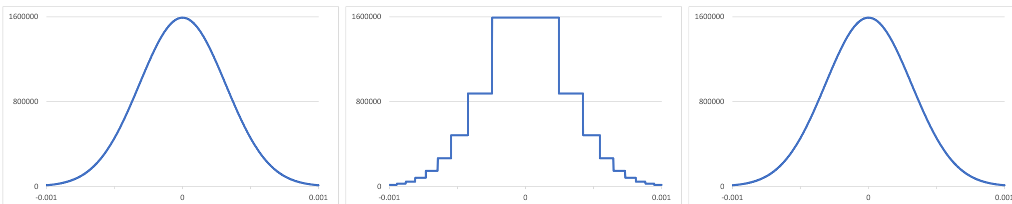
A Numerically Stable Implementation of the von Mises–Fisher Distribution on $\mathbb{S}^2$

YUSUKE TOKUYOSHI, Advanced Micro Devices, Inc., Japan



(a) Eq. 2 (double precision) (b) Eq. 2 (single precision) (c) Eq. 4 (single precision)

Fig. 1. Plots of a sharp vMF distribution ($\kappa = 10000000$, $\boldsymbol{\mu} = [0, 0, 1]$) with the traditional form (a, b) and our form (c). The horizontal axis is the angle between a direction $\boldsymbol{\omega}$ and the vMF center axis $\boldsymbol{\mu}$. (b) The traditional vMF form with single precision produces a significant numerical error. (c) Our vMF form is more numerically stable than the traditional form.

1 Introduction

The von Mises–Fisher (vMF) distribution [1953] on $\mathbb{S}^2$ is a normalized spherical Gaussian defined as

$$p(\boldsymbol{\omega}; \boldsymbol{\mu}, \kappa) = \frac{\kappa}{4\pi \sinh \kappa} \exp(\kappa(\boldsymbol{\omega} \cdot \boldsymbol{\mu})), \tag{1}$$

where $\boldsymbol{\omega} \in \mathbb{S}^2$ is a unit vector, $\boldsymbol{\mu} \in \mathbb{S}^2$ is the center axis of the vMF distribution, and $\kappa \in [0, \infty)$ is the sharpness of the vMF distribution. This distribution has often been used in computer graphics, such as real-time lighting approximation [Tsai and Shih 2006] and path guiding [Dong et al. 2023; Ruppert et al. 2020]. However, a straightforward implementation of the vMF distribution using floating points can produce a noticeable numerical error. Therefore, we describe a numerically stable implementation of the vMF distribution.

2 Numerically Stable Form of the vMF Distribution

Eq. 1 can produce NaN because $\exp(\kappa(\boldsymbol{\omega} \cdot \boldsymbol{\mu}))$ and $\sinh(\kappa)$ can overflow for large κ (e.g., $\kappa > \operatorname{arsinh}((2 - 2^{-23}) \times 2^{127}) \approx 89.4$ for single precision). To avoid such NaN, computer graphics applications have often used the following equivalent form:

$$p(\boldsymbol{\omega}; \boldsymbol{\mu}, \kappa) = \frac{\kappa}{2\pi(1 - \exp(-2\kappa))} \exp(\kappa((\boldsymbol{\omega} \cdot \boldsymbol{\mu}) - 1)), \tag{2}$$

where $\exp(\kappa((\boldsymbol{\omega} \cdot \boldsymbol{\mu}) - 1)) \in (0, 1]$ is the unnormalized spherical Gaussian. On the other hand, Eq. 2 in floating point can produce a significant error for $\kappa \to 0$ and $\kappa \to \infty$. Therefore, we use a more numerically stable form. To improve the stability for small κ, we use an accurate implementation of $a(x) = x/(\exp(x) - 1)$ [Higham 2002] for the normalization factor $\frac{\kappa}{2\pi(1-\exp(-2\kappa))}$ as follows:

$$p(\boldsymbol{\omega}; \boldsymbol{\mu}, \kappa) = \frac{a(-2\kappa)}{4\pi} \exp(\kappa((\boldsymbol{\omega} \cdot \boldsymbol{\mu}) - 1)). \tag{3}$$

Advanced Micro Devices, Inc. Technical Report, No. 25-01-5053, January 2025.

Author's Contact Information: Yusuke Tokuyoshi, yusuke.tokuyoshi@amd.com, Advanced Micro Devices, Inc., Japan.

For an HLSL implementation of $a(x)$, please see Listing 1. Although the above form is accurate for the normalization factor, the unnormalized spherical Gaussian term $\exp(\kappa((\boldsymbol{\omega}\cdot\boldsymbol{\mu})-1))$ is still numerically unstable for $\boldsymbol{\omega}\to\boldsymbol{\mu}$. This numerical error can be noticeable for a sharp vMF distribution (Fig. 1). For applications that require numerical accuracy for such high-frequency distributions, we use the Euclidean distance between $\boldsymbol{\omega}$ and $\boldsymbol{\mu}$ instead of $(\boldsymbol{\omega}\cdot\boldsymbol{\mu})-1$ as follows:

$$\boxed{p(\boldsymbol{\omega};\boldsymbol{\mu},\kappa)=\frac{a(-2\kappa)}{4\pi}\exp\left(-\frac{\kappa}{2}\|\boldsymbol{\omega}-\boldsymbol{\mu}\|^2\right)}. \tag{4}$$

Listing 2 shows our vMF implementation using the above form.

Listing 1. $a(x)=x/(\exp(x)-1)$ with cancellation of rounding errors [Higham 2002] (HLSL).

```
float x_over_expm1(float x) {
 float u = exp(x);
 if (u == 1.0f) { return 1.0f; }
 float y = u - 1.0f;
 if (abs(x) < 1.0f) { return log(u) / y; }
 return x / y;
}
```

Listing 2. Our numerically stable vMF implementation (HLSL). Instead of using $\boldsymbol{\omega}\cdot\boldsymbol{\mu}$, we use the Euclidean distance between $\boldsymbol{\omega}$ and $\boldsymbol{\mu}$.

```
float vmf(float3 dir, float3 axis, float sharpness) {
 float3 d = dir - axis;
 return exp(-0.5f * sharpness * dot(d, d)) * x_over_expm1(-2.0f * sharpness) / (4.0f * M_PI);
}
```

3 Sampling of the vMF Distribution

To sample a direction $\boldsymbol{\omega}$ according to the vMF distribution $p(\boldsymbol{\omega};\boldsymbol{\mu},\kappa)$, we first sample a direction $[\cos\phi\sin\theta,\cos\phi\sin\theta,\cos\theta]\in S^2$ in a local frame, where $\theta\in[0,\pi]$ and $\phi\in[0,2\pi)$ are the polar coordinates of this local direction. Then, we rotate the local direction into world space. For this case, the azimuthal angle ϕ is uniformly distributed as follows:

$$\phi=2\pi\xi_0, \tag{5}$$

where $\xi_0\in[0,1)$ is a uniform random number. To sample $\cos\theta=\boldsymbol{\omega}\cdot\boldsymbol{\mu}$ using a different uniform random number $\xi_1\in[0,1)$, Jakob [2012] improved the numerical stability from Jung [2009] by deriving the following form:

$$\cos\theta=1+\frac{1}{\kappa}\log\left(\xi_1+(1-\xi_1)\exp(-2\kappa)\right). \tag{6}$$

However, this sampling can still produce a significant error for small κ, because the precision of the random variable is lost by $\xi_1+(1-\xi_1)\exp(-2\kappa)\longrightarrow 1$ for $\kappa\longrightarrow 0$. To reduce the error, we replace ξ_1 with $1-\xi_1$ in Eq. 6 as follows:

$$\cos\theta=1+\frac{1}{\kappa}\log\left(1-\xi_1+\xi_1\exp(-2\kappa)\right)=1+\frac{1}{\kappa}\mathrm{log1p}\left(\xi_1\mathrm{expm1}(-2\kappa)\right), \tag{7}$$

where $\mathrm{expm1}(x)=\exp(x)-1$ and $\mathrm{log1p}(x)=\log(1+x)$ are built-in functions available in some programming languages (e.g., C++), and they are numerically stable for small $|x|$. When κ is small, $|\xi_1\mathrm{expm1}(-2\kappa)|$ is small. Therefore, Eq. 7 reduces the numerical error for small κ. The same form was used by Frisch and Hanebeck [2023] for their deterministic sampling.

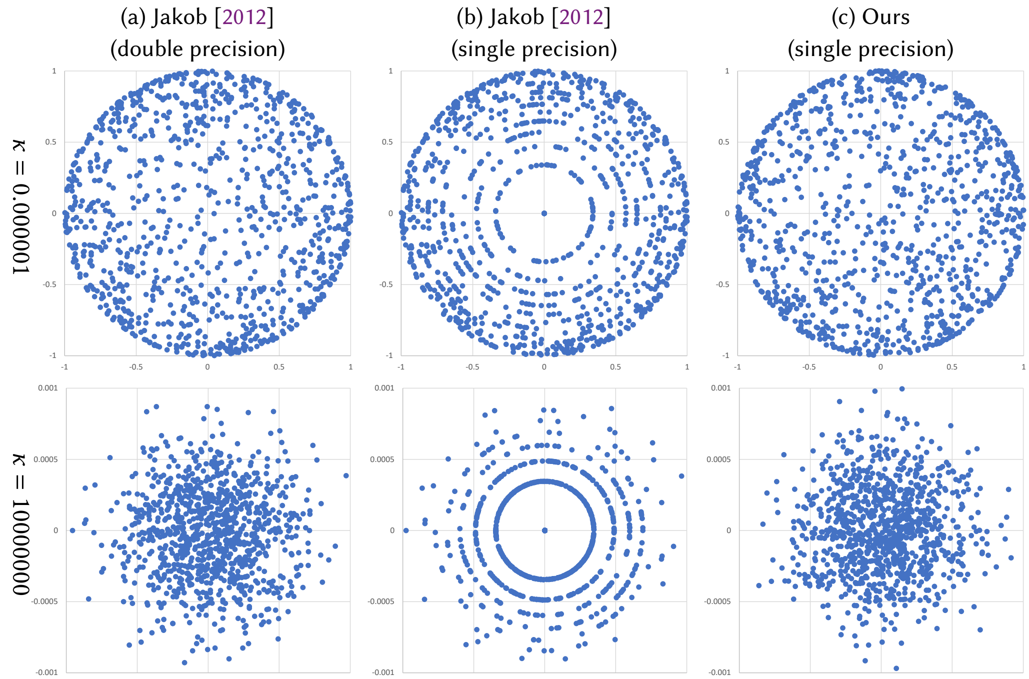


Fig. 2. Plots of sample directions $[\cos\phi\sin\theta, \sin\phi\sin\theta]$ in the local frame for vMF distributions. For low-frequency distribution (upper row) and high-frequency distribution (lower row), Jakob [2012]'s method with single precision (b) generates highly correlated samples due to numerical errors, while ours (c) does not.

Once we get $\cos\theta$, we then calculate $\sin\theta$. Although Jakob [2012] used $\sin\theta = \sqrt{1-\cos^2\theta}$, it can produce a noticeable error due to catastrophic cancellation when $\cos\theta \to 1$. To avoid the catastrophic cancellation for $\sin\theta$, we use the following equation:

$$r = \begin{cases} \frac{1}{\kappa}\mathrm{log1p}\left(\xi_1 \mathrm{expm1}(-2\kappa)\right) & \text{if } \kappa > t \\ -2\xi_1 & \text{if } \kappa \le t \end{cases}, \tag{8}$$

$$\boxed{\cos\theta = 1 + r}, \tag{9}$$

$$\boxed{\sin\theta = \sqrt{-r^2 - 2r} = \sqrt{-\mathrm{fma}(r, r, 2r)}}, \tag{10}$$

where $t = 0$ for the exact solution, and $\mathrm{fma}(x, y, z) = x \times y + z$ is the fused multiply-add operation to reduce the numerical error in floating-point arithmetic. Even if the built-in fma function is not available, the calculation of $\sin\theta = \sqrt{-r^2 - 2r}$ is still more numerically stable than $\sin\theta = \sqrt{1-\cos^2\theta}$ for $\cos\theta \to 1$. For a sharp distribution with large κ, r is densely and precisely distributed around zero. Therefore, Eq. 10 produces accurate $\sin\theta$ around zero. Fig. 2 shows plots of samples generated using our method. Listing 3 shows an HLSL implementation for our sampling routine.

To further improve the numerical stability, we use $\boxed{t = \epsilon/4}$ where ϵ is the machine epsilon. This is because, let $\mathrm{fl}(f(\cdot))$ be an operation $f(\cdot)$ in floating-point arithmetic, and x be a floating point value, then $\mathrm{fl}(\mathrm{expm1}(x)) = x$ and $\mathrm{fl}(\mathrm{log1p}(x)) = x$ when $|x| \le \epsilon/2$. Therefore, for floating-point κ

and ξ_1, we obtain

$$r \approx \mathrm{fl}\left(\frac{\mathrm{fl}\left(\mathrm{log1p}\left(\xi_1 \mathrm{fl}(\mathrm{expm1}(-2\kappa))\right)\right)}{\kappa}\right) = \mathrm{fl}\left(\frac{-2\kappa\xi_1}{\kappa}\right) \approx -2\xi_1 \quad \text{for } 0 < \kappa \leq \epsilon/4. \tag{11}$$

The rightmost approximation $r \approx -2\xi_1$ is more accurate than calculating the exact form in floating-point arithmetic.

Listing 3. Numerically stable sampling of the vMF distribution. Since HLSL does not have a built-in fma function for single precision, we use the mad function instead. For the implementation details of expm1, log1p, and orthonormal_basis functions, please see Listings 4, 5, and 6, respectively.

```
float3 sample_vmf(float2 rand, float3 axis, float sharpness) {
 float phi = 2.0f * M_PI * rand.x;
 float THRESHOLD = FLT_EPSILON / 4.0f;
 float r = sharpness > THRESHOLD ? log1p(rand.y * expm1(-2.0f * sharpness)) / sharpness
                                 : -2.0f * rand.y;
 float cos_theta = 1.0f + r;
 float sin_theta = sqrt(-mad(r, r, 2.0f * r));
 float3 dir = {cos(phi) * sin_theta, sin(phi) * sin_theta, cos_theta};
 float3x3 frame = orthonormal_basis(axis);
 return mul(dir, frame);
}
```

Listing 4. $\mathrm{expm1}(x) = \exp(x) - 1$ with cancellation of rounding errors [Higham 2002] (HLSL). Since HLSL does not have a built-in expm1 function unlike C++, we use this implementation as a workaround.

```
float expm1(float x) {
 float u = exp(x);
 if (u == 1.0f) { return x; }
 float y = u - 1.0f;
 if (abs(x) < 1.0f) { return x * y / log(u); }
 return y;
}
```

Listing 5. $\mathrm{log1p}(x) = \log(x + 1)$ with cancellation of rounding errors [Goldberg 1991] (HLSL). Since HLSL does not have a built-in log1p function unlike C++, we use this implementation as a workaround. For this classic algorithm, aggressive compiler optimization must be disabled for floating points.

```
float log1p(float x) {
 // For this algorithm, we must prevent compilers from optimizing (x + 1) - 1 to x.
 volatile float u = x + 1.0f;
 if (u == 1.0f) { return x; }
 float y = log(u);
 if (x < 1.0f) { return x * y / (u - 1.0f); }
 return y;
}
```

Listing 6. Building of an orthonormal basis [Duff et al. 2017] (HLSL). We use this basis for the local frame of the vMF distribution.

```
float3x3 orthonormal_basis(float3 axis) {
 float s = axis.z >= 0.0f ? 1.0f : -1.0f;
 float c = -1.0f / (s + axis.z);
 float b = axis.x * axis.y * c;
 float3 b1 = {1.0f + s * axis.x * axis.x * c, s * b, -s * axis.x};
 float3 b2 = {b, s + axis.y * axis.y * c, -axis.y};
 return float3x3(b1, b2, axis);
}
```

References

Honghao Dong, Guoping Wang, and Sheng Li. 2023. Neural Parametric Mixtures for Path Guiding. In *SIGGRAPH '23 Conference Proceedings*. Article 29, 10 pages. https://doi.org/10.1145/3588432.3591533

Tom Duff, James Burgess, Per Christensen, Christophe Hery, Andrew Kensler, Max Liani, and Ryusuke Villemin. 2017. Building an Orthonormal Basis, Revisited. *J. Comput. Graph. Tech.* 6, 1 (2017), 1–8. http://jcgt.org/published/0006/01/01/

Ronald Aylmer Fisher. 1953. Dispersion on a sphere. *Proc. R. Soc. Lond. Ser. A* 217, 1130 (1953), 295–305. https://doi.org/10.1098/rspa.1953.0064

Daniel Frisch and Uwe D. Hanebeck. 2023. Deterministic Von Mises–Fisher Sampling on the Sphere Using Fibonacci Lattices. In *SDF-MFI '23*. 1–8. https://doi.org/10.1109/SDF-MFI59545.2023.10361396

David Goldberg. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Comput. Surv.* 23, 1 (1991), 5–48. https://doi.org/10.1145/103162.103163

Nicholas J. Higham. 2002. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics.

Wenzel Jakob. 2012. *Numerically stable sampling of the von Mises Fisher distribution on S^2 (and other tricks)*. Technical Report. https://www.mitsuba-renderer.org/~wenzel/files/vmf.pdf

Sungkyu Jung. 2009. *Generating von Mises Fisher distribution on the unit sphere (S2)*. Technical Report. U. Pittsburgh. https://www.stat.pitt.edu/sungkyu/software/randvonMisesFisher3.pdf

Lukas Ruppert, Sebastian Herholz, and Hendrik P. A. Lensch. 2020. Robust fitting of parallax-aware mixtures for path guiding. *ACM Trans. Graph.* 39, 4, Article 147 (2020), 15 pages. https://doi.org/10.1145/3386569.3392421

Yu-Ting Tsai and Zen-Chung Shih. 2006. All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation. *ACM Trans. Graph.* 25, 3 (2006), 967–976. https://doi.org/10.1145/1141911.1141981