Sampling with Halton Points on n-Sphere

Wai-Shing Luk

September 28, 2023

0.1 Abstract

We discuss the generation of low discrepancy sequences over n-sphere. The introduction provides an overview of the importance of low discrepancy sequences in various applications, such as numerical integration, optimization, and simulation. The paper then discusses the desirable properties of samples over n-sphere, including uniformity, determinism, and incrementality.

The proposed method for generating low discrepancy sequences over n-sphere is then presented, which is based on the Van der Corput sequence. The paper provides a detailed explanation of the algorithm and its implementation. The paper also discusses the numerical experiments conducted to evaluate the performance of the proposed method, including the comparison with randomly generated sequences and other proposed methods.

1 Motivation and Applications

1.1 Problem Formulation

The desirable properties of samples over n-sphere include:

- being uniform,
- deterministic, and
- incremental.
  - The uniformity measures are optimized with every new point, and this is because in some applications, it is unknown how many points are needed to solve the problem in advance.

1.2 Motivation

- The topic has been well studied for sphere in 3D, i.e. $n = 2$
- Yet it is still unknown how to generate for $n > 2$.
- Some potential applications for $n > 2$ include:

  - Robotic Motion Planning (S^3 and SO(3)) (Yershova et al. 2010)
  - Spherical coding in MIMO wireless communication (Utkovski and Lindner 2006):
    * Cookbook for Unitary matrices
    * A code word = a point in S^n
  - Multivariate empirical mode decomposition (Rehman and Mandic 2010)
  - Filter bank design (Mandic et al. 2011)

1.3 Halton Sequence on S^n

- Halton sequence on S^2 has been well studied (Cui and Freeden 1997) by using cylindrical coordinates.
- Yet it is still little known for S^n where $n > 2$.
- Note: The generalization of cylindrical coordinates does NOT work in higher dimensions.

2 Review of Low Discrepancy Sequence

2.1 Basic: Van der Corput sequence

- Generate a low discrepancy sequence over $[0, 1]$
- Denote $\text{vdc}(k, b)$ as a Van der Corput sequence of k points, where b is the base of a prime number.

Figure 1: Example of Van der Corput sequence

2.2 Python code

```
def vdc_basic(n, base=2):
    vdc, denom = 0.0, 1.0
    while n:
        denom *= base
        n, remainder = divmod(n, base)
        vdc += remainder / denom
    return vdc

def vdc(n, base=2):
    '''
    n - number of vectors
```

```
    base - seeds
    '''
    for i in range(n):
        yield vdc_basic(i, base)
```

2.3 Halton sequence on $[0, 1]$

- Halton sequence: using 2 Van der Corput sequences with different bases.
- Example:

$$[x, y] = [\text{vdc}(k, 2), \text{vdc}(k, 3)]$$

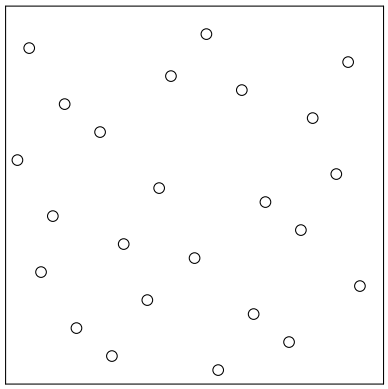

Figure 2: Example of Halton sequnce

2.4 Halton sequence on $[0, 1]^n$

- Generally we can generate Halton sequence in a unit hypercube $[0, 1]^n$:

$$[x_1, x_2, \ldots, x_n] = [\text{vdc}(k, b_1), \text{vdc}(k, b_2), \ldots, \text{vdc}(k, b_n)]$$

- A wide range of applications on Quasi-Monte Carlo Methods (QMC).

2.5 Unit Circle S^1

Can be generated by mapping the Van der Corput sequence to $[0, 2\pi]$

- $\theta = 2\pi \cdot \text{vdc}(k, b)$
- $[x, y] = [\cos\theta, \sin\theta]$

2.6 Unit Sphere S^2

Has been applied for computer graphic applications (Wong, Luk, and Heng 1997)

- Use cylindrical mapping.
- $[z, x, y]$
  $= [\cos\theta, \sin\theta\cos\varphi, \sin\theta\sin\varphi]$
  $= [z, \sqrt{1-z^2}\cos\varphi, \sqrt{1-z^2}\sin\varphi]$
- $\varphi = 2\pi \cdot \mathrm{vdc}(k, b_1)$ % map to $[0, 2\pi]$
- $z = 2 \cdot \mathrm{vdc}(k, b_2) - 1$ % map to $[-1, 1]$

2.7 Sphere S^n and SO(3)

- Deterministic point sets
  - Optimal grid point sets for S^3, SO(3) (Mitchell 2008; Yershova et al. 2010)
- No Halton sequences so far to the best of our knowledge.
- Note that cylindrical mapping method cannot be extended to higher dimensions.

2.8 SO(3) or S^3 Hopf Coordinates

- Hopf coordinates (cf. (Yershova et al. 2010))
  - $x_1 = \cos(\theta/2)\cos(\psi/2)$
  - $x_2 = \cos(\theta/2)\sin(\psi/2)$
  - $x_3 = \sin(\theta/2)\cos(\varphi + \psi/2)$
  - $x_4 = \sin(\theta/2)\sin(\varphi + \psi/2)$
- S^3 is a principal circle bundle over the S^2

2.9 Hopf Coordinates for SO(3) or S^3

Similar to the Halton sequence generation on S^2, we perform the mapping:

- $\varphi = 2\pi \cdot \mathrm{vdc}(k, b_1)$ % map to $[0, 2\pi]$
- $\psi = 2\pi \cdot \mathrm{vdc}(k, b_2)$ % map to $[0, 2\pi]$ for SO(3), or
- $\psi = 4\pi \cdot \mathrm{vdc}(k, b_2)$ % map to $[0, 4\pi]$ for S^3
- $z = 2 \cdot \mathrm{vdc}(k, b_3) - 1$ % map to $[-1, 1]$
- $\theta = \cos^{-1} z$

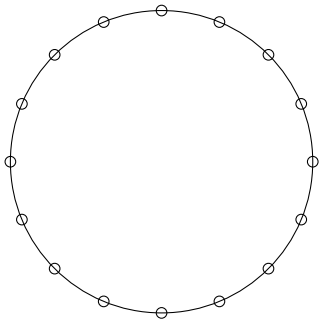

Figure 3: Sequnce mapping to a unit circle

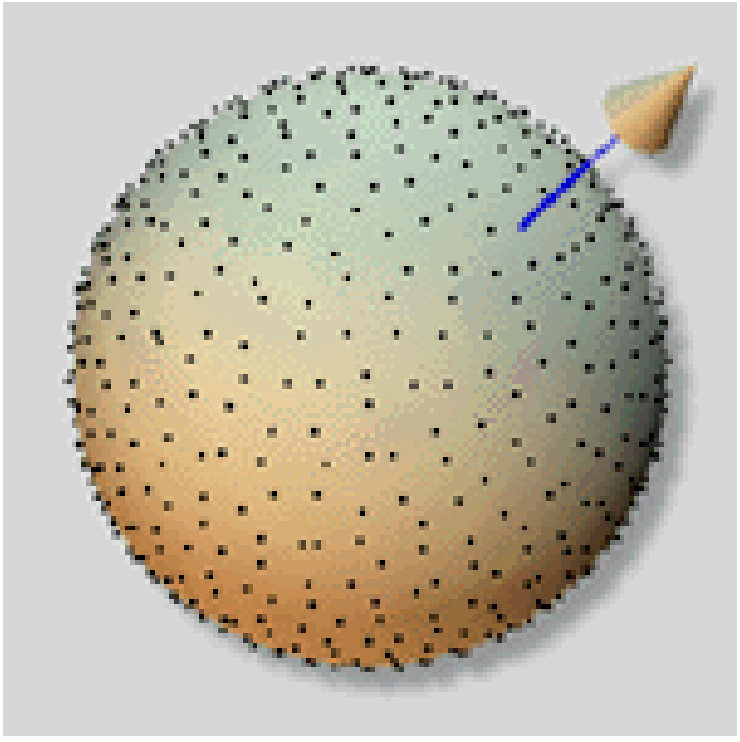

Figure 4: image

Figure 5: image

2.10 Python Code

```
def sphere3_hopf(k, b):
    vd = zip(vdc(k, b[0]), vdc(k, b[1]), vdc(k, b[2]))
    for vd0, vd1, vd2 in vd:
        phi = 2*math.pi*vd0   # map to [0, 2*math.pi]
        psy = 4*math.pi*vd1   # map to [0, 4*math.pi]
        z = 2*vd2 - 1         # map to [-1., 1.]
        theta = math.acos(z)
        cos_eta = math.cos(theta/2)
        sin_eta = math.sin(theta/2)
        s = [cos_eta * math.cos(psy/2),
             cos_eta * math.sin(psy/2),
             sin_eta * math.cos(phi + psy/2),
             sin_eta * math.sin(phi + psy/2)]
        yield s
```

3 Our approach

3.1 3-sphere

- Polar coordinates:
  - $x_0 = \cos\theta_3$
  - $x_1 = \sin\theta_3 \cos\theta_2$
  - $x_2 = \sin\theta_3 \sin\theta_2 \cos\theta_1$
  - $x_3 = \sin\theta_3 \sin\theta_2 \sin\theta_1$
- Spherical surface element:

$$dA = \sin^2(\theta_3)\sin(\theta_2)\, d\theta_1\, d\theta_2 d\theta_3$$

3.2 n-sphere

- Polar coordinates:
  - $x_0 = \cos\theta_n$
  - $x_1 = \sin\theta_n \cos\theta_{n-1}$
  - $x_2 = \sin\theta_n \sin\theta_{n-1} \cos\theta_{n-2}$
  - $x_3 = \sin\theta_n \sin\theta_{n-1} \sin\theta_{n-2} \cos\theta_{n-3}$
  - $\cdots$
  - $x_{n-1} = \sin\theta_n \sin\theta_{n-1} \sin\theta_{n-2} \cdots \cos\theta_1$

- $x_n = \sin\theta_n \sin\theta_{n-1} \sin\theta_{n-2} \cdots \sin\theta_1$

- Spherical surface element:

$$d^n A = \sin^{n-2}(\theta_{n-1}) \sin^{n-1}(\theta_{n-2}) \cdots \sin(\theta_2)\, d\theta_1\, d\theta_2 \cdots d\theta_{n-1}$$

3.3 How to Generate the Point Set

- $p_0 = [\cos\theta_1, \sin\theta_1]$ where $\theta_1 = 2\pi \cdot \text{vdc}(k, b_1)$
- Let $f_j(\theta) = \int \sin^j \theta \mathrm{d}\theta$, where $\theta \in (0, \pi)$.

  - Note 1: $f_j(\theta)$ can be defined recursively as:

$$f_j(\theta) = \begin{cases} \theta & \text{if } j = 0, \\ -\cos\theta & \text{if } j = 1, \\ (1/n)(-\cos\theta \sin^{j-1}\theta + (n-1)\int \sin^{j-2}\theta \mathrm{d}\theta) & \text{otherwise.} \end{cases}$$

  - Note 2: $f_j(\theta)$ is a monotonic increasing function in $(0, \pi)$

- Map $\text{vdc}(k, b_j)$ uniformly to $f_j(\theta)$:
  $t_j = f_j(0) + (f_j(\pi) - f_j(0))\text{vdc}(k, b_j)$
- Let $\theta_j = f_j^{-1}(t_j)$
- Define p_n recursively as:
  $p_n = [\cos\theta_n, \sin\theta_n \cdot p_{n-1}]$

4 Numerical Experiments

4.1 Testing the Correctness

- Compare the dispersion with the random point-set
  - Construct the convex hull for each point-set
  - Dispersion roughly measured by the difference of the maximum distance and the minimum distance between every two neighbour points:

$$\max_{a \in \mathcal{N}(b)} \{D(a, b)\} - \min_{a \in \mathcal{N}(b)} \{D(a, b)\}$$

  where $D(a, b) = \sqrt{1 - a^\mathsf{T} b}$

4.2 Random sequences

- To generate random points on S^n, spherical symmetry of the multidimensional Gaussian density function can be exploited.
- Then the normalized vector $(x_i/\|x_i\|)$ is uniformly distributed over the hypersphere S^n. (Fishman, G. F. (1996))

4.3 Convex Hull with 600 points

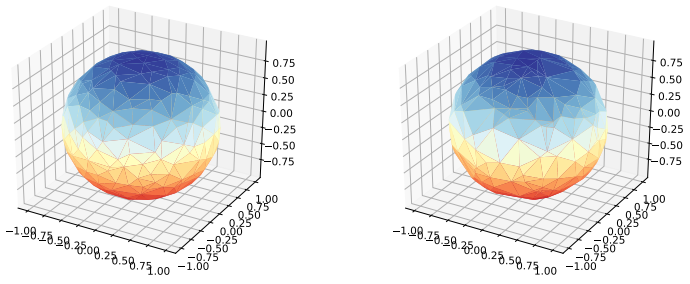

Figure 6: image

Left: our, right: random

4.4 Result for S^3

Compared with Hopf coordinate method.

4.5 Result for S^3 (II)

Compared with cylindrical mapping method.

4.6 Result for S^4

Compared with cylindrical mapping method

5 Conclusions

5.1 Conclusions

- Proposed method generates low-discrepancy point-set in nearly linear time
- The result outperforms the corresponding random point-set, especially when the number of points is small

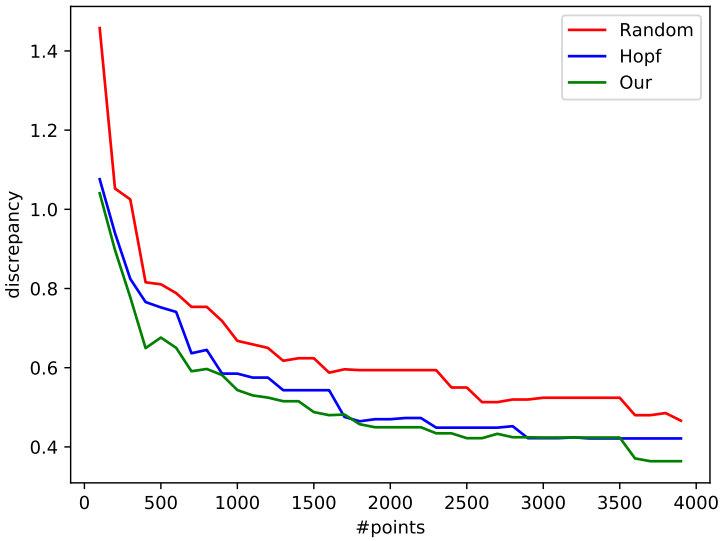

Figure 7: image

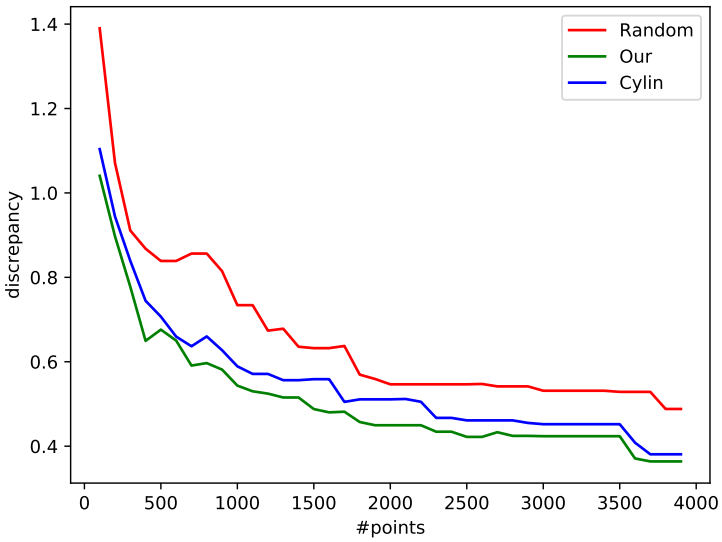

Figure 8: image

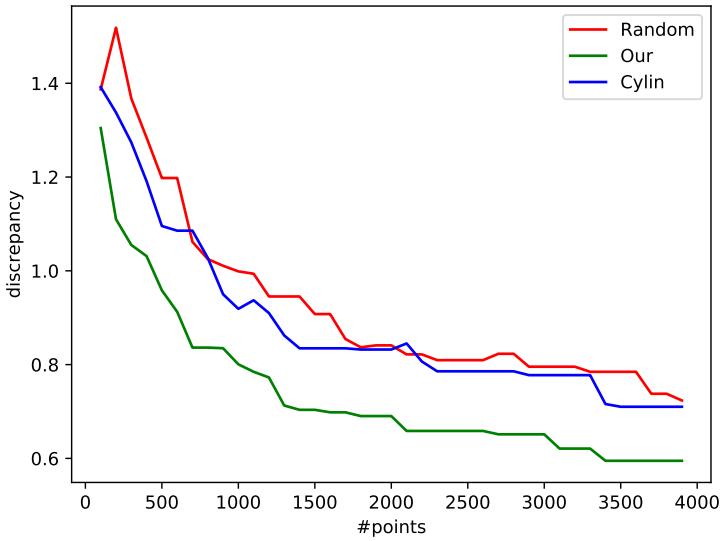

Figure 9: image

- Python code is available at here

References

Cui, Jianjun, and Willi Freeden. 1997. "Equidistribution on the Sphere." *SIAM Journal on Scientific Computing* 18 (2): 595–609.

Mandic, DP et al. 2011. "Filter Bank Property of Multivariate Empirical Mode Decomposition." *Signal Processing, IEEE Transactions on* 59 (5): 2421–26.

Mitchell, Julie C. 2008. "Sampling Rotation Groups by Successive Orthogonal Images." *SIAM Journal on Scientific Computing* 30 (1): 525–47.

Rehman, Naveed, and Danilo P Mandic. 2010. "Multivariate Empirical Mode Decomposition." *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science* 466 (2117): 1291–1302.

Utkovski, Zoran, and Juergen Lindner. 2006. "On the Construction of Non-Coherent Space Time Codes from High-Dimensional Spherical Codes." In *Spread Spectrum Techniques and Applications, 2006 IEEE Ninth International Symposium on*, 327–31. IEEE.

Wong, Tien-Tsin, Wai-Shing Luk, and Pheng-Ann Heng. 1997. "Sampling with Hammersley and Halton Points." *Journal of Graphics Tools* 2 (2): 9–24.

Yershova, Anna, Swati Jain, Steven M LaValle, and Julie C Mitchell. 2010. "Generating Uniform Incremental Grids on SO (3) Using the Hopf Fibration." *The International Journal of Robotics Research* 29 (7): 801–12.