la suite. Il en va de même pour les résidus, membres de gauche et de droite des équations.

Les tests portent les noms suivants (e0_* pour la première équation du bloc, e1_* pour la deuxième, ...) :

- **e0_n** : nombre de période du sample
- **e0_k** : nombre de coefficients estimés
- **e0_stdev** : std dev des résidus
- **e0_meany** : moyenne de Y
- **e0_ssres** : somme du carrés des résidus
- **e0_stderr** : std error
- **e0_stderrp** : std error %
- **e0_fstat** : F-Stat
- **e0_r2** : R carré
- **e0_r2adj** : R carré ajusté
- **e0_dw** : Durbin-Watson
- **e0_loglik** : Log Likelihood

Les séries calculées sont également sauvées dans une variable sous les noms:

- **_YCALC0** pour le membre de droite de la première équation du bloc, **_YCALC1** pour la deuxième équation, etc.
- **_YOBS0** pour le membre de gauche de la première équation du bloc, **_YOBS1** pour la deuxième équation, etc.
- **_YRES0** pour les résidus de la première équation du bloc, etc.

En dehors du sample d'estimation, les valeurs de la série sont --.

2.3.5 IDENTITÉS

Les identités sont des expressions LEC, formules utilisées pour construire des séries sur base d'autres variables. Le nom d'une identité est celui de la série qui sera construite sur base de la formule définie.

Les identités sont introduites à l'aide d'un champ EDITOR et sont donc des textes s'étendant éventuellement sur plusieurs lignes.

L'édition se fait en trois étapes:

- Sélection des objets à éditer (voir plus haut)
- Edition d'un tableau d'objets (voir plus haut)
- Edition des objets individuels

Les deux premières sont standards et ont été détaillées en début de chapitre.

A noter que l'exécution des identités se fait via le point "Compute" de la barre d'actions en sélectionnant l'option "Execute identities".

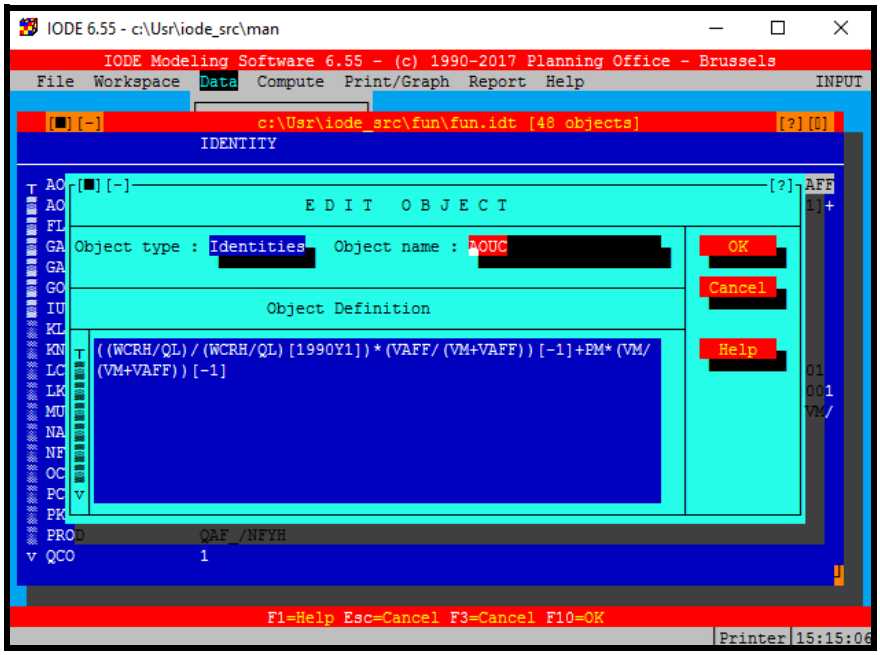
TOUCHES FONCTIONS SPÉCIFIQUES AU IDENTITÉS

- F7 dans le tableau déroulant des identités exécute l'identité courante sur toute la période du WS
- Shift-F7 lance la fonction #IdtExecute

EDITION DES IDENTITÉS

Lors d'une création ou d'une modification des identités, l'écran de saisie est identique à celui des commentaires ou des listes et se présente comme suit :

FIGURE 48



Nom

Le nom de l'objet apparaît dans le premier champ de l'écran de saisie. Si le nom d'un objet est modifié alors que celui-ci existe, une nouvelle identité est créée portant le nouveau nom. L'ancienne existe toujours, dans son état d'origine. Cette propriété permet par exemple de copier des identités sans recourir à la fonction spécifique de copie d'objet, en opérant éventuellement une modification dans la définition de la nouvelle identité. Ainsi, si l'identité X_B est définie comme :

```
X1_B + X2_B + X3_B
```

et que l'on veuille créer une identité X_F identique, mais où les _B sont remplacés par des _F, il suffit d'éditer cette identité X_B, et

- d'en changer le nom : X_F
- de remplacer les _B par des _F dans le texte :

```
X1_F + X2_F + X3_F
```

En pressant F10, une nouvelle identité est construite sur base de la première.

Valeur

Le texte de l'identité s'édite dans le champ EDITOR de l'écran de saisie. Toutes les fonctions habituelles de ce type de champ sont disponibles.

La description complète du langage LEC - syntaxe, opérateurs et fonctions - se trouve dans un chapitre séparé.

Touches fonctions

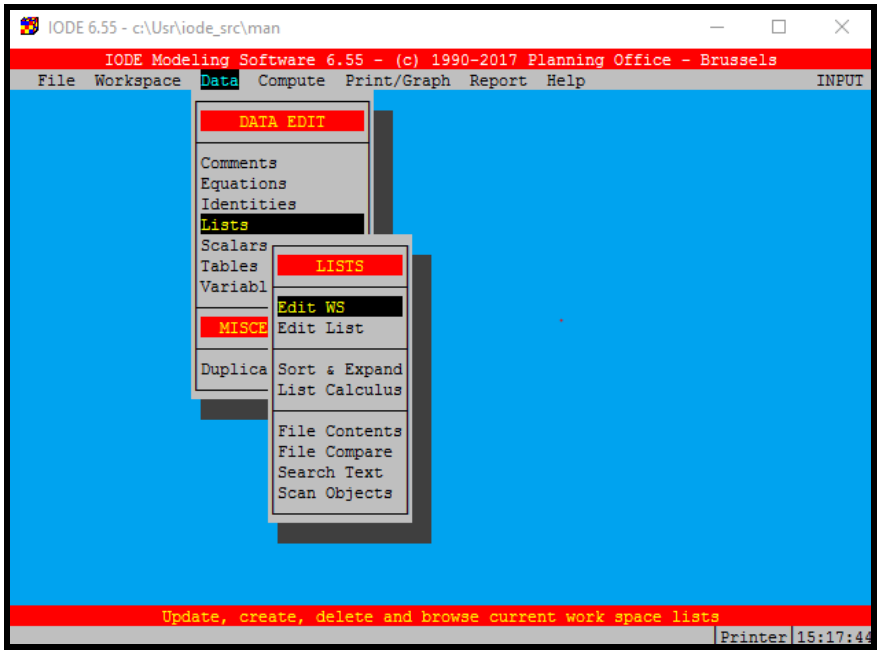
Pour sauver l'objet après encodage ou modification, il suffit de presser F10. Si l'objet existe dans le WS courant, il faut confirmer le remplacement de l'ancien objet par la nouvelle définition.

En cas d'erreur de syntaxe dans la forme LEC, un message indique l'erreur et le programme revient en édition. Il est impossible de sauver une identité dont la forme LEC est syntaxiquement incorrecte.

Pour quitter sans sauver, il suffit de presser ESCAPE.

2.3.6 LISTES

FIGURE 49



Les listes sont des textes libres, représentant en général des suites de noms d'objets. Elles peuvent cependant contenir n'importe quel texte. Seul le contexte de l'utilisation des listes décidera de leur validité.

Les listes sont éditées à l'aide d'un champ EDITOR et sont donc des textes s'étendant éventuellement sur plusieurs lignes, voir plusieurs pages. La limite de 32K, valable pour tous les objets de IODE, est la seule contrainte.

L'édition se fait en trois étapes:

- Sélection des objets à éditer (voir plus haut)
- Edition d'un tableau d'objets (voir plus haut)
- Edition des objets individuels

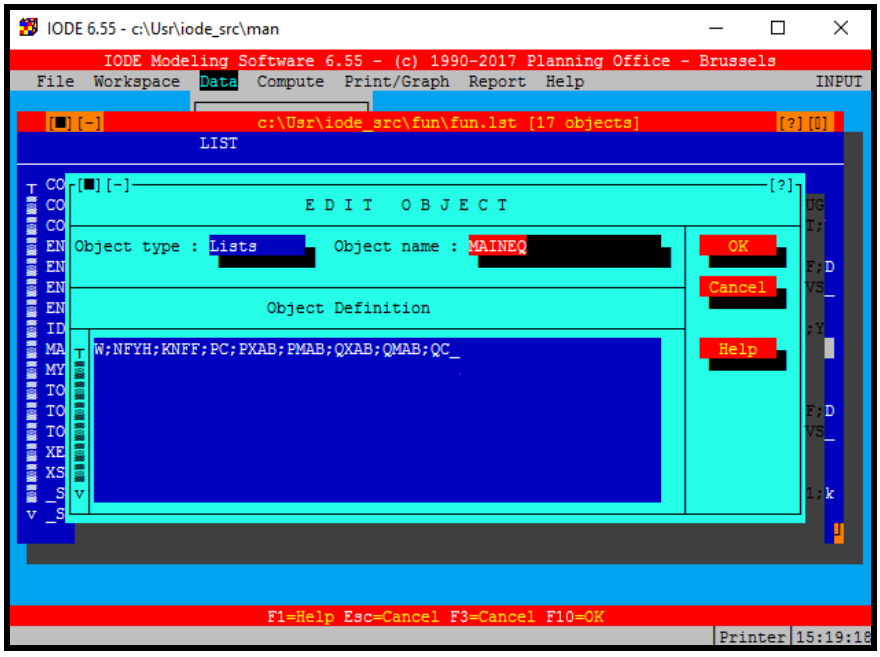
D'autre part, une série d'opérations peuvent être réalisées pour construire des listes:

- Sort & Expand : tri et extension des listes
- List Calculus : opérations sur les listes (intersection, différence, union).
- File Contents : crée une liste avec le contenu du fichier
- File Compare : comparaison entre un fichier et le WS
- Search Text : recherche de texte dans les WS
- Scan objects : recherche des objets référencés dans d'autres

Les deux premières ont été détaillées en début de chapitre.

Lors d'une création ou d'une modification de liste, l'écran de saisie se présente comme suit :

FIGURE 50



Nom

Le nom de l'objet apparaît dans le premier champ de l'écran de saisie. Si le nom d'un objet est modifié alors que celui-ci existe, une nouvelle liste est créée portant le nouveau nom. L'ancienne existe toujours, dans son état d'origine. Cette propriété permet par exemple de copier des listes sans recourir à la fonction spécifique de copie d'objet, en opérant éventuellement une modification dans la définition de la nouvelle liste.

Valeur

Le texte de la liste s'édite dans le champ EDITOR de l'écran de saisie. Toutes les fonctions habituelles de ce type de champ sont disponibles.

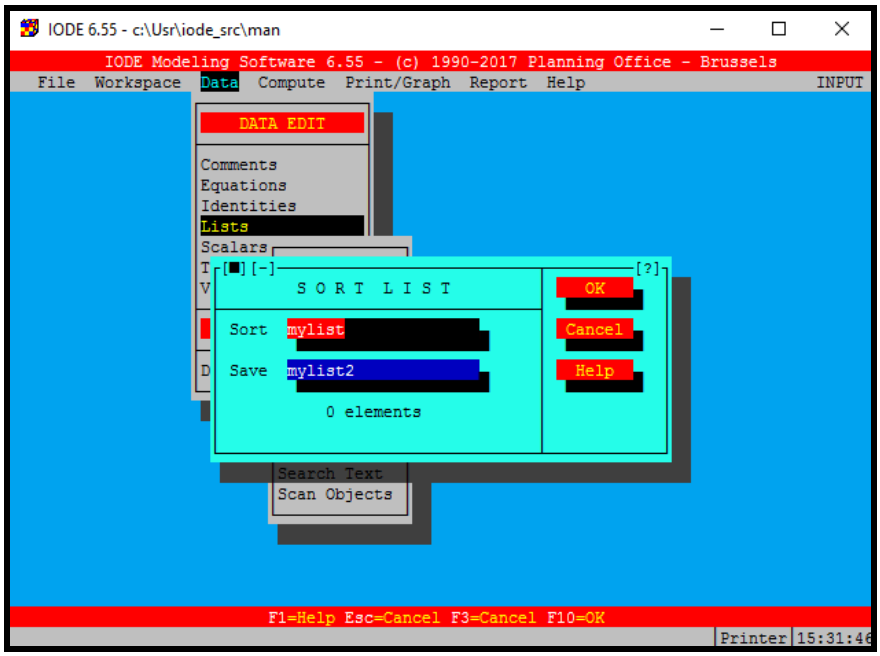
Touches fonctions

Pour sauver l'objet après encodage ou modification, il suffit de presser F10. Si l'objet existe dans le WS courant, il faut confirmer le remplacement de l'ancien objet par la nouvelle définition.

Pour quitter sans sauver, il suffit de presser ESCAPE.

2.3.6.1 SORT & EXPAND

FIGURE 51



Cette fonction permet d'effectuer un tri alphanumérique sur le contenu d'une liste.

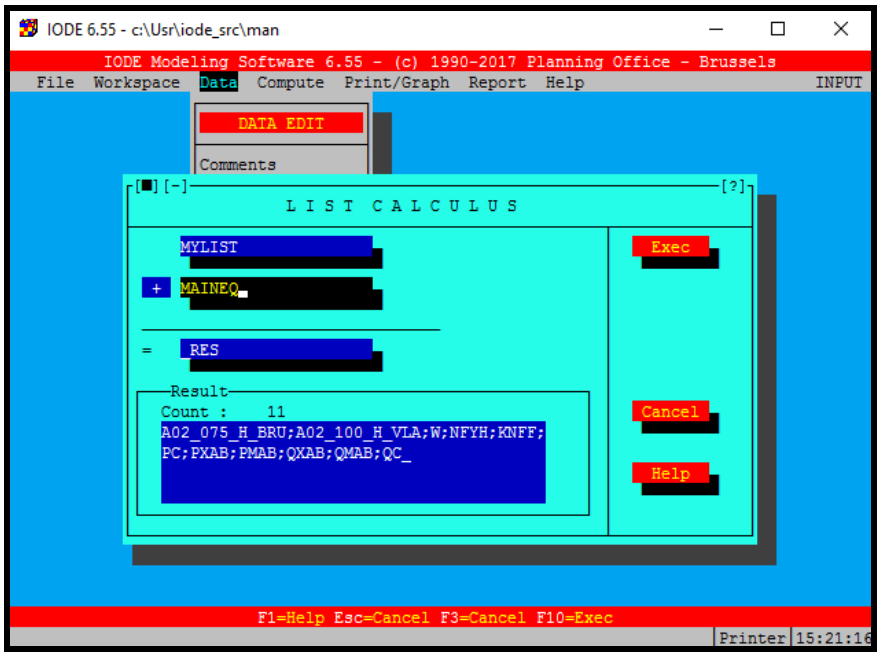
Les deux champs à remplir dans la page de saisie permettent de préciser le nom de la liste à trier et le nom de la liste résultat.

Si la liste résultat existe déjà, elle est remplacée par la nouvelle valeur. Le tri s'effectue de la façon suivante : la liste est découpée en "mots" aux positions des séparateurs (espace, virgules et sauts de lignes). Ces mots sont triés par ordre alphabétique et une liste est reconstruite en séparant les mots par des virgules. Les lignes de plus de soixante caractères sont découpées.

2.3.6.2 LIST CALCULUS

Cette fonction permet de réaliser des opérations simples sur des listes.

FIGURE 52



- intersection : calcul des éléments communs à deux listes
- différence : calcul des éléments d'une liste non présents dans une autre
- union : création d'un nouvelle liste reprenant de façon unique tous les éléments de deux listes

Quatre champs doivent être introduits pour préciser :

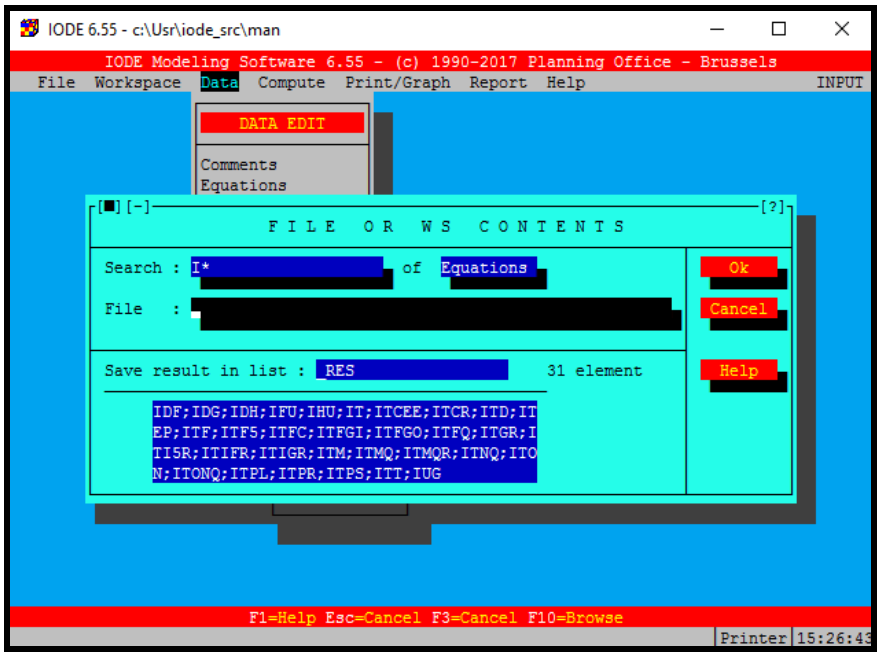
- le nom de la liste résultat
- le nom de la première opérande
- l'opérateur à utiliser :
  - \+ pour l'union
  - \- pour la différence
  - * pour l'intersection
- la seconde opérande

Le résultat du calcul est placé dans la liste dont le nom est fixé par le premier champ de l'écran. Ce résultat ainsi que le nombre d'éléments qu'il contient est également représenté dans la seconde partie de l'écran.

2.3.6.3 File Contents

Cette fonction permet de créer une liste avec les noms d'objets du WS actif ou d'un fichier qui correspondent à un critère donné.

FIGURE 53



Search

Indique le critère sur le nom des objets à retenir dans la liste. Ce critère peut contenir les caractères `*` ou `?`. Pour obtenir toutes les séries commençant par `I`, on introduira :

```
Search : I*
```

On précisera dans le champ suivant le type d'objet à rechercher.

File

Indique le nom du fichier où la recherche doit être effectuée. Si ce nom est laissé blanc, la recherche a lieu dans le WS actif.

Save result in ...

Indique le nom de la liste qui doit recevoir le résultat de la recherche.

Résultat

La deuxième partie de l'écran contient le nombre d'objets trouvés (et sauvés dans la liste résultat), ainsi que les premiers éléments de cette liste.

2.3.6.4 File Compare

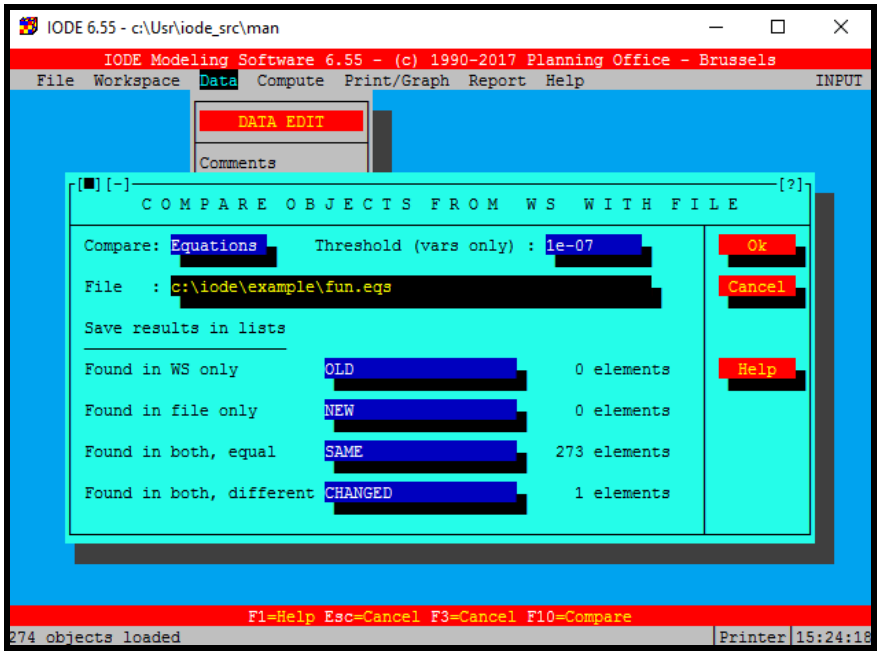
Le contenu du WS courant peut être comparé à celui d'un fichier. Le résultat de cette comparaison

est composé de 4 listes :

- objets trouvés dans le WS seulement
- objets trouvés dans le fichier seulement
- objets trouvés dans les deux, avec la même définition
- objets trouvés dans les deux, avec une définition différente

Pour spécifier les paramètres de la requête, l'écran contient les champs suivants:

FIGURE 54



Compare

Spécifie le type d'objet à comparer.

File

Ce champ est ici requis. Il indique le fichier dont le contenu doit être comparé à celui du WS actif.

Save results in lists

Les quatre champs qui suivent contiendront le nom des listes destinées à recevoir les informations calculées.

A droite, on trouvera, après le calcul, le nombre d'éléments de chaque liste.

FONCTION DE RAPPORTS

La même fonction peut êtte utilisée dans un rapport :

```
$DataCompareXxx file ONE TWO THREE FOR

      ONE         in WS only
      TWO         in file only
      THREE       in both, equal
      FOR         in both, different
```

2.3.6.5 SEARCH TEXT

Il est souvent utile de savoir dans quels objets intervient un texte, comme un nom de série ou de scalaire par exemple. Cette fonction permet de rechercher dans un des WS courants la liste des objets contenant un string donné.

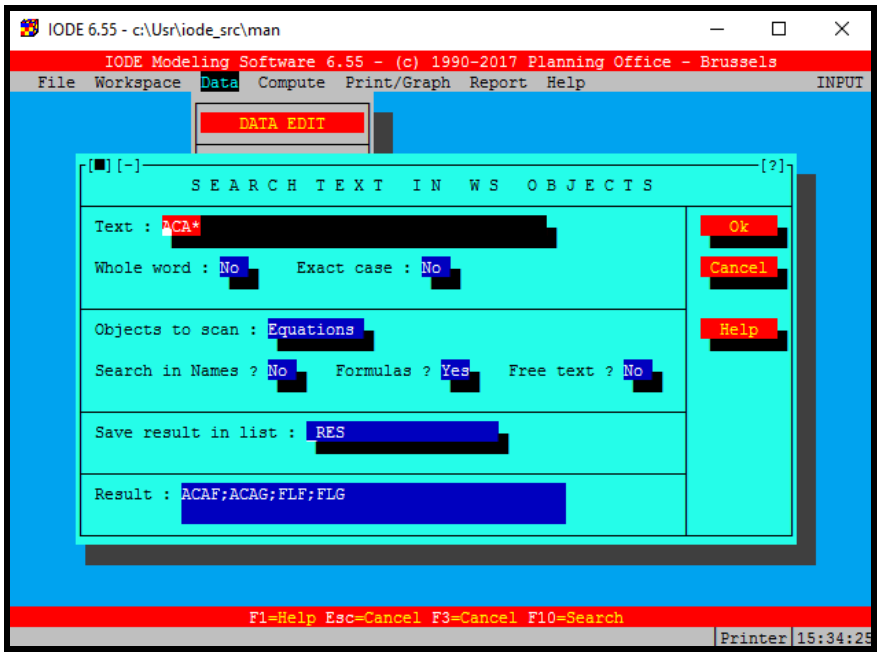
Une application immédiate est de rechercher toutes les équations ou tous les tableaux qui contiennent dans leur définition un nom de variable.

Il est également possible de générer automatiquement la liste de toutes les variables dont le nom répond à certaines caractéristiques.

Le résultat de cette recherche est une liste d'objets qui peut au besoin être sauvée dans le WS courant de listes. Par défaut la liste "_RES" contient le résultat de la dernière recherche.

L'écran se présente comme suit:

FIGURE 55



Text

Le texte à rechercher peut contenir des caractères spéciaux qui permettent de spécifier les limites

de la recherche :

- * : n'importe quelle suite de caractères (même vide)
- ? : un et un seul caractère (quelconque)
- @ : n'importe quel caractère alphanumérique
- ampersand : n'importe quel caractère non alphanumérique
- | : n'importe quel caractère alphanumérique ou aucun en début et fin de string
- ! : n'importe quel caractère non alphanumérique ou aucun en début et fin de string
- \ placé devant un des caractères spéciaux supprime son interprétation

Ainsi la recherche de

```
A*_B
```

aboutira à la sélection des équations

```
ln X := c1 + c2 * l A12_BC
                    |**|
A_B  := X + Y
|||
Z    := c1 + a * A + c3 * C_B
                 |*********|
```

mais pas de

```
ln AB := c1 + c2 * X
```

Mot entier/partiel (Whole word)

Préciser Yes si la chaîne à rechercher doit être un mot entier et non une partie de mot. Indiquer No si cela n'a pas d'importance.

Majuscule/minuscule (Exact case)

Préciser si la recherche doit différencier majuscules et minuscules dans la chaîne à rechercher.

Type d'objet (Object to scan)

Le type d'objet doit être indiqué dans ce champ. En pressant TAB, la liste des types apparaît. Il suffit de se placer sur le type choisi et de presser ENTER.

Les recherches sont différentes en fonction du type d'objet :

- Commentaires : le nom et le texte du commentaire sont analysés
- Equations : le nom et la forme LEC de l'équation sont analysés
- Identités : le nom et la forme LEC de l'identité sont analysés
- Listes : le nom et le texte de la liste sont analysés
- Scalaires : le nom du scalaire est analysé
- Tableaux : le nom, les titres et les formes LEC du tableau sont analysés
- Variables : le nom de la variable est analysé

Recherche dans ... (Search in ...)

Il est possible de restreindre la recherche aux noms d'objets, aux formules ou au texte libre, ou à une combinaison de plusieurs éléments. Il suffit pour chaque cas de préciser s'il faut ou non l'analyser.

Liste résultat

La liste des objets répondant au critère peut être sauvée dans le WS de listes courant. Le nom de cette liste doit être indiqué et être un nom de liste valide. Si une liste de même nom existe, elle sera remplacée par le résultat du calcul. Le nom "_RES" est proposé par défaut.

Résultat

Dans le bas de l'écran, un champ reprend les premiers objets satisfaisant au critère de recherche.

Exemples

Pour construire la liste de toutes les variables commençant par B_ :

```
       Text            : B_*
       Whole word      : No
       Exact case      : Yes
       Objects to scan : Variables
       Search in names : Yes
              formulas : No
            Free text  : No
```

Pour construire la liste de tous les tableaux contenant la variable XXX :

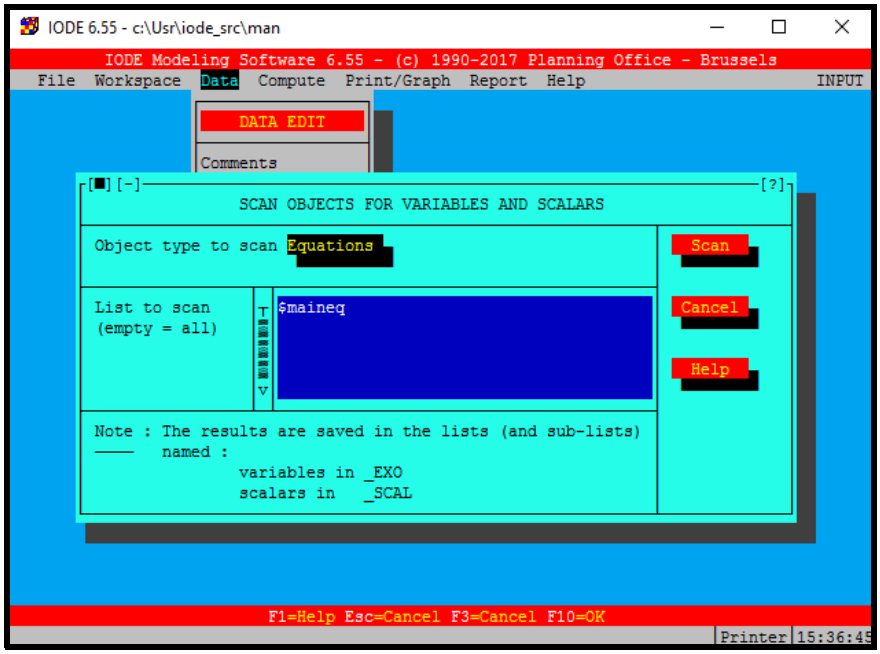
```
       Text            : XXX
       Whole word      : Yes
       Exact case      : Yes
       Objects to scan : Tables
       Search in names : No
              formulas : Yes
            Free text  : No
```

Pour construire la liste de tous les commentaires contenant le mot "export" ou "exportation" :

```
       Text            : export
       Whole word      : No
       Exact case      : No
       Objects to scan : Tables
       Search in names : No
              formulas : No
            Free text  : Yes
```

2.3.6.6 SCAN OBJECTS

FIGURE 56



Cette fonction construit deux listes à partir des formules LEC contenues dans des objets d'un type donné.

Elle permet par exemple de construire la liste des variables qui apparaissent dans des tableaux donnés, pour construire à partir de là un fichier ne contenant que les variables utiles au calcul de ces tableaux.

Elle fournit en sortie deux listes :

- _EXO : liste des noms de variables
- _SCAL : liste des noms de scalaires

Le premier champ fournit le type d'objets à scanner. Seuls les tableaux, identités et équations sont acceptés, les autres objets ne contenant pas de formule LEC.

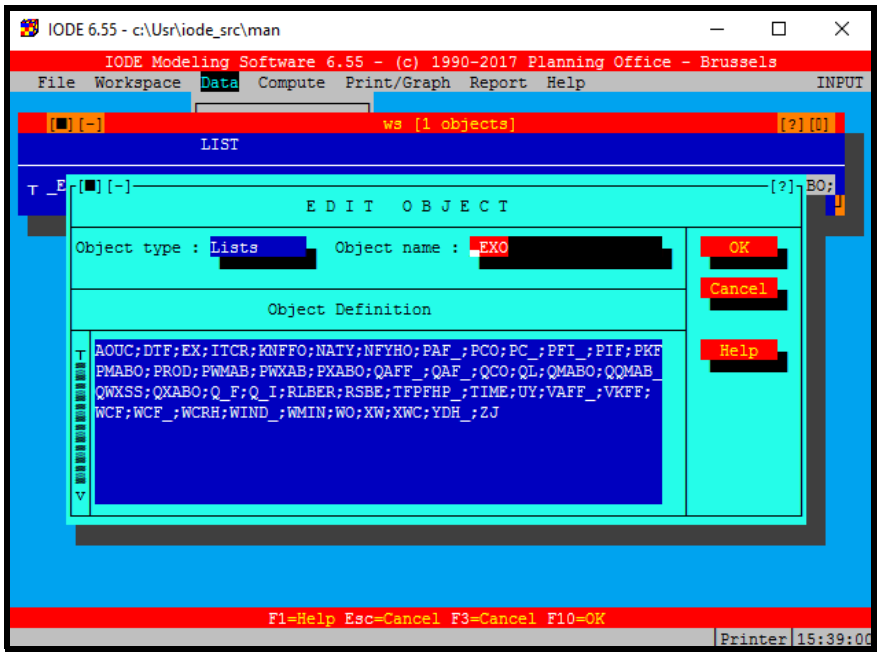
Le second champ (de type EDITOR) permet de limiter le nombre d'objets à scanner.

Par exemple, pour trouver toutes les variables apparaissant dans les équations de $maineq, on introduira :

```
Object type to scan : Equations
List to scan        : $maineq
```

Le résultat sera stoqué dans la liste _EXO du workspace courant :

FIGURE 57



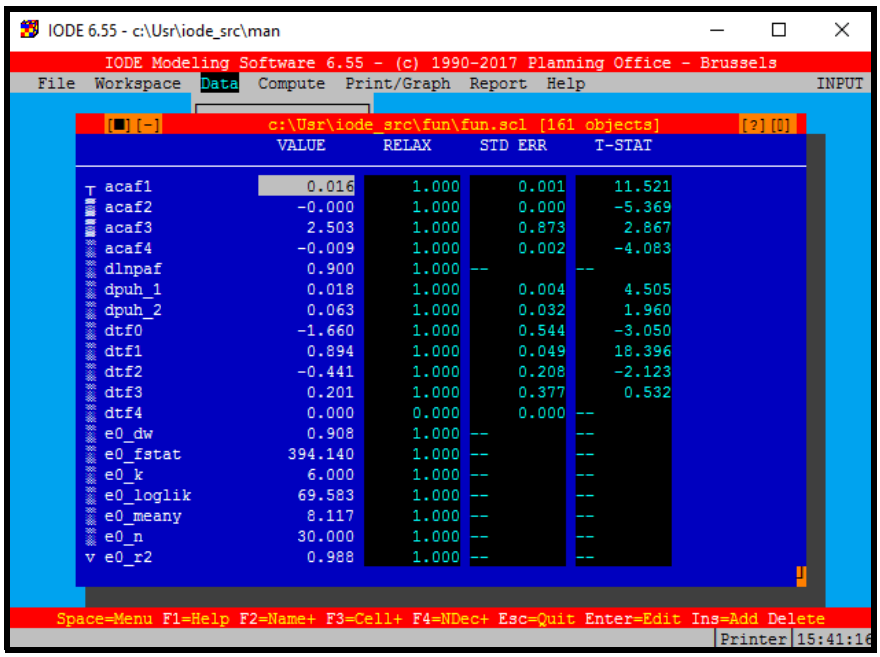
Autre exemple pour rechercher les variables apparaissant dans des tableaux:

```
Object type to scan : Tables
List to scan        : TAB1, TAB2
```

Le résultat remplacera le contenu de la liste _EXO.

2.3.7 SCALAIRES

FIGURE 58



Les scalaires sont des valeurs utilisées de plusieurs façons :

- comme coefficient à estimer dans les équations
- comme résultat secondaire d'estimation (tests statistiques, nombre d'itérations...)
- comme constantes dans les formules

Dans le cas des coefficients à estimer, outre la valeur calculée ou fixée du coefficient, trois autres valeurs pertinentes sont retenues dans l'objet scalaire : le paramètre de relaxation (permettant de "bloquer" un coefficient), le test en t et la déviation standard du coefficient. Ces deux dernières valeurs sont conservées uniquement à titre de documentation.

L'édition se fait en trois étapes:

- Sélection des objets à éditer
- Edition d'un tableau d'objets
- Edition des objets individuels

Les deux premières ont été détaillées en début de chapitre.

L'édition proprement dite des scalaires se fait en déplaçant un "curseur" dans le tableau vers la valeur à modifier, en pressant ENTER et en introduisant la nouvelle valeur dans la fenêtre affichée en avant-plan (la souris permet le même traitement : il suffit de se placer sur la cellule à éditer et de cliquer 2 fois). Seules sont modifiables la valeur du scalaire lui-même et le paramètre de relaxation de l'estimation.

Création

La création se fait en pressant INS. Une fenêtre permet alors d'introduire le nom du scalaire. En pressant F10, le nouveau scalaire est ajouté au WS avec pour valeur 0.9, comme paramètre de relaxation 1.0 et -- (Not Available) pour les tests statistiques.

Le nom d'un scalaire est toujours en minuscules pour le distinguer dans une forme LEC des noms de variables temporelles.

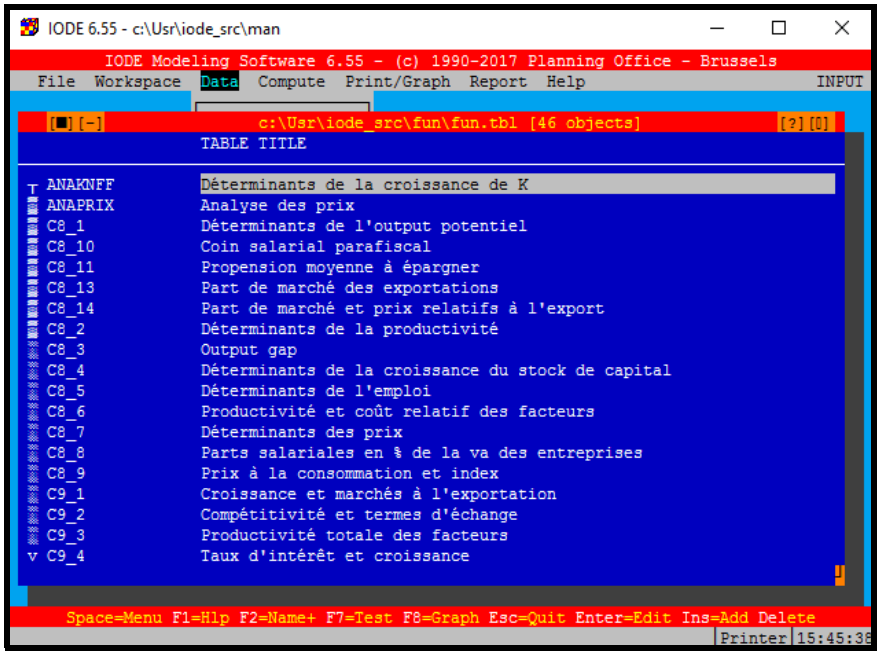
Edition

Pour modifier la valeur d'un scalaire, il faut d'abord se placer sur celui à modifier à l'aide des touches fléchées. Lorsque la case souhaitée est atteinte, la touche ENTER ouvre une fenêtre dans laquelle la valeur est affichée et peut être modifiée.

F10 ou ENTER ou une touche fléchée (haut, bas, gauche) permet de sauver la valeur modifiée et de se placer directement en édition de la case suivante (selon la touche tapée).

2.3.8 TABLES

FIGURE 59



Les tableaux sont des objets destinés à présenter des variables et des scalaires - résultats ou données - sous forme de tableaux de chiffres ou de graphiques. Ils sont définis comme un ensemble de lignes de différents types (titre, formule, lignes intercalaires, etc). Ces lignes contiennent des formules qui seront calculées lors de l'impression sur base des valeurs des séries, par exemple après une simulation. Les tableaux ne contiennent donc pas de nombres, mais bien les formules qui permettront de calculer les valeurs à imprimer dans les tableaux.

L'édition et la création des tableaux se fait en trois étapes:

- Sélection des objets à éditer (voir plus haut)
- Edition d'un tableau d'objets (voir plus haut)
- Création et édition des tableaux individuels

Les deux premières étapes sont standards et ont été détaillées en début de chapitre.

Avant de passer à la description de la phase d'édition des tableaux, il est indispensable de bien comprendre le mécanisme de fonctionnement des tableaux ainsi que ce qu'ils peuvent contenir.

DÉFINITION D'UN TABLEAU

Le tableau se présente comme un ensemble de lignes et de colonnes. Chaque ligne possède un ensemble d'attributs, dont le plus important est son type. Ce type conditionne la manière dont la ligne sera imprimée et le contenu possible de la définition de la ligne. Les autres attributs concernent le type de caractères à utiliser lors de l'impression et le cadrage de la ligne. Le tableau contient en outre des attributs graphiques qui concernent le layout du graphique qui sera éventuellement construit avec ce tableau.

Les types de lignes suivants sont définis :

- **TITLE** : il s'agit d'une ligne de titre. Le texte encodé dans la première colonne s'étendra lors de l'impression à travers toute la largeur du tableau. Une forme LEC dans ce type de ligne n'a pas de sens et est rejetée par l'éditeur.
- **CELL** : il s'agit d'une ligne "normale" du tableau : la première colonne contient un titre de ligne, la seconde une forme LEC qui, une fois calculée pour les périodes demandées, fournira les valeurs des colonnes 2 et suivantes du tableau. En fait, les deux colonnes peuvent être indifféremment du texte ou une forme LEC, mais l'intérêt de placer une forme LEC dans la première colonne n'est pas évident. Par contre, on peut placer deux titres dans les deux colonnes, ou à tout le moins laisser vide la seconde colonne pour marquer une séparation dans le tableau.
- **LINE** : ce type de ligne ne peut contenir ni texte, ni forme LEC : il ne s'agit que d'une ligne de séparation au milieu du tableau
- **FILES** : cette ligne contiendra, lors de l'impression, le nom des fichiers contenant les données imprimées s'étendant à travers toute la largeur du tableau. Si plusieurs fichiers sont imprimés en comparaison, cette ligne est automatiquement multipliée. Aucune donnée (texte ou LEC) ne peut être encodée dans les colonnes de ce type de ligne.
- **MODE** : il s'agit d'une ligne qui dépend des opérations sur les périodes effectuées lors de l'impression du tableau. Par exemple, si une colonne est imprimée en taux de croissance, un report sera indiqué dans la ligne correspondant à MODE. Aucune donnée ne peut être encodée dans les colonnes de ce type de ligne.
- **DATE** : une ligne de ce type contiendra la date d'impression du tableau.

Exemple

Le tableau suivant est le résultat d'un calcul basé sur 2 séries A et B.

```
                              Titre du tableau
  ------------------------------------------------------------------------
                  1970      1971      1972      1973      1974      1975

   Série A         0.00      1.00      2.00      3.00      4.00      5.00
   Série B        70.00     71.00     72.00     73.00     74.00     75.00
   Série A + B    70.00     72.00     74.00     76.00     78.00     80.00
  ------------------------------------------------------------------------
   [1] : ws.var
   03/05/92
```

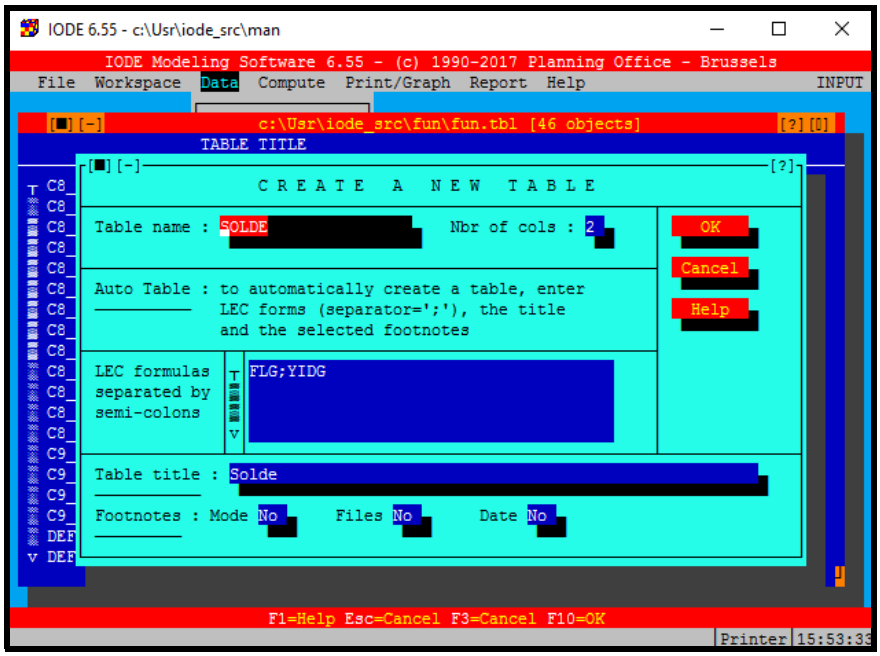
L'objet tableau ayant produit ce tableau est défini comme suit :

```
   ------------------------------------------------
   Type de ligne  |   Colonne 1          | Colonne 2
   ---------------|----------------------|-------------
   [TITLE]        |  "Titre du tableau"  |
   [LINE]         |                      |
   [CELL]         |                      |  "#t"
   [TITLE]        |                      |
   [CELL]         |  "Série A"           |  A
   [CELL]         |  "Série B"           |  B
   [CELL]         |  "Série A + B"       |  A+B
   [LINE]         |                      |
   [MODE]         |                      |
   [FILES]        |                      |
   [DATE]         |                      |
   ------------------------------------------------
```

CRÉATION D'UN TABLEAU

Pour créer un nouveau tableau, il suffit de presser la touche INS. Une fenêtre permet de définir rapidement un nouveau tableau:

FIGURE 60



Le premier champ permet d'encoder le nom du tableau à créer.

Le second champ définit le nombre de "colonnes" du tableau. Par défaut, le nombre de colonnes vaut 2, ce qui signifie qu'il y a une colonne de titres (la première) et une colonne de formules (la seconde). Cette dernière colonne sera répétée pour toutes les périodes à calculer lors de l'exploitation du tableau par le programme d'impression ou de graphique.

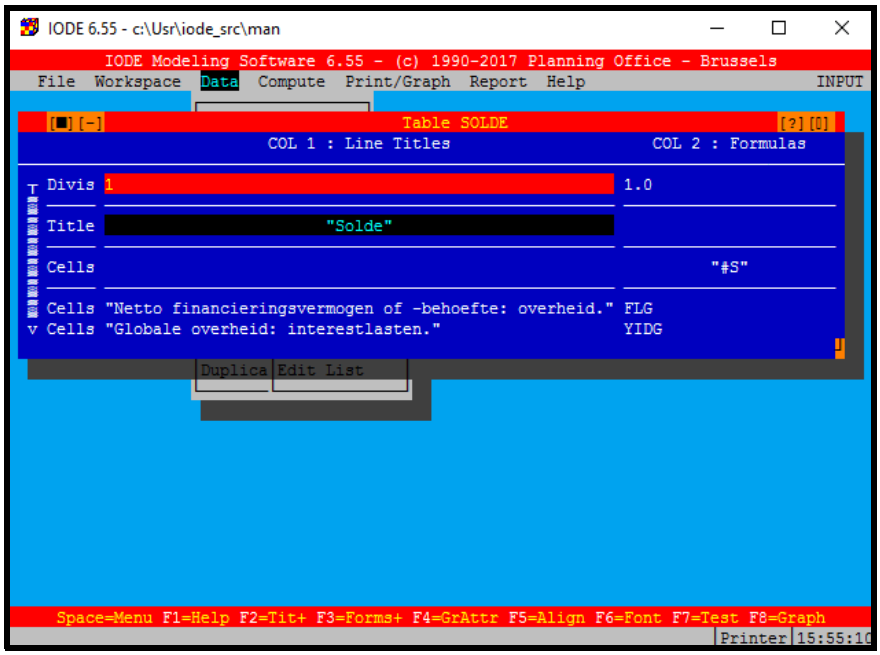
Le troisième champ est du type EDITOR. Il permet de construire un tableau de façon automatique. On introduit dans ce champ les formes LEC qui constitueront les lignes du tableau. Ces formes LEC sont obligatoirement séparées par un point-virgule (p. ex. X; Y; X+Y) ou par un saut de ligne. La formule LEC sera recopiée dans les colonnes du tableau, sous forme de titre dans la première colonne et sous forme de LEC dans les colonnes suivantes.

Dans le bas de la page de configuration du tableau, on donne le titre général du tableau (placé par défaut en tête de tableau) et on précise si des lignes de type MODE, FILES et DATE devront figurer en footnote.

Une fois le tableau créé (F10), on entre en édition d'un tableau. Le nouveau tableau n'est pas vide : il contient déjà un certain nombre de lignes standards ainsi que les informations de type automatique définies ci-dessus.

EDITION D'UN TABLEAU

FIGURE 61



Pour éditer un tableau existant, il suffit de se placer sur le tableau souhaité et de presser la touche ENTER. On se retrouve par ailleurs directement en édition lors de la création d'un nouveau tableau.

Une fois le tableau affiché à l'écran, on peut s'y déplacer à l'aide des touches fléchées (Up, Down, PgUp, PgDn), de façon à choisir la ligne et la colonne à modifier ou la position dans le tableau où on veut opérer une transformation. Ensuite, en pressant une touche fonction, on opérera la modification souhaitée.

Définition des cellules

Les cellules des lignes de type TITLE et CELL peuvent contenir des textes. Les cellules des lignes de type CELL peuvent de surcroît contenir des formes LEC. Les cellules des autres types de lignes ne contiennent aucune information, car aucune impression n'en résulterait.

Pour distinguer une cellule texte d'une cellule LEC, la cellule de texte devra commencer par les doubles guillemets (qui ne seront pas imprimés). Une cellule ne débutant pas par un double guillemet sera considérée comme une forme LEC et syntaxiquement vérifiée.

Interpétation des cellules de texte

Les cellules de texte d'un tableau peuvent contenir des éléments variables qui s'adaptent en fonction du sample lors de l'impression. Ces variables sont composées du caractère #, suivi d'une lettre et optionnellement d'un chiffre.

La lettre indique la valeur à reprendre : 'y' pour l'année par exemple.

Le chiffre indique la période à prendre en compte. En effet, plusieurs périodes peuvent intervenir dans le calcul de la colonne (80/79 par exemple). 80 correspondra à la période 1, et 79 à la période 2. Le chiffre peut donc prendre les valeurs 1 et 2.

Pour certaines variables, le chiffre n'a pas de sens : #O indique l'opération effectuée sur les années (taux de croissance par exemple) et ne doit pas être suivie de 1 ou de 2.

Si une des variables n'est pas suivie d'un chiffre, celui-ci vaut 1.

Les éléments constitutifs définis sont les suivants :

Syntaxe	Signification		Exemple
#y1 ou #y2	année de la colonne		80
#Y1 ou #Y2	année de la colonne		1980
#p	périodicité		q
#P	périodicité		Q
#M1 ou #M2	sous-période	Mensuel	Février
		Trimestriel	IV
		Annuel	vide
		Autre	52
#m1 ou #m2	sous-période	Mensuel	Fév
		Trimestriel	iv
		Annuel	vide
		Autre	52
#r1 ou #r2	sous-période	Mensuel	ix
		Trimestriel	iv
		Annuel	vide
		Autre	52
#R1 ou #R2	sous-période	Mensuel	IX
		Trimestriel	IV
		Annuel	vide
		Autre	52
#n1 ou #n2	sous-période		7
#N1 ou #N2	sous-période		11
#o	opération sur les périodes		/
#O	opération sur les périodes		Growth Rates
#F ou #F	fichiers utilisés		[1-2]

Pour faciliter l'écriture, les macros suivantes sont définies :

```
#t équivaut à #y1#P1#n1#o1#y2#P2#n2
#T     "      #Y1#P1#n1#o1#Y2#P2#n2
#S     "      #T#F
#s     "      #t#f
```

Selon le sample utilisé, certaines variables sont vides ou non : s'il n'y a pas d'opération sur les périodes (taux de croissance), toutes les variables suivies de 2 (#Y2) restent vides, de même que l'opération #O et #o.

Si un seul fichier intervient dans le calcul du tableau, #f et #F restent vides.

Si la périodicité est annuelle, les variables #P, #N, #M, #R restent vides, qu'elles soient en majuscules ou minuscules, suivies de 1 ou de 2.

Notez que plusieurs cellules d'un même tableau peuvent contenir ces éléments variables et que ceux-ci ne doivent pas être tous les mêmes :

```
ligne 1 : "#Y1" l
igne 2 : "(#O)"
```

Donne comme résultat pour le sample 90/89:4

```
          1990          1991          1992          1993
      (Différences) (Différences) (Différences) (Différences)
```

Touches fonctions

- ENTER : ouvre une fenêtre avec le contenu actuel de la cellule sur laquelle se trouve le curseur. On peut modifier le contenu de cette cellule. Cependant, toutes ne sont pas éditables : les lignes de type LINE, DATE, MODE, FILES ne peuvent contenir d'informations dans leurs cellules.
- INS : ouvre une petite fenêtre permettant d'indiquer le type de la ligne à insérer et la position relative de la cellule (avant ou après la ligne courante). F10 ajoute la ligne dans le tableau.
- DEL : détruit la ligne courante.
- F2 : augmente d'un caractère la largeur visible de la première colonne.
- s+F2 : diminue d'un caractère la largeur visible de la première colonne.
- F3 : augmente d'un caractère la largeur visible des formules.
- s+F3 : diminue d'un caractère la largeur visible des formules.
- F4 : permet de fixer les attributs graphiques du tableau et de la ligne courante (voir ci-dessous).
- F5 : permet de modifier le cadrage de la cellule courante: Center, Left, ou Right.
- F6 : permet de modifier les attributs typographiques de la cellule courante: Normal, Bold, Italic, ou Bold+Italic
- F7 : visualisation test du tableau en cours de définition sur les premières périodes. Pour rendre cette opération réalisable, il est clair que tous les éléments apparaissant dans les formules du tableau (variables, scalaires) doivent être présent en workspace. En cours de visualisation du tableau, la touche F4 donne la liste des fichiers auxquels il est fait référence.
- F8 : visualisation test du tableau sous forme graphique. Pour rendre cette opération réalisable, il est clair que tous les éléments apparaissant dans les formules du tableau (variables, scalaires) doivent être présents en workspace.
- F10 : sauve le tableau et quitte son édition. Si le tableau existait auparavant, la fonction demande confirmation du remplacement de l'ancien tableau.
- ESCAPE : quitte le tableau sans sauver. Un confirmation est demandée pour éviter la perte du travail suite à une fausse manoeuvre.

La souris peut être utilisée pour faciliter la plupart des opérations de définition du tableau : sélection d'une cellule, copie/collage de parties de texte ou de formules, déplacement de la fenêtre, etc.

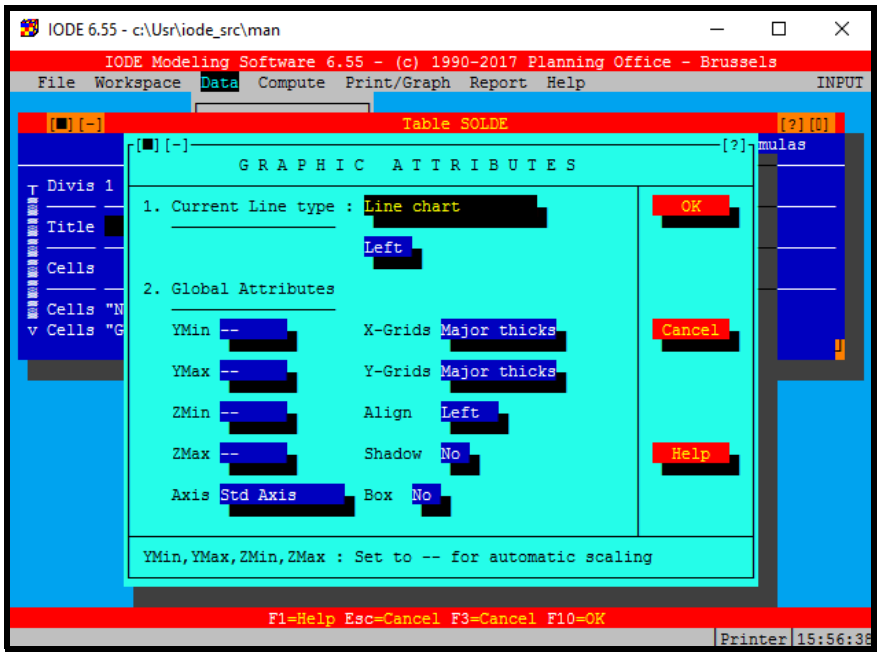
De plus, des touches fonctions semblables à celles de l'éditeur MMT sont également exploitables en édition des tableaux. L'éditeur de tableaux permet de marquer et copier des blocs de lignes, y compris entre différents tableaux.

- Ctrl-L : marque la ligne courante ou étend la sélection jusqu'à celle-ci
- Ctrl-U : supprime les marques
- Ctrl-C : copie les marques à l'endroit du curseur
- Ctrl-D : détruit les marques et place les lignes détruites dans le buffer.
- Ctrl-K : détruit la ligne courante sans demande de confirmation
- Ctrl-Y : place les marques dans un buffer pour une copie dans un autre tableau. Ctrl-C et Ctrl-D place également les lignes marquées dans le buffer. Le contenudu buffer est remplacé à chaque fois.
- Ctrl-P : copie le buffer à l'endroit du curseur

Attributs graphiques

La touche F4 permet de fixer les attributs graphiques du tableau et de la ligne courante. La page suivante est affichée :

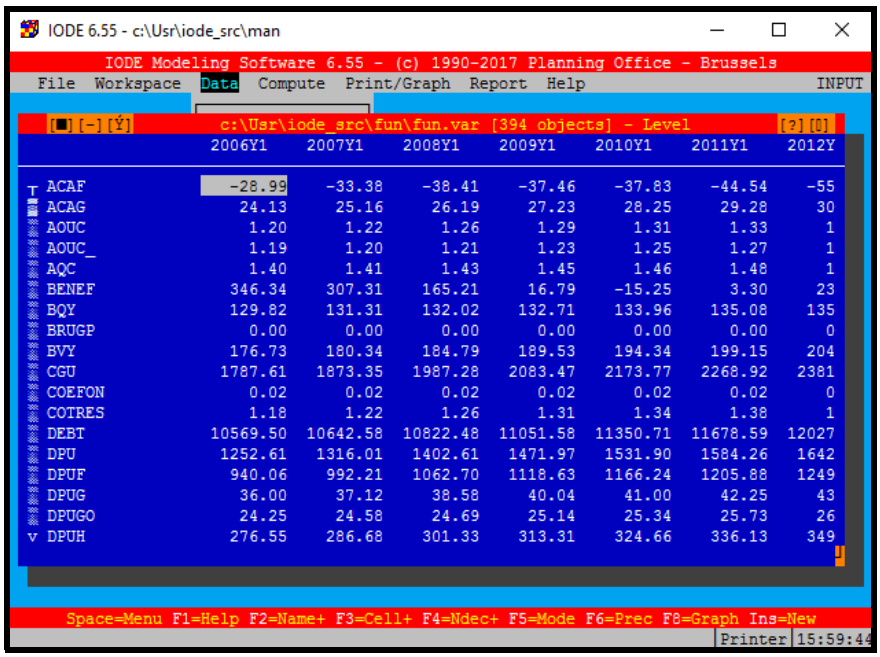
FIGURE 62



- **Current line** type offre le choix entre 4 types de courbes pour la ligne courante du tableau : Line Chart, Scatter Chart (suite de symboles), Bar Chart et Scatter Line (ligne enrichie de symboles).
- **Left ou Right** indique si l'axe des ordonnées doit être à gauche ou à droite du graphique.
- **YMin et Ymax** permettent de fixer le minimum et le maximum sur l'axe des Y. Si les données sortent de ces valeurs, l'échelle de l'axe s'adapte aux données. La valeur -- peut être fixée pour YMin et/ou YMax : dans ce cas, le programme graphique calculera une valeur d'échelle optimale.
- **Y-Grids et X-Grids** offrent le choix entre trois options de quadrillage du graphique en Y et en X : Major qui trace une ligne à travers tout le graphique à chaque graduation principale de l'axe, Minor qui en trace une à toutes les graduations et None qui supprime le quadrillage du graphique.
- **Axis** permet de choisir le type des axes : Std Axis (en niveau), Log Axis (échelle logarithmique), Semi-Log Axis (échelle semi-logarithmique), Percent Axis (échelle en Y en pourcentage de 0 à 100). Cette option n'est pas implémentée.
- **Box** indique si le graphique doit être encadré
- **Shadow** permet de donner un effet d'ombrage au graphique.

2.3.9 VARIABLES

FIGURE 63



Les variables sont des séries de nombres définies sur une période de temps déterminée. Ce sont donc simplement des suites de valeurs qui peuvent être visualisées et modifiées sous la forme d'un grand tableau de nombres.

L'édition se fait en trois étapes:

- Sélection des objets à éditer (voir plus haut)
- Edition d'un tableau d'objets (voir plus haut)
- Edition des valeurs individuelles

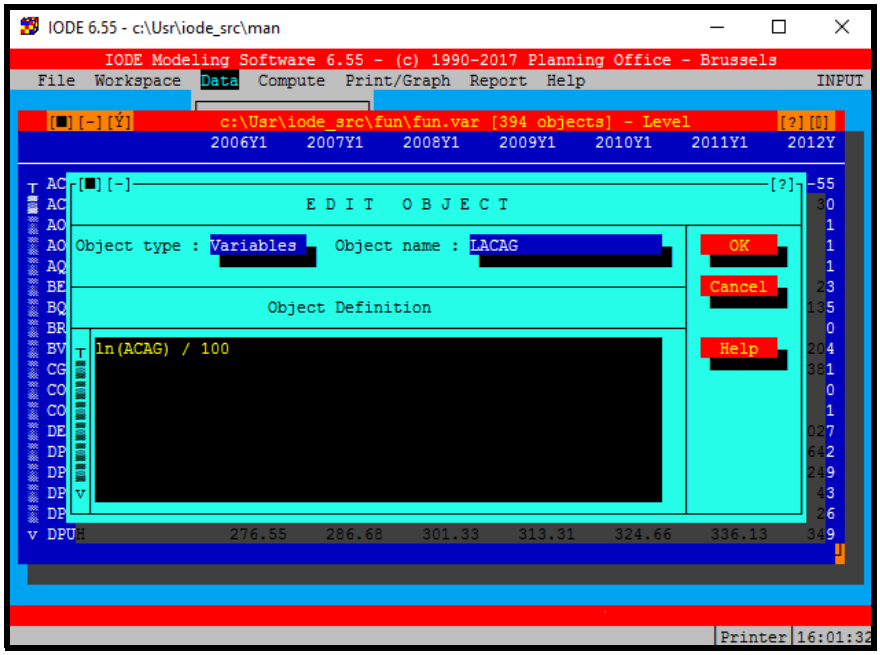
Les deux premières étapes ont été détaillées en début de chapitre.

Il est également possible d'éditer des séries en les présentant sous la forme de tableaux définis par ailleurs: c'est l'objet de l'option Edit Tables.

L'édition proprement dite des séries se fait en déplaçant un "curseur" dans le tableau vers la valeur à modifier, en pressant ENTER et en introduisant la nouvelle valeur dans la fenêtre affichée en avant-plan.

Création

FIGURE 64



La création d'une nouvelle variable se fait en pressant INS. Une fenêtre permet alors d'introduire le nom de la variable ainsi qu'une expression en LEC indiquant les valeurs initiales à donner à la variable.

La formule LEC peut contenir n'importe quelle expression, mais les variables et scalaires nécessaires à son calcul doivent impérativement être présents en workspace pour que le calcul soit possible.

Une forme LEC constante comme "ln(3)" est également valable. En cas d'erreur de syntaxe, la création n'a pas lieu et le programme revient en édition de la formule afin de permettre d'y apporter les corrections nécessaires.

Par exemple :

```
1970 + t
ln(X)
c1 * c2 + Y
2
```

En pressant F10, la nouvelle variable est ajoutée au WS avec pour valeurs celles résultant du calcul de la formule LEC éventuelle. En l'absence de forme LEC, la valeur NA est donnée à tous les éléments de la série de nombres.

La formule LEC n'est pas conservée dans la définition de la variable : après son calcul, elle est perdue.

Edition

Pour modifier la valeur d'une variable à une période de temps, il faut d'abord se placer sur la bonne série à la bonne période. Cela se fait à l'aide des touches fléchées (UP, DOWN, PGUP, PGDN) ou en pressant la première lettre du nom de la série.

Lorsque la case souhaitée est atteinte, la touche ENTER ouvre une fenêtre dans laquelle la valeur est affichée et peut être modifiée.

Le fait de presser ENTER ou une des touches fléchées sauve la valeur modifiée et remet le programme directement en édition de la case suivante (au sens de la touche fléchée utilisée).

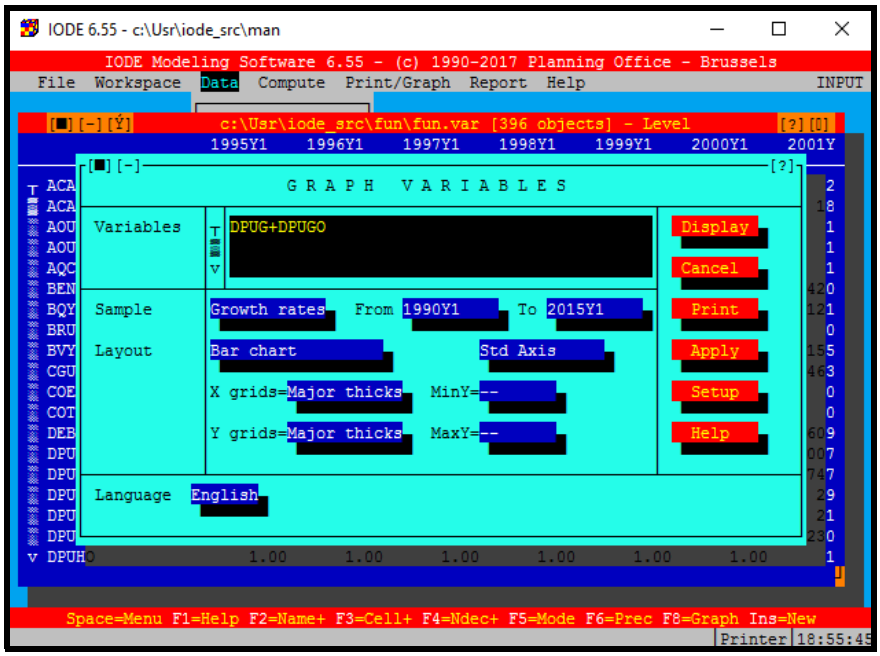
Touches fonction

Les touches fonctions standards sont définies pour le tableau de séries. En plus :

- la touche TAB fait défiler d'un écran vers la droite
- la touche s-TAB fait défiler d'un écran vers la gauche
- F2 et shift-F2 : respectivement augmente et diminue la largeur de la colonne reprenant le nom des variables
- F3 et shift-F3 : respectivement augmente et diminue la largeur des colonnes de valeurs
- F4 et shift-F4 : respectivement augmente et diminue le nombre de décimales affichées
- F5 : le mode de visualisation passe de "Level" (valeur réelle), en mode "Difference" (différence période courante - période précédente) et en mode "Growth Rate" (taux de croissance) à chaque fois que la touche F5 est pressée. Le mode courant est indiqué dans la ligne de titre de la fenêtre.
- F8 : affiche le graphique de la série sur laquelle le curseur se trouve avec les propriétés du plus récent appel à la fonction Shift-F8
- Shift-F8 permet de grouper plusieurs séries sur le même graphique. De plus, cette fonction permet de modifier les propriétés (Sample, type, ...) des futurs graphiques affichés à l'aide de F8.

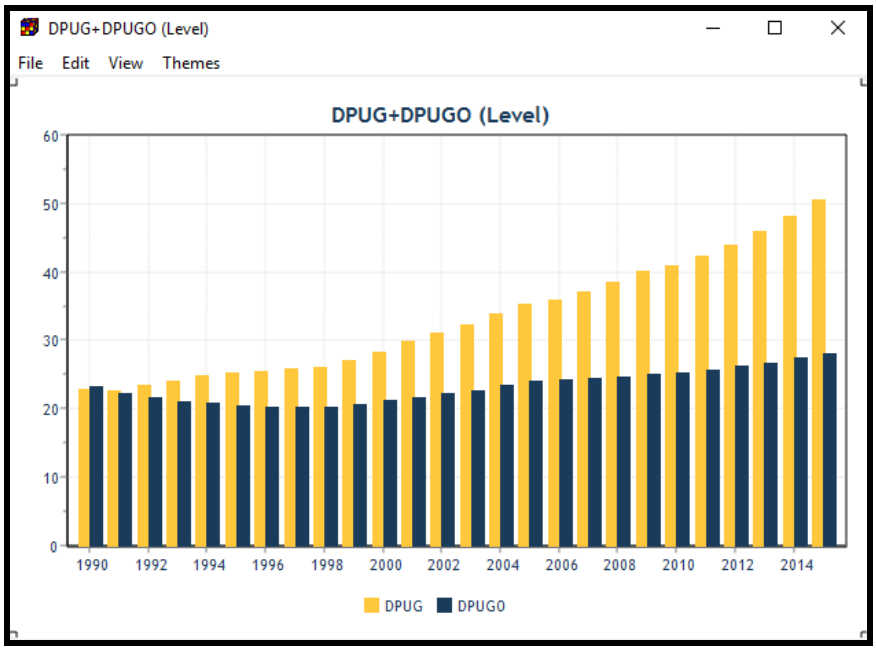
Par exemple pour visualiser deux variables sur un même graphique sans avoir à contruire un tableau :

FIGURE 65



Donnecomme résultat :

FIGURE 66



Rappelons les autres touches principales :

- Ctrl-R : Rotation du tableau en inversant lignes et colonnes
- Ctrl-O : déplacement de la fenêtre dans l'écran
- Ctrl-Z : modification de la taille de la fenêtre
- Ctrl-X : maximisation de la fenêtre
- lettre : positionne sur la variable suivante dont le nom commence par la lettre pressée
- F1 : accès au manuel
- espace : accès au menu de la fenêtre

La souris peut être utilisée pour faciliter la plupart des opérations : sélection d'une cellule, accès au menu, copie d'une formule, déplacement et agrandissement de la fenêtre, etc.

2.3.9.1 Edit Tables

Cette fonction, accessible directement à partir de l'édition des tableaux ou à partir du meu Print/ View Tables permet la mise à jour des séries décrites par les formules.

En pressant ENTER sur une cellule, la valeur courante peut être modifiée par l'utilisateur. Plusieurs cas sont possibles, certains rejettant la modification :

- Si la colonne contient une opération (taux de croissance, différence de fichiers, ...), la mise à jour est rejettée
- Si la formule définissant la cellule commence par `0+...`, cela signifie que cette cellule est protégée. L'édition est rejettée
- Si la formule fait apparaître plusieurs variables, la première seule est modifiée
- Si la formule ne fait pas apparaître de variable, la cellule n'est pas éditable
- Si la formule n'est pas inversible par rapport à la première variable qui y apparaît, la méthode de la sécante (non linéaire) est appliquée pour tenter de résoudre l'équation. Si celle-ci n'aboutit pas, la valeur reste inchangée.

Chaque changement de valeur a un effet dans le WS de séries. Le tableaux est recalculé à chaque modification, un peu comme dans une feuille de calcul.

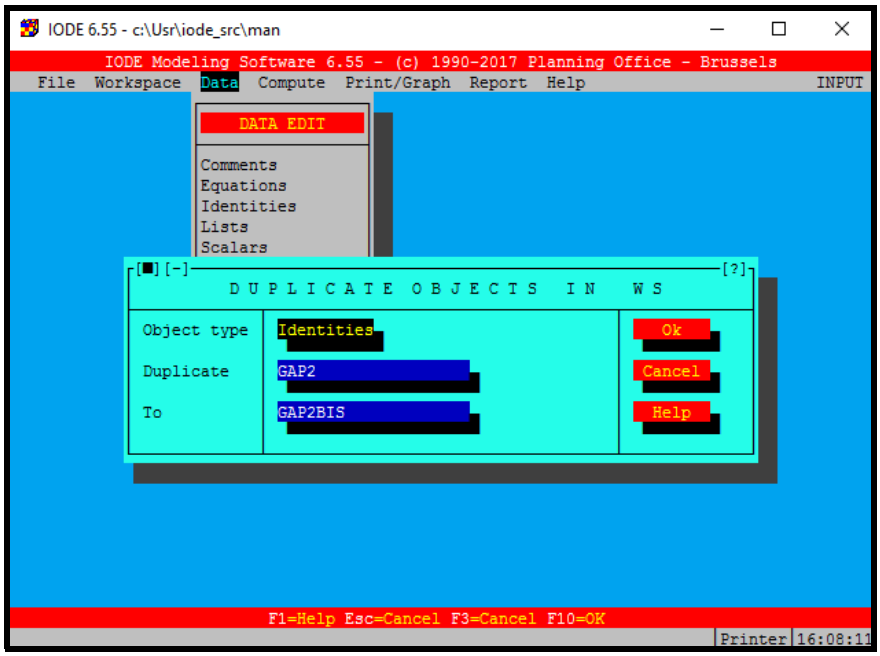
2.3.10 Duplicate objects

Les objets définis dans les WS courants peuvent être copiés dans des objets du même type mais d'un nom différent. Pour certains objets, il existe une méthode simple pour effectuer cette opération : dans l'écran de définition de l'objet à duplicier, il suffit de changer le nom de l'objet et de le sauver. Un nouvel objet ayant les mêmes caractéristiques que l'ancien est ainsi créé.

La fonction décrite ici permet la même opération. Elle fonctionne pour tous les objets sauf pour les équations. En effet, une équation porte le nom de sa variable endogène. Changer son nom change donc le nom de l'endogène, ce qui en général n'a pas de sens.

L'écran se présente comme suit:

FIGURE 67



Le type d'objet à duplicier peut être sélectionné en se plaçant sur le champ indiquant le type et en pressant ENTER: un menu reprenant tous les types d'objets permet d'effectuer la sélection.

Les deux autres champs contiennent respectivement le nom de l'objet à copier et le nom du nouvel objet.

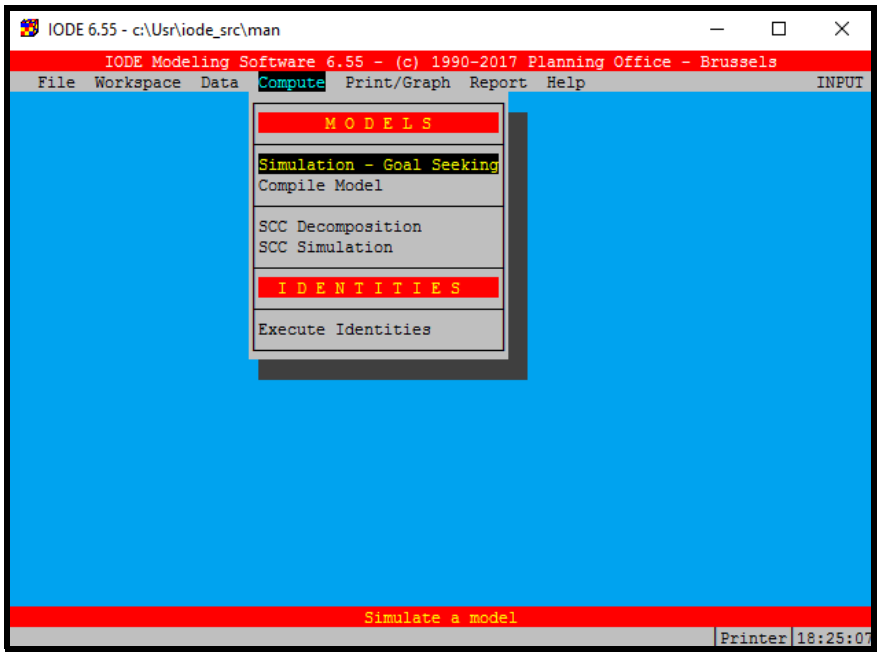
Touches fonctions

- F10 duplicie l'objet
- ESCAPE quitte l'écran en abandonnant l'opération en cours

Il n'y a pas de vérification sur l'existence du nouveau nom. En cas d'existence, l'ancien objet est remplacé par la copie.

2.4 COMPUTE

FIGURE 68

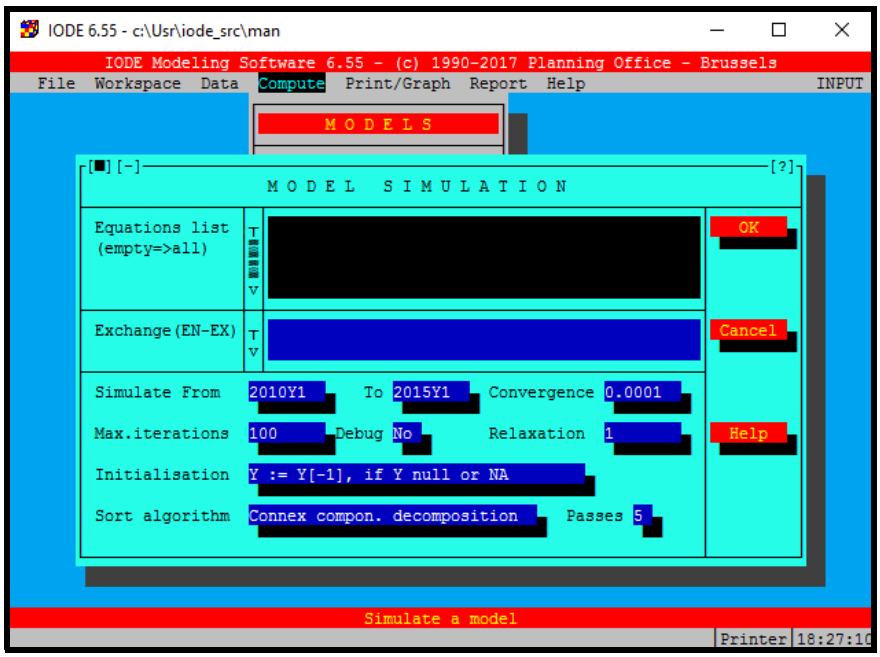


Ce menu comprend les fonctions principales de calcul :

- Simulation - Goal Seeking : simulation d'un modèle et recherche d'objectifs
- Compile Model : recompile une liste d'équations
- SCC Decomposition : calcule les composantes fortement connexes du modèle et les réordonne.
- SCC Simulation : simule un modèle préalablement décomposé en composantes fortement connexes.
- Execute Identities : calcul d'un groupe d'identités de construction et création ou modification des séries associées

2.4.1 SIMULATION D'UN MODÈLE

FIGURE 69



DÉFINITION D'UN MODÈLE

Un modèle, au sens de IODE, est simplement une liste d'équations. Aucune autre construction n'est nécessaire : de cette façon, pour modifier un modèle, il suffit de modifier la liste qui le définit. Par exemple, si un modèle est découpé logiquement en 5 blocs, on définira 5 listes d'équations : le modèle sera défini par la liste contenant les 5 "sous-listes".

```
Définition des listes
---------------------
BLOC1 : A, B, C, D
BLOC2 : X, Y
BLOC3 : C1, C2, C3, C4
BLOC4 : X_A1, X_A2, X_A3
BLOC5 : R1, R2, R3, S_12

MODSIM : $BLOC1, $BLOC2, $BLOC3, $BLOC4, $BLOC5
```

Pour un modèle de quelques équations, il n'est pas indispensable de définir des listes de noms : les noms des équations peuvent être fournis in extenso dans l'écran de simulation.

Modifier un modèle (supprimer ou ajouter un bloc par exemple), revient donc à changer la liste des équations définissant le modèle.

Comme la variable endogène d'une équation porte le nom de l'équation, il n'est pas possible de placer dans un modèle deux équations ayant même variable endogène.

Pour simuler un modèle, il faut que toutes les équations du modèle soient présentes dans le WS d'équations, que toutes les variables et tous les scalaires utilisés dans les équations du modèle soient définis dans les WS de variables et de scalaires. Les valeurs des exogènes et des scalaires ne peuvent être NA (Not Available) sur la période utile.

DÉFINITION DU MODÈLE, DE LA PÉRIODE ET DES PARAMÈTRES

La première partie de l'écran permet de spécifier la liste des équations qui définissent le modèle, la partie inférieure de l'écran définit les paramètres techniques de la simulation.

Liste d'équations

En général, on définira une liste d'équations (avec des sous-listes) plutôt que d'introduire dans l'écran tous les noms des équations.

L'ordre d'introduction des équations dans la liste peut influencer le comportement de l'algorithme de simulation. En effet, comme il s'agit d'un algorithme de type Gauss-Seidel, les informations calculées sont directement utilisées dans la suite du calcul. Si X dépend de Y, il vaut donc mieux placer Y avant X dans la liste d'équations.

Si toutes les équations du WS font partie du modèle, on peut laisser la liste vide.

Exchange Endo-Exo

Ce champ permet de spécifier un ensemble de couples endogène-exogène afin de réaliser une recherche d'objectif (Goal seeking). Pour chaque paire, le statut des variables est échangé : l'endogène devient exogène et inversement. Cela permet de faire passer le modèle par des valeurs connues des endogènes et de déduire les valeurs nécessaires des exogènes associées. Les exogènes prennent la valeur calculée sur toute la fin de la période du WS.

Période de simulation

La période de simulation est indiquée par les deux périodes qui en définissent les limites. La période indiquée doit faire partie de la période de définition du WS de variables. Les variables exogènes (non calculées par le modèle) doivent être définies (non NA). Il en va de même pour les scalaires et les variables endogènes "laggées".

Convergence

Indique le seuil de convergence en-deçà duquel la simulation se termine positivement pour la période courante. En d'autres termes, pour une période donnée, lorsque les différences relatives entre deux itérations successives (k et k-1)

$$\frac{||Y[k] - Y[k-1]||}{||Y[k]||}$$

sont inférieures à ce seuil pour toutes les variables endogènes, on considère que le modèle a convergé.

Maximum number of iterations

Si après le nombre d'itérations indiqué, le modèle n'a pas convergé, le processus s'arrête sur un message d'erreur. Ce paramètre permet d'éviter les bouclages.

Relaxation parameter

Ce paramètre doit être compris entre 0 et 1. Il permet de modifier le comportement de l'algorithme de la façon suivante : la variable X, calculée par l'équation du modèle associée, n'est pas conservée telle quelle pour les itérations suivantes, mais remplacée par

```
X[k] = X' * lambda + X[k-1] * (1 - lambda)

   où X'        est le résultat du calcul de l'équation
      lambda    est le paramètre de relaxation
      X[k - 1]  est le résultat du calcul à l'itération précédente
      X[k]      est la nouvelle valeur
```

On constate que si lambda vaut 1, X[k] est le résultat du calcul de l'équation sans modification. S'il vaut 0, X ne changera jamais. Entre les deux, une partie de la valeur de l'itération précédente est conservée dans la nouvelle valeur de X, évitant ainsi un comportement parfois trop "brutal" du modèle.

Ce paramètre est à utiliser en fixant une valeur entre 0.5 et 1 lorsqu'un modèle ne converge pas ou qu'il converge trop lentement.

Debug

L'option debug génère automatiquement des listes comprenant des équations pré- et post-récursives : **_PRE**, **_INTER** et **_POST**.

Le fichier de trace (simul.dbg) comprenant la liste des équations pour lesquelles une sous-itération (Newton-Raphson) a été nécessaire et le nombre de sous-itérations peut être généré via la commande de rapport correspondant.

Sorting algorithm

La liste des équations qui définit le modèle peut être utilement réorganisée de façon à accélérer la simulation. Deux méthodes sont utilisables séparément ou en combinaison. Toutes deux font appel à des algorithmes de théorie des graphes.

- Décomposition en composantes fortement connexes : il s'agit de découper le modèle en blocs non interdépendants. Cette méthode permet de limiter le nombre d'équations à calculer en se basant sur les dépendances des équations les unes par rapport aux autres. Dans l'exemple suivant, les blocs 1 et 3 sont non interdépendants, ce qui signifie qu'un seul calcul est nécessaire par période. Ils n'entrent pas dans la boucle de simulation. Le paramètre de relaxation n'est pas utilisé dans ces blocs. Le bloc 2 est la partie interdépendante du modèle. Il faut itérer sur ce bloc et sur lui seul. On réduit ainsi le nombre d'équations à résoudre à 3 au lieu des 6 qui forment en réalité le modèle. La décomposition peut donner lieu à plus de trois blocs :

```
Soient les équations suivantes :

A = f1(B,C,D)
B = f2(C,A)
C = f3(B,X)
D = f4(X,Y)
X = f5(Y)
Z = f6(A,B,C,X)

Les blocs suivants sont construits (dans cet ordre)

     1 :   X = f5(Y)        ne dépend que de Y exogène
           D = f4(X,Y)      dépend de X

     2 :   A = f1(B,C,D)    boucle avec B et C
           B = f2(C,A)      boucle avec A et C
           C = f3(B,X)      boucle avec B, donc avec A

     3 :   Z = f6(A,B,C,X) n'est utilisé par aucune équation
```

- Pseudo-triangulation : cette méthode réorganise les équations à l'aide d'un algorithme spécifique. En gros, les dépendances sont analysées et les équations sont organisées en conséquence : si B dépend de A, B est placé après A, etc. Plusieurs passages sont en général utiles pour obtenir un résultat satisfaisant. Le nombre optimal de passages est fonction du modèle et laissé au choix de l'utilisateur.

Une combinaison des deux méthodes est possible. Si l'ordre de la liste est considéré comme optimal, on peut supprimer toute méthode de réordonnancement (Order = 'None') ou se contenter de la méthode des composantes connexes.

Pour conserver l'ordre d'une liste donnée, il faut fournir cette liste ET préciser None comme algorithme de réordonnancement.

Pseudo-triangulation passes

Il s'agit du nombre de passages à effectuer à l'aide de l'algorithme de pseudo-triangulation.

Starting values

Au début de chaque période à simuler, une valeur de départ doit être choisie pour chaque variable endogène. Cette valeur peut être :

- Y := Y[-1], if Y null or NA : chaque endogène nulle ou NA au départ prend la valeur de la période précédente
- Y := Y[-1], always : chaque endogène prend au départ la valeur de la période précédente
- Y := extrapolation, if Y null or NA : chaque endogène nulle ou NA prend comme valeur une extrapolation linéaire des deux périodes précédentes
- Y := extrapolation, always : chaque endogène prend comme valeur une extrapolation linéaire des deux périodes précédentes, qu'elle soit nulle ou non au départ
- Y unchanged : les endogènes ne sont pas initialisées. Elles gardent leur valeur qu'elle soit nulle ou non.
- Y := Y[-1], if Y = NA : chaque valeur NA prend la valeur de la période précédente
- Y := extrapolation, if Y = NA : chaque valeur NA prend comme valeur une extrapolation linéaire des deux périodes précédentes

Touches fonctions

- **ESCAPE** quitte la page sans effectuer la simulation
- **F10** lance la simulation.

SIMULATION

Lorsque la touche F10 est pressée dans l'écran de simulation, le processus mathématique de simulation est lancé.

A la fin de chaque itération, la différence entre les deux itérations est indiquée. Le processus continue jusqu'à convergence, jusqu'au dépassement du nombre maximum d'itérations ou jusqu'à une erreur de calcul éventuelle.

Si le modèle ne converge pas ou s'arrête sur une erreur de calcul, les valeurs calculées pour les variables endogènes sont conservées. Elles peuvent donc être consultées pour essayer de déterminer l'origine du problème.

Sous-algorithme de Newton

Lorsqu'une équation n'est pas définie explicitement par rapport à son endogène ou lorsque celle-ci apparaît plusieurs fois dans l'équation, l'algorithme de simulation essaie de résoudre cette équation par une méthode de Newton. Si cette méthode ne fournit pas de résultat, une méthode de type sécante recherche à son tour une solution pour l'équation.

Il n'est cependant pas garanti qu'une solution soit obtenue dans tous les cas. Les fonctions non continues (singularités) comme

```
X := a + b * 1 / (X + 1)
```

peuvent être impossibles à résoudre autour de leur point singulier. Dans le cas de ce type de problème, la seule solution est de modifier la forme de l'équation :

```
(X - a) * (X + 1) := b
```

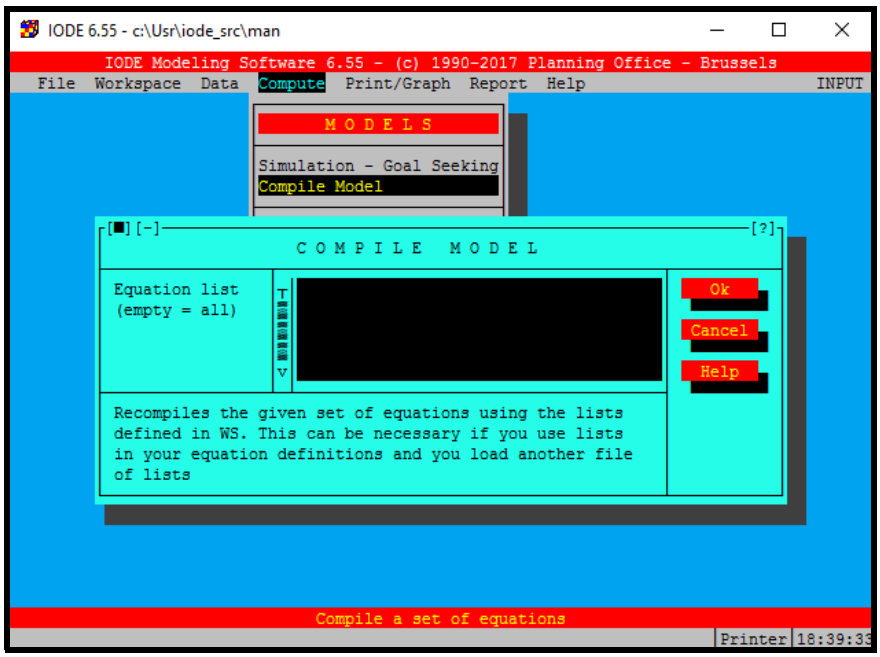
INTERRUPTION DE LA SIMULATION

La simulation peut être interrompue par l'utilisateur en pressant la touche ESCAPE. L'arrêt ne se produira qu'à la fin d'une itération. En fonction de la taille du modèle et de la machine sur laquelle on travaille, cela peut donc prendre jusqu'à quelques secondes.

Un message permet à l'utilisateur de continuer ou de stopper la simulation. En cas d'arrêt, les valeurs calculées des séries sont conservées dans les variables.

2.4.2 COMPILE MODEL

FIGURE 70



Cette fonction recompile une liste d'équations ou toutes les équations si aucune liste n'est spécifiée.

Pour comprendre l'utilité éventuelle de ce module, il faut se rappeler que les équations stockées en WS (en mémoire ou sur disque) sont composées de plusieurs éléments : le forme LEC, les inputs d'estimation et - ce qui nous occupe plus particulièrement ici - la forme compilée de l'équation.

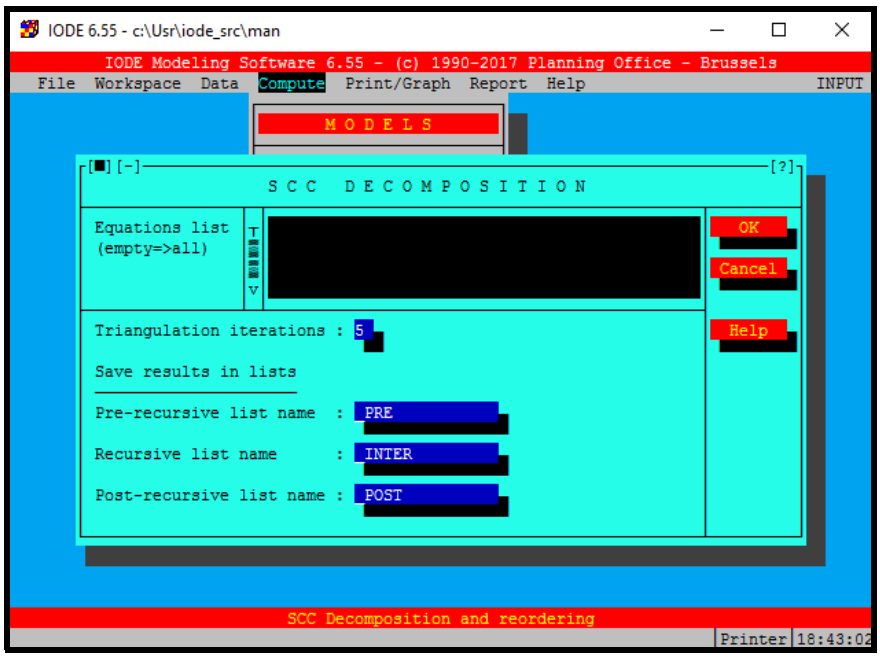
D'autre part, les formes LEC peuvent contenir des macros ($liste). Si une liste apparaissant dans une équation est modifiée, l'équation compilée ne correspond donc plus à la forme LEC. Rien en effet (sauf à le vérifier à tout moment) ne permet de savoir qu'une liste donnée appartient à la définition d'une équation.

Ce cas peut se présenter lorsqu'un nouveau fichier de listes est chargé en mémoire, lorsqu'une liste est modifiée manuellement ou copiée du disque, ou encore lorsqu'un fichier d'équations est chargé en mémoire.

En cas de doute, il est facile de "synchroniser" les équations et leurs macros en recompilant toutes les équations à l'aide de la fonction "Recompile Model".

2.4.3 SCC DECOMPOSITION

FIGURE 71



Cette fonction décompose le modèle en composantes fortement connexes (CFC ou SCC pour Strongly Connex Components) et le réordonnancer. Trois listes sont donc créées : équations pré-récursives, interdépendantes et postrécursives.

Lors du réordonnancement du modèle, le nombre d'itérations de triangulation (tri) pour le block interdépendant doit être spécifié. Cette valeur n'a évidemment d'effet que sur la liste des équations interdépendantes.

Ces 3 listes ne doivent être contruites qu'une seule fois pour une version donnée du modèle.

Voir également SCC Simulation pour la partie simulation..

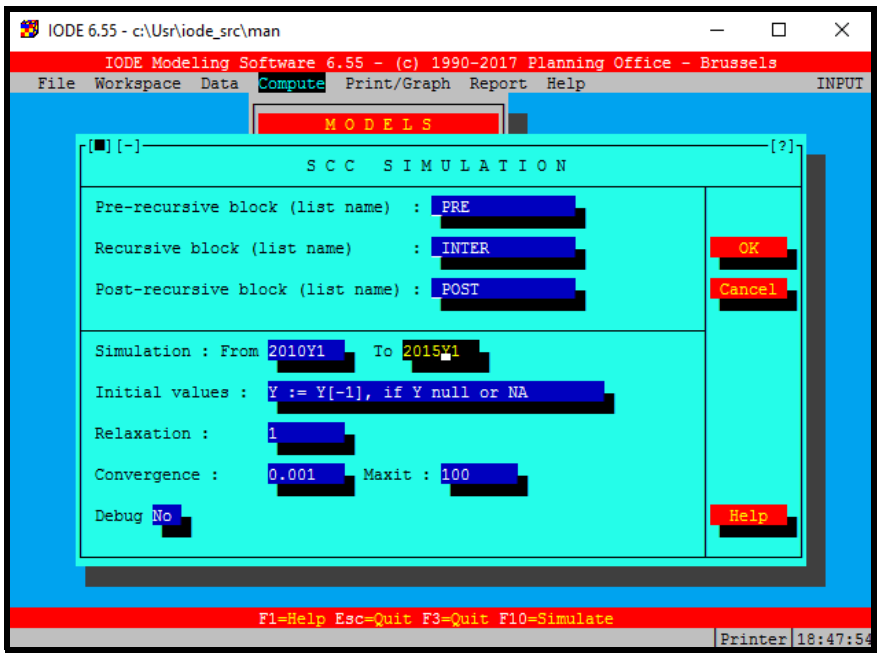
Choix du nombre de tris

Après la décompostion en CFC, le bloc interdépendant est trié pour augmenter la vitesse de la simulation. Le nombre de passage de l'algorithme de tri peut être spécifié à plusieurs endroits :

- Dans l'écran de simulation "standard" : paramètre "Passes" fixé
- Dans l'écran de calcul de la décomposition du modèle : paramètre "Triangulation Iterations"
- Comme paramètre dans la commande rapport $ModelCalcSCC
- Comme dernier paramètre dans la commande rapport $ModelSimulateParms

2.4.4 SCC SIMULATION

FIGURE 72



Cette fonction simule un modèle préalablement décomposé en composantes fortement connexes et réordonné.

Pour améliorer la performance au démarrage de la simulation, le processus de simulation a été divisé en 2 étapes. La première ne s'occupe que du réordonnancement du modèle (voir SCC Decomposition, la seconde de la simulation.

Cette seconde étape lance la simulation du modèle sur base des trois listes préalablement construites par la fonction $ModelCalcSCC (ou à la main).

Les autres paramètres sont semblables à ceux de la fonction standard de simulation.

La version rapport de la fonction est :

```
$ModelSimulateSCC from to pre inter post
```

où :

- from et to déterminent le sample de simulation
- pre, inter et post sont les listes qui définissent l'ordre d'exécution du modèle.

2.4.5 EXECUTE IDENTITIES

Une identité est une forme LEC qui indique comment une série doit être construite. Le programme d'exécution calcule toutes les identités demandées et sauve dans le WS de séries courant le

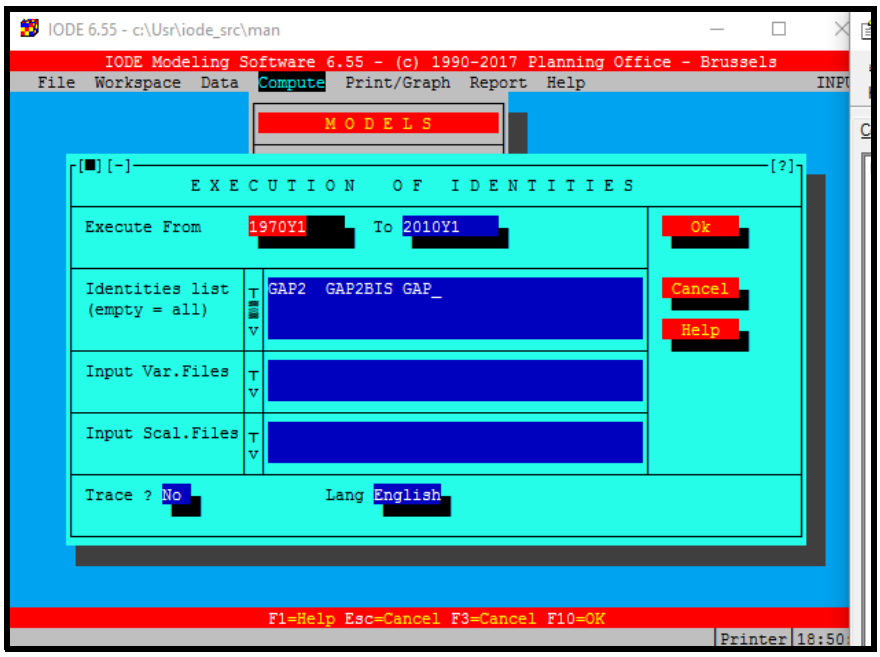
résultat des calculs.

Ces identités sont ordonnancées dans l'ordre logique d'exécution : si A dépend de B, B sera calculé avant A. L'utilisateur n'a pas à se préoccuper de cet ordonnancement. Si une boucle est détectée dans les identités, l'exécution ne peut commencer et un message est affiché.

L'écran de saisie permet de définir la période à calculer, la liste des identités à exécuter et les noms des fichiers où les objets pourront être recherchés.

De plus, le calcul peut si on le souhaite ne s'opérer que sur une sous-période.

FIGURE 73



Les touches fonctions définies dans cet écran sont :

- ESCAPE : quitte sans exécuter
- F10 : lance l'exécution des identités spécifiées

Si la liste des identités est laissée vide, toutes les identités en mémoire sont exécutées.

Si, parmi les séries ou les scalaires utilisés dans les identités, un ou plusieurs objets ne sont pas définis en WS, il faut préciser la liste des fichiers où ces informations peuvent être recherchées.

Lorsqu'une série ou un scalaire utile à l'exécution d'une identité ne se trouve pas en mémoire, les fichiers indiqués sont analysés à tour de rôle. Dès que l'objet recherché est trouvé, il est amené en mémoire pour la durée du calcul et est ensuite détruit.

Si un objet - variable ou scalaire - n'est ni en WS, ni dans un des fichiers indiqués, le processus de calcul s'arrête.

L'utilisateur peut suivre la trace de la construction des identités en en spécifiant Yes dans le champ "Debug" de la page de saisie. Les informations utiles (par exemple, le fichier d'origine

d'une variable utilisée pour construire l'identité) seront imprimés à la fin du processus.

CALCUL SUR UNE SOUS-PÉRIODE

Lors de l'exécution d'une identité sur une période ne couvrant pas toute celle du WS courant, les valeurs en dehors de la période d'exécution ne sont pas modifiées.

Avant de commencer le calcul, les valeurs d'origine de la série output sont fixées comme suit :

- Si la série est en WS, les valeurs du WS sur la période du WS, NA pour les autres périodes
- Si la série n'est pas en WS, mais dans un des fichiers précisés dans la liste des fichiers input, les valeurs du premier fichier input qui la contient
- Si la série n'est ni en WS, ni dans un des fichiers input, toutes ses valeurs sont fixées à NA.

CHOIX DES FICHIERS DE VARIABLES ET SCALAIRES

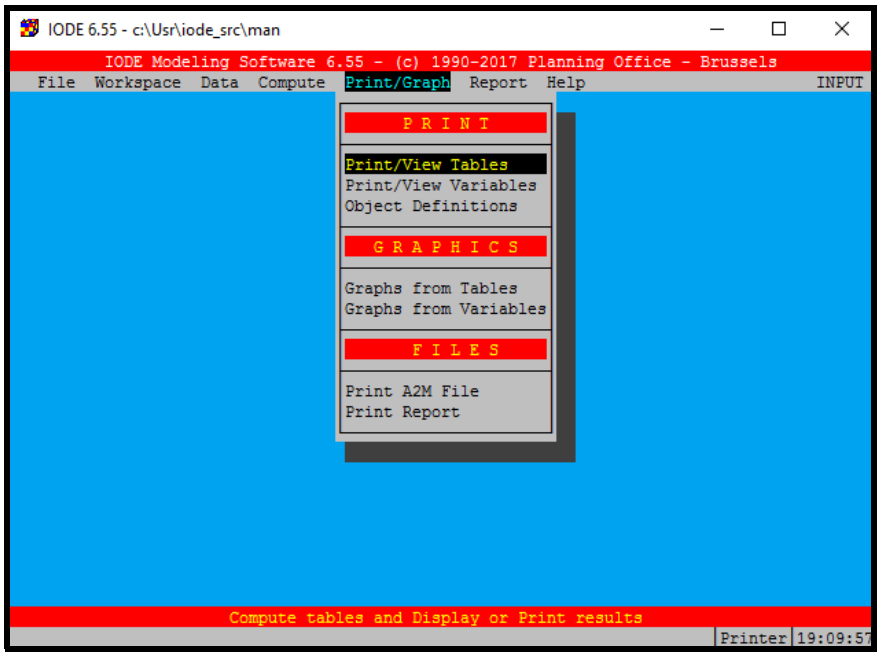
Si aucun nom de fichier n'est spécifié, les WS de variables et de scalaires sont utilisés pour les valeurs des objets nécessaires à l'exécution des identités.

Si au moins un nom de fichier est donné, le WS n'est pas utilisé, sauf en spécifiant comme nom "spécial" WS (en majuscule) dans la liste des fichiers. Il convient de différencier WS de ws : ws représente en fait le fichier disque "ws.var".

Ceci permet de n'utiliser les données en mémoire (en WS) que si on le souhaite réellement. De plus, la place de WS dans la liste des fichiers est quelconque : d'autres fichiers peuvent être consultés avant le WS lui-même.

2.5 PRINT/GRAPH

FIGURE 74



Un des outils essentiels de la construction, du test et de la productivité d'un modèle est la visualisation et l'impression de tableaux et de graphiques. IODE dispose des outils nécessaires à cette fin. IODE permet également d'imprimer tous les objets utilisés dans un modèle.

Toutes les impressions peuvent être dirigées vers l'imprimante ou redirigées vers un fichier. S'il s'agit d'un fichier, le nom du fichier d'impression sera demandé au début de l'impression, pour permettre éventuellement de changer de nom et/ou de format de fichier.

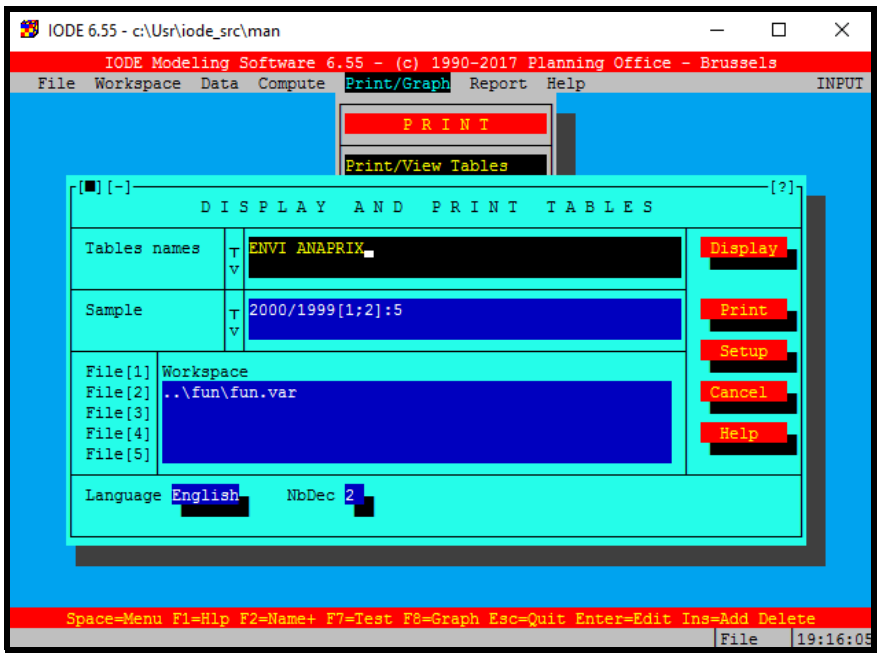
Les parties suivantes détaillent les outils disponibles dans ce menu pour imprimer les définitions des objets, les tableaux et les graphiques:

- Print/View Tables : compilation et impression de tables
- Print/View Variables : impression des variables (sample donné)
- Object Definitions : impression de la définition des objets
- Graphs from Tables : graphiques construits sur des tables
- Graphs from Variables : graphiques avec le WS "Variables"
- Print A2M File : impression d'un fichier A2M
- Print Report : impression de la définition d'un rapport

Dans la suite de ce chapite, il est fait régulièrement référence à la notion de sample d'impression ou sample généralisé utilisé pour préciser le calcul à effectuer pour chaque colonne des tableaux ou chaque courbe des graphiques. La syntaxe précise des samples d'impressions peut être trouvée en annexe.

2.5.1 IMPRESSION ET VISUALISATION DES TABLES

FIGURE 75



Ce point du menu permet le construction, la visualisation et l'impression des tableaux.

Lorsque plusieurs tableaux doivent être affichés (sous forme de graphique ou de tableau de nombres), un tableau déroulant reprenant le nom de chaque tableau ainsi que son titre est proposé.

Il suffit de presser ENTER sur le tableau souhaité pour afficher le résultat sous forme de tableau ou de graphique.

La page de saisie contient les champs suivants:

- **Table names** : liste des tableaux séparés par des blancs, des virgules ou des retours à la ligne
- **Sample** : sample d'impression (voir annexes). Il s'agit d'un sample généralisé acceptant des définitions de taux de croissance, de moyennes, etc, et la comparaison de fichiers.
- **File[1..5]** : quatre champs actifs afin de définir la liste des fichiers d'origine des variables, le fichier numéro 1 étant par défaut le WS courant. Ces fichiers sont référencés dans le sample généralisé par des [1], [2], etc.
- **Language** : le langage utilisé pour les textes automatiques (taux de croissance, etc)
- **NbDec** : nombre de décimales imprimées

La liste des tableaux à imprimer doit être définie. Elle peut contenir des noms de liste ($...) et/ou une énumération de noms d'objets séparés par un blanc, une virgule, un point-virgule ou un retour à la ligne.

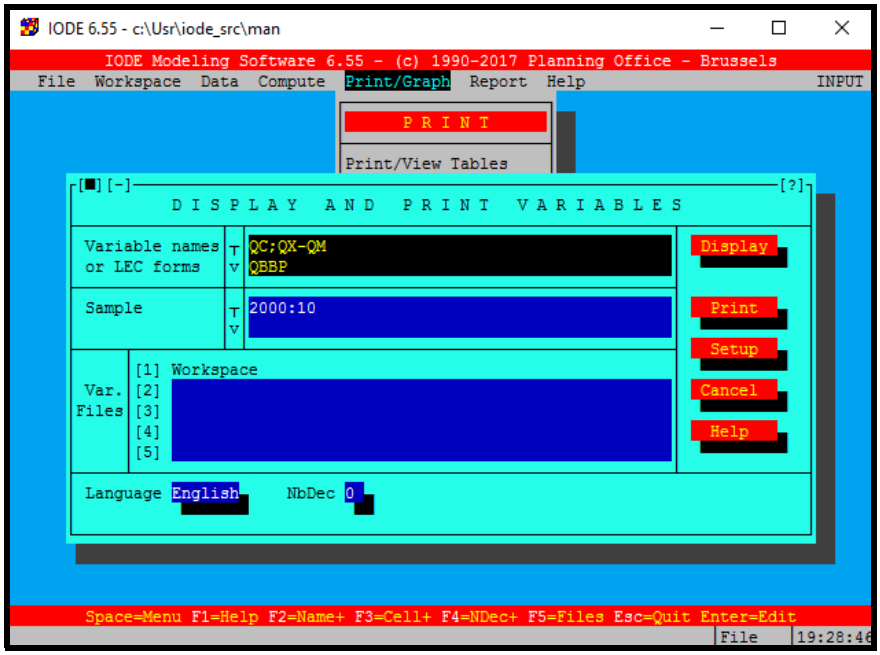
Touches fonctions

Les touches de fonctions applicables au niveau de l'écran de saisie sont:

- **F3** ou ESCAPE : quitte l'écran
- **F1** : affiche l'aide
- **F6** : setup de l'impression (choix entre fichier ou imprimante)
- **F8** : lance l'impression
- **F10** : affiche le résultat à l'écran

2.5.2 IMPRESSION DE VARIABLES

FIGURE 76



Ce point du menu permet la visualisation et l'impression d'un tableau contenant des variables sur une période donnée. Il s'agit d'une fonction comparable à celle d'impression de tableaux, à ceci près que le tableau est construit localement sur base des formes LEC données.

Les formes LEC (ou les noms des variables) sont séparées par des points-virgules ou des retours à la ligne.

La page de saisie contient les champs de saisie suivants:

- **Variable names or LEC-forms** : liste de variables ou de formes LEC séparées par des points-virgules ou des retours à la ligne
- **Sample** : sample d'impression (voir annexes). Il s'agit d'un sample généralisé acceptant des définitions de taux de croissance, de moyennes, etc, et la comparaison de fichiers.
- **File[1..5]** : quatre champs actifs afin de définir la liste des fichiers d'origine des variables, le fichier numéro 1 étant par défaut le WS courant. Ces fichiers sont référencés dans le sample généralisé par des [1], [2], etc.
- **Language** : le langage utilisé pour les textes automatiques (taux de croissance, etc)
- **NbDec** : nombre de décimales imprimées

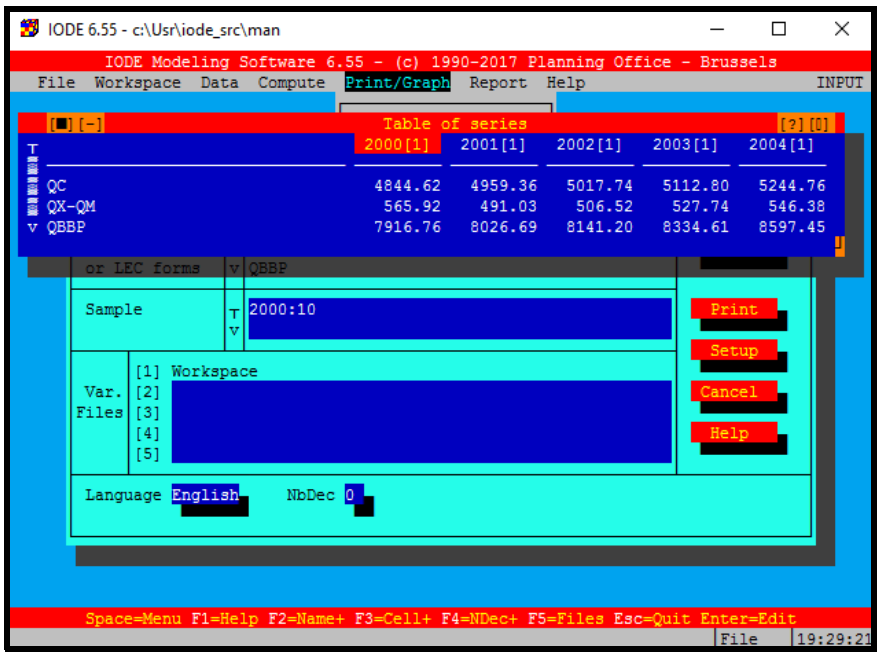
Touches fonctions

Les touches de fonctions applicables au niveau de l'écran de saisie sont:

- **F3** ou ESCAPE : quitte l'écran
- **F1** : affiche l'aide
- **F6** : setup de l'impression (choix entre fichier ou imprimante)
- **F8** : lance l'impression
- **F10** : affiche le résultat à l'écran

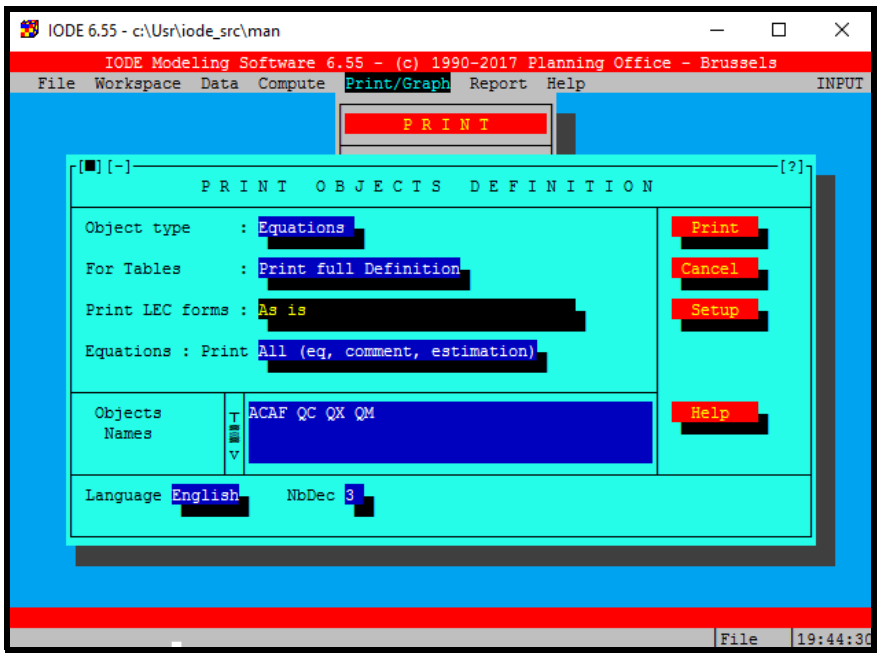
Exemple d'affichage (F10)

FIGURE 77



2.5.3 IMPRESSION DE LA DÉFINITION D'OBJETS

FIGURE 78



Ce point du menu permet l'impression de la définition des objets. Dans le cas de tableaux, par exemple, les formes LEC, les types de lignes sont imprimés, plutôt que les valeurs calculées.

La page de saisie contient les champs suivants:

- **Object type** : type d'objet à imprimer,
- **For Tables** : choix entre impression des titres ou de la définition complète des tableaux
- **Print LEC forms** :
  - dans leur forme normale
  - avec remplacement des coefficients par leur valeur
  - avec remplacement des coefficients par leur valeur et les t-tests à côté des coefficients
- **Object names** : liste des objets à imprimer. Si la liste est vide, tous les objets du type spécifié, présents dans le WS, seront imprimés. Les listes ($...) sont admises.
- **Language** : le langage utilisé pour les textes automatiques (taux de croissance, etc)
- **NbDec** : nombre de décimales imprimées

Les objets SCALAIRES, TABLEAUX, VARIABLES sont imprimés sous forme de tableaux, les autres objets (LISTES, COMMENTAIRES, EQUATIONS et IDENTITES) sont imprimés sous forme de texte enrichi.

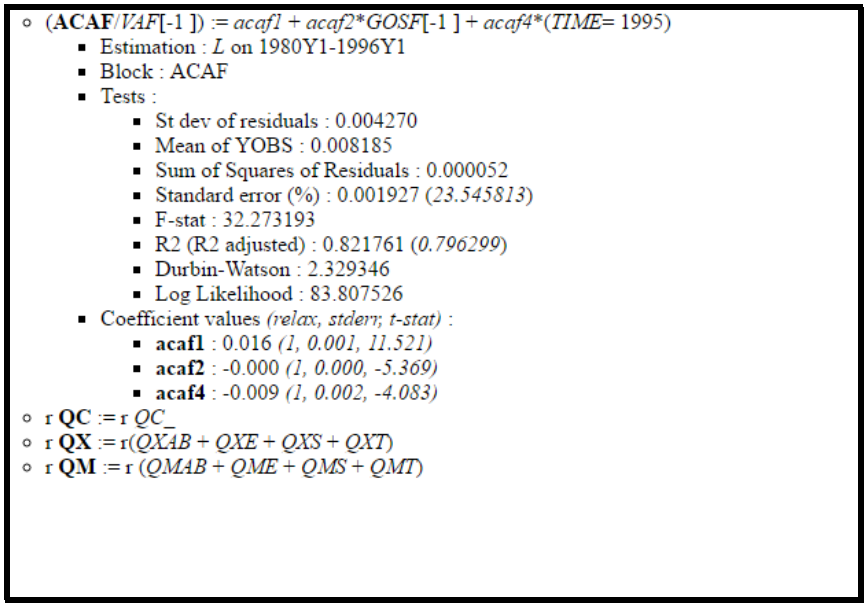
Touches fonctions

Les touches de fonctions applicables au niveau de l'écran de saisie sont:

- **F3** ou **ESCAPE** : quitte
- **F1** : affiche l'aide
- **F6** : setup de l'impression (choix entre fichier ou imprimante)
- **F8** : lance l'impression
- **F10** : affiche les séries à l'écran

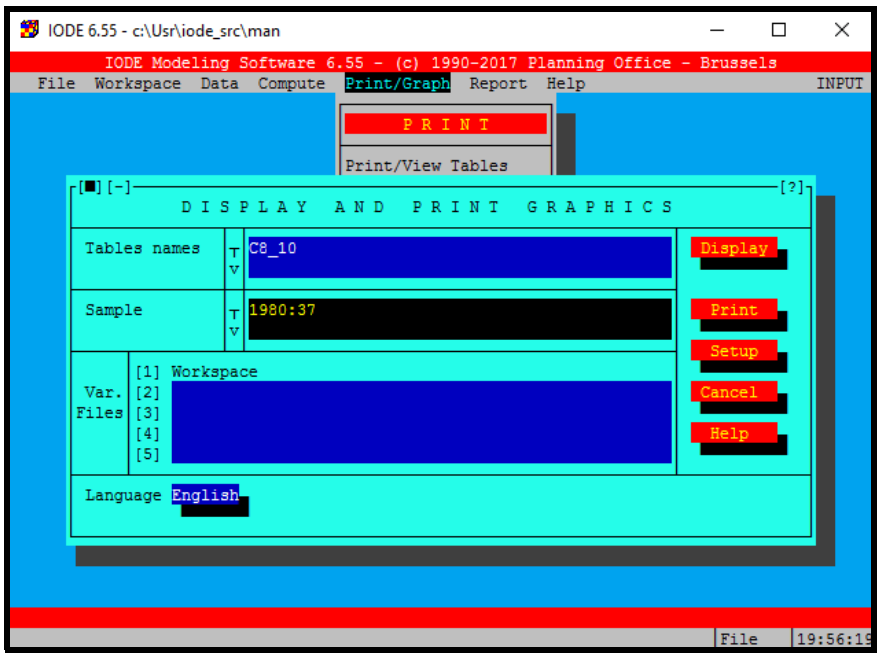
EXEMPLE D'OUTPUT

FIGURE 79



2.5.4 IMPRESSION ET VISUALISATION DES GRAPHES SUR BASE D'UNE TABLE

FIGURE 80



Ce point du menu permet la construction, la visualisation et l'impression des graphiques construits au départ de tables. L'ensemble des attributs graphiques aura été spécifié lors de la construction des tables.

Les paramètres sont semblables à ceux définis pour l'impression et la visualisation de tableaux. Les mêmes objets sont en effet exploités, mais pour construire des graphiques au lieu de tableaux de valeurs.

La page de saisie contient les champs suivants:

- **Table names** : liste des tableaux séparés par des blancs, des virgules ou des retours à la ligne
- **Sample** : sample d'impression (voir annexes). Il s'agit d'un sample généralisé acceptant des définitions de taux de croissance, de moyennes, etc, et la comparaison de fichiers.
- **File[1..5]** : quatre champs actifs afin de définir la liste des fichiers d'origine des variables, le fichier numéro 1 étant par défaut le WS courant. Ces fichiers sont référencés dans le sample généralisé par des [1], [2], etc.
- **Language** : le langage utilisé pour les textes automatiques (taux de croissance, etc)
- **NbDec** : nombre de décimales imprimées

La liste des tableaux à imprimer doit être définie. Elle peut contenir des noms de liste ($...) et/ou une énumération de noms d'objets séparés par un blanc, une virgule, un point-virgule ou un retour à la ligne.

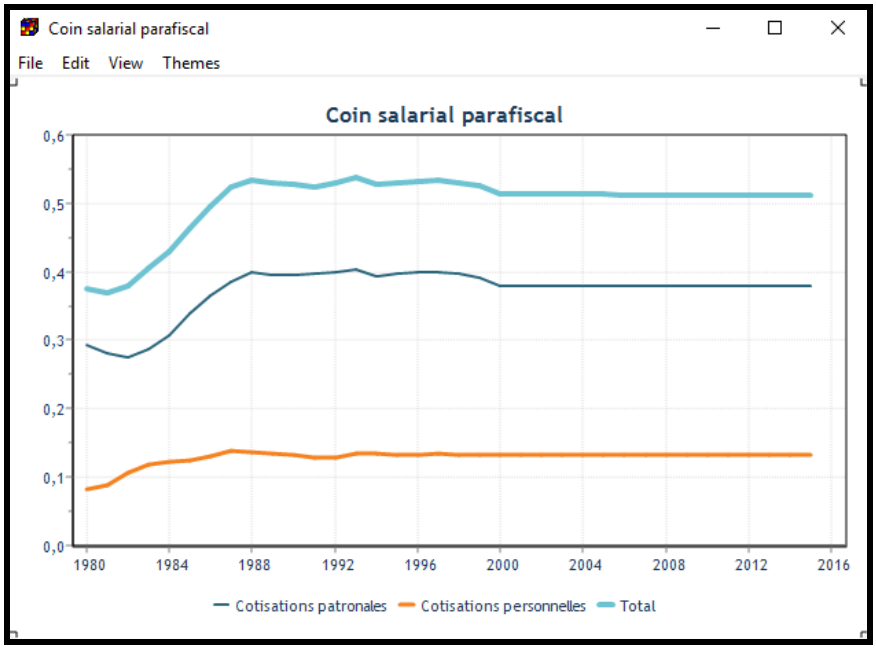
Touches fonctions

Les touches de fonctions applicables au niveau de l'écran de saisie sont:

- F3 ou ESCAPE : quitte
- F1 : affiche l'aide
- F6 : setup de l'impression (choix entre fichier ou imprimante)
- F8 : lance l'impression des tableaux
- F10 : affiche les tableaux à l'écran

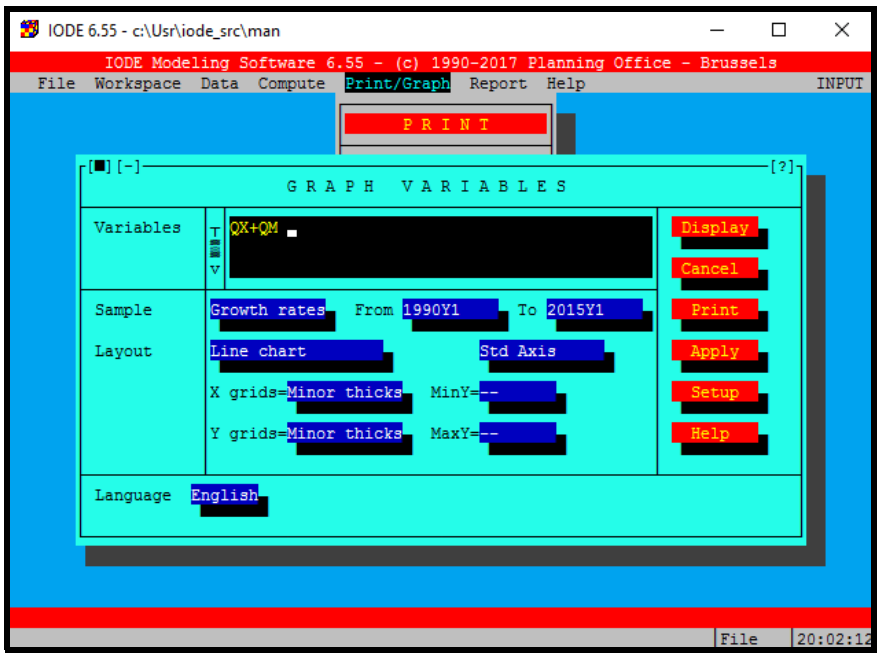
EXEMPLE DE RÉSULTAT

FIGURE 81



2.5.5 IMPRESSION ET AFFICHAGE DES GRAPHES SUR BASE DES VARIABLES

FIGURE 82



Ce point du menu permet la construction, la visualisation et l'impression de graphiques construits au départ des variables contenues dans le WS courant.

La page de saisie permet de paramétrer la construction du ou des graphiques. Elle contient les champs suivants:

- **Variable names** : liste des variables à inclure dans le graphique. Les noms (ou liste) peuvent être séparés par un blanc, un + ou une combinaison des deux. Les variables séparées par un blanc seront visualisées dans des graphiques séparés sur la même page, tandis que les variables "additionnées" seront groupées dans un même graphique. Ainsi, si ce champ contient **`"X Y+Z T"`**, trois graphiques seront construits avec respectivement les variables X (premier graphique), Y et Z (groupées dans le deuxième graphique) et T (dernier graphique).
- **Sample (menu)** : il s'agit d'un champ MENU permettant de préciser si les variables doivent être visualisées en valeur, en différence ou en taux de croissance
- **Sample** (from, to) : permet de préciser la période d'impression (de période_1 à période_2), par exemple from 19970Y1 to 2010Y1.
- **Layout** : différents paramètres on un effet sur l'apparence du graphique résultat :
  - Type de courbes : il s'agit d'un champ MENU permettant de préciser le type de graphique à produire: line, scatter, scatter line ou bar
  - Axis : ce champ du type MENU permet de sélectionner le type de transformation de coordonnées à utiliser: valeur, logarithme, ...
  - XGrids et Ygrids : il s'agit également de champs MENU permettant de définir le

type de quadrillage en X et en Y (pas de quadrillage, quadrillage majeur ou quadrillage mineur)

- MinY et MaxY : ces deux champs permettent de définir les valeurs minimum et maximum de l'axe des Y. Si ces valeurs sont NA (--), l'axe est calculé automatiquement en fonction des valeurs à afficher.

- **Language** : les textes fixés par le programme (taux de croissance, etc) peuvent être imprimés en français, néerlandais ou anglais.

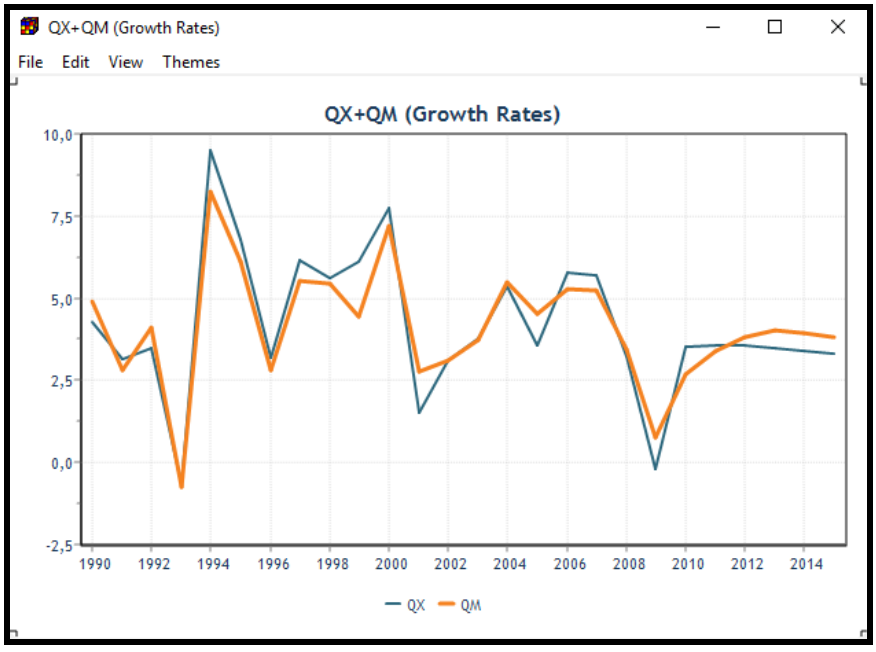
Touches fonctions

Les touches de fonctions applicables au niveau de l'écran de saisie sont:

- F3 ou ESCAPE : quitte
- F1 : affiche l'aide
- F6 : setup de l'impression (choix entre fichier ou imprimante)
- F7 : applique les définitions pour l'affichage ultérieur de séries dans le tableau d'édition des séries
- F8 : lance l'impression
- F10 : affiche à l'écran

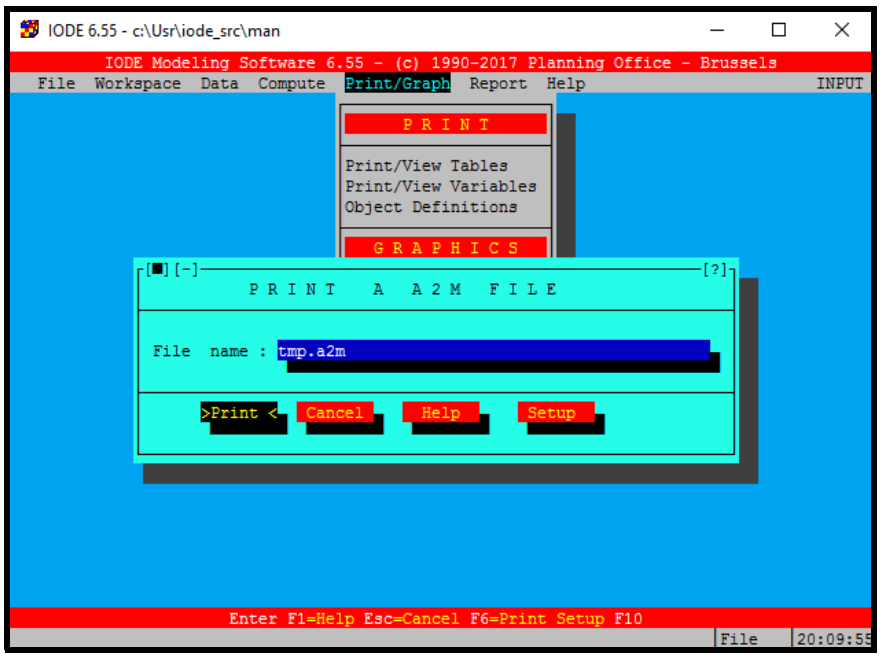
EXEMPLE DE RÉSULTAT

FIGURE 83



2.5.6 PRINT A2M FILE

FIGURE 84



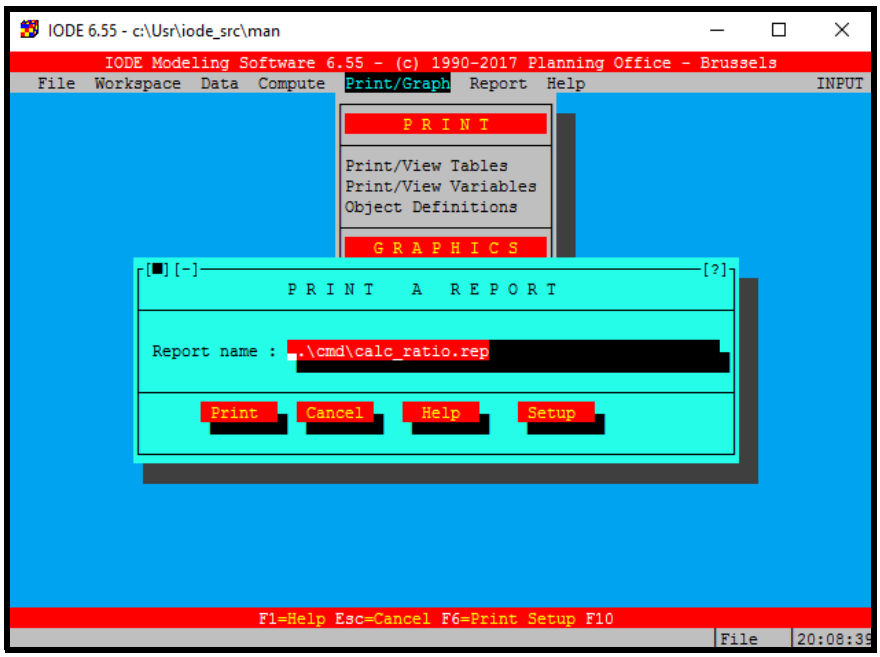
Cette fonction permet d'imprimer un fichier A2M sans quitter IODE. Il suffit de préciser dans l'unique champ de la page le nom du fichier à imprimer.

Le bouton SETUP (ou la touche F6) permet de sélectionner la destination d'impression. Si cette destination est un fichier, le nom du fichier résultat sera demandé dans une fenêtre au moment du lancement de l'impression.

Les options définies pour toutes les impressions à travers la fonction "Print Setup" sont également d'application dans cette fonction.

2.5.7 PRINT REPORT

FIGURE 85



Cette fonction permet d'imprimer un rapport sans en exécuter les commandes. Elle fournit en quelque sorte le code du rapport.

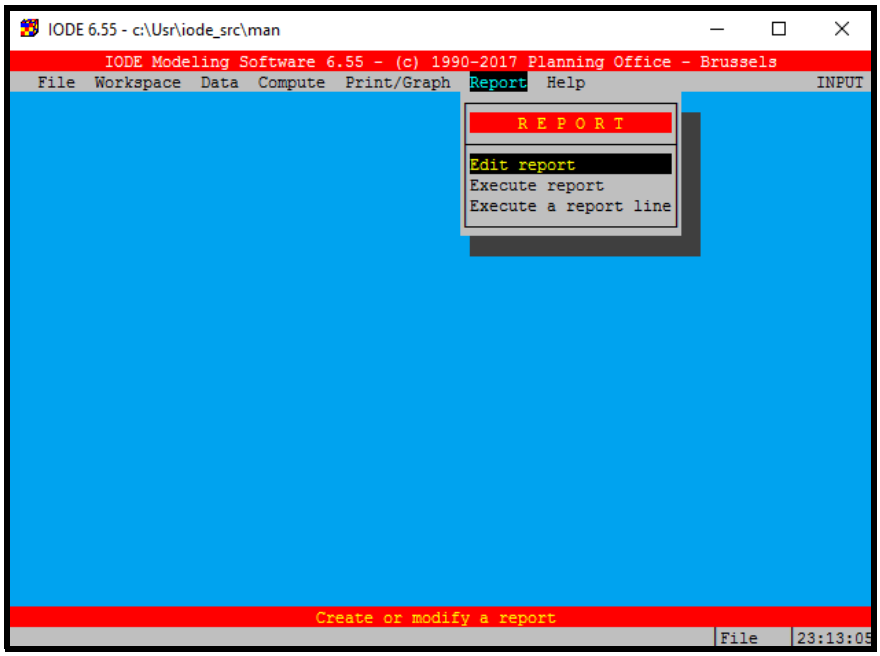
Notons cependant que la structuration éventuelle en paragraphes exploitant les propriétés du langage A2M est exploitée.

Le bouton SETUP (ou la touche F6) permet de sélectionner la destination d'impression. Si cette destination est un fichier, le nom du fichier résultat sera demandé dans une fenêtre au moment du lancement de l'impression.

Les options définies pour toutes les impressions à travers la fonction "Print Setup" sont également d'application dans cette fonction.

2.6 REPORT

FIGURE 86



En utilisant les rapports, les gains de productivité peuvent être considérables. IODE dispose des outils nécessaires pour automatiser l'enchaînement d'une série d'opérations et imprimer le rapport correspondant.

Il est possible de réaliser de réels programmes à l'aide des rapports de IODE. Le déroulement d'un rapport peut être contrôlé de manière interactive (fonctions ASK et PROMPT) ou calculée (GOTO). Par l'utilisation des macros (define) et des arguments des rapports, un même rapport peut être utilisé dans différentes situations.

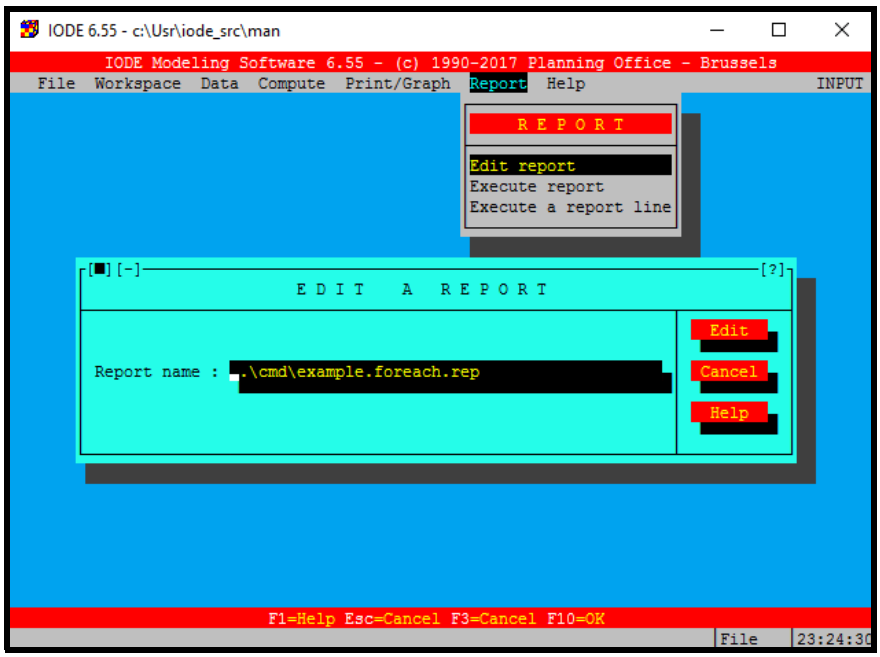
La théorie sur la construction des rapports peut être trouvée dans le chapitre consacré aux rapports.

La suite du chapitre détaille les opérations nécessaires à la production et à l'exploitation des rapports.

- Edit report : édition ou modification d'un rapport
- Execute report : exécution d'un rapport
- Execute a report line : exécution d'une ligne de rapport

2.6.1 EDIT REPORT

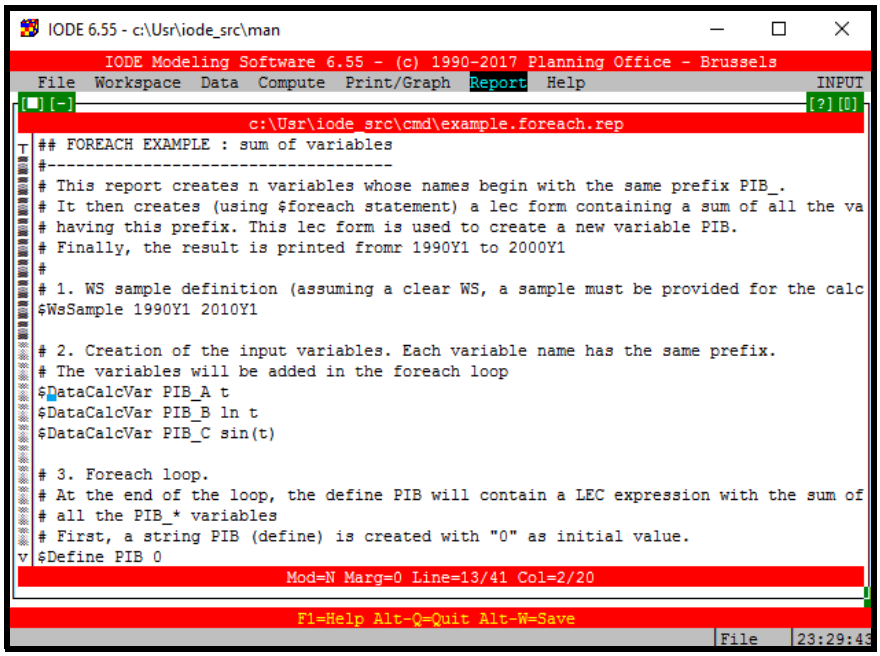
FIGURE 87



Cette fonction permet de créer ou de modifier un rapport.

Le nom du fichier à éditer ou à créer est indiqué dans le champ de saisie de la page. La touche **F10** lance l'édition du fichier avec l'éditeur MMT (voir Annexes).

FIGURE 88

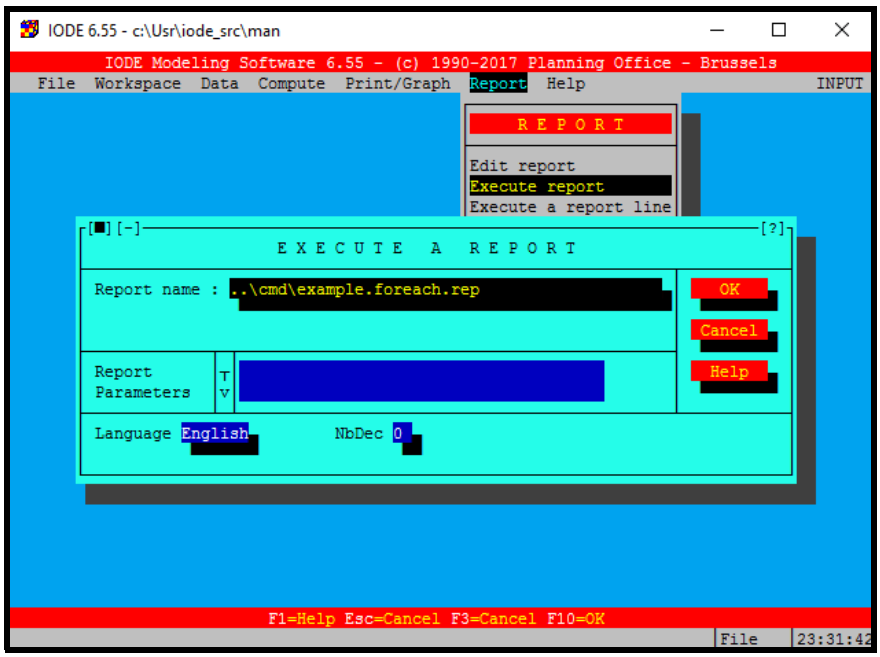


La touche **ENTER** permet de consulter la liste des fichiers présents sur le disque.

Des éditeurs comme Notepad++ ou Textpad peuvent remplacer avantageusement l'éditeur de rapport. Des fichiers de configuration adaptés aux rapports sont disponibles pour ces deux logiciels sur le site iode.plan.be.

2.6.2 EXECUTE REPORT

FIGURE 89



Cette fonction lance l'interprétation, l'exécution et l'impression d'un rapport.

La page de saisie comprend les champs suivants:

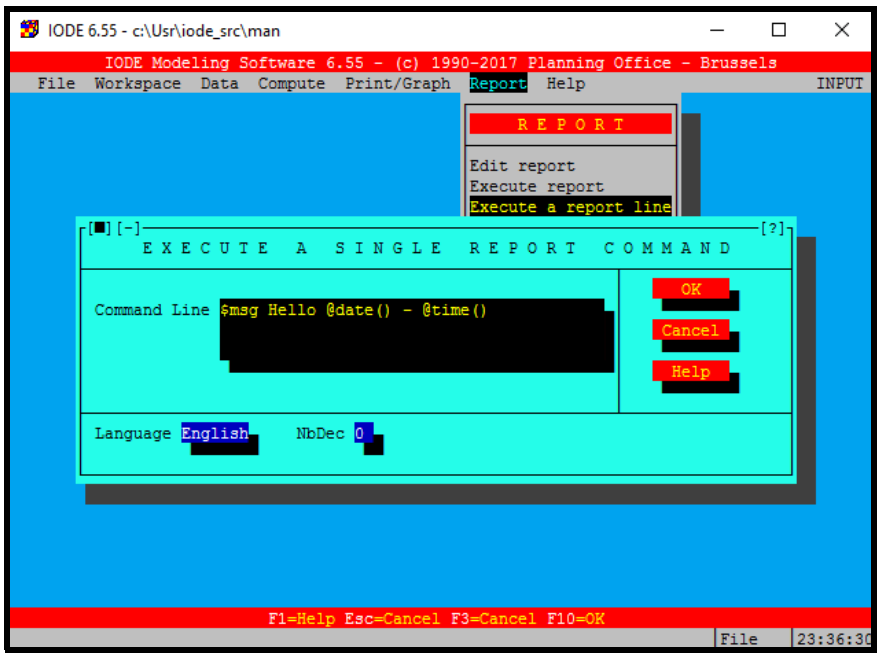
- **Report name** : nom du fichier de commandes à exécuter
- **Report parameters** : paramètres du rapport (référencés à l'intérieur du rapport par %n%). Les paramètres sont séparés par des blancs, des virgules ou des points-virgules. Ils doivent tous être placés sur la même ligne.
- **Language** : permet de spécifier la langue dans laquelle les éventuels textes fixes (dans les tableaux, par exemple) devront être imprimés.
- **Nb Dec** : nombre de décimales par défaut lors de l'impression des variables ou de tableaux.

Les impressions se font avec le nombre de décimales et dans le langage défini dans les derniers champs de la page, sauf si ces valeurs sont modifiées dans le rapport.

Si des messages appropriés ont été prévus (commande show et/ou beep), l'utilisateur peut suivre le déroulement des opérations par les messages qui s'affichent dans la fenêtre de commentaires de IODE. Si des commandes "plein écran" sont implémentées dans le rapport, l'utilisateur est amené en cours d'exécution à compléter les pages de saisie correspondantes.

2.6.3 EXECUTE A REPORT LINE

FIGURE 90



Cette fonction lance l'interprétation, l'exécution et l'impression d'une ligne de commande. Les commandes valides sont toutes les commandes de rapport (voir le chapitre sur les Rapports).

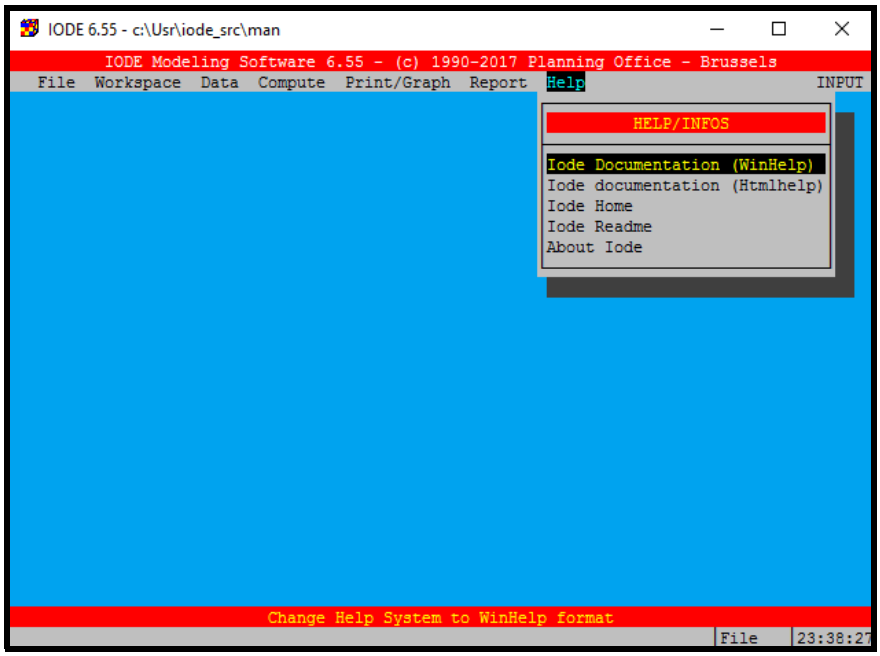
IODE exécute la commande indiquée puis rend la main à l'utilisateur.

L'écran permet en outre de préciser deux valeurs:

- **Language** : permet de spécifier la langue dans laquelle les éventuels textes fixes (dans les tableaux, par exemple) devront être imprimés.
- **Nb Dec** : nombre de décimales par défaut lors de l'impression des variables ou de tableaux.

2.7 HELP : FONCTIONS D'INFORMATIONS

FIGURE 91



Ce chapitre détaille des fonctions générales qui sont d'ailleurs souvent communes à plusieurs modules de IODE:

- Iode Documentation (WinHelp) : manuel format pré-Vista
- Iode Documentation (HtmlHelp) : manuel htmlhelp
- Iode Home : site web de IODE
- Readme : informations sur les dernières modifications
- About IODE : affiche l'écran d'introduction

2.7.1 IODE DOCUMENTATION (WINHELP)

Ce système d'aide nécessite l'installation du programme Winhelp qui n'estplus proposé en standard par Windows depuis la version Vista. Il présentait cependant un avantage certain par rapport à la nouvelle version : à tout moment, une aide contextuelle était directement accessible en pressant la touche F1. La nouvelle version (HtmlHelp) est moins souple à cet égard.

En installant le logiciel adéquat, il reste possible de profiter de l'ancienne version de l'aide en ligne.

2.7.2 IODE DOCUMENTATION (HTMLHELP)

Il s'agit du même manuel que la version WinHelp (ou pdf), mais dans le système standard de Win-

dows, à savoir le système HtmlHelp.

2.7.3 IODE HOME

Cette fonction ouvre la home page du site web de IODE.

2.7.4 README

Affiche dans le browser par défaut le fichier `readme.htm` qui contient des commentaires sur la version actuelle de IODE et sur les dernières modifications apportées au programme.

2.7.5 ABOUT IODE

Affiche l'écran de copyright qui contient notamment le numéro de version du logiciel.

3. Le langage LEC

Qu'il s'agisse d'équations, d'identités de construction ou encore de graphiques, un langage d'écriture de formules mathématiques adéquat est indispensable. Le langage LEC (Langage Econométrique Condensé) offre l'avantage d'être à la fois concis dans son écriture et particulièrement adapté aux formules faisant intervenir des séries chronologiques. Il est utilisé chaque fois qu'une formule est nécessaire dans le logiciel IODE.

Le LEC est également "naturel", en ce sens que sa syntaxe est proche de l'écriture de formules que l'on peut trouver dans la littérature.

```
Equation de consommation
------------------------
                Yt
    Ct = a + b -- + c.C                          (texte)
                Pt      t-1

    C := a + b * Y / P + c * C[-1]               (LEC)


Equation de production
----------------------
    ln Qt = a ln Kt + (1 - a) ln Lt + c.t + b    (texte)

    ln Q := a * ln K + (1 - a) ln L + c * t + b  (LEC)
```

On peut classer les éléments constitutifs du langage en plusieurs classes :

- les constantes du LEC
- les variables
- les scalaires
- les opérateurs logiques
- les opérateurs algébriques
- les fonctions mathématiques
- les fonctions temporelles
- les listes ou macros
- lags, leads et valeurs de période
- les commentaires

En fin de chapitre, les points suivants sont détaillés :

- priorité des opérateurs
- écriture des équations
- récapitulatif de la syntaxe

3.1 LES CONSTANTES DU LEC

LES CONSTANTES NUMÉRIQUES

Celles-ci sont de simples valeurs qui peuvent être entières ou réelles :

Constantes entières :

```
2, 51234, 12345634
```

Constantes réelles en format flottant :

```
2.234, 0.000123
```

Constantes réelles en format exponentiel :

```
2.234e-3, 1E5, 1.2E-03
```

LES CONSTANTES TEMPORELLES

Variable t

Parmi les objets de IODE, les variables, qui sont des séries chronologiques sont définies sur une période limitée de temps : le SAMPLE du workspace de variables détermine cette période. Par exemple, 1970Y1 à 2000Y1 signifie que les séries sont annuelles (Y) et définies de 1970 à 2000.

On peut utiliser dans le langage LEC des constantes temporelles de format semblable. Elles sont utilisées pour prendre la valeur d'une série à une période fixée par exemple.

Exemples :

```
1990Y1
70Q4
2010M11
```

Elles peuvent également être utiles dans des opérations du type:

```
X * (t < 1990Y1) + Y * (t >= 1990Y1)
```

Cette dernière formule vaut X si la période en cours d'exécution est inférieure à 1990 et Y sinon. On verra plus loin que t représente l'index de la période en cours d'exécution.

Les constantes temporelles prennent une valeur entière qui est celle de l'index dans la série temporelle de la période en cours d'exécution. Supposons que le WS de variables soit défini sur un SAMPLE allant de 1970Y1 à 1995Y1. L'équivalent numérique de la constante temporelle 1970Y1 est 0. De même 1980Y1 vaut 10, etc.

Pourquoi ne pas simplement indiquer 0 ou 10 au lieu de 1970Y1 ou 1990Y1 ? Tout simplement parce que la même formule peut être exécutée sur des WS de SAMPLE différents, et rester correcte. Si le SAMPLE s'étend de 1975Y1 à 1995Y1, la valeur de 1980Y1 ne vaut plus 10 comme dans

le premier cas, mais 5. La formule n'a pas été changée, mais son exécution s'est adaptée au contexte!

La constante `t` indique donc l'index de la période en cours d'exécution. Elle a une valeur entière.

Penons un autre exemple. Supposons que le WS de variables courant soit défini sur le SAMPLE `1990Q1 - 1999Q4` et qu'une simulation d'un modèle soit en cours sur la période `1992Q1` à `1994Q4.` La valeur de `t` changera à chaque période simulée : en `1992Q1, t` vaut 8, en `1992Q2, t` vaut 9, et ainsi de suite.

Dans la formule

```
if(t < 1993Q1, 0, 12.3),
```

`1993Q1` vaut 12 et t vaut 8, 9, 10, 11, ...19, selon la période en cours de simulation. On constate que la valeur de la formule reste 0 tant que t est plus petit que 12, c'est-à-dire jusqu'en 1992Q4, et 12.3 ensuite.

Le même principe peut être appliqué au calcul d'identités qui elles aussi sont définies sur un SAMPLE et donc qui font évoluer la valeur de t.

Variable i

Cette variable du langage LEC permet de connaître lors d'un calcul la différence entre l'année de calcul d'une sous-expression par rapport à l'année de calcul de toute la formule. Cette variable s'appelle i et son comportement s'apparente à celui de t. Elle est donc invariante par rapport à l'année de calcul d'une formule LEC.

On calcule toujours une forme LEC pour une valeur du temps t donnée, mais dans une sous-expression, t peut être différent. C'est par exemple le cas pour les opérateurs d(), dln(), sum(), ... On peut, grace à i, connaître la différence entre le t de calcul de la formule et le t de calcul de la sous-expression courante.

On peut donc calculer des expressions comme

```
sum(t-2, t-4, A / (1 - i)**2),
```

qui équivaut à :

```
A[-2]/(1-(-2))**2 + A[-3]/(1-(-3))**2 + A[-4]/(1-(-4))**2
```

Sans i, une telle expression ne peut être écrite en LEC.

Cet opérateur a été implémenté pour permettre la traduction de l'opérateur SUM de TROLL.

Ainsi, si on calcule une expression pour un t donné, i prend les valeurs suivantes selon les cas. Attention, les expressions indiquées ci-dessous sont les formules complètes, et pas une sous-expression!

```
- A + i == A + 0
- d(A+i) == A + 0 - (A[-1] + -1)
- l(i+1, A) == "A[-(i+1)]" == "A[-(0+1)]" == A[-1]
- sum(t-1, t-2, i**2) == (-1)**2 + (-2)**2
- sum(t-1, t-2, l(i-2, A) / i**2) ==
    l((-1)-2, A) / (-1)**2 +
    l((-2)-2, A) / (-2)**2        ==
        l(-3, A) / 1 +
        A[+3] + A[+4] / 4
```

LES CONSTANTES DU LANGAGE

Trois constantes mathématiques sont définies dans le langage :

- pi qui vaut 3.141592653589
- e qui vaut 2.7182818284
- euro vaut 40.3399

3.2 LES VARIABLES

Les variables représentent des séries numériques temporelles. Leur nom est toujours écrit en caractères majuscules pour les distinguer des scalaires (voir ci-dessous).

Les noms des variables peuvent contenir jusqu'à 10 caractères (20 à partir de la version 6.01). Ces caractères doivent être alphabétiques (A-Z) ou numériques (0-9), ou encore le caractère de soulignement. Chaque nom de variable doit commencer par un caractère alphabétique MAJUSCULE.

EXEMPLE

```
A, B_PNB, A123456789
```

sont des noms corrects de variables.

EXEMPLE

```
_A1, 1A34, A_123456789, z_AV, A-2, B.X
```

sont des noms incorrects (z_AV est un nom de scalaire)

Le fait d'écrire une variable dans une formule signifie qu'il faut en prendre la valeur à la période de temps en cours d'exécution (que ce soit dans le cadre d'un modèle, d'un jeu d'identités ou d'un tableau).

Pour en prendre la valeur à la période précédente (lag), on écrira :

```
PNB[-1]
```

La valeur à la période suivante (lead) s'écrit :

```
PNB[+1]
```

La valeur à la période 1990Y1 (dégénérescence) s'écrit :

```
PNB[1990Y1]
```

Cette syntaxe est plus générale et peut s'appliquer à des formules entières.

3.3 Les scalaires

Les scalaires sont des variables sans dimension, donc définies par une seule valeur. Ce sont par exemple les coefficients estimés dans une équation ou des valeurs constantes à travers le temps.

Les noms des scalaires peuvent contenir jusqu'à 10 caractères (20 à partir de la version 6.01). Ces caractères doivent être alphabétiques (A-Z) ou numériques (0-9), ou encore le caractère de soulignement. Chaque nom doit commencer par un caractère alphabétique minuscule.

Exemple

```
a, c1, a_123456789
```

sont des noms corrects de scalaires.

Exemple

```
_a1, 1A34, a_123456789, Z_av
```

sont des noms incorrects (Z_av est un nom de variable)

Les opérations de lag, lead et dégénérescence appliquées à un scalaire sont sans effet, mais permises.

3.4 Les opérateurs logiques

Pour indiquer les opérations logiques à effectuer sur les valeurs définies par les constantes, va-

riables ou scalaires, on dispose des opérateurs suivants, présentés dans leur ordre de priorité :

- not ou ! : négation d'une expression
- or : ou logique entre deux expressions
- and : et logique entre deux expressions
- < : inférieur
- <= : inférieur ou égal
- = : égalité
- != ou <> : différent
- >= : inférieur ou égal
- > : inférieur

Les opérateurs logiques ont une valeur nulle lorsque la condition exprimée est fausse et 1 si les valeurs satisfont à la condition.

Notons qu'une fonction if() est également définie et se trouve dans les fonctions décrites plus bas dans le texte.

EXEMPLE

```
Soient X = 1
       Y = 0
       Z = 2

A la période courante

Les formules                    Valent
-----------------------------------
!X                                0
X and !Y                          1
X or !Y                           1
!(X + Y)                          0
!(2.32 + X)                       0
X == 0 and Y == 0 or Z == 2       1
Z < 1 * 3                         1
```

3.5 LES OPÉRATEURS ALGÉBRIQUES

Les calculs algébriques sont effectués à l'aide des opérateurs suivants :

- + : addition
- - : soustraction
- / : division
- * : multiplication
- ** : exponentiation

Ceux-ci ont les significations et les priorités habituelles.

3.6 LES FONCTIONS MATHÉMATIQUES

De nombreuses fonctions mathématiques sont implémentées en LEC. Certaines fonctions prennent un argument, d'autres plusieurs. Parfois, une fonction accepte un nombre variable d'argument, les arguments absents pouvant prendre une valeur par défaut.

De manière générale, une fonction qui n'a qu'un argument ne demande pas les parenthèses. Ainsi

```
ln X + 2        est identique à     ln(X) + 2
d X             est identique à     d(X)
```

Toutes les fonctions ont un nom entièrement écrit en minuscules. Des coefficients ne peuvent donc pas prendre un des noms réservés aux fonctions.

Dans la syntaxe qui suit, les notations suivantes sont utilisées :

- expr : n'importe quelle expression LEC (formule)
- crochets [] : partie optionnelle
- points de suspension : répétition du dernier argument autant de fois que souhaité

Les fonctions mathématiques sont :

- -expr : moins unaire
- +expr : plus unaire
- log([expr], expr) : logarithme en base quelconque
- ln(expr) : logarithme népérien
- exp([expr,] expr) : exponentielle en base quelconque
- max(expr, ...) : maximum d'une liste de valeurs
- min(expr, ...) : minimum d'une liste de valeurs
- lsum(expr, expr, ...): somme de formules
- sin(expr) : sinus
- cos(expr) : cosinus
- acos(expr) : arccosinus
- asin(expr) : arcsinus
- tan(expr) : tangente
- atan(expr) : arctangente
- tanh(expr) : tangente hyperbolique
- sinh(expr) : sinus hyperbolique
- cosh(expr) : cosinus hyperbolique
- abs(expr) : valeur absolue
- sqrt(expr) : racine carrée
- int(expr) : partie entière
- rad(expr) : transforme des degrés en radians
- if(expr, expr, expr) : condition
- sign(expr) : signe d'une expression
- isan(expr) : retourne 0 si expr est NAN et 1 sinon
- lmean(expr, ...) : retourne la moyenne des expressions
- lprod(expr, ...) : retourne le produit des expressions
- lcount(expr, ...) : retourne le nombre d'expressions
- floor(expr) : partie entière de l'expression
- ceil(expr) : partie entière de l'expression plus 1
- round(expr [, n]) : arrondi de expr à la nème décimale. Si n n'est pas défini, il est fixé à 0.
- random(expr) : fournit un nombre aléatoire compris entre -expr/2 et +expr/2.

Quelques opérateurs demandent un complément d'explication :

Plus unaire

Le plus unaire est sans effet : ++++++X vaut simplement X.

Logarithme

ln(X) est le logarithme népérien de X. log(X) vaut ln(X) tandis que log(10, X) est le logarithme en base 10 de X.

Exponentielle

exp(X) est équivalent à e**X et exp(10, X) vaut 10**X.

Max, min et lsum

max(1, X, Y, Z + 2) prend la valeur maximale de toutes les valeurs passées comme argument (à la période courante). La fonction min() en prend le minimum. La fonction lsum() effectue la somme de toutes les valeurs passées comme argument.

Dans les trois cas, le nombre d'argument soit être compris entre 2 et 50.

Partie entière

`int(X)` retourne l'entier le plus proche de X :

```
int(2.2) = 2
int(2.6) = 3
int(2.5) = 3
```

Fonction if

La fonction if permet de simplifier l'écriture de conditions : le premier argument est la condition, le second la valeur si la condition est vérifiée, le troisième la valeur dans le cas contraire. Il faut signaler que les trois arguments sont calculés et qu'en cas d'erreur dans l'exécution de l'un d'eux, la formule retourne la valeur NA (not available) même si c'est l'expression inutile qui a généré l'erreur.

EXEMPLE

```
if(t < 1992Y1, 2, X)    vaut 2 avant 1992Y1
                             X à partir de 1992Y1
if(t < 1992Y1, 2, X/0)  vaut NA à partir de 1992Y1
                             2 avant 1992Y1.
```

Fonction sign

La fonction sign retourne le signe d'une expression :

```
sign(expr) vaut
                1 si expr >= 0
                -1 si expr < 0
```

Fonction random

La fonction **`random`** fournit un nombre pseudo-aléatoire compris entre **`-expr/2`** et **`+expr/2.`** Sa syntaxe est:

```
random(expr)
```

3.7 LES FONCTIONS TEMPORELLES

Un langage économétrique ne mériterait pas son nom sans fonctions temporelles. De nombreuses équations et formules demandent des calculs du type

```
c1 * (PNB - PNB[-1]) + c2 * (VXA - VXA[-1])
```

Les opérateurs temporels permettent de simplifier cette écriture en la remplaçant par :

```
c1 * d PNB + c2 * d VXA
```

Dans la syntaxe qui suit, les notations suivantes sont utilisées :

- expr : n'importe quelle expression LEC (formule)
- [] : partie optionnelle

Le LEC possède des opérateurs temporels pour calculer les différences de périodes, les taux de croissance, les moyennes, les écarts-types, etc. En voici la liste :

- l([expr,] expr) : lag d'une expression
- d([expr,] expr) : différence de périodes
- r([expr,] expr) : rapport de périodes
- dln([expr,] expr) : différence des logarithmes des périodes
- grt([expr,] expr) : taux de croissance
- ma([expr,] expr) : moyenne mobile (Moving Average)
- mavg([expr,] expr) : moyenne mobile (identique à ma)
- vmax([expr,[expr,]] expr) : maximum sur une période
- vmin([expr,[expr,]] expr) : minimum sur une période
- sum([expr,[expr,]] expr) : somme sur une période
- prod([expr,[expr,]] expr) : produit sur une période
- mean([expr,[expr,]] expr) : moyenne de période
- index([expr,[expr,]] expr1, expr2) : index d'une valeur
- acf([expr,[expr,]] expr, expr) : facteur d'autocorrélation
- var([from [,to],] expr) : variance
- covar([from [,to],] expr1, expr2) : covariance
- covar0([from [,to],] expr1, expr2) : covariance autour de l'origine
- corr([from [,to],] x, y) : corrélation
- stderr([expr,[expr,]] expr) : déviation standard (non biaisé)
- stddev([from [,to],] expr) : déviation standard (biaisé)
- lastobs([from [,to],] expr) : calcule la dernière observation sur une période donnée
- interpol(expr) : fournit une valeur à expr en t en interpolant ou extrapolant
- app(expr1, expr2) : fournit une valeur à expr1 en t en utilisant la série expr2 comme série apparentée par une méthode géométrique
- dapp(expr1, expr2) : fournit par différences une valeur à expr1 en t en utilisant la série expr2 comme série apparentée
- appdif(expr1, expr2) : alias de dapp()

On trouvera ci-dessous la signification précise de tous les opérateurs temporels.

Lag d'une expression

L'expression l(expr1, expr2) est équivalente à expr2[expr1]. Cette formulation permet de calculer ou de paramétrer des lags. Par exemple, l'expression

```
X[c1 + 2]
```

est syntaxiquement fausse, tandis que

```
l(c1 + 2, X)
```

est parfaitement correcte et équivalente.

Si expr1 n'est pas définie, elle est remplacée par 1. Ainsi

```
l(X) est identique à l(1, X)
```

Différence de périodes

L'expression d(expr1, expr2) est équivalente à expr2 - expr2[expr1], où le résultat du calcul de expr1 est considéré comme un lag s'il est négatif et un lead s'il est positif. Par exemple :

```
d(2, X + Y)  est identique à (X + Y) - (X + Y)[-2]
```

expr1 peut être une expression quelconque. Si expr1 n'est pas définie, elle est remplacée par 1. Ainsi

```
d(X) est identique à d(1, X)
```

Rapport de périodes

L'expression r(expr1, expr2) est équivalente à expr2 / expr2[expr1], où le résultat du calcul de expr1 est considéré comme un lag s'il est négatif et un lead s'il est positif. Par exemple :

```
r(-2, X + Y)  est identique à (X + Y) / (X + Y)[+2]
```

expr1 peut être une expression quelconque. Si expr1 n'est pas définie, elle est remplacée par 1. Ainsi

```
r(X) est identique à r(1, X)
```

Différence des logarithmes des périodes

L'expression dln(expr1, expr2) équivaut à ln(expr2) - ln(expr2)[expr1], où le résultat du calcul de expr1 est considéré comme un lag s'il est négatif et un lead s'il est positif. Par exemple :

```
dln(3 - 2, X + Y) vaut ln(X + Y) - ln(X + Y)[-1]
```

expr1 peut être une expression quelconque. Si expr1 n'est pas définie, elle est remplacée par 1. Ainsi

```
dln(X) est identique à dln(1, X)
```

Taux de croissance

L'expression grt(expr1, expr2) équivaut à :

```
100 * (expr2 / expr2[expr1] - 1)
```

où le résultat du calcul de expr1 est considéré comme un lag s'il est négatif et un lead s'il est positif. Par exemple :

```
grt(2, X + Y) vaut 100 * ((X + Y) / (X + Y)[-2] - 1)
```

expr1 peut être une expression quelconque. Si expr1 n'est pas définie, elle est remplacée par 1. Ainsi

```
grt(X) est identique à grt(1, X)
```

Moyenne mobile (Moving Average)

L'expression ma(expr1, expr2) donne la moyenne des expr1 dernières valeurs de l'expression expr2. Si expr1 est négatif ou nul, la valeur courante de expr2 est retournée par cette expression.

expr1 peut être une expression quelconque. Si expr1 n'est pas définie, elle est remplacée par 1. Ainsi

```
ma(X) est identique à ma(1, X) et donc à X
```

Maximum sur une période

L'expression vmax(expr1, expr2, expr3) calcule le maximum de l'expression expr3 sur la période définie par les indices expr1 et expr2. Ces deux dernières expressions sont quelconques, mais leur résultat est arrondi à la partie entière et considéré comme un index de la série calculée par expr3.

On utilisera par exemple :

```
vmax(1970Y1, 1990Y1, X + Y)
```

pour calculer le maximum de la série X + Y sur la période 1970Y1 à 1990Y1.

Utilisé avec deux arguments, le second (absent) est remplacé par t, soit l'indice actuel dans le cours de l'exécution.

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
vmax(1970Y1, X + Y)  vaut   vmax(1970Y1, t, X + Y)
vmax(X + Y)          vaut   vmax(0, t, X + Y)
```

Minimum sur une période

L'expression vmin(expr1, expr2, expr3) calcule le minimum de l'expression expr3 sur la période définie par les indices expr1 et expr2. Ces deux dernières expressions sont quelconques, mais leur résultat est arrondi à la partie entière et considéré comme un index de la série calculée par expr3.

On utilisera par exemple :

```
vmin(t - 2, t, X + Y)
```

pour calculer le minimum de la série X + Y sur les trois dernières valeurs de la série (par rapport à la période de calcul courante).

Utilisé avec deux arguments, le second (absent) est remplacé par t, soit l'indice actuel dans le cours de l'exécution.

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
vmin(1970Y1, X + Y)   vaut   vmin(1970Y1, t, X + Y)
vmin(X + Y)           vaut   vmin(0, t, X + Y)
vmin(t, t, X + Y)     vaut   (X + Y)
```

Somme sur une période

L'expression sum(expr1, expr2, expr3) calcule la somme de l'expression expr3 sur la période définie par les indices expr1 et expr2. Ces deux dernières expressions sont quelconques, mais leur résultat est arrondi à la partie entière et considéré comme un index de la série calculée par expr3.

On utilisera par exemple :

```
sum(0, t, X + Y)
```

pour calculer le total de la série X + Y depuis son origine jusqu'à la période de calcul courante.

Utilisé avec deux arguments, le second (absent) est remplacé par t, soit l'indice actuel dans le cours de l'exécution.

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
sum(1970Y1, X + Y)    vaut   sum(1970Y1, t, X + Y)
sum(X + Y)            vaut   sum(0, t, X + Y)
```

Produit sur une période

L'expression prod(expr1, expr2, expr3) calcule le produit de l'expression expr3 sur la période définie par les indices expr1 et expr2. Ces deux dernières expressions sont quelconques, mais leur résultat est arrondi à la partie entière et considéré comme un index de la série calculée par expr3.

On utilisera par exemple :

```
prod(1985Y1, t - 1, X / X[85Y1])
```

pour calculer le produit cumulé de la série X mise en base 85 depuis 1985 jusqu'à la période courante - 1.

Utilisé avec deux arguments, le second (absent) est remplacé par t, soit l'indice actuel dans le cours de l'exécution.

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
prod(1970Y1, X + Y)  vaut   prod(1970Y1, t, X + Y)
prod(X + Y)          vaut   prod(0, t, X + Y)
```

Moyenne de période

L'expression mean(expr1, expr2, expr3) calcule la moyenne de l'expression expr3 sur la période définie par les indices expr1 et expr2. Ces deux dernières expressions sont quelconques, mais leur résultat est arrondi à la partie entière et considéré comme un index de la série calculée par expr3.

On utilisera par exemple :

```
mean(0, t, X)
```

pour calculer la moyenne de la série X sur la période allant de l'origine à l'année courante de calcul.

Utilisé avec deux arguments, le second (absent) est remplacé par t, soit l'indice actuel dans le cours de l'exécution.

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
mean(1970Y1, X + Y)  vaut   mean(1970Y1, t, X + Y)
mean(X + Y)          vaut   mean(0, t, X + Y)
mean(t - 1, t, X)    vaut   ma(2, X)
mean(t, t, X)        vaut   X
```

Index d'une valeur dans une série

L'expression **`index(2.5, X)`** retourne la position de 2.5 dans la série X.

Facteur d'auto-corrélation

L'expression **`acf([expr,[expr,]] expr1, expr2)`** retourne le facteur d'autocorrélation de l'expression expr2 de degré expr1 sur la période (optionnelle) passée comme premiers arguments. En l'absence de définition la période est [0, t].

Le degré (expr1) ne peut dépasser le quart du nombre d'observations de la période considérée. Dans le cas contraire, la fonction retourne --.

```
acf(1970Y1, X + Y)  vaut   acf(1970Y1, t, X + Y)
acf(X + Y)          vaut   acf(0, t, X + Y)
```

Variance

L'expression **`var([from [,to],] expr)`** retourne la variance de expr sur la période [from, to].

```
        var([from [,to],] X)              == sum((Xi-Xm)**2) / n
        où Xm est la moyenne de X et n le nombre d'observations.
```

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
       var(1970Y1, X + Y)  vaut   var(1970Y1, t, X + Y)
       var(X + Y)          vaut   var(0, t, X + Y)
```

Covariance et covariance autour de l'origine

L'expression **`covar([from [,to],] expr1, expr2)`** retourne la covariance de expr1 et expr2 sur la période [from,to].

`covar0()` retourne la covariance autour de l'origine.

```
        covar([from [,to],] X, Y)  == sum((Xi-Xm)*(Yi-Ym)) / n
        covar0([from [,to],] X, Y) == sum(Xi * Yi) / n

        où Xm est la moyenne de X et n le nombre d'observations.
           Ym est la moyenne de Y et n le nombre d'observations.
```

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
       covar(1970Y1, X + Y)  vaut   covar(1970Y1, t, X + Y)
       covar(X + Y)          vaut   covar(0, t, X + Y)
```

Corrélation

L'expression **`corr([from [,to],] X, Y)`** retourne la corrélation entre X et Y.

```
        corr(X, Y) == covar(X, Y) / sqrt(var(X) * var(Y))
```

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
       corr(1970Y1, X + Y)  vaut   corr(1970Y1, t, X + Y)
       corr(X + Y)          vaut   corr(0, t, X + Y)
```

Déviation standard sur une période

L'expression stderr(expr1, expr2, expr3) calcule la déviation standard de l'expression expr3 sur la période définie par les indices expr1 et expr2. Ces deux dernières expressions sont quelconques, mais leur résultat est arrondi à la partie entière et considéré comme un index de la série calculée par expr3.

On utilisera par exemple :

```
stderr(1970Y1, 1990Y1, X)
```

pour calculer l'écart type de la série X sur la période 1970 à 1990.

Utilisé avec deux arguments, le second (absent) est remplacé par t, soit l'indice actuel dans le cours de l'exécution.

Utilisé avec un seul argument, le premier argument est fixé à 0 (origine de la série) et le second à t (indice courant).

Ainsi :

```
stderr(1970Y1, X + Y)  vaut   stderr(1970Y1, t, X + Y)
stderr(X + Y)          vaut   stderr(0, t, X + Y)
```

La même syntaxe est d'application pour la fonction stddev(). A la différence de stderr(), stddev() est un estimateur biaisé.

```
stddev(x) == sqrt(var(x))
stderr(x) == sqrt(sum((xi-xm)**2/(n-1)
```

Calcul de la dernière observation

Cette fonction permet d'obtenir la valeur de la dernière valeur non NA d'une série sur une période donnée. Sa syntaxe est:

```
lastobs([from [,to],] expr)
```

Calcul d'interpolation

Cette fonction fournit une valeur à expr en t en interpolant ou extrapolant les valeurs connues. Cette inter ou extrapolation se fait uniquement de façon linéaire.

```
interpol(expr)
```

Si on dispose de A en 90 et 92, **`interpol(A)`** en 91 vaudra **`(A[90Y1]+A[92Y1])/2.`** Plusieurs valeurs successives peuvent être manquantes. Dans ce cas, une pondération est utilisée en fonction de la proximité des valeurs disponibles.

Séries apparentées

La fonction **app** fournit une valeur à une expression lorsque l'observation est manquante en utilisant une autre série comme série apparentée. Sa syntaxe est :

```
app(expr1, expr2)
```

La calcul se fait de façon géométrique en se basant sur les valeurs connues les plus proches dans la série à compléter.

Le résultat de la fonction **app(A,** B) est obtenu comme suit au temps t :

- Si **A[t]** est défini, **A[t]** est retourné
- Si **A[t]** n'est pas défini, on calcule les valeurs **t0** et **t1** autour de **t** telles que **A[t0]** et **A[t1]** soient définis et non nuls. Si ni **t0** ni **t1** ne peuvent être trouvés, retourne **NaN.** Il s'agit dans ce cas d'un série **A** valant **NaN** sur toute la période.
- si seul **t0** est défini, retourne **B[t] * (A[t0] / B[t0])**
- Si seul **t1** est défini, retourne **B[t] * (A[t1] / B[t1])**
- Si **t0** et **t1** sont définis, calcule d'abord **Delta = (A[t1]/A[t0]) / (B[t1]/B[t0])** puis retourne **A[t0] * (B[t]/B[t0]) * Delta ** ((t-t0)/(t1-t0))**

3.8 LES LISTES OU MACROS

Il est possible de paramétrer des expressions LEC en y introduisant des noms de listes (gérées et définies dans le WS courant de listes). Ces listes permettent aussi de raccourcir l'écriture des équations.

On peut également demander au LEC de calculer une liste en plaçant des étoiles ou des points d'interrogations. Dans ce cas l'expression à "étendre" sera placée entre simples quotes.

LISTES NOMMÉES

Une liste porte un nom répondant aux mêmes règles syntaxiques qu'une variable. Elle est précédée dans l'équation par un dollar ($) pour la distinguer d'une variable :

Supposons que la liste A ait comme définition

```
PNB * c1 + c2
```

L'expression

```
A + B + $A + 2
```

sera équivalente à

```
A + B + PNB * c1 + c2 + 2
```

Note : les listes doivent exister au moment de la compilation des formules. Une fois compilées, les formes LEC ne le sont plus tant qu'il n'y a pas de modification dans la formule LEC elle-même. Les modifications apportées aux définitions des listes n'auront donc un impact au niveau des formes LEC (équations, identités, tableaux) que si les formes LEC sont recompilées!

LISTES CALCULÉES

Lorsque l'opérateur le permet (**lsum, max**, etc), on peut utiliser les wildcards dans le langage LEC. Cependant, pour distinguer l'étoile et l'opérateur de multiplication, la liste à étendre doit être entourée de quotes simples('). Ainsi,

```
lsum('A*')
```

est équivalent à

```
lsum(A1,A2,AX)
```

si **A1, A2** et **AX** sont les seules séries commençant pas **A** dans le WS courant.

On peut également utiliser une combinaison de noms de séries :

```
max('*G;*F')
```

Le nombre maximum d'opérandes des opérateurs est porté à 255, de façon à pouvoir exploiter au mieux les wildcards.

Note

Les noms contenant des wildcards dans les formes LEC sont résolus en fonction du contenu du WS COURANT. La forme LEC compilée est mémorisée avec ces noms. Si on change le contenu du WS, il est possible que certaines séries n'existent plus ou que de nouvelles apparaissent. La forme compilée n'étant pas changée automatiquement, il faut si on souhaite adapter la forme LEC au nouveau contenu, recompiler la forme en l'éditant.

3.9 UTILISATION DES LAGS, LEADS ET PÉRIODES DANS LE FORMULES

Les variables utilisées dans une expression LEC ont toujours une dimension temporelle implicite : l'expression

```
A + PNB
```

signifie

```
A[t] + PNB[t]
```

Cette dernière écriture est cependant syntaxiquement fausse.

On peut appliquer à des variables ou plus généralement à des expressions trois types de modifications d'indice temporel :

- le lag (retard)
- le lead (avance)
- la fixation de période

Dans les trois cas, on ajoute à l'expression modifiée une valeur entre crochets représentant le lag, le lead ou la période.

```
expr[-lag]
expr[+lead]
expr[period]
```

Un lag et un lead sont nécessairement des nombres entiers précédés par le signe - (pour le lag) ou le signe + (pour le lead) indiquant le nombre de périodes de temps dont on veut déplacer l'expression. La période est une constante temporelle, soit par exemple 1970Q3.

Des combinaisons de lag, lead et fixations de période sont possibles. La règle est simple : dès qu'une période est fixée, les lag et les lead n'ont plus d'effet sur l'expression. Les lag et lead sont simplement additionnés.

Le report des lag, lead et fixation de période s'effectue avant le calcul des fonctions. Ainsi max(A, B)[-1] est identique à max(A[-1], B[-1]).

Voici quelques exemples :

Expression	Equivalent ou signification
X[-1]	X[t-1] (faux syntaxiquement)
(X + Y + c1)[1970Y1]	X[1970Y1] + Y[1970Y1] + c1
(A + B[+1])[-2]	A[-2] + B[-1]
(A[70Y1] + B)[-1][-2]	A[70Y1] + B[-3]
(A[+1] + B[70Y1])[80Y1]	A[81Y1] + B[70Y1]
(a + 2)[-1]	a + 2
d X[-2]	X[-2] - X[-3]
(grt(X))[-2]	X[-2] / X[-3]
max(A, B)[1970Y1]	max(A[70Y1], B[70Y1])
d(A)[1970Y1]	0 (A[70Y1] - A[70Y1])
t[-1]	t (constante)
mean(X)[-1]	NA (hors sample)
mean(1, t, X)[-1]	vaut NA la première année
A[1]	faux
B[70Y1 + 1]	faux
C[t-1]	faux

3.10 LES COMMENTAIRES

Des commentaires peuvent être introduits dans une expression LEC. Il suffit de les placer entre /* et */.

Par exemple :

```
A + B + /* première partie  */
C       /* série temporaire */
+ D     /* suite            */
```

L'introduction d'un commentaire permet entre autre de supprimer temporairement une partie d'équation :

```
ln Y := c1 + /* c2 * ln Z + */ c3 * ln Y[-1]
```

Une méthode alternative est de placer en fin d'expression un point-virgule suivi d'un texte libre :

```
ln Y := c1 + c2 * ln Y[-1] ; commentaire libre
```

A la différence des /* et */, le point-virgule termine l'équation : tout le texte qui suit est ignoré, y compris sur les lignes suivantes :

```
A + B +
C       ; * série temporaire
+ D     /* suite */
```

est équivalent à :

```
A + B + C
```

3.11 PRIORITÉ DES OPÉRATEURS

Les priorités définies des opérateurs sont classiques. Par ordre de priorité croissante :

- l'opérateur logique or
- l'opérateur logique and
- les opérateurs de comparaison <, <=, etc
- la somme et la soustraction
- le produit et la division
- l'exposant
- les fonctions mathématiques et temporelles

Par "a priorité sur", il faut entendre "s'exécute après" dans une formule. Ainsi

```
2.2 * X < 100
```

vaut 1 si (2.2 * X) est plus petit que 100 et 0 sinon.

Les opérateurs de priorités égales s'exécutent de gauche à droite :

```
2 - 1 + 2
```

vaut 3 car c'est équivalent à (2 - 1) + 2.

Les parenthèses peuvent être utilisées pour modifier cet ordre :

```
2.2 * (X < 100)
```

vaut 2.2 si X est plus petit que 100 et 0 sinon.

Les opérateurs algébriques ont priorité sur les fonctions, que celles-ci soient mathématiques ou temporelles :

```
ln X + 2
```

est équivalent à

```
ln(X) + 2
```

3.12 Ecriture des équations

Une équation est simplement la juxtaposition de deux expressions LEC, séparées par les caractères :=

```
expr1 := expr2
```

Toutes les règles décrites plus haut s'appliquent à chaque terme de l'équation.

Comme une équation porte le nom de la variable endogène, celle-ci doit apparaître dans l'équation. Elle peut y apparaître plus d'une fois.

3.13 Récapitulatif de la syntaxe

Les lignes suivantes reprennent l'ensemble des éléments du langage et leur syntaxe.

```
constantes numériques   : 2, 2.0, 2.12E2 0.001e-03
constantes temporelles  : 1990Y1, 80S2, 78Q4, 2003M12
constantes du langage   : pi, e, t
variables               : A, A_1, A123456789
scalaires               : a, a_1, x123456789
opérateurs logiques     : not expr
                          expr or expr
                          expr and expr
                          expr < expr
                          expr <= expr
```

```
                          expr = expr
                          expr != expr
                          expr >= expr
                          expr > expr

opérateurs algébriques :  expr + expr
                          expr - expr
                          expr / expr
                          expr * expr
                          expr ** expr

fonctions mathématiques:  -expr
                          +expr
                          log([expr], expr)
                          ln(expr)
                          exp([expr,] expr)
                          max(expr, ...)
                          min(expr, ...)
                          lsum(expr, expr, ...)
                          sin(expr)
                          cos(expr)
                          acos(expr)
                          asin(expr)
                          tan(expr)
                          atan(expr)
                          tanh(expr)
                          sinh(expr)
                          cosh(expr)
                          abs(expr)
                          sqrt(expr)
                          int(expr)
                          rad(expr)
                          if(expr, expr, expr)
                          sign(expr)
                          isan(expr)
                          lmean(expr, ...)
                          lprod(expr, ...)
                          lcount(expr, ...)
                          floor(expr)
                          ceil(expr)
                          round(expr [, n])
                          random(expr)

fonctions temporelles  :  l([expr,] expr)
                          d([expr,] expr)
                          r([expr,] expr)
                          dln([expr,] expr)
                          grt([expr,] expr)
                          ma([expr,] expr)
                          mavg([expr,] expr)
                          vmax([expr,[expr,]] expr)
                          vmin([expr,[expr,]] expr)
                          sum([expr,[expr,]] expr)
                          prod([expr,[expr,]] expr)
                          mean([expr,[expr,]] expr)
                          index([expr,[expr,]] expr1, expr2)
                          acf([expr,[expr,]] expr, expr)
                          var([from [,to],] expr)
                          covar([from [,to],] expr1, expr2)
                          covar0([from [,to],] expr1, expr2)
                          corr([from [,to],] x, y)
                          stderr([expr,[expr,]] expr)
                          stddev([from [,to],] expr)
                          lastobs([from [,to],] expr)
                          interpol(expr)
```

```
                              app(expr1, expr2)
                              dapp(expr1, expr2)

   listes ou macros         : $LISTNAME

   lags, leads et périodes: [+n] [-n] [1990Y1]

   commentaires             : /* Comment */

   équations                : expr := expr
```

4. Les rapports

En utilisant les rapports de IODE, les gains de productivité peuvent être considérables. IODE dispose en effet des outils nécessaires pour automatiser l'enchaînement d'une série de fonctions et imprimer s'il y a lieu les résultats correspondant.

De plus, il est possible de réaliser de réels programmes à l'aide des rapports de IODE. Le déroulement d'un rapport peut être contrôlé de manière interactive (fonctions ASK et PROMPT) ou calculée (GOTO). Par l'utilisation des macros (define) et des arguments des rapports, un même rapport peut être utilisé dans différentes situations.

DÉFINITION D'UN RAPPORT

Un rapport est un texte ASCII comprenant un ensemble d'instructions, de macros et de texte libre, dont l'interprétation et l'exécution se traduisent d'une part par l'enchaînement d'opérations (charger un fichier, estimer, simuler, ...) et d'autre part par la construction ou l'impression d'un document "fini". Ce document peut êtte aussi bien un document papier qu'un fichier RTF, MIF, CSV ou encore HTML.

ORGANISATION

Les parties de ce chapitre fournissent la liste exhaustive des fonctions utilisables dans les rapports. Pour chaque fonction sont fournies la description, la syntaxe et éventuellement un exemple.

- La première partie décrit la structure des rapports en présentant tous les éléments constitutifs du "langage" des rapports (boucle, erreurs, etc).
- La partie 2 reprend les commandes d'exécution des rapports
- La partie 3 définit les macros dans les rapports
- La partie 4 montre comment utiliser les expressions LEC dans les rapports
- La partie 5 indique comment insérer des valeurs d'un worksheet Excel dans les rapports
- La partie 6 explique les fonctions de rapports
- La partie 7 donne quelques exemples d'interprétation des lignes de rapports
- La dernière partie présente les commandes de IODE dans les Rapports, en fournit la syntaxe et les classe par catégorie.

4.1 LA STRUCTURE DES RAPPORTS

Un rapport est un texte ASCII qui peut être édité avec n'importe quel éditeur ASCII. Les éditeurs MT et MMT seront utilisés de préférence pour leur simplicité et leur convivialité. MMT fait partie intégrante de IODE.

Le texte du rapport contient les types d'informations suivants:

- du texte libre: titres, commentaires, etc
- des commandes A2M (formatage du texte)
- les commandes d'exécution des rapports
- les macros
- les expressions LEC
- les fonctions de rapports
- des commandes IODE

Les commandes RAPPORT et les commandes IODE peuvent être des commandes "plein écran" ou des lignes de commande (sans interface écran).

COMMANDE "PLEIN ÉCRAN"

Les commandes "plein écran" sont constituées d'un mot clé précédé par le caractère # et ont la syntaxe suivante:

```
#commande
```

Par exemple : la commande

```
#WsLoadVar ou #WsLoad
```

provoquera l'affichage d'un écran de saisie permettant de préciser le nom du fichier de variables à charger.

COMMANDE "EN LIGNE"

Les commandes en ligne sont constituées d'un mot clé précédé par le caractère $ et ont la syntaxe suivante:

```
$commande [options] [paramètres]
```

Par exemple, la commande

```
$PrintTbl 84:8 TEST
```

imprimera le tableau TEST pour le sample indiqué.

Le mot clé des commandes en ligne comprend souvent deux parties (c'est le cas de toutes les commandes en ligne qui manipulent des objets) : la commande proprement dite (en général identique à la commande plein écran) et un suffixe qualifiant l'objet manipulé. Par exemple à la commande plein écran #WsLoad correspondent les commandes en ligne $WsLoadVar, $WsLoadIdt, etc... respectivement pour les objets variable et identité. Les suffixes valides sont les suivants (tous ne sont pas valables pour toutes les fonctions) :

```
cmt,    eqs,    idt,    lst,    scl,    tbl,    var,
ac,     ae,     ai,     al,     as,     at,     av,
rep,    a2m,    prf,    dif,    mif,    rtf,
ps,     asc,    txt,    htm,    csv
```

RÈGLES SYNTAXIQUES

Le caractère $ ou # indiquant une commande DOIT commencer la ligne et être suivi DIRECTEMENT par le mot clé de la commande (s'il s'agit d'une commande à exécuter). Un caractère $ ou # situé plus loin dans la ligne sera imprimé tel quel. Il ne peut donc y avoir qu'une commande par ligne de rapport.

Le mot clé peut être écrit indifféremment en minuscules, en majuscules ou les deux. Par exemple :

```
WsLoadVar == wsloadvar == WSLOADVAR
```

La liste des mots clés est reprise plus loin dans le texte.

Les commandes dont le résultat (erreur ou nom) doit être ignoré sont indiquées avec le signe - entre le $ (ou #) et le mot clé. Par exemple, si la commande

```
$-WsLoadTbl
```

n'aboutit pas, son résultat sera ignoré (aucune action ne sera exécutée quelle que soit la commande de $OnError).

Les commandes dont le signe $ ou # est répété deux fois ($$ ou ##) seront imprimées dans le rapport sans être exécutées.

COMMENTAIRES

Les lignes dont le signe $ ou # est suivi par un espace sont des lignes de commentaires, non imprimées et non exécutées.

IMBRICATIONS

Les rapports peuvent être imbriqués (nombre non limité d'imbrications), c'est-à-dire qu'un rapport peut en exécuter un autre.

RÉSULTAT DE L'IMPRESSION

Le rapport est "imprimé" dans le fichier a2m sélectionné (en ajout au fichier existant s'il existe). La commande $PrintDest permet de spécifier le nom de ce fichier.

EXEMPLE

Le rapport suivant charge un fichier de définition de tableaux et un fichier de séries, affiche un message, construit les tableaux, les imprime et termine l'exécution par un beep sonore.

```
$wsloadvar bist92\\bistel
$wsloadtbl bist92\\tbistelf

$printdest bist92\\bistelf1.a2m
#show processing french tables  file 1/2
$printnbdec 1
$PrintTbl 89:8 HYPEIR
$PrintTbl 89/88:8 HYPEIIR

$printnbdec 0
$PrintTbl 89:8 RESL00

$printnbdec 1
$PrintTbl 89/88:8 RESL00R
$PrintTbl 89:8 RESL03
#beep
```

Le rapport produit contient également les éventuels messages d'erreur.

4.2 LES COMMANDES D'EXÉCUTION DES RAPPORTS

Ces commandes sont indifféremment précédées par les caractères $ ou #, elles contrôlent le déroulement de l'interprétation du rapport.

Ces commandes (mots clés) sont les suivantes:

- $define : définition d'une macro
- $label : localisation d'un point de branchement
- $goto : branchement (in)conditionnel vers un label
- $onerror : action à mener en cas d'erreur
- $return : quitte le rapport courant
- $abort : interruption du rapport (courant et niveaux supérieurs)
- $quitode : quitte IODE
- $quit : quitte IODE
- $show : affichage d'un message dans le bas de l'écran
- $msg : affichage d'un message et attente d'un touche
- $beep : production d'un beep sonore
- $ask : question et branchement conditionnel vers un label
- $prompt : définition d'une macro par question à l'utilisateur
- $settime : definition de la variable t pour le calcul des formules LEC définies dans le rapport
- $incrtime : incremente la variable t (par défaut de 1) pour le calcul des formules LEC définies dans le rapport
- $system : exécution d'une commande système
- $shift : décale les arguments du rapport d'un vers la gauche
- $minimize : minimise la fenêtre de IODE
- $maximize : restaure la fenêtre de IODE
- $sleep : stoppe le processus pendant un courte période
- $debug : indique le fichier et la ligne en cours d'exécution
- $repeat : boucle sur les arguments
- $repeatstring : définition des caractères à remplacer dans $repeat
- $chdir : change le directory courant
- $mkdir : crée un nouveau directory
- $rmdir : détruit un directory
- $vseps : redéfinit les séparateurs utilisés par les fonctions `@vdrop()`, `@vtake()` et `@vcount()`.
- $foreach : boucle sur une liste de valeurs
- $next : fin de boucle
- $procdef : définition d'une procédure
- $procend : fin de définition d'une procédure
- $procexec : exécution d'une procédure
- $indent : indentation acceptées ou non des instructions

4.2.1 Commande $DEFINE

Cette commande permet de définir des macros qui pourront être utilisées plus tard dans le rap-

port et ses sous-rapports. Une macro est une sorte de variable globale qui reste définie durant toute la durée d'exécution du rapport courant.

Syntaxe

```
$Define name text
name := nom utilisé pour faire référence au contenu (texte) de la
        macro (le nom doit commencer par une lettre et ne pas
        dépasser 10 caractères)
```

Le nom est référencé dans un rapport en l'encadrant de signes %.

Exemple

```
$Define a Hello World!
$Show %a%
```

Exemple

```
$Define VAR A
$GoTo exist {$DataExistVar %VAR%}
$Show Variable %VAR% does not exist
$Return
....
$Label exist
$Show Variable %VAR% exists
$Return
...
```

Si un rapport possède des arguments, ceux-ci sont utilisables comme des macros dont le nom est %n%, où n est le numéro de l'argument. La valeur spéciale %0% contient le nombre d'arguments passés lors de l'appel au rapport.

Le rapport "exist.rep" qui suit

Exemple

```
$Define VAR %1%
$GoTo exist {$DataExistVar %VAR%}
$Show Variable %VAR% does not exist
$Return
....
$Label exist
$Show Variable %VAR% exists
$Return
...
```

exécuté avec A comme argument, vérifie si la variable A existe.

On peut également conserver des valeurs calculées dans des macros.

Exemple

```
$Define PNB {PNB}
$Define GPNB {r PNB * 100}
La valeur du PNB est %PNB% (différence de %GPNB% pourcents)
```

Mode majuscules/minuscules

Si le pourcent est immédiatement suivi du caractère

- # : le contenu de la macro est transposé en majuscules
- ! : le contenu de la macro est transposé en minuscules
- - : le caractères non alphanumérique sont supprimés de la macro

Par exemple,

```
$define TOTO TBL1
$define ARG0 Abc[-3]
Premier  tableau : %!TOTO%
Deuxième tableau : %#TOTO%
ARG1 : %ARG0O%
ARG2 : %-ARG0%
```

Donnera à l'exécution :

```
Premier  tableau : tbl1
Deuxième tableau : TBL1
ARG1 : Abc[-3]
ARG2 : Abc3
```

4.2.2 Commande $LABEL

Cette instruction indique et nomme un point de branchement dans un rapport. Ce point de branchement pourra par la suite être atteint par une instruction comme $goto ou $ask.

Exemple

```
$label again
$goto fin {%0%=0}
#show arg0 : %1%
$DataCalcVar %1% if(%1%=0, 1/0, %1%)
$shift
$goto again
$label fin
```

Dans cet exemple, on remplace tous les 0 par des NA pour la liste de variables (inconnue au départ) passées comme paramètre au rapport. Pour mémoire, %0% est le nombre d'arguments, %1% le premier argument, etc.

4.2.3 Commande $GOTO

Cette commande provoque un branchement vers un label dans le rapport courant.

Syntaxe

```
$GoTo label [{condition}]
label := point de branchement indiqué par la commande Label
condition := expression LEC ou commande de rapport
```

Si le label n'existe pas, l'exécution du rapport (et de tous les rapports de niveaux supérieurs) est interrompue.

Exemple

```
....
$Label boucle
...
$GoTo boucle
...
```

Il est également possible de construire certains tests pour diriger la suite de l'exécution du rapport.

Si on place une condition après le nom du label, celle-ci est exécutée en premier lieu. Le condition peut être soit une commande IODE (par exemple $DataExistVar) ou une formule LEC (par exemple "t != 1993Y1"). La commande doit être placée entre accollades.

Le Goto a lieu si la commande IODE peut être valablement exécutée ou si le résultat de la formule LEC est non nul. Pour différencier commande IODE et formule LEC, la commande doit commencer (comme toujours) par le caractère $.

Exemple

```
...
$GoTo exist {$DataExistVar A}
$Show Variable A does not exist
$Return
....
$Label exist
$Show Variable A exists
$Return
...

ou
...
$GoTo end {X + Y > 1000}
...
$Return
....
$Label end
$Show X + Y exceeds the limit of 1000
$Return
...
```

Il est également possible d'inverser le résultat d'une commande IODE, exactement comme celui d'une formule LEC. Il suffit pour cela de placer un point d'exclamation (!) après le $. L'exemple qui suit indique comment procéder :

Exemple

```
...
$GoTo notexist {$!DataExistVar A}
$Show Variable A does exist
$Return
....
$Label notexist
$Show Variable A does not exist
$Return
...
```

Les formules LEC peuvent quant à elles être aisément adaptées pour retourner la valeur souhaitée (opérateur !).

4.2.4 COMMANDE $ONERROR

Cette commande indique l'action à mener en cas d'erreur rencontrée dans une commande IODE, si cette erreur ne doit pas être ignorée (signe - après le $ ou le #).

Les actions peuvent être: RETURN, ABORT, QUIT, DISPLAY (par défaut), PRINT, NOPRINT, et IGNORE (par défaut).

L'action IGNORE est l'action par défaut: ignore l'erreur et poursuit l'exécution du rapport.

L'action RETURN provoque l'interruption du rapport courant, avec retour au niveau supérieur éventuel.

L'action ABORT provoque l'interruption du rapport courant et de tous les rapports de niveaux supérieurs.

L'action QUIT termine le programme IODE. Elle est utile notamment en combinaison avec l'option -rep du programme iode.

L'action DISPLAY est active par défaut, elle provoque l'impression dans le rapport d'un message d'erreur correspondant à l'erreur rencontrée lors de l'exécution de la dernière commande IODE. Elle signale également à l'écran tous les messages d'erreur correspondant. L'action PRINT se contente d'imprimer les messages d'erreur. L'action NOPRINT interdit l'impression du message.

Les actions RETURN, ABORT, QUIT et IGNORE sont exclusives entre elles, l'action exécutée est celle qui correspond à la dernière commande ONERROR rencontrée. Les actions DISPLAY, PRINT et NOPRINT sont également exclusives mais peuvent être combinées avec une des quatre instructions ci-dessus.

Une seule action peut figurer par ligne de commande.

Syntaxe

```
$OnError action
action := RETURN, ABORT, QUIT, IGNORE, DISPLAY, PRINT ou NOPRINT
```

Exemple

```
$OnError noprint
$OnError ignore
...
$OnError noprint
$OnErrot abort
....
```

4.2.5 COMMANDE $RETURN

Cette instruction indique que l'exécution du rapport courant est terminée. S'il s'agit d'un sous-rapport, le rapport appelant se poursuit.

Syntaxe

```
$Return
```

Exemple

```
Rapport createvar.rep
---------------------
# Create series %1%
$GoTo exist {$DataExistVar %1%}
$DataCalcVar %1% 0.9
$Show Variable %1% created
$Return
$Label exist
$Show Variable %1% exists

Appel
-----
$ExecRep createvar A
```

4.2.6 COMMANDE $ABORT

Cette instruction indique que l'exécution du rapport doit être interrompue. Si le rapport courant est un sous-rapport, les rapports de niveau supérieur doivent également être quittés.

Syntaxe

```
$Abort
```

Exemple

```
...
$GoTo continue {$DataExistVar A}
$Show Variable A does not exist
$Abort
....
$Label continue
...
```

4.2.7 Commande $QUITODE

Voir $quit.

4.2.8 Commande $QUIT

Cette instruction indique que l'exécution du rapport doit être interrompue et que le programme IODE doit être quitté.

Syntaxe

```
$Quit
```

Exemple

```
$WsLoadVar refsim
$ModelSimulate 1997Y1 1998Y1
$WsSaveVar myws
$Quit
```

Cette fonction est notamment utile lorsqu'un rapport est lancé à partir d'un fichier de commandes (DOS ou Unix). En permettant de quitter IODE, elle rend la main au fichier de commandes qui peut enchaîner avec un autre programme (impression, autre simulation, etc).

4.2.9 Commande $SHOW

Cette commande provoque un beep sonore et l'affichage d'un message dans la fenêtre de commentaire du programme IODE pendant l'exécution du rapport.

Syntaxe

```
$Show message
message := texte quelconque
```

4.2.10 COMMANDE $MSG

Cette commande affiche le texte de l'argument et attend une touche avant de continuer. Cela permet de stopper momentanément le rapport pour afficher un message à l'intention de l'utilisateur, contrairement à la fonction $show qui affiche dans le bas de l'écran mais ne stoppe pas.

Syntaxe

```
$Msg message
message := texte quelconque
```

4.2.11 COMMANDE $BEEP

Cette commande provoque un beep sonore pendant l'exécution du rapport.

Syntaxe

```
$Beep
```

4.2.12 COMMANDE $ASK

Cette commande provoque l'affichage d'une question dont la réponse peut être Yes ou No. Si la réponse est non, le rapport continue à la ligne suivante. Si la réponse est oui, l'exécution du rapport se poursuit à partir du label indiqué. Si le label n'existe pas, l'exécution du rapport (et de tous les rapports de niveaux supérieurs) est interrompue.

La question et le label sont séparés par une virgule ou un point-virgule.

Syntaxe

```
$Ask label,question

où label    := point de branchement indiqué par la commande Label
   question := texte quelconque
```

Exemple

```
....
$Label boucle
...
$Ask boucle,retour à boucle ?
...
```

4.2.13 COMMANDE $PROMPT

Cette commande permet de définir des macros qui pourront être utilisées plus tard dans le rap-

port et ses sous-rapports. Une macro est une sorte de variable globale qui reste définie durant toute la durée d'exécution du rapport courant.

La fonction Prompt pose une question à l'utilisateur du rapport et stocke sa réponse dans une variable exploitable dans la suite du rapport.

Syntaxe

```
#Prompt VarName Question
Question := question posée
VarName := nom utilisé pour faire référence au contenu (texte) de la
           macro (le nom doit commencer par une lettre et ne pas
           dépasser 10 caractères)
```

Le nom est référencé dans un rapport en l'encadrant de signes %.

Exemple

```
#Prompt Begin Période de début ?
$SetTime %Begin%
```

4.2.14 COMMANDE $SETTIME

Cette commande fixe la période (t) pour laquelle les formules LEC utilisées dans les rapports doivent être calculées.

Elle DOIT être exécutée avant de calculer une formule LEC dans un rapport, sans quoi la période d'exécution est inconnue et la formule ne s'exécute pas.

Syntaxe

```
$SetTime period
period := la valeur de période à donner à t
```

Exemple

```
$SetTime 1991Y1
La valeur du PNB pour l'année {t@T} est de {PNB} milliards.
```

donne en sortie :

Exemple

```
La valeur du PNB pour l'année 1991Y1 est de 4000.32 milliards.
```

4.2.15 COMMANDE $INCRTIME

Cette commande augmente la période (t) d'autant d'unités qu'indiqué. Sans argument, l'incrément en fixé à 1.

Syntaxe

```
$IncrTime [step]
step := nombre de périodes à ajouter
```

Exemple

```
$SetTime 1990Y1
$Label print
La valeur du PNB pour l'année {t@T} est de {PNB} milliards.
$IncrTime 5
$goto print {t < 2000Y1}
$Return
```

fournit comme résultat :

Exemple

```
La valeur du PNB pour l'année 1990Y1 est de 4132.32 milliards.
La valeur du PNB pour l'année 1995Y1 est de 4240.32 milliards.
```

4.2.16 COMMANDE $SYSTEM

Cette commande de rapport est utilisée pour exécuter une commande système. La commande étant exécutée, le rapport se poursuit à la ligne suivante.

La commande système (suivi de ses éventuels paramètres) est passée en paramètre. Plusieurs commandes peuvent d'enchaîner, elles doivent être séparées par un point-virgule (;).

Avant l'exécution de la commande système, le terminal est remis en mode standard et l'écran est vidé. Après l'exécution de la commande, l'écran est réaffiché dans son état initial.

Attention

- aucune vérification n'est faite quant à la validité ou l'opportunité de la commande système. Une commande comme "del *.*" ou "format c:" est acceptée...
- l'espace mémoire est limité pour l'exécution de la commande système car IODE reste chargé en mémoire (version DOS)
- il peut être intéressant de terminer la liste des commandes sytème à exécuter par la commande PAUSE (en DOS) de façon à maintenir momentanément le terminal en mode standard (visualisation des résultats de la commande système).

Syntaxe

```
$System commande1 [;commande2;...]
commande := commande valide du système d'exploitation [+ arguments]
```

Exemple

```
$System qode -d qms result.a2m
```

4.2.17 COMMANDE $SHIFT

Cette fonction décale les arguments du rapport d'un élément vers la gauche. Par la même occasion, le premier argument est perdu.

Cette fonctionnalité permet d'exécuter des opérations sur un nombre d'éléments inconnus. En effet, comme %0% représente le nombre d'argument du rapport, après $shift, cette valeur est décrémentée. De la sorte, un simple goto calculé comme dans l'exemple ci-dessous, permet de quitter la boucle dès que %0% est nul.

Exemple

```
$label again
$goto fin {%0%=0}
#show arg0 : %1%
$DataCalcVar %1% if(%1%=0, 1/0, %1%)
$shift
$goto again
$label fin
```

4.2.18 COMMANDE $MINIMIZE

DESCRIPTION

Cette commande minimise la fenêtre de IODE (par exemple pour afficher des graphiques de Excel générés par la simulation en cours).

Syntaxe

```
$Minimize
```

4.2.19 Commande $MAXIMIZE

Description

Cette commande restaure la fenêtre de IODE.

Syntaxe

```
$Maximize
```

4.2.20 Commande $SLEEP

Description

Arrête le processus pendant nn millièmes de seconde. Cela permet à un client ou un serveur DDE de conserver le contrôle en cas de requêtes trop rapides.

Syntaxe

```
$Sleep nn
    où nn = nombre de millisecondes d'arrêt
```

4.2.21 Commande $DEBUG

Cette commande affiche dans la ligne de commentaire de l'écran le nom du rapport en cours et la ligne courante. Elle permet de la sorte de connaître aisément une ligne litigieuse en cas d'erreur par exemple.

Syntaxe

```
$debug {Yes|No}
```

4.2.22 Commande $REPEAT

Cette fonction permet d'exécuter une commande sur une liste d'arguments sans devoir créer un rapport à cette fin. La position de l'argument dans la commande à répéter est représentée par le string défini par $repeatstring.

Par défaut, le caractère _ est le caractère de remplacement.

Syntaxe

```
$repeat commande arg1 arg2 arg3
```

commande est exécutée pour chaque argument en remplaçant _ par **arg1**, puis **arg2**, ...

Exemple

```
1. Tri de plusieurs listes
```

```
$RepeatString ++
$Repeat "$DataListSort ++ ++" A B C
```

équivaut aux trois commandes

```
$DataListSort A A
$DataListSort B B
$DataListSort C C
```

2. Duplicate Vars

Avec les fonctions @fn(), on peut copier toutes les variables d'un WS en une seule opération :

```
$Repeat "$DataDuplicateVar _ _1" @vexpand(*)
```

4.2.23 COMMANDE $REPEATSTRING

Cette fonction permet de préciser les caractères à remplacer dans le premier argument de $repeat. Par défaut, ce string est composé du seul caractère de soulignement (_).

Syntaxe

```
$RepeatString string
```

Exemple

```
$RepeatString --
$Repeat "$DataListSort -- --" A B C
```

VOIR ÉGALEMENT

$Repeat

4.2.24 COMMANDE $CHDIR

Cette commande de rapport change le répertoire courant.

- les fichiers sont toujours chargés relativement au répertoire courant. A la fin du rapport, le répertoire précédent n'est PAS restauré.

Syntaxe

```
$chdir dirname
dirname := nom de répertoire (relatif ou absolu)
```

Exemple

```
$chdir ..
$chdir c:\usr\iode
```

4.2.25 COMMANDE $MKDIR

Cette commande de rapport crée un nouveau répertoire.

Syntaxe

```
$mkdir dirname
dirname := nom de répertoire (relatif ou absolu)
```

Exemple

```
$mkdir ..\test
$mkdir c:\usr\iode\test
```

4.2.26 COMMANDE $RMDIR

Cette commande de rapport détruit un répertoire.

Syntaxe

```
$rmdir dirname
dirname := nom de répertoire (relatif ou absolu)
```

Exemple

```
$rmdir ..\test
$rmdir c:\usr\iode\test
```

DESCRIPTION

Cette commande redéfinit les séparateurs utilisés par les fonctions **@vdrop(), @vtake()** et **@vcount()** ainsi que dans **$foreach** et **$next**.

Syntaxe

```
$vseps {seps}
```

Exemple

```
$vseps ;|
```

Voir aussi @vtake().

4.2.27 COMMANDE $FOREACH

Cette commande permet de simplifier l'écriture de boucles ou de boucles imbriquées.

La commande **$foreach** permet de spécifier un index et la liste de valeurs que cet index doit successivement prendre.

La commande **$next** permet de revenir au point de départ de la boucle **$foreach** et de passer à la valeur suivante de l'index.

Syntaxe

```
$foreach {index} {values}
...
$next {index}
```

où

- **{index}** est un nom de macro de maximum 10 caractères (par exemple **i, idx, PAYS**, ...)
- **{values}** est une liste de valeurs séparées par des virgules, blancs ou point-virgules. Les séparateurs peuvent être modifiés par la commande **$vseps**

Exemple 1 : boucles imbriquées

```
$foreach I BE BXL VL WAL
$foreach J H F
$show [%I%;%J%]
$next J
$next I
```

Exemple 2 : utilisation de listes

```
$DataUpdateLst MYLIST X,Y,Z
$Define n 0
$foreach I @lvalue(MYLIST)
$Define n {%n% + 1}
$show Element %n% : [%I%]
$next I
```

4.2.28 COMMANDE $NEXT

Voir $foreach.

4.2.29 COMMANDE $PROCDEF

Cette commande indique le début de la définition d'une procédure IODE. Elle est liée aux commandes **`$procdef, $procend et $procexec`**.

Les instructions de ce groupe permettent de construire des procédures, c'est à dire des listes de commandes qui peuvent être réutilisées et paramétrées.

```
$procdef procname [fparm1 ...]
..
$procend
```

où

- **procname** est le nom de la procédure (case sensitive).
- **fparm1** est le premier paramètre *formel* de la procédure

Appel d'une procédure

L'appel se fait simplement par la commande :

```
$procexec nom_proc aparm1 ...
```

où

- **procname** est le nom de la procédure (case sensitive).
- **aparm1** est le premier paramètre *actuel* de la procédure

Paramètres

Les paramètres formels sont traités dans la procédure comme des **`$defines`** (locaux à la procédure) : ils doivent être appelés par **`%fparm%`**.

Leurs valeurs sont fixées comme suit :

- S'il y a moins de paramètres actuels que de paramètres formels ou si leur nombre est égal, les valeurs des paramètres actuels sont assignées dans l'ordre aux premiers paramètres formels. Les paramètres formels excédentaires sont considérés comme vides.
- S'il y a plus de paramètres actuels que de paramètres formels, les paramètres formels, **sauf le dernier**, reçoivent les valeurs de premiers paramètres actuels, dans l'ordre de leur passage. **Le dernier paramètres formel reçoit la valeur de tous les paramètres actuels restants**.

Exemple avec plus de paramètres actuels de formels :

```
$indent
$procdef print list
    $foreach i %list%
        $show print : %i%
    $next i
$procend
$procexec print A B C
```

Résultat :

```
print : A
print : B
print : C
```

On constate que le paramètre formel **`list`** reçoit toutes les valeurs passées à la procédure. La boucle itère donc 3 fois.

Exemple avec moins de paramètres actuels de formels :

```
$indent
$procdef print titre list
    $show %titre%
    $foreach i %list%
        $show print : %i%
    $next i
$procend
$procexec print "Mon Titre"
```

Résultat :

```
Mon Titre
```

Cette fois, le premier paramètre **`titre`** contient **`"Mon Titre"`** qui est imprimé avant la boucle. Par contre, la boucle ne s'exécute pas car le paramètre **`list`** est vide.

Portée de la définition d'une procédure

1. Les procédures doivent être définies avant de pouvoir être appelées.

2. Une fois définie, une procédure reste appelable au sein de la même session de IODE, même si le rapport qui l'a définie est terminé. On peut exécuter un rapport qui n'a d'autre effet que de charger des procédures en mémoire. Ces procédures resteront disponibles pendant toutes la session IODE.

3. Une procédure peut être remplacée par une autre du même nom à tout moment.

Portée des paramètres formels

Les paramètres formels sont traités comme des defines dont la portée est limitée à la procédure courante.

Par conséquent, si un define existe avant l'appelU de la procédure avec le même nom qu'un des paramètres, ce define ne peut être utilisé au sein de la procédure. Après la procédure, il reprend sa valeur antérieure.

Exemple :

```
$define titre Quotients de mortalité
$show Avant la proc : %titre%

$procdef print titre list
    $show Pendant la proc :%titre%
    $foreach i %list%
        $show print : %i%
    $next i
$procend

$procexec print "Mon Titre"
$show Après la proc :%titre%
```

Résultat :

```
Avant la proc : Quotients de mortalité
Pendant la proc :Mon Titre
Après la proc :Quotients de mortalité
```

4.2.30 Commande $PROCEND

Cette commande indique la fin de définition d'une procédure. Voir $procdef pour plus de détail.

4.2.31 Commande $PROCEXEC

Cette commande permet l'exécution d'une procédure. Voir $procdef pour plus de détail.

4.2.32 Commande $INDENT

Par défaut, les commandes (**$commande** ou **#commande**) doivent être collées à la marge. Si ce n'est pas le cas, elles sont considérées comme du texte simple.

Pour pouvoir indenter les commandes dans les rapports, il faut l'indiquer par la commande **$indent**. Sans argument, elle indique qu'à partir de ce moment, les commandes ne doivent plus être collées à la marge. Avec l'argument N ou n ou 0, les commandes doivent être collées à la marge.

Pour éviter les problèmes de compatibilité avec les rapports crées dans les versions anciennes de IODE, la valeur par défaut est de ne pas accepter les indentations.

```
    $indent [{Nn0}]
```

Par exemple, on peut écrire :

```
$indent

$procdef print list
$------------------
    $foreach i %list%
        $show print : %i%
    $next i
$procend

$procdef print2 list
$-------------------
    $foreach i %list%
        $show print2 : %i%
        $procexec print 1 2 3
    $next i
$procend

$procexec print2 A B C
```

Le résultat produit est le suivant :

```
print2 : A
print : 1
print : 2
print : 3
print2 : B
print : 1
print : 2
print : 3
```

4.3 Les macros dans les rapports

Deux sortes de macros peuvent être utilisées dans les rapports IODE.

Les premières sont les macros locales qui sont uniquement connues du rapport en cours d'exécution. Il s'agit des arguments du rapport (%n%). On peut utiliser ces valeurs pour peu qu'on connaisse l'ordre dans lequel elles sont passées. Un sous-rapport reçoit bien-sûr de nouvelles définitions pour les variables %n%. Lors du retour dans le rapport de niveau supérieur, les valeurs de %n% sont remises à leur état d'origine.

Si le caractère % doit se trouver dans le texte, il faut le doubler. Ainsi,

```
    %%1%%
```

n'est pas considéré comme une macro mais remplacé dans l'output par le texte %1%.

Syntaxe

```
%n%
n := position de l'argument
(%0% := nombre d'arguments passés au rapport)
(%*% := tous les arguments passés au rapport)
```

Soit le rapport invert.rep :

Exemple

```
$show %2% %1%
$show %*%
$shift
$show %*%
$Return
```

L'appel

Exemple

```
$ReportExec invert een twee
```

donne :

Exemple

```
twee een
een twee
twee
```

Le deuxième type de macros sont globales : elles sont connues dès leur définition et utilisables dans tous les sous-rapports via leur nom. Leur contenu peut être modifié pendant l'exécution du rapport (ou des sous-rapports) par l'instruction $Define avec le même nom. La dernière définition recontrée annule les précédentes.

Syntaxe

```
%naam%
naam := nom de la macro (cfr. $Define)
```

On peut également les utiliser pour globaliser une variable locale et par là exploiter un paramètre de rapport dans un sous-rapport :

Exemple

```
$Define FILE %1%
$Define VAR %2%
$WsCopyVar %FILE% %VAR%
```

4.4 Les expressions LEC dans les rapports

On peut entrelarder le texte et les commandes de formules LEC : cela permet par exemple de baser des conditions de $goto sur les valeurs des variables ou des scalaires. Si des variables sont utilisées, la période de calcul correspond à la valeur actuelle de t, fixé par les commandes $SetTime et $IncrTime.

Pour indiquer une formule LEC à calculer dans un rapport, il suffit de placer celle-ci entre accolades. On peut en formatter le résultat comme suit :

Syntaxe

```
{LEC}       le résultat est formatté de manière automatique
{LEC@T}     le résultat est formatté sous forme de période (1990Y1)
{LEC@99.9}  le résultat est formatté sur 4 positions avec une décimale
{LEC@.99}   le résultat est formatté sur avec deux décimales
```

Exemple

```
$SetTime 1990Y1
Le PNB en {t@T} vaut {PNB@9999.9} milliards de FB.
```

donne comme résultat :

Exemple

```
Le PNB en 1990Y1 vaut 3089.0 milliards de FB.
```

Si le caractère '{' doit se trouver dans le texte, il faut le doubler. Ainsi,

```
{{exemple de formule lec}
```

n'est pas considéré comme une formule LEC à calculer mais est remplacé dans l'output par le texte :

```
{exemple de formule lec}
```

4.5 Valeurs d'un worksheet Excel dans les rapports

Si un worksheet Excel contient des données nécessaires à l'exécution d'un rapport, il est facile d'intégrer dynamiquement des valeurs ou des ranges de valeurs dans le texte d'un rapport.

Il suffit de placer la référence aux cellules nécessaires entre accollades dans une ligne de rapport :

```
$DataUpdateVar {=Sheet1!R2C1} %YEAR% {=Sheet1!R2C2:R3C10}
```

La première référence **{=Sheet1!R2C1}** est lue dans Excel (le fichier doit évidemment être préalablement ouvert) et est replacée dans la ligne de commande. Dans l'exemple, il s'agira du nom de la série à modifier.

La seconde référence **{=Sheet1!R2C2:R3C10}** est également remplacée par la valeur dans Excel. S'il y a plusieurs cellules, celles-ci sont séparées par des blancs.

Dans l'exemple, on aura par exemple :

```
$DataUpdateVar PNB 1990 3.23 2.34 2.56 3.12 3.45
```

Après remplacement, la commande est exécutée comme n'importe quelle autre commande de rapport.

4.6 Les Fonctions de rapports

Ces fonctions permettent de réaliser toute un série d'opérations parmi lesquelles

- le traitement de strings ou de listes de strings,
- le remplacement de texte,
- le comptage d'objets,
- le traitement de fichiers ASCII,
- l'interrogation de bases de données relationnelles, @EN les informations sur les processus de calcul
- etc

Ces fonctions s'exécutent dans le cadre d'une ligne de rapport et le résultat de la fonction est imprimé dans l'output du rapport.

Par exemple :

```
Le tableau T1 a pour titre @ttitle(T1). Ce titre en majuscules
est par conséquent @upper(@ttitle(T1)). Ce texte est imprimé
le @date() à @time().
```

Pour placer le titre du tableau T1 dans le fichier toto.txt :

```
@fappend(toto.txt, @ttitle(T1))
```

La syntaxe générale de ces fonctions est :

```
@function_name(arg1, arg2, ...)
```

Les fonctions suivantes sont définies:

Gestion de strings

- @upper(texte) : mise d'un texte en majuscules
- @lower(texte) : mise d'un texte en minuscules
- @replace(string, from, to) : substitution d'un texte par une autre
- @fmt(val,fmt) : formatte un entier
- @take(n,string) : extrait les `n` premiers caractères de `string`. Si `n` est négatif, extrait les n derniers.
- @drop(n,string) : supprime les `n` premiers caractères de `string`. Si `n` est négatif, supprime les n derniers.
- @count(string) : retourne le nombre d'éléments de string (séparés par des virgules)
- @index(n,list) : retourne l'élément `n` de `list`
- @sqz(string) : supprime les blancs de string
- @strip(string) : supprime les blancs de fin de string
- @ansi(texte) : transforme un texte Ascii en Ansi
- @equal(t1,t2) : compare t1 et t2 : retourne 1 si égaux, 0 sinon
- @void(t1,..) : ne retourne aucun texte, quels que soient les arguments

Gestion de listes de strings

- @vtake(n,list) : take the n first elements of the list (or n last elements if n is negative)
- @vdrop(n,list) : drop the n first elements of the list (or n last elements if n is negative)
- @vcount(list) : return the number of elements in the list

Gestion de fichiers

- @fdelete(filename) : détruit le fichier filename
- @fappend(filename,string|NL, ...) : écrit dans un fichier les textes

Gestion de directories

- @getdir() : retourne le directory courant
- @chdir(dirname) : change le directory courant vers *dirname* et retourne le nouveau directory courant
- @mkdir(dirname) : crée un nouveau directory *dirname*
- @rmdir(dirname) : détruit le directory *dirname*

Dates et heures

- @date([format]) : retourne la date
- @time([format]) : retourne l'heure
- @month(mois, langue) : retourne le texte du mois dans la langue donnée
- @ChronoReset() : remet le chrono à 0
- @ChronoGet() : retourne le temps écoulé (en msecs) depuis le dernier reset du chrono

Liste d'objets

- @cexpand(pattern, ...) : retourne la liste de commentaires correspondant à pattern
- @eexpand(pattern, ...) : retourne la liste d'équations correspondant à pattern
- @iexpand(pattern, ...) : retourne la liste d'identités correspondant à pattern
- @lexpand(pattern, ...) : retourne la liste de listes correspondant à pattern
- @sexpand(pattern, ...) : retourne la liste de scalaires correspondant à pattern
- @texpand(pattern, ...) : retourne la liste de tableaux correspondant à pattern
- @vexpand(pattern, ...) : retourne la liste de variables correspondant à pattern
- @vliste(objname, ...) : retourne la liste des variables dans les eqs objname
- @sliste(objname, ...) : retourne la liste des scalaires dans les eqs objname

Contenu des objets

- @ttitle(tablename,tablename, ...) : retourne les titres des tableaux
- @srelax(sclname,sclname, ...) : retourne la valeur des relax pour des scalaires
- @sstderr(sclname,sclname, ...) : retourne la valeur des stderr pour des scalaires
- @cvalue(cmtname,cmtname, ...) : retourne le texte d'un commentaire
- @vvalue(varname,varname, ...) : retourne les valeurs de variables sous forme de texte
- @sample(B|E|) : retourne le texte du sample courant

Contenu des équations

- @evalue(eqname,eqname, ...) : retourne le texte LEC d'une équation
- @eqsample(eqname) : retourne le sample d'estimation de l'équation **`eqname`**
- @eqsamplefrom(eqname) : retourne la partie FROM du sample d'estimation
- @eqsampleto(eqname) : retourne la partie TO du sample d'estimation
- @eqlhs(eqname) : retourne le membre de gauche d'une équations
- @eqrhs(eqname) : retourne le membre de droite d'une équations

Interface avec les bases de données (ODBC)

Les fonctions suivantes permettent d'interroger et/ou d'alimenter des bases de données relationnelles dans un rapport IODE:

- @SqlOpen : ouverture de la base de données
- @SqlQuery : recherche/sélection/alimentation (SQL)
- @SqlNext : enregistrement suivant
- @SqlField : valeur d'un champ
- @SqlRecord : valeur d'un enregistrement complet
- @SqlClose : fermeture de la base de données

Simulations

Les fonctions suivantes permettent d'obtenir la valeur de certains paramètres de simulation :

- @SimEps() : retourne la valeur du critère de convergence utilisé pour la dernière simulation
- @SimRelax() : retourne la valeur du paramètre de relaxation utilisé pour la dernière simulation
- @SimMaxit() : retourne la valeur du maximum d'itérations utilisé pour la dernière simulation

Les résultats par période de simulation peuvent être récupérés via les fonctions suivantes :

- @SimNiter(period) : nombre d'itérations nécessaires à la résolution du modèle à l'année period
- @SimNorm(period) : seuil de convergence atteint à la résolution du modèle à l'année period

4.6.1 Fonction @upper

Description

Cette fonction met en majuscules le texte passé comme argument.

Syntaxe

```
@upper("texte libre")
```

Exemple

```
@upper("texte libre")
```

résultat

```
TEXTE LIBRE
```

4.6.2 Fonction @lower

Description

Cette fonction met en minuscules le texte passé comme argument.

Syntaxe

```
@lower("texte libre")
```

Exemple

```
@lower("Texte Libre")
```

résultat

```
texte libre
```

4.6.3 Fonction @replace

Description

Cette fonction remplace un string par un autre. Le remplacement différencie majuscules et minuscules.

Syntaxe

```
@replace("texte libre", replacefrom, replaceby)
```

Exemple

```
@replace("texte libre",texte,cuba)
```

résultat

```
cuba libre
```

4.6.4 Fonction @fmt

Formatte un entier **val** suivant un format **fmt** donné. Le résultat est un string transformé de longueur égale à celle de **fmt.**

Les caractères reconnus dans le format sont : 'X', 'x', '9', '0'. Ils signifient qu'aux seules positions de ces caractères seront placés dans leur ordre d'apparition les caractères résultant du formattage de **val.** Seul cas particulier : le caractère **'0'** qui sera remplacé par un **'0'** si le caractère correspondant dans le formattage de **val** est ' '.

Syntaxe

```
@fmt(val,fmt)
où
val = valeur entière
fmt = format
```

EXEMPLE

```
@fmt(123,0009)    ---> 0123
@fmt(123,A0000A)    ---> A00123A
```

Attention, les blancs avant et après le format sont repris dans le résultat.

4.6.5 FONCTION @TAKE

DESCRIPTION

Extrait les **n** premiers caractères de **string**. Si **n** est négatif, extrait les n derniers.

Syntaxe

```
@take(n,texte)
```

Exemple

```
@take(3,IODE)
@take(-3,IODE)
```

résultat

```
IOD
ODE
```

4.6.6 FONCTION @DROP

DESCRIPTION

Supprime les **n** premiers caractères de **string**. Si **n** est négatif, supprime les n derniers.

Syntaxe

```
@drop(n,texte)
```

Exemple

```
@drop(2,IODE)
@drop(-2,IODE)
```

résultat

```
DE
IO
```

4.6.7 FONCTION @COUNT

Retourne le nombre d'éléments de string (séparés par des virgules) Supprime les **n** premiers caractères de **string**. Si **n** est négatif, supprime les n derniers. &DE

Syntaxe

```
@count(list)
```

Exemple

```
@count(A,B,C,E)
@count(ABC,,,B)
```

résultat

```
4
2
```

4.6.8 FONCTION @INDEX

Retourne l'élément **n** de **list**.

Syntaxe

```
@index(n,list)
```

Exemple

```
@index(2,A,B,C,E)
@index(1,ABC,,,B)
```

résultat

```
B
ABC
```

4.6.9 FONCTION @SQZ

Supprime les blancs d'un string.

SYNTAXE

```
@sqz(string)
où
string = string quelconque
```

EXEMPLE

```
@sqz(' ABC D  ')   ---> 'ABCD'
```

4.6.10 FONCTION @STRIP

Supprime les blancs de fin d'un string.

SYNTAXE

```
@strip(string)
où
string = string quelconque
```

EXEMPLE

```
@strip(' ABC D  ')   ---> ' ABC D'
```

4.6.11 FONCTION @ANSI

DESCRIPTION

Cette fonction transforme un string codé en Ascii en caractères codés en Ansi. Elle est par exemple utile lorsqu'on génére des fichiers output dont le format doit être impérativement codé en Ansi (fichiers HTML par exemple).

Syntaxe

```
@ansi(texte)
```

Exemple

```
<TC>@ansi(@month(2,F))</TC>
```

Résultat

```
<TC>Février</TC>
```

4.6.12 Fonction @equal

Vérifie si deux valeurs sont identiques.

Syntaxe

```
equal(a,b)
où
a et b = textes quelconques
```

Exemple

```
@equal(123,123)    ---> 1
@equal(123, 123)    ---> 0
@equal(%i%,10)    ---> 1 (ou 0 selon la valeur de i)
```

Attention : les espaces sont comptés comme des caractères.

4.6.13 Fonction @void(args)

Vide le texte de ses arguments, quels que soient les arguments.

Par exemple, l'appel à @chdir() retourne le nom du nouveau directory courant. Avec @void() ce texte n'apparait pas dans l'output.

```
Voici de résultat de chdir()  : @chdir(..)
Voici de résultat avec void() : @void(@chdir(..))
```

Résultat :

```
Voici de résultat de chdir()  : \usr\iode
Voici de résultat avec void() :
```

4.6.14 Fonction @vtake(n,values)

Description

Ne conserve que les n premiers strings de **`values.`** Si **`n`** est négatif, conserve les n derniers.

Les séparateurs entre les strings dans values sont définis par la commande **`$vseps`**. Par défault, il s'agit de ",", ";" et blanc.

La virgule est toujours séparateurs, même si elle n'est pas reprise dans $vseps.

Le résultat est une liste dont le séparateur est le premier de ceux définis par **$vseps**.

Syntaxe

```
@vtake(n,values)
```

Exemple

```
$show @vtake(1,A,B,C)    -> A
$show @vtake(-1,A,B,C)   -> C
$show @vdrop(-1,A,B,C)   -> A,B
$show @vdrop(2,A,B,C)    -> C
$vseps |
$show @vtake(1,A B|C)    -> A B
$show @vtake(1,"A,B"|C)  -> A,B
$show @vdrop(-1,A,B,C)   -> A|B
```

Exemple

```
$vseps ;
$define LIST A;B;C;D
$label next
$define ELEMENT @vtake(1, %LIST%)
  ... Some operation on element %ELEMENT% ...
$define LIST @vdrop(1,%LIST%)
$goto next @vcount(%LIST%)
```

4.6.15 FONCTION @VDROP(N,VALUES)

DESCRIPTION

Voir @vtake

4.6.16 FONCTION @VCOUNT(N,VALUES)

DESCRIPTION

Cette fonction retourne le nombre d'éléments dans une liste de strings.

Syntaxe

```
@vcount({liste de strings})
```

Exemple

```
$vseps |
$show @vcount(A B|C)    -> 2
$show @vcount(A;B;C)    -> 1
```

Voir aussi @vtake.

4.6.17 FONCTION @FDELETE

DESCRIPTION

Cette fonction détruit un fichier. Elle ne retourne pas de résultat.

Syntaxe

```
@fdelete(filename)
```

Exemple

```
@fdelete(test.htm)
```

résultat

```
aucun - Le fichier test.htm est détruit.
```

4.6.18 FONCTION @FAPPEND

DESCRIPTION

Cette fonction ajoute des strings au contenu d'un fichier ASCII. Elle ne retourne pas de résultat.

Syntaxe

```
@fappend(filename,text|NL,text,...)
où NL indique un saut de ligne
```

Exemple

```
Rapport test.rep
----------------
@fappend(test.htm,"Fichier de données",NL)
@fappend(test.htm,"Paramètres:",%*%)

Appel
-----
test A B C
```

résultat

```
Fichier test.htm
----------------
Fichier de données
Paramètres:A B C
```

4.6.19 Fonction @getdir()

Retourne le directory courant.

Exemple

```
Directory courant : @getdir()
```

Résultat :

```
Directory courant : c:\usr\iode
```

4.6.20 Fonction @chdir(dirname)

Change le directory courant vers *dirname* et retourne le nouveau directory courant.

Exemple

```
Dir courant : @getdir()
Nouveau dir : @chdir(..)
```

Résultat :

```
Dir courant : c:\usr\iode
Nouveau dir : c:\usr
```

4.6.21 Fonction @mkdir(dirname)

Crée un nouveau directory de nom *dirname.*

Ne retourne aucune valeur.

EXEMPLE

```
Dir courant :          @getdir()
Création de subdir     @mkdir(subdir)
Nouveau dir courant :  @getdir()
```

Résultat :

```
Dir courant :          c:\usr\iode
Création de subdir
Nouveau dir courant :  c:\usr\iode\subdir
```

4.6.22 FONCTION @RMDIR(DIRNAME)

Détruit le directory *dirname*. A utiliser avec prudence...

Ne retourne aucune valeur.

4.6.23 FONCTION @DATE

DESCRIPTION

Cette fonction retourne la date du jour.

Syntaxe

```
@date([format])
où format indique le format de la date
Par défaut, le format est dd-mm-yyyy
```

Exemple

```
La date du jour est @date()
Autre format : @date("dd/mm/yy")
```

résultat

```
La date du jour est 31-05-1999
Autre format : 31/05/99
```

4.6.24 FONCTION @TIME

DESCRIPTION

Cette fonction retourne l'heure.

Syntaxe

```
@time([format])
où format indique le format de l'heure
Par défaut, le format est hh:mm:ss
```

Exemple

```
Il est @time()
Autre format : @time("hh heures mm minutes")
```

résultat

```
Il est 23:12:55
Autre format : 23 heures 12 minutes
```

4.6.25 FONCTION @MONTH

DESCRIPTION

Cette fonction retourne le nom d'un mois dans une langue donnée.

Syntaxe

```
@month(mois[,langue])
où langue est F, N ou E (E par défaut)
```

Exemple

```
Le mois numéro 3 est : @month(3)
En français et majuscules : @upper(@month(3,F))
En néerlandais et minuscules : @lower(@month(3,N))
```

Résultat

```
Le mois numéro 3 est : March
En français et majuscules : MARS
En néerlandais et minuscules : maart
```

4.6.26 Fonction @ChronoReset()

Un chrono virtuel a été ajouté pour permettre de calculer les durées de traitement.

@ChronoReset() remet le chrono à 0.

Voir exemple dans la fonction @SimEps.

4.6.27 Fonction @ChronoGet()

Retourne le temps écoulé (en msecs) depuis le dernier appel à @ChronoReset.

Voir exemple dans la fonction @SimEps.

4.6.28 Fonction @cexpand

Description

Cette fonction retourne la liste des commentaires dont le nom correspond à un des strings passés comme argument.

Syntaxe

```
@cexpand(pattern1, pattern2, ...)
```

Exemple

```
@cexpand(A*,B*,AE)
```

Résultat

```
A1;A2;BA;BCXS;AE
```

La ligne

```
$Repeat "$DataDuplicateVar _ _1" @vexpand(*)
```

exécute la commande DataDuplicateVar sur toutes les variables du WS (les nouvelles variables portent le nom de la variable d'origine avec 1 comme suffixe).

Voir également

@eexpand(), @iexpand(), @lexpand(), @sexpand(), @texpand(), @vexpand()

4.6.29 Fonction @eexpand

Description

Cette fonction retourne la liste des équations dont le nom correspond à un des strings passés comme argument.

Syntaxe

```
@eexpand(pattern1, pattern2, ...)
```

Exemple

```
@eexpand(X*)
```

Résultat

```
Toutes les équations dont le nom commence par X
```

Voir également

@cexpand(), @iexpand(), @lexpand(), @sexpand(), @texpand(), @vexpand()

4.6.30 Fonction @iexpand

Description

Cette fonction retourne la liste des identités dont le nom correspond à un des strings passés comme argument.

Syntaxe

```
@iexpand(pattern1, pattern2, ...)
```

Exemple

```
@iexpand(*_1)
```

Résultat

```
Toutes les identités dont le nom se termine par _1
```

VOIR ÉGALEMENT

@cexpand(), @eexpand(), @lexpand(), @sexpand(), @texpand(), @vexpand()

4.6.31 FONCTION @LEXPAND

DESCRIPTION

Cette fonction retourne la liste des listes dont le nom correspond à un des strings passés comme argument.

Syntaxe

```
@lexpand(pattern1, pattern2, ...)
```

Exemple

```
$DataUpdateLst list1 @lexpand(_*)
```

Crée une liste `list1` contenant toutes les listes commençant par _.

VOIR ÉGALEMENT

@cexpand(), @eexpand(), @iexpand(), @sexpand(), @texpand(), @vexpand()

4.6.32 FONCTION @SEXPAND

DESCRIPTION

Cette fonction retourne la liste des scalaires dont le nom correspond à un des strings passés comme argument.

Syntaxe

```
@sexpand(pattern1, pattern2, ...)
```

Exemple

```
@sexpand(*)
```

retourne la liste de tous les scalaires du WS.

VOIR ÉGALEMENT

@cexpand(), @eexpand(), @iexpand(), @lexpand(), @texpand(), @vexpand()

4.6.33 FONCTION @TEXPAND

DESCRIPTION

Cette fonction retourne la liste des tableaux dont le nom correspond à un des strings passés comme argument.

Syntaxe

```
@texpand(pattern1, pattern2, ...)
```

Exemple

```
$PrintTbl 1990:5 @texpand(*)
```

Imprime tous les tableaux du WS sur le sample 1990 à 1994.

VOIR ÉGALEMENT

@cexpand(), @eexpand(), @iexpand(), @lexpand(), @sexpand(), @vexpand()

4.6.34 FONCTION @VEXPAND

DESCRIPTION

Cette fonction retourne la liste des variables dont le nom correspond à un des strings passés comme argument.

Syntaxe

```
@vexpand(pattern1, pattern2, ...)
```

Exemple

```
$DataUpdateTbl Newtbl TITRE;@vexpand(BE*)
```

Crée un tableau **`Newtbl`** contenant toutes les séries commençant par BE.

VOIR ÉGALEMENT

@cexpand(), @eexpand(), @iexpand(), @lexpand(), @sexpand(), @texpand()

4.6.35 FONCTION @VLISTE

DESCRIPTION

Cette fonction retourne la liste des variables utilisées dans les équations dont les noms sont passés comme argument.

Syntaxe

```
@vliste(eq1,eq2,...)
```

Exemple

```
$DataUpdateLst VarLst @vliste(@eexpand(*))
```

Crée la liste **VarLst** contenant toutes les séries utilisées dans les équations du WS.

VOIR ÉGALEMENT

@sliste()

4.6.36 FONCTION @SLISTE

DESCRIPTION

Cette fonction retourne la liste des scalaires utilisés dans les équations dont les noms sont passés comme argument.

Syntaxe

```
@sliste(eq1,eq2,...)
```

Exemple

```
$DataUpdateLst SCLLST @sliste(@eexpand(*))
```

Crée la liste **SCLLST** contenant tous les scalaires utilisés dans les équations du WS.

VOIR ÉGALEMENT

@vliste()

4.6.37 FONCTION @TTITLE

DESCRIPTION

Cette fonction retourne les titres des tableaux passés comme arguments.

Syntaxe

```
@ttitle(tbl1,tbl2,...)
```

Exemple

```
@ttitle(tbl)
```

retourne par exemple

```
Titre du tableau
```

4.6.38 FONCTION @SRELAX

DESCRIPTION

Cette fonction retourne les valeurs des paramètres de relaxation des scalaires passés comme arguments.

Syntaxe

```
@srelax(scl1,scl2,...)
```

Exemple

```
@srelax(scl)
```

retourne par exemple

```
0.9
```

4.6.39 FONCTION @SSTDERR

DESCRIPTION

Cette fonction retourne les valeurs des stderr des scalaires passés comme arguments.

Syntaxe

```
@sstderr(scl1,scl2,...)
```

Exemple

```
@sstderr(scl1)
```

retourne par exemple

```
0.001
```

4.6.40 FONCTION @CVALUE

DESCRIPTION

Cette fonction retourne le texte des commentaires dont les noms sont passés comme arguments.

Syntaxe

```
@cvalue(cmtname,cmtname, ...)
```

Exemple

```
@cvalue(CMT1,A)
```

retourne par exemple

```
Commentaire 1;Commentaire de A
```

4.6.41 FONCTION @VVALUE

DESCRIPTION

Cette fonction retourne les valeurs formattées des variables dont les noms sont passés comme

arguments.

Syntaxe

```
@vvalue(varname,varname, ...)
```

Exemple

```
@vvalue(A,B)
```

retourne par exemple

```
  na na 1.2342 -1.22323 1000 1001 1002 1003
```

4.6.42 FONCTION @SAMPLE

DESCRIPTION

Cette fonction retourne le sample courant.

Syntaxe

```
@sample(B|E|)
    B : première période
    E : dernière période
    nil : les deux
```

Exemple

```
@sample()
@sample(B)
@sample(E)
```

retourne par exemple :

```
  1960Y1 1990Y1
  1960Y1
  1990Y1
```

4.6.43 FONCTION @EVALUE

DESCRIPTION

Cette fonction retourne la formule LEC définissant une équation.

Syntaxe

```
@evalue(eqname,eqname, ...)
```

Exemple

```
@evalue(EQ1)
```

retourne par exemple

```
log(EQ1) := c1 + c2 * V3 + c3 * X
```

4.6.44 FONCTION @EQSAMPLE(EQNAME)

Retourne le sample d'estimation de l'équation **eqname**

4.6.45 FONCTION @EQSAMPLEFROM(EQNAME)

Retourne la partie FROM du sample d'estimation.

Dans l'exemple suivant, on extrait la première année d'estimation de l'équation ENDO1 pour réestimer l'équation sur un sample prolongé à droite par exemple suite à l'obtention . de nouvelles observations.

```
$EqsEstimate @eqsamplefrom(ENDO1) 2012Y1 ENDO1
```

4.6.46 FONCTION @EQSAMPLETO(EQNAME)

Retourne la partie TO du sample d'estimation.

Voir exemple dans @EqSampleFrom.

4.6.47 FONCTION @EQLHS(EQNAME)

Retourne le membre de gauche d'une équation.

4.6.48 FONCTION @EQRHS(EQNAME)

Retourne le membre de droite d'une équation.

4.6.49 Fonction @SqlOpen

Description

Opent een ODBC-sessie met een databank. De naam van de databank (DSN-naam) die u doorgeeft is de naam waarmee de databank in uw ODBC-omgegeving is gedefiniëerd (Start > Settings > Control Panel > ODBC 32). Voor sommige databanken is er een user-naam en een paswoord vereist.

Bij success krijgt u "1" als resultaat anders "0".

Fonction interactive

Deze functie heeft geen Fullscreen tegenhanger

Fonction non interactive

Syntaxe

```
@SqlOpen(DSN-naam[,User-naam,Paswoord])
```

Exemple

```
$goto continue, @SqlOpen(RSZ)
$Show  ODBC-Databank not opened
$Return

$label continue
$Show  ODBC-Databank opened
....

@SqlClose()
```

4.6.50 Fonction @SqlQuery

Description

Dit commando voert een SQL-query opdracht uit op uw databank. Als die opdracht lukt wordt er een cursor gecrëeerd op de "Dynaset" die u kan doorlopen met SqlNext, tot het einde van de beschikbare data. Plaats de SQL-opdracht tussen quotes zodat er geen verwarring kan zijn als u speciale tekens zoals "," of ")" gebruikt in uw SQL-opdracht. Groepeer ook de volledige opdracht in één lijn.

Bij success krijgt u het aantal kolommen van de "DynaSet" als resultaat, anders "0".

U moet @SqlNext() gebruiken na een @SqlQuery() indien u de velden van een record wil gebruiken.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
@SqlQuery("SQL-opdracht")
```

Exemple

```
$goto continue, @SqlOpen(RSZ)
$Show  ODBC-Databank not opened
$Return

$label continue
$Show  ODBC-Databank opened
$Show  @SqlQuery("SELECT DISTINCT TRIM FROM RSZ_bruto;")
....

@SqlClose()
```

4.6.51 FONCTION @SQLNEXT

DESCRIPTION

Dit commando doorloopt de "Dynaset" gecrëeerd door de laatste SQL-opdracht. Zolang er nog records beschikbaar zijn in de "Dynaset" krijgt u "1" als resultaat anders "0".

Bij success krijgt u "1" als resultaat anders "0".

U moet @SqlNext() gebruiken na een @SqlQuery() indien u de velden van een record wil gebruiken.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
@SqlNext()
```

Exemple

```
$goto continue, @SqlOpen(RSZ)
$Show  ODBC-Databank not opened
$Return

$label continue
$Show  ODBC-Databank opened
$Show  @SqlQuery("SELECT DISTINCT TRIM FROM RSZ_bruto;")
@SqlNext()
$Define BFLD @SqlField(0)

$label again_date
$Show Skipping @SqlField(0)
$goto again_date, @SqlNext()
$Define EFLD @SqlField(0)

$Define BEGIN @replace(%BFLD%,/,Q)
$Define END   @replace(%EFLD%,/,Q)

$WsSample %BEGIN% %END%
....

@SqlClose()
```

4.6.52 FONCTION @SQLFIELD

DESCRIPTION

Met dit commando krijgt u de inhoud van het veld "n" van de huidige record.

Bij success krijgt u de inhoud van het veld anders "0".

U moet @SqlNext() gebruiken voor SqlField() of SqlRecord() werkt.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
@SqlField(veld-nummer)
```

Exemple

```
$goto continue, @SqlOpen(RSZ)
$Show  ODBC-Databank not opened
$Return

$label continue
$Show  ODBC-Databank opened
$Show  @SqlQuery("SELECT DISTINCT TRIM FROM RSZ_bruto;")
@SqlNext()
$Define BFLD @SqlField(0)

$label again_date
$Show Skipping @SqlField(0)
$goto again_date, @SqlNext()
$Define EFLD @SqlField(0)

$Define BEGIN @replace(%BFLD%,/,Q)
$Define END   @replace(%EFLD%,/,Q)

$WsSample %BEGIN% %END%
....

@SqlClose()
```

4.6.53 FONCTION @SQLRECORD

DESCRIPTION

Met dit commando krijgt u de inhoud van de huidige record. U krijgt alle velden als u geen argumenten aan de functie doorspeelt, of alle velden vanaf een bepaalde kolom bij één argument of alle velden tussen 2 kolommen als u twee argumenten doorgeeft.

Bij success krijgt u de inhoud van de velden anders "0".

U moet @SqlNext() gebruiken voor SqlField() of SqlRecord() werkt.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
@SqlRecord() : alle velden
@SqlRecord(veld1-nummer) : alle velden vanaf veld1
@SqlRecord(veld1-nummer,veld2-nummer) : alle velden tussen veld1 en veld2
```

Exemple

```
      $goto continue, @SqlOpen(RSZ)
      $Show  ODBC-Databank not opened
      $Return

      $label continue
      $Show  ODBC-Databank opened

      $Define PIVOT RETOMA_DBF
      $Define PREP  RT

      $Msg  NBCOLS @SqlQuery("TRANSFORM Sum(RSZ_bruto.%PIVOT%) AS AGGR SELECT
  RSZ_bruto.CODRED, NaceTbl.NACE FROM NaceTbl INNER JOIN RSZ_bruto ON NaceTbl.FICTIF
  =RSZ_bruto.FICTIF GROUP BY RSZ_bruto.CODRED, NaceTbl.NACE PIVOT RSZ_bruto.TRIM;")
      $Show  @SqlNext()
      $label again
      $DataUpdateVar %PREP%@SqlField(0)@SqlField(1) %BEGIN% @SqlRecord(2, 8)
      $goto again, @SqlNext()

      ...
      @SqlClose()
```

4.6.54 FONCTION @SQLCLOSE

DESCRIPTION

Sluit de huidige ODBC-sessie.

Bij success krijgt u "1" als resultaat anders "0".

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
      @SqlClose()
```

Exemple

```
      $goto continue, @SqlOpen(RSZ)
      $Show  ODBC-Databank not opened
      $Return

      $label continue
      $Show  ODBC-Databank opened
      ....

      @SqlClose()
```

4.6.55 FONCTION @SIMEPS

Retourne la valeur du critère de convergence utilisé pour la dernière simulation.

Exemple

L'exemple ci-dessous compare les performances selon le nombre de tri de l'algorithme de simulation.

```
$ Parameters
$ ----------
$define modeldir C:\iode\nemesis
$define model base_neujobs_rd
$define nbtri 5
$define simfrom 2011Y1
$define simto 2012Y1
$define simper %simfrom% %simto%
$ModelSimulateParms 0.00001 0.6 500 Both 0 No 1

$ Output file
$ -----------
$PrintDest compare.html HTML

$ Load fresh files
$ ----------------
$WsLoadVar %modeldir%\%model%
$WsLoadScl %modeldir%\%model%
$WsLoadEqs %modeldir%\%model%

$ ==== $ModelCalcSCC ===
.par1 tit_1
Comparaison entre les performances selon le nombre de tri

$ ==== Loop on Tri tests ===
$define i 0
$label nexttri

$ Calcul SCC
$ ----------
$show $ModelCalcSCC %i%
@ChronoReset()
$ModelCalcSCC %i% pre inter post
$define cpu1  @ChronoGet()

$ Simulate
$ --------
$ Reload vars for a clean start and modify exo
$show Reload Vars ...
$WsLoadVar %modeldir%\%model%
$DataUpdateVar RPOIL 2011Y1 1.20

$show $ModelSimulateSCC  %simper% pre inter post
@ChronoReset()
$ModelSimulateSCC %simper% pre inter post
$define cpu2 @ChronoGet()

$ Reporting
$ ---------
.par1 enum_1
Tris : %i%
.par1 enum_2
Calc SCC : %cpu1% msec
.par1 enum_2
Simulation (eps = @simeps(); maxit=@simmaxit()) : %cpu2% msec

$define j {%simfrom%}
```

```
$define totit 0

$label nextsimper
.par1 enum_3
{%j%@T} : conv = @simnorm({%j%@T}), niter = @simniter({%j%@T})
$define simit @simniter({%j%@T})
$define totit {%totit% + %simit%}
$define j {%j% + 1}
$goto nextsimper {%j% <= %simto%}
.par1 enum_3
Total iterations : %totit%
$define i {%i% + 1}
$goto nexttri {%i% <= %nbtri%}
```

4.6.56 Fonction @SimRelax()

Retourne la valeur du paramètre de relaxation utilisé pour la dernière simulation.

Voir exemple dans la fonction @SimEps.

4.6.57 Fonction @SimMaxit()

Retourne la valeur du maximum d'itérations utilisé pour la dernière simulation.

Voir exemple dans la fonction @SimEps.

4.6.58 Fonction @SimNiter(period)

Retourne le nombre d'itérations nécessaires à la résolution du modèle à l'année **`period.`**

Voir exemple dans la fonction @SimEps.

4.6.59 Fonction @SimNorm(period)

Retourne le seuil de convergence atteint à la résolution du modèle à l'année **`period.`**

Voir exemple dans la fonction @SimEps.

4.7 Interprétation des lignes de rapports

Une ligne de rapport est interprétée avant d'être (éventuellement) exécutée. Cette interprétation se fait de la façon suivante:

L'ordre d'interprétation est le suivant :

Les caractères spéciaux sont : **%, { et @.**

Chaque ligne est interprétée de gauche à droite. Dès qu'un des caractères spéciaux est rencontré, un traitement particulier est appliqué.

Si on rencontre % :

- si le suivant est %, un seul % est conservé comme du texte

```
Exemple : augmentation de 10%% du PNB -> un seul % reste dans le texte
```

- sinon, la macro est remplacée par sa valeur ou vide si la macro n'existe pas :

```
Exemple : la variable %VAR% -> la variable XYZ
```

Si on rencontre { :

- si le suivant est =, le contenu entre accollades est considéré comme une référence Excel et remplacé
- si le suivant est {, un seul { est conservé comme du texte le texte est lu jusqu'à }, les macros sont remplacées
  - si le texte résultat commence par $ ou #, il s'agit d'une commande de rapport qui est exécutée et le résultat (0 ou 1) se retrouve dans le texte.
  - sinon, le texte résultat est calculé comme une formule LEC à la période courante définie par $SetTime. Si la formule se termine pas @T ou @999.999, le résultat est formaté en conséquence.

Si on rencontre @ :

si le suivant est @, un seul @ est conservé comme du texte le texte est lu jusqu'à la parenthèse fermante la fonction correspondante est exécutée. A noter qu'en l'absence de parenthèses, le texte reste inchangé (Ex.: gb@plan.be reste tel quel).

```
$define VAL 123.123
$msg {%VAL%@999.9} gb@plan.be
```

Donne :

```
123.1 gb@plan.be
```

4.8 Les Commandes de IODE dans les Rapports

Les commandes IODE permettent d'exécuter pratiquement toutes les fonctions de IODE. Elles

sont regroupées par fonction:

- Opérations sur des fichiers
- Opérations sur les WS
- Opérations sur les données
- Opérations spécifiques aux équations
- Configuration de l'imprimante
- Impressions d'objets
- Compilation et impression de tables
- Graphiques à partir de tableaux
- Opérations sur des modèles
- Exécutions d'identités
- Opérations sur des rapports
- Interface Excel
- Interface Datastream
- Traduction des fichiers A2M
- Autres fonctions de rapports

Pour rappel également, les commandes RAPPORT et les commandes IODE peuvent être des commandes plein écran ou des lignes de commande (sans interface écran).

Les commandes plein écran sont constituées d'un mot clé précédé par le caractère #. Elles ont la syntaxe suivante :

```
#commande
```

Par exemple, la commande #WsLoadVar provoquera l'affichage d'un écran de saisie permettant de préciser le nom du fichier de variables à charger. Le fonctionnement de ces commandes est en général décrit de manière détaillée dans le "Manuel de l'utilisateur". Lorsque c'est le cas, il y a lieu de se reporter à ce manuel.

Les commandes en ligne sont constituées d'un mot clé précédé par le caractère $. Elles ont la syntaxe suivante :

```
$commande options paramètres
```

Par exemple, la commande

```
$PrintTbl 84:8 TEST
```

imprimera le tableau TEST pour le sample indiqué.

Les mots clés sont des mnémoniques des fonctions exécutées. Par exemple le mot clé WsCopyScl effectue une copie (Copy) d'un Work Space (Ws) de scalaires (Scl). DataDuplicateIdt dédouble (Duplicate) une donnée (Data) du WS des identités (Idt).

Une table résumée de la syntaxe de toutes les commandes de rapports peut être trouvée en annexe.

4.8.1 Opérations sur des fichiers

Les commandes opérant sur des fichiers permettent de copier, éditer, effacer, renommer, imprimer, etc, tout fichier d'un type reconnu par IODE. Les commandes sont les suivantes:

- filelist
- fileprint
- fileedit
- filedelete
- filerename
- filecopy
- SysCopyFile
- SysMoveFile
- SysDeleteFile
- SysAppendFile
- FileImportVar
- FileImportCmt

4.8.1.1 FileList

Fonction annulée à partir de la version 5 de IODE.

4.8.1.2 FilePrint

Description

Cette commande permet d'imprimer un fichier.

Le fichier doit être un fichier texte.

La fonction imprime le fichier sans interprétation sur l'imprimante définie (voir Print Destination). Le fichier est simplement ajouté au fichier-imprimante sélectionné.

Fonction interactive

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

Le nom du fichier est indiqué dans le champ de saisie de la page. Le champ de saisie est du type DIR. La touche F10 valide l'opération lorsque la page est complétée.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#FilePrint
```

FONCTION NON INTERACTIVE

Le nom du fichier (ou un masque) est passé en paramètre de la commande.

Syntaxe

```
$FilePrint filename
```

Exemple

```
$FileDeleteA2m  imp
$PrintDest      imp.a2m
$FilePrint      bist92\read.me
$FileEdit       imp.a2m
$show           End of report...
```

4.8.1.3 FILEEDIT

Fonction annulée à partir de la version 5 de IODE.

4.8.1.4 FILEDELETEXXX

Xxx prend l'une des valeurs :

```
cmt,    eqs,    idt,    lst,    scl,    tbl,    var,
ac,     ae,     ai,     al,     as,     at,     av,
rep,    a2m,    agl,    prf,    dif,    mif,    rtf,
ps,     asc,    txt,
```

DESCRIPTION

La fonction efface un fichier sur disque portant une des extensions suivantes :

```
cmt,    eqs,    idt,    lst,    scl,    tbl,    var,
ac,     ae,     ai,     al,     as,     at,     av,
rep,    a2m,    agl,    prf,    dif,    mif,    rtf,
ps,     asc,    txt
```

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

La page de saisie contient deux champs: le premier est un champ MENU (voir lexique): on y sélectionne le type de fichier à effacer (le type choisi est proposé par défaut); le second est un champ DIR (voir lexique): on y spécifie le nom du fichier à effacer.

Le fichier est effacé lorque la page est validée avec F10.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#FileDeleteCmt
```

FONCTION NON INTERACTIVE

On passe en paramètre la liste des fichiers portant l'extension choisie à effacer.

Syntaxe

```
$FileDeleteCmt fichier [liste de fichiers]
```

Exemple

```
$FileDeleteA2m  imp
$PrintDest      imp.a2m

$PrintObjDefEqs
$FileEdit       imp.a2m
$show End of report...
```

4.8.1.5 FILERENAMEXXX

Xxx prend l'une des valeurs :

```
cmt,    eqs,    idt,    lst,    scl,    tbl,    var,
ac,     ae,     ai,     al,     as,     at,     av,
rep,    a2m,    agl,    prf,    dif,    mif,    rtf,
ps,     asc,    txt,
```

DESCRIPTION

La fonction renomme un fichier sur disque portant une des extensions :

```
cmt,    eqs,    idt,    lst,    scl,    tbl,    var,
ac,     ae,     ai,     al,     as,     at,     av,
rep,    a2m,    agl,    prf,    dif,    mif,    rtf,
ps,     asc,    txt
```

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

Le type du fichier est spécifié dans le champ supérieur de la page (champ MENU) le nom du fichier source et le nom du fichier destination sont spécifiés dans les deux autres champs.

La touche F10 valide l'opération lorsque la page est complétée.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#FileRenameCmt
```

FONCTION NON INTERACTIVE

On passe en paramètre le nom du fichier source (ancien nom) et le nom du fichier destination (nouveau nom).

Syntaxe

```
$FileRenameCmt fichier_source fichier_dest
```

Exemple

```
$FileRenameCmt  maribel old
```

4.8.1.6 FILECOPYXXX

Xxx prend l'une des valeurs :

```
cmt,    eqs,    idt,    lst,    scl,    tbl,    var,
ac,     ae,     ai,     al,     as,     at,     av,
rep,    a2m,    agl,    prf,    dif,    mif,    rtf,
ps,     asc,    txt,
```

DESCRIPTION

La fonction copie un fichier portant une des extensions suivantes :

```
cmt,    eqs,    idt,    lst,    scl,    tbl,    var,
ac,     ae,     ai,     al,     as,     at,     av,
rep,    a2m,    agl,    prf,    dif,    mif,    rtf,
ps,     asc,    txt
```

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

Le type du fichier est spécifié dans le champ supérieur de la page (champ MENU) le nom du fichier source et le nom du fichier destination sont spécifiés dans les deux autres champs.

La touche F10 valide l'opération lorsque la page est complétée.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#FileCopyCmt
```

FONCTION NON INTERACTIVE

On passe en paramètre le nom du fichier source (ancien nom) et le nom du fichier destination (nouveau nom).

Syntaxe

```
$FileCopyCmt fichier_source fichier_dest
```

Exemple

```
$FileCopyCmt  matibel old
```

4.8.1.7 SysCopyFile

Cette fonction permet de copier un fichier quelconque sur disque. Contrairement à la fonction **`FileCopyXxx`**, l'extension peut être quelconque. Attention, le fichier à copier doit être fermé pour pouvoir être copié.

En cas d'impossibilité, la fonction retourne un code d'erreur.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$SysCopyFile filein fileout
```

Exemple

```
$PrintDest indprix.a2m a2m
...
$PrintDest test a2m         (pour fermer le fichier indprix.a2m)
$SysCopyFile indprix.a2m indprix.htm
```

4.8.1.8 SysMoveFile

Cette fonction permet de renommer un fichier quelconque sur disque. Contrairement à la fonction **FileRenameXxx,** l'extension peut être quelconque. Attention, le fichier à renommer doit être fermé. Le fichier destination ne peut exister.

En cas d'impossibilité, la fonction retourne un code d'erreur.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$SysMoveFile filein fileout
```

Exemple

```
$PrintDest indprix.a2m a2m
...
$PrintDest test a2m         (pour fermer le fichier indprix.a2m)
$SysMoveFile indprix.a2m indprix.htm
```

4.8.1.9 SysDeleteFile

Cette fonction permet de détruire un fichier quelconque sur disque. Contrairement à la fonction **FileDeleteXxx,** l'extension peut être quelconque.

Si le fichier n'existe pas, la fonction NE retourne PAS de code d'erreur.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$SysDeleteFile file1 file2 ...
```

Exemple

```
$PrintDest indprix.a2m a2m
...
$PrintDest test a2m          (pour fermer le fichier indprix.a2m)
$SysDeleteFile indprix.htm
$SysMoveFile indprix.a2m indprix.htm
```

4.8.1.10 SysAppendFile

Cette fonction permet de copier un fichier à la suite d'une autre. En cas d'impossibilité, la fonction retourne un code d'erreur.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$SysAppendFile filein fileout
```

Exemple

```
$PrintDest tmp.a2m a2m
...
$PrintDest test a2m                (pour fermer le fichier tmp.a2m)
$SysAppendFile tmp.a2m result.a2m
$PrintDest tmp.a2m a2m
...
$PrintDest test a2m                (pour fermer le fichier tmp.a2m)
$SysAppendFile tmp.a2m result.a2m (accumule les tmp.a2m dans result.a2m)
```

4.8.1.11 FileImportVar

DESCRIPTION

Cette fonction concerne les importations de VARIABLES en différents formats comme DIF, ASCII, rotated ASCII, DIF Belgostat, TXT Belgostat, PRN Aremos et GEM (Belgostat).

Voir Import pour plus d'informations.

FONCTION INTERACTIVE

#FileImportVar

FONCTION NON INTERACTIVE

Syntaxe

```
$FileImportVar format rule infile outfile from to  [trace]
```

où

```
format = {A, R, D, N, G, P, T}
rule = rulefile to use to translate names
infile = file to import
outfile = IODE-file with imported series
from = begin of sample
to   = end of sample
trace = debug file (optional)
```

EXEMPLE

```
$FileImportVar TXT bstrule.txt bf-06obs.txt bh\p6y.var 1980Y1 2000Y1 p6y.log
$FileImportVar GEM rule.gem infile.gem iode\test.var 1980Y1 1990Y1
```

4.8.1.12 FILEIMPORTCMT

DESCRIPTION

Cette fonction concerne les importations de commentaires en différents formats comme DIF, ASCII, rotated ASCII, DIF Belgostat, TXT Belgostat et PRN Aremos.

Voir Import pour plus d'informations.

FONCTION INTERACTIVE

#FileImportCmt

FONCTION NON INTERACTIVE

Syntaxe

```
$FileImportCmt format rule infile outfile language [trace]
```

où

```
format = {Ascii, Rotated_Ascii, DIF, NIS, GEM, PRN, TXT_Belgostat}
rule = rulefile to use to translate names
infile = file to import
outfile = IODE-file with imported series
language =  {E,F,D}
trace = debug file (optional)
```

EXEMPLE

```
$FileImportCmt TXT bstrule.txt bf-06ser.txt bh\p6d.cmt D p6d.log
```

4.8.2 OPÉRATIONS SUR LES WS

Ces fonctions permettent de joindre, copier, charger et sauver les fichiers work spaces :

- WsLoad : charge un ws cmt | eqs | idt | lst | scl | tbl | var
- WsCopy : copie un ws cmt | eqs | idt | lst | scl | tbl | var
- WsMerge : fusionne un ws cmt | eqs | idt | lst | scl | tbl | var
- WsClear : détruit un ws cmt | eqs | idt | lst | scl | tbl | var
- WsClearAll : vide tous les ws
- WsDescr : décrit un ws cmt | eqs | idt | lst | scl | tbl | var
- WsSave : sauve un ws cmt | eqs | idt | lst | scl | tbl | var
- WsSaveCmp : sauve un ws en comprimant cmt | eqs | idt | lst | scl | tbl | var
- WsSample : change le sample du WS
- WsExtrapolate : prolonge des séries par extrapolation
- WsLtohStock : construit des séries de périodicité supérieure pour des stocks
- WsLtohFlow : construit des séries de périodicité supérieure pour des flux
- WsHtolSum : construit des séries de périodicité inférieure (somme)
- WsHtolMean : construit des séries de périodicité inférieure (moyenne)
- WsHtolLast : construit des séries de périodicité inférieure (dernière obs)
- WsSeasonAdj : construit des séries désaisonnalisées
- WsSeasAdj : sélectionne le critère de saisonnalisation
- WsTrend : calcul de séries de trend (Hodrick-Prescott)
- WsImport : importation d'un fichier ASCII
- WsExport : exportation en format ASCII
- WsImportEviews : importation d'équations et scalaires E-Views

4.8.2.1 WSLOADXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

L'opération de chargement (LOAD) d'un workspace remplace le contenu du WS courant par le contenu d'un fichier disque : tous les objets actuellement définis sont détruits et remplacés par ceux définis dans le fichier à charger.

FONCTION INTERACTIVE

Cette fonction ouvre un écran dans lequel on peut entrer un nom de fichier par type d'objet. Au départ, les noms sont ceux des WS courants.

Syntaxe

```
#WsLoadCmt
#WsLoadEqs
#WsLoadIdt
#WsLoadLst
#WsLoadScl
#WsLoadTbl
#WsLoadVar
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

On passe en paramètre à la commande le nom du fichier à charger.

Syntaxe

```
$WsLoadCmt fichier
$WsLoadEqs fichier
$WsLoadIdt fichier
$WsLoadLst fichier
$WsLoadScl fichier
$WsLoadTbl fichier
$WsLoadVar fichier
```

Exemple

```
$WsLoadCmt maribel
```

Si l'on charge un nouveau fichier de variables, la période d'échantillonnage du WS est remplacée par celle du fichier chargé.

4.8.2.2 WSCOPYXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

La fonction permet d'ajouter ou de remplacer des objets à partir de WS sauvés sur disque.

Les objets sélectionnés sont copiés dans le WS courant et remplacent les valeurs actuelles. Le processus s'arrête dès qu'un objet ne peut être trouvé.

Le WS courant garde intacts tous les autres objets qui en font partie.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

Les paramètres sont le type d'objet à traiter, le nom du ou des fichiers contenant les WS où se trouvent les objets à copier, séparés par une virgule ou point-virgule (sans blancs) et enfin la liste des noms d'objets.

S'il s'agit de fichiers de variables, la période à copier peut également être choisie.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#WsCopyCmt
#WsCopyEqs
#WsCopyIdt
#WsCopyLst
#WsCopyScl
#WsCopyTbl
#WsCopyVar
```

FONCTION NON INTERACTIVE

On passe en paramètre à la commande le nom du fichier WS sur disque et la liste des objets à copier.

Si on copie des variables (VAR) d'un autre fichier, le sample intervient de la façon suivante:

- si aucun sample n'est défini: celui du fichier à copier est utilisé
- si un ws de variables est en cours d'utilisation, seules les données du nouveau fichier qui se situent dans la période d'échantillonnage courante sont copiées
- on peut également définir la période d'échantillonnage dans laquelle seront copiées les données du fichier.

Si des séries n'existent pas dans le ws courant, elles sont créées.

Syntaxe

```
$WsCopyCmt fichier;fichier;.. {objet | liste d'objets}
                              {cmt | eqs | idt | lst | scl | tbl}
```

Exemple

```
$WsCopyCmt maribel.cmt CMT1 $CMT
```

Syntaxe

```
$WsCopyVar file;file;.. [from to] {objet |liste d'objets }
(from, to := periode)
```

Exemple

```
$WsCopyVar maribel 1990Y1 2000Y1 A
```

copie la variable A du fichier maribel dans le WS pour la période 1990 à 2000 seulement. Si A n'existe pas, elles est créée et seules les valeurs pour la période seront fixées.

Exemple

```
$WsCopyVar maribel;hermes  A B C
```

idem, mais sur tout le sample du fichier maribel.

4.8.2.3 WsMergeXxx

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Cette fonction réalise l'opération de fusion de plusieurs fichiers avec le WS courant : elle ajoute dans le WS actif les objets définis dans un fichier.

FONCTION INTERACTIVE

Cette fonction ouvre un écran dans lequel on peut entrer le type d'objet et le nom du fichier à fusionner.

Syntaxe

```
#WsMergeCmt
#WsMergeEqs
#WsMergeIdt
#WsMergeLst
#WsMergeScl
#WsMergeTbl
#WsMergeVar
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

On passe en paramètre à la commande le nom du fichier.

Syntaxe

```
$WsMergeXxx fichier
{Xxx = cmt | eqs | idt | lst | scl | tbl | var}
```

Exemple

```
$WsMergeCmt mycmts
```

4.8.2.4 WSCLEARXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Cette fonction détruit tous les objets du type spécifié.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

La fenêtre présente pour chaque type de WS le nombre d'objets actuellement définis dans le WS.

Presser sur le bouton correspondant au type souhaité ou sur le bouton "Clear All".

Syntaxe

```
#WsClearCmt
#WsClearEqs
#WsClearIdt
#WsClearLst
#WsClearScl
#WsClearTbl
#WsClearVar
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

On ne passe pas de paramètre.

Syntaxe

```
$WsClearXxx
{Xxx = cmt | eqs | idt | lst | scl | tbl | var}
```

Exemple

```
$WsClearEqs
```

4.8.2.5 WsClearAll

Maakt alle WS-en leeg.

4.8.2.6 WsDescrXxx

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

Description

Cette fonction crée ou remplace la description du fichier. Cette description sera sauvée ultérieurement si l'on sauve le fichier.

Fonction interactive

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

La fenêtre présente pour chaque type de WS le commentaire associé au WS. Si le WS a été chargé à partir du disque, le commentaire est celui provenant du fichier d'origine.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#WsDescrCmt
#WsDescrEqs
#WsDescrIdt
#WsDescrLst
#WsDescrScl
#WsDescrTbl
#WsDescrVar
```

Fonction non interactive

On passe en paramètre à la description du fichier.

Syntaxe

```
$WsDescrXxx texte
{Xxx = cmt | eqs | idt | lst | scl | tbl | var}
```

Exemple

```
$WsDescrCmt  Le fichier de commentaires de MIRABEL
```

4.8.2.7 WsSaveXxx

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Cette fonction réalise l'opération inverse de la fonction LOAD : elle copie sur disque le WS dans son état actuel, en remplaçant éventuellement les fichiers existants.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

La fenêtre présente pour chaque type de WS

- le nom du fichier associé au WS. Par défaut, ce nom est ws.type où type indique le type d'objet. Ce nom peut être modifié par la fonction "Name Workspace". Si le WS a été chargé à partir du disque, le nom du WS est le nom du fichier d'origine.
- le nombre d'objets actuellement définis dans le WS

Le nom du fichier peut être modifié avant le sauvetage. Un nom vide ne donnera pas lieu à un sauvetage. Les WS courants ne sont pas modifiés par cette fonction.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#WsSaveCmt
#WsSaveEqs
#WsSaveIdt
#WsSaveLst
#WsSaveScl
#WsSaveTbl
#WsSaveVar
```

FONCTION NON INTERACTIVE

On passe en paramètre à la commande le nom du fichier à sauver.

Syntaxe

```
$WsSaveXxx fichier
{Xxx = cmt | eqs | idt | lst | scl | tbl | var}
```

Exemple

```
$wssavecmt  current2
```

4.8.2.8 WSSAVECMP

Les différents WS peuvent être comprimés au moment de leur sauvetage. Selon le type d'objet et leur définition, la compression peut aller de 30 à 75%.

Chaque objet étant comprimé individuellement, le coût au niveau du temps de sauvetage peut être important pour les fichiers contenant de nombreux objets ou sur des machines lentes (1000 variables de 25 obs. à la sec. sur un Pentium III 450 MHz).

Iode reconnaît et transforme automatiquement les fichiers comprimés.

Le chargement n'est pratiquement pas pénalisé par la procédure de décompression. Un seul cas fait exception : les fichiers de variables dont les séries doivent être chargées par une fonction de type $WsCopyVar: dans ce cas, la lecture est ralentie par le fait que les longueurs des séries comprimées deviennent variables et que le fichier doit par conséquent être lu séquentiellement.

Le panneau de sauvetage présente un checkbox qui permet d'indiquer si on souhaite ou non comprimer les fichiers.

Cette fonction est identique à WsSave à ceci près qu'elle comprime les fichiers en sauvant les données.

Syntaxe

```
$WsSaveCmpXxx fichier
{Xxx = cmt | eqs | idt | lst | scl | tbl | var}
```

4.8.2.9 WSSAMPLE

DESCRIPTION

Cette fonction permet de modifier le sample du WS.

Si le sample est plus court que le sample courant, les données situées au delà sont détruites.

Si au contraire le sample est plus long que le sample courant, la valeur NA (--) est donnée aux périodes ajoutées.

FONCTION INTERACTIVE

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#WsSample
```

FONCTION NON INTERACTIVE

Syntaxe

```
$WsSample from to
(from, to := IODE periodes)
```

Exemple

```
$WsSample 1990Y1 2000Y1
```

4.8.2.10 WSEXTRAPOLATE

DESCRIPTION

Cette fonction permet de compléter des séries selon une méthode au choix basée sur les périodes précédentes.

Les méthodes possibles sont les suivantes :

- 0 = Y := Y[-1], if Y null or NA
- 1 = Y := Y[-1], always
- 2 = Y := extrapolation, if Y null or NA
- 3 = Y := extrapolation, always
- 4 = Y := unchanged, always
- 5 = Y := Y[-1], if Y is NA
- 6 = Y := extrapolation, if Y is NA

Il faut fournir la période (inclue dans celle définie dans le WS courant) sur laquelle le calcul doit être effectué.

De plus, la liste des variables à adapter peut également être précisée. Si cette liste est laissée vide, toutes les séries du WS sont modifiées.

FONCTION INTERACTIVE

Syntaxe

```
#WsExtrapolate
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

Syntaxe

```
$WsExtrapolate [method] from to [liste de variables]
où method : 0 ... 6
   from, to := périodes de IODE (yyyyPpp)
```

Exemple

```
$WsExtrapolate 1993Y1 2000Y1 A B C
ou
$WsExtrapolate 1993Y1 2000Y1
```

4.8.2.11 WSLTOHSTOCK

DESCRIPTION

Construit des séries de périodicité supérieure pour des données de type stock (Chômage, Dette, ...).

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément la périodicité de ces séries. La nouvelle périodicité est celle actuellement définie dans le WS actif.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

Cette procédure existe pour les cas suivants :

- annuel vers mensuel
- annuel vers trimestriel
- trimestriel vers mensuel

Deux méthodes sont disponibles, l'une pour les stock, l'autre pour les flux (WsLtoHFlow).

Dans le cas des stock, La méthode d'interpolation est au choix :

- linéaire : A[1980Q{1,2,3,4}] = A[1979Y1] + i * (A[1980Y1] - A[1979Y1])/4 i = 1,2,3,4
- cubic splines = interpolation cubique
- step : A[1980Q{1,2,3,4}] = A[1980Y1]

FONCTION INTERACTIVE

Syntaxe

```
#WsLtoH
```

FONCTION NON INTERACTIVE

Syntaxe

```
$WsLtoHStock {L|C|S} Filename VarList

avec L pour interpolation linéaire
     C pour interpolation par Cubic Splines
     S pour interpolation par Steps
```

Exemple

```
$WsLtoHStock C Fichier1 A B C
```

4.8.2.12 WSLTOHFLOW

DESCRIPTION

Construit des séries de périodicité supérieure pour des données de type flux (PNB, Déficit, ...).

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément la périodicité de ces séries. La nouvelle périodicité est celle actuellement définie dans le WS actif.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

Cette procédure existe pour les cas suivants :

- annuel vers mensuel
- annuel vers trimestriel
- trimestriel vers mensuel

Deux méthodes sont disponibles, l'une pour les stock, l'autre pour les flux.

Dans le cas des flux, la série est répartie sur les sous-périodes :

- interpolation linéaire : A[1980Q1] = A[1980Y1] / n (ou n = nbre de sous-périodes)
- interpolation par splines : interpolation cubique
- interpolation par steps : linéaire

FONCTION INTERACTIVE

Syntaxe

```
#WsLtoH
```

FONCTION NON INTERACTIVE

Syntaxe

```
$WsLtoHFlow {L|C|S}  Filename VarList

avec L pour interpolation linéaire
     C pour interpolation par Cubic Splines
     S pour interpolation par Steps
```

Exemple

```
$WsLtoHFlow S Fichier1 A B C
```

4.8.2.13 WSHTOLSUM

DESCRIPTION

Construit des séries de périodicité inférieure en additionnant les sous-périodes.

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément la périodicité de ces séries. La nouvelle périodicité est celle actuellement définie dans le WS actif.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

Cette procédure existe pour les cas suivants :

- mensuel vers annuel (observation annuelle = somme des 12 mois)
- trimestriel vers annuel (observation annuelle = somme des 4 trimestres)
- mensuel vers trimestriel (observation trimestrielle = somme des 3 mois)

Trois méthodes sont disponibles :

- Addition de sous-périodes (flux) : WsHtoLSum
- Moyenne de sous-périodes (stock) : WsHtoLMean
- Dernière observation (stock) : WsHtoLLast

En cas de valeur inexistante (NA) pour l'une des sous-périodes, le résultat est NA.

FONCTION INTERACTIVE

Syntaxe

```
#WsHtoL
```

FONCTION NON INTERACTIVE

Syntaxe

```
$WsHtoLSum Filename VarList
```

Exemple

```
$WsHtoLSum Fichier1 A B C
```

4.8.2.14 WSHTOLMEAN

DESCRIPTION

Construit des séries de périodicité inférieure en prenant la moyenne des sous-périodes.

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément la périodicité de ces séries. La nouvelle périodicité est celle actuellement définie dans le WS actif.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

Cette procédure existe pour les cas suivants :

- mensuel vers annuel (observation annuelle = moyenne des 12 mois)
- trimestriel vers annuel (observation annuelle = moyenne des 4 trimestres)
- mensuel vers trimestriel (observation trimestrielle = moyenne des 3 mois)

Trois méthodes sont disponibles :

- Addition de sous-périodes (flux) : WsHtoLSum
- Moyenne de sous-périodes (stock) : WsHtoLMean
- Dernière observation (stock) : WsHtoLLast

En cas de valeur inexistante (NA) pour l'une des sous-périodes, le résultat est NA.

FONCTION INTERACTIVE

Syntaxe

```
#WsHtoL
```

FONCTION NON INTERACTIVE

Syntaxe

```
$WsHtoLMean Filename VarList
```

Exemple

```
$WsHtoLMean Fichier1 A B C
```

4.8.2.15 WSHTOLLAST

DESCRIPTION

Construit des séries de périodicité inférieure en prenant la dernière observation des sous-périodes.

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément la périodicité de ces séries. La nouvelle périodicité est celle actuellement définie dans le WS actif.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

Cette procédure existe pour les cas suivants :

- mensuel vers annuel (observation annuelle = celle de décembre)
- trimestriel vers annuel (observation annuelle = celle du dernier trimestre)
- mensuel vers trimestriel (observation trimestrielle = celle du dernier mois du trimestre)

Trois méthodes sont disponibles :

- Addition de sous-périodes (flux) : WsHtoLSum
- Moyenne de sous-périodes (stock) : WsHtoLMean
- Dernière observation (stock) : WsHtoLLast

En cas de valeur inexistante (NA) pour l'une des sous-périodes, le résultat est NA.

FONCTION INTERACTIVE

Syntaxe

```
#WsHtoL
```

FONCTION NON INTERACTIVE

Syntaxe

```
$WsHtoLLast Filename VarList
```

Exemple

```
$WsHtoLLast Fichier1 A B C
```

4.8.2.16 WSSEASONADJ

DESCRIPTION

Construit des séries dessaisonnalisées à l'aide de la méthode Census XI, ainsi que les composantes de trend cyclique et stochastiques.

Nomenclature :

- la série contenant la composante de trend cyclique se nomme :

```
_Cname où name est le nom original
```

- la série contenant la composante stochastique se nomme :

```
_Iname où name est le nom original
```

- la série dessaisonnalisée garde son nom.

Notons que la série dessaisonnalisée est le produit des deux autres.

Pour ce faire, la fonction charge en WS la liste de séries du fichier spécifié et modifie simultanément les séries sélectionnées si nécessaire.

Les nouvelles séries sont ajoutées ou remplacent (pour des noms existants) celles du WS actif.

En cas de valeur inexistante (NA) pour l'une des périodes, le résultat est NA pout toute l'année.

FONCTION INTERACTIVE

Syntaxe

```
#WsSeasonAdj
```

FONCTION NON INTERACTIVE

Syntaxe

```
$WsSeasonAdj Filename VarList
```

Exemple

```
$WsSeasonAdj Fichier1 A B C
```

VOIR ÉGALEMENT

$WsSeasAdj

4.8.2.17 WSSEASADJ

DESCRIPTION

Cette fonction est identique à $WsSeasonAdj à ceci près qu'elle permet de fixer le paramètre qui permet de libérer le critère vérifiant si une influence saisonnière est présente dans une série.

FONCTION INTERACTIVE

Syntaxe

```
#WsSeasonAdj
```

Le paramètre peut être introduit dans le panneau de lancement.

FONCTION NON INTERACTIVE

```
$WsSeasAdj Filename Varname Varname ... Eps
```

VOIR ÉGALEMENT

$WsSeasonAdj

4.8.2.18 WsTrend

Description

Implémentation de la méthode Hodrick-Prescott pour la construction de série de trend. Le principe est le même que pour la dessaisonnalisation: les séries lues dans un fichier sont importées et transformées simultanément.

Fonction interactive

Syntaxe

```
#WsTrend
```

Fonction non interactive

Syntaxe

```
$WsTrend VarFilename Lambda series1 series2 ...
```

4.8.2.19 WsImportXxx

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

Description

Cette opération permet d'effectuer un WsLoad à partir d'un fichier ASCII. Rappelons que l'opération de LOAD d'un workspace remplace le contenu du WS courant par le contenu d'un fichier disque : tous les objets actuellement définis sont détruits et remplacés par ceux définis dans le fichier à charger.

Fonction interactive

Cette fonction ouvre l'écran WsLoad dans lequel on peut entrer un nom de fichier par type d'objet. Si le nom d'un des fichiers a une autre extension que celle des WS de IODE (.av au lieu de .var par exemple), l'importation remplace le LOAD.

Syntaxe

```
#WsImportCmt
#WsImportEqs
#WsImportIdt
#WsImportLst
#WsImportScl
#WsImportTbl
#WsImportVar
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

On passe en paramètre à la commande le nom du fichier à charger.

Syntaxe

```
$WsImportCmt fichier
$WsImportEqs fichier
$WsImportIdt fichier
$WsImportLst fichier
$WsImportScl fichier
$WsImportTbl fichier
$WsImportVar fichier
```

Exemple

```
$WsImportCmt myfile.ac
```

Si l'on charge un nouveau fichier de variables, la période d'échantillonnage du WS est remplacée par celle du fichier chargé.

4.8.2.20 WSEXPORTXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Cette fonction réalise l'opération inverse de la fonction WsImportXxx : elle copie sur disque le WS dans son état actuel en format ASCII.

FONCTION INTERACTIVE

Elle est identique à celle de la fonction #WsSave. Il suffit de placer comme extension aux fichiers une extension différente que celle des WS de IODE.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#WsExportCmt
#WsExportEqs
#WsExportIdt
#WsExportLst
#WsExportScl
#WsExportTbl
#WsExportVar
```

FONCTION NON INTERACTIVE

On passe en paramètre à la commande le nom du fichier à sauver.

Syntaxe

```
$WsExportXxx fichier
{Xxx = cmt | eqs | idt | lst | scl | tbl | var}
```

Exemple

```
$WsExportCmt mytest.ac
```

4.8.2.21 WSIMPORTEVIEWS

DESCRIPTION

Extraction of equations, scalars and identities from E-Views export data.

Format of E-Views data

```
Forecasting Equation:
=========================
D(B_PQVC) = C(1)*D(PQVC) + C(2)*(D2001+D2002) + C(3)*D(B_PQVC(-3))

Substituted Coefficients:
=========================
D(B_PQVC) = 0.894661573413*D(PQVC) - 0.0284533462569*(D2001+D2002) +
0.241546373731*D(B_PQVC(-3))

@IDENTITY gro_w_produz_def  = w_produz_def  / w_produz_def(-1)  - 1

Identities:
```

```
==========
KEEP COEFS:
B_QHOA  = b_1(2)  * b_c0122222
B_QHOB  = b_qh_b_14(1)  * B_CO14

DROP COEFS:
B_QHOCADD  = (b_qh_c_1(1)  * B_CO1)  + (b_qh_c_2(1)  * B_CO2)  + (b_qh_c_7(1)  *
B_CO7)  + (b_qh_c_13(1)  * B_CO13)  + (b_qh_c_14(1)  * B_CO14)
B_QHOCR_NC  = b_qh_cr_14(1)  * B_CO14
```

The E-views file is interpreted as decribed below :

- the 2 equations following the titles "Forecasting Equation" and "Substituted Coefficients" are extracted.
- the first equation is translated into IODE format :
  - `D(...)` is replaced by `d(...)`
  - `C(<n>)` is replaced by the `endoname_<n>`
  - `Expr(<lag>)` is replaced by `Expr[<lag>]`
  - the first encountered variable is chosen as the endogenous variable
  - the first = sign is considered as the separator between left and right members of the equation and therefore replaced by :=
- the members NAME(n) are remplaced by name_n or name. If the last directive is **KEEP COEFS:**, the `_n` is kept. If the last directive is **DROP COEFS:**, `_n` is dropped.
- the coefficients values are extracted from the second equation ("Substituted Coefficients").
- the lines **Estimation Command:** and **Estimation Equation:** are ignored
- the lines beginning with **@INNOV** are skipped.
- the lines beginning by **@IDENTITY** are extracted and translated in IODE equations with no coefficient.
- every equation is added in the current Equations WS.
- every detected coefficient is saved in the Scalars WS.

Lines following the line `IDENTITIES:`

Les lignes contenant des identités sont interprétées comme équations à partir du moment où :

- soit une ligne contenant le texte **Identities:** est trouvée
- soit une première identité préfixée par **@IDENTITY** est trouvée

A partir de ce moment toutes les lignes contenant du texte sont interprétées par le programme comme des équations IODE.

En cas d'erreur de syntaxe, la lecture s'arrête et un message d'erreur est produit.

Directives KEEP COEFS: et DROP COEFS:

Ces directives doivent se trouver seules sur une ligne. Elles peuvent être en majuscules, minuscules ou un mélange des deux.

Elles déterminent la façon dont les termes des équations ou identités EVIEWS du type **name(1)** ou **NAME(n)** doivent être traduites en IODE.

```
     KEEP COEFS:
     DROP COEFS:
```

- Après KEEP COEFS:, la traduction de **name(1)** ou **NAME(1)** donne **name_1**.
- Après DROP COEFS:, la traduction de **name(1)** ou **NAME(1)** donne **name**.

Au début de la lecture fichier, **DROP COEFS:** est la valeur initiale.

Exemple de fichier EVIEWS

```
   Identities:
   ===========
   KEEP COEFS
   B_QHOA  = b_1(2)  * b_c0122222
   B_QHOB  = b_qh_b_14(1)  * B_CO14

   DROP COEFS
   B_QHOCADD  = (b_qh_c_1(1)  * B_CO1)  + (b_qh_c_2(1)  * B_CO2)  + (b_qh_c_7(1)  * B_CO7)
   B_QHOCR_NC  = b_qh_cr_14(1)  * B_CO14
```

Résultat en IODE

```
   B_QHOA := b_1_2*B_C0122222
   B_QHOB := b_qh_b_14_1*B_CO14
   B_QHOCADD := (b_qh_c_1*B_CO1)+(b_qh_c_2*B_CO2)+(b_qh_c_7*B_CO7)
   B_QHOCR_NC := b_qh_cr_14*B_CO14
```

Fonction de rapport

Pour exploiter ce format, il faut appeler la fonction de rapport suivante :

```
       $WsImportEviews filename
```

où filename est le nom du fichier à importer. Les WS courants sont augmentés des équations et scalaires détectés.

4.8.3 Opérations sur les données

Les commandes opérant sur des données des workspaces actifs (en mémoire) permettent de co-

pier, éditer, effacer, renommer, imprimer, etc les données d'un workspace:

- datacreate : crée un objet (cmt | eqs | idt | lst | scl | tbl | var)
- datadelete : détruit un objet (cmt | eqs | idt | lst | scl | tbl | var)
- dataexist : teste un objet (cmt | eqs | idt | lst | scl | tbl | var)
- dataedit : édite un objet (cmt | eqs | idt | lst | scl | tbl | var)
- dataupdate : modifie un objet (cmt | eqs | idt | lst | scl | tbl | var)
- dataappend : ajoute un objet (cmt | lst)
- dataduplicate : dédouble un objet (cmt | idt | lst | scl | tbl | var)
- datarename : renomme un objet (cmt | eqs | idt | lst | scl | tbl | var)
- datasearch : recherche un objet cmt | eqs | idt | lst | scl | tbl | var
- datascan : parcoure les objets eqs | idt | tbl
- datalistXxx : crée une liste d'objets dont les noms répondent à un critère donné cmt | eqs | idt | lst | scl | tbl | var
- datalistsort : trie une liste par ordre alphabétique
- datacompareEps : fixe le seuil d'égalité pour la comparaison des variables
- datacompareXxx : compare le WS et un fichier et crée des listes
- datacalclst : effectue des opérations logiques sur des listes
- datacalcvar : calcule une variable sur base d'une forme LEC
- datadisplaygraph : affiche un graphique sur base de séries (sans tableau)
- datasavegraph : sauve un graphique calculé sur base de séries
- datawidthvar : fixe la largeur des colonnes d'édition des séries
- datandecvar : fixe le nombre de décimales pour l'édition des séries
- datamodevar : fixe le mode pour l'édition des séries
- datastartvar : fixe la première période pour l'édition des séries
- datawidthtbl : fixe la largeur des colonnes d'édition des tableaux
- datawidthscl : fixe la largeur des colonnes d'édition des scalaires
- datandecscl : fixe le nombre de décimales pour l'édition des scalaires
- dataeditcnf : change les options d'édition des variables
- datarasvar : méthode RAS de complétion d'une matrice de séries
- datapatternXXX : création de listes de noms à partir de pattern

4.8.3.1 DataCreateXxx

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

Description

La fonction permet la création d'objets dans le WS courant. Il n'y a pas de fonction plein écran équivalente, sinon les fonctions standard d'édition d'objets (menu DATA).

FONCTION NON INTERACTIVE

Si l'objet "nom" existe déjà, la fonction retourne et signale l'erreur. Sinon l'objet est créé avec une valeur par défaut :

- commentaires, listes, identitées et tableaux : vide
- équation : "NOM : =NOM"
- scalaires : 0.9
- variables : une série avec des NA (--) sur toute la période du WS
- tableaux : crée un tableau vide

Syntaxe

```
$DataCreateVar nom
```

4.8.3.2 DATADELETEXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

La fonction permet de détruire des objets dans le WS courant. Il n'y a pas de fonction plein écran équivalente, sinon la fonction d'édition habituelle (menu Data).

FONCTION NON INTERACTIVE

Si l'objet "nom" n'existe pas, la fonction retourne et signale l'erreur.

Syntaxe

```
$DataDeleteVar nom
```

Depuis la version 5.13, les wildcards sont acceptées dans le nom

4.8.3.3 DATAEXISTXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

La fonction vérifie l'existence d'un objet dans le WS courant.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Si l'objet "nom" n'existe pas, la fonction retourne et signale l'erreur.

Syntaxe

```
$DataDeleteVar nom
```

4.8.3.4 DATAEDITXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

La fonction permet l'édition d'objets du WS courant.

Un écran présente les objets sélectionnés sous forme d'un tableau déroulant (ou tableau d'édition) dans lequel on pourra se déplacer.

Le tableau d'édition est composé des éléments suivants :

- le nom du WS dans la ligne supérieure
- le nom des objets dans la colonne de gauche
- la définition (ou une partie de la définition) des objets en regard du nom dans la partie de droite
- un scrollbar indiquant la position courante dans le tableau et la proportion visible du tableau

L'objet courant est indiqué par le fait que la ligne qui lui correspond est en vidéo inverse.

Les touches de fonction des tableaux déroulant permettent l'affichage de la valeur, la modification et la création des objets.

FONCTION INTERACTIVE

La liste des objets à éditer est passée en argument de la commande. Les éléments de la liste sont séparés par des blancs, virgules ou points-virgules. Si la liste est vide (pas d'argument), tous les objets apparaissent dans le tableau d'édition.

Syntaxe

```
#DataEditXxx [objet [liste d'objets]]
{Xxx = cmt | eqs | idt | lst | scl | tbl | var}
```

FONCTION NON INTERACTIVE

La fonction n'est exploitable qu'en mode plein écran.

4.8.3.5 DATAUPDATEXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

La fonction change le contenu d'un objet dans le WS courant.

FONCTION INTERACTIVE

Il n'y a pas de fonction plein écran équivalente, si ce n'est la fonction habituelle d'édition (Menu DATA).

FONCTION NON INTERACTIVE

Si l'objet "nom" n'existe pas, un objet contenant la nouvelle définition est créé. Si l'objet existe, son contenu est remplacé. La fonction prend plusieurs arguments. Les arguments diffèrent selon le type d'objet à mettre à jour. Le premier argument est toujours le nom de l'objet à changer.

Commentaires

```
$DataUpdateCmt nom commentaire
```

Equations

```
$DataUpdateEqs nom équation_lec
```

Identités

```
$DataUpdateIdt nom identité_lec
```

Listes

```
$DataUpdateLst nom liste
```

Scalaires

```
$DataUpdateScl nom valeur [relax]
```

Tableaux

```
$DataUpdateTbl table_name title;lec1;lec2;...
```

Si title est un nom de commentaire, le commentaire est utilisé comme titre. De même, si on trouve des noms de variables comme forme lec, et qu'il existe un commentaire pour ces variables, le titre de la ligne correspondante est remplacé par la valeur du commentaire.

Supposons qu'il existe en WS un commentaire pour A et non pour B, et un commentaire TIT :

```
Commentaire A   : "Produit national brut"
Commentaire TIT : "Titre de mon tabelo"
```

La ligne

```
$DataUpdateTbl T1 TIT;A;B;A+B
```

crée le tableau T1 avec la forme suivante :

```
+-----------------------------------------+
|          Titre de mon tabelo            |
|                                         |
|  Produit national brut              A   |
|  B                                  B   |
|  A+B                                A+B |
+-----------------------------------------+
```

Variables

```
$DataUpdateVar nom [L,l | D,d | G,g] période valeur1 valeur2 ...
 où L,l := en valeur (défaut)
    D, d := en différence
    G, g := en taux de croissance
    période := la période du début de la mise à jour
```

4.8.3.6 DataAppendXxx

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

La fonction ajoute un texte à la définition d'un objet du WS courant.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Si l'objet "nom" n'existe pas, un objet contenant la nouvelle définition est créé. Si l'objet existe, son contenu est complété.

- Commentaires

```
$DataAppendCmt nom commentaire
```

- Listes

```
$DataAppendLst nom list
```

4.8.3.7 DATADUPLICATEXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Les objets définis dans les WS courants peuvent être copiés dans des objets du même type mais d'un nom différent. Remarque : il existe une méthode simple pour effectuer cette opération : dans l'écran de définition de l'objet à dupliquer, il suffit de changer le nom de l'objet et de le sauver. Un nouvel objet ayant les mêmes caractéristiques que l'ancien est ainsi créé.

La fonction décrite ici permet la même opération. Elle fonctionne de la même manière pour tous les objets sauf pour les équations.

FONCTION INTERACTIVE

La commande affiche un écran de saisie contenant trois champs.

Le type d'objet à éditer peut être sélectionné en se plaçant sur le champ indiquant le type et en pressant TAB : un menu reprenant tous les types d'objets permet d'effectuer la sélection.

Les deux autres champs contiennent respectivement l'ancien et le nouveau nom de l'objet.

Syntaxe

```
#DataDuplicateCmt
```

FONCTION NON INTERACTIVE

La fonction prend successivement deux paramètres: l'ancien et le nouveau nom de l'objet.

Syntaxe

```
$DataDuplicateCmt ancien_nom nouveau_nom
```

Exemple

```
$DataDuplicateCmt A01 A02
```

La fonction DataDuplicateEqs n'est pas implémentée et n'a d'ailleurs pas de signification. En effet, une équation porte le nom de sa variable endogène. Changer son nom change donc le nom de l'endogène, ce qui en général n'a pas de sens.

L'utilisateur curieux qui utiliserait malgré tout cette fonction recevrait un message d'avertissement.

4.8.3.8 DATARENAMEXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

La fonction change le nom d'un objet dans le WS courant.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Si l'objet "nom" n'existe pas, la fonction retourne et signale l'erreur. Si l'objet existe, il sera sauvé sous un nouveau nom. S'il y a déjà un objet portant ce nouveau nom, il est remplacé.

Syntaxe

```
$DataRenameVar nom nouveau_nom
```

4.8.3.9 DATASEARCHXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Cette fonction permet de rechercher dans un des WS courants la liste des objets contenant un string donné.

Le résultat de cette recherche est une liste d'objets qui peut au besoin être sauvée dans le WS courant de listes. Par défaut la liste _RES contient le résultat de la dernière recherche.

FONCTION INTERACTIVE

La fonction affiche une page de saisie comprenant les champs suivants :

- Texte à rechercher : le texte à rechercher peut contenir des caractères spéciaux qui permettent de spécifier les limites de la recherche :
  - * : n'importe quelle suite de caractères (même vide)
  - ? : un et un seul caractère (quelconque)
  - @ : n'importe quel caractère alphanumérique
  - ampersand : n'importe quel caractère non alphanumérique
  - | : n'importe quel caractère alphanumérique ou aucun en début et fin de string
  - ! : n'importe quel caractère non alphanumérique ou aucun en début et fin de string
  - \ placé devant un des caractères spéciaux supprime son interprétation
- Mot entier/partiel (Whole word) : préciser Yes si la chaîne à rechercher doit être un mot entier et non une partie de mot. Indiquer No si cela n'a pas d'importance.
- Majuscule/minuscule (Exact case) : préciser si la recherche doit différencier majuscules et minuscules dans la chaîne à rechercher.
- Type d'objet: le type d'objet doit être indiqué dans ce champ. En pressant TAB, la liste des types apparaît. Il suffit de se placer sur le type choisi et de presser ENTER.

Les recherches sont différentes en fonction du type d'objet :

- Commentaires : le nom et le texte du commentaire sont analysés
- Equations : le nom et la forme LEC de l'équation sont analysés
- Identités : le nom et la forme LEC de l'identité sont analysés
- Listes : le nom et le texte de la liste sont analysés
- Scalaires : le nom du scalaire est analysé
- Tableaux : le nom, les titres et les formes LEC du tableau sont analysés
- Variables : le nom de la variable est analysé
- Liste résultat: il est possible de sauver la liste des objets répondant au critère dans le WS de listes courant. Le nom de cette liste doit être indiqué et être un nom de liste valide. Si une liste de même nom existe, elle sera remplacée par le résultat du calcul.
- Résultat: dans le bas de l'écran, un champ reprend les premiers objets satisfaisant au critère.

La recherche commence lorsque la page est validée avec F10.

Syntaxe

```
#DataSearchCmt
```

FONCTION NON INTERACTIVE

Sept paramètres doivent être passés à la commande.

Syntaxe

```
$DataSearchXxx mask word ecase in_name in_formula in_text list_result

(word, ecase, in_name, in_formula, in_text := 0 ou 1)
(mask := caractères et éventuellement ?, *, ...)
(list_result := le nom de la liste résultat)
```

Exemple

```
$datasearchCmt TE?T 0 0 1 0 1 NEW2
#DataEditCMT $NEW2
```

4.8.3.10 DATASCANXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Cette fonction permet de rechercher dans un des WS courants la liste des variables et scalaires utilisés dans la définition des objets de ce WS.

Le résultat de cette recherche est sauvé dans deux listes, contenant d'une part les noms des scalaires (liste _SCAL) et d'autre part ceux des variables (liste _EXO). Si on ne passe pas d'argument à cette fonction, tout le WS est examiné.

FONCTION INTERACTIVE

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

```
$DataScanEqs [nom1,nom2,...]
$DataScanIdt [nom1,nom2,...]
$DataScanTbl [nom1,nom2,...]
```

4.8.3.11 DATALISTXXX

DESCRIPTION

Cette fonction construit une liste de noms d'objets sur base d'un critère (pattern) de sélection sur les noms. Elle peut fonctionner sur les objets en WS ou sur les objets d'un fichier.

Elle permet par exemple d'obtenir la liste des objets dont le noms commence par A et se termine par BEL (A*BEL).

FONCTION INTERACTIVE

Les 4 champs à remplir permettent de préciser le nom de la liste à contruire, le critère, le type d'objet et le nom du fichier. Si ce dernier est laissé vide, les objets du WS sont utilisés.

Syntaxe

```
#DataListXxx
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

La commande prend deux ou trois paramètres, le premier précise le nom de la liste à créer, le deuxième le pattern et le dernier (optionnel) le nom du fichier.

Syntaxe

```
$DataListXxx listname pattern [filename]
```

Exemple

```
$DataListVar _BEL_ *BEL*
```

Fournit dans la liste _BEL_ toutes les variables sont le nom contient le texte BEL: AX1BEL, BEL1, MIRABELLE, etc.

4.8.3.12 DATALISTSORT

DESCRIPTION

Cette fonction permet un tri alphanumérique sur le contenu d'une liste.

FONCTION INTERACTIVE

Les deux champs à remplir permettent de préciser le nom de la liste à trier et le nom de la liste résultat.

Syntaxe

```
#DataListSort
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

La commande prend deux paramètres, le premier précise le nom de la liste à trier et le deuxième le nom de la liste résultat.

Syntaxe

```
$DataListSort liste liste_triée
```

4.8.3.13 DATACOMPAREEPS

DESCRIPTION

Cette fonction permet de fixer le seuil en-deçà duquel le test d'égalité des variables est considéré comme ayant été satisfait.

Le test de comparaison est :

```
si x1 <> 0 :   |(x1 - x2) / x1| < eps
sinon :        |x2| < eps
```

SYNTAXE

```
$DataCompareEps eps
```

L'écran de comparaison de WS (Data/List/File Compare) permet également de spécifier cette valeur (paramètre Threshold).

SEUIL DE COMPARAISON PAR DÉFAUT

Le seuil de comparaison est fixé à 1e-7 par défaut.

4.8.3.14 DATACOMPAREXXX

DESCRIPTION

Le contenu du WS courant peut être comparé à celui d'un fichier. Le résultat de cette comparaison est composé de 4 listes :

- objets trouvés dans le WS seulement
- objets trouvés dans le fichier seulement
- objets trouvés dans les deux, avec la même définition
- objets trouvés dans les deux, avec une définition différente

La comparaison s'effectue en fonction du type des objets.

FONCTION INTERACTIVE

Les champs à remplir sont de deux types: les deux premiers spécifient le type de WS à comparer (VAR, etc) et le nom du fichier à comparer au WS.

Les quatre suivants permettent d'indiquer les noms à donner aux listes résultat.

Syntaxe

```
#DataCompare
```

FONCTION NON INTERACTIVE

La fonction prend 5 paramètres (le type d'objet fait partie du nom de la fonction): le fichier à comparer les 4 listes à construire.

Syntaxe

```
$DataCompareXxx filename ONE TWO THREE FOR

      ONE         in WS only
      TWO         in file only
      THREE       in both, equal
      FOR         in both, different
```

Exemple

```
$DataCompareVar myws WS FILE EQ DIFF
```

4.8.3.15 DATACALCLST

DESCRIPTION

Cette fonction calcule une liste sur base de deux listes en en prenant la réunion, l'intersection ou la différence.

FONCTION INTERACTIVE

Les 4 champs à remplir permettent de préciser

- le nom de la liste résultat
- le noms des opérandes (listes)
- l'opérateur

Syntaxe

```
#DataCalcLst
```

FONCTION NON INTERACTIVE

La commande prend 4 paramètres, le premier précise le nom de la liste résultat, le deuxième celui de la première liste, le troisième l'opération et le quatrième le nom de la deuxième liste.

Syntaxe

```
$DataCalcLst res lst1 op lst2
   où op = + : union
           * : intersection
           - : différence
```

Exemple

```
$DataCalcLst _RES LST1 * LST2
```

Fournit dans la liste _RES les noms présents dans LST1 et LST2.

4.8.3.16 DATACALCVAR

DESCRIPTION

Cette fonction permet de calculer une nouvelle série à partir d'une formule LEC. Cette formule est exécutée sur toute la période du WS courant et ce en utilisant les valeurs des variables du WS courant.

FONCTION INTERACTIVE

La fonction plein écran équivalente est la fonction standard d'édition des variables.

FONCTION NON INTERACTIVE

On passe deux paramètres à cette fonction :

- le nom de la variable à créer
- la formule LEC à calculer

Syntaxe

```
$DataCalcVar nom formule
```

Exemple

```
$DataCalcVar X    beta + gamma * ln B + alpha * ln C
```

4.8.3.17 DATADISPLAYGRAPH

DESCRIPTION

Cette fonction permet de visualiser des séries sous forme de graphique.

FONCTION INTERACTIVE

Les champs à remplir permettent de préciser les noms des variables, le mode de visualisation, la

période, le type les axes, les grids, ...

- Variables name : liste des variables à inclure dans le graphique. Les noms (ou liste) peuvent être séparés par un blanc, un + ou une combinaison des deux. Les variables séparées par un blanc seront visualisées dans des graphiques séparés sur la même page, tandis que les variables "additionnées" seront groupées dans un même graphique. Ainsi, si ce champ contient "X Y+Z T", trois graphiques seront construits avec respectivement les variables X (premier graphique), Y et Z (groupées dans le second graphique) et T (dernier graphique).
- Mode : il s'agit d'un champ MENU permettant de préciser si les variables doivent être visualisées en valeur, en différence ou en taux de croissance :

```
L=level
D=differences
G=growthrates
d=YoY differences : différences sur 1 année
g=YoY growth rates : taux de croissance sur 1 année
```

- Chart type : il s'agit d'un champ MENU permettant de préciser le type de graphique à produire: line, scatter, scatter line ou bar
- Grids : il s'agit également d'un champ MENU permettant de définir le type de quadrillage en X et en Y (pas de quadrillage, quadrillage majeur ou quadrillage mineur)
- Y scaling: ces deux champs permettent de définir les valeurs minimum et maximum de l'axe des Y. Si ces valeurs sont nulles, l'axe est calculé automatiquement en fonction des valeurs à afficher.
- Axis : ce champ du type MENU permet de sélectionner le type de transformation de coordonnées à utiliser: valeur, logarithme, ...
- Sample : préciser la période d'impression (de période_1 à période_2). La syntaxe d'une période fait l'objet d'un chapitre séparé.
- Save graph in file: le graphique peut être sauvegardé dans un fichier au format AGL éditable par le programme GODE.

Syntaxe

```
#DataDisplayGraph
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

On passe les paramètres suivants à cette fonction :

Syntaxe

```
$DataDisplayGraph {Level | Diff | Grt | diffYoY | grtYoY}
                  {Line | Scatter | Bar | Mixt}
                  {NoXGrids | MinorXGrids | J(MajorXGrids)}
                  {NoYGrids | MinorYGrids | J(MajorYGrids)}
                  {Level | G(Log) | Semi-Log | Percent}
                  {ymin | --} {ymax | --}
                  perfrom perto varname1 varname2 ...
```

La signification des paramètres est expliquée plus haut. Il est à noter que tous les paramètres sont requis et doivent être passés dans l'ordre indiqué.

4.8.3.18 DATASAVEGRAPH

DESCRIPTION

Cette fonction permet de construire des graphiques à partir de séries et d'en sauvegarger le résultat dans un fichier AGL.

FONCTION INTERACTIVE

Les champs à remplir permettent de préciser les noms des variables, le mode de visualisation, la période, le type les axes, les grids, ...

- Variables name : liste des variables à inclure dans le graphique. Les noms (ou liste) peuvent être séparés par un blanc, un + ou une combinaison des deux. Les variables séparées par un blanc seront visualisées dans des graphiques séparés sur la même page, tandis que les variables "additionnées" seront groupées dans un même graphique. Ainsi, si ce champ contient "X Y+Z T", trois graphiques seront construits avec respectivement les variables X (premier graphique), Y et Z (groupées dans le second graphique) et T (dernier graphique).
- Mode : il s'agit d'un champ MENU permettant de préciser si les variables doivent être visualisées en valeur, en différence ou en taux de croissance
- Chart type: il s'agit d'un champ MENU permettant de préciser le type de graphique à produire: line, scatter, scatter line ou bar
- Grids : il s'agit également de champs MENU permettant de définir le type de quadrillage en X et en Y (pas de quadrillage, quadrillage majeur ou quadrillage mineur)
- Y scaling: ces deux champs permettent de définir les valeurs minimum et maximum de l'axe des Y. Si ces valeurs sont nulles, l'axe est calculé automatiquement en fonction des valeurs à afficher.
- Axis : ce champ du type MENU permet de sélectionner le type de transformation de coordonnées à utiliser: valeur, logarithme, ...
- Sample : préciser la période d'impression (de période_1 à période_2). La syntaxe d'une période est données dans le manuel de référence.
- Save graph in file: le graphique peut être sauvegardé dans un fichier au format AGL éditable par le programme GODE.

Syntaxe

```
#DataDisplayGraph
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

On passe les mêmes paramètres que ceux décrits ci-dessus à cette fonction :

Syntaxe

```
$DataSaveGraph    aglfilename
                  {Level | Diff | Grt}
                  {Line | Scatter | Bar | Mixt}
                  {NoXGrids | MinorXGrids | J(MajorXGrids)}
                  {NoYGrids | MinorYGrids | J(MajorYGrids)}
                  {Level | G(Log) | Semi-Log | Percent}
                  {ymin | --} {ymax | --}
                  perfrom perto varname1 varname2 ...
```

Il est à noter que tous les paramètres sont requis et doivent être passés dans l'ordre indiqué.

4.8.3.19 DATAWIDTHVAR

DESCRIPTION

Cette fonction permet de spécifier la largeur des colonnes d'édition des séries statistiques. La valeur doit être comprise entre 2 et 12.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F3 et s-F3 permettent de changer la largeur des colonnes lors de l'édition.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataWidthVar n
n entre 2 et 12
```

4.8.3.20 DATANDECVAR

DESCRIPTION

Cette fonction permet de spécifier le nombre de décimales lors de l'édition des séries statisti-

ques. La valeur doit être comprise entre -1 (nombre de décimales variable) et 6.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F4 et s-F4 permettent de changer la largeur des colonnes de l'édition.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataNdecVar n
n entre -1 et 6
```

4.8.3.21 DATAMODEVAR

DESCRIPTION

Cette fonction permet de choisir le mode d'édition des séries statistiques: Niveau, différences ou taux de croissance.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. La touche F5 permet de changer la largeur des colonnes lors de l'édition.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataModeVar n
n = 0 pour niveau
n = 1 pour différences
n = 2 pour taux de croissances
```

4.8.3.22 DATASTARTVAR

DESCRIPTION

Cette fonction permet de choisir la première colonne visible lors de l'édition des séries statistiques. La valeur doit être comprise entre 0 (première colonne) et le nombre de périodes disponibles.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataStartVar n
n entre 0 et nombre de périodes
```

4.8.3.23 DATAWIDTHTBL

DESCRIPTION

Cette fonction permet de spécifier la largeur des colonnes d'édition des tableaux. La valeur doit être comprise entre 2 et 60.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F3 et s-F3 permettent de changer la largeur des colonnes lors de l'édition.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataWidthTbl n
n entre 2 et 60
```

4.8.3.24 DATAWIDTHSCL

DESCRIPTION

Cette fonction permet de spécifier la largeur des colonnes d'édition des séries scalaires. La valeur doit être comprise entre 2 et 12.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F3 et s-F3 permettent de changer la largeur des colonnes lors de l'édition.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataWidthScl n
n entre 2 et 12
```

4.8.3.25 DATANDECSCL

DESCRIPTION

Cette fonction permet de spécifier le nombre de décimales lors de l'édition des scalaires. La valeur doit être comprise entre -1 (nombre de décimales variable) et 6.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F4 et s-F4 permettent de changer la largeur des colonnes de l'édition.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataNdecScl n
n entre -1 et 6
```

4.8.3.26 DATAEDITCNF

DESCRIPTION

Cette fonction permet de changer le mode de visualisation du tableau de séries en "Level" (valeur réelle) en mode "Difference" et en mode "Growth Rate" (taux de croissance). Elle permet aussi de choisir le nombre de décimales à afficher.

FONCTION INTERACTIVE

Les deux champs à remplir permettent de préciser le mode de visualisation et le nombre de décimales à afficher.

Syntaxe

```
#DataEditCnf
```

Cette fonction est décrite dans le "Manuel de l'utilisateur".

FONCTION NON INTERACTIVE

On passe deux paramètres à cette fonction :

- le mode de visualisation {Level | Differences | GrowthRates}
- le nombre de décimales à afficher (-1, 0, 1, 2, ...)

Syntaxe

```
$DataEditCnf {L | D | G} n
(n : = -1, 0, 1, 2, 3, ...)
```

4.8.3.27 DATARASVAR

Vertrekkende van de waarden van variabelen beantwoordend aan pattern, worden de waarden verdeeld via een RAS methode.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataRasVar pattern X_dimensie Y_dimensie ref_jaar h_jaar [maxit [eps]]
```

- **pattern:** de variabelen worden gebruikt die voldoen aan de volgende criteria x wordt vervangen door alle waarden uit $X, y door die uit $Y
- **X_dimensie:** lijst van de waarden die "x" uit het pattern kan aannemen OPGELET: laatste uit de lijst is de SOM over de x dimensie
- **Y_dimensie:** lijst van de waarden die "y" uit het pattern kan aannemen OPGELET: laatste uit de lijst is de SOM over de y dimensie
- **ref_jaar:** referentie jaar: het jaar waarvoor alle gegevens bekend zijn
- **h_jaar:** referentie jaar: het jaar waarvoor alleen de sommen bekend zijn
- **maxit:** maximaal iteraties (default=100)
- **eps:** de drempel (default=0.001)

Voorbeeld

```
$WsLoadVar ras.av
$DataUpdateLst X R1;R2;R3;R4;R5;R6;RT
$DataUpdateLst Y C1;C2;C3;C4;C5;CT
$DataRasVar xy $X $Y 1980Y1 1981Y1 10 0.00001
```

de RAS matrix ziet er dan als volgt uit:

```
R1C1 R1C2 R1C3 R1C4 R1C5 R1C6 | R1CT
R2C1 R2C2 R2C3 R2C4 R2C5 R2C6 | R2CT
R3C1 R3C2 R3C3 R3C4 R3C5 R3C6 | R3CT
R4C1 R4C2 R4C3 R4C4 R4C5 R4C6 | R4CT
R5C1 R5C2 R5C3 R5C4 R5C5 R5C6 | R5CT
------------------------------
RTC1 RTC2 RTC3 RTC4 RTC5 RTC6 | RTCT
```

met de waarden bij het jaar **`ref_jaar`**. De nieuwe rij en kolom sommen hebben de waarde uit het **`h_jaar.`** Als er waarden bekend in **`h_jaar`** voor enkele cellen dan worden die gebruikt.

RAS berekent dan de cellen zo dat de nieuwe randvoorwaarden gelden en overschrijft de onkende waarden (Not Available)

4.8.3.28 DATAPATTERNXXX

Deze functie creëert lijsten met daarin de namen van objecten die voldoen aan een opgegeven patroon.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$DataPatternXXX lijst pattern X_dimensie [Y_dimensie]
```

- **lijst:** de naam van de lijst met het resultaat
- **pattern:** : patroon waaraan de naam van de objecten moeten voldoen, als men "x" vervangt door de elementen uit de X_dimensie en indien opgegeven "y" door de elementen uit de Y_dimensie
- **`X_dimensie`** : lijst met "x"-mogelijkheden
- **`[Y_dimensie`** : lijst met "y"-mogelijkheden] : optioneel

Opgelet: alleen bestaande elementen worden in de lijst opgenome.

Voorbeeld

```
$WsLoadVar ras.av
$DataUpdateLst X R1;R2;R3
$DataUpdateLst Y C1;C2
$DataPatternVar RC xy $X $Y

$RC=R1C1,R1C2,R2C1,R2C2,R3C1,R3C2
inzoverre R1C1,R1C2,R2C1,R2C2,R3C1,R3C2 variabelen zijn in WS
```

4.8.4 Opérations spécifiques aux équations

Les commandes suivantes permettent de modifier les paramètres de l'estimation et d'estimer des équations :

- EqsEstimate
- EqsStepWise
- EqsSetCmt
- EqsSetSample
- EqsSetMethod
- EqsSetInstrs
- EqsSetBloc

4.8.4.1 EqsEstimate

Description

Cette fonction réestime une équation ou un bloc d'équations.

Fonction interactive

Cette fonction est détaillée dans la manuel de l'utilisateur. Elle permet également de définir et de modifier une équation.

Syntaxe

```
#EqsEstimate
```

Fonction non interactive

Seule la période peut être adaptée dans cette fonction. Les autres paramètres (méthode, instruments, etc) sont ceux définis actuellement dans l'équation. D'autres fonctions permettent de modifier les instruments, la méthode : il s'agit des fonctions EqsSet....

Syntaxe

```
$EqsEstimate perfrom perto eqname1 ...
```

Résultats explicites

En fin d'estimation, certaines variables et scalaires sont automatiquement crées si le processus a convergé. Ces variables et scalaires peuvent être exploités à des fins de calculs et, comme ils s'agit d'objets faisant partie du WS, peuvent être sauvés pour une utilisation future.

Il s'agit des tests résultant de la dernière estimation sont sauvés dans des scalaires. Il en va de même pour les résidus, membres de gauche et de droite des équations.

Les tests portent les noms suivants (e0_* pour la première équation du bloc, e1_* pour la deuxième, ...) :

- e0_n : nombre de période du sample
- e0_k : nombre de coefficients estimés
- e0_stdev : std dev des résidus
- e0_meany : moyenne de Y
- e0_ssres : somme du carrés des résidus
- e0_stderr : std error
- e0_stderrp : std error %
- e0_fstat : F-Stat
- e0_r2 : R carré
- e0_r2adj : R carré ajusté
- e0_dw : Durbin-Watson
- e0_loglik : Log Likelihood

Les séries calculées sont également sauvées dans une variable sous les noms :

- _YCALC0 pour le membre de droite de la première équation du bloc, _YCALC1 pour la deuxième équation, etc.
- _YOBS0 pour le membre de gauche de la première équation du bloc, _YOBS1 pour la deuxième équation, etc.
- _YRES0 pour les résidus de la première équation du bloc, ...

En dehors du sample d'estimation, les valeurs de la série sont --.

4.8.4.2 EqsStepWise

Description

Cette fonction estime un bloc d'équations et recherche les meilleurs tests possibles pour toutes les combinaisons de coefficients possibles.

Syntaxe

```
$EqsStepWise from to eqname leccond {r2|fstat}
    from to : période d'estimation
    eqname  : équation à estimer
    leccond : condition d'acceptation
```

4.8.4.3 EqsSetCmt

Description

Cette fonction fixe le commentaire d'une équation.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$EqsSetCmt eqname comment

    où eqname est le nom de l'équation
       comment est une texte libre
```

Exemple

```
$DataUpdateEqs E E := c1 + c2 * F
$EqsSetCmt E Ceci est le commentaire de E
$EqsSetSample 1971Y1 1990Y1 E
$EqsSetMethod 1 B E
$EqsSetInstrs E instruments
$EqsSetBloc B E
```

4.8.4.4 EQSSETSAMPLE

DESCRIPTION

Cette fonction fixe la période d'estimation d'une liste d'équations. Elle est utile pour changer de période d'estimation dans le cours de l'exécution d'un rapport.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

La fonction modifie la période pour toutes les équations dont le nom est passé en paramètre.

Syntaxe

```
$EqsSetSample from to eqname1 eqname2 ...

    où from et to sont les périodes limites du sample
       eqname1 .. sont des noms d'équations existantes
```

Exemple

```
$DataUpdateEqs E E := c1 + c2 * F
$EqsSetCmt E Ceci est le commentaire de E
$EqsSetSample 1971Y1 1990Y1 E
$EqsSetMethod 1 B E
$EqsSetInstrs E instruments
$EqsSetBloc B E
```

4.8.4.5 EQSSETMETHOD

DESCRIPTION

Cette fonction fixe la méthode d'estimation d'une liste d'équations. Elle est utile pour changer de méthode d'estimation dans le cours de l'exécution d'un rapport.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

La fonction modifie la méthode pour toutes les équations dont le nom est passé en paramètre.

Syntaxe

```
$EqsSetMethod   {0|1|2|3} eqname1 eqname2 ...

    où 0 indique la méthode LSQ (moindres carrés)
       1 indique la méthode Zellner
       2 indique la méthode INF (2 stages avec instruments)
       3 indique la méthode GLS (3 stages avec instruments)
       4 indique la méthode MAXLIK (Maximum likelihood - BHHH - à partir de la
version 6.21)

       eqname1 .. sont des noms d'équations existantes
```

Exemple

```
$DataUpdateEqs E E := c1 + c2 * F
$EqsSetCmt E Ceci est le commentaire de E
$EqsSetSample 1971Y1 1990Y1 E
$EqsSetMethod 1 B E
$EqsSetInstrs E instruments
$EqsSetBloc B E
```

4.8.4.6 EQSSETINSTRS

DESCRIPTION

Cette fonction fixe les instruments à utiliser pour l'estimation d'une équation.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

La fonction modifie les instruments pour l'équation spécifiée.

Syntaxe

```
$EqsSetInstrs eqname lec_intr_1;lec_instr_2;...

    où eqname est le nom de l'équation
       lec_instr_1, ... sont les formes lec des instruments
```

Exemple

```
$DataUpdateEqs E E := c1 + c2 * F
$EqsSetCmt E Ceci est le commentaire de E
$EqsSetSample 1971Y1 1990Y1 E
$EqsSetMethod 1 B E
$EqsSetInstrs E instruments
$EqsSetBloc B E
```

4.8.4.7 EQSSETBLOC

DESCRIPTION

Cette fonction fixe le bloc d'estimation d'une liste d'équations. Elle est utile pour fixer le bloc d'estimation dans le cours de l'exécution d'un rapport.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

La fonction fixe le bloc de toutes les équations dont le nom est passé en paramètre. Le bloc est la liste d'équations elle-même.

Syntaxe

```
$EqsSetBloc eqname1 eqname2 ...

    où eqname1 .. sont des noms d'équations existantes
```

Exemple

```
$DataUpdateEqs E E := c1 + c2 * F
$EqsSetCmt E Ceci est le commentaire de E
$EqsSetSample 1971Y1 1990Y1 E
$EqsSetMethod 1 B E
$EqsSetInstrs E instruments
$EqsSetBloc B E
```

4.8.5 CONFIGURATION DE L'IMPRIMANTE

Les commandes suivantes sont utilisées pour configurer l'imprimante ou le fichier de réception

des impressions:

- printdest : fixe la destination de l'impression
- printdestnew : printdest et réinitialisation du fichier d'impression
- printnbdec : fixe le nombre de décimales pour l'impression
- printlang : fixe la langue par défaut pour l'impression
- PrintA2mAppend : permet de ne pas vider le fichier a2m avant l'impression
- PrintFont : fixe la police de caractère pour l'impression
- PrintTableFont : fixe la police de caractères pour les tableaux
- PrintTableBox : fixe la largeur du cadre des tableaux
- PrintTableColor : permet d'utiliser ou non la couleur dans les tableaux
- PrintTableWidth : fixe la largeur des tableaux en Frame et Rtf
- PrintTableBreak : permet ou non les coupures des tableaux sur plusieurs pages
- PrintTablePage : force un saut de page avant chaque tableau
- PrintBackground : fixe la couleur du background (tableaux et graphiques)
- PrintGraphBox : fixe la largeur du cadre de graphiques
- PrintGraphBrush : fixe la densité du background des graphiques
- PrintGraphSize : fixe la taille des graphiques
- PrintGraphPage : force un saut de page avant chaque graphique
- PrintRtfHelp : génère un fichier RTF pour un help Windows
- PrintRtfTopic : crée un nouveau topic (Help Windows)
- PrintRtfLevel : change le niveau hiérarchique des topics suivants
- PrintRtfTitle : fixe le titre de l'aide Windows
- PrintRtfCopyright : fixe le texte du copyright de l'aide Windows
- PrintHtmlHelp : génère un fichier HTML pour un help HtmlHelp
- PrintHtmlStrip : Lors de la génération de fichier HTML (A2mToHtml), ne génère pas d'entête ni de footer
- PrintParanum : permet de numéroter les titres
- PrintPageHeader : fixe le titre des pages imprimées
- PrintPageFooter : fixe la footnote des pages imprimées
- SetPrinter : fixe l'imprimante par défaut
- PrintOrientation : fixe l'orientation du papier dans l'imprimante
- PrintDuplex : fixe le mode recto-verso de l'imprimante
- PrintGIFBackColor : définit la couleur de fond des graphiques
- PrintGIFTransColor : définit la couleur considérée comme "transparente"
- PrintGIFTransparent : indique si le fichier GIF doit être au format transparent
- PrintGIFInterlaced : indique si le fichier GIF doit être au format interlacé
- PrintGIFFilled : indique s'il faut remplir les barres dans les bar charts
- PrintGIFFont : indique si le numéro du font à utiliser

4.8.5.1 PRINTDEST

DESCRIPTION

La fonction permet de spécifier s'il faut imprimer dans un fichier et, si c'est le cas, le nom et le type du fichier résultat. Cette définition est valable pour toutes les impressions faites au départ d'un menu ou d'un rapport de IODE, jusqu'à la fin du rapport en exécution ou jusqu'à ce qu'une autre commande du rapport spécifie une autre destination, auquel cas la configuration de l'impression est modifiée pour toutes les impressions ultérieures.

Le fichier destination est toujours réinitialisé, même dans le cas des fichiers a2m. Ceci est nouveau à partir de la version 5 de IODE.

FONCTION INTERACTIVE

Dans chaque écran d'impression les différents paramètres requis peuvent être spécifiés : nombre de décimales et langue par défaut. Un bouton Setup permet en outre de déterminer s'il faut imprimer dans un fichier ou sur une imprimante.

FONCTION NON INTERACTIVE

Le nom du fichier destination est passé en paramètre de la commande :

Syntaxe

```
$PrintDest [nom_fichier] [format]

    où format = A2M, MIF, HTML, RTF ou CSV
    par défaut, A2M est le format choisi
```

Si **nom_fichier** n'est pas spécifié, l'impression aura lieu sur une imprimante.

Exemple

```
$PrintDest test.mif MIF
```

4.8.5.2 PRINTDESTNEW

DESCRIPTION

Cette fonction est identique à $PrintDest, à ceci près qu'elle détruit le fichier output.

FONCTION NON INTERACTIVE

Le nom du fichier destination est passé en paramètre de la commande :

Syntaxe

```
$PrintDestNew [nom_fichier] [format]

    où format = A2M, MIF, HTML, RTF ou CSV
    par défaut, A2M est le format choisi
```

Si **nom_fichier** n'est pas spécifié, l'impression aura lieu sur une imprimante.

4.8.5.3 PRINTNBDEC

DESCRIPTION

La fonction permet de spécifier le nombre de décimales à imprimer lors des impressions de tableaux ou de variables.

FONCTION INTERACTIVE

Dans chaque écran d'impression les différents paramètres requis sont spécifiés.

Un de ces paramètres est "NbDec". Un nombre entier positif spécifie le nombre de décimales qui seront imprimées. Si la valeur -1 est spécifiée, le nombre de décimales est celui de la définition interne de la valeur à imprimer (en général quelconque).

FONCTION NON INTERACTIVE

Le nombre de décimales est passé en paramètre de la commande :

Syntaxe

```
$PrintNbDec  nb
```

Exemple

```
$wsloadvar bist92\bistel
$wsloadtbl bist92\tbistelf

$printdest bist92\bistelf1.a2m
#show processing french tables  file 1/2
$printnbdec 1
$PrintTbl 89:8 HYPEIR
$PrintTbl 89/88:8 HYPEIIR
```

4.8.5.4 PRINTLANG

DESCRIPTION

La fonction permet de spécifier la langue d'impression quand on imprime les tableaux. La langue par défaut est l'anglais.

FONCTION INTERACTIVE

Dans chaque écran d'impression les différents paramètres requis sont spécifiés. Le champ "Language" est un champ MENU dans lequel on précise la langue d'impression.

FONCTION NON INTERACTIVE

La langue d'impression est passé en paramètre de la commande :

Syntaxe

```
$PrintLang {English | French | Dutch}
```

Exemple

```
$wsloadvar bist92\bistel
$wsloadtbl bist92\tbistelf

$printdest bist92\bistelf1.a2m
#show processing dutch tables  file 1/2
$printnbdec 1
$printlang Dutch
$PrintTbl 89:8 HYPEIR
$PrintTbl 89/88:8 HYPEIIR
```

4.8.5.5 PRINTA2MAPPEND

DESCRIPTION

Permet de ne pas vider le fichier a2m avant l'impression. Par défaut, les fichiers résultat des impressions sont vidés lors d'une nouvelle impression. Dans le seul cas des fichiers a2m, il est possible d'éviter ce traitement par défaut et d'accumuler de la sorte plusieurs impressions successives dans un seul et même fichier.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences/A2M", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintA2mAppend [NO|Yes]
```

Exemple

```
$PrintA2mAppend Yes
$PrintDest test.a2m A2M
```

Ces deux appels définissent le fichier **`test.a2m`** comme fichier résultat des impressions suivantes et ne vide pas ce fichier.

4.8.5.6 PRINTFONT

DESCRIPTION

Fixe la police de caractères pour l'impression des paragraphes de texte. En plus de la famille, la taille de base (en points) et l'incrément de la taille peuvent également être spécifiés.

La taille de base correspond aux paragraphes de textes, l'incrément indique de combien de points chaque niveau de titre doit être augmenté.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

Cette valeur n'a d'effet qu'avant la fonction $PrintDest et plus en cours de rapport.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintFont Times|Helvetica|Courier|Bookman|Palatino [size [incr]]
```

Les polices Bookman et Palatino ne sont pas exploitables pour toutes les destinations d'impression.

Le premier caractère du nom de la police suffit.

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintFont Times 10 2
```

Fixe la taille des paragraphes par_1, enum_1, etc à 10 points, le paragraphe tit_2 à 12 points, tit_1 à 14 points et tit_0 à 18 points.

4.8.5.7 PRINTTABLEFONT

DESCRIPTION

Fixe la police de caractères pour les tableaux seuls. Cette police peut donc être différente de celle du reste du texte, à la fois en taille et en famille.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

Cette fonction n'a d'effet qu'avant la fonction $PrintDest et plus en cours de rapport.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintTableFont Times|Helvetica|Courier|Bookman|Palatino [size]
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintTableFont Palatino 8
```

Fixe la police pour tous les tableaux à Palatino, 8 points.

4.8.5.8 PRINTTABLEBOX

DESCRIPTION

Fixe la largeur du cadre des tableaux en points. Cette valeur peut est nulle pour éviter les cadres.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintTableBox  n
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintTableBox  0
```

Supprime le cadre des tableaux.

4.8.5.9 PRINTTABLECOLOR

DESCRIPTION

Permet d'utiliser ou non la couleur dans les tableaux.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintTableColor [NO|Yes]
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintTableColor No
```

Supprime la couleur dans l'impression des tableaux.

4.8.5.10 PRINTTABLEWIDTH

DESCRIPTION

Fixe la largeur des tableaux en Frame et Rtf. Les paramètres doivent être exprimés en mm. Trois valeurs peuvent être passées :

- la largeur totale
- la largeur de la première colonne
- la largeur des colonnes suivantes

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintTableWidth width [col1 [coln]]
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintTableWidth 160 60 15
```

4.8.5.11 PRINTTABLEBREAK

DESCRIPTION

Permet ou non les coupures des tableaux sur plusieurs pages: si un tableau est placé sur la page de telle sorte qu'il doivent être imprimé sur deux pages, un saut de page automatique est généré avant le tableau.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintTableBreak [NO|Yes]
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintTableBreak Yes
```

Les coupures des tableaux sont autorisées lors de l'impression.

4.8.5.12 PRINTTABLEPAGE

DESCRIPTION

Force un saut de page avant chaque tableau.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintTablePage  [NO|Yes]
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintTablePage Yes
```

Force un saut de page pour chaque tableau.

4.8.5.13 PRINTBACKGROUND

DESCRIPTION

Fixe la couleur du background (tableaux et graphiques). Un tableau sera hachuré si la valeur des attributs graphiques "Shadow" est fixée à Yes.

Pour éviter les background dans tous les tableaux, on peut utiliser comme background White.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintBackground Black|Blue|Magenta|Cyan|Red|Green|Yellow|White
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintBackground Blue
```

La couleur de la partie hachurée en background des graphiques et tableaux est fixée à bleu.

4.8.5.14 PRINTGRAPHBOX

DESCRIPTION

Fixe la largeur du cadre de graphiques en points.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGraphBox n
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintGraphBox 2
```

Les graphiques suivants auront un cadre épais.

4.8.5.15 PRINTGRAPHBRUSH

DESCRIPTION

Fixe la densité du background des graphiques. La valeur est exprimée en pourcents.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGraphBrush pct|Yes
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintGraphBrush 10
```

La densité de la "brosse" de hachure est fixée à 10%. La couleur est déterminée par la fonction $PrintBackground.

4.8.5.16 PRINTGRAPHSIZE

DESCRIPTION

Fixe la taille des graphiques en mm et celle de la police utilisée.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGraphSize  width [height] [fontsize]
```

où :

- **`width`** et **`height`** sont exprimés en mm.
- **`fontsize`** est exprimé en points

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintGraphSize 150 60 8
```

Les graphiques suivants occuperont un cadre de 15 cm sur 6 cm et les textes auront 8 points de corps.

4.8.5.17 PRINTGRAPHPAGE

DESCRIPTION

Force un saut de page avant chaque graphique.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGraphPage [NO|Yes]
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintGraphPage Yes
```

Chaque graphique suivant sera placé en début de page.

4.8.5.18 PRINTRTFHELP

DESCRIPTION

Génère un fichier RTF pour un help Windows. Ce fichier pourra être compilé par la suite à l'aide du compilateur HCW de Microsoft.

Les fichiers RTF générés de la sorte ne sont pas identiques à ceux générés pour WinWord.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintRtfHelp [YES|No]
```

Une fois cette valeur fixée, elle reste d'application pour toutes les impressions suivantes, y compris dans les rapports suivants. Comme elle est sauvée dans le panneau de configuration des impressions, elle reste également valable pour les sessions suivantes de IODE.

Exemple

```
$PrintRtfHelp YES
```

Le fichier suivant défini par $PrintDest ... RTF sera du type Help Windows.

4.8.5.19 PRINTRTFTOPIC

DESCRIPTION

Crée un nouveau topic pour un fichier d'aide Windows (ou a2m). Ce topic fera partie de la table des matières automatique générés par IODE. Le niveau hiérarchique du topic est fixé par la commande **$PrintRtfLevel**.

Le texte du topic sera également repris comme titre.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintRtfTopic titre du topic
```

Exemple

```
$PrintRtfTopic Tableaux récapitulatifs
```

Génère en a2m (voir syntaxe du langage a2m) :

```
.topic [numéro_auto] [level] Tableaux récapitulatifs
.par1 tit_[level]
Tableaux récapitulatifs
```

Si le fichier généré est un fichier a2m, la commande **.topic** n'est exploitée que dans le cas d'une traduction en Help Windows. Par contre, **.par1 tit_...** et le texte du topic généreront à l'impression un titre.

4.8.5.20 PRINTRTFLEVEL

DESCRIPTION

Change le niveau hiérarchique des topics suivants. Ce changement de niveau permet de générer des indentations dans la table de matières du fichier d'aide résultat.

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintRtfLevel [+|-|n]
```

Exemple

```
$PrintRtfLevel +
```

Le niveau du topic suivant sera incrémenté de 1, créant de la sorte une indentation dans la table des matières générée.

4.8.5.21 PRINTRTFTITLE

DESCRIPTION

Fixe le titre du fichier d'aide Windows.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences/RTF"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintRtfTitle titre du help
```

Exemple

```
$PrintRtfTitle Modèle Hermes
```

4.8.5.22 PRINTRTFCOPYRIGHT

DESCRIPTION

Fixe le texte du copyright de l'aide Windows. Ce texte apparaît dans l'option Help du programme Winhelp.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences/RTF"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintRtfCopyright copyright text
```

Exemple

```
$PrintRtfCopyright (c) Bureau fédéral du Plan - 1998
```

4.8.5.23 PRINTHTMLHELP

DESCRIPTION

Spécifie que la prochaine génération de fichier Html devra se faire pour un HtmlHelp de Windows. Ce fichier pourra être compilé par la suite à l'aide du compilateur HHC de Microsoft.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintHtmlHelp [YES|No]
```

Exemple

```
$PrintHtmlHelp YES
$A2mToHtml myfile.a2m myfile.htm
```

4.8.5.24 PRINTHTMLSTRIP

DESCRIPTION

Spécifie que la prochaine génération de fichier Html ne doit pas inclure les sections de header et de footer HTML (<HEAD>, <BODY>, </BODY>, etc). Seul le texte proprement dit est traduit en vue d'être incorporé plus tard manuellement dans un fichier HTML.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintHtmlStrip [YES|No]
```

Exemple

```
$PrintHtmlStrip YES
$PrintDest test.htm HTML
$PrintTbl ....
```

4.8.5.25 PRINTPARANUM

DESCRIPTION

Permet de numéroter les titres lors de l'impression (1., 1.1, ...).

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

Cette fonction n'a d'effet qu'avant la fonction $PrintDest et plus en cours de rapport.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintParanum NO
$PrintParanum [NO|Yes]
```

Exemple

```
$PrintParanum NO
```

Supprime la numérotation automatique des titres.

4.8.5.26 PRINTPAGEHEADER

DESCRIPTION

Fixe le titre des pages imprimées. Les caractères %d sont remplacés par le numéro de la page courante.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences/Printer", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintPageHeader titre des pages suivantes
```

Exemple

```
$PrintPageHeader Modèle Modtrim
```

4.8.5.27 PRINTPAGEFOOTER

DESCRIPTION

Fixe la footnote des pages imprimées. Les caractères %d sont remplacés par le numéro de la page courante.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences/Printer", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintPageFooter  footnote des pages suivantes
```

Exemple

```
$PrintPageFooter - Page %d -
```

4.8.5.28 SETPRINTER

DESCRIPTION

Fixe l'imprimante Windows par défaut pour les impressions suivantes.

FONCTION INTERACTIVE

Cette option est modifiable dans Windows (Control Panel).

FONCTION NON INTERACTIVE

Syntaxe

```
$SetPrinter printer_name
```

Exemple

```
$SetPrinter 6_qms
```

4.8.5.29 PRINTORIENTATION

DESCRIPTION

Fixe l'orientation de l'imprimante Windows courante.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences/Printer", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintOrientation {Portrait|Landscape}
```

Exemple

```
$PrintOrientation Portrait
```

4.8.5.30 PRINTDUPLEX

DESCRIPTION

Fixe le mode recto-verso de l'imprimante Windows courante.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences/Printer", accessible dans la fonction "Print Setup".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintDuplex {Simplex|Duplex|VerticalDuplex}
```

Exemple

```
$PrintDuplex Duplex
```

4.8.5.31 PRINTGIFBACKCOLOR

DESCRIPTION

Définit la couleur de fond des graphiques GIF générés lors de la création d'un document HTML.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences/GIF"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGIFBackColor {Black|Blue|Magenta|Cyan|Red|Green|Yellow|White}
```

Exemple

```
$PrintGIFBackColor Yellow
```

4.8.5.32 PRINTGIFTRANSCOLOR

DESCRIPTION

Définit la couleur considérée comme "transparente" dans les fichiers GIF générés lors de la création d'un document HTML. Cela permet de faire en sorte qu'une des couleurs se confonde avec le fond de l'écran. L'option "Transparente" doit être définie par la commande **`$PrintGIFTransparent Yes`**.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences/GIF"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGIFTransColor {Black|Blue|Magenta|Cyan|Red|Green|Yellow|White}
```

Exemple

```
$PrintGIFTransColor White
```

4.8.5.33 PRINTGIFTRANSPARENT

DESCRIPTION

Indique si les fichiers GIF générés lors de la création d'un document HTML doivent être au format "transparent". Cela permet de faire en sorte qu'une des couleurs (à préciser par la commande $PrintGIFTransColor) se confonde avec le fond de l'écran.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences/GIF"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGIFTransparent {Yes|No}
```

Exemple

```
$PrintGIFTransparent No
```

4.8.5.34 PRINTGIFINTERLACED

DESCRIPTION

Indique si les fichiers GIF générés lors de la création d'un document HTML doivent être au format interlacé.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences/GIF"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGIFInterlaced {Yes|No}
```

Exemple

```
$PrintGIFInterlaced No
```

4.8.5.35 PRINTGIFFILLED

DESCRIPTION

Indique s'il faut remplir les barres dans les bar charts des graphiques GIF.

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Pre-**

ferences/GIF".

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGIFilled {Yes|No}
```

Exemple

```
$PrintGIFFilled No
```

4.8.5.36 PRINTGIFFONT

DESCRIPTION

Détermine la police de caractères à utiliser dans la création des graphiques en GIF. Les polices disponibles sont :

- 0: Tiny
- 1: Small
- 2: MediumBold
- 3: Large
- 4: Giant
- 5: Grayed

Cette fonction modifie également la valeur des champs correspondants dans le panneau **"Print Preferences/GIF"**.

FONCTION INTERACTIVE

Cette option est modifiable dans le Panneau "Print Preferences", accessible dans la fonction "Print Setup". Chaque destination peut prendre une valeur différente.

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintGIFFont FontNb (entre 0 et 5)
```

Exemple

```
$PrintGIFFont 1
```

4.8.6 IMPRESSIONS D'OBJETS

Les commandes suivantes sont utilisées pour imprimer les définitions d'objets :

- printobjdef : imprime une définition d'objet cmt | eqs | idt | lst | scl | tbl | var
- printobjtitle : indique si les titres seuls sont à imprimer
- printobjlec : indique le type d'équation à imprimer
- printobjinfos : indique les informations à imprimer

4.8.6.1 PRINTOBJDEFXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Cette commande permet l'impression au format A2M de la définition des objets du ws courant.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande.

La page de saisie contient deux champs: le premier est un champ MENU (voir lexique) : on y sélectionne le type d'objet à imprimer (le type CMT est proposé par défaut); le second est un champ EDITOR (voir lexique): on y spécifie la liste des objets à imprimer.

L'imression se fait sur l'imprimante sélectionnée lorsqu'on valide la page avec F10. La définition est imprimé en format A2M.

Si la liste des objets est vide, tous les objets du type spécifié dans le WS courant sont imprimés. La liste peut contenir des noms de listes ($...) et/ou une énumération de noms d'objets séparés par un blanc, une virgule ou un point-virgule.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#PrintObjDefCmt
```

FONCTION NON INTERACTIVE

La liste des commentaires à imprimer est passée en paramètre de la commande.

Syntaxe

```
$PrintObjDefXxx [liste de paramètres]
où Xxx = {cmt | idt | lst | scl | tbl | var}.
```

Exemple

```
$PrintObjDefCmt CMT1 CMT2
```

4.8.6.2 PRINTOBJTITLE

DESCRIPTION

Cette fonction s'utilise avant $PrintObjDefTbl. Elle indique si les titres des tableaux sont seuls à être imprimés ou si toute la définition doit l'être.

FONCTION INTERACTIVE

Les mêmes informations sont introduites dans l'écran "Print Objects definition".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintObjTitle 0 ou 1
0 : définition complète
1 : titres seuls
```

4.8.6.3 PRINTOBJLEC

DESCRIPTION

Cette fonction s'utilise avant $PrintObjDefXxx. Elle indique la façon dont des équations doivent être imprimées: telles quelles ou avec remplacement des coefficients par leurs valeurs ou encore avec les t-tests.

FONCTION INTERACTIVE

Les mêmes informations sont introduites dans l'écran "Print Objects definition".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintObjLec 0, 1 ou 2
0 : forme LEC normale
1 : remplacement des coefs par leur valeur
2 : remplacement des coefs par leur valeur+t-tests
```

4.8.6.4 PRINTOBJINFOS

DESCRIPTION

Cette fonction s'utilise avant $PrintObjDefEqs. Elle indique les informations à imprimer :

- équations seules
- équations, commentaires
- équations, commentaires et résultats d'estimation

FONCTION INTERACTIVE

Les mêmes informations sont introduites dans l'écran "Print Objects definition".

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintObjInfos 0, 1 ou 2
0 : équation seule
1 : équation + commentaire
2 : équation + commentaire + estimation
```

4.8.7 COMPILATION ET IMPRESSION DE TABLES

Les commandes suivantes sont utilisées pour compiler des tableaux et les imprimer ou les visualiser :

- PrintTblFile
- PrintTbl
- ViewTblFile
- ViewTbl
- ViewByTbl
- PrintVar
- ViewVar
- ViewWidth
- ViewWidth0
- ViewNdec

4.8.7.1 PRINTTBLFILE

DESCRIPTION

Cette commande permet de définir les fichiers à utiliser lors de l'impression d'un tableau de comparaison entre plusieurs fichiers (maximum 4). Le sample de l'impression peut contenir des références vers d'autres fichiers que le WS, placées entre crochets (ex. 90:2^[1;2]). Le fichier [1] est toujours le WS, les autres peuvent être définis ou dans la page de saisie ou comme argument dans un rapport.

L'objet de cette fonction est d'indiquer les fichiers référencés par [2], [3], etc.

FONCTION INTERACTIVE

Il n'existe pas de fonction équivalente en plein écran : la fonction #PrintTbl affiche une page de saisie dans laquelle on définit notamment les noms de fichiers à utiliser lors de l'impression.

Syntaxe

```
#PrintTbl
```

FONCTION NON INTERACTIVE

Le numéro de fichier (2, 3, 4 ou 5) suivi du nom du fichier sont passés en paramètre de la commande.

Syntaxe

```
$PrintTblFile n varfilename
(n := 2, 3, 4, 5)
```

Exemple

```
$wsloadvar bist92\bistel
$wsloadtbl bist92\tbistelf

$printdest bist92\bistelf1.a2m
#show processing french tables  file 1/2
$PrintNbDec 1
$PrintTblFile 2 variante
$PrintTbl 89:8[1;2] HYPEIR
```

Après un appel à **`$PrintTbl`**, il faut redéfinir les fichiers à comparer: **`$PrintTbl`** "consomme" les définitions de fichiers à comparer.

Il ne peut y avoir d'espace dans l'énoncé du sample.

4.8.7.2 PRINTTBL

DESCRIPTION

Cette commande permet le construction et l'impression des tableaux au format A2M.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande. Son fonctionnement est décrit dans le manuel de l'utilisateur.

Syntaxe

```
#PrintTbl
```

FONCTION NON INTERACTIVE

Le sample d'impression suivi de la liste des tableaux à imprimer sont passés comme paramètres de la commande.

Le dernier appel à $PrintTblFile permet d'imprimer des comparaisons de fichiers. Cependant, l'appel à $PrintTbl ou $PrintVar "consomme" les fichiers définis par $PrintTblFile. Si on veut faire appel plusieurs fois à $PrintTbl pour comparer des fichiers, il faut repréciser les fichiers à comparer avant chaque appel.

Syntaxe

```
$PrintTbl sample tableau1 [tableaux suivants]
```

Exemple

```
$wsloadvar bist92\bistel
$wsloadtbl bist92\tbistelf

$printdest bist92\bistelf1.a2m
#show processing french tables  file 1/2
$printnbdec 1
$PrintTbl 89:8 HYPEIR
$PrintTbl 89/88:8 HYPEIIR
$PrintTbl 89:8 RENAT01 RENAT0D RESL3R RESL31R RESL2
```

Il ne peut y avoir d'espace dans l'énoncé du sample.

Appels successifs

Le dernier appel à **$PrintTblFile** permet d'imprimer des comparaisons de fichiers. Cependant, l'appel à **$PrintTbl** ou **$PrintVar** "consomme" les fichiers définis par **$PrintTblFile.** Si on veut faire appel plusieurs fois à **$PrintTbl** pour comparer des fichiers, il faut repréciser les fichiers à comparer avant chaque appel.

Exemple

```
$PrintTblFile 2 ref.var
$PrintTblFile 3 var1.var
$PrintTbl 89:8[1;2;3] HYPEIR
$PrintTblFile 2 ref.var
$PrintTbl 89/88:8[1/2] HYPEIIR
```

4.8.7.3 VIEWTBLFILE

DESCRIPTION

Cette commande permet de définir les fichiers à utiliser lors de la visualisation d'un tableau de comparaison entre plusieurs fichiers (max 4). Le sample de l'impression peut contenir des références vers d'autres fichiers que le WS. (ex. 90:2[1;2]). Le fichier [1] est toujours le WS, les autres peuvent être définis ou dans la page de saisie ou comme argument dans un rapport.

FONCTION INTERACTIVE

Il n'y a pas de fonction directement équivalente en plein écran : la fonction d'impression de tableaux permet d'encoder les fichiers à comparer :

Syntaxe

```
#ViewTbl
```

FONCTION NON INTERACTIVE

Le numéro de fichier (2, 3, 4 ou 5) suivi du nom du fichier sont passés en paramètre de la commande.

Syntaxe

```
$ViewTblFile n varfilename
(n := 2, 3, 4, 5)
```

Exemple

```
$wsloadvar bist92\bistel
$wsloadtbl bist92\tbistelf

$PrintNbDec 1
$ViewTblFile 2 variante
$ViewTbl 89:8[1;2] HYPEIR
```

Après un appel à **`$ViewTbl`**, il faut redéfinir les fichiers à comparer: **`$ViewTbl`** "consomme" les définitions de fichiers à comparer.

4.8.7.4 VIEWTBL

DESCRIPTION

La fonction construit et affiche les tables dans un tableau déroulant (ou tableau d'édition).

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande. Cette fonction est décrite dans le "Manuel de l'utilisateur" : Menu Print/Graph, Option "Print Ta-

bles".

Syntaxe

```
#ViewTbl
```

FONCTION NON INTERACTIVE

Le sample d'impression suivi de la liste des tableaux à visualiser sont passés en paramètre de la commande.

Syntaxe

```
$ViewTbl sample tableau [liste de tableaux]
```

Exemple

$ViewTbl 90:6 CIIIR0 CIIR

Il ne peut y avoir d'espace dans l'énoncé du sample.

4.8.7.5 VIEWBYTBL

DESCRIPTION

Alias de la fonction $ViewTbl.

4.8.7.6 PRINTVAR

DESCRIPTION

Cette commande permet la construction et l'impression des tableaux de comparaison de séries en format A2M. Les formes LEC sont admises et doivent être séparées par des points-virgules.

Pour définir les fichiers à comparer, il faut utiliser la fonction $PrintTblFile comme pour les tableaux et graphiques.

FONCTION INTERACTIVE

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#PrintVar
```

FONCTION NON INTERACTIVE

Syntaxe

```
$PrintVar gsample var1;var2
```

Exemple

```
$PrintVar (80Y1/79Y1):10 X+Y;Z;ln(X)
```

Il ne peut y avoir de blancs dans le sample généralisé.

4.8.7.7 VIEWVAR

DESCRIPTION

Cette commande permet la visualisation des tableaux de comparaison de séries en format A2M.

FONCTION INTERACTIVE

Cette fonction n'existe pas en mode plein écran.

FONCTION NON INTERACTIVE

Syntaxe

```
$ViewVar gsample lec1 [lec2 ...]

où gsample est un sample généralisé sans blanc
   lec1 et lec2 sont des formes LEC sans blanc
```

Exemple

```
$ViewVar (80Y1/79Y1):10  X+Y  Z  ln(X)
```

Il ne peut y avoir de blancs dans le sample généralisé.

4.8.7.8 VIEWWIDTH

DESCRIPTION

Cette fonction permet de spécifier la largeur des colonnes lors de l'affichage de tableaux (calculés). La valeur doit être comprise entre 2 et 12.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F3 et s-F3 permettent de changer la largeur des colonnes lors de l'affichage.

FONCTION NON INTERACTIVE

Syntaxe

```
$ViewWidth n
n entre 2 et 12
```

4.8.7.9 VIEWWIDTH0

DESCRIPTION

Cette fonction permet de spécifier la largeur de la première colonne (colonne de titres) lors de l'affichage de tableaux (calculés). La valeur doit être comprise entre 2 et 60.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F2 et s-F2 permettent de changer la largeur de la première colonne lors de l'affichage.

FONCTION NON INTERACTIVE

Syntaxe

```
$ViewWidth0 n
n entre 2 et 60
```

4.8.7.10 VIEWNDEC

DESCRIPTION

Cette fonction permet de spécifier le nombre de décimale des valeurs affichées dans les tableaux (calculés). La valeur doit être comprise entre -1 (nombre variable) et 6.

FONCTION INTERACTIVE

Cette fonction n'est pas disponible en mode plein écran. Les touches F4 et s-F4 permettent de changer le nombre de décimales lors de l'affichage.

FONCTION NON INTERACTIVE

Syntaxe

```
$ViewNdec n
n entre -1 (variable) et 6
```

4.8.8 GRAPHIQUES À PARTIR DE TABLEAUX

Les commandes suivantes sont utilisées pour compiler des tableaux et les visualiser sous forme de graphiques ou les sauver dans un fichier A2M.

- ViewGr
- PrintGrAll (obsolete)
- PrintGrWin (obsolete)
- PrintGrData (obsolete)
- PrintGr
- DataPrintGraph

4.8.8.1 VIEWGR

DESCRIPTION

Cette fonction permet de visualiser des tableaux sous forme de graphique. Plusieurs graphiques peuvent être affichés sur le même écran. Le langage est défini par $PrintLang.

FONCTION INTERACTIVE

Cette fonction est décrite en détail dans le "Manuel de l'utilisateur".

Syntaxe

```
#ViewGr
```

FONCTION NON INTERACTIVE

Syntaxe

```
$ViewGr gsample tbl1[+tbl2] tbl3 ...

où gsample est un sample généralisé sans blanc
   tbl1, tbl2 et tbl3 sont des noms de tableaux
```

Lorsque deux tableaux sont associés par le signe +, ils sont affichés sur le même écran sous forme de deux graphiques séparés.

Il ne peut y avoir de blancs dans le sample généralisé.

4.8.8.2 PRINTGRALL

Cette fonction est annulée à partir de la version 5 de IODE. Elle est remplacée par la fonction **$PrintGr**.

4.8.8.3 PRINTGRWIN

Cette fonction est annulée à partir de la version 5 de IODE.

4.8.8.4 PRINTGRDATA

Cette fonction est annulée à partir de la version 5 de IODE.

4.8.8.5 PRINTGR

Cette fonction imprime un ou des graphiques définis à partir de tableaux.

Syntaxe

```
$PrintGr gsample table_names
```

Exemple

```
$PrintGr 80/79:15 T1 T2
```

4.8.8.6 DATAPRINTGRAPH

DESCRIPTION

Cette fonction imprime des graphiques construits directement à partir de variables (sans passer par des tableaux). Elle correspond à l'utilisation de la touche F8 lors de l'édition d'un WS de variables.

Syntaxe

```
$DataPrintGraph   {Level | Diff | Grt}
                  {Line | Scatter | Bar | Mixt}
                  {NoXGrids | MinorXGrids | J(MajorXGrids)}
                  {NoYGrids | MinorYGrids | J(MajorYGrids)}
                  {Level | G(Log) | Semi-Log | Percent}
                  {ymin | --} {ymax | --}
                  perfrom perto varname1 varname2 ...
```

Exemple

```
$DataPrintGraph  Level Line No No Level -- -- 1980Y1 1995Y1 X1 Y1 X1+Y1
```

4.8.9 Opérations sur des modèles

Les commandes suivantes permettent de trier et de simuler un modèle, d'estimer et de recompiler des équations :

- ModelCalcSCC
- ModelSimulateParms
- ModelSimulate
- ModelSimulateSCC
- ModelExchange
- ModelCompile
- ModelSimulateSaveNiters
- ModelSimulateSaveNorms

4.8.9.1 ModelCalcSCC

Cette fonction permet de décomposer le modèle en composantes fortement connexes (CFC ou SCC pour Strongly Connex Components) et de le réordonnancer.

Trois listes sont donc créées : équations prérécursives, interdépendantes et postrécursives.

Lors du réordonnancement du modèle, le nombre d'itérations de triangulation (tri) pour le block interdépendant doit être spécifié. Cette valeur n'a évidemment d'effet que sur la liste des équations interdépendantes.

Ces 3 listes ne doivent être contruites qu'une seule fois pour une version donnée du modèle.

Syntaxe

```
$ModelCalcSCC nbtri lstpre lstinter lstpost
```

où :

- nbtri est le nombre d'itérations de triangulation à effectuer
- lst* sont les NOMS des listes destinées à contenir les résultats du tri des équations

Choix du nombre de tris

Après la décompostion en CFC, le bloc interdépendant est trié pour augmenter la vitesse de la simulation. Le nombre de passage de l'algorithme de tri peut être spécifié à plusieurs endroits :

- Dans l'écran de simulation "standard" : paramètre "Passes" fixé
- Dans l'écran de calcul de la décomposition du modèle : paramètre "Triangulation Iterations"
- Comme paramètre dans la commande rapport $ModelCalcSCC
- Comme dernier paramètre dans la commande rapport $ModelSimulateParms

Dans les versions antérieures, le nombre de passages de la triangulation spécifié dans l'écran de simulation n'avait pas d'effet

4.8.9.2 MODELSIMULATEPARMS

DESCRIPTION

Cette commande permet de spécifier les paramètres complémentaires (essentiellement techniques) d'une simulation. La simulation est effectivement lancée par la commande ModelSimulate.

Les paramètres que fixe cette commande sont décrits dans le "Manuel de l'utilisateur".

FONCTION INTERACTIVE

Il n'y a pas de fonction spécifique plein écran équivalente. L'écran de simulation appelé par #ModelSimulate contient les paramètres de simulation.

Syntaxe

```
#ModelSimulate
```

FONCTION NON INTERACTIVE

Cette fonction fixe les paramètres suivants de simulation :

- seuil de convergence (eps)
- paramètre de relaxation (relax)
- nombre maximum d'itérations (maxit)
- algorithme de réordonnacement (Connex, Triangulation ou None)
- valeurs initiales (0 à 4)
  - 0 = Y := Y[-1], if Y null or NA
  - 1 = Y := Y[-1], always
  - 2 = Y := extrapolation, if Y null or NA
  - 3 = Y := extrapolation, always
  - 4 = Y unchanged
- simulation avec debugging
- debugging de la sous-itération de Newton-Raphson
- nombre d'itération de tri des composantes connexes

Syntaxe

```
$ModelSimulateParms eps relax maxit
                    {Connex | Triang | None }
                    0 - 4 (starting values)
                    {Yes | no }
                    {yes | No }
                    nbtri
(eps   := seuil de convergence : nombre réel)
(relax := nombre réel compris entre 0.1 et 1)
```

Exemple

```
$ModelSimulateParms 0.001 0.7 100 Connex 0 Yes No 2
```

Option Debug

L'option debug de la simulation génère automatiquement des listes comprenant des équations pré- et post-récursives.

Version 6.36 : l'option debug activait la génération d'un fichier simul.dbg qui contenait une quantité énorme d'informations. Dans cette version, seules les listes **_PRE**, **_INTER** *et* **_POST** *(avec la découpe du modèle) sont générées.*

Option Debug sous-itération

Un fichier de trace est généré si cette option est fixée à "Y". Ce fichier se nomme "simul.dbg" et se trouve dans le directory d'où IODE a été lancé. Le fichier de trace comprend la liste des équations pour lesquelles une sous-itération (Newton-Raphson) a été nécessaire et le nombre de sous-itérations.

Choix du nombre de tris

Après la décompostion en CFC, le bloc interdépendant est trié pour augmenter la vitesse de la simulation.

4.8.9.3 MODELSIMULATE

DESCRIPTION

La fonction lance la simulation d'un modèle.

Pour simuler un modèle, il faut que toutes les équations du modèle soient présentes dans le WS d'équations, que toutes les variables et tous les scalaires utilisés dans les équations du modèle soient définis dans les WS de variables et de scalaires. Les valeurs des exogènes et des scalaires ne peuvent être NA (Not Available) sur la période utile.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de simulation. Cette fonction est décrite dans "Manuel de l'utilisateur".

Syntaxe

```
#ModelSimulate
```

FONCTION NON INTERACTIVE

Les paramètres de la simulation sont définis par la commande annexe ModelSimulateParms. La simulation proprement dite s'effectue par la commande ModelSimulate.

Les deux périodes limites (début et fin) et la liste des équations à simuler sont passées en argument à la fonction. Si la liste est vide, la simulation sera effectuée pour le modèle constitué de toutes les équations en mémoire.

Les paramètres complémentaires peuvent être définis par la commande modelsimulateparms décrite ci-dessous.

Syntaxe

```
$ModelSimulate periode1 periode2 [liste_d_equations]
```

Exemple

```
$-filedeletea2m ode.a2m
$printdest ode.a2m
#modelsimulate
$modelsimulate 1990Y1 1995Y1
$show test1
$modelsimulate 1990Y1 1995Y1 PC0 PIF
$show test2
$modelsimulate 1990Y1 1995Y1 $LIST
$show end of report
```

4.8.9.4 MODELSIMULATESCC

DESCRIPTION

La fonction lance la simulation d'un modèle préalablement décomposé en CFC et trié.

La performance des simulations, spécifiquement lors du démarrage (link, sort), est meilleure avec cette fonction qu'avec $ModelSimulate car le réordonnancement du modèle n'a lieu qu'une seule fois et non à chaque lancement de simulation.

Le processus de simulation a été divisé en 2 étapes. La première ne s'occupe que du réordonnancement du modèle ($ModelCalcSCC), la seconde de la simulation.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de simulation et les listes qui composent le modèle trié.

Syntaxe

```
#ModelSimulateSCC
```

FONCTION NON INTERACTIVE

La simulation du modèle se base sur trois listes préalablement construites par la fonction $ModelCalcSCC (ou à la main).

Sa syntaxe est :

```
$ModelSimulateSCC from to pre inter post
```

où :

- from et to déterminent le sample de simulation
- pre, inter et post sont les listes qui définissent l'ordre d'exécution du modèle.

4.8.9.5 ModelExchange

Description

Cette fonction permet de définir ou d'annuler les échanges endogènes-exogènes avant de lancer la simulation d'un modèle avec recherche d'objectif (Goal seeking) :

Fonction interactive

Cette fonction fait partie de la fonction de simulation :

Syntaxe

```
#ModelSimulate
```

Fonction non interactive

Syntaxe

```
$ModelExchange eqname1-varname1,eqname2-varname2,...

où eqname1 et eqname2 sont des noms d'équations (et donc
                    de variables endogènes)
   varname1 et varname2 sont des noms d'exogènes
```

Pour annuler l'échange ENDO-EXO, il suffit d'appeler la fonction sans paramètre :

```
$ModelExchange
```

4.8.9.6 ModelCompile

Description

Cette fonction recompile une liste d'équations ou toutes les équations si aucune liste n'est spécifiée. Elle est uniquement utile si des équations font appel à des macros dans leur forme LEC.

Fonction interactive

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#ModelCompile
```

FONCTION NON INTERACTIVE

Syntaxe

```
$ModelCompile  [eqname1, eqname2, ... ]
où eqname1 et eqname2 sont des noms d'équations.
```

4.8.9.7 MODELSIMULATESAVENITERS

DESCRIPTION

Cette commande permet de sauver dans une variable le nombre d'itérations qui ont été nécessaires pour chaque période lors de la dernière simulation.

Le nom de la variable résultat est passé comme paramètre.

S'il n'y a pas encore eu de simulation dans le cours de la session, la variable est créée avec des valeurs NA.

Si une simulation a déjà eu lieu, les valeurs des périodes non simulées sont fixées à 0 et celles des périodes simulées contiennent le nombre d'itérations nécessaires à la convergence pour cette période. S'il n'y a pas de convergence, la valeur est celle du paramètre MAXIT de la simulation.

FONCTION INTERACTIVE

Il n'y a pas de fonction spécifique plein écran équivalente.

FONCTION NON INTERACTIVE

Syntaxe

```
$ModelSimulateSaveNiters varname
```

Exemple

```
$ModelSimulateSaveNiters SIM_NITERS
```

4.8.9.8 MODELSIMULATESAVENORMS

DESCRIPTION

Cette commande permet de sauver dans une variable le seuil de convergence (la norme) atteint pour chaque période lors de la dernière simulation.

Le nom de la variable résultat est passé comme paramètre.

S'il n'y a pas encore eu de simulation dans le cours de la session, la variable est créée avec des valeurs NA.

Si une simulation a déjà eu lieu, les valeurs des périodes non simulées sont fixées à 0 et celles des périodes simulées contiennent le seuil de convergence pour cette période.

FONCTION INTERACTIVE

Il n'y a pas de fonction spécifique plein écran équivalente.

FONCTION NON INTERACTIVE

Syntaxe

```
$ModelSimulateSaveNorms varname
```

Exemple

```
$ModelSimulateSaveNiters SIM_NORMS
```

4.8.10 EXÉCUTIONS D'IDENTITÉS

Les commandes suivantes permettent d'exécuter des identités :

- idtexecute
- idtexecutetrace
- idtexecutevarfiles
- idtexecutesclfiles

4.8.10.1 IDTEXECUTE

DESCRIPTION

Une identité est une forme LEC qui indique comment une série doit être construite. Le programme d'exécution calcule toutes les identités demandées et sauve dans le WS de séries le résultat des calculs.

Ces identités sont ordonnancées dans l'ordre logique d'exécution : si A dépend de B, B sera calculé avant A. L'utilisateur n'a pas à se préoccuper de cet ordonnancement. Si une boucle est détectée dans les identités, l'exécution ne peut commencer et un message est affiché.

FONCTION INTERACTIVE

Cette fonction est décrite dans le "Manuel de l'utilisateur". Elle est appelée par :

Syntaxe

```
#IdtExecute
```

FONCTION NON INTERACTIVE

Les deux périodes limites (début et fin) et la liste des identités à calculer sont passées en argument à la fonction. Si la liste est vide, le calcul sera effectuée pour toutes les identités en mémoire.

Les paramètres complémentaires peuvent être définis par les commandes IdtExecuteVarFiles et IdtExecuteSclFiles décrites ci-dessous.

Syntaxe

```
$IdtExecute periode1 periode2 [liste_d_identites]
```

Exemple

```
$IdtExecute 1953Y1 1996Y1 IDT1 IDT2
$IdtExecute 1953Y1 1996Y1 $IDTLIST
```

4.8.10.2 IDTEXECUTETRACE

DESCRIPTION

La trace de la construction des identités peut être sauvée dans le fichier A2M courant défini par $PrintDest. Les informations utiles (par exemple, le fichier d'origine d'une variable utilisée pour construire l'identité) seront énumérées dans le fichier "Trace file".

FONCTION INTERACTIVE

Il n'y a pas de fonction équivalente interactive.

FONCTION NON INTERACTIVE

Selon le paramètre, la trace est sauvée ou non lors du prochain appel à la commande $IdtExecute.

```
$IdtExecuteTrace  {Yes | No}
```

4.8.10.3 IDTEXECUTEVARFILES

DESCRIPTION

Cette commande fait partie d'un groupe de commandes permettant de spécifier les paramètres complémentaires (fichiers de recherche) du calcul des identités. Les commandes associées sont IdtExecute et IdtExecuteSclFiles. Le calcul des identités est effectivement lancé par la commande IdtExecute.

La fonction permet d'introduire les noms de fichiers où les séries référencées dans les identités doivent être recherchés.

Lorsqu'une série ou un scalaire utile à l'exécution d'une identité est rencontré, les fichiers indiqués sont analysés à tour de rôle. Dès que l'objet recherché est trouvé, il est amené en mémoire pour la durée du calcul. Il est ensuite détruit, sauf s'il était initialement en mémoire.

Si un objet - variable ou scalaire - n'est ni en WS, ni dans un des fichiers indiqués, le processus de calcul s'arrête.

Cette fonction n'a tout son sens que lorsque la commande IdtExecute est utilisée en mode ligne de commande.

Si la liste des fichiers est vide, seul le WS courant de variables est utilisé pour la recherche. Sinon, il faut spécifier WS (en majuscules) dans la liste des fichiers, à la position de recherche souhaitée.

FONCTION INTERACTIVE

Il n'y a pas de fonction plein écran équivalente : l'écran d'exécution d'identités contient ces paramètres.

Syntaxe

```
#IdtExecute
```

FONCTION NON INTERACTIVE

La liste des fichiers est passée en paramètre de la fonction.

Syntaxe

```
$IdtExecuteVarFiles fichier [liste de fichiers]
```

Exemple

```
$IdtExecuteVarFiles maribel.var WS test.var
```

4.8.10.4 IDTEXECUTESCLFILES

Description

Cette commande fait partie d'un groupe de commandes permettant de spécifier les paramètres complémentaires (fichiers de recherche) du calcul des identités. Les commandes associées sont IdtExecuteVarFiles et IdtExecute. Le calcul des identités est effectivement lancé par la commande IdtExecute.

La fonction permet d'introduire les noms de fichiers où les scalaires référencés dans les identités doivent être recherchés.

Lorsqu'un scalaire utile à l'exécution d'une identité est rencontré, les fichiers indiqués sont analysés

à tour de rôle. Dès que l'objet recherché est trouvé, il est amené en mémoire pour la durée du calcul. Il est ensuite détruit, sauf s'il provient du WS actif.

Si un objet - variable ou scalaire - n'est ni en WS, ni dans un des fichiers indiqués, le processus de calcul s'arrête.

Cette fonction n'a tout son sens que lorsque la commande IdtExecute est utilisée en mode ligne de commande.

Si la liste des fichiers est vide, seul le WS courant de scalaires est utilisé pour la recherche. Sinon, il faut spécifier WS (en majuscules) dans la liste des fichiers, à la position de recherche souhaitée.

FONCTION INTERACTIVE

Il n'y a pas de fonction plein écran équivalente : l'écran d'exécution d'identités contient ces paramètres.

Syntaxe

```
#IdtExecute
```

FONCTION NON INTERACTIVE

La liste des fichiers est passé en paramètre de la fonction.

Syntaxe

```
$IdtExecuteSclFiles fichier [liste de fichiers]
```

Exemple

```
$IdtExecuteSclFiles WS maribel.scl test.scl
```

4.8.11 OPÉRATIONS SUR DES RAPPORTS

Les commandes suivantes permettent de manipuler des rapports (ces fonctions sont récursives):

- reportexec
- reportedit

4.8.11.1 REPORTEXEC

DESCRIPTION

La commande provoque l'exécution d'un rapport "fils". La fonction est récursive et autorise plusieurs niveaux de sous-rapports. Lorsque les rapports "fils" sont terminés sans erreur critique, l'exécution continue avec les commandes du rapport de niveau supérieur.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou les paramètres de la commande. Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#ReportExec
```

FONCTION NON INTERACTIVE

Le rapport à exécuter et ses arguments sont passés en paramètre de la commande.

Syntaxe

```
$ReportExec rapport [arguments]
```

Exemple

```
$ReportExec mysimul.rep 1990Y1 1992Y1
```

La fonction $ReportExec accepte une liste comme argument :

```
$ReportExec test $LISTE
```

reçoit comme argument les éléments de la liste $LISTE et permet donc d'exécuter un rapport type sur un nombre initialement inconnu d'arguments. Par exemple, pour remplacer tous les 0 par des NA pour la liste $LISTE de variables, il suffit de faire :

```
FICHIER ZEROTONA.REP

    $label again
    $goto fin {%0%=0}
    #show Series traitée : %1%
    $DataCalcVar %1% if(%1%=0, 1/0, %1%)
    $shift
    $goto again
    $label fin
```

La commande

```
    $ReportExec zerotona $LISTE
```

replacera les 0 par des NA pour les séries de la liste $LISTE.

4.8.11.2 REPORTEDIT

DESCRIPTION

Cette fonction permet de créer ou de modifier un rapport.

FONCTION INTERACTIVE

La fonction affiche une page de saisie dans laquelle on définit le ou le nom du rapport à éditer.

Cette fonction est décrite dans le "Manuel de l'utilisateur".

Syntaxe

```
#ReportEdit
```

FONCTION NON INTERACTIVE

Le fichier à éditer est passé en paramètre de la commande. L'éditeur MMT est lancé pour l'édition du fichier. L'extension .rep peut être omise.

Syntaxe

```
$ReportEdit filename
```

Exemple

```
$ReportEdit bist92\bistel.rep
```

4.8.12 INTERFACE EXCEL

Les fonctions suivantes permettent d'interagir avec Excel à partir d'un rapport de IODE. Il est également possible d'utiliser les données de IODE dans Excel à travers le serveur DDE de IODE. Le détail et des exemples sur ce possibilités peuvent être trouvés dans un chapitre séparé.

- ExcelOpen : lance Excel et ouvre un fichier Excel
- ExcelSave : sauve le fichier courant d'Excel
- ExcelSaveAs : sauve le fichier courant d'Excel sous un autre nom
- ExcelClose : ferme le fichier courant d'Excel
- ExcelNew : crée un nouveau fichier vide dans Excel
- ExcelPrint : imprime le fichier courant d'Excel
- ExcelSetXxx : copie des objets de IODE dans Excel
- ExcelWrite : écrit des informations quelconques dans un sheet Excel
- ExcelGetVar : copie dans IODE des variables définies dans Excel
- ExcelGetScl : copie dans IODE des scalaires définis dans Excel
- ExcelGetCmt : copie dans IODE des commentaires définis dans Excel
- ExcelLang : informe IODE de la langue d'Excel pour formater les ranges
- ExcelDecimal : informe IODE du séparateur décimal utilisé par Excel pour formater les variables envoyées à Excel.
- ExcelCurrency: informe IODE du symbole de devise utilisé par Excel
- ExcelThousand: informe IODE du séparateur de milliers utilisé par Excel

4.8.12.1 ExcelOpen

Description

Excel opent de file FileName. In FileName geeft u het volledig pad aan waar Excel u file kan vinden.

Als Excel niet draait en het pad naar Excel.exe staat in uw systeemvariabele PATH dan wordt Excel opgestart. Indien niet dan krijgt u een foutmelding.

Fonction interactive

Deze functie heeft geen Fullscreen tegenhanger

Fonction non interactive

Syntaxe

```
$ExcelOpen FileName
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelSetTbl INF 90:11 Sheet1!R1C1
$ExcelSaveAs c:\usr\iode\inflatie.xls
```

4.8.12.2 ExcelSave

Description

Slaat de openstaande workbook op onder haar oorpsronkelijke naam.

Fonction interactive

Deze functie heeft geen Fullscreen tegenhanger

Fonction non interactive

Syntaxe

```
$ExcelSave
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelSetTbl INF 90:11 Sheet1!R1C1
$ExcelSave
```

4.8.12.3 EXCELSAVEAS

DESCRIPTION

Excel opent de file FileName. In FileName geeft u het volledig pad aan waar Excel u file kan vinden.

Als FileName al bestaat krijgt u de gewone Excel-foutmeldingen

Als Excel niet draait en het pad naar Excel.exe staat in uw systeemvariabele PATH dan wordt Excel opgestart. Indien niet dan krijgt u een foutmelding.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelSaveAs FileName
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelSetTbl INF 90:11 Sheet1!R1C1
$ExcelSaveAs c:\usr\iode\inflatie.xls
```

4.8.12.4 EXCELCLOSE

DESCRIPTION

Excel opent de file FileName. In FileName geeft u het volledige pad aan waar Excel uw file kan vinden.

Als Excel niet draait en het pad naar Excel.exe staat in uw systeemvariabele PATH dan wordt Excel opgestart. Indien niet dan krijgt u een foutmelding.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelOpen FileName
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelSetTbl INF 90:11 Sheet1!R1C1
$ExcelSaveAs c:\usr\iode\inflatie.xls
```

4.8.12.5 EXCELNEW

DESCRIPTION

Excel creëert een nieuw spreadsheet, afhankelijk van de versie en de configuratie van uw Excel-omgeving heeft dit werkbook 3 (Excel 8) of 14 werkbladen (Excel 7)

Als Excel niet draait en het pad naar Excel.exe staat in uw systeemvariabele PATH dan wordt Excel opgestart. Indien niet dan krijgt u een foutmelding.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelNew
```

Exemple

```
$ExcelNew
$ExcelSetTbl INF 90:11 Sheet1!R1C1
$ExcelSaveAs c:\usr\iode\inflatie.xls
```

4.8.12.6 EXCELPRINT

DESCRIPTION

Print het openstaande workbook op uw Windows-default printer.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelPrint
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelSetTbl INF 90:11 Sheet1!R1C1
$ExcelPrint
```

4.8.12.7 EXCELSETXXX

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Deze functie plaatst IODE-objecten in een openstaand Excel-werkblad. Zij is alleen beschikbaar voor variabelen, scalairen, commentaren, lijsten en berekende tabellen.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

ExcelSetXxx neemt een aantal argumenten, afhankelijk van het type van het object. Het eerste argument is altijd de naam van het IODE-object. Het laatste geeft de plaats (range) aan waar u in Excel de informatie terugvindt.

Een range in Excel bestaat uit de naam van het werkblad gevolgd door de namen van de linkerboven cel en de rechteronder cel. Zo duidt "Sheet1!R1C1:R2C4" de eerste vier cellen van de eerste rij en de tweede rij aan. Om het u gemakkelijker te maken volstaat het de naam van de linkerboven cel te geven, IODE vult de rest aan naargelang de grootte van het te transfereren object.

Commentaren

```
$ExcelSetCmt naam excelrange
```

Exemple

```
$ExcelNew
$ExcelSetCmt A0G Sheet1!R1C1
```

Tabellen

```
$ExcelSetTbl naam {gsample} excelrange
```

gsample is een generalised sample zoals u dit kent van het PrintTbl commando. Vult u hier echter geen gsample in dan wordt de tabel afgedrukt over de gehele sample van de workspace.

Exemple

```
$ExcelNew
$ExcelSetTbl INF 90:11 Inflation!R1C1
$ExcelPrint
```

Variabelen

```
$ExcelSetVar naam excelrange
```

Exemple

```
$ExcelNew
$ExcelSetVar A0G Sheet1!R1C1
```

De gevraagde variabele wordt over de gehele workspace door gegeven.

4.8.12.8 ExcelWrite

Description

Cette fonction permet d'écrire du texte ou des valeurs quelconques dans une feuille Excel.

Fonction interactive

N'existe pas sous forme Fullscreen.

Fonction non interactive

Remplace les cellules spécifiées par la ou les valeurs données. Plusieurs colonnes sont séparées par `\t` et plusieurs lignes par `\n`.

Syntaxe

```
$ExcelWrite MySheet!R4C4 texte[\ttexte][\ntexte][...]
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelGetCmt MySheet!R1C1 Titre\n A:\t{A[1970Y1]}\t{A[1980Y1]}\t{A[1990Y1]}
$ExcelSaveAs c:\usr\iode\test.xls
```

4.8.12.9 EXCELGETVAR

DESCRIPTION

Deze functie haalt waarden uit een Excel spreadsheet en doet een update van de variable in IODE.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Bestaat de variabele al dan worden de oude waarden overschreven met die uit Excel. Bestaat ze nog niet dan wordt de variabele gemaakt en krijgt als waarde de waarden uit het Excel-spreadsheet.

Deze functie gelijkt sterk op het DataUpdateVar commando, alleen worden hier de nieuwe waarden via een Excel-range gedefinieerd.

Syntaxe

```
$ExcelGetVar VARNAME [periode] excelrange
```

Haalt uit de Excel-range de waarden voor de update van "varname". Wordt de "begin period" niet aangegeven, dan wordt de begin periode van de sample aangenomen.

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelGetVar A 1980Y1 Sheet1!R1C1:R2C10
$ExcelSaveAs c:\usr\iode\inflatie.xls
```

4.8.12.10 EXCELGETSCL

DESCRIPTION

Deze functie haalt waarden uit een Excel spreadsheet en doet een update van de scalaire in IODE.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Bestaat de scalaire al dan worden de oude waarde overschreven met die uit Excel. Bestaat ze nog niet dan wordt de scalaire gemaakt en krijgt als waarde de waarde uit het Excel-spreadsheet.

Syntaxe

```
$ExcelGetVar SCLNAME excelrange
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelGetScl c1 Sheet1!R1C1:R1C1
$ExcelSaveAs c:\usr\iode\inflatie.xls
```

4.8.12.11 ExcelGetCmt

DESCRIPTION

Deze functie haalt text uit een Excel spreadsheet en doet een update van de commentaar in IODE.

FONCTION INTERACTIVE

Deze functie heeft geen Fullscreen tegenhanger

FONCTION NON INTERACTIVE

Bestaat de commentaar al dan wordt de oude waarde overschreven met die uit Excel. Bestaat ze nog niet dan wordt de commentaar gemaakt en krijgt als waarde de text uit het Excel-spreadsheet.

Syntaxe

```
$ExcelGetCmt CMTNAME excelrange
```

Exemple

```
$ExcelOpen c:\usr\iode\test.xls
$ExcelGetCmt A Sheet1!R1C1:R1C1
$ExcelSaveAs c:\usr\iode\inflatie.xls
```

4.8.12.12 ExcelLang

DESCRIPTION

Par défaut, IODE considère que la version linguistique de Excel est l'anglais. Cela implique que les `Ranges` dans Excel doivent s'écrire `RnCn` (par exemple R2C4 pour ligne (Row) 2, colonne (Column) 4).

Dans les autres langues, cette écriture ne convient plus et IODE doit envoyer les données par exem-

ple vers `L2C4` au lieu de `R2C4` dans la version française.

Pour assurer qu'un rapport IODE reste valable quelle que soit la langue d'Excel, la commande **`$ExcelLang`** a été ajoutée à IODE.

FONCTION INTERACTIVE

N'existe pas sous forme Fullscreen.

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelLang {F|N}
```

Tout autre paramètre (E par exemple) remettra IODE en version anglaise.

Dans l'exemple ci-dessous, IODE enverra le contenu de la variable AAA à partir de la cellule L1C1 du Sheet1 du fichier Excel ouvert.

Exemple

```
$ExcelLang F
$ExcelSetVar AAA Sheet1!R1C1
```

Note

Il faut noter que dans la commande **`$ExcelSetXxx`**, **le range reste défini comme** `R1C1`. C'est uniquement en interne qu'IODE transformera cette information en `L1C1`.

4.8.12.13 EXCELDECIMAL

DESCRIPTION

Par défaut, IODE envoie les données numériques vers Excel en utilisant **le séparateur décimal défini dans les "Regional Settings" de Windows (voir Control Panel)**.

Cette option ne fonctionne que si le séparateur décimal n'a pas été modifié dans Excel : on peut en effet décider d'utiliser le **point décimal** dans Excel alors que dans Windows, c'est la **virgule** qui a été choisie.

Pour permettre de forcer l'envoi vers une version Excel dans laquelle le séparateur décimal ne serait pas celui de Windows, une nouvelle fonction de rapport a été introduite.

FONCTION INTERACTIVE

N'existe pas sous forme Fullscreen.

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelDecimal {C}
```

L'absence de paramètre C ou tout autre paramètre que C (comma) sélectionne le point décimal.

Dans l'exemple ci-dessous, IODE enverra le contenu de la variable AAA avec des virgules comme séparateur décimal à partir de la cellule L1C1 du Sheet1 du fichier Excel ouvert.

Exemple

```
$ExcelDecimal C
$ExcelLang F
$ExcelSetVar AAA Sheet1!R1C1
```

4.8.12.14 ExcelCurrency

DESCRIPTION

Par défaut, IODE lit la définition de la devise dans les paramètres "régionaux" (Regional Settings). Si cette valeur ne convient pas, elle peut être modifiée par la commande de rapport $ExcelCurrency.

FONCTION INTERACTIVE

N'existe pas sous forme Fullscreen.

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelCurrency [char]
```

La valeur de char remplace celle du caractère indiquant la devise dans les paramètres de Windows. Lorsque ce caractère est rencontré lors du transfert d'Excel vers IODE, il est ignoré.

Certaines valeurs spéciales de **`char`** ont une interprétation particulière :

- d, D : dollar
- e ou E : euro
- p ou P : sterling pound
- pas d'argument : il n'y a pas de signe de devise
- tout autre valeur est prise telle quelle comme signe de devise

Exemple

```
    |    B       C       D       E
--  |-------------------------------
3   | $10,000 $20,000 $30,000 $40,000
4   | $12,345 $23,456 $34,567 $45,678
```

Rapport :

```
$WSSample 1989Y1 2000Y1
$ExcelThousand C
$ExcelCurrency D
$ExcelGetVar A 1990Y1 Sheet1!R3C2:R3C10
```

4.8.12.15 EXCELTHOUSAND

DESCRIPTION

Par défaut, IODE lit la définition des valeurs des séparateurs de milliers dans les paramètres "régionaux" (Regional Settings). Si cette valeur ne convient pas, elle peut être modifiée par la commande de rapport $ExcelThousand.

FONCTION INTERACTIVE

N'existe pas sous forme Fullscreen.

FONCTION NON INTERACTIVE

Syntaxe

```
$ExcelThousand [char]
```

La valeur de char remplace celle du séparateur de milliers définie dans les paramètres de Windows. Lorsque ce caractère est rencontré lors du transfert d'Excel vers IODE, il est ignoré.

Certaines valeurs spéciales de **char** ont une interprétation particulière :

- d, D, p ou P : le séparateur est le point
- c ou C : le séparateur est la virgule
- n, N ou pas d'argument : il n'y a pas de séparateur
- **space** ou **s** ou **Space** : le séparateur est l'espace
- tout autre valeur est prise telle quelle comme séparateur

Exemple

```
    |    B       C       D       E
--  |-------------------------------
3   | $10,000 $20,000 $30,000 $40,000
4   | $12,345 $23,456 $34,567 $45,678
```

Rapport :

```
$WSSample 1989Y1 2000Y1
$ExcelThousand C
$ExcelCurrency D
$ExcelGetVar A 1990Y1 Sheet1!R3C2:R3C10
```

4.8.13 INTERFACE DATASTREAM

La fonction suivante permet de charger des données de Datastream à partir de la base de données centrale (Londres). Pour être opérationnelle, la connexion à Internet est indispensable.

- DSImportDB : charge des données à partir du site Datastream

4.8.13.1 DSIMPORTDB

DESCRIPTION

Cette fonction assure l'importation de données de DataStream.

FONCTION INTERACTIVE

Cette fonction n'a pas d'interface écran.

FONCTION NON INTERACTIVE

Syntaxe

```
$DSImportDB nom_data_stream1, ...
```

Cette fonction va lire sur le sample courant la ou les variables définies par **nom_data_stream1,** ... et crée dans le WS courant une série dont le nom est construit en remplaçant les caractères non alpha-numériques par des underscore.

Exemple

```
$DSImportDB BEGDP..A12
```

crée **BEGDP__A12** dans le WS courant.

Si le nom_data_stream contient le caractère pourcent (%), la fonction ne s'exécute pas car % est un caractère réservé dans les rapports. Pour pallier ce problème, il faut créer une liste contenant le nom des séries et appeler DSImportDB avec la liste comme argument.

```
$DSImportDB $MYLST
```

Cette fonction n'est opérationnelle actuellement que sur les machines qui ont un accès enregistré à DataStream.

4.8.14 TRADUCTION DES FICHIERS A2M

Les fonctions suivantes permettent de traduire des fichiers A2M en différents formats:

- A2mToHtml : traduction en HTML
- A2mToRtf : traduction en Word (Rtf)
- A2mToMif : traduction en Frame (MIF)
- A2mToCsv : traduction en format CSV
- A2mToPrinter : interprète et imprime un fichier a2m

4.8.14.1 A2MTOHTML

DESCRIPTION

Traduit en HTML un fichier A2M. Si un précédent appel à la commande $PrintHtmlHelp a été effectué, le résultat sera un fichier HTML ou plusieurs fichiers permettant de créer un Help Html.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive

FONCTION NON INTERACTIVE

Syntaxe

```
$A2mToHtml filein fileout
```

où

```
filein est le nom du fichier A2M input
filout est le nom du fichier résultat
```

Dans le cas de la création d'un HtmlHelp, plusieurs fichiers résultat seront créés :

```
fileout.hhp
fileout.hhc
fileout.hhk
fileout.htm
T*.htm
```

4.8.14.2 A2MTORTF

DESCRIPTION

Traduit en RTF un fichier A2M. Si un précédent appel à la commande $PrintRtfHelp a été effectué, le résultat sera un fichier RTF permettant de créer une Aide Windows.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive

FONCTION NON INTERACTIVE

Syntaxe

```
$A2mToRtf filein fileout
```

où

```
filein est le nom du fichier A2M input
filout est le nom du fichier résultat
```

Dans le cas de la création d'une Aide Windows, plusieurs fichiers résultat seront créés :

```
fileout.rtf
fileout.hpj
fileout.cnt
```

4.8.14.3 A2MTOMIF

DESCRIPTION

Traduit en MIF un fichier A2M.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive

FONCTION NON INTERACTIVE

Syntaxe

```
$A2mToMif filein fileout
```

où

```
filein est le nom du fichier A2M input
filout est le nom du fichier résultat
```

4.8.14.4 A2MTOCSV

DESCRIPTION

Traduit en CSV un fichier A2M.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive

FONCTION NON INTERACTIVE

Syntaxe

```
$A2mToCsv filein fileout
```

où

```
filein est le nom du fichier A2M input
filout est le nom du fichier résultat
```

4.8.14.5 A2MTOPRINTER

DESCRIPTION

Traduit un fichier a2m et l'imprime sur l'imprimante courante.

FONCTION INTERACTIVE

Cette fonction n'existe pas sous forme interactive

FONCTION NON INTERACTIVE

Syntaxe

```
$A2mToPrinter file.a2m
```

4.8.15 AUTRES FONCTIONS DE RAPPORTS

- StatUnitRoot : tests de Dickey-Fuller
- WsAggrChar : définit le caractère à introduire dans le code des séries créées par WsAggr*
- WsAggrSum : effectue la somme de séries
- WsAggrMean : effectue la moyenne de séries
- WsAggrProd : effectue le produit de séries

4.8.15.1 STATUNITROOT

DESCRIPTION

Les tests de Dickey-Fuller sont disponibles au niveau des rapports et du panneau d'estimation d'équations.

Les tests sont sauvegardés dans des scalaires dont le nom est composé du préfixe `df_` et du nom de

la première série apparaissant dans la formule à tester. Par exemple, le test pour la formule `d(A0GR+A0GF)` est `df_a0gr.`

FONCTION INTERACTIVE

Dans le panneau d'estimation, la touche F3 ou le bouton "Unit Root" permettent de spécifier et de tester une ou plusieurs formules. Les résultats sont affichés dans la fenêtre.

Le seul scalaire sauvegardé est celui correspondant à la dernière expression testée.

FONCTION NON INTERACTIVE

La commande qui permet d'effectuer le calcul dans les rapports est:

```
$StatUnitRoot drift trend order expression
```

où

- **`drift`** : 0 ou 1 selon que la formule à estimer doive incorporer un terme constant (1) ou non (0)
- **`trend`** : 0 ou 1 selon que la formule à estimer doive incorporer un terme de trend (1) ou non (0)
- **`order`** : l'ordre du polynome à estimer pour obtenir les tests
- **`expression`** : forme LEC à tester

Par exemple :

```
$StatUnitRoot 1 1 3 A
```

L'équation estimée est :

```
d(A) := df_ * A[-1]+
        df_d +    /* DRIFT */
        df_t * t +  /* TREND */
        df1 * d(A[-1]) + df2*d(A[-2]) + df3*d(A[-3]) /* ORDER */
```

Seuls le test de Dickey-Fuller est sauvegardé dans un scalaire sous le nom **`df_a`** dans le cas de l'exemple.

4.8.15.2 WsAggrChar

Définit le caractère à introduire dans le code des séries créées par WsAggr*

Voir WsAggrSum

4.8.15.3 WsAggrSum

Des fonctions de rapports permettent d'effectuer des agrégations, des produits ou des sommes de séries. Les séries à traiter peuvent se trouver en WS ou dans un fichier externe.

Quatre fonctions de rapport ont été définies à cet effet:

```
$WsAggrChar [char] : définit le caractère à introduire dans le
                     code des séries créées
$WsAggrSum  pattern [filename] : somme des séries définies par pattern
$WsAggrMean pattern [filename] : moyenne des séries
$WsAggrProd pattern [filename] : produit des séries
```

Si **filename** est défini, les séries du fichier seront utilisées à la place de celles du WS.

pattern est défini comme une séquence de parties de noms placées entre crochets ou parenthèses. Chaque partie peut contenir des caractères alphanumériques ou un point d'interrogation.

Les parties de noms entre parenthèses ne sont pas agrégées. Celles entre crochets le sont.

Exemple

```
Soit un WS avec des séries par pays (BE, FR, GE), et par secteur
(S101..S999):
```

```
BES101, BES102 ... BES199
BES201, BES202 ... BES299
...
BES901, BES902 ... BES999

FRS101, FRS102 ... FRS199
FRS201, FRS202 ... FRS299
...
FRS901, FRS902 ... FRS999

...

GBS101, GBS102 ... GBS199
GBS201, GBS202 ... GBS299
...
GBS901, GBS902 ... GBS999
```

On peut créer la somme de tous les secteurs pour chaque pays par les commandes :

```
$WsAggrChar _
$WsAggrSum (??)[????]
```

Les séries créées seront :

```
BE____, FR____, ..., GB____
```

Les points d'interrogations entre () permettent de préciser les codes de séries à traiter. Les autres indiquent les parties à agréger. Dans ce cas les points d'interrogation sont remplacés par des _ (ou un autre caractère selon l'argument de **$WsAggrChar**) dans les séries résultats. Ce caractère peut être blanc. Dans l'exemple, les séries créées sont alors BE, FR et GB.

On peut également créer la somme de tous les pays par secteur ou la somme de tous les pays pour certains secteurs :

```
$WsAggrSum [??](????) : crée __S101, __S102, ...
$WsAggrSum [??](??)[??] : crée __S1__, __S1__, ...
```

On peut limiter la création à un seul ou à quelques codes :

```
$WsAggrSum (BE)[S??9] : crée BES__9
$WsAggrSum (BES)[?](?9) : crée BES_09, BES_19, BES_29, ... BES_99
```

Caractère de regroupement

La commande **$WsAggrChar** permet de spécifier le caractère à placer dans les séries générées. Ce caractère peut être blanc. Pour éviter que des séries déjà agrégées soient reprises dans une agrégation ultérieure, ces séries ne sont pas retenues dans le calcul si le caractère d'agrégat courant se trouve à une position correspondant dans pattern à un point d'interrogation entre crochets. Ainsi, la série **BE**____ ne sera pas reprise dans le calcul **(BE)[????]**. Par contre elle sera reprise dans le calcul **[??](????)**, car dans ce dernier cas elle n'intervient pas dans la somme.

Supposons que les séries **BE**____, **FR**____ et **GB**____ soient définies ainsi que **BES101**, ...

$WsAggrSum (??)[????] génère **BE**____, **FR**____, etc. Elle n'utilise donc pas les séries contenant _ après la deuxième position, comme **BE**____. En effet, si on les reprenaient, on additionnerait deux fois les mêmes séries.

$WsAggrSum [??](????) génère _____ en prenant la somme de **BE**____, **FR**____, **GB**____, ce qui est correct car les autres séries (comme **BES101**) donnent lieu à d'autres séries (__**S101**).

4.8.15.4 WsAggrMean

Effectue la moyenne de séries. Voir WsAggrSum.

4.8.15.5 WsAggrProd

Effectue le produit de séries. Voir WsAggrSum.

5. Le programme iodecmd

Ce programme permet d'exécuter un rapport IODE sans passer par l'interface de IODE.

A l'aide du langage de commandes des rapports IODE, il est ainsi possible d'automatiser la plupart des opérations effectuées normalement dans l'interface de IODE.

SYNTAXE

```
iodecmd [-nbrun n] [-alloclog filename] [-v] [-y] [-h] reportfile [arg1] ... [argn]
```

where :

- **`-nbrun n`** : n = nb of repetitions of the report execution (default 1)
- **`-allocdoc filename`** : memory debugging info stored in filename (developers only)
- **`-v`** : verbose version (all messages displayed)
- **`-y`** : answer yes to all questions asked during the report execution
- **`-h`** : display program syntax (this message)
- **`reportfile arg1 ...`** : IODE report to execute including its optional arguments

-nbrun

Cet option permet de relancer plusieurs fois un même rapport. Cela peut servir à déterminer une instabilité, un "memory leak" ou encore à effectuer des tests de performance.

-alloclog filename

Le fichier filename contiendra en fin de traitement la liste des allocations mémoire non désallouées. Cette option a été créée à des fins de debugging de IODE.

-y ou --forceyes

Cette option permet de continuer automatiquement l'exécution lorsqu'une question est posée à l'utilisateur. La réponse automatique est toujours positive.

-v ou --verbose

Cette option donne des informations sur l'état de la mémoire après chaque itération.

Exemple :

```
c:\> iodecmd -nbrun 2 -v example.foreach.rep
```

Résultat:

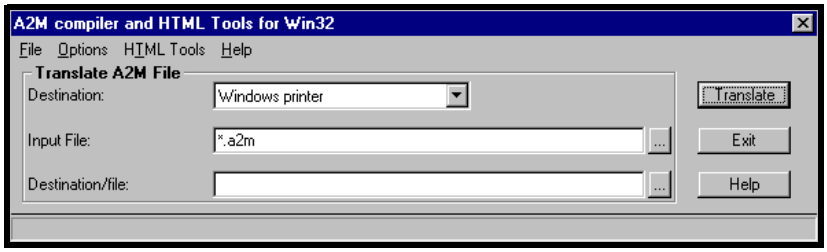
```
IODE Modeling Software 6.56 - (c) 1990-2017 Planning Office - Brussels
PIB[1990Y1] = 0 +  + 0 =
PIB[1991Y1] = 1 + 0 + 0.84147098 = 1.841471
....
PIB[1999Y1] = 9 + 2.1972246 + 0.41211849 = 11.609343
PIB[2000Y1] = 10 + 2.3025851 + -0.54402111 = 11.758564
*****Run 1/2 *****
rc = 0 -- total residual allocation : 4010740
PIB[1990Y1] = 0 +  + 0 =
PIB[1991Y1] = 1 + 0 + 0.84147098 = 1.841471
...
PIB[1999Y1] = 9 + 2.1972246 + 0.41211849 = 11.609343
PIB[2000Y1] = 10 + 2.3025851 + -0.54402111 = 11.758564
*****Run 2/2 *****
rc = 0 -- total residual allocation : 6124268
After last run : 6124268 bytes allocated
```

-h our --help

Affiche la syntaxe du programme.

6. Le programme A2M

FIGURE 83



Le programme `a2m` convertit des textes ASCII comprenant ou non des commandes A2M en divers autres formats et permet de lancer l'impression de fichiers sur divers types d'imprimantes selon des procédures définies par l'utilisateur.

- Introduction
- L'interface utilisateur : le panneau principal
- Le menu File : quitter l'application
- Le menu Options : les options de conversion
- Le menu HTML Tools : opérations sur des fichiers HTML
- Le menu Help : l'aide en ligne

6.1 INTRODUCTION

Les commandes A2M sont des commandes de formatage de paragraphes ou de textes, elle permettent également d'inclure et de formater des graphiques, des tableaux, de gérer les couleurs du document, etc...

Un fichier en format A2M peut être traduit en différents formats :

- GDI, format d'impression directe sous Windows
- RTF, interprétable par Word, WordPerfect, etc
- MIF, langage structuré des documents Frame Maker
- RTF-HLP, source d'un fichier d'aide de Windows 95
- HTML, format des documents Internet (Web)
- CSV, comma separated value

Selon la destination de l'impression, des tags ou commandes seront soit interprétés différemment, soit inopérants. Ainsi par exemple, les images ne peuvent être incorporées dans des fichiers ASCII ou les sujets d'aide n'ont de signification que dans le cadre des fichiers d'aide.

6.2 L'INTERFACE UTILISATEUR

L'interface utilisateur comprend, outre les éléments classiques d'une fenêtre Windows:

- le menu déroulant permetant notamment de définir les options de conversion. Les points du menu sont décrite ci-après.
- le bouton "Translate" qui lance la conversion du document A2M
- le bouton "Exit" qui permet de quitter l'application
- le bouton "Help" qui donne accès à l'aide en ligne
- le menu déroulant "Destination" qui permet de définir le type de conversion à opérer sur le document A2M. Un clic dans la flèche vers le bas de ce champ donne la liste des destinations possibles.
- le champ "Input file" qui doit contenir le nom du fichier à convertir. Le bouton "..." à droite du champ permet de parcourir le système de fichier afin de localiser ce fichier si nécessaire.
- le champ "Destination file" qui doit contenir (sauf dans le cas d'une impression directe sur une imprimante windows) le nom du fichier de destination, résultat de la conversion. Le bouton "..." à droite du champ permet de parcourir le système de fichier afin de préciser la localisation de ce fichier.

Les touches de fonction F10, Esc et F1 permettent respectivement de lancer la conversion, de quitter l'application ou d'appeler l'aide en ligne de l'application. D'autres touches de fonction sont éventuellement actives dans les autres pages, leur liste apparait en commentaire en bas de fenêtre.

Pendant la conversion, un message en bas de fenêtre informe l'utilisateur du déroulement des opérations (nombre d'objet lus et interprétés, nombre de pages imprimées).

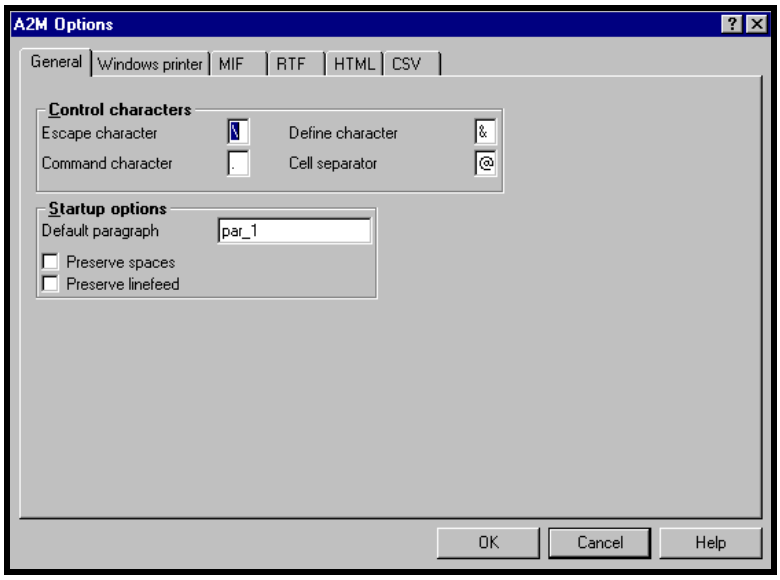
Enfin, le programme peut être lancé en cliquant sur un fichier A2M identifié par l'icone appropriée à l'aide de l'exploreur de Windows. Dans ce cas, le nom du fichier sélectionné est automatiquement importé dans le champ "Input File" de l'application.

6.3 LE MENU FILE

Le menu File permet de quitter l'application. On peut quitter l'application également avec le bouton "Exit" du panneau principal ou en tapant simultanément les touches ALT et X en n'importe quel endroit du programme.

6.4 LE MENU OPTIONS

FIGURE 84



Le menu option donne accès aux paramètres de définition du fichier A2M d'origine (panneau "General") ainsi qu'aux paramètres pouvant modifer le résultat final de la conversion en différents formats (imprimante, MIF, RTF, HTML, ...).

Les options standard sont correctes dans la grandes majorité des cas, et l'utilisateur n'a pas en principe à les modifier.

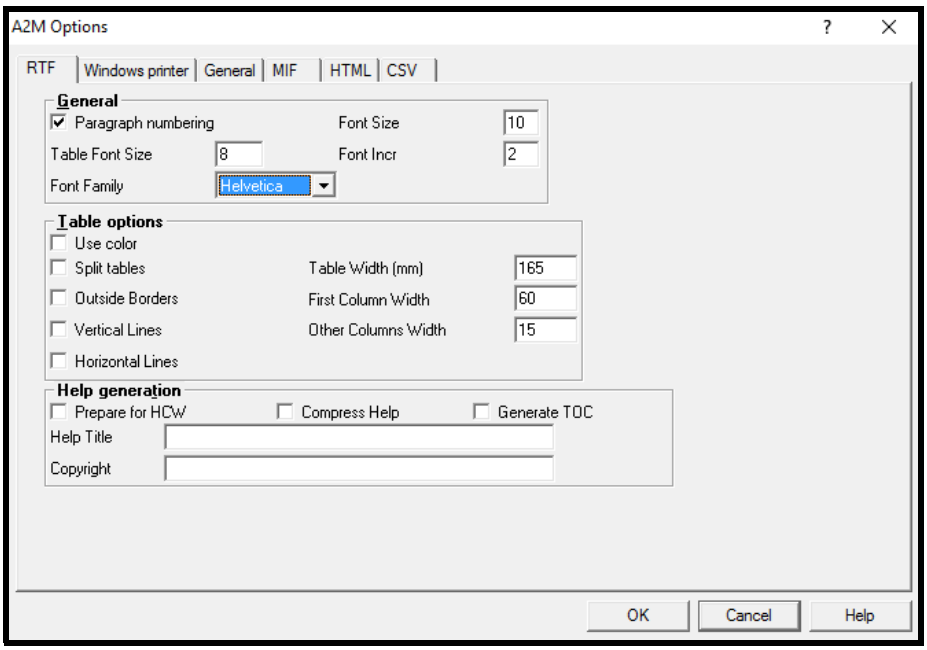
Les modifications sont conservées lors des utilisations ultérieures du programme.

Les paramètres suivants sont accessibles:

- la fiche "General" : structure du fichier A2M d'origine
- la fiche "Windows printer": particularités d'une impression
- la fiche "MIF" : output dans un fichier MIF
- la fiche "RTF" : output dans un fichier RTF
- la fiche "HTML" : output dans un fichier HTML
- la fiche "CSV" : output dans un fichier CSV

6.4.1 LA FICHE "GENERAL"

FIGURE 85



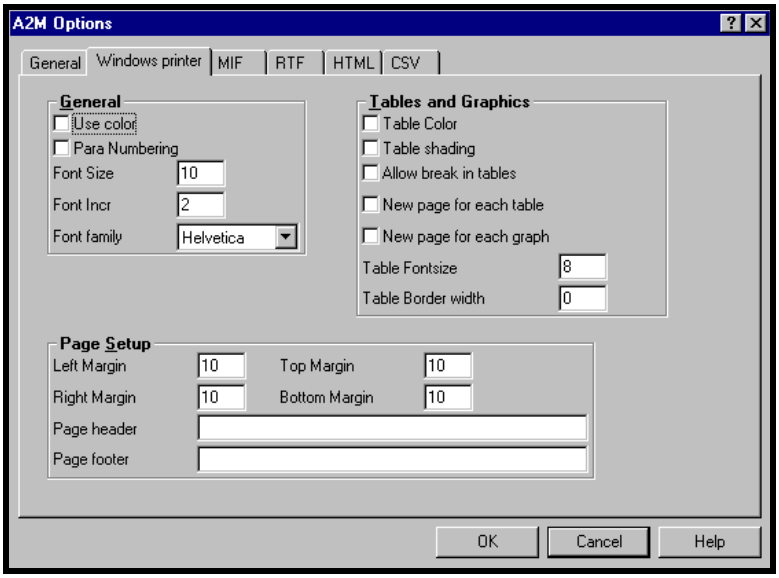
Nous renvoyons au manuel de référence pour les détails de la syntaxe de A2M.

- Le caractère d'échappement (Escape character) permet de définir le caractère qui débute une séquence d'attribut typographiques, d'insertion d'objet ou d'hyperlien dans la suite du texte. Par défaut, ce caractère est le BACKSLASH (\). La commande ".esc" dans le fichier A2M permet de modifier cette option.
- Le caractère de commande est celui qui annonce, en début de ligne, une commande de mise en page du document.
- Le caractère de définition est le caractère de début d'une macro (define). Ce caractère est interprété uniquement si le mot qui le suit est une macro. Si ce n'est pas le cas, il est imprimé comme tel.

6.4.2 LA FICHE "WINDOWS PRINTER"

La fiche permet de configurer certaines options de filtrages appliquées sur le fichier A2M avant l'envoi sur le pilote d'impression de Windows.

FIGURE 86



Pour le texte, l'utilisateur peut choisir de conserver les attributs de couleur, la numérotation des paragraphes, la police et la taille du caractère et l'incrément utilisé par la commande d'attribut typographique S.

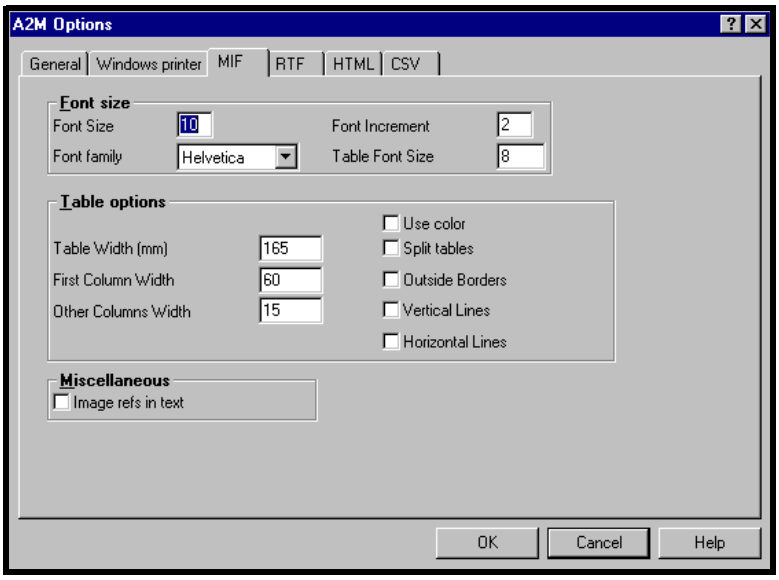
Pour la page, l'utilisateur peut définir les marges et les références de haut et de bas de page.

Pour les tableaux et les graphiques, l'utilisateur peut choisir diverses options de découpage et de mise en page.

6.4.3 LA FICHE "MIF"

La fiche permet de configurer certaines options de filtrages appliquées sur le fichier A2M lors de la transformation en fichier MIF (Maker Interchange Format).

FIGURE 87



L'utilisateur peut choisir la police et la taille des caractères à utiliser dans le texte et les tableaux.

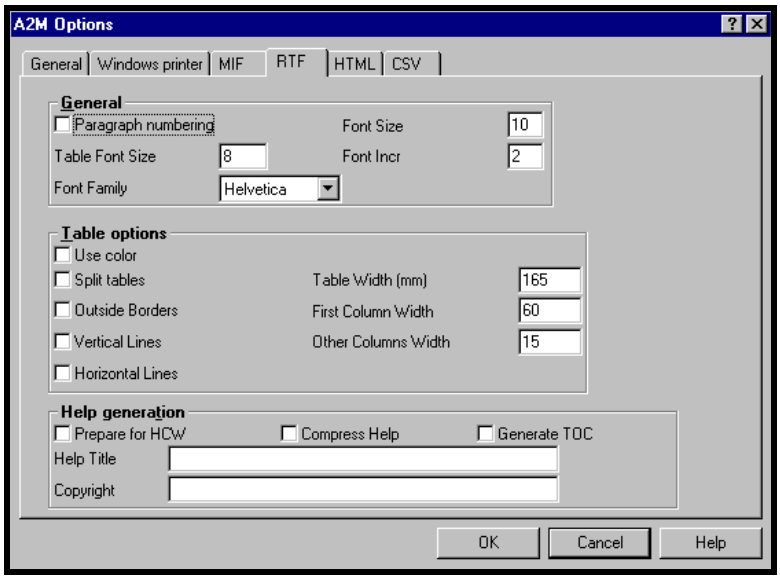
Pour les tableaux l'utilisateur peut choisir diverses options de découpage et de formatage (taille des colonnes et quadrillage).

On peut enfin inclure ou non l'ancrage d'images (commande .a dans le fichier A2M).

6.4.4 LA FICHE "RTF"

La fiche permet de configurer certaines options de filtrages appliquées sur le fichier A2M lors de la transformation en fichier RTF (Rich Text Format).

FIGURE 88



Pour le texte, l'utilisateur peut choisir de conserver la numérotation des paragraphes, la police et la taille du caractère et l'incrément utilisé par la commande d'attribut typographique S.

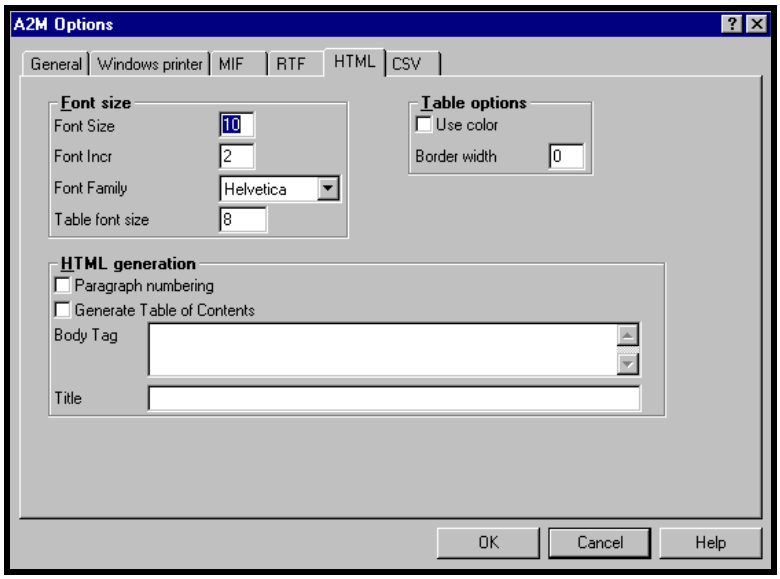
Pour les tableaux, l'utilisateur peut choisir diverses options de découpage et de mise en page (taille des colonnes et quadrillage).

Enfin, certaines options peuvent être définies en vue de la préparation d'un fichier prêt à être compilé par compilateur d'aide de Windows (Help Compiler for Windows).

6.4.5 LA FICHE "HTML"

La fiche permet de configurer certaines options de filtrages appliquées sur le fichier A2M lors de la transformation en fichier HTML (HyperText Markup Language, version 2.0 standardisée par la note RFC1866).

FIGURE 89



L'utilisateur peut choisir la police et la taille des caractères à utiliser dans le texte et les tableaux.

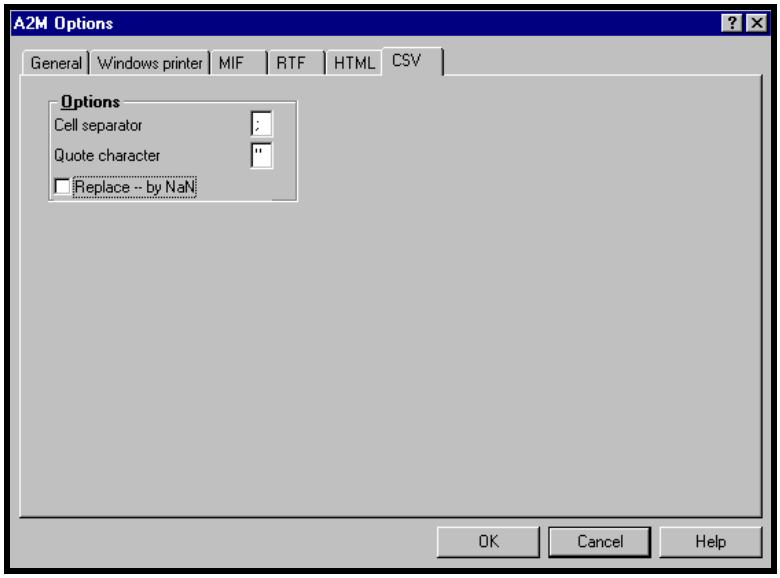
Pour les tableaux l'utilisateur peut choisir un filtrage couleur et l'épaisseur de l'encadrement.

Enfin, l'utilisateur peut spécifier diverses options propres au langage HTML comme un titre ou des options paticulières à la commande BODY.

6.4.6 LA FICHE "CSV"

La fiche CSV permet de paramètrer la conversion de tableaux en format A2M en format CSV (comma saparated value).

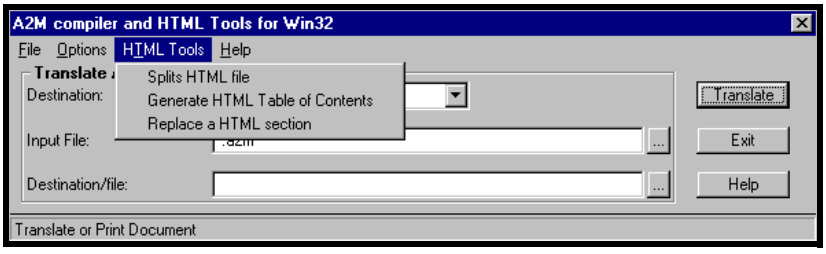
FIGURE 90



On peut définir le séparateur de cellules, le caractère "quote" et le remplacement éventuel du contenu "Not Available".

6.5 LE MENU HTML TOOLS

FIGURE 91



Le menu HTML Tools donne accès à une série d'outils permettant de modifier un fichier HTML après sa création:

- Split HTML file : découpe un fichier HTML
- Generate Table of Contents: génère une table des matières
- Replace a HTML section : remplace une section de fichier

Les références internes dans le fichier et entre éléments des nouveaux fichiers sont conhérentes.

6.5.1 Split HTML file

Ce programme découpe un fichier HTML en sous fichiers. Les découpes ont lieu sur les tags Hn qui indiquent les niveaux de chapitres dans les fichiers HTML.

A partir d'un fichier unique, éventuellement de grande taille, un nombre quelconque de fichiers sont générés, selon le niveau de découpe demandé.

Le titre de chaque sous-fichier est repris dans le fichier de niveau supérieur avec le lien vers le sous-fichier.

Les noms des fichiers générés sont constitués en ajoutant un numéro d'ordre à la racine du fichier output.

La page permet de spécifier les paramètres utiles à la découpe:

- Le nom du fichier source et
- le niveau n de la découpe.
- La racine du nom du fichier output
- Le titre, l'image du background et l'icone de retour à la page précedente.

6.5.2 Generate Table of Contents

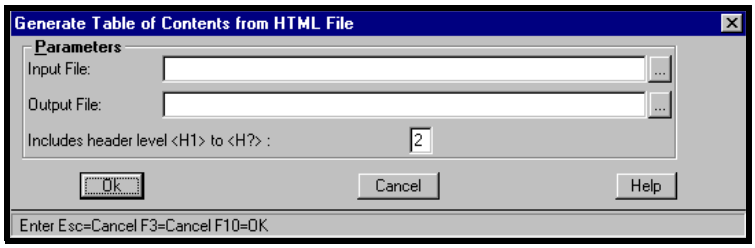
Ce programme analyse un fichier HTML et ses sous fichiers. Il génére simultanément une table des matières sur base des tags <Hn> (qui indiquent les niveaux de chapitres dans les fichiers HTML).

Le titre de chaque chapitre est repris avec le lien vers le sous-fichier et si un tag NAME est présent, vers la position dans ce sous-fichier.

Le fichier résultat n'est pas un fichier HTML complet dans la mesure où il est par la suite destinné à être intégré dans un fichier plus vaste.

Le fichier débute par un tag spécial !STARTTOC et se termine par un autre: !ENDTOC. Ces deux tags pourront par la suite servir d'indication au sous-programme de substitution (voir "Replace a HTML section").

FIGURE 92



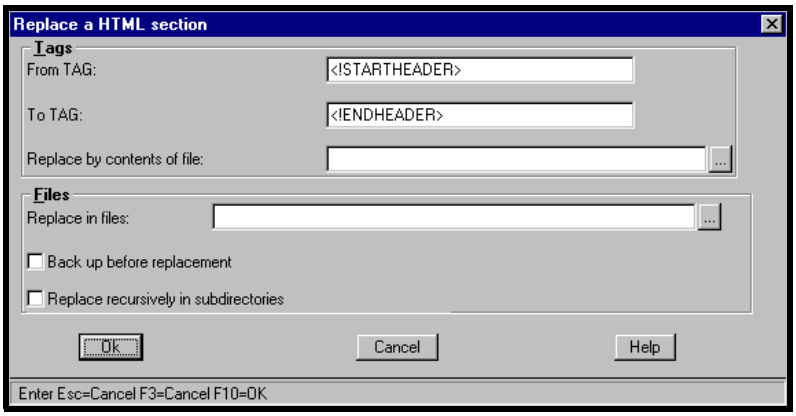
La page permet de spécifier le fichier source, le fichier résultat et le niveau de chapitre à inclure dans la table des matières.

6.5.3 REPLACE A HTML SECTION

Ce programme recherche deux tags spéciaux (from TAG et to TAG) dans un fichier et remplace tout ce qui se trouve entre ces deux tags par le contenu d'un autre fichier.

Cette fonction est particulièrement utile dans le cas des fichiers HTML: elle permet par exemple de remplacer en une seule opération les barres de navigation dans tous les fichiers d'un site, ou d'inclure une table des matières (!STARTTOC et !ENDTOC).

FIGURE 93



La page permet de définir les tags de début et de fin de substitution (par défaut !STARTHEADER et !STOPHEADER), le fichier contenant le texte à substituer et le fichier où doit s'opérer la substitution. Il est possible (et recommandé) de sauvegarder le texte d'origine (backup) et de parcourir récurcivement un arbre de répertoires.

6.6 LE MENU HELP

Ce point de la barre d'action permet d'appeler, au même titre que la touche F1 ou le bouton Help, une aide conceptuelle en ligne avec table des matières et index.

7. Interfaces avec Excel, APL, ODBC

IODE peut dialoguer avec d'autres logiciels à travers différents interfaces :

- Interface tussen Relationele DB's en Iode : fonctions de recherche dans les bases de données relationnelles
- Interface entre Excel et Iode : le spreadsheet iode.xls
- Interface entre l'Apl et Iode : le WS Apl iode.w3

7.1 Interface tussen Relationele DB's en Iode

IODE kan gegevens uitwisselen met Databanken bereikbaar via ODBC. ODBC is een standaard voorgesteld door Microsoft waarin de functies worden geïmplementeerd voor het ondervragen van een Relationele Databank.

Elke databank-constructeur (DB2, Oracle, SQL-server, Access, SAS, ...) beschikt over een ODBC-driver die de gebruiker toelaat om zo een databank ontwikkeld in zijn omgeving vanuit een ander programma te raadplegen en updaten.

Om die databanken te gaan ondervragen wordt de SQL (Structered Query Language) gebruikt. Zij is min of meer gestandardiseerd (ANSI), elke constructeur heeft wel zijn eigen dialectversie uitgebracht door er bijkomende mogelijkheden aan toe te voegen. Het hangt dus van uw databanksysteem af welk SQL-dialect u moet gebruiken.

IODE beschikt nu over een aantal rapportfuncties die u toe laten gegevens opgeslaan in een relationele databank te gaan recupereren. Er wordt verondersteld dat u over de nodige SQL-kennis beschikt om de data te formateren zodat ze in IODE bruikbaar wordt.

IODE is slechts een SQL-client en geen SQL-server. Het is momenteel onmogelijk in andere omgevingen via SQL-commando's IODE data te gaan ondervragen.

Hier volgen de rapport-commando's waarover u kan beschikken.

- @SqlOpen
- @SqlQuery
- @SqlNext
- @SqlField
- @SqlRecord
- @SqlClose

Als u in een IODE-rapport SQL wil gaan gebruiken moet u in uw rapport een vaste structuur gebruiken. Eerst moet de ODBC-databank toegankelijk worden gemaakt (SqlInit), daarna moet de SQL-instructie (SqlQuery) worden doorgegeven en de resultaten doorlopen (SqlNext) en behandeld (SqlField of SqlRecord). Vergeet op het einde niet de ODBC-sessie af te sluiten (SqlClose).

STRUCTUUR VAN EEN SQL-RAPPORT

Als u in een IODE-rapport SQL wil gaan gebruiken moet u in uw rapport een vaste structuur gebruiken. Eerst moet de ODBC-databank toegankelijk worden gemaakt (SqlInit), daarna moet de SQL-instructie (SqlQuery) worden doorgegeven en de resultaten doorlopen (SqlNext) en behandeld (SqlField of SqlRecord). Vergeet op het einde niet de ODBC-sessie af te sluiten (SqlClose).

- SqlOpen() : opent de ODBC-sessie
- SqlQuery("SQL-query") : creëert een "Dynaset" met de resultaten
- SqlNext() : verplaatst zich in de "Dynaset"
- SqlField(n), ... of SqlRecord() : van de "Dynaset" om de gegevens te recupereren
- SqlClose() : sluit de ODBC-sessie af

Exemple

```
$goto continue, @SqlOpen(RSZ)
$Show  ODBC-Databank not opened
$Return

$label continue
$Show  ODBC-Databank opened
++++++++++  SAMPLE  ++++++++++++

@SqlQuery("SELECT DISTINCT TRIM FROM RSZ_bruto;")
@SqlNext()
$Define BFLD @SqlField(0)

$label again_date
$Show Skipping @SqlField(0)
$goto again_date, @SqlNext()
$Define EFLD @SqlField(0)

$Define BEGIN @replace(%BFLD%,/,Q)
$Define END   @replace(%EFLD%,/,Q)

$WsSample %BEGIN% %END%

++++++++++  RECORDS ++++++++++++

$Define PIVOT RETOMA_DBF
$Define PREP  RT

    $Msg  NBCOLS @SqlQuery("TRANSFORM Sum(RSZ_bruto.%PIVOT%) AS AGGR SELECT
RSZ_bruto.CODRED, NaceTbl.NACE FROM NaceTbl INNER JOIN RSZ_bruto ON NaceTbl.FICTIF
=RSZ_bruto.FICTIF GROUP BY RSZ_bruto.CODRED, NaceTbl.NACE PIVOT RSZ_bruto.TRIM;")
    $Show  @SqlNext()
    $label again
    $DataUpdateVar %PREP%@SqlField(0)@SqlField(1) %BEGIN% @SqlRecord(2, 8)
    $goto again, @SqlNext()

    $Show  @SqlClose()
```

7.2 INTERFACE TUSSEN EXCEL EN IODE

IODE kan gegevens uitwisselen met andere Windows Programma's. Dit gebeurt via het DDE (Dy-

namic Data Exchange) protocol. DDE laat twee programma's toe met elkaar te "spreken" en zo gegevens uit te wisselen. DDE automatiseert in feite het "cut" en "paste" proces tussen programma's.

Het programma dat de gegevens verstrekt is de DDE-server. De applicatie die de gegevens opvraagt is de DDE-client. Beide programma's moeten tegelijkertijd draaien en kunnen alleen de gegevens uitwisselen die op dat ogenblik geladen zijn. IODE is een DDE-server (IODE-data naar Excel of APL) en een DDE-client (Excel-data of APL-vectoren naar IODE).

Data opgeslaan in IODE is gekend via het type (VAR, IDT, ...) en de naam van het gewenste object. Data opgeslaan in Excel wordt gereferenceerd via een "range". Een range is ofwel een "named range, ofwel een samenstelling van de naam van het werkblad en de namen van de linkerboven cel en de rechteronder cel. Zo duidt "Sheet1!R1C1:R2C4" de eerste vier cellen van de eerste rij en de tweede rij aan. Als we nu een variable A vanuit IODE naar Excel willen kopiëren dan moeten wij in Excel een range gebruiken die groot genoeg is om de gegevens op te slaan.

Niet alle data kan worden uitgewisseld naar Excel. Wij beperken ons tot variabelen, commentaren, lijsten en berekende tabellen. Identiteiten, scalairen en vergelijkingen hebben geen vergelijkbare tegenhanger in Excel. In de andere richting kunnen alleen IODE-variabelen worden geüpdatet met gegevens uit Excel.

Gebruikt u IODE als DDE-server, laat dan slechts een instantie van het programma lopen. Als er meerdere gelijktijdige sessies zijn, kan men niet voorspellen welke sessie zal antwoorden op de gestelde vragen.

IODE gebruikt het decimaal punt ipv de komma. Om gebruik te maken van de DDE-mogelijkheden moet uw Windows-systeem ook het decimaal punt gebruiken. Pas dit in uw "Control Panel" aan. Indien niet krijgt u bizarre resultaten.

Dit hoofdstuk beschrijft in detail de manier waarop u data tussen IODE en Excel kan uitwisselen:

- Vanuit IODE-rapporten
- Vanuit Excel
- Technisch addendum DDE-server functies

7.2.1 Vanuit IODE-rapporten

Een IODE-rapport kan niet alleen gegevens uitwisselen met Excel, u kan Excel ook enkele van zijn kunstjes laten vertonen. U kan Excel een Workbook van disk laten laden, enkele aanpassingen doen en zo live uw voorgedefiniëerde grafieken zien veranderen, uw resultaten laten afdrukken of wegschrijven voor later gebruik.

De Excel-commando's bruikbaar vanuit rapporten zijn de volgende

- Gegevens uitwisselen
  - ExcelGetVar
  - ExcelGetScl
  - ExcelGetCmt
  - ExcelSetXxx

- ExcelWrite
- ExcelLang
- ExcelDecimal
- ExcelThousand
- ExcelCurrency

• Excel commando's uitvoeren

- ExcelOpen
- ExcelNew
- ExcelClose
- ExcelPrint
- ExcelSave
- ExcelSaveas

• Demo

7.2.1.1 DEMO: BESTUUR EXCEL VANUIT EEN IODE RAPPORT

EXEMPLE

```
Load the variable file
$WsLoadVar c:\usr\mt\anl\deval\dip\nov

Load the table file
$WsLoadTbl c:\usr\mt\anl\deval\dip\dip

Open in Excel the spreadsheet with the defined graphics
$ExcelOpen c:\usr\mt\anl\deval\dip\dip.xls

Place the table(s) with calculated series, base version
$ExcelSetTbl DIP1 90:11 Growth!R2C12
$ExcelSetTbl DIP2 90:11 Unemployment!R2C12
$ExcelSetTbl DIP3 90:11 CurrentAccount!R2C12
$ExcelSetTbl DIP4 90:11 GovtAccount!R2C12
$ExcelSetTbl DIP5 90:11 Inflation!R2C12
$ExcelSetTbl DIP6 90:11 RealDisposableIncome!R2C12

Place the table(s) with calculated series, simulated version
$ExcelSetTbl DIP1 90:11 Growth!R9C12
$ExcelSetTbl DIP2 90:11 Unemployment!R9C12
$ExcelSetTbl DIP3 90:11 CurrentAccount!R9C12
$ExcelSetTbl DIP4 90:11 GovtAccount!R9C12
$ExcelSetTbl DIP5 90:11 Inflation!R9C12
$ExcelSetTbl DIP6 90:11 RealDisposableIncome!R9C12

$Ask simulation, Start Simulation?
$return

$Label simulation

Change the hypothesis and see what heapens

$WsLoadVar oeso2.var
$WsLoadIdt NOTE
$WsSample 1970Y1 2000Y1
$WsExtrapolate 0 1998Y1 2000Y1
```

```
Change the Exchange Rate BF/$

$DataUpdateIdt BELVAR7 BELVAR7*1.15

Change the Effective exchange rate

$DataUpdateIdt BELVAR6 BELVAR6*.9
$IdtExecute 1993Y1 2000Y1 BELVAR7 BELVAR6

$IdtExecute 1991Y1 2000Y1 PWXDAN PWXDEU PWXESP PWXFRA PWXGRE PWXIRL
$IdtExecute 1991Y1 2000Y1 QWXAB QWXS PWXAB PWXS PWMAB EX EER RLGY RSGY

$WsSaveVar oeso2test.var

$WsClearIdt Note
$WsLoadEqs nov
$WsLoadVar nov
$WsLoadScl note
$WsCopyVar oeso2test QWXAB PWXAB PWMAB PWMS PWXS QWXS EX EER RLGY

Initiliase the first year and simulate a year at a year
Your predefined graphics in Excel change at the same year

$SetTime 1992Y1
$Label again
$IncrTime 1

$ModelSimulateParms .001 0.5 100 C 4 N N 2
$modelsimulate {t@T} {t@T}


Visualize the changes in Excel
$ExcelSetTbl DIP1 90:11 Growth!R9C12
$ExcelSetTbl DIP2 90:11 Unemployment!R9C12
$ExcelSetTbl DIP3 90:11 CurrentAccount!R9C12
$ExcelSetTbl DIP4 90:11 GovtAccount!R9C12
$ExcelSetTbl DIP5 90:11 Inflation!R9C12
$ExcelSetTbl DIP6 90:11 RealDisposableIncome!R9C12

$goto continue {t > 1999Y1}
$WsCopyVar nov {t+1@T} 2000Y1 QBBP PC VBBP FLG VX VM YDH U NAT
$label continue

Loop until 2000 reached
$goto again {t < 2000Y1}
```

7.2.2 VANUIT EXCEL

U kan ook vanuit Excel IODE besturen. Niet alleen data is uitwisselbaar, ook IODE-commando's en rapporten kunnen vanuit Excel worden uitgevoerd.

Er zijn twee manieren om vanuit Excel IODE-data en commando's te gebruiken. Eerst is er de DDE-interface die u in gelijk welke cel van een Excel-werkblad invult. Daar moet u zich wel strikt houden aan de in IODE geprogrammeerde DDE-calls. De syntax is niet zo eenvoudig en is beschreven in het technisch addendum DDE-server functies .

Een tweede manier is gebruik te maken van het Iode.xls werkblad dat bij uw software wordt geleverd. Daar is alles voorgeprogrammeerd en is zelfs door een leek te gebruiken.

Gebruikt u IODE als DDE-server, laat dan slechts een instantie van het programma lopen. Als er meerdere gelijktijdige sessies zijn, kan men niet voorspellen welke sessie zal antwoorden op de gestelde vragen.

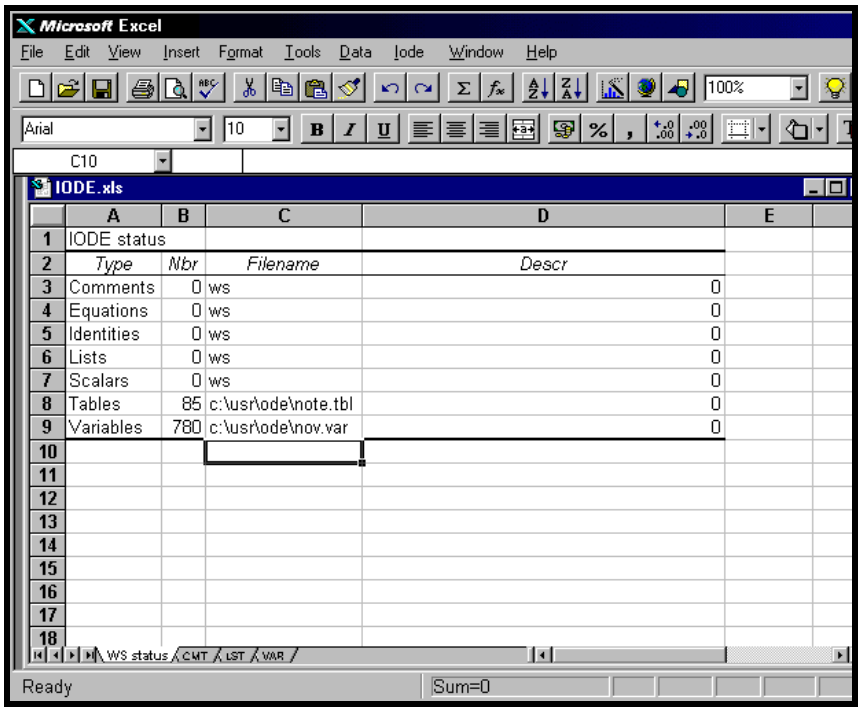
IODE gebruikt het decimaal punt ipv de komma. Om gebruik te maken van de DDE-mogelijkheden moet uw Windows-systeem ook het decimaal punt gebruiken. Pas dit in uw "Control Panel" aan. Indien niet krijgt u bizarre resultaten.

7.2.2.1 IODE.XLS WERKBLAD

Als u bij de IODE-installatie ook Excel interface heeft gekozen vindt u bij het "IODE 5" menu een submenu "Excel-IODE interface". Infeite gaat het hier om een werkboek met enkele Visual Basic procedures. Zij maken gebruik van de verder beschreven DDE-functies in IODE. Alleen zijn zij verpakt in gemakkelijk te gebruiken interface.

Als u in Excel het werkblad "Iode.xls" opent, wordt een verbinding gelegd met een openstaande IODE-sessie. In het "IODE Status" werkblad krijgt u een overzicht van de workspaces die op dat ogenblik geladen zijn.

FIGURE 94



Daarnaast vindt u de werbladen "VAR", "CMT", "LST", waarin u de IODE-data zal visualiseren.

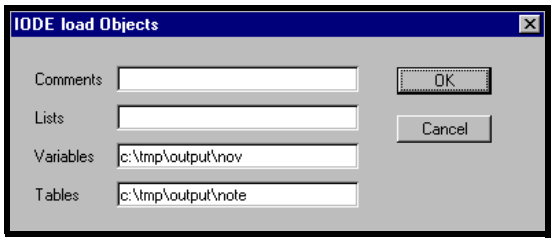
U ziet dat de MenuBar van Excel met een "Iode" menu is uitgebreid. Hier vindt u alle IODE-mogelijkheden waarover u kan beschikken:

- Execute IODE commands
  - Load IODE objects
  - Execute IODE report
  - Execute report commands
- Get IODE Objects
  - Comments

- Lists
- Tables
- Variables

1. LOAD IODE OBJECTS

FIGURE 95



DESCRIPTION

Dit menu laadt IODE-data files, voor gebruik in Excel in.

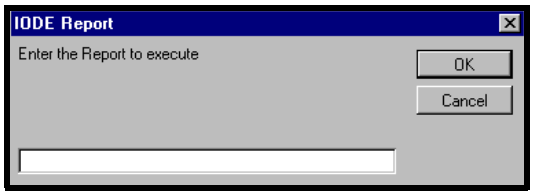
Excel opent een dialoog-venster waarin u de filename invult met de data waarin u in deze Excel-sessie wil over beschikken.

U vult alleen de filename in van de gegevens in waarover u wil beschikken. Excel vraagt IODE de files in te laden. Als ui de prompt krijgt voor

Geef het volledig pad, en vergeet de object extensie (bijv. var) niet.

2. EXECUTE IODE REPORT

FIGURE 96



DESCRIPTION

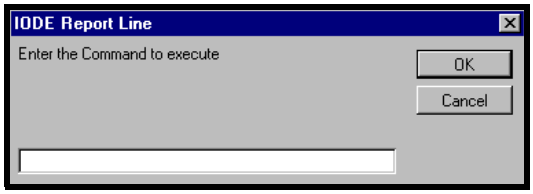
Dit menu laat u een IODE-rapport uitvoeren

Excel opent een dialoog-venster waarin u de filename van het rapport invult dat u wil uitvoeren. Het gaat hier over gelijk wel geldig IODE-rapport. U kan dus ook alle $ExcelXxx commando's van IODE gebruiken.

Geef het volledig pad, en vergeet de extensie (rep) niet.

3. Execute report commands

FIGURE 97



Description

Dit menu laat u een IODE-commando uitvoeren

Excel opent een dialoog-venster waarin u de filename van het IODE-commando invult dat u wil uitvoeren. Het gaat hier over gelijk wel geldig IODE-commando. U kan dus ook alle $ExcelXxx commando's van IODE gebruiken.

U kan alleen die objecten bekijken die op dat ogenblijk in IODE geladen zijn.

4. Comments

Description

Dit menu laat u een IODE-commentaren zien in uw "CMT"-werkblad

Excel opent een dialoog-venster waarin u de naam van de IODE-commentaar kan invullen. In uw "CMT-werkblad verschijnt de naam en de inhoud van de gevraagde commentaar. Wil u meerdere zien tegelijkertijd; vul de namen in afgewisseld met een blanco of komma.

Vult u niets in en drukt u op de "OK"-toets dan wordt de gehele CMT-databank vanuit IODE naar Excel geschreven.

U kan alleen die objecten bekijken die op dat ogenblijk in IODE geladen zijn.

5. Lists

Description

Dit menu laat u een IODE-lijsten zien in uw "LST"-werkblad

Excel opent een dialoog-venster waarin u de naam van de IODE-list kan invullen. In uw "LST"-werkblad verschijnt de naam en de inhoud van uw gevraagde lijst.

Wil u meerdere zien tegelijkertijd; vul de namen in afgewisseld met een blanco of komma.

Vult u niets in en drukt u op de "OK"-toets dan wordt de gehele LST-databank vanuit IODE naar Excel geschreven.

U kan alleen die objecten bekijken die op dat ogenblijk in IODE geladen zijn.

6. TABLES

DESCRIPTION

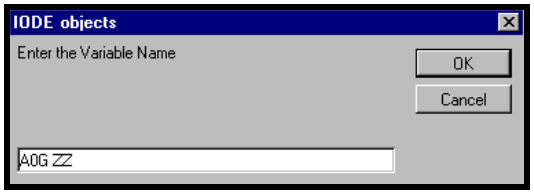
Dit menu laat u een IODE-tabellen zien. Voor elke tabel wordt een werkblad "Tbl tabelnaam" gecreëerd, waarin u de tabel ziet berekend over de gehele sample van de geladen VAR-databank

Excel opent een dialoog-venster waarin u de naam van de IODE-tabel kan invullen. In uw "Tbl tabelnaam"-werkblad verschijnt de naam en de inhoud van uw gevraagde tabel.

U kan alleen die tabelen bekijken die op dat ogenblijk in IODE geladen zijn en waarvoor alle nodige variabelen aanwezig zijn voor het berekenen van de tabel.

7. VARIABLES

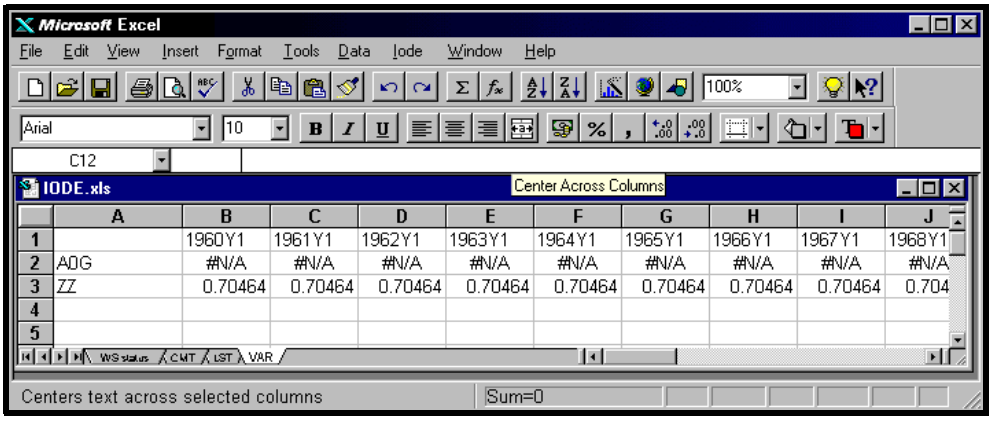
FIGURE 98



DESCRIPTION

Dit menu laat u een IODE-variabelen zien.

FIGURE 99



Excel opent een dialoog-venster waarin u de naam van de IODE-variabele kan invullen. In uw "VAR"-werkblad verschijnt de naam en de inhoud van uw gevraagde variabele over de gehele sample van de gealden var-file. Op de eerste lijn vindt u aanduiding van de sample.

Wil u meerdere zien tegelijkertijd; vul de namen in afgewisseld met een blanco of komma.

Vult u niets in en drukt u op de "OK"-toets dan wordt de gehele VAR-databank vanuit IODE naar Excel geschreven.

U kan alleen die objecten bekijken die op dat ogenblijk in IODE geladen zijn.

7.2.3 FONCTIONS DU SERVEUR DDE

Les données techniques qui suivent permettent d'exploiter le serveur DDE dans le cadre d'autres programmes.

Service : IODE

Topics : WS (Request et Poke), REP (Poke), PLAY (Poke), XVAR, XCMT CMT, EQS, IDT, LST, SCL, TBL, VAR (Request et Poke)

- Request
- Poke

7.2.3.1 REQUEST

ITEMS DE WS

- SAMPLE = sample du WS IODE courant
- List
  - CLIST = liste des commentaires du WS IODE courant
  - ELIST = liste des équations du WS IODE courant
  - ILIST = liste des identités du WS IODE courant
  - LLIST = liste des listes du WS IODE courant
  - SLIST = liste des scalaires du WS IODE courant
  - TLIST = liste des tableaux du WS IODE courant
  - VLIST = liste des variables du WS IODE courant
- Name
  - CNAME = nom du fichier des commentaires de IODE
  - ENAME = nom du fichier des équations de IODE
  - INAME = nom du fichier des identités de IODE
  - LNAME = nom du fichier des listes de IODE
  - SNAME = nom du fichier des scalaires de IODE
  - TNAME = nom du fichier des tableaux de IODE
  - VNAME = nom du fichier des variables de IODE
  - CDESCR = description du fichier des commentaires de IODE
  - EDESCR = description du fichier des équations de IODE
  - IDESCR = description du fichier des identités de IODE

- LDESCR = description du fichier des listes de IODE
- SDESCR = description du fichier des scalaires de IODE
- TDESCR = description du fichier des tableaux de IODE
- VDESCR = description du fichier des variables de IODE

• Nb

- CNB = nombre de commentaires dans IODE
- ENB = nombre de équations dans IODE
- INB = nombre de identités dans IODE
- LNB = nombre de listes dans IODE
- SNB = nombre de scalaires dans IODE
- TNB = nombre de tableaux dans IODE
- VNB = nombre de variables dans IODE

ITEMS DE CMT, EQS, IDT, LST, SCL, TBL, VAR

L'item est le nom de l'objet. La valeur retournée est la définition de l'objet.

ITEMS DE XVAR

L'item est composé de trois éléments séparés par des ! :

- nom de la ou des variables séparés par des virgules ou des blancs. t représente une ligne de périodes
- nom du sheet où placer le résultat
- range de la première cellule dans ce sheet

Cette requête génère un Poke vers excel avec la définition des variables et leur code.

Si aucun nom de série n'est donné, une série avec les valeurs des périodes précède. Sinon, on peut placer les périodes en indiquant la série t (minuscule).

S'il n'y a pas de sheet, Sheet1 est choisi comme destination S'il n'y a pas de range, R1C1 est choisi comme destination

Exemple

```
IODE|XVAR!' ' : toutes les séries dans le sheet1, en R1C1
                avec les périodes
IODE|XVAR!'t,QC0,QAFF!Sheet2!R1C1' : séries QC0 et QAFF
                dans le Sheet2 en position R1C1, avec
                les périodes en première ligne
IODE|XVAR!'QC0!Sheet2!R1C1' : série QC0 sans les périodes
```

ITEMS DE XCMT

L'item est composé de trois éléments séparés par des ! :

- nom de la ou des commentaires séparés par des virgules ou des blancs
- nom du sheet où placer le résultat
- range de la première cellule dans ce sheet

Cette requête génère un Poke vers excel avec la valeur des commentaires et leurs codes.

S'il n'y a pas de sheet, Sheet1 est choisi comme destination S'il n'y a pas de range, R1C1 est choisi comme destination

Exemple

```
IODE|XCMT!' ' : tous les commentaires dans le sheet1,
               en R1C1
IODE|XVAR!'QC0,QAFF!Sheet2!R1C1' : comments QC0 et QAFF
               dans le Sheet2 en position R1C1
IODE|XVAR!'QC0!Sheet2!R5C2' : comment QC0 en Sheet2, ligne
                          5 colonne 2
```

7.2.3.2 POKE

ITEMS DE WS (DATA EN FONCTION DU CONTEXTE)

- SAMPLE = change le sample du WS IODE courant (format IODE)
- Name
  - CNAME = change le nom du fichier des commentaires de IODE
  - ENAME = change le nom du fichier des équations de IODE
  - INAME = change le nom du fichier des identités de IODE
  - LNAME = change le nom du fichier des listes de IODE
  - SNAME = change le nom du fichier des scalaires de IODE
  - TNAME = change le nom du fichier des tableaux de IODE
  - VNAME = change le nom du fichier des variables de IODE
- Descr
  - CDESCR = change la description des commentaires de IODE
  - EDESCR = change la description des équations de IODE
  - IDESCR = change la description des identités de IODE
  - LDESCR = change la description des listes de IODE
  - SDESCR = change la description des scalaires de IODE
  - TDESCR = change la description des tableaux de IODE
  - VDESCR = change la description des variables de IODE
- Clear

- CCLEAR = Clear des commentaires dans IODE
- ECLEAR = Clear des équations dans IODE
- ICLEAR = Clear des identités dans IODE
- LCLEAR = Clear des listes dans IODE
- SCLEAR = Clear des scalaires dans IODE
- TCLEAR = Clear des tableaux dans IODE
- VCLEAR = Clear des variables dans IODE

• Load
- CLOAD = Load un fichier de commentaires dans IODE
- ELOAD = Load un fichier d'équations dans IODE
- ILOAD = Load un fichier d'identités dans IODE
- LLOAD = Load un fichier de listes dans IODE
- SLOAD = Load un fichier de scalaires dans IODE
- TLOAD = Load un fichier de tableaux dans IODE
- VLOAD = Load un fichier de variables dans IODE

• Save
- CSAVE = Sauve le fichier de commentaires de IODE
- ESAVE = Sauve le fichier d'équations de IODE
- ISAVE = Sauve le fichier d'identités de IODE
- LSAVE = Sauve le fichier de listes de IODE
- SSAVE = Sauve le fichier de scalaires de IODE
- TSAVE = Sauve le fichier de tableaux de IODE
- VSAVE = Sauve le fichier de variables de IODE

ITEMS DE PLAY

- TEXT : data = texte à placer dans le buffer clavier de IODE
- KEYS : data = texte des touches à placer dans le buffer clavier : aA..aZ, aF1..aF10, cA..cZ, cF1..cF10, sF1..sF10, ESC, ENTER, BACKSPACE, HOME, END, LEFT, RIGHT, UP, DOWN, PGUP, PGDN, DEL, TAB

ITEMS DE CMT, EQS, IDT, LST, SCL, TBL, VAR

Nom de l'objet à modifier. Data contient la valeur avec des TAB comme séparateur dans le cas des variables.

ITEMS DE REP

La valeur de l'item est sans signification. La valeur de data est une commande de rapport à exécuter (ex. '$WsLoadVar test')

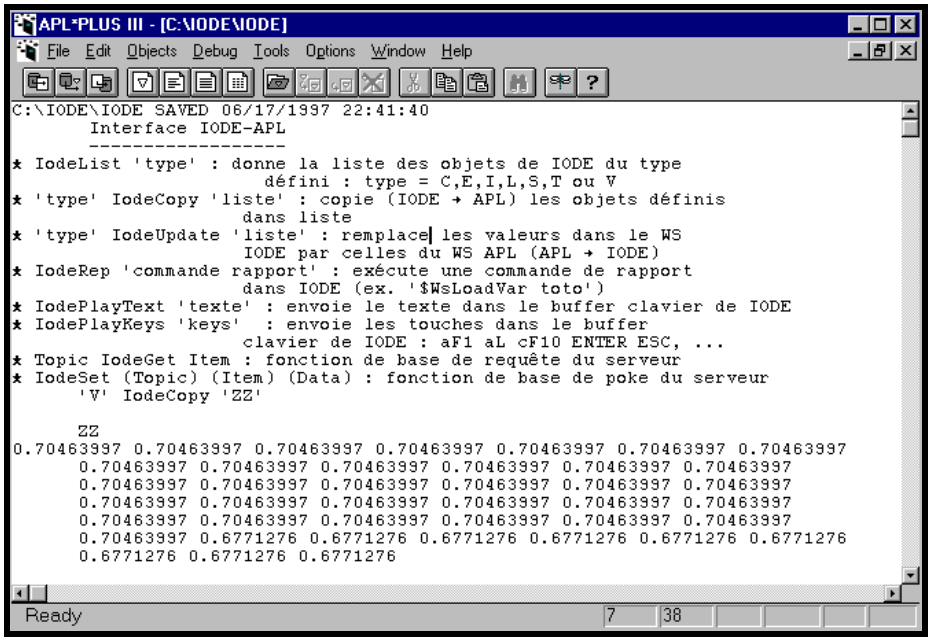
7.3 INTERFACE ENTRE L'APL ET IODE : LE WS IODE.W3

iode.w3 is een workspace met functies in de APL programmeertaal. Deze functies laten toe om objecten uit Iode in de workspace te kopiëren, met APL instructies deze gegevens aan te passen en terug in de Iode omgeving te brengen.

Het is ook mogelijk om vanuit APL Iode rapportcommando's uit te voeren in de Iode omgeving. De gegevens worden via het DDE protocol met elkaar uitgewisseld. Iode en APL moeten opgestart zijn en in APL moet de iode.w3 workspace geladen zijn.

- IodeList geeft de lijst van de IODE geladen objecten
- IodeCopy kopiëert IODE-objecten in APL vectoren
- IodeUpdate update IODE-objecten met waarden uit APL vectoren
- IodeRep laat IODE een rapport uitvoeren
- Demo

FIGURE 100



7.3.1 IODELIST

Xxx prend l'une des valeurs :

```
cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Geeft een lijst van de objecten van het bepaalde type in de Iode omgeving. type is C, E, I, L, S, T of V

Syntaxe

```
      IodeList 'type'
```

Exemple

```
      VARLIST <- IodeList 'V'
```

7.3.2 IODECOPY

Xxx prend l'une des valeurs :

```
      cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Kopieert de objecten, van het bepaalde type, die genoemd zijn in de lijst uit Iode naar APL.

Syntaxe

```
  'type' IodeCopy 'lijst'
```

EXEMPLE

```
      'V' IodeCopy 'ZKT VAR2 VAR3'
```

7.3.3 IODEUPDATE

Xxx prend l'une des valeurs :

```
      cmt | eqs | idt | lst | scl | tbl | var
```

DESCRIPTION

Objecten van het bepaalde type en die genoemd zijn in de lijst worden in de Iode omgeving vervangen door de waarde die ze in APL hebben.

Syntaxe

```
      'type' IodeUpdate 'lijst'
```

Exemple

```
      'V' IodeUpdate 'ZKT VAR2 VAR3'
```

7.3.4 IODEREP

DESCRIPTION

Deze APL functie voert in de Iode omgeving een rapportcommando uit.

Syntaxe

```
  IodeRep 'rapportcommando'
```

Exemple

```
      IodeRep '$WsSaveVar snacap90'
```

7.3.5 APLDEMO

Exemple

```
      IodeRep '$WsLoadVar c:\usr\ode\nov.var'
      'V' IodeCopy 'ZZ ZJ'
      ZZ<- ZZ+ZJ
      'V' IodeUpdate 'ZZ'
      IodeRep 'WsLoadVar c:\usr\ode\dec.var'
```

Deze APL-sessie, zorgt ervoor dat IODE de nov.var variable file laadt. De variabele ZZ en ZJ wordt uit IODE gehaald, opgeteld en als nieuwe waarde voor ZZ in IODE geladen. De laatste lijn schrijft de aangpaste DB weg in de nieuwe dec.var variable file.

8. Methods and algorithms

- Estimation methods
- Computational method for least squares techniques
- Standard statistics
- Simulation algorithm

8.1 Estimation methods

Introduction

The estimation methods owe largely to *Fair* (1984). These first two sections are drawn from this book that we highty recommend.

Econometric models are generally nonlinear, simultaneous, and large. They also tend to have error terms that are serially correlated. The estimation methods of IODE try to handle these characteristics.

The notations that will be used in this chapter are as follows.

Write the model as :

$$f_i(y_t, x_t, a_i) = u_{it}, i = 1, \ldots, n, t = 1, \ldots, T \quad (1)$$

where

- y_t is an n-dimensional vector of endogenous variables,
- x_t is a vector of predetermined variables,
- a_i is a vector of unknown coefficients,
- u_{it} is an error term.

Assume that the first m equations are stochastic, with the remaining u_{it} (i = m+1,...,n) identically zero for all t.

Let J_t be the n x n Jacobian matrix whose ij element is $f_i/\partial y_{jt}$ (i, j = 1..n).

Also, let u_i be the T-dimensional vector (u_{i1},..., u_{iT} ,.., u_{m1} ...,u_{mT})'.

Let **a** denote the k-dimensional vector (a_1',..',a_m') of all the unknown coefficients.

Finally, let Gt' be the ki x T matrix whose t column is $\partial f_i(y_t, x_t, a_i)/\partial a_i$, where ki is the dimension of ai, and let G' be the k x m.T matrix,

$$\begin{bmatrix} G'_1 & 0 & \dots & 0 \\ 0 & G'_2 & \dots & 0 \\ 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & G'_m \end{bmatrix}$$

where $k = {}_i\Sigma_{=1}^{m} k_i$ .These vectors and matrices will be used in the following section.

SERIAL CORRELATION AND DYNAMIC ADJUSTMENT MODEL

A convenient way of dealing with serially correlated error terms is to treat the serial correlation coefficients as structural coefficients and to transform the equations into equations with serially uncorrelated error terms. This introduces nonlinear restrictions on the coefficients, but otherwise the equations are like any others with serially uncorrelated coefficients.

Consider the ith equation of (I) and assume that uit, is first-order serially correlated:

$$u_{it} = \rho_i u_{it-1} + e_{it}, t = 2, \dots, T \qquad (2)$$

where e_{it} is not serially correlated.

Lagging (1) one period, multiplying by ρ_i, and subtracting the resulting expression from (1) yields :

$$f_i(y_t, x_t, a_i) - \rho_i f_i(y_{t-1}, x_{t-1}, a_i) = u_{it} - \rho_i u_{it-1} = e_{it}, t = 2, \dots, T \qquad (3)$$

The equation includes now y_{t-1} and x_{t-1} as variables, and ρ_i as a coefficient, but it is no more general than (1), and thus one can deal directly with (4) under the asumption that serial correlation has been eliminated through transformation.

The same procedure can handle serial correlation of higher orders.

With respect to testing for serial correlation, it is well known that the Durbin-Watson (DW) test is biased toward accepting the null hypothesis of no serial correlation if there is a lagged dependent variable in the equation.

ESTIMATION TECHNIQUES

- Ordinary Least squares (OLS)
- Two-Stage Least Squares (2SLS)
- Three-Stage Least Squores (3SLS)
- Joint estimation of a non simultaneous system

8.1.1 ORDINARY LEAST SQUARES (OLS)

The OLS technique is a special case of the 2SLS technique, where Di in (5) and (6) below is the identity matrix. It is thus unnecessary to consider this technique separately from the 2SLS technique.

8.1.2 TWO-STAGE LEAST SQUARES (2SLS)

2SLS estimates of a_i (say $\hat{a}_i$) are obtained by minimizing

$$u'_i Z_i (Z'_i Z_i)^{-1} Z'_i u_i = u'_i D'_i u_i \quad (4)$$

with respect to a_i, where Z_i is a T x K_i matrix of predetermined variables. Z_i and K_i can differ from equation to equation. An estimation of the covariance matrix of $\hat{a}_i$ (say V^$_{2ii}$) is :

$$\hat{V}_{2ii} = \hat{\sigma}_{ii} (\hat{G}'_i D_i \hat{G}_i)^{-1} \quad (5)$$

where G^$_i$ is G_i evaluated at $\hat{a}_i$, and $\hat{\sigma}_{ii} = T_i^{-1} \Sigma_{=1}^{T} u^2_{it}$, $u_{it} = f_i(y_{t-1}, x_{t-1}, \hat{a}_i)$

The 2SLS estimator in this form is presented in Amemiya(L974).It handles the case of nonlinearity in both variables and coefficients. In earlier work, Kelejian (1971) considered the case of nonlinearity in variables only. Bierens (1981,p.106) has pointed out that Amemiya s proof of consistency of this estimator is valid only in the case of linearity in coefficients, that is, only in Kelejian's case. Bierens supplies a proof of consistency and asymptotic normality in the general case.

8.1.3 THREE-STAGE LEAST SQUORES (3SLS)

3SLS of a (say â) are obtained by minimizing

$$u'[\hat{S}^{-1} \otimes Z(Z'Z)^{-1} Z']u = u'Du \quad (6)$$

with respect to a, where S^ is a consistent estimate of S and Z is a T x K matrix of predetermined variables. As estimate of the covarialce matrix of â (say $\hat{V}_3$) is

$$\hat{V}_3 = (\hat{G'D\hat{G}})^{-1} \quad (7)$$

where $\hat{G}$ is G evaluated at â. S is usually estimated from the 2SLS estimated residuals. This es-

timator is presented in Jorgenson and Laffont (1974), and it is further discussed in Amemiya(1977). Both prove consistency and asymptotic normality of 3SLS.

8.1.4 JOINT ESTIMATION OF A NON SIMULTANEOUS SYSTEM

Two aims can lead to jointly estimate several equations : equality restrictions on coefficients of different equations, and the presumed existence of contemporaneous correlation of the residuals (Seemingly unrelated equations : SURE). In this case, the joint estimation of the equations increases the efficiency of the estimators.

Algorithmically, the method used is the Zellner method which consists in minimizing a weighted sum of the residuals. The weighting matrix is made of the contemporaneous covariance matrix of the residuals estimated with the residuals of a first step OLS on the equations. The Zellner can then simply be considered as a special case of the 3SLS method where the matrix of instruments is an identity matrix. The matrix D above is identically equal to the matrix $\hat{S}^{-1} \otimes I$

8.1.5 FULL INFORMATION MAXIMUM LIKELIHOOD (FIML)

Note : the method described below was implemented in KaA but is not yet available in IODE.

Under the assumption that (u_{1t} , ..., u_{mt}) is independently and identically distributed as multivariate N(O,S), the density function for one observation is

$$(2\pi)^{-m/2}|S^*|^{1/2}|J_t|\exp(-1/2\Sigma_{i,j}u_{it}s_{ij}^*u_{jt}) \tag{8}$$

where $S^* = S^{-1}$ and S_{ij}^* is the ij element of S^*. The Jacobian J_t has been defined before.

The likelihood function of the sample t=1,...,T is

$$L^* = (2\pi)^{-m/2}|S^*|^{1/2}\Pi_{t=1}|J_t|\exp(-1/2\Sigma_{i,j}u_{it}s_{ij}^*u_{jt}) \tag{9}$$

and the log of L* is

$$\log L^* = -(mT)/2\log 2\pi + T/2\log|S^*| + \Sigma_{t=1}^{T}\log|J_t| - 1/2\Sigma_{i,j,t}u_{it}s^*_{ij}u_{jt} \tag{10}$$

Since log L* is a monotonic function of L*, maximizing log L* is equivalent to maximizing L*.

The problem of maximizing log L* can be broken into two parts: the first is to maximize log L* with respect to the elements of S*, and the second is to substitute the resulting expression for S* into (11) and to maximize this "concentrated" likelihood function with respect to a. The derivative of log L* with respect to s_{ij}^* is

$$\partial\log L^*/\partial s^*_{ij} = T/2s^*_{ij} - 1/2\Sigma_{t=1}^{T}u_{it}u_{jt} \tag{11}$$

where s_{ij} is the ij element of S^{*-1}. This derivative uses the fact that = a^*_{ij} for a matrix A. Setting (12) equal to zero and solving for s^*_{ij}= yields

$$s^{*ij} = 1/T\Sigma_{t=1}^{T}u_{it}u_{jt} \tag{12}$$

since and therefore . Substituting (13) into (11) yelds

$$\log L^* = -(mT)/2\log 2\pi + T/2\log|S^*| + {}_i\Sigma^T_{=1}\log|J_t| - Tm/2\,. \qquad (13)$$

The Tm/2 term comes from the fact that

$-(1/2)_i\Sigma_{j,t}u_{it}s^*_{ijjt}u = -(1/2)\Sigma_{i,j}s^*_{ijj}\Sigma^T_{=1}u_{it}u_{jt} = 1/2\Sigma_{i,j}s^*_{ij}Ts^*ij = -Tm/2$. The first and last terms on the RHS of (14) are constants, and thus the expression to be maximized with respect to a consists ofjust the middle two terms. Since $\log|S^*| = \log|S^{-1}| = -\log|S|$, the function to be maximized can be written

$$L = -T/2\log|S| + {}_i\Sigma^T_{=1}\log|J_t| \qquad (14)$$

where, as noted earlier, the ij element of S, S_{ij}, is .

FIML estimates of a are thus obtained by maximazing L with respect to a. An estimate of the covariance matrix of these estimates (say V^$_4$) is :

$$\hat{V}_4 = -(\partial^2 L/\partial a\partial a')^{-1} \qquad (15)$$

where the derivatives are evaluated at the optimum.

Phillips (1982) has pointed out that Amemiya's proof of consistency and asymptotic efficiency (1977) is based on an incorrect lemma. This is corrected in a later paper (Amemiya Ig82).Amemiya's article (1977), as corrected, shows that in the nonlinear case FIML is asymptotically more efficient than 3SI^S under the assumption of normality. In the linear case FIML is consistent even if the error terms are not normally distributed, where "FIML" means the full information maximum liketihood estimator derived under the assumption of normality. In the nonlinear case this is not in general true, although it sometimes is. Phillips (1982) presents an example of a non linear model for which FIML is consistent for a wide class of error distributions.

8.2 Computational method for least squares techniques

Linear and nonlinear least squares

The least squares estimation routine has been designed to be very general and involve all the least squares methods: linear and nonlinear, single or system of equations, uses of instrumental variables or not, with or without correction for contemporaneous autocorrelation of residuals. The core of the computation is made of an iterative process which will be shown with the help of a little example.

Assume that the following equation is to be estimated:

$$y = f(X, a) + e \qquad (16)$$

where y is a (T,1) vector of observations of the dependent variable, X is the (T,Q) matrix of observations of the independent variables. The (Q, 1) vector a of coefficient are to be estimated.

and e is a (T,1) vector of residuals, normally distributed with mean O and standard deviation s. It is also assumed that $E(ee')=s^2 I$, I being the (T,T) identity matrix. The function f is not necessarily linear in the variables and in the coefficients.

The estimator of a is the vector â which minimizes the sum of squares of the residuals of the equation (17):

$$y = f(X, \hat{a}) + e \tag{17}$$

Let S=ee' be the sum of squares of the residuals. Equation (17) can be linearized by a Taylor expansion limited to the first term:

$$y = y_0 + (\partial f / \partial \hat{a})_{|a=\hat{a}_0|}(\hat{a} - \hat{a_0}) + v \tag{18}$$

where $v=u(\hat{a}_O) + e$, and $u(\hat{a}_O)$ represents the terms neglected in the Taylor expansion, $\hat{a}_O$ is a starting value for â, and y_O is defined by $y_O = f(X, \hat{a}_O)$. Equation (18) is linear in $\Delta\hat{a_1} = (\hat{a} - \hat{a_0})$. These parameters can be estimated by OLS, i.e. by

$$MinS = Min(v'v) = Min(u+e)'(u+e) = u'u + 2u'e + e'e \tag{19}$$

A new value for the coefficients can then be computed

$$\hat{a}_1 = \hat{a}_0 + \Delta\hat{a}_1 \tag{20}$$

This operation will be iterated until $\Delta\hat{a}_k$, with k the iteration number, becomes sufficiently close to zero. At this stage, $\hat{a}_k = \hat{a}$, and then $y_k = f(X, \hat{a})$, $u_k = 0$ and S = e'e.

The iterative process could well not converge, although it can be proved that, if at each iteration $\Delta\hat{a}_k$ is weighted by λ, with $|\lambda| < 1$, so that $\hat{a}_k = \hat{a}_k + \lambda\hat{a}_k$, the process must converge for sufficiently small values of λ, provided that there is a minimum.

It is very easy to approximate the derivative of the f function, this is done numerically. In the linear case, the approximation is exact and gives the matrix of regressor observations. The iterative process reaches a solution within two iterations. In the nonlinear case, the matrix of derivatives is no more the matrix of observations, and a progressive correction process of the coefficients is repeated until the correction to apply is sufficiently small, i.e. smaller than a convergence threshold given by the user. At this point, the estimators minimize a weighted sum of the residuals of the equations, the S = S(â) function, the complexity of which is determined by the degree of non linearity of the system of equations. Nothing guaranties in very special nonlinear cases that the process will eventually converge, or converge within the maximum number of iterations which was given by the user.

A weight (λ in the foregoing) can be introduced to accelerate convergence for each coefficient, these weights are called step relaxation parameters or smoothing parameters, by default their values are one, this value can be updated in the equation update screen. The step relaxation parameters are important for another reason: if the value of one or more relaxation parameters is set to zero, the initial value of the corresponding coefficients will be unchanged. This is a way to constraint a coefficient to a predetermined value.

The same iterative procedure is applied for a system of equations, y_k and u_k are no longer vectors of the T computed values of the endogenous variable and residuals, but are matrices of rank T times the number of equations of the system.

$$G_k = \partial f(y_t, x_t, a_{k-1})/\partial a_k \tag{21}$$

$$\Delta\hat{a}_k = (G'_K G_K)^{-1} G_k (y - y_k) \tag{22}$$

$$\hat{a}_k = \hat{a}_{k-1} + \lambda \Delta\hat{a}_k \tag{23}$$

2SLS and 3SLS are incorporated in the procedure by first calculating a weighting matrix (a metric) using the specified instrumental variables, and then using this metric in the iterative computation of the least squares estimators of $\Delta\hat{a}$. In this case, (22) becomes

$$\Delta\hat{a}_k = (G'_K D_1 G_K)^{-1} G_k D_1 (y - y_k) \tag{24}$$

with $D_1 = I \otimes Z(Z'Z)^{-1}Z'$

3SLS and Zellner method require an additional iterative process : the first iterative process allows to compute an estimate of the covariance matrix of the contemporaneous residuals, the second iterative process reestimates the coefficients by correcting the metric matrix. (22) becomes :

$$\Delta\hat{a}_k = (G'_K D_2 G_K)^{-1} G_k D_2 (y - y_k) \tag{25}$$

with $D_2 = S^{-1} \otimes Z(Z'Z)^{-1}Z'$, and S estimated during the first step.

The least squares estimation program is very general. It can handle linear or nonlinear equations, single equations or a system of equations. The estimated coefficients have the BLUE properties if the model is linear, but only the asymptotic properties are ensured for non-linear models.

The estimated covariance matrix of the coefficients is exact for the linear models, but only asymptotically exact for non-linear models.

FIML

The algorithm used in KaA to calculate FIML estimates is a Gauss-Newton one. Numerical derivatives of the likelihood function are computed like in the least squares algorithm. The weighting matrix is the same as in the 3SLS procedure, implying that the user must specify a meaningful set of instrumental variables. (Moreover, the 3SLS estimates are strongly recommended as starting values.)

8.3 Standard statistics

A limited number of statistics are given as standard output by the estimation program.

Per coefficient : the value of the coefficient, the standard error of the coefficient, and the t-statistics. In the non-linear case the standard errors of the coefficients only represent a lower limit of the true standard errors which will be reached asymptotically.

When the model is a linear single equation with a constant term, the total variance of the endogenous variable is equal to the variance explained by the regression plus the residual variance. The R2 statistics represents the explained share of the total variance in percents. This statistics is corrected for the number of coefficients. Out of this limited case, the interpretation of this statistics is difficult and hazardous.

The standard error gives a more straightforward interpretation of the quality of the adjustment.

8.4 SIMULATION ALGORITHM

THE GAUSS-SEIDEL TECHNIQUE

Most macroeconometric models are solved using the Gauss-Seidel technique. It is a simple technique and in most cases works remarkably well. This technique is used in IODE to simulate models. The Gauss-Seidel technique is easy to describe by means of an example.

Assume that the model consists of three equations, and let X_{it} denote the vector of predetermined variables in equation i. The model is as follows :

$$f_1(y_{1t}, y_{2t}, y_{3t}, x_{1t}, a_1) = u_{1t} \tag{26}$$

$$f_2(y_{1t}, y_{2t}, y_{3t}, x_{2t}, a_2) = u_{2t} \tag{27}$$

$$f_3(y_{1t}, y_{2t}, y_{3t}, x_{3t}, a_3) = u_{3t} \tag{28}$$

The technique requires that the equations be rewritten with each endogenous variable on the LHS of one equation. This is usually quite easy for macroeconometric models, since most equations have an obvious LHS variable. If, say, the LHS variable for an estimated equation is `ln(y2t /y1t)`, then y_{2t} can be written on the LHS by taking exponents and multiplying the resulting expression by y_{3t}. The program can do that job for you since, for each equation, it requires the definition of the endogenous variable specified by the equation, provided that the endogenous variable appears only once in the equation. The endogenous variable can appear either on the LHS or in the RHS. Simulation of a model requires moreover that the endogenous variable be specified only once in the full set of equations.

When these transformations of the equations are done, the model can be rewritten as follows :

$$y_{1t} = g_1(y_{2t}, y_{3t}, x_{1t}, a_1, u_{1t}) \tag{29}$$

$$y_{2t} = g_2(y_{1t}, y_{3t}, x_{2t}, a_2, u_{2t}) \tag{30}$$

$$y_{3t} = g_3(y_{1t}, y_{2t}, x_{3t}, a_3, u_{3t}) \tag{31}$$

In order to solve the model, values of the coefficients and the error terms are needed. Given these values and given values of the predetermined variables, the solution proceeds as follows. Initial values of the endogenous variables are guessed. These are either the actual values or extrapolations from the previous period according to some rules. IODE allow various strategies. Given these values the equations can be computed successively and produce a new set of values for the endogenous variables. Given this new set of values, the equations can again be solved to get another set. Convergence is reached if for each endogenous variables the values on successive iterations are within some prescribed tolerance level.

There is no guarantee that the Gauss-Seidel method will converge. The advantage of the technique, however, is that it can usually be made to converge (assuming an actual solution exists) with sufficient damping. By damping is meant the following. Let y_{k-1} denote the solution value of y for iteration k-1 and let $y^{\wedge}_k$ denote the value computed by solving the equation on iteration k. Instead of using y_k as the solution value for iteration k, one can adjust y_{k-1} only partway toward

y_k.

If λ is 1, there is no damping.

$$y_k = y_{k-1} + \lambda(\hat{y}_k - y_{k-1}), o < \lambda \leq 1 \tag{32}$$

The question of what to use as a stopping rule is not as easy as it might sound. The stopping rule can either be in absolute or percentage terms. In absolute terms it is

$$|y_{it,k} - y_{it,k-1}| < \varepsilon_i \tag{33}$$

and in percentage it is

$$\left| \frac{y_{it,k} - y_{it,k-1}}{y_{it,k-1}} \right| \varepsilon_i \tag{34}$$

where ε_i, is the tolerance criterion for variable i. If damping is used $y_{it,k}$ must be replaced by $\hat{y}$. A global criterion can also be used

$$\left(\sum_i (y_{it,k} - y_{it,k-1})^2 \right)^{1/2} < \varepsilon_1 \tag{35}$$

The problem is to choose tJee values for ε_1. It is inconvenient to have to choose different values of the tolerance criterion for different variables, and one would like to use just one value of e throughout. This is not, however, a sensible procedure if the units of the variables differ and if the absolute criterion is used. The problem is lessened if the percentage criterion is used, but in this case one must be concerned with variables that can be close to zero or can be zero.

The solution adopted in IODE is to compute a stopping rule which combines the three afore-mentioned criteria

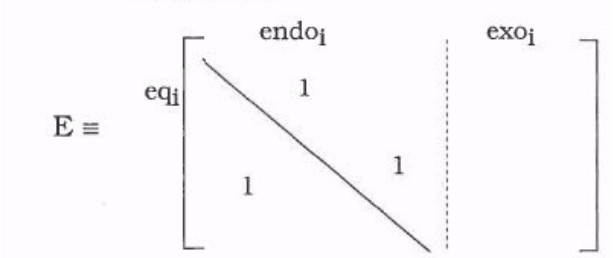
$$Min\left(\sum_i (y_{it,k} - y_{it,k-1})^2 \right)^{1/2}, \underset{i}{Max} \left\{ Min\left(\left| \frac{y_{it,k} - y_{it,k-1}}{y_{it,k-1}} \right| |y_{it,k} - y_{it,k-1}| \right) \right\} < \varepsilon_1 \tag{36}$$

EQUATION REORDERING

To increase the simulation speed, the model can be splitted Into connex components. Each component will be reorganized by an heuristical algorithm to obtain a quasi subdiagonal model.

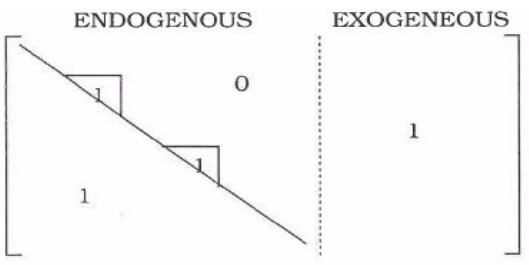
Let E denote tle incidence matrix:

E_{ij} = 1 if the variable j belong to the equation
= 0 otherwhise



(37)

After decomposition, the incidence matrix E has the following form :



CONNEX COMPONENTS CONSTRUCTION

Let VARS be the list of endogenous variables

- 1. Take the first remaining endogenous variable in VARS : $ENDO_0$
- 2. Compute SUCC = list of variables depending on $ENDO_0$ (recursively)
- 3. Compute PRE = list of variables from which depends $ENDO_0$ (recursively)
- 4. Compute CFC_i intersection between SUCC and PRE : new connex component
- 5. Let VARS = VARS \ CFC_i
- 6. If $VARS \neq \varnothing$ go to 1, else stop

Reordering of the connex components

Construct the *E'* matrix :

$$E'ij = 1 \Leftrightarrow \exists k \in \exists 1 \in CFCj : Ekl = 1 \qquad (38)$$
$$0 \text{ otherwise}$$

To make E' lower triangular, apply the following algorithm (O is the new order of connex components) :

- 1. O={}
- 2. Choose $i \notin 0$ so that $E'_{ij} = 0 \forall j \neq i$
- 3. Set $E'_{ij} = 0 \forall j$
- 4. Add i in O
- 5. Go to 2 if at least one *CFC* remains

Triangulation of a connex component

Each connex component (*CFC*) can be reorganized to minimize the computation time required to solve the corresponding system. The following algorithm reduces approximatively the number of iterations by a factor 2. It is based on the fact that to maximize the information during an iteration, the extra-diagonal elements of E must be as close as possible to the diagonal.

- 1. compute the sum of extra-diagonal elements,

$$S1 = \sum_{i} \sum_{<j} E_{ij}$$
$$S1 = \sum_{i} \sum_{<j} E_{ij} \cdot (j - i)$$
$$S3 = \max_{j} \{ -i : E_{ij} = 1 \}$$

- 2. $\forall i \in CFC$ set equation *i* after equation *k*: $k = {}_{k}max\ {}_{>i}k : (E_{ki} = 1)$

- 3. Compute *S1',S2'* and *S3'* as *S1, S2* and *S3*
  if *S1'<S1 and S2'<S2 and S3'<S3:*
  S1=S1',S2=S2', S3=S3'
  go to 2

  else stop

Goal seeking

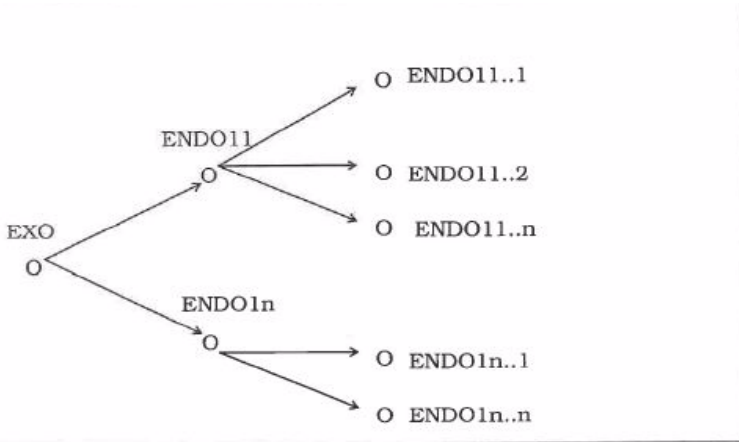
Given n pair ENDOi-EXOi variables, the solution of the system of equations must now be computed for the new set of variables consisting of the initial group in wich the ENDO variables for the given list have been replaced by EXO variables.

If we where to use a NEWTON algoritm or an assimilated method, we would just have to solve the system for the new set of variables. In the case of the GAUSS algorithm however one must declare a new output variable for each equation.

As an example suppose we do one endo-exo exchange. (for more pairs we just repeat the proces).

The goal is to find a pair of equations, the first of wich contains EXO and the endogenious variable of the second being ENDO

We therefore construct a structured tree as below where an arrow indicates that a variable belongs to the definition equation of another variable.



9. ANNEXES

- Annexe 1 : Syntaxe du programme iode
- Annexe 2 : Syntaxe du programme iodecmd
- Annexe 3 : Syntaxe des rapports
- Annexe 4 : Sample d'impression
- Annexe 5 : Formats ASCII des objets
- Annexe 6 : Lexique
- Annexe 7 : L'éditeur MMT
- Annexe 8 : Génération d'un fichier d'aide Windows
- Annexe 9 : Syntaxe des fichiers a2m

9.1 SYNTAXE DU PROGRAMME IODE

Le programme IODE peut être lancé avec ou sans argument.

Les arguments éventuels passés à IODE servent à exécuter certaines tâches au lancement de IODE, comme charger un workspace ou lancer l'exécution d'un rapport.

En utilisant la commande

```
c:\iode> iode -h
```

la syntaxe de IODE est affichée.

SYNTAXE DE IODE

```
OPTION    ARGUMENTS      DESCRIPTION
------    ---------      -----------
-h                       display this message and exit
-ws                      load ws.* at startup
-rep       reportname    exécute le rapport indiqué
-erep      reportname    édite le rapport indiqué
-l         wsname        lance IODE et charge un WS
-p         wsname        lance IODE et imprime un WS
-nc        number        set the number of columns of the iode window
-nl        number        set the number of lines of the iode window
-fontsize nPts           set the character size (in points : [3-50]
-fontname fontname       set the font (default Courier New)
-inifile  filename       file containing iode parms

Advanced parameters
-------------------
-seg       segsize       set the allocation segment size (dft 16384)
-pbf       filename      execute a 'play back' file
-pbs       string        execute a 'play back' string
-rec       filename      record keys in a playback file
```

L'option -ws

Cette option charge les workspaces ws.* présents dans le directory d'où IODE est lancé.

L'option -rep

Cette option permet de démarrer un rapport automatiquement. Cela offre la possibilité par exemple de créer un environnement de travail ou encore de démarrer une procédure d'encodage ou d'exemple.

On entrera par exemple :

```
c:\usr\iode> iode -rep simul.rep
```

L'option -erep

Cette option permet de lancer l'éditeur de rapports automatiquement. Contrairement à l'option -rep, celle-ci n'exécute pas le rapport.

On entrera par exemple :

```
c:\usr\iode> iode -erep simul.rep
```

L'option -l

Cette option permet de démarrer IODE et de charger automatiquent un WS.

On entrera par exemple :

```
c:\usr\iode> iode -l mydata.var
```

L'option -p

Cette option permet de démarrer IODE et d'imprimer automatiquent la définition des objets d'un WS.

On entrera par exemple :

```
c:\usr\iode> iode -p mydata.eqs
```

Les options -nl et -nc

Ces options permettent de changer au démarrage la taille de la fenêtre de IODE. La taille est exprimée en caractères.

Elles sont prioritaires par rapport aux définitions du fichier iode.ini.

On entrera par exemple :

```
c:\usr\iode> iode -nl 35 -nc 100
```

Les opions -fontsize et -fontname

Ces options déterminent la taille et la police de caractères de la fenêtre de IODE.

```
c:\iode> iode -fontsize 10 -fontname Terminal
```

Elles sont prioritaires par rapport aux définitions du fichier iode.ini.

Les options -rec, -pbf et -pbs

L'option `-rec` enregistre dans un fichier les séquences de touches utilisées pendant une session IODE.

```
c:\iode> iode -rec filename
```

Le fichier (ici **`filename`**) contiendra des séquences de touches codifiées selon la syntaxe décrite dans la version 6.27 de iode. Ce fichier peut être édité à l'aide d'un éditeur ascii.

Ce fichier de séquences de touches peut ensuite être "rejoué" automatiquementà l'aide de l'option **`-pbf`**.

```
c:\iode> iode -pbf reckeys.keys
```

Cela permet par exemple :

- de lancer des opérations répétitives sans avoir à passer par les rapports
- de lancer des procédures de test unitaires

Toutes les opérations ne sont cependant pas exploitables : les opérations effectuées à l'aide de la souris ne sont pas supportées. Il faut également être attentif au fait que la position dans un menu ou le contenu d'un écran de saisie peuvent changer entre deux lancements successifs de IODE.

L'option **`-pbs`** permet de "jouer" une séquence de touches passées comme paramètre.

```
c:\iode> iode -pbs "&aw&ar@R"
```

L'option -seg

Cette option permet de définir la taille des blocs mémoire alloués par IODE.

```
c:\iode> iode -segsize 32768
```

9.2 SYNTAXE DU PROGRAMME IODECMD

SYNTAXE

```
iodecmd [-nbrun n] [-alloclog filename] [-v] [-y] [-h] reportfile [arg1] ... [argn]
```

where :

- **`-nbrun n`** : n = nb of repetitions of the report execution (default 1)
- **`-allocdoc filename`** : memory debugging info stored in filename (developers only)
- **`-v`** : verbose version (all messages displayed)
- **`-y`** : answer yes to all questions asked during the report execution
- **`-h`** : display program syntax (this message)
- **`reportfile arg1 ...`** : IODE report to execute including its optional arguments

9.3 SYNTAXE DES RAPPORTS

Les mots-clés commencent par une majuscule (Label, NoPrint, ...). Les paramètres qui ne sont pas des mots-clés commencent par une minuscule (labelname, question_text, command, ...).

COMMANDES SPÉCIFIQUES AUX RAPPORTS

Toutes ces commandes sont préfixées indifféremment par $ ou #.

```
$Ask          labelname question_text
$Beep
$Goto         labelname [{$cmd} | {LEC-formula} | 0 | 1}]
$Label        labelname
$OnError      {Ignore | Return | Abort | QuitOde
               | Print | NoPrint | Display}
$Return
$Abort
$Quit
$Show         free_text
$Msg          free_text
#Prompt       macroname question
$System       command;command;...
$Define       macroname text
$SetTime      period
$IncrTime     [step]
$shift
```

DÉFINITION DES OUTPUTS

```
$PrintDest  [filename [{A2M|RTF|MIF|CSV|HTML}]]
$PrintDestNew  [filename [{A2M|RTF|MIF|CSV|HTML}]]
$PrintNbDec nn (-1 pour une adaptation automatique à la valeur)
$PrintLang  {Fr | Nl | En}

$PrintA2mAppend [NO|Yes]
$PrintFont      Time|Helvetica|Courier|Bookman|Palatino [size [incr]]
$PrintTableFont Time|Helvetica|Courier|Bookman|Palatino [size]
$PrintTableBox  n
$PrintTableColor [NO|Yes]
$PrintTableWidth width [col1 [coln]]
$PrintTableBreak [NO|Yes]
$PrintTablePage  [NO|Yes]

$PrintBackground Black|Blue|Magenta|Cyan|Red|Green|Yellow|White
$PrintGraphBox   n
$PrintGraphBrush pct|Yes
$PrintGraphSize  width_mm [height_mm [fontsize_pts]]
$PrintGraphPage  [NO|Yes]

$PrintRtfHelp    [YES|No]
$PrintRtfTopic   titre du topic
$PrintRtfLevel   [+|-|n]
$PrintRtfTitle   titre du help
$PrintRtfCopyright copyright text
$PrintParanum    [NO|Yes]
$PrintPageHeader titre des pages suivantes
$PrintPageFooter  footnote des pages suivantes
```

FONCTIONS DE FICHIERS

Les fonctions ayant A2m comme suffixe existent pour toutes les extensions suivantes :

```
Cmt,    Eqs,    Idt,    Lst,    Scl,    Tbl,    Var,    Ooo,
Asc,    A2m,    Dif,    Rep,    Log,    Prf,    Mif,    Rtf,    Agl
```

Non Fullscreen

```
$FileList       mask
$FileEdit       filename
$FileDeleteA2m  filename ...
$FileRenameA2m  filename newname
$FileCopyA2m    filename newname
```

Fullscreen

```
#FileList
#FileEdit
#FileDelete
#FileRename
#FileCopy
```

FONCTIONS DE WORKSPACES

Les fonctions ayant Cmt comme suffixe existent pour toutes les extensions suivantes :

```
Cmt,    Eqs,    Idt,    Lst,    Scl,    Tbl,    Var
```

Non Fullscreen

```
$WsLoadCmt      filename
$WsSaveCmt      filename
$WsDescrCmt     newdescription
$WsClearCmt
$WsCopyCmt      file1,file2,... name1 name2 ...
$WsCopyVar      file1,file2,... [perfrom perto] name name ...
$WsMergeCmt     file
$WsSample       perfrom perto
$WsExtrapolate  [method] perfrom perto [name name ...]
```

Fullscreen

```
#WsLoad
#WsCopy
#WsMerge
#WsSave
#WsDescr
#WsClear
#WsSample
#WsExtrapolate
```

FONCTIONS DE DONNÉES

Les fonctions ayant Cmt comme suffixe existent pour toutes les extensions suivantes, sauf mention du contraire :

```
Cmt,    Eqs,    Idt,    Lst,    Scl,    Tbl,    Var
```

Non Fullscreen

```
$DataUpdateCmt name value
$DataUpdateEqs name value
$DataUpdateIdt name value
$DataUpdateLst name value
$DataUpdateScl name [value [relax]]
$DataUpdateTbl name lec;lec;lec...
$DataUpdateVar name [{Level | Grt | Diff}] perfrom val1 val2 ...

$DataCreateCmt      name
$DataExistCmt       name
$DataDeleteCmt      name
$DataRenameCmt      oldname newname /* Not for Eqs */
$DataDuplicateCmt   oldname newname /* Not for Eqs */
$DataAppendCmt      name text
$DataAppendIdt      name text

$DataSearchCmt      text word ecase names lec text outlistname
                    where word, ecase, names, lec, text are {0 | 1}
$DataListSort       inputlistname outputlistname
$DataScanIdt        [name1 name2 ...]
$DataScanTbl        [name1 name2 ...]
$DataScanEqs        [name1 name2 ...]

$DataEditCnf        {Level | Grt | Dif} nbdec
$DataCalcVar        varname lec_expression
$DataDisplayGraph   {Level | Diff | Grt}
```

```
                        {Line | Scatter | Bar | Mixt}
                        {NoXGrids | MinorXGrids | J(MajorXGrids)}
                        {NoYGrids | MinorYGrids | J(MajorYGrids)}
                        {Level | G(Log) | Semi-Log | Percent}
                        {ymin | --} {ymax | --}
                        perfrom perto varname1 varname2 ...

    $DataSaveGraph      aglfilename
                        {Level | Diff | Grt}
                        {Line | Scatter | Bar | Mixt}
                        {NoXGrids | MinorXGrids | J(MajorXGrids)}
                        {NoYGrids | MinorYGrids | J(MajorYGrids)}
                        {Level | G(Log) | Semi-Log | Percent}
                        {ymin | --} {ymax | --}
                        perfrom perto varname1 varname2 ...
```

Fullscreen

```
    #DataEditCmt        [name1 name2 ...]
    #DataSearch
    #DataDuplicate
    #DataListSort
    #DataDisplayGraph
    #DataSaveGraph
    #DataEditCnf
```

FONCTIONS D'IMPRESSION

Les fonctions ayant Cmt comme suffixe existent pour toutes les extensions suivantes, sauf mention du contraire :

```
    Cmt,    Eqs,    Idt,    Lst,    Scl,    Tbl,    Var
```

Non Fullscreen

```
    $PrintObjDefCmt     [name ...] (dft = all)
    $PrintTblFile       2-5 varfilename
    $PrintTbl           gsample tblname1[+tblname2+..] ...
    $PrintVar           gsample lec1 lec2 ...

    $ViewTblFile        2-5 varfilename
    $ViewTbl            gsample tblname1[+tblname2+..] ...
    $ViewVar            gsample lec1 lec2 ...

    $ViewGr             gsample tblname1[+tblname2+..] ...
    $PrintGrAll         gsample tblname1[+tblname2+..] ...
    $PrintGrData        gsample tblname1[+tblname2+..] ...
    $PrintGrWin         gsample tblname1[+tblname2+..] ...
```

Fullscreen

```
    #PrintObjDefCmt
    #PrintTbl
    #ViewTbl
    #ViewGr
    #PrintGr
```

SIMULATION

Non Fullscreen

```
$ModelCalcSCC        nbtri lstpre lstinter lstpost
$ModelSimulate       perfrom perto [eqname1 eqname2 ...]
$ModelSimulateSCC    perfrom perto prelist interlist postlist
$ModelSimulateParms  eps relax maxit
                     {Connex | Triang | None}
                     0-4                          (starting values)
                     {Yes | No}                   (debug)
                     {Yes | No}                   (debug sous-itération)
                     nbtri
$ModelExchange       eqname1-varname1,eqname2-varname2,...
$ModelCompile        [eqname1 eqname2,...]
```

Fullscreen

```
#ModelSimulate
#ModelCompile
```

EXÉCUTION D'IDENTITÉS

Non Fullscreen

```
$IdtExecute             perfrom perto [idtname1 ...]
$IdtExecuteVarFiles     varfilename1 ...
$IdtExecuteSclFiles     sclfilename1 ...
$IdtExecuteTrace        {Yes | No}
```

Fullscreen

```
#IdtExecute
```

ESTIMATION

Non Fullscreen

```
$EqsSetMethod   {0|1|2|3|4} eqname1 eqname2 ...
$EqsSetSample   from to   eqname1 eqname2 ...
$EqsSetBloc     eq1 eq2 eq3 ...
$EqsSetInstrs   eqname instr1;intr2;instr3;...
$EqsSetCmt      eqname comment
$EqsEstimate    perfrom perto eqname1 ...
```

Fullscreen

```
#EqsEstimate
```

RAPPORTS

Non Fullscreen

```
$ReportExec     reportfilename1 ...
$ReportEdit     reportfilename
```

Fullscreen

```
#ReportExec
#ReportEdit
```

9.4 DÉFINITION DU SAMPLE D'IMPRESSION

La définition du sample d'impression concerne la compilation et l'impression ou la visualisation des tables et des graphiques.

Les valeurs à imprimer ou à visualiser figurent dans la définition des tables et des graphiques sous forme d'une formule LEC. Cette forme LEC ne donne aucune indication sur l'échantillonnage (période(s) d'impression) ni sur la représentation (en valeur relative, en taux de croissance,...) de la valeur à imprimer. Ces dernières informations sont apportées par le sample d'impression.

Le sample d'impression contient des informations sur:

- l'échantillonnage des périodes relatives aux variables ou aux valeurs calculées à imprimer
- les opérations à effectuer sur les variables à imprimer
- les fichiers où se trouvent les variables à imprimer ou intervenant dans le calcul des valeurs à imprimer.

SYNTAXE D'UN SAMPLE D'IMPRESSION

La syntaxe d'un sample répond aux règles décrites ci-dessous. Un sample est composé de périodes, d'opérations sur des périodes, d'opérations sur des fichiers, de facteur(s) de répétition.

Syntaxe d'une période

- une période s'indique comme en LEC : `yyPpp` ou `yyyyPpp` où `yyyy` indique l'année, `P` la périodicité et `pp` la sous-période `(1990Y1)`
- une période peut être décalée de n périodes vers la gauche ou la droite à l'aide des opérateurs `<n` et `>n`
- When used with a null argument, the shift oprerators have a special meaning :
  - `<0` means "first period of the year"

- `>0` means "last period of the year"

- les périodes spéciales **`BOS, EOS`** et **`NOW`** peuvent être utilisées pour représenter le début, la fin du sample courant ou la période actuelle (horloge du PC).
- les périodes spéciales **`BOS1, EOS1`** et **`NOW1`** sont équivalentes aux précédentes à ceci près qu'elles sont déplacées à la première sous période de l'année de **`BOS, EOS`** et **`NOW`** respectivement (si **`NOW`** = **`2012M5`**, **`NOW1`** = **`2012M1`**).
- Chaque période est séparée de la suivante par un point-virgule.
- Une période ou un groupe des périodes peuvent être répétées : il suffit de placer après la définition de la colonne ou du groupe de colonnes le caractère double point (:) suivi du nombre de répétitions souhaité. La répétition se fait avec un incrément de une période, sauf si elle est suivie d'une astérisque et d'une valeur. Cette valeur est alors l'incrément de répétition. Elle peut être négative auquel cas les périodes se présenteront de manière décroissante.
- La répétition, l'incrément et le shift peuvent être les mots **`PER`** (ou **`P`**) ou **`SUB`** (ou **`S`**) qui indiquent respectivement le nombre de périodes dans une année du sample courant et la sous période courante.
- La définition des fichiers est optionnelle et est placée entre crochets. Elle s'applique à toute la définition de période qui précède.

Opérations sur les fichiers

Les opérations possibles sur les fichiers sont :

- valeur absolue `[1]`
- différence `[1-2]`
- différence en pourcents `[1/2]`
- somme `[1+2]`
- moyenne `[12]`

Opérations sur les périodes

Les opérations sur les périodes sont :

- valeur (75)
- taux de croissance sur une ou plusieurs périodes (75/74, 75/70)
- taux de croissance moyen (75//70)
- différence (75-74, 75-70)
- différence moyenne (75--70)
- moyenne (7574)
- somme de période à période consécutives (70Q1+70Q4)
- valeur en indice ou en base (76=70)

La répétition peut s'effectuer avec un incrément supérieur à 1 ou inférieur à 0 : il suffit de placer une étoile suivie du pas après le nombre de répétitions (70:3*5 = 70, 75, 80).

EXEMPLE

```
70; 75; 80:6
= 70:3*5; 81:5
= 70; 75; 80; 81; 82; 83; 84; 85

70/69:2
= 70/69; 71/70

(70; 70-69):2
= 70; 70-69; 71; 71-70;

70[1,2]:2*5
= 70[1]; 70[2]; 75[1]; 75[2]

(70;75)[1,2-1]
= 70[1];75[1];70[2-1];75[2-1]

(70;75;(80; 80/79):2)[1,2]
= 70[1]; 70[2]; 75[1]; 75[2]; 80[1];
    80[2]; 80/79[1]; 80/79[2] 81[1]; 81[2];
        81/80[1]; 81/80[2]


2000Y1>5
= 2005Y1

1999M1>12
= 2000M1

EOS<1
= 2019Y1                    (si EOS == 2020Y1)

BOS<1
= 1959Y1                    (si BOS == 1960Y1)

EOS<4:5*-1
=2016;2017;2018;2019;2020   (si EOS = 2020Y1)
```

In may 2012, assuming that **1990M6:2020M12** is the current sample, the following samples are equivalent :

```
BOS:2;EOS   ==   1990M6;1990M7;2020M12
EOS;EOS1    ==   2020M12:2020M1
NOW;NOW1    ==   2012M5;2012M1
NOW:P       ==   2012M5;2012M6;...;2013M4
NOW:3*P     ==   2012M5;2013M5;2014M5
NOW1:SUB    ==   2012M5

2000Y1>5      === 2005Y1
1999M1>12     === 2000M1
EOS<1         === 2020M11
EOS<P         === 2019M12
```

If the sample is 1960Y1:2020Y1,

```
EOS:5*-1       === 2020;2019,2018,2017,2016
BOS<1          === 1959Y1
```

In may 2012, if the current sample is 1990Q1:2012Q4, the following samples are equivalent :

```
NOW;NOW1<4     === 2012Q2;2011Q1 (for quaterly data)

NOW<0;NOW>0     === 2012Q1;2012Q4
NOW<0:P         === 2012Q1;2012Q2;2012Q3;2012Q4
NOW:3*P         === 2012Q2;2013Q2;2014Q2

NOW;(NOW<0>1/NOW<0)>P    === 2012Q2;2013Q2/2013Q1
```

9.5 FORMATS ASCII DES OBJETS

Les formats de fichiers suivants sont décrits dans ce chapitre :

- Format ASCII de IODE : reprend la syntaxe de définition des 7 types d'objets gérés par IODE
- Format DIF de IODE : interface (obsolète) avec des programmes externes
- Format PRN d'AREMOS : interface (obsolète) avec le logiciel AREMOS

9.5.1 FORMAT ASCII DE IODE

Tous les objets de IODE peuvent être décrits par un format ASCII approprié. L'intérêt en est évident: d'une part il est possible d'éditer tous les objets à l'aide d'un simple éditeur ASCII, d'autre part, il est aisé de développer des filtres permettant de convertir les définitions d'objet vers d'autres langages d'interface.

Les formats ASCII de IODE se composent des éléments suivants:

- un nom d'objet
- des délimiteurs de blocs (en général { et })
- des mots clés spécifiques suivis de leurs paramètres
- des commentaires.

Nous allons décrire successivement les formats ASCII des diférents objets:

- format ASCII des commentaires
- format ASCII des équations
- format ASCII des identités
- format ASCII des listes
- format ASCII des scalaires
- format ASCII des tableaux
- format ASCII des variables

9.5.1.1 FORMAT ASCII DES COMMENTAIRES

Définition

Le format ASCII d'un fichiers de commentaire se compose successivement de la définition ASCII individuelle des commentaires. La fin du fichier signale la fin de la définition des commentaires.

```
début du fichier
format ASCII commentaire_1
format ASCII commentaire_2
format ASCII commentaire_3
format ASCII commentaire_4
...
fin du fichier
```

Le format ASCII d'un commentaire se compose:

- du nom du commentaire (un nom valide au sens IODE, c-à-d un nom de 10 caractères maximum (20 à partir de la version 6.01) -lettre ou chiffre- commencant toujours par une lettre -ou le caractère '_'- ,en majuscule) suivi du texte du commentaire entre guillemets.

Exemple

```
CMT1 "ceci est le commentaire 1"
CMT2 "ceci est le commentaire 2"
```

9.5.1.2 FORMAT ASCII DES ÉQUATIONS

Un fichier ASCII d'équations est composé d'une suite de définition d'équations. Pour chaque équation, on trouve son nom suivi, entre accolades, des informations qui la définissent.

Parmi ces informations, seule la forme LEC (entre guillemets) est obligatoire. Les autres sont optionnelles et concernent essentiellement l'estimation.

Les informations optionnelles sont :

- un commentaire : COMMENT "texte"
- la méthode d'estimation pouvant prendre une des valeurs suivantes :
  - LSQ : méthode des moindres carrés
  - ZELLNER : méthode de Zellner
  - INF : méthode instrumentale (2 stages)
  - GLS : moindre carrés généralisés
- le sample d'estimation sous la forme SAMPLE from to
- la date de dernière mise à jour : DATE yyyymmdd
- le block auquel appartient l'équation sous la forme BLOC "eq1;eq2;..."
- les instruments servant dans l'estimation : INSTRUMENTS "lec1;lec2..."
- les tests statistiques résultant de l'estimation :
  - standard deviation : STDEV valeur
  - moyenne des observations : MEANY valeur
  - somme des carrés des résidus : SSRES valeur
  - standard error : STDERR valeur
  - f-stat : FSTAT valeur
  - r carré : R2 valeur
  - r carré ajusté : R2ADJ valeur
  - Durbin-Watson : DW valeur

- Log likelihood : LOGLIK valeur

Exemple

```
B {
        "B := c1 + c2 * A"
                BLOCK "B"
        COMMENT " "
        SAMPLE 1970Y1 1990Y1
        DATE 19930226
}

X {
        "X := c1 + c2 * Y"
        LSQ
        BLOCK "X"
        COMMENT " "
        SAMPLE 1960Y1 1980Y1
        DATE 19921018
        STDEV 0.228351
        MEANY 2.68489
        SSRES 0.00182236
        STDERR 0.0142297
        FSTAT 2566.21
        R2 0.996505
        R2ADJ 0.996117
        DW 0.512136
        LOGLIK 32.272
}
```

9.5.1.3 FORMAT ASCII DES IDENTITÉS

Définition

Le format ASCII d'un fichiers d'identités se compose successivement de la définition ASCII individuelle des identités. La fin du fichier signale la fin de la définition des commentaires.

```
début du fichier
format ASCII identité_1
format ASCII identité_2
format ASCII identité_3
format ASCII identité_4
...
fin du fichier
```

Le format ASCII d'une identité se compose:

- du nom de l'identité (un nom valide au sens IODE, c-à-d un nom de 10 caractères maximum (20 à partir de la version 6.01) -lettre ou chiffre- commencant toujours par une lettre -ou le caractère '_'- ,en majuscule) suivi
- de la forme LEC de l'identité entre guillemets.

Exemple

```
IDT1 "A + B + C"
IDT2 "ln( pi * (A - 1) / e) * ( 1 / (C + D))"
IDT3 "1 / IDT1"
```

9.5.1.4 FORMAT ASCII DES LISTES

Définition

Le format ASCII d'un fichiers de listes se compose successivement de la définition ASCII individuelle des listes. La fin du fichier signale la fin de la définition des commentaires.

```
début du fichier
format ASCII liste_1
format ASCII liste_2
format ASCII liste_3
format ASCII liste_4
...
fin du fichier
```

Le format ASCII d'une liste se compose:

- du nom de la liste (un nom valide au sens IODE, c-à-d un nom de 10 caractères maximum (20 à partir de la version 6.01) -lettre ou chiffre- commencant toujours par une lettre -ou le caractère '_'- ,en majuscule) suivi
- du texte de la liste entre guillemets.

Exemple

```
BLOCK "VAR3 VAR4"
LIST1 "VAR1 VAR2 $BLOCK"
LIST2 "TBL1 TBL2"
```

9.5.1.5 FORMAT ASCII DES SCALAIRES

Définition

Le format ASCII d'un fichiers de scalaires se compose successivement de la définition ASCII individuelle des scalaires. La fin du fichier signale la fin de la définition des commentaires.

```
début du fichier
format ASCII scal_1
format ASCII scal_2
format ASCII scal_3
format ASCII scal_4
...
fin du fichier
```

Le format ASCII d'un scalaire se compose:

- du nom du scalaire (un nom valide au sens IODE, c-à-d un nom de 10 caractères maximum (20 à partir de la version 6.01) -lettre ou chiffre- commencant toujours par une lettre -ou le caractère '_'- ,en minuscule) suivi
- de la valeur du scalaire, suivi
- du paramètre de relaxation, suivi des deux résultats de l'estimation du paramètre (écart type et test t) ou na (ces dernières valeurs n'on qu'un rôle d'information).

Exemple

```
c1 0.9 1 na
c2 1.4567 0.5 na
c3 -0.98765 0.8 na
```

9.5.1.6 FORMAT ASCII DES TABLEAUX

Définition

Attention: il s'agit bien de la définition de l'objet tableau et non de la définition de l'output tableau, celui-ci étant décrit par ailleurs.

Le format ASCII d'un fichier de tableaux de compose de la succession des définitions des tableaux individuels. La série des tableaux se termine à la fin du fichier.

```
début du fichier
format ASCII tableau_1
format ASCII tableau_2
format ASCII tableau_3
format ASCII tableau_4
...
fin du fichier
```

Le format ASCII d'un tableau se compose:

- d'un nom de tableau (un nom valide au sens IODE, c-à-d un nom de 10 caractères maximum (20 à partir de la version 6.01) -lettre ou chiffre- commencant toujours par une lettre -ou le caractère '_'- ,en majuscule)
- et d'un bloc de définition commencant par '{' et se terminant par '}'

```
NOM                             nom du tableau
{                               début de bloc
    .....                       bloc de définition
}                               fin de bloc
```

Le bloc de définition contient

- en première ligne le mot clé DIM suivi de la dimension du tableau. La dimension du tableau est 1 minimum. La première colonne est supposée être une colonne de titres -elle n'est pas répétée-, les colonnes suivantes sont répétées pour chaque période du sample. La dimension du tableau qui sera compilé et imprimé sera de (1 + (DIM-1) x nb. périodes du sample).
- les attributs graphiques :

```
BOX {0 | 1}
XGRID {0 | 1 | 2} : None, Minor, Major Grids
YGRID {0 | 1 | 2} : None, Minor, Major Grids
ALIGN {LEFT | Center | Right}
YMIN na ou valeur
YMAX na ou valeur
ZMIN na ou valeur
ZMAX na ou valeur
```

- en deuxième ligne, optionnellement, les diviseurs. La ligne des diviseurs est indiquée par le mot-clé DIV. Les diviseurs sont des formules en LEC (repérées par le mot clé LEC suivi de la forme LEC entre guillemets), valant LEC "1" par défaut. Le diviseur de la dernière colonne de la définition de tableau est reporté sur toutes les colonnes suivantes.
- successivements sur les lignes suivantes le contenu du tableau: des titres (TITLE), des lignes (LINE), des chaînes de caractères quelconques ("...") et des formes LEC (LEC "...."). Chaque ligne débute par le signe '-'.
- à n'importe quelle ligne (mais généralement en fin de tableau) le mot clé FILES qui sert à imprimer les références de fichiers
- à n'importe quelle ligne (mais généralement en fin de tableau) le mot clé MODES qui sert à imprimer la signification des signes utilisés dans le sample (growth rate, ...).
- à n'importe quelle ligne, des commentaires délimités par "/*" et "*/"
- pour chaque ligne, les informations graphiques :

```
GRLINE | GRBAR | GRSCATTER : type de graphique
RAXIS | LAXIS : axe de référence
```

```
NOM                                  nom du tableau
{                                    début de bloc
DIM 2                                dimension
DIV LEC "diviseur1" LEC "diviseur2"  diviseurs
- TITLE "titre principal"            ligne de titre
- TITLE "sous-titre"                 ligne de titre
- LINE                               ligne séparatrice
- " " LEFT "#s" LEFT                 ligne de périodes
- LINE                               ligne séparatrice
- "..." LEFT LEC "formule1"          ligne de cellules
- "..." LEFT LEC "formule2"              idem
- "" ""                                  idem
- LINE                               ligne séparatice
- FILES                              références des fichiers
- MODE                               signes du sample
/* END OF TABLE */                  commentaire
}                                    fin de bloc
```

La ligne des diviseurs se construit de la façon suivante

- la ligne débute par le mot clé DIV
- le mot clé DIV est suivi de cellules, autant par ligne que la dimension du tableau. Chaque cellule une forme LEC représentée entre guillemets, précédée du mot cle LEC (cette formule sera calculée pour chaque période du sample d'impression le résultat du calcul sera appliqué en diviseur des cellules à contenu numérique de la colonne correspondante).

Chaque cellule peut être suivie d'un mot clé indiquant la justification de la colonne du tableau à l'impression (les mots clés admis sont LEFT -option par défaut-, RIGHT, CENTER et DECIMAL) et/ou un type d'enrichissement typographique (sont admis les types BOLD, ITALIC, UNDERLINE et, NORMAL -

par défaut-). La justification et l'enrichissement typographique définis au niveu du diviseur de la colonne sont appliqués par défaut à toutes les cellules de la colonne.

Le "contenu" du tableau est construit de la façon suivante

- chaque ligne débute par le caractère '-'
- les lignes de titre sont désignées par le mot clé TITLE suivi entre guillemets de la chaîne de caractère du titre. La chaine de caractères peut être suivie d'un mot clé indiquant la justification du titre à l'impression (les mots clés admis sont LEFT -option par défaut-, RIGHT, CENTER et DECIMAL) et/ou un type d'enrichissement typographique (sont admis les types BOLD, ITALIC et, NORMAL -par défaut-). Le titre sera imprimé à travers toutes les cellules du tableau avec la justification et les enrichissement typographiques spécifiés.
- les lignes séparatrices sont désignées par le mot clé LINE sans paramètre. Une ligne transversale sera imprimée dans le tableau cet endroit.
- les lignes ordinaires sont constituées de cellules, autant par ligne que la dimension du tableau. Chaque cellule est soit une chaîne de caractères délimitée par des guillemets (cette chaîne sera imprimée telle quelle à la bonne place dans le tableau), soit une forme LEC représentée entre guillemets, précédée du mot cle LEC (cette formule sera calculée pour chaque période du sample d'impression et le résultat sera imprimé). Chaque cellule peut être suivie d'un ou plusieurs mot clé indiquant sa justification dans la colonne du tableau à l'impression (les mots clés admis sont LEFT -option par défaut-, RIGHT, CENTER et DECIMAL) et/ou un type d'enrichissement typographique (sont admis les types BOLD, ITALIC, UNDERLINE et, NORMAL -par défaut-)
- une cellule consituée d'une chaîne de caractères peut contenir une des macros suivantes: #s sera remplacé à l'impression par le sample courrant complet, #t sera remplacé à l'impression par la période courrante, et #f sera remplacé à l'impression par le nom du fichier traité.

```
exemple : "File #f - Per. #s"
```

Exemple

```
TABLE {
DIM 2
ENGLISH
BOX 0 AXIS 0 XGRID 0 YGRID 0
ALIGN LEFT
YMIN na
YMAX 1000.000000
ZMIN 2.000000
DIV LEC "1" LEFT ""
- TITLE "Titel" BOLD CENTER
- LINE
- "" "#s" CENTER  LAXIS GRLINE
- LINE
- "lijn A" LEFT LEC "A" DECIMAL  LAXIS GRLINE
- "lijn B" LEFT LEC "B" DECIMAL  LAXIS GRLINE
- "Totaal" LEFT LEC "A+B" DECIMAL  RAXIS GRBAR
/* END OF TABLE */
}
```

Résumé des mots clés

- { et } : délimiteurs de bloc
- " : délimiteur de chaîne de caractère (ou forme LEC)
- /* et */ : délimiteurs de commentaires
- DIM nb : dimension du tableau, nb := nombre entier positif
- DIV cellules : ligne de diviseurs, cellules := formes LEC
- LEFT, CENTER, RIGHT ou DECIMAL : justification de la cellule
- NORMAL, ITALIC, UNDERLINE ou BOLD : enrichissement typographique de la cellule
- TITLE "chaîne" : ligne de titre, chaîne := le titre à imprimer
- LINE : ligne séparatrice
- "chaîne" : chaîne := contenu non numérique d'une cellule ordinaire
- LEC "formule" : cellule à contenu numérique calculé par la formule LEC
- FILES : référence du/des fichier(s)
- MODE : signification des sigles imprimés par la macro #s
- #s, #t et #f : macros définissant l'impression du sample, de la période et du ou des fichier(s).

9.5.1.7 FORMAT ASCII DES VARIABLES

Définition

Le format ASCII d'un fichier de variables se compose de la façon suivante:

- sur la première ligne: le mot clé *sample* suivi *des périodes limites d'échantillonnage*
- successivement sur les lignes suivantes et à raison d'une par ligne, les variables en commencant par leur *nom* (un nom valide au sens IODE, c-à-d un nom de 10 caractères maximum (20 à partir de la version 6.01) -lettre ou chiffre- commencant toujours par une lettre -ou le caractère '_'- ,en majuscule) suivi des *valeurs (ou na)* pour chaque période. Chaque élément (nom et valeurs) sont séparés par un blanc.
- la fin de fichier signale la fin de définition des variables.

mot(s) clés

Un seul mot clé est utilisé:

```
sample yyyyPppp  yyyyPppp
                1         2
```

La définition des périodes limites doit répondre à la syntaxe suivante:

```
yyyyPppp    yyyy est l'année
            P est la périodicité
                P := Y (yearly), Q (quaterly), M (monthly), W (weekly)
            ppp est la période dans l'année
```

Exemple

```
sample 1970Y1 1996Y1
A0G 621800 644700 805700 ... 874900 939000 1.0538e+06 1.1965e+06
A0GBNB 49500 40800 39600 ... 42900 46100 48400 65500 88900 108600
 ...
A0GCCP 57300 57200 66000 ... 63400 65700 69400 76800 79900 83700 80900
A0GFMI 12400 2400 6100 7300 ... 6600 4700 1500 4100 19500 19800 37000
```

9.5.2 Formats DIF de IODE

Deux formats DIF différents sont exploitables via XODE. Le premier concerne n'importe quel programme (LOTUS, EXCEL, etc) capable de fournir un output DIF. Le format du tableau exporté en DIF doit évidemment répondre à des règles, sans quoi XODE ne saurait comment interpréter le contenu. La première ligne doit contenir les périodes, les suivantes le code d'une série suivie des valeurs.

On trouve donc ci-dessous une description du tableau comme il est vu à l'écran en LOTUS par exemple.

DIF (tel que visible en 123)

```
      CODE 1980Y1 1981Y1 1982Y1 1983Y1 ....
      A    1      1       2      4
      B    1      1       2      5
      ...
```

Le second format est généré par le logiciel de gestion de tableaux de Belgostat et se présente de la façon suivante en LOTUS lorsqu'une exportation est effectuée :

Belgostat DIF (tel que visible en 123)

```
123456789012345 1       1980    1             456900     1       1
(series nb)     (freq)  (year)  (per in year) (value)  (status)  (free)
   ........
```

9.5.3 Format PRN d'AREMOS

Ce format assure l'interface avec le logiciel AREMOS.

Le fichier de variables se présente comme une répétition de lignes au format suivant:

```
          "VARNAME" val1 val2 ....
          ...
```

9.6 LEXIQUE

Cette partie du manuel reprend la définition de quelques sujets souvent abordés dans le texte. Les

explications concernant l'utilisation des champs, des écrans, etc, y sont également traitées.

- Le format A2M
- Champ DIR
- Champ EDITOR
- Champ menu
- Editeur MMT
- Les "listes"
- NA (Not Available)
- Tableau déroulant

9.6.1 A2M

A2M est un format d'échange de documents entre logiciels développé par et pour les besoins du Bureau fédéral du Plan.

Un texte au format A2M contient des caractères ASCII du jeu étendu accentué et éventuellement des commandes et des instructions d'enrichissement typographique. La découpe du texte en paragraphe, les sauts de lignes et les lignes blanches revêtent également une signification particulière. Ces instructions et commandes sont toutes optionnelles, tout texte ASCII étant par définition au format A2M.

Le langage d'interface A2M (Ascii TO Mif) a été développé initialement afin de pouvoir taper ou construire un document en utilisant un simple éditeur ASCII, et d'obtenir un texte correctement mis en page sur FRAME, sans intervention au niveau de FRAME, via son format d'interface MIF. Les premières versions du filtre SCR3_A2M effectuaient la convertion A2M vers MIF.

L'idée a ensuite été élargie, A2M est devenu un format d'interface généré par plusieurs logiciels produits par la Cellule Informatique du Bureau fédéral du Plan (Iode, Stock, Matrix, Lex, ...). Un format de tableaux et de graphiques a été implémenté. Des modules ont été développés pour transformer le format A2M en d'autres formats d'échange (RTF, CSV, HTML en autre) ou directement en format d'impression Windows.

9.6.2 Champ DIR

Un champ de type DIR est un champ de saisie permettant la recherche d'un nom de fichier en fonction d'un masque de saisie. On commence par encoder dans le champ le masque de fichier exactement comme dans un champ normal de texte (p.ex. "*.var").

La touche ENTER affiche alors la liste des fichiers correspondant au masque de saisie. Les sous-répertoires sont également mentionnés. Le nom du fichier désiré peut alors être sélectionné à l'aide des touches fléchées et de sauts de pages haut et bas. Le nom choisi est validé par la touche ENTER. Ce nom s'inscrit dans la page de saisie à la place du masque de recherche. Si le nom sélectionné est un nom de répertoire, la recherche se poursuit dans ce répertoire. Ces champs fonctionnent par ailleurs comme un champ de texte normal.

Si aucun nom affiché dans le menu ne doit être sélectionné, la touche ESCAPE permet de quitter le menu sans exporter de nom dans le champ.

Dans le menu, on trouve pour chaque fichier plusieurs informations comme sa taille, la date de dernière modification, ses propriétés d'accès, ...

9.6.3 Champ EDITOR

Un champ EDITOR est un champ de saisie dont la dimension n'est pas limitée. On entre en édition du champ avec la touche ENTER lorsque le champ est sélectionné (c'est-à-dire apparaît en vidéo inverse). On quitte le champ avec la touche ESCAPE. Lorsqu'on est en édition, outre les touches fléchées, les touches de saut de page et les touches home et end, les séquences suivantes sont admises:

Les séquences de touches de l'éditeur MMT sont valides, pour autant que leur effet se limite à une seule ligne.

La position de la fenêtre visible par rapport au contenu du champ peut être représentée par une "scroll bar".

Si on dispose d'une souris, celle-ci offre des facilités pour le marquage (bouton de gauche) et la fonction copie/collage (bouton de droite).

9.6.4 Champ menu

Un champ MENU est un champ de saisie dont le contenu est limité aux options d'un menu. Le menu est accessible avec la touche ENTER, on s'y déplace avec les touches fléchées et de saut de page haut et bas. L'option sélectionnée est validée avec la touche ENTER, le texte correspondant s'inscrit dans la page de saisie. On peut également sélectionner le texte de l'option en tapant sa première lettre ou un espace pour passer d'option en option.

9.6.5 Editeur MMT

L'éditeur MMT est un éditeur ASCII qui permet d'éditer simultanément plusieurs textes dans plusieurs fenêtres visibles à l'écran. Il a été développé par et pour des développeurs en vue de rendre performantes les opérations d'édition et de modification des fichiers de code source, comme des rapports de IODE.

L'éditeur MMT est utilisé dans les champs de type éditeur ainsi que dans la fonction d'édition des rapports.

Parmi les options intéressantes de cet éditeur figure la possibilité de faire du "cut and paste" d'un champ à l'autre d'un programme ou d'un champ vers un texte, ou d'un texte vers un champ.

Une description complète de l'éditeur MMT figure dans les annexes.

9.6.6 Les listes

Une liste est l'un des sept objets manipulés par le logiciel IODE. Elle est identifiée par un nom de 10 caractères (20 à partir de la version 6.01) majuscules maximum commençant toujours par une lettre (ou le caractère '_').

Certaines fonctions acceptent comme paramètres des listes. Il s'agit alors généralement d'une énumération dans un champ EDITOR. Chaque élément de la liste peut être un objet quelconque (y compris une liste identifiée par le préfixe $), les noms des objets étant séparés par un blanc, une virgule, un point-virgule ou un saut de ligne.

9.6.7 NA (NOT AVAILABLE)

La valeur NA est utilisée dans les variables et dans les scalaires pour indiquer une valeur inconnue. Plusieurs raisons peuvent donner lieu à cette valeur :

- lors de la création d'une série, ses valeurs sont NA
- lors du calcul d'une série par identité, une valeur NA peut provenir d'une erreur de calcul ou d'une information manquante dans les variables de référence
- lors d'une simulation, une valeur NA peut venir d'une faute de calcul (log(-1), A/0, etc).

La valeur NA est représentée par deux tirets (--). Elle peut être introduite de cette façon par l'utilisateur dans l'éditeur de variables.

9.6.8 TABLEAU DÉROULANT OU TABLEAU D'ÉDITION

Un tableau déroulant (ou tableau SCROLL) est un écran permettant d'afficher un tableau du type de ceux des tableurs ordinaires et selon le cas d'éditer le contenu des cellules. Le tableau peut être déplacé sur l'écran, inversé, agrandi ou réduit, etc.

Lorsque le tableau s'affiche, les touches suivantes peuvent être utilisées (en plus des touches de fonction globales du programme):

- ESCAPE : quitte l'édition
- UP, DOWN : déplace le curseur vers la cellule supérieure ou inférieure
- PGUP, PGDN : déplace le curseur d'un écran vers le haut ou le bas
- LEFT, RIGHT : déplace le curseur vers la cellule de gauche ou de droite
- TAB et Shift-TAB : déplace le curseur d'un écran vers la droite ou la gauche
- HOME : place le curseur en première colonne de la ligne courante
- CTRL + HOME : place le curseur en première ligne de la colonne courante
- END : place le curseur en dernière colonne de la ligne courante
- CTRL + END : place le curseur en dernière ligne de la colonne courante
- ENTER : édite la cellule courante (si autorisé)
- Ctrl-R : inverse le tableau (lignes vs colonnes)
- Ctrl-O : déplace le tableau dans les limites de l'écran suivant les touches fléchées utilisées ensuite. L'édition reprend après que ENTER ait été pressé.
- Ctrl-Z : change la taille du tableau dans les limites de l'écran suivant les touches fléchées utilisées ensuite. L'édition reprend après que ENTER ait été pressé.
- Lettre ou chiffre : positionne le curseur sur la prochaine ligne dont le titre commence par le caractère pressé.

9.7 L'ÉDITEUR MMT

Ce chapitre décrit les fonctionnalités de l'éditeur MMT. Cet éditeur est utilisé dans IODE pour l'édition des fichiers (rapports, fichiers ASCII, fichiers A2M, etc).

C'est également MMT qui est exploité pour la saisie des équations et de façon plus générale pour tous les champs de taille variable dans les écrans de IODE.

Ce chapitre se compose des sections suivantes :

- Généralités sur MMT
- Fonctionnalités de MMT
- La commande de lancement de MMT
- Les touches de fonction de MMT
- La souris dans MMT
- Les champs EDITOR

9.7.1 Généralités sur MMT

L'éditeur MMT est un éditeur ASCII qui permet d'éditer simultanément plusieurs textes dans plusieurs fenêtres visibles à l'écran. Il a été développé par et pour des développeurs en vue de rendre performantes les opérations d'édition et de modification des fichiers de code source.

Il gère une souris compatible Microsoft sans nécessité d'installer un driver de souris.

Il gère en outre (en DOS) la mémoire étendue (type EMS) si elle est présente, ainsi que le swapping sur disque si nécessaire. Des textes ASCII de grandes tailles peuvent donc être chargés.

MMT est développé en C, sous DOS et UNIX, en utilisant les librairies du 4GL SCR/AL1. MMT fait partie intégrante de SCR/AL1.

Auteurs: Jean-Marc Paul et Hubert Castelain, Cellule Informatique du Bureau du Plan.

Mars 1993.

9.7.2 Fonctionnalités de MMT

L'éditeur MMT offre les principales fonctionnalités suivantes:

- L'éditeur est piloté soit avec la souris soit avec des touches de fonction.
- Copy, Cut and Paste d'un bloc (rectangle, lignes ou texte) au sein d'un texte et d'un texte à l'autre. Ces fonctions sont étendues à tous les champs EDITOR d'une application SCR/AL1 : copy, cut and paste d'un champ à l'autre, d'un champ vers un texte et d'un texte vers un champ. Un "clipboard" permet de visualiser à tout endroit le bloc en cours de transfert.
- Gestion de fenêtre: les textes peuvent se présenter en cascade, en fenêtres juxtaposées ou en plein écran. Les fenêtres peuvent être déplacées et changées de tailles, ou encore "zoomées", tout cela avec la souris ou le clavier..
- Gestion de la souris: marquage de bloc, déplacement du curseur, déplacement du "focus" sur un autre texte, copy-cut-paste d'un bloc marqué, etc... Le driver de souris utilisé est le driver interne de SCR/AL1 qui répond à une souris compatible Microsoft.
- Edition de texte: outre le marquage et le déplacement de blocs de texte (en insersion ou en superposition), les principales fonctions suivantes sont actives: recherche d'un texte, remplacement d'un texte, remplissage d'un bloc, déplacement dans tout l'écran, reconstruction de ligne, visualisation de background, suppression ou insertion de lignes, effacement de début ou fin de ligne, ...
- Tracé de cadres (caractères graphiques ou non)
- Aide en ligne, "screen save" automatique, etc...

9.7.3 La commande de lancement de MMT

La commande suivante permet d'éditer (ou de créer s'il est inexistant) un ou plusieurs (100 maximum) texte(s) avec MMT :

```
MMT file1 [file2 file3 ...]
```

Les touches de fonction ALT + Q et ALT + X permettent de quitter le programme.

Si plusieurs textes sont ouverts, le texte actif est celui dont le titre est "rétro-éclairé".

9.7.4 Les touches de fonction de MMT

Remarques:

- les touches de fonctions sont susceptibles d'être modifiées au gré des versions de MMT. Un résumé des touches de fonctions est accessible en cours d'exécution de MMT soit via la touche F1 soit via le menu dynamique du programme (ALT + M)
- toutes les fonctions sont accessibles dans mmt par menu. Les menus sont activés par la touche ALT+M.
- dans le cas des champs EDITOR d'une application, les touches de l'application ont priorité sur les touches de fonction de MMT. Dans ce cas, il est nécessaire d'accéder aux fonctionnalités de MMT via les menus ou le bouton droit de la souris.
- sauf quelques exceptions, les touches de l'éditeur MMT sont compatibles avec celles de l'éditeur MT.

Dans les paragraphes qui suivent, les abréviations suivantes sont utilisées:

- A_ touche ALT enfoncée
- C_ touche CTRL enfoncée
- S_ touche SHIFT enfoncée
- CSR touche curseur

Les touches de fonction sont groupées par sujets dans les sections qui suivent :

- Ouvrir/quitter/sauver
- Déplacement dans le texte
- Opérations de bloc
- Opérations sur une ligne
- Opérations sur le texte
- Gestion du multi-fenêtrage
- Touches diverses
- Résumé des touches de fonction

La touche F1 affiche ce présent manuel en plus d'un résumé des touches fonctions utiles.

9.7.4.1 Ouvrir/quitter/sauver

- F4 charger un fichier
- A_F1 sauver et quitter le fichier courant
- A_F2 sauver tous les textes en édition
- A_F3 changer le nom du fichier
- A_Q quitter le texte courant
- A_X quitter tous les textes
- A_W sauver le fichier courant
- ESCAPE quitter le texte courant

9.7.4.2 Déplacement dans le texte

- CSR_LEFT curseur à gauche
- CSR_RIGHT curseur à droite
- CSR_UP curseur au dessus
- CSR_DOWN curseur en dessous
- CSR_HOME curseur en début de ligne
- CSR_END curseur en fin de ligne
- CSR_PG_UP curseur un écran plus haut
- CSR_PG_DN curseur un écran plus bas
- C_CSR_HOME curseur en début de texte
- C_CSR_END curseur en fin de texte
- C_CSR_LEFT curseur sur le mot précédent
- C_CSR_RIGHT curseur sur le mot suivant
- TAB curseur quatre caractères plus loin
- INSERT mode insertion/remplacement
- DELETE effacer le caractère courant
- BACKSPACE effacer le caractère précédent
- C_BACKSPACE effacer le mot courant
- ENTER passer à la ligne suivante
- Shift-F1 marque la position courante (drapeau)
- Shift-F2 retourne au dernier drapeau

9.7.4.3 Opérations de bloc

- C_Y Copy to clipboard
- C_P Paste clipboard
- A_O Overlay clipboard
- A_C Copy block
- A_D Delete (cut) block
- A_B Mark Block
- A_L Mark Lines
- A_T Mark Text
- A_U Unmark block
- A_S Show clipboard
- A_F5 Lower block
- A_F6 Upper block
- A_F7 Fill block
- F3 Outline block (box)
- F5 Shift left (lines)
- F6 Shift right (lines)

9.7.4.4 Opérations sur une ligne

- A_G Goto line
- A_I Insert line
- A_J Join next line
- A_K Kill cur.line
- A_N New line(split)
- A_Z Del.end of line
- C_J Insert Line
- C_K Del end of line
- C_U Del.beg.of line

9.7.4.5 Opérations sur le texte

- C_F, Find text
- C_R, Replace text
- F2, Underline text
- A_I, Insert line
- A_K, Kill line
- A_N, New line (split)
- A_J, Join line
- A_G, Go to line
- A_Z, Delete end of line
- C_U, Delete beg.of line
- C_F8, Right margin
- C_F9, Activate margin
- A_P, Reflow paragraph

9.7.4.6 Gestion du multi-fenêtrage

- A_F8, Cascade
- A_F9, Little box
- A_F10, Full screen
- C_Z, Zoom text
- A_E, Edit next text
- A_S, Show clipboard
- C_V, Redisplay
- F7, Move box
- F8, Resize box
- C_B, Show dos screen
- C_A, Screen colors

9.7.4.7 Touches diverses

- F1, MMT on line help
- F10, SCR4/AL1 topics
- F7 , Edition d'un hypertext
- C_F1, System Shell
- C_B, Show dos screen
- C_A, Screen colors
- A_A, About MMT
- A_M, Menu déroulant de MMT

9.7.4.8 Résumé des touches de fonction

Les touches de fonctions sont susceptibles d'être modifiées au gré des versions de MMT. Un résumé des touches de fonctions est accessible en cours d'exécution de MMT soit via la touche F1 soit via le

menu dynamique du programme (ALT + M).

9.7.5 La souris dans MMT

Les opérations gérées par la souris sont les suivantes:

Le bouton de gauche:

- La souris dans le texte
- La souris sur le scrollbar
- La souris sur le cadre
- Le bouton de droite de la souris

9.7.5.1 La souris dans le texte

Dans le texte, la souris permet de:

- Se positionner à un endroit donné: "clic" à cet endroit
- Marquer un bloc: "clic" au début de bloc, tirer en laissant le bouton enfoncé.

9.7.5.2 La souris sur le scrollbar

On peut soit:

- Se positionner à un endroit du texte: "clic" dans le scrollbar
- Faire défiler le texte: "clic" dans le scrollbar, tirer en laissant le bouton
- Faire défiler le texte ligne par ligne vers le haut ou vers le bas: "clic" sur les flèches haut et bas du scrollbar.

9.7.5.3 La souris sur le cadre

Le cadre d'une fenêtre contient des caractères spéciaux entre crochets : un carré en haut à gauche du cadre, une liste à côté du carré , une double flèche en haut à droite du cadre et un double coin en bas à droite du cadre.

Un clic sur ces caractères spéciaux ou sur le cadre aura les conséquences suivantes:

- Le carré en haut à gauche: quitter le texte courant
- La liste : affichage d'un menu principal
- La double flèche : zoom
- Le point d'interrogation: aide en ligne
- Le coin inférieur droit : modification de la taille de la fenêtre (tirer le coin avec la souris en laissant le bouton enfoncé). &EN Le cadre : déplacement de la fenêtre (tirer le cadre avec la souris en laissant le bouton enfoncé).

9.7.5.4 Le bouton de droite de la souris

Le bouton de droite affiche un menu qui offre les options suivantes:

- CUT (to clipboard) équivallent à la touche de fonction (a-D)
- COPY (to clipboard) équivallent à la touche de fonction (c-Y)
- PASTE (from clipboard) équivallent à la touche de fonction (c-P)
- SHOW clipboard équivallent à la touche de fonction (a-S)
- UnMark équivallent à la touche de fonction (a-U)
- Mark Block équivallent à la touche de fonction (a-B)
- Mark Lines équivallent à la touche de fonction (a-L)
- Mark Text équivallent à la touche de fonction (a-T)
- Fill block équivallent à la touche de fonction (a-F7)
- Outline block équivallent à la touche de fonction (F3)

9.7.6 Les champs EDITOR

Les champs de saisie EDITOR sont en réalité des textes MMT. L'édition des ces champs offre donc toutes les possibilités de l'édition d'un texte MMT. Toutes les fonction y sont actives sauf: F4 (chargement d'un fichier) et les opérations de déplacement et de modification de taille de la fenêtre (le zoom reste cependant actif).

En particulier, les champs EDITOR supportent toutes les fonctions de copie via le clipboard. Le clipboard étant connu de l'ensemble de l'application SCR en cours d'exploitation, il est possible de copier un champ dans un autre, un texte MMT dans un champ, un champ dans un texte, etc...

dans le cas des champs EDITOR d'une application, les touches de l'application ont priorité sur les touches de fonction de MMT. Dans ce cas, il est nécessaire d'accéder aux fonctionnalités de MMT via les menus ou le bouton droit de la souris.

9.8 Génération d'un fichier d'aide Windows

IODE permet de générer des outputs sous différents formats : fichiers RTF pour Winword, MIF pour Frame, HTML pour les browser Internet, CSV pour Excel. D'autres formats pourront encore s'ajouter dans les versions futures.

Le format RTF (Rich Text Format) est particulier en ce sens qu'il a un double usage: en plus des nombreux programmes de traitement de texte qui sont capables d'interpréter ce format, le programme d'aide de Windows, `Winhelp.exe` (ou `WinHlp32.exe`), peut, moyennant une compilation particulière, exploiter ce langage.

Il est donc possible de construire un fichier d'aide au format Windows, contenant par exemple des tableaux et des équations de IODE. Fichier aisément distribuable sur toute machine dotée de Windows 9x ou de Windows NT.

La méthode à suivre pour construire et distribuer des fichiers d'aide Windows consiste en trois éta-

pes:

- la création d'un fichier hiérarchique RTF-Help
- la compilation des fichiers RTF-Help
- la distribution des fichiers Help Windows

Pour conclure, un exemple de création d'un fichier d'aide Windows est présenté. Cet exemple imprime toute la définition d'un projet : équations, variables, etc.

9.8.1 Création d'un fichier hiérarchique RTF-Help

Toute impression de IODE dans un fichier a2m ou directement dans un fichier RTF peut être la source d'un fichier d'aide Windows. Cela peut se faire aussi bien à partir des points du Menu Print que d'un Rapport.

Pour obtenir un fichier Rtf pour Aide de Windows, il faut simplement cocher la case HCW dans le panneau RTF des "Print Preferences" et choisir comme destination des impressions un fichier RTF (Print Setup).

Cependant, il est préférable pour la lisibilité du fichier d'aide de décomposer son fichier en sujets et sous-sujets.

Certaines impressions de IODE génèrent automatiquement des sous-sujets : c'est le cas des impressions:

- de tableaux
- de définitions de tableaux
- de définitions d'équations avec résultats d'estimation

Dans tous les autres cas, l'impression a lieu dans un sujet.

Pour ajouter une hiérarchie à un document, il faudra, soit directement en langage A2M, soit à travers les commandes de rapport **`$PrintRtfTopic`** et **`$PrintRtfLevel`**, indiquer trois informations :

- où la découpe doit se faire
- le titre du topic
- le niveau du topic dans la hiérarchie

Par exemple, si on réalise un fichier d'aide constitué de tableaux on pourra indiquer :

```
$PrintRtfLevel 0

$PrintRtfTopic Principales hypothèses de la projection
$PrintRtfLevel +
$PrintNbDec 1
$PrintTbl %NIV0% HYPE
$PrintTbl %NIV0% HYPFP
$PrintTbl %NIV0% HYPSZ DSB

$PrintRtfLevel -
$PrintRtfTopic Résultats Macroéconomiques
$PrintRtfLevel +
$PrintNbDec 2
$PrintTbl %GRT0% RESL00R
$PrintNbDec 1
$PrintTbl %NIV0% RESL03
```

$PrintRtfLevel permet de déterminer le niveau hiérarchiques des sujets suivants :

- "Principales hypothèses ..." est au niveau 0 (supérieur)
- Chaque tableau imprimé est à un niveau 1 (**$PrintRtfLevel** +)
- On revient ensuite au niveau 0 (**$PrintRtfLevel** -)

Le titre des topics est constitué du paramètre de la fonction **$PrintRtfTopic**.

9.8.2 Compilation des fichiers RTF-Help

Lorsqu'un fichier RTF-Help est construit à l'aide de IODE ou du programme **a2m.exe**, il faut encore le compiler pour obtenir un fichier exploitable par le programme **Winhelp.exe**.

En fait, trois fichiers sont générés par IODE ou A2M:

- **filename.rtf** : texte de l'aide
- **filename.hpj** : fichier de définition du projet d'aide
- **filename.cnt** : fichier contenant la table de matières

La compilation nécessite l'installation du programme **HCW.EXE**, distribué avec la plupart des outils de programmation de Microsoft ou de Borland.

La commande de compilation est :

```
hcw /C/A nom_du_fichier[.hpj]
```

Comme résultat, on obtient le fichier **filename.hlp**.

9.8.3 Distribution des fichiers Help Windows

Pour fournir une information complète, deux fichiers doivent être distribués:

- **filename.cnt** : fichier contenant la table de matières
- **filename.hlp** : fichier contenant le texte de l'aide

On peut regrouper ces deux fichiers dans une archive par exemple à l'aide de la commande **pak.exe** ou **pkzip.exe**.

```
pak a /EXE filename.exe filename.hlp filename.cnt
pkzip -a filename.zip filename.hlp filename.cnt
```

9.8.4 Exemple de création d'un fichier d'aide Windows

L'exemple suivant imprime la définition de tous les objets présentes dans les WS courants.

```
$PrintDest test.rtf RTF
$PrintRtfLevel 0
$PrintRtfTitle Modèle MODTRIM
$PrintRtfCopyright Bureau fédéral du Plan (c) 1998
$PrintFont Helvetica 9 1
$PrintTableFont Times 8 2

$PrintObjInfos 2
$PrintRtfTopic Equations
$PrintRtfLevel ++
```

```
$PrintObjDefEqs
$PrintRtfLevel --

$PrintRtfTopic Comments
$PrintRtfLevel ++
$PrintObjDefCmt
$PrintRtfLevel --

$PrintRtfTopic Identities
$PrintRtfLevel ++
$PrintObjDefIdt
$PrintRtfLevel --

$PrintRtfTopic Lists
$PrintRtfLevel ++
$PrintObjDefLst
$PrintRtfLevel --

$PrintRtfTopic Scalars
$PrintRtfLevel ++
$PrintObjDefScl
$PrintRtfLevel --

$PrintRtfTopic Tables
$PrintRtfLevel ++
$PrintObjDefTbl
$PrintRtfLevel --

$PrintRtfTopic Variables
$PrintRtfLevel ++
$PrintObjDefVar
$PrintRtfLevel --

$PrintRtfLevel 0
```

Il reste à traduire le fichier **`test.rtf`** en fichier .hlp :

```
HCW /C/A test
```

et enfin à le tester :

```
Winhelp test
```

Pour le distribuer, on fera par exemple :

```
pak a /EXE test.exe test.hlp test.cnt
```

9.9 Syntaxe des fichiers a2m

La syntaxe reprise ci-dessous correspond à la version d'avril 97 du langage a2m. Il s'agit d'une refonte importante du langage, compatible avec la version antérieure, mais contenant de nombreux nouveaux tags, parmi lesquels des commandes de définition de tableaux, de graphiques et de sujets pour les fichiers d'aide de Windows 95, la prise en compte de la couleur, etc.

Un fichier en format A2M peut être traduit en différents formats :

- GDI, format d'impression directe sous Windows
- RTF, interprétable par Word, WordPerfect, etc
- MIF, langage structuré des documents Frame Maker
- RTF-HLP, source d'un fichier d'aide de Windows 95
- HTML, format des documents Internet
- CSV, format simplifié de tables de calcul (Excel)

Selon la destination de l'impression, des tags seront soit interprétés différemment, soit inopérants. Ainsi par exemple, les images ne peuvent être incorporées dans des fichiers CSV. De même, les sujets d'aide n'ont de signification que dans le cadre des fichiers d'aide.

La syntaxe des fichiers A2M est structurée comme suit :

- Caractères réservés
- Directives
- Blancs et sauts de lignes
- Paragraphes
- Commandes de mise en page
- Définition des tableaux
- Définition des graphiques
- Insertion d'images
- Enrichissements typographiques
- Définition des topics d'aide
- Hyperliens

9.9.1 Caractères réservés

Quatre caractères ont une signification spéciale dans un fichier A2M. Chacun peut être redéfini.

Ces caractères sont :

- le point(.) qui débute une ligne de commande. Ce caractère n'est pas interprété s'il n'est pas le premier d'une ligne.
- le backslash (\) qui débute une séquence d'enrichissement typographique ou un caractère spécial. Il peut se trouver n'importe où dans le texte.
- le caractère de début d'une macro (define). Ce caractère est interprété uniquement si le mot qui le suit est une macro. Si ce n'est pas le cas, il est imprimé comme tel.
- l'ampersand (&) qui indique le début d'une cellule dans un tableau.
- le signe # (NUMBER) qui indique le début d'une directive. Il n'est interprété qu'en début de ligne.

Les quatre premiers caractères peuvent être définis pas une des commandes suivantes:

```
.cmd c    (Default .)
.esc c    (Default \)
.def c    (Default &)
.sep c    (Default &)
```

Exemple

```
.cmd .
.esc $
.defch &
.sep @
```

9.9.1.1 COMMANDE .cmd

Cette commande permet de définir le caractère qui débute une commande dans la suite du texte. Par défaut, ce caractère est le point.

Ce caractère est interprété uniquement s'il est le premier d'une ligne. Pour imprimer ce caractère en début de ligne, il faut le précéder du caractère esc (voir .esc).

SYNTAXE

```
.cmd ch
```

EXEMPLE

```
.cmd
```

9.9.1.2 COMMANDE .esc

Cette commande permet de définir le caractère qui débute une séquence d'attribut typographiques, d'insertion d'objet ou d'hyperlien dans la suite du texte. Par défaut, ce caractère est le BACKSLASH (\).

Ce caractère est interprété où qu'il se trouve dans le texte. Pour imprimer ce caractère, il faut le doubler.

SYNTAXE

```
.esc ch
```

EXEMPLE

```
.esc &
```

9.9.1.3 COMMANDE .def

Cette commande permet de définir le caractère qui débute une référence à une macro dans la suite du texte. Par défaut, ce caractère est le AMPERSAND (&).

Ce caractère est interprété où qu'il se trouve dans le texte. Pour imprimer ce caractère, il faut le faire suivre d'un point-virgule (;).

Si la macro dont le nom suit n'existe pas, ou s'il n'y a pas de macro qui suit ce caractère, il est rendu à l'impression tel quel, de même que le nom de la macro.

SYNTAXE

```
.def ch
```

EXEMPLE

```
.def $
.define LST par1, par2, par3, par4
Liste : $LST
Liste : $ LST
Liste : $;LST
Liste : $LIST
```

sera interprété comme suit:

```
Liste : par1, par2, par3, par4
Liste : $ LST
Liste : $LST
Liste : $LIST
```

9.9.1.4 COMMANDE .sep

Cette commande indique le caractère qui débute une cellule dans la définition d'un tableau. Par défaut, ce caractère est le AMPERSAND (&).

EXEMPLE

```
.sep @
.tb 3
@1CTitre 1  @1CTitre 2   @1CTitre 3
@1LCell 2,1 @1LCell 2,2  @1LCell 2,3
.te
```

9.9.2 DIRECTIVES

Les directives suivantes sont interprétées :

- #include "filename"
- #define MacroName MacroDefinition
- #undef MacroName
- #ifdef MacroName
- #else
- #endif

Pour qu'une telle ligne ne soit pas interprétée, il suffit de placer un point-virgule (;) après le caractère

9.9.2.1 LA DIRECTIVE #include

Cette command permet d'include un fichier dans le fichier en cours de lecture. Tout se passe comme si le contenu du fichier inclus était intégré dans le fichier courant.

SYNTAXE

```
#include "filename"
```

9.9.2.2 LA DIRECTIVE #define

Cette commande permet de définir des raccourcis d'écriture (macros).

Le texte d'une macro peut s'étendre sur plusieurs lignes: il suffit pour cela que le dernier caractère de la ligne de définition (et des suivantes au besoin) soit l'AMPERSAND (&).

SYNTAXE

```
#define MACRONAME texte libre
```

Le nom des macros peuvent être quelconque mais ne doit pas contenir de caractères autres qu'alphanumériques ou de soulignement (_).

L'utilisation des macros se fait à l'aide du caractère & suivi du nom de la macro :

```
#define UINT  unsigned short
...
&UINT i;
```

Pour annuler l'interprétation d'une macro, il suffit de placer un point-virgule (;) après le caractère &.

Le séparateur qui suit le nom de la macro peut être quelconque. S'il s'agit d'un point-virgule (;), ce caractère est consommé.

```
#define TX unsigned short
&;TX x,y
&TX x,y
&TXx,y
```

Donne à l'interprétation

```
&TX x,y
unsigned short x,y
unsigned shortx,y
```

Pour prolonger le texte d'une macro sur plusieurs lignes, il suffit de placer en fin de ligne le caractère #. La ligne suivante ne sera pas interprétée (sauf en ce qui concerne les "sous-macros" utilisées) et fera partie de la macro.

Pour placer le caractère # en fin de macro, il suffit de le doubler en fin de ligne.

```
#define PAR1 .par1 par_1#

#define PAR2 .par1 par_2##
&PAR1 texte1
&PAR2 texte2
```

Donne à l'interprétation

```
.par1 par_1
 texte1
.par1 par2# texte2
```

Si une macro n'est pas définie, le texte du fichier n'est pas interprété. Ainsi:

```
#define A AVAL
&A
&B
```

Donne à l'interprétation

```
AVAL
&B
```

9.9.2.3 La directive #undef

Cette directive supprime la définition d'une macro.

Syntaxe

```
#undef MACRONAME
```

9.9.2.4 La directive #ifdef

Cette directive permet de ne traiter une partie du texte que si une macro est définie.

Syntaxe

```
#ifdef MACRONAME
```

Si MACRONAME n'est pas définie, la suite du texte, jusqu'au prochain #endif ou #else n'est pas traitée.

9.9.2.5 La directive #else

Cette directive permet de ne traiter la partie suivante du texte que si le précédent #ifdef annule le texte situé après le #ifdef.

SYNTAXE

```
#else
```

La portée du #else s'étend jusqu'au #endif suivant.

9.9.2.6 LA DIRECTIVE #endif

Termine une suite #ifdef .. #else .. #endif.

SYNTAXE

```
#endif
```

9.9.3 BLANCS ET SAUTS DE LIGNES

Toute suite d'espaces dans le texte est normalement ramenée à un espace unique. On peut modifier ce comportement par la commande **bl_on**. Pour revenir au traitement normal, on utilisera la commande **bl_off**.

Ces commandes peuvent se trouver n'importe où dans le fichier. Elles doivent être collées à la marge.

```
.bl_on      ---> conserver les doubles blancs
.bl_off     ---> supprimer les blancs excédentaires
```

Un paragraphe est défini comme une suite de lignes de texte contiguës. Une ligne de commande ou une ligne blanche indique la fin d'un paragraphe.

Le paragraphe est reformatté en fonction de la destination et les sauts de ligne ne se situent pas nécessairement à la même place que dans la source. La commande **lf_on** permet de conserver les sauts de lignes. La commande **lf_off** revient au traitement par défaut.

```
.lf_on       ---> conserver les sauts de lignes
.lf_off      ---> reformatter le paragraphe
```

Exemple

Le texte suivant

```
.bl_on
.lf_on
La cigalle ayant chanté
    tout l'été,
        se trouva fort dépourvue
            quand la bise fut venue !
```

est interprété comme suit :

```
La cigalle ayant chanté
    tout l'été,
        se trouva fort dépourvue
            quand la bise fut venue !
```

Par contre,

```
.bl_off
.lf_off
La cigalle ayant chanté
    tout l'été,
        se trouva fort dépourvue
            quand la bise fut venue !
```

est interprété comme suit :

```
La cigalle ayant chanté tout l'été, se trouva fort dépourvue quand la bise fut venue !
```

9.9.4 PARAGRAPHES

Lors de la génération du fichier destination, les propriétés des paragraphes influencent la présentation.

De nouveaux paragraphes peuvent être crées dans le cours du texte.Les paragraphes peuvent être (re-)définis à n'importe quel endroit du fichier. Les modifications prennent cours après la commande.

Deux commandes permettent de fixer le tag des paragraphes suivants. Un commande est associée à la (re-)définition des paragraphes.

```
.par partag
.par1 partag
.pardef parm=val ...
```

Les paragraphes suivants sont définis par défaut:

Titres

```
.pardef tit_7 copy=tit_3 html=5 number=5
.pardef tit_6 copy=tit_3 html=5 number=5
.pardef tit_5 copy=tit_3 html=5 number=5
.pardef tit_4 copy=tit_3 html=5 number=5
.pardef tit_3 copy=tit_2 html=4 number=4
.pardef tit_2 copy=tit_1 html=3 spaceb=12 spacea=6 size=12 number=3
.pardef tit_1 html=2 bold=y spaceb=14 spacea=7 keepwn=y size=14
              number=2 lmargin=16
.pardef tit_0 html=1 spaceb=18 spacea=9 bold=y keepwn=y size=18
              number=1 justify=center newpage=y
.pardef pari copy=par_0 italic=y underline=y keepwn=y spaceb=12
             spacea=6 size=12
.pardef parb2 copy=parb
.pardef parb copy=par_0 bold=y keepwn=y spaceb=12 spacea=6 size=12
```

Texte

```
.pardef par_7 copy=par_0
.pardef par_6 copy=par_0
.pardef par_5 copy=par_0
.pardef par_4 copy=par_0
.pardef par_3 copy=par_0
.pardef par_2 copy=par_0
.pardef par_1 copy=par_0
.pardef par_0 spaceb=2 spacea=4 size=10 lmargin=16
```

Enumérations

```
.pardef enum_3 copy=enum_0 lmargin=48
.pardef enum_2 copy=enum_0 lmargin=40
.pardef enum_1 copy=enum_0 lmargin=32
.pardef enum_0 copy=par_0 lmargin=24 bullet=y
```

Courier

```
.pardef cmd_4 copy=cmd_0
.pardef cmd_3 copy=cmd_0
.pardef cmd_2 copy=cmd_0
.pardef cmd_1 copy=cmd_0
.pardef cmd_0 spaceb=2 spacea=4 size=10 color=green lmargin=16
              family=courier bold=y
```

Autres

```
.pardef note spaceb=6 spacea=6 size=10 family=Times
             color=red lmargin=8
.pardef PageNum italic=y lmargin=0 size=8 spacea=0
             spaceb=0 family=Times
```

Graphiques

```
.pardef GraphFooter spaceb=0 spacea=0 lmargin=0 size=8
.pardef GraphHeader spaceb=0 spacea=0 lmargin=0 size=10 bold=yes
.pardef GraphLegend spaceb=0 spacea=0 lmargin=0 size=8 bold=yes
.pardef GraphLabel spaceb=0 spacea=0 lmargin=0 size=8
.pardef GraphTitle copy=TableTitle
.pardef GraphFootnote copy=TableFootnote
```

Tableaux

```
.pardef TableTitle bold=y lmargin=0 size=12 spacea=12 spaceb=12
                   family=Times
.pardef TableFootnote italic=y lmargin=0 size=10 spacea=10
                   spaceb=5 family=Times
.pardef TableFooter copy=TableCell bold=y italic=yes
.pardef TableHeader copy=TableCell bold=y
.pardef TableCell lmargin=0 size=8 spacea=0 spaceb=0
```

9.9.4.1 La commande .par

Cette instruction fixe le type de tous les paragraphes suivants. Sa portée s'arrête lorsqu'une nouvelle commande **.par** est rencontrée.

Syntaxe

```
.par partag
```

Exemple

```
.par enum_1
```

9.9.4.2 La commande .par1

Cette instruction fixe le type du seul paragraphe qui suit la commande. Sa portée s'arrête dès la fin du paragraphe suivant.

Syntaxe

```
.par1 partag
```

Exemple

```
.par par_1
.par1 tit_1
TITRE en format tit_1

Texte du paragraphe en format par_1
```

9.9.4.3 La commande .pardef

Cette commande permet de définir de nouveaux paragraphes et de modifier les propriétés de paragraphes existants. Plusieurs propriétés peuvent être modifiées dans la même commande.

Syntaxe

```
.pardef partag prop=val ...

Valeur de prop
--------------
    copy     = partag
    lmargin  = nn
    spacea   = nn
    spaceb   = nn
    number   = nn
    html     = nn
    justify  = Left | Center | Right | Decimal | Justify
```

```
keepwn    = Yes | No
newpage   = Yes | No
bullet    = Yes | No
color     = Black | white | red | green | blue |
            cyan | magenta | yellow
family    = Arial | Times | Courier
size      = nn
italic    = Yes | No
bold      = Yes | No
underline= Yes | No
```

Si **partag** correspond à un nom de paragraphe déjà connu, les propriétés de ce dernier sont modifiées par la commande. Sinon, un nouveau paragraphe est créé.

PROPRIÉTÉS PAR DÉFAUT

Lors de la création d'un nouveau pragraphe, les propriétés suivantes sont définies par défaut:

```
lmargin   = 0
spacea    = 2
spaceb    = 4
number    = 0
html      = 0
justify   = Left
keepwn    = No
newpage   = No
bullet    = No
color     = Black
family    = Arial
size      = 10
italic    = No
bold      = No
underline= No
```

copy=partag

Tous les attributs du paragraphe **partag** sont copiées dans ce paragraphe. La position du paramètre a de l'importance: si cette propriété apparaît après une autre, la première est sans effet.

Exemple

```
.pardef par_1 copy=par_0 lmargin=20
.pardef par_2 copy=par_1 lmargin=30
```

lmargin=nn

La valeur de nn indique en points la marge de gauche du paragraphe. Un point vaut 1/72e de pouce. Par défaut, cette valeur est fixée à 0.

spacea=nn

La valeur de nn indique en points l'espace blanc avant le paragraphe. Un point vaut 1/72e de pouce. Par défaut, cette valeur est fixée à 2 pts.

spaceb=nn

La valeur de nn indique en points l'espace blanc après le paragraphe. Un point vaut 1/72e de pouce. Par défaut, cette valeur est fixée à 4 pts.

number=nn

La valeur de nn indique le niveau de numérotation du paragraphe.

- nn=0 si le paragraphe ne fait pas partie d'une suite de paragraphes numérotés (défaut)
- nn=1 si le paragraphe n'est pas numéroté mais que les numéros doivent être réinitialisés (par exemple tit_0).
- nn>1 si le paragraphe est préfixé par nn - 1 numéros automatiquement incrémentés. Le dernier nombre est incrémenté, les autres restant à leur valeur antérieure.

Exemples

```
.par1 tit_0
TITRE niveau 0
.par1 tit_1
TITRE niveau 1
.par1 tit_1
TITRE niveau 1
.par1 tit_2
TITRE niveau 2
.par1 tit_2
TITRE niveau 2
.par1 tit_1
TITRE niveau 1
.par1 tit_0
TITRE niveau 0
.par1 tit_1
TITRE niveau 1
```

Donne après interprétation:

```
TITRE niveau 0
  1. TITRE niveau 1
  2. TITRE niveau 1
    2.1 TITRE niveau 2
    2.2 TITRE niveau 2
  3.TITRE niveau 1
TITRE niveau 0
  1. TITRE niveau 1
```

html=nn

Si nn est supérieur à 0, indique que ce paragraphe doit être transposé en HTML sous forme d'un tag <Hnn>. Cette propriété n'a pas d'effet dans les autres destinations.

justify=L|C|R|J|D

Cette propriété permet de choisir la justification du paragraphe. Cinq possibilités sont disponibles, mais toutes ne sont pas opérationnelles pour toutes les destinations.

- Left=cadré à gauche (défaut)
- Center=centré
- Right=cadré à droite
- Decimal=cadré sur le point décimal (en général, cadré à droite)
- Justify=cadré à gauche et à droite (pas opérationnel en GDI)

keepwn

Force un saut de page s'il n'y a pas assez de place sur la page pour imprimer le paragraphe courant ET le suivant. Si plusieurs paragraphes successifs ont cette propriété, un saut de page a lieu si tous ces paragraphes ne peuvent se trouver sur la même page.

- keepwn=yes : force un saut de page au besoin
- keepwn=no : ne force pas de saut de page (défaut)

newpage

Force un saut de page avant le paragraphe.

- newpage=yes : force un saut de page
- newpage=no : ne force pas de saut de page (défaut)

bullet

Place un carré avant le paragraphe (énumération). Le carré est décalé de 8 pts par rapport à la marge de gauche du paragraphe.

- bullet=yes : place un bullet
- bullet=no : ne place pas un bullet (défaut)

color

Fixe la couleur du texte du paragraphe. Seule la première lettre est interprétée. Les couleurs suivantes sont disponibles:

- color=Black (défaut)
- color=white
- color=red
- color=green
- color=blue
- color=cyan
- color=magenta
- color=yellow

Black commence par une majuscule, blue par une minuscule!

family

Détermine la famille de caractères utilisée pour le paragraphe. Trois possibilités sont disponibles:

- family=Arial (défaut)
- family=Times
- family=Courier

size

Fixe la taille en points de la police. Par défaut, cette valeur est fixée à 10.

italic

Détermine si la police doit être en italic ou non.

- italic=Yes
- italic=No (défaut)

bold

Détermine si la police doit être en gras ou non.

- bold=Yes
- bold=No (défaut)

underline

Détermine si la police doit être soulignée ou non.

- underline=Yes
- underline=No (défaut)

9.9.5 COMMANDES DE MISE EN PAGE

Les commandes de mise en page suivantes sont disponibles:

```
.page
.line1
.line2
.pghead text
.pgfoot text
```

9.9.5.1 LA COMMANDE .page

Cette commande force un saut à la page.

9.9.5.2 LA COMMANDE .line1

Cette commande insère une ligne horizontale dans le texte.

9.9.5.3 LA COMMANDE .line2

Cette commande insère une ligne horizontale de double épaisseur dans le texte.

9.9.5.4 LA COMMANDE .pghead

Cette commande définit le texte qui devra se trouver en haut de page lors de l'impression. Elle n'est exploitée que pour les impressions directes (GDI).

SYNTAXE

```
.pghead texte libre
```

Si on souhaite insérer le numéro de la page courante, il faut introduire dans le texte %d.

9.9.5.5 LA COMMANDE .pgfoot

Cette commande définit le texte qui devra se trouver en bas de page lors de l'impression. Elle n'est exploitée que pour les impressions directes (GDI).

SYNTAXE

```
.pgfoot texte libre
```

Si on souhaite insérer le numéro de la page courante, il faut introduire dans le texte %d.

9.9.6 DÉFINITION DES TABLEAUX

Les tableaux commencent par la commande .tb et se terminent par .te. Chaque tableau peut contenir 5 sections distinctes:

- un titre (.ttitle) en dehors du corps du tableau
- une footnote (.tfootnote) en dehors du corps du tableau
- des lignes d'entête (.theader)
- des lignes de fin (.tfooter)
- des lignes centrales (.tbody)

Des attributs spécifiques peuvent être définis également, comme la largeur en cm des colonnes (.twidth) ou les couleurs de pourcentages d'hachurage des différentes sections.

SYNTAXE

```
.tb [nc]
.tpage 0|1
.tbreak 0|1
.tshading HFColor HFShading BodyColor BodyShading
    HFcolor et BodyColor = B : black
    HFcolor et BodyColor = r : red
    HFcolor et BodyColor = g : green
    HFcolor et BodyColor = b : blue
    HFcolor et BodyColor = y : yellow
    HFcolor et BodyColor = m : magenta
    HFcolor et BodyColor = c : cyan
    HFcolor et BodyColor = w : white
    HFShading et BodyShading=0..100

.tborder 1
.twidth col1 col2 ...
.ttitle text
.tfootnote text
.theader
&cell_definition &cell_definition ...
    ...
.tbody
&cell_definition &cell_definition ...
    ...
.tfooter
&cell_definition &cell_definition ...
    ...
.tl
.te
```

Cellule d'un tableau

Les lignes de chaque section du tableau se présentent de façon semblable. Elles sont constituées d'un ensemble de cellules commençant par le caractère défini par la commande .sep. Par défaut ce caractère est &.

Chaque cellule se compose d'une description de ses caractéristiques suivie du texte de la cellule. Les blancs en fin de cellule sont ignorés. Cela permet un éventuel alignement sans dommage pour le résultat obtenu. Les blancs en début de cellule sont conservés pour la seule première colonne.

```
sep span justification [partag] text
```

où

- sep est le caractère défini par .sep
- span est le nombre de colonnes couvertes par la cellule
- justification est
  - L pour justification à gauche
  - C pour centrer le texte dans la cellule
  - R pour justification à droite
  - D pour justification sur le point décimal
  - J pour justification à gauche et à droite
- partag est le paragraphe de la cellule (opt.)
- text est le texte de la cellule

Les champs **span** et justification sont optionnels sauf si le texte de la cellule commence par un chiffre. Par défaut, les cellules sont alignées à gauche dans le corps du tableau et centrée dans le header et le footer.

Exemples :

```
&2C<Title>Titre du tableau
&1L\iTexte italic\I
&Cellule normale&Deuxième colonne
```

Ligne d'un tableau

Une ligne est composée de cellules :

```
CellDefn CellDefn ...
```

Un tableau pet comporter autant de lignes que souhaité. En cas de report à la page, les lignes d'entête du tableau sont reproduites.

EXEMPLE 1

```
.tb 2
.sep @
.twidth 3 3
.tshading b 30 b 3
.ttitle La section GLOBAL
.theader
@2CGLOBAL
@1LKeyword@1LValues
.tbody
@1L\cDATE_FORMAT      @1L\cformat_defn
@1L\cTIME_FORMAT      @1L\cformat_defn
@1L\cNB_DEC           @1L\cnn
@1L\c                 @1L\c
@1L\cCLOCK_POS        @1L\cline col
@1L\cDATE_POS         @1L\cline col
@1L\cMEM_POS          @1L\cline col
@1L\cKEY_POS          @1L\cline col
.te
```

Cet exemple donne le résultat suivant (diffère selon la destination):

GLOBAL	
Keyword	**Values**
DATE_FORMAT	format_defn
TIME_FORMAT	format_defn
NB_DEC	nn
CLOCK_POS	line col
DATE_POS	line col
MEM_POS	line col
KEY_POS	line col

EXEMPLE 2

```
.tb 5
.sep &
&5L\bCompte de l'ensemble des administrations publiques \B
&1L\i(En pour-cent du PIB)\I&1L&1L&1L&1L
.tl
&1L &1C90&1C91&1C92&1C93
.tl
&1L\bRecettes courantes\B&1D46.26&1D46.36&1D46.42&1D47.15
&1L&1L&1L&1L&1L
&1L       1. Excédent brut d'exploitation&1D0.37&1D0.38&1D0.37&1D0.37
&1L       2. Impôts directs&1D16.95&1D16.55&1D16.45&1D16.53
&1L          particuliers&1D14.49&1D14.13&1D14.28&1D14.07
&1L          sociétés&1D2.46&1D2.42&1D2.17&1D2.46
&1L       3. Impôts indirects&1D10.86&1D10.78&1D10.88&1D11.25
&1L       4. Cotisations de Sécurité sociale&1D17.11&1D17.72&1D17.94&1D18.30
&1L       5. Revenus de le propriété&1D1.49&1D1.57&1D1.45&1D1.42
&1L          dont: chômage (C.C.I.)&1D1.11&1D1.21&1D1.34&1D1.54
&1L       6. Autres recettes nettes&1D-0.50&1D-0.63&1D-0.67&1D-0.73
&1L\bDépenses courantes\B&1D50.08&1D50.99&1D51.54&1D52.36
&1L       1. Consommation publique&1D14.10&1D14.48&1D14.27&1D14.78
&1L       2. Subventions&1D2.24&1D2.26&1D2.14&1D2.05
&1L       3. Transferts aux ménages&1D22.94&1D23.82&1D24.08&1D24.50
&1L       4. Intérêts effectifs&1D10.79&1D10.43&1D11.05&1D11.04
&1L\bEpargne courante\B&1D-3.82&1D-4.62&1D-5.12&1D-5.22
&1L\bSolde des opérations en capital\B&1D-1.81&1D-1.89&1D-2.05&1D-2.24
```

```
&1L  dont: Investissements&1D1.20&1D1.26&1D1.34&1D1.43
&1L\bCapacité de financement\B&1D-5.63&1D-6.51&1D-7.17&1D-7.46
&1L  dont: Solde primaire&1D5.16&1D3.91&1D3.87&1D3.58
&1L\bOpérations financières\B&1D1.92&1D1.07&1D0.69&1D2.04
&1L\bSolde net à financer\B&1D-7.55&1D-7.58&1D-7.87&1D-9.50
&1L\bDette publique\B&1D127.49&1D129.33&1D130.64&1D137.03
.tl
&5L09/05/97
.tl
.te
```

Cet exemple donne le résultat suivant (diffère selon la destination):

Compte de l'ensemble des administrations publiques

(En pour-cent du PIB)

	90	91	92	93
Recettes courantes	46.26	46.36	46.42	47.15
1. Excédent brut d'exploitation	0.37	0.38	0.37	0.37
2. Impôts directs	16.95	16.55	16.45	16.53
particuliers	14.49	14.13	14.28	14.07
sociétés	2.46	2.42	2.17	2.46
3. Impôts indirects	10.86	10.78	10.88	11.25
4. Cotisations de Sécurité sociale	17.11	17.72	17.94	18.30
5. Revenus de le propriété	1.49	1.57	1.45	1.42
dont: chômage (C.C.I.)	1.11	1.21	1.34	1.54
6. Autres rectettes nettes	-0.50	-0.63	-0.67	-0.73
Dépenses courantes	50.08	50.99	51.54	52.36
1. Consommation publique	14.10	14.48	14.27	14.78
2. Subventions	2.24	2.26	2.14	2.05
3. Transferts aux ménages	22.94	23.82	24.08	24.50
4. Intérêts effectifs	10.79	10.43	11.05	11.04
Epargne courante	-3.82	-4.62	-5.12	-5.22
Solde des opérations en capital	-1.81	-1.89	-2.05	-2.24
dont: Investissements	1.20	1.26	1.34	1.43
Capacité de financement	-5.63	-6.51	-7.17	-7.46
dont: Solde primaire	5.16	3.91	3.87	3.58
Opérations financières	1.92	1.07	0.69	2.04
Solde net à financer	-7.55	-7.58	-7.87	-9.50
Dette publique	127.49	129.33	130.64	137.03

09/05/97

9.9.6.1 LA COMMANDE .TB

Cette instruction débute un nouveau tableau. L'unique paramètre indique le nombre de colonnes du tableau. Certaines lignes peuvent ne pas comporter pas assez de colonnes. Si plus de colonnes qu'indiqué apparaissent lors de la lecture, le nombre de colonnes s'adapte pour toutes les lignes. Dans les versions antérieure, trop de colonnes génèrait un message d'erreur.

SYNTAXE

```
.tb [nc]
```

9.9.6.2 LA COMMANDE .TE

Cette commande indique la fin du tableau.

SYNTAXE

```
.te
```

9.9.6.3 LA COMMANDE .TTITLE

Cette commande fixe le texte du titre du tableau. Ce titre ne fait pas partie du corps du tableau mais est le paragraphe qui le précède.

Le paragraphe est de type **`TableTitle`**.

Dans les systèmes que le permettent, le tableau est accroché à ce paragraphe (anchored).

Le titre peut être composé de plusieurs lignes, chacune faisant l'objet d'une commande différente.

SYNTAXE

```
.ttitle texte
.ttitle suite du texte
```

EXEMPLE

```
.ttitre Production par secteur
.ttitre \s(en % du PNB)\S
```

9.9.6.4 LA COMMANDE .TFOOTNOTE

Cette commande fixe le texte de la footnote du tableau. Ce texte ne fait pas partie du corps du tableau mais est le paragraphe qui le suit. Ce paragraphe se trouve sur la même page que le tableau.

Le paragraphe est de type **`TableFootnote`**.

Le texte peut être composé de plusieurs lignes, chacune faisant l'objet d'une commande différente.

SYNTAXE

```
.tfootnote texte
.tfootnote suite du texte
```

EXEMPLE

```
.tfootnote Source: Bureau fédéral du Plan, INS
```

9.9.6.5 LA COMMANDE .THEADER

Cette commande indique que les lignes suivantes du tableau font partie des lignes d'entête (Heading) du tableau. Toutes les lignes jusqu'au prochain .tfooter ou .tbody en feront partie.

Les paragraphes de ces cellules sont par défaut de type **`TableHeader`**.

SYNTAXE

```
.theader
&line definition
    ...
```

9.9.6.6 LA COMMANDE .TFOOTER

Cette commande indique que les lignes suivantes du tableau font partie des lignes de fin (Footer) du tableau. Toutes les lignes jusqu'au prochain .tbody ou .tbody en feront partie.

Les paragraphes de ces cellules sont par défaut de type **`TableFooter`**.

SYNTAXE

```
.tfooter
&line definition
    ...
```

9.9.6.7 LA COMMANDE .TBODY

Cette commande indique que les lignes suivantes du tableau font partie des lignes de corps (Body) du tableau. Toutes les lignes jusqu'au prochain .tfooter ou .theader en feront partie.

Les paragraphes de ces cellules sont par défaut de type **`TableCell`**.

SYNTAXE

```
.tbody
&line definition
    ...
```

9.9.6.8 LA COMMANDE .TWIDTH

Cette commande permet de déterminer la largeur des colonnes du tableau. Les valeurs sont exprimées en cm.

SYNTAXE

```
.twidth col1 col2 ...
```

La dernière valeur est répétée pour les colonnes suivantes.

Par défaut,

- GDI : les valeurs sont calculées en fonction du texte avec un maximum de 45% de la largeur du papier pour la première colonne, sauf s'il n'y a qu'une seule colonne.
- dans les autres systèmes : 6 cm pour la première colonne, 1.5 pour les suivantes.

EXEMPLE

```
.twidth 6.5 1.5   ---> 6,5 cm pour la colonne 1
                       1,5 pour les autres colonnes
```

9.9.6.9 LA COMMANDE .TPAGE

Cette commande permet de forcer ou non un saut de page à chaque début de tableau. Elle peut être introduite n'importe où dans le fichier a2m et prend effet dès le tableau en cours ou le tableau suivant. Elle est valable jusqu'à la fin du fichier. Si plusieurs impressions ont lieu au sein du même programme, la dernière valeur rencontrée aura effet pour les fichiers suivants.

SYNTAXE

```
.tpage 0 | 1
```

EXEMPLE

```
.tpage 1 ---> force un saut de page
```

9.9.6.10 LA COMMANDE .TBREAK

Cette commande permet de forcer ou non un tableau à être entièrement sur la même page (pour peu que la page soit assez grande). Elle peut être introduite n'importe où dans le fichier a2m et prend effet dès le tableau en cours ou le tableau suivant. Elle est valable jusqu'à la fin du fichier. Si plusieurs impressions ont lieu au sein du même programme, la dernière valeur rencontrée aura effet pour les fichiers suivants.

SYNTAXE

```
.tbreak 0 | 1
```

EXEMPLE

```
.tbreak 1 ---> tout le tableau se trouvera sur la même page
```

9.9.6.11 LA COMMANDE .TSHADING

Cette commande permet de préciser le type de hachurage souhaité dans le tableau courant ou dans les suivants.

SYNTAXE

```
.tshading HFColor HFShading BodyColor BodyShading
    HFcolor et BodyColor = B : black
    HFcolor et BodyColor = r : red
    HFcolor et BodyColor = g : green
    HFcolor et BodyColor = b : blue
    HFcolor et BodyColor = y : yellow
    HFcolor et BodyColor = m : magenta
    HFcolor et BodyColor = c : cyan
    HFcolor et BodyColor = w : white
    HFShading et BodyShading=0..100
```

Quatre paramètres doivent être passés :

- HFColor = couleur des header et footer
- HFShading = pourcentage du hachurage des header et footer
- BodyColor = couleur des lignes du corps du tableau
- HFShading = pourcentage du hachurage des lignes du corps du tableau

Les pourcentages font l'objet d'une approximation en fonction du système destination.

EXEMPLE

```
.tshading r 30 b 3  ---> rouge 30% et bleu 3%
```

9.9.6.12 LA COMMANDE .TBORDER

Cette commande permet de spécifier la largeur du cadre du tableau. Les valeurs sont exprimées en points.

SYNTAXE

```
.tborder nn
```

Il existe une valeur par défaut pour chacune des destinations d'impression. Cette valeur est déterminée lors de l'impression. En RTF, cette commande n'a pas d'effet dans la version actuelle.

EXEMPLE

```
.tborder 0 ---> pas de cadre
```

9.9.6.13 LA COMMANDE .TL

Cette commande insère une ligne horizontale dans le tableau.

SYNTAXE

```
.tl
```

9.9.7 DÉFINITION DES GRAPHIQUES

Le langage de définition des graphiques en A2M est suffisamment complet pour permettre une vaste gamme de dessins différents.

Une graphique commence toujours par la commande .gb et se termine par .ge. Comme pour les tableaux, un titre et une footnote extérieurs au dessin lui-même permettent d'intégrer le graphique dans le texte.

Les courbes sont représentées par des couples (x,y), ce qui offre un maximum de possibilités. Elles peuvent être de trois types: ligne, histogramme ou points.

Deux axes verticaux sont permis. Les minima et maximum peuvent être choisis pour chacun des axes (.gxmin, .gxmax, .gymin, .gymax, .gzmin, .gzmax).

Une grille peut être dessinée soit automatiquement (.ggrid), soit en fonction des desiderata de l'utilisateur (.gxgrids, .gygrids, .gzgrids), soit encore des deux manières combinées.

Une légende peut être ajoutée sur chaque axe (.gxlegend, .gylegend, .gzlegend).

```
.gb width height    (en centimètres)
.gpage 0 | 1
.gtitle text        (peut être répété)
.gfootnote text     (id.)
.gheader text       (id.)
.gfooter text       (id.)

.gxlegend text      (id.)
.gylegend text      (id.)
.gzlegend text      (id.)

.gxmin val
.gxmax val
.gymin val
.gymax val
.gzmin val
.gzmax val

.ggrid None | Minor | Major

.gxgrids [type] xval [type] xval ...
.gygrids [type] yval [type] yval ...
```

```
.gzgrids [type] zval [type] zval ...

.gxlabs (xval "text") (xval "text) ...
.gylabs (yval "text") (yval "text) ...
.gzlabs (zval "text") (zval "text) ...

.gxy L|B|S|P "Title" (xval, yval) (xval, yval) ...
.gxz L|B|S|P "Title" (xval, zval) (xval, zval) ...

.ge
```

EXEMPLE

```
.gb 8 5
.gtitle Consommation privée
.gfootnote Source: INR, Bureau fédéral du Plan
.ggrid Major
.gxy L "LGDP" 60 2049 61 2151 62 2263 63 2362 64 2526 65 2616
              66 2698 67 2803 68 2921 69 3115 70 3313 71 3434
              72 3616 73 3830 74 3986 75 3927 76 4146 77 4165
              78 4279 79 4371 80 4559 81 4514 82 4582 83 4603
              84 4702 85 4741 86 4805 87 4901 88 5144 89 5330
              90 5503 91 5601 92 5679 93 5586 94 5616 95 5691
.ge

.gb 16 5
.gtitle Kwartaal verloop privé-verbruik in constante prijzen
.gfootnote Bron : INR, federaal Planbureau
.gxlegend in % van het BNP
.gylegend (duizendtallen)
.gzlegend (percenten)
.ggrid Major
.gxMin 59
.gxMax 95
.gxlabs 85 "Break-point" 92 "Turn-point"
.gylabs 2500 "Y-Pnt"
.gzlabs 2.25 "Z-Pnt"
.gxy B "LTREND" 60 2057 61 2157 62 2263 63 2373 64 2488 65 2608
                66 2734 67 2864 68 3000 69 3141 70 3284 71 3428
                72 3571 73 3708 74 3839 75 3961 76 4074 77 4179
                78 4276 79 4366 80 4450 81 4530 82 4607 83 4685
                84 4766 85 4851 86 4942 87 5038 88 5139 89 5242
                90 5344 91 5442 92 5535 93 5625 94 5712 95 5800
.gxz L "TREND"  60 1.408142 61 1.476598 62 1.549162 63 1.624463
                64 1.703188 65 1.785335 66 1.87159 67 1.960583
                68 2.053683 69 2.150206 70 2.248099 71 2.346675
                72 2.444568 73 2.538353 74 2.62803 75 2.711546
                76 2.788902 77 2.860781 78 2.927183 79 2.988794
                80 3.046297 81 3.101062 82 3.153773 83 3.207169
                84 3.262618 85 3.320806 86 3.383101 87 3.448819
                88 3.517959 89 3.588469 90 3.658294 91 3.725381
                92 3.789046 93 3.850656 94 3.910213 95 3.970454
.ge
```

Contrairement à ce que semble indiquer l'exemple, chaque courbe doit se trouver sur une seule ligne.

9.9.7.1 LA COMMANDE .GB

Cette commande débute un graphique. Elle indique la taille de celui-ci en cm.

SYNTAXE

```
.gb width height (en centimètres)
```

9.9.7.2 LA COMMANDE .GE

Cette commande indique la fin de la définition ’un graphique.

SYNTAXE

```
.ge
```

9.9.7.3 LA COMMANDE .GXMIN

Cette commande fixe le minimum de l’axe des X. Si ce minimum ne permet pas d’inclure toutes les observations, il est automatiquement adapté.

SYNTAXE

```
.gxmin val
```

9.9.7.4 LA COMMANDE .GXMAX

Cette commande fixe le maximum de l’axe des X. Si cette valeur ne permet pas d’inclure toutes les observations, elle est automatiquement adaptée.

SYNTAXE

```
.gxmax val
```

9.9.7.5 LA COMMANDE .GYMIN

Cette commande fixe le minimum de l’axe des Y. Si cette valeur ne permet pas d’inclure toutes les observations, elle est automatiquement adaptée.

SYNTAXE

```
.gymin val
```

9.9.7.6 LA COMMANDE .GYMAX

Cette commande fixe le maximum de l’axe des Y. Si cette valeur ne permet pas d’inclure toutes les observations, elle est automatiquement adaptée.

SYNTAXE

```
.gymax val
```

9.9.7.7 LA COMMANDE .GZMIN

Cette commande fixe le minimum de l'axe des Z. Si cette valeur ne permet pas d'inclure toutes les observations, elle est automatiquement adaptée.

SYNTAXE

```
.gzmin val
```

9.9.7.8 LA COMMANDE .GZMAX

Cette commande fixe le maximum de l'axe des Z. Si cette valeur ne permet pas d'inclure toutes les observations, elle est automatiquement adaptée.

SYNTAXE

```
.gzmax val
```

9.9.7.9 LA COMMANDE .GGRID

Cette commande détermine la grille qui doit apparaître à travers le graphique. Cette grille est constitutée de droites traversant le graphique de part en part en prolongation des graduations des axes.

SYNTAXE

```
.ggrid None | Minor | Major
```

- None = pas de grille
- Minor = une droite pour chaque graduation
- Major = une droite pour chaque graduation principale

Par défaut, les "Major" grids sont retenus.

9.9.7.10 LA COMMANDE .GXGRIDS

Cette commande permet d'ajouter des droites verticales traversant le graphiques à des positions libres sur l'axe des X.

SYNTAXE

```
.gxgrids [type] xval [type] xval ...
```

- type = épaisseur du trait (1-3)
- xval = abscisse du trait

9.9.7.11 LA COMMANDE .GYGRIDS

Cette commande permet d'ajouter des droites horizontales traversant le graphiques à des positions libres sur l'axe des Y.

SYNTAXE

```
.gygrids [type] yval [type] yval ...
```

- type = épaisseur du trait (1-3)
- yval = ordonnée du trait

9.9.7.12 LA COMMANDE .GZGRIDS

Cette commande permet d'ajouter des droites horizontales traversant le graphiques à des positions libres sur l'axe des Z.

SYNTAXE

```
.gzgrids [type] zval [type] zval ...
```

- type = épaisseur du trait (1-3)
- zval = valeur sur l'axe des Z de la position du trait

9.9.7.13 LA COMMANDE .GXLEGEND

Cette commande indique le texte à placer en légende de l'axe des X.

SYNTAXE

```
.gxlegend text
```

Le texte sera imprimé avec les caractéristique du paragraphe **GraphLegend**.

9.9.7.14 LA COMMANDE .GYLEGEND

Cette commande indique le texte à placer en légende de l'axe des Y.

SYNTAXE

```
.gylegend text
```

Le texte sera imprimé avec les caractéristique du paragraphe **GraphLegend**.

9.9.7.15 LA COMMANDE .GZLEGEND

Cette commande indique le texte à placer en légende de l'axe des Z. L'axe des Z est placé verticale-

ment à droite du graphique.

SYNTAXE

```
.gzlegend text
```

Le texte sera imprimé avec les caractéristique du paragraphe **GraphLegend**.

9.9.7.16 LA COMMANDE .GPAGE

Cette commande permet de forcer ou non un saut de page à chaque début de graphique. Elle peut être introduite n'importe où dans le fichier a2m et prend effet dès le graphique en cours ou le graphique suivant. Elle est valable jusqu'à la fin du fichier. Si plusieurs impressions ont lieu au sein du même programme, la dernière valeur rencontrée aura effet pour les fichiers suivants.

SYNTAXE

```
.gpage 0 | 1
```

EXEMPLE

```
.gpage 1 ---> force un saut de page avant le graphique
```

9.9.7.17 LA COMMANDE .GTITLE

Cette commande indique le titre du graphique. Ce titre, éventuellement composé par plusieurs commandes **.gtitle** successives fait l'objet du paragraphe qui précède le graphique.

Ce paragraphe est de type **GraphTitle**.

Dans les systèmes que le permettent, le graphique est accroché à ce paragraphe (anchored).

SYNTAXE

```
.gtitle texte
.gtitle suite du texte
```

EXEMPLE

```
.gtitre Production par secteur
.gtitre \s(en % du PNB)\S
```

9.9.7.18 LA COMMANDE .GFOOTNOTE

Cette commande indique la footnote du graphique. Ce texte, éventuellement composé par plusieurs commandes .gfootnote successives fait l'objet du paragraphe qui suit le graphique.

Ce paragraphe est de type **GraphFootnote**.

La footnote se trouve sur la même page que le graphique.

SYNTAXE

```
.gfootnote text
```

9.9.7.19 LA COMMANDE .GHEADER

Cette commande indique la ou les premières lignes du titre du graphique inclus dans l'image. Ce texte sera suivi des textes et lignes référençant les courbes du graphique définies via les commandes **.gxy** ou **.gxz**.

SYNTAXE

```
.gheader text
```

Le texte sera imprimé avec les caractéristique du paragraphe **GraphHeader**.

9.9.7.20 LA COMMANDE .GFOOTER

Cette commande indique la ou les dernière lignes du titre du graphique inclus dans l'image. Ce texte est précédé des textes et lignes référençant les courbes du graphique définies via les commandes **.gxy** ou **.gxz**.

SYNTAXE

```
.gfooter text
```

Le texte sera imprimé avec les caractéristique du paragraphe **GraphFooter**.

9.9.7.21 LA COMMANDE .GXLABS

Cette commande permet d'ajouter des textes et des droites verticales traversant le graphiques à des positions libres sur l'axe des X.

Les parenthèses sont ignorées et permettent simplement de clarifier la source A2M si c'est souhaité.

SYNTAXE

```
.gxlabs (xval "text") (xval "text) ...
```

- xval = abscisse du texte et du trait
- "text" = texte à indiquer sur l'axe

Le texte sera imprimé avec les caractéristique du paragraphe **GraphLabel**.

EXEMPLE

```
.gxlabs 97 Simulation
```

9.9.7.22 LA COMMANDE .GYLABS

Cette commande permet d'ajouter des textes et des droites horizontales traversant le graphiques à des positions libres sur l'axe des Y.

SYNTAXE

```
.gylabs (yval "text") (yval "text) ...
```

- yval = ordonnée du texte et du trait
- "text" = texte à indiquer sur l'axe

Les parenthèses sont ignorées et permettent simplement de clarifier la source A2M si c'est souhaité.

Le texte sera imprimé avec les caractéristique du paragraphe **`GraphLabel`**.

9.9.7.23 LA COMMANDE .GZLABS

Cette commande permet d'ajouter des textes et des droites horizontales traversant le graphiques à des positions libres sur l'axe des Z.

SYNTAXE

```
.gzlabs (zval "text") (zval "text) ...
```

Les parenthèses sont ignorées et permettent simplement de clarifier la source A2M si c'est souhaité.

- zval = valeur en Z de la position du texte et du trait
- "text" = texte à indiquer sur l'axe

Le texte sera imprimé avec les caractéristique du paragraphe **`GraphLabel`**.

9.9.7.24 LA COMMANDE .GXY

Cette commande permet de définir une courbe en X-Y. Elle est constituée de trois parties:

- type de dessin courbe
- titre de la série
- couples de valeurs

SYNTAXE

```
.gxy type "titre" (xval, yval) (xval, yval) ...
```

- type indique le type de la courbe. On peut choisir entre
  - Line : courbe continue

- Bar : histogramme
- Scatter : points

• titre indique le titre qui sera repris avec la référence à la courbe dans le bas du graphique. Ce texte doit se trouver entre doubles quotes.

• couples de valeurs représentant les points de passage de la courbe.

Les parenthèses sont ignorées et permettent simplement de clarifier la source A2M si c'est souhaité.

Le valeurs peuvent être représentées dans tous les formats, avec ou sans décimales. Les notations scientifiques sont supportées.

Une valeur non définie doit être représentée par --. La courbe subit une rupture à cette position.

Le texte sera imprimé avec les caractéristique du paragraphe **GraphHeader**.

EXEMPLE

```
.gxy B "Pommes de terre" 60 20.57E2 70 2157 80 -- 90 .2373E4
```

9.9.7.25 LA COMMANDE .GXZ

Cette commande permet de définir une courbe en X-Z. L'axe Z est un axe alternatf qui est positionné à gauche du graphique. Il permet de placer sur un même graphique des données en différentes unités.

La commande **.gxz** est constituée de trois parties:

• type de dessin courbe

• titre de la série

• couples de valeurs

SYNTAXE

```
.gxz type "titre" (xval, zval) (xval, zval) ...
```

• type indique le type de la courbe. On peut choisir entre

- Line : courbe continue
- Bar : histogramme
- Scatter : points

• titre indique le titre qui sera repris avec la référence à la courbe dans le bas du graphique. Ce texte doit se trouver entre doubles quotes.

• couples de valeurs représentant les points de passage de la courbe.

Les parenthèses sont ignorées et permettent simplement de clarifier la source A2M si c'est souhaité.

Le valeurs peuvent être représentées dans tous les formats, avec ou sans décimales. Les notations scientifiques sont supportées.

Une valeur non définie doit être représentée par --. La courbe subit une rupture à cette position.

Le texte sera imprimé avec les caractéristique du paragraphe **GraphHeader**.

EXEMPLE

```
.gxz B "Pommes de terre (%PNB)" 60 0.12E1 70 -- 80 1.1 90 1.3E0
```

9.9.8 INSERTION D'IMAGES

Il est possible d'insérer des images dans le texte, du moins pour certaines destinations, comme Frame, RTF ou HTML.

Tous les types de fichiers ne sont pas utilisables. On utisera les fichiers .gif ou .bmp pour frame, .bmp pour rtf.

Cette fonction sera généralisée plus tard.

```
\a"filename"
\a'filename'
\A"filename"
\A'filename'
```

Dans le premier cas, l'image est "flottante" (FLOAT) en Frame, dans le second elle est liée à la ligne où elle est définie.

9.9.8.1 LA SÉQUENCE \A

Cette séquence insère à l'endroit du texte où elle se trouve un point d'ancrage pour un fichier image.

Le nom du fichier doit se trouver entre doubles ou simples quotes.

SYNTAXE

```
\a"filename"
\a'filename'
```

Il n'y a pas de texte de remplacement de cette séquence contrairement à \A. L'image n'a pas de titre.

Frame Maker

Un Frame de la largeur de la page est créé pour contenir l'image. Ce frame est "flottant" si le nom du fichier est entre doubles quotes et fixé sous la ligne s'il est entre simples quotes.

EXEMPLE

```
Comme on le voit sur la figure suivante, \a'ecran1.gif' ...
```

Dans le texte on aura :

```
Comme on le voit sur la figure suivante, ...

 ---------------------------------------------------------
|                                                         |
|                      I M A G E                          |
|                                                         |
 ---------------------------------------------------------
```

et l'image elle-même suivra.

9.9.8.2 LA SÉQUENCE \A

Cette séquence insère à l'endroit du texte où elle se trouve un point d'ancrage pour un fichier image. Le nom du fichier doit se trouver entre doubles ou simples quotes.

SYNTAXE

```
\A"filename"
\A'filename'
```

Dans le texte lui-même, les mots "Figure nn" remplacent la séquence, permettant de faire référence à l'image qui peut ne pas se situer juste sous la ligne. Si on ne souhaite pas ce texte, il faut utiliser \a.

Frame Maker

Un Frame de la largeur de la page est créé pour contenir l'image. Ce frame est "flottant" si le nom du fichier est entre doubles quotes et fixé sous la ligne s'il est entre simples quotes.

EXEMPLE

```
Comme on le voit sur la \A"ecran1.gif", ...
```

Dans le texte on aura :

```
Comme on le voit sur la Figure 1 ...

 ---------------------------------------------------------
|                      Figure 1                           |
|                                                         |
|                      I M A G E                          |
|                                                         |
 ---------------------------------------------------------
```

et l'image elle-même suivra avec les mots "Figure 1" comme titre.

9.9.9 ENRICHISSEMENTS TYPOGRAPHIQUES

Ces séquences peuvent se retrouver n'importe où dans le texte.

L'objet de ces séquence est de changer, à l'intérieur d'un paragraphe, les attributs typographiques des caractères. A partir de l'endroit où la séquence de caractères est rencontrée, et jusqu'à la séquence qui l'annule ou jusqu'à la fin du paragraphe, l'attribut sera d'application.

Les attributs sont cumulatifs : on peut donc mettre un mot en souligné italique, exposant, taille plus petite si on le souhaite.

Ces séquences peuvent se retrouver n'importe où dans le texte, y compris dans une cellule de tableau ou dans les textes d'un graphique.

Les modifications d'attributs, toujours préfixées par un \ ou le caractère défini par la commande .esc, sont :

```
\b ...\B      Bold       On ... Off
\i ...\I      Italic     On ... Off
\u ...\U      Underline  On ... Off
\o ...\O      Strikethrough On ... Off
\c ...\C      Courier    On ... Off
\e            suppression de Bold, Italic et Underline
\s            Size--
\S            Size++
\+            Normal
\-            Indice
\=            Exposant
\p            Newline
\t            Tab
\kcolor
    color=B : black
    color=r : red
    color=g : green
    color=b : blue
    color=y : yellow
    color=m : magenta
    color=c : cyan
    color=w : white
```

9.9.9.1 La séquence \b ... \B

Cette séquence permet de mettre en caractères gras le texte qu'elle encadre.

Syntaxe

```
\b ...\B
```

9.9.9.2 La séquence \i ... \I

Cette séquence permet de mettre en caractères italiques le texte qu'elle encadre.

Syntaxe

```
\i ...\I
```

9.9.9.3 La séquence \u ... \U

Cette séquence permet de souligner le texte qu'elle encadre.

SYNTAXE

```
\u ...\U
```

9.9.9.4 LA SÉQUENCE \O ... \O

Cette séquence permet de barrer le texte qu'elle encadre.

SYNTAXE

```
\o ...\O
```

9.9.9.5 LA SÉQUENCE \C ... \C

Cette séquence permet de mettre en caractères non proportionnels le texte qu'elle encadre. De plus, ce texte est imprimé en vert. Elle est surtout utilisée pour des exemples.

SYNTAXE

```
\c ...\C
```

9.9.9.6 LA SÉQUENCE \E

Cette séquence permet d'annuler les attributs typographiques gras, souligné et italique en cours.

SYNTAXE

```
\e
```

9.9.9.7 LA SÉQUENCE \S

Cette séquence permet de diminuer de deux points la taille des caractères qui suivent.

SYNTAXE

```
\s
```

9.9.9.8 LA SÉQUENCE \S

Cette séquence permet d'augmenter de deux points la taille des caractères qui suivent.

SYNTAXE

```
\s
```

9.9.9.9 LA SÉQUENCE \+

Cette séquence permet de mettre en exposant les caractères qui suivent.

SYNTAXE

```
\+
```

9.9.9.10 LA SÉQUENCE \-

Cette séquence permet de mettre en indice les caractères qui suivent.

SYNTAXE

```
\-
```

9.9.9.11 LA SÉQUENCE \=

Cette séquence permet de mettre les caractères qui suivent sur la ligne de base (annule \+ et \-).

SYNTAXE

```
\=
```

9.9.9.12 LA SÉQUENCE \P

Cette séquence force un saut à la ligne.

SYNTAXE

```
\p
```

9.9.9.13 LA SÉQUENCE \T

Cette séquence insère une tabulation dans le texte.

SYNTAXE

```
\t
```

9.9.9.14 La séquence \k

Cette séquence permet de changer la couleur des caractères qui suivent.

Syntaxe

```
\kcolor
```

Les couleurs disponibles sont:

- color=B : black
- color=r : red
- color=g : green
- color=b : blue
- color=y : yellow
- color=m : magenta
- color=c : cyan
- color=w : white

Exemple

```
Voici le mot \kbBLEU\kB, ... en bleu!
```

donne :

Voici le mot BLEU, ... en bleu!

9.9.10 Définition des topics d'aide

Un fichier a2m peut servir de source à la constitution d'un fichier d'aide de Windows. En général, les commandes nécessaires sont générées automatiquement à partir d'un fichier .hlp (scr4_hm), mais rien n'interdit de le faire manuellement.

Pour construire un fichier d'aide avec des hyperliens, il suffit de définir le début et la fin des sujets ainsi que les références dans le texte.

En a2m, la fin d'un sujet coïncide avec le début du suivant et est indiquée par la commande **.topic**.

Les références se feront par les séquences **\h ...\H** explicitées plus bas.

Le programme scr4w_ah qui permet de construire des fichiers HTML ou les fichiers nécessaires à la construction des fichiers .chm (fichier HtmlHelp) offre en plus la faculté d'automatiser les références entre les topics.

Les commandes associées à la définition du mode automatique sont:

- .topiclink : modification des paramètres pour le topic courant
- .topicslink : modification des paramètres pour tous les topics à partir du suivant.

De plus, il est possible d'associer un mot ou un groupe de mot à un topic donné à l'aide de la commande .topicalias.

9.9.10.1 LA COMMANDE .TOPIC

La commande **.topic** détermine le début d'un sujet et les informations nécessaires à son exploitation.

SYNTAXE

```
.topic nn level title
```

- nn est le numéro de référence du sujet, utilisé par les séquences **\h...H**.
- level est le niveau dans la table des matières du sujet. La table des matières de l'aide de Windows sera constituée sur base de cete information.
- title est le titre du sujet. Ce titre sera exploité pour créer un index de recherche sous Windows et pour construire une base de mots-clés.

Pour pouvoir utiliser a2m pour créer une aide Windows avec un Sommaire, il faut impérativement que les sujets soient classés dans le bon ordre (ce qui est normalement le cas).

EXEMPLE

```
.topicslink case=no
.topic 0 0 Le programme scr4w_am
.topicalias scr4w_am
    ...
    \h<1>Syntaxe des fichiers a2m\H
    ...
.topic 1 1 Syntaxe des fichiers a2m
.topicalias a2m
    ...
    \h<2>Caractères réservés\H
    \h<5>Directives\H
    ...
.topic 2 2 Caractères réservés
    ...
    \h<3>.cmd\H
    \h<4>.esc\H
    ...
.topic 3 3 .cmd
    ...
.topic 4 3 .esc
    ...
.topic 5 2 Directives
    ...
```

La commande **.topic** et les références aux sujets **\h..\H** n'ont pas d'effet sur le fichier résultat, sauf si une aide Windows est demandée (fichier rtf).

9.9.10.2 LA SÉQUENCE \H...\H

La séquence **\h<nn>...texte...\H** indique que le texte doit être lié à un sujet de l'aide déterminé par le numéro qui suit **\h**.

Lorsque l'utilisateur de l'aide sélectionnera ce texte, il pourra se déplacer directement sur le sujet référencé.

SYNTAXE

```
\h<nn> ...\H
```

Les caractères < et > doivent être présents en contiennent le numéro du sujet référencé, défini à l'aide de la commande **.topic**.

EXEMPLE

```
La liste des \h<2>caractères réservés\H se trouve ci-dessous.
```

9.9.10.3 LA COMMANDE .TOPICLINK

Cette commande permet de changer pour le topic en cours ou le dernier alias les paramètres qui détermineront les liens automatiques qui seront générés par scr4w_ah.

SYNTAXE

```
.topiclink case=yes|no auto=yes|no partial=yes|no
```

- **case=yes** : le remplacement ne s'effectue que si le texte du topic ou de l'alias sont trouvés exactement. **case=no** ne différencie pas majuscules et minuscules.
- **auto=yes** : indique que le texte du topic ou de l'alias, s'ils sont trouvés ailleurs dans le document, doivent être liés vers le topic courant. **auto=no** exclut le topic ou l'alias de ce traitement.
- **partial=no** : indique que le lien vers le topic n'est effectué que si le texte trouvé est précédé et suivi de caractères non alphanumériques. **partial=yes** permet les remplacements même si le texte du topic fait partie d'un mot.

EXEMPLE

```
.topic 2 1 L'objet ISAM
.topicalias ISAM
.topiclink case=no partial=yes auto=yes
....
.topic ....
Les Isams sont utilisés par la fonction SCR_page_to_isam().
```

Le texte de la dernière ligne sera remplacé par

```
Les \h<2>Isam\Hs sont utilisés par la fonction SCR_page_to_\h<2>isam\H().
```

Avec **partial=no**, SCR_page_to_isam() serait resté tel quel.

Si la fonction SCR_page_to_isam est un topic du document, le lien vers cette fonction sera établi avant celui vers ISAM (les sujets les plus longs sont remplacés prioritairement) et on aura :

```
Les \h<2>Isam\Hs sont utilisés par la fonction
\h<234>SCR_page_to_isam\H().
```

9.9.10.4 La commande .topicslink

Cette commande est presqu'identique à `.topiclink` à ceci près qu'elle agit sur tous les topics à partir du suivant (pas le courant!). Elle agit en quelque sorte en fixant les valeurs par défaut pour la suite du document.

Syntaxe

```
.topicslink case=yes|no auto=yes|no partial=yes|no
```

9.9.10.5 La commande .topicalias

Cette commande permet de définir les mots qui, s'ils sont recontrés ailleurs dans le document, doivent pointer automatiquement vers le topic courant.

Syntaxe

```
.topicalias texte libre
```

Pour un exemple, voir .topiclink.

9.9.11 Hyperliens

Les hyperliens peuvent être introduits dans le texte par la séquence \l"lien"texte\L.

Ces liens renvoient à des pages HTML et sont ignorés dans le cadre des générations vers d'autres formats.

D'autres hyperliens sont possibles à l'aide de \h ...\H. Dans ce dernier cas, il s'agit de liens internes au document, renvoyant à d'autres topic d'aide.