Importance Sampling Microfacet-Based BSDFs with the Distribution of Visible Normals

Supplemental Material 1/2

Analytic Sampling for Beckmann and GGX Distributions

Eric Heitz[1] and Eugene d'Eon[2]

[1]INRIA ; CNRS ; Univ. Grenoble Alpes
[2]The Jig Lab

Abstract

In this document we describe our analytic importance sampling strategy of the distribution of visible normals with Beckmann and GGX distributions with the Smith microsurface profile.

Contents

1 Generic Sampling Algorithm

Generic Algorithm. The idea is that if P^{22} is shape-invariant, and if there is a way to sample $(x_{\tilde{m}}, y_{\tilde{m}})$ with the PDF $P^{22}_{\omega_i}$ with roughness $\alpha = 1$, then it is possible to use it to sample the same distribution with any parameters α_x and α_y. We recall Algorithm 4 from the paper:

Algorithm 1 Sample $D_{\omega_i}(\omega_m)$ (Smith profile)

$\omega_i' \leftarrow \mathcal{S}^{\alpha_x,\alpha_y}(\omega_i)$ ▷ 1. stretch

$x_{\tilde{m}} \leftarrow {C^{2-}_{\omega_i'}}^{-1}(\mathcal{U}_1, 1, 1)$ ▷ 2.a. sample $P^{2-}_{\omega_i'}(x_{\tilde{m}}, 1, 1)$

$y_{\tilde{m}} \leftarrow {C^{2|2}}^{-1}(\mathcal{U}_2|x_{\tilde{m}}, 1, 1)$ ▷ 2.b. sample $P^{2|2}(y_{\tilde{m}}|x_{\tilde{m}}, 1, 1)$

$\begin{pmatrix} x_{\tilde{m}} \\ y_{\tilde{m}} \end{pmatrix} \leftarrow \begin{pmatrix} \cos\phi_i' & -\sin\phi_i' \\ \sin\phi_i' & \cos\phi_i' \end{pmatrix} \begin{pmatrix} x_{\tilde{m}} \\ y_{\tilde{m}} \end{pmatrix}$ ▷ 3. rotate

$\begin{pmatrix} x_{\tilde{m}} \\ y_{\tilde{m}} \end{pmatrix} \leftarrow \begin{pmatrix} \alpha_x\, x_{\tilde{m}} \\ \alpha_y\, y_{\tilde{m}} \end{pmatrix}$ ▷ 4. unstretch

$\omega_m \leftarrow \frac{(-x_{\tilde{m}}, -y_{\tilde{m}}, 1)}{\sqrt{x_{\tilde{m}}^2 + y_{\tilde{m}}^2 + 1}}$ ▷ 5. compute normal

Objectives. In this document, we show how to compute the inverse CDFs

$$x_{\tilde{m}} = {C^{2-}_{\omega_i'}}^{-1}(\mathcal{U}_1, 1, 1),$$
$$y_{\tilde{m}} = {C^{2|2}}^{-1}(\mathcal{U}_2|x_{\tilde{m}}, 1, 1),$$

analytically with Beckmann (Algorithm 2 of this document) and GGX (Algorithm 3 of this document) distributions.

1.1 C++ Implementation

We describe a C++ implementation of the paper's Algorithm 4 and of the associated precomputations steps. We implement a virtual class **VNDFSampler** (Visible Normal Distribution Function Sampler). The API contains one public method **sample**(), which takes as arguments the incident direction ω_i, anisotropic roughness α_x and α_y, and two uniform random numbers $\mathcal{U}_1$ and $\mathcal{U}_2$. It generates a microfacet normal ω_m with PDF $D_{\omega_i}(\omega_m, \alpha_x, \alpha_y)$.

A child class must implement the protected virtual method **sample11**(), which returns a slope vector $(x_{\tilde{m}}, y_{\tilde{m}})$ sampled from the stretched configuration with roughness $\alpha = 1$, i.e. from distribution $P^{22}_{\omega'_i}(x_{\tilde{m}}, y_{\tilde{m}}, 1, 1)$. The implementation of the child classes for Beckmann and GGX distributions are provided in Sections 3 and 4.

```
class VNDFSampler
{
public:
        // generate a view dependent normal
        void sample(
                        // input
                        const double omega_i[3], // incident direction
                        const double alpha_x, const double alpha_y, // anisotropic roughness
                        const double U1, const double U2, // random numbers
                        // output
                        double omega_m[3] // normal
                        );

protected:
// method implemented by child class
        // inverse slope CDF
        virtual void sample11(
                        // input
                        const double theta_i, // incident direction
                        double U1, double U2, // random numbers
                        // output
                        double& slope_x, double& slope_y // slopes
                        ) = 0;
};
```

```
void VNDFSampler::sample(
        // input
        const double omega_i[3], // incident direction
        const double alpha_x, const double alpha_y, // anisotropic roughness
        const double U1, const double U2, // random numbers
        // output
        double omega_m[3]) // micronormal
{
// 1. stretch omega_i
        double omega_i_[3];
        omega_i_[0] = alpha_x * omega_i[0];
        omega_i_[1] = alpha_y * omega_i[1];
        omega_i_[2] = omega_i[2];
        // normalize
        double inv_omega_i = 1.0 / sqrt(omega_i_[0]*omega_i_[0] + omega_i_[1]*omega_i_[1] + omega_i_[2]*omega_i_[2]);
        omega_i_[0] *= inv_omega_i;
        omega_i_[1] *= inv_omega_i;
        omega_i_[2] *= inv_omega_i;
        // get polar coordinates of omega_i_
        double theta_ = 0.0;
        double phi_ = 0.0;
        if (omega_i_[2] < 0.99999)
        {
                theta_ = acos(omega_i_[2]);
                phi_ = atan2(omega_i_[1], omega_i_[0]);
        }

// 2. sample P22_{omega_i}(x_slope, y_slope, 1, 1)
        double slope_x, slope_y;
        sample11(       theta_,
                        U1, U2,
                        slope_x, slope_y);

// 3. rotate
        double tmp = cos(phi_)*slope_x - sin(phi_)*slope_y;
        slope_y = sin(phi_)*slope_x + cos(phi_)*slope_y;
        slope_x = tmp;

// 4. unstretch
        slope_x = alpha_x * slope_x;
        slope_y = alpha_y * slope_y;

// 5. compute normal
        double inv_omega_m = sqrt(slope_x*slope_x + slope_y*slope_y + 1.0);
        omega_m[0] = -slope_x/inv_omega_m;
        omega_m[1] = -slope_y/inv_omega_m;
        omega_m[2] = 1.0/inv_omega_m;
}
```

2 Sparing Random Numbers

We use several conditional branchings in our sampling algorithms. But we can spare the random numbers used for these branchings. Indeed, each time a random number $\mathcal{U}$ is used in a branching with probability p, its value can be used again by renormalizing it:

if $\mathcal{U} < p$ **then**
  $\mathcal{U} \leftarrow \frac{\mathcal{U}}{p}$ ▷ renormalize $\mathcal{U}$
  do something with $\mathcal{U}$
else
  $\mathcal{U} \leftarrow \frac{\mathcal{U}-p}{1-p}$ ▷ renormalize $\mathcal{U}$
  do something else with $\mathcal{U}$
end if

3 Sampling Beckmann Distributions

3.1 The Beckmann Distribution of Slopes

The Beckmann distribution of slopes with $\alpha = 1$ is the 2D Gaussian

$$P^{22}(x_{\tilde{m}}, y_{\tilde{m}}) = \frac{1}{\pi} \exp\left(-x_{\tilde{m}}^2 - y_{\tilde{m}}^2\right).$$

3.2 Sample $P_{\omega_i}^{2-}(x_{\tilde{m}})$

The Marginalized Distribution of Slopes is the 1D Gaussian distribution

$$\begin{aligned} P^{2-}(x_{\tilde{m}}) &= \int_{-\infty}^{+\infty} P^{22}(x_{\tilde{m}}, y_{\tilde{m}})\, dy_{\tilde{m}} \\ &= \frac{1}{\sqrt{\pi}} \exp\left(-x_{\tilde{m}}^2\right), \end{aligned}$$

and its CDF is:

$$\begin{aligned} C^{2-}(x_{\tilde{m}}) &= \int_{-\infty}^{x_{\tilde{m}}} P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} \\ &= \frac{1}{2} + \frac{1}{2} \operatorname{erf}(x_{\tilde{m}}). \end{aligned}$$

The Marginalized Distribution of Visible Slopes is

$$\begin{aligned} P_{\omega_i}^{2-}(x_{\tilde{m}}) &= \frac{(-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}})}{\int_{-\infty}^{+\infty} (-x'_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x'_{\tilde{m}} \sin\theta_i + \cos\theta_i)\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}}} \\ &= \frac{G_1(\omega_i)}{\cos\theta_i} (-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}}), \end{aligned}$$

and the associated CDF is

$$\begin{aligned} C_{\omega_i}^{2-}(x_{\tilde{m}}) &= \frac{G_1(\omega_i)}{\cos\theta_i} \int_{-\infty}^{x_{\tilde{m}}} (-x'_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x'_{\tilde{m}} \sin\theta_i + \cos\theta_i)\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}} \\ &= \frac{G_1(\omega_i)\ \tan\theta_o}{2\sqrt{\pi}} \exp\left(-x_{\tilde{m}}^2\right) + G_1(\omega_i) \left(\frac{1}{2} + \frac{1}{2} \operatorname{erf}(x_{\tilde{m}})\right). \end{aligned}$$

The problem is that this CDF is not directly invertible.

Choosing an interval. The CDF $C^{2-}_{\omega_i}(x_{\tilde{m}})$ is invertible in the interval $]-\infty, -\cot\theta_i]$ and we use the anti-symmetry of the projection factor to invert it in $]-\cot\theta_i, \cot\theta_i]$. We choose randomly the interval $]-\infty, -\cot\theta_i]$ or $]-\cot\theta_i, \cot\theta_i]$ according to their weights given by the CDF value at $-\cot\theta_i$:

---

if $\mathcal{U} \leq C^{2-}_{\omega_i}(-\cot\theta_i)$ **then**
  sample $x_{\tilde{m}} \in]-\infty, -\cot\theta_i]$
else
  sample $x_{\tilde{m}} \in]-\cot\theta_i, \cot\theta_i]$
end if

---

and we have:

$$
\begin{aligned}
&C^{2-}_{\omega_i}(-\cot\theta_i) \\
&= \frac{G_1(\omega_i)}{\cos\theta_i} \int_{-\infty}^{-\cot\theta_i} (-x_{\tilde{m}}\sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} \\
&= \frac{G_1(\omega_i)}{\cos\theta_i} \int_{-\infty}^{\cot\theta_i} (-x_{\tilde{m}}\sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} \\
&- \frac{G_1(\omega_i)}{\cos\theta_i} \int_{-\cot\theta}^{\cot\theta_i} (-x_{\tilde{m}}\sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} \\
&= 1 - \frac{G_1(\omega_i)}{\cos\theta_i} \int_{-\cot\theta}^{\cot\theta_i} (-x_{\tilde{m}}\sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} \\
&= 1 - G_1(\omega_i) \int_{-\cot\theta}^{\cot\theta_i} P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} \\
&= 1 - G_1(\omega_i)\left(C^{2-}(\cot\theta_i) - C^{2-}(-\cot\theta_i)\right) \\
&= 1 - G_1(\omega_i)\,\mathrm{erf}\left(\cot\theta_i\right).
\end{aligned}
$$

Sample $P^{2-}_{\omega_i}(x_{\tilde{m}})$ in $]-\infty, -\cot\theta_i]$.

If $x_{\tilde{m}} \in]-\infty, -\cot\theta_i]$, we have $-x_{\tilde{m}}\sin\theta_i + \cos\theta_i > 0$, so we can drop the heaviside term and separate the integral:

$$\begin{aligned}
C^{2-}_{\omega_i}(x_{\tilde{m}}) &= \frac{G_1(\omega_i)}{\cos\theta_i}\int_{-\infty}^{x_{\tilde{m}}} (-x'_{\tilde{m}}\sin\theta_i + \cos\theta_i)\ \chi^+(-x'_{\tilde{m}}\sin\theta_i + \cos\theta_i)\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}} \\
&= \frac{G_1(\omega_i)}{\cos\theta_i}\int_{-\infty}^{x_{\tilde{m}}} (-x'_{\tilde{m}}\sin\theta_i + \cos\theta_i)\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}} \\
&= \frac{G_1(\omega_i)}{\cos\theta_i}\int_{-\infty}^{x_{\tilde{m}}} (-x'_{\tilde{m}}\sin\theta_i)\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}} + \frac{G_1(\omega_i)}{\cos\theta_i}\int_{-\infty}^{x_{\tilde{m}}} \cos\theta_i\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}} \\
&= \frac{G_1(\omega_i)}{\cos\theta_i}(f_1(x) + f_2(x)),
\end{aligned}$$

where

$$\begin{aligned}
f_1(x_{\tilde{m}}) &= \int_{-\infty}^{x_{\tilde{m}}} (-x'_{\tilde{m}}\sin\theta_i)\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}}, \\
f_2(x_{\tilde{m}}) &= \int_{-\infty}^{x_{\tilde{m}}} \cos\theta_i\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}}.
\end{aligned}$$

We can sample with PDF $P^{2-}_{\omega_i}(x_{\tilde{m}})$ by choosing either f_1 or f_2 according to their respective weight ratios, and inverting it. Their weights are given by:

$$\begin{aligned}
w_1 = f_1(-\cot\theta_i) &= \int_{-\infty}^{-\cot\theta_i} (-x_{\tilde{m}}\sin\theta_i)\, P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} = \frac{\sin\theta_i}{2\sqrt{\pi}}\ \exp\left(-\cot^2\theta_i\right), \\
w_2 = f_2(-\cot\theta_i) &= \int_{-\infty}^{-\cot\theta_i} \cos\theta_i\, P^{2-}(x_{\tilde{m}})\, dx_{\tilde{m}} = \cos\theta_i\ \left(\frac{1}{2} - \frac{1}{2}\operatorname{erf}(\cot\theta_i)\right).
\end{aligned}$$

The branching probability p is given by:

$$p = \frac{w_1}{w_1 + w_2},$$

and the functions to inverse are f_1 and f_2 divided by their respective weights to transform them into PDFs before the inversion:

$$\begin{aligned}
\left(\frac{1}{w_1} f_1\right)^{-1}(\mathcal{U}) &= -\sqrt{-\log\left(\mathcal{U}\ \exp\left(-\cot^2\theta_i\right)\right)} \\
\left(\frac{1}{w_2} f_2\right)^{-1}(\mathcal{U}) &= \operatorname{erf}^{-1}\left(\mathcal{U} - 1 - \mathcal{U}\operatorname{erf}(\cot\theta_i)\right).
\end{aligned}$$

The algorithm we use to sample $P^{2-}_{\omega_i}(x_{\tilde{m}})$ in $]-\infty, -\cot\theta_i]$ is thus:

$w_1 \leftarrow \frac{\sin\theta_i}{2\sqrt{\pi}}\ \exp\left(-\cot^2\theta_i\right)$
$w_2 \leftarrow \cos\theta_i\ \left(\frac{1}{2} - \frac{1}{2}\operatorname{erf}\left(\cot\theta_i\right)\right)$
$p = \frac{w_1}{w_1+w_2}$
if $\mathcal{U} \leq p$ **then**
  $\mathcal{U} \leftarrow \frac{\mathcal{U}}{p}$ ▷ renormalize $\mathcal{U}$
  $x_{\tilde{m}} \leftarrow -\sqrt{-\log\left(\mathcal{U}\ \exp\left(-\cot^2\theta_i\right)\right)}$
else
  $\mathcal{U} \leftarrow \frac{\mathcal{U}-p}{1-p}$ ▷ renormalize $\mathcal{U}$
  $x_{\tilde{m}} \leftarrow \operatorname{erf}^{-1}\left(\mathcal{U} - 1 - \mathcal{U}\operatorname{erf}\left(\cot\theta_i\right)\right)$
end if

Sample $P^{2-}_{\omega_i}(x_{\tilde{m}})$ in $]-\cot\theta_i, \cot\theta_i]$.

The PDF of a *couple* of slope values $(-x_{\tilde{m}}, x_{\tilde{m}})$ in the interval $]-\cot\theta_i, \cot\theta_i]$ is:

$$\left(x_{\tilde{m}}\sin\theta_i + \cos\theta_i\right) P^{2-}(-x_{\tilde{m}}) + \left(-x_{\tilde{m}}\sin\theta_i + \cos\theta_i\right) P^{2-}(-x_{\tilde{m}}) = 2\ \cos\theta_i\ P^{2-}(x_{\tilde{m}}).$$

The idea is similar to the sampling scheme for the V-cavity profile. We start by generate a couple $(-x_{\tilde{m}}, x_{\tilde{m}})$ with PDF $P^{2-}(x_{\tilde{m}})$ in the interval $]-\cot\theta_i, \cot\theta_i]$:

$$\begin{aligned} x_{\tilde{m}} &= C^{2--1}\left((1-\mathcal{U})\, C^{2-}(-\cot\theta_i) + \mathcal{U}\, C^{2-}(\cot\theta_i)\right) \\ &= \operatorname{erf}^{-1}\left((-1+2\mathcal{U})\operatorname{erf}\left(\cot\theta_i\right)\right), \end{aligned}$$

and then we choose between $-x_{\tilde{m}}$ and $x_{\tilde{m}}$ with a probability proportionnal to their projected areas $\left(x_{\tilde{m}}\sin\theta_i + \cos\theta_i\right)$ and $\left(-x_{\tilde{m}}\sin\theta_i + \cos\theta_i\right)$. In practice, we swap $x_{\tilde{m}}$ and $-x_{\tilde{m}}$ with probability:

$$\begin{aligned} p &= \frac{-x_{\tilde{m}}\ \sin\theta_i + \cos\theta_i}{-x_{\tilde{m}}\ \sin\theta_i + \cos\theta_i + x_{\tilde{m}}\ \sin\theta_i + \cos\theta_i} \\ &= \frac{-x_{\tilde{m}}\ \sin\theta_i + \cos\theta_i}{2\ \cos\theta_i}. \end{aligned}$$

The algorithm we use to sample $P^{2-}_{\omega_i}(x_{\tilde{m}})$ in $]-\cot\theta_i, \cot\theta_i]$ is thus:

$x_{\tilde{m}} \leftarrow \operatorname{erf}^{-1}\left((-1+2\mathcal{U}_1)\operatorname{erf}\left(\cot\theta_i\right)\right)$
if $\mathcal{U}_2 > \frac{-x\ \sin\theta_i+\cos\theta_i}{2\ \cos\theta_i}$ **then**
  $x_{\tilde{m}} \leftarrow -x_{\tilde{m}}$
end if

Note that we need 2 random numbers $\mathcal{U}_1$ and $\mathcal{U}_2$ in this algorithm. In the final algorithm, we renormalize $\mathcal{U}_2$ which is used here for branching only and use it again in the sampling of $P^{2|2}(y_{\tilde{m}}|x_{\tilde{m}})$.

3.3 Sample $P^{2|2}(y_{\tilde{m}}|x_{\tilde{m}})$

The conditional PDF of slopes is the 1D Gaussian distribution

$$\begin{aligned} P^{2|2}(y_{\tilde{m}}|x_{\tilde{m}}) &= \frac{P^{22}(x_{\tilde{m}}, y_{\tilde{m}})}{\int_{-\infty}^{+\infty} P^{22}(x_{\tilde{m}}, y_{\tilde{m}})\, dy_{\tilde{m}}} \\ &= \frac{1}{\sqrt{\pi}} \exp\left(-y_{\tilde{m}}^2\right), \end{aligned}$$

and the associated CDF is

$$\begin{aligned} C^{2|2}(y_{\tilde{m}}|x_{\tilde{m}}) &= \int_{-\infty}^{y_{\tilde{m}}} P^{2-}(y'_{\tilde{m}}|x_{\tilde{m}})\, dy'_{\tilde{m}} \\ &= \frac{1}{2} + \frac{1}{2} \operatorname{erf}(y_{\tilde{m}}), \end{aligned}$$

which can be inverted:

$$\begin{aligned} \mathcal{U} &= C^{2|2}(y_{\tilde{m}}|x_{\tilde{m}}) \\ \Rightarrow y_{\tilde{m}} &= \operatorname{erf}^{-1}(2\mathcal{U} - 1). \end{aligned}$$

3.4 Sample $P^{22}_{\omega_i}(x_{\tilde{m}}, y_{\tilde{m}}, 1, 1)$ for Beckmann Distributions

The complete algorithm is

Algorithm 2 Sample $P^{22}_{\omega_i}(x_{\tilde{m}}, y_{\tilde{m}}, 1, 1)$ for Beckmann distributions

$C \leftarrow 1 - G_1(\omega_i) \operatorname{erf}(\cot \theta_i)$

if $\mathcal{U}_1 \leq C$ **then** ▷ sample $P^{22}_{\omega_i}(x_{\tilde{m}})$ in $]-\infty, -\cot\theta_i]$

  $\mathcal{U}_1 \leftarrow \frac{\mathcal{U}}{C}$ ▷ renormalize $\mathcal{U}_1$

  $w_1 \leftarrow \frac{\sin\theta_i}{2\sqrt{\pi}} \exp(-\cot^2\theta_i)$

  $w_2 \leftarrow \cos\theta_i \left(\frac{1}{2} - \frac{1}{2}\operatorname{erf}(\cot\theta_i)\right)$

  $p = \frac{w_1}{w_1 + w_2}$

  if $\mathcal{U}_1 \leq p$ **then**

    $\mathcal{U}_1 \leftarrow \frac{\mathcal{U}}{p}$ ▷ renormalize $\mathcal{U}_1$

    $x_{\tilde{m}} \leftarrow -\sqrt{-\log(U_1 \exp(-\cot^2\theta_i))}$

  else

    $\mathcal{U}_1 \leftarrow \frac{\mathcal{U}-p}{1-p}$ ▷ renormalize $\mathcal{U}_1$

    $x_{\tilde{m}} \leftarrow \operatorname{erf}^{-1}(\mathcal{U}_1 - 1 - \mathcal{U}_1 \operatorname{erf}(\cot\theta_i))$

  end if

else ▷ sample $P^{22}_{\omega_i}(x_{\tilde{m}})$ in $]-\cot\theta_i, \cot\theta_i]$

  $\mathcal{U}_1 \leftarrow \frac{\mathcal{U}_1 - C}{1-C}$ ▷ renormalize $\mathcal{U}_1$

  $x_{\tilde{m}} \leftarrow \operatorname{erf}^{-1}((-1 + 2\mathcal{U}_1)\operatorname{erf}(\cot\theta_i))$

  if $\mathcal{U}_2 > \frac{-x_{\tilde{m}}\sin\theta_i + \cos\theta_i}{2\cos\theta_i}$ **then**

    $\mathcal{U}_2 \leftarrow \frac{\mathcal{U}_2 - p}{1-p}$ ▷ renormalize $\mathcal{U}_2$

    $x_{\tilde{m}} \leftarrow -x_{\tilde{m}}$

  else

    $\mathcal{U}_2 \leftarrow \frac{\mathcal{U}_2}{p}$ ▷ renormalize $\mathcal{U}_2$

  end if

end if

$y_{\tilde{m}} \leftarrow \operatorname{erf}^{-1}(2\mathcal{U}_2 - 1)$ ▷ sample $P^{22}(y_{\tilde{m}}|x_{\tilde{m}})$

3.5 C++ Implementation

The C++ implementation of Algorithm 2 is

```
class VNDFSamplerBeckmann : public VNDFSampler
{
protected:
	virtual void sample11(
			// input
			const double theta_i, // incident direction
			double U1, double U2, // random numbers
			// output
			double& slope_x, double& slope_y // slopes
			)
	{
		// special case (normal incidence)
		if(theta_i < 0.0001)
		{
			const double r = sqrt(-log(U1));
			const double phi = 6.28318530718 * U2;
			slope_x = r * cos(phi);
			slope_y = r * sin(phi);
			return;
		}

		// precomputations
		const double sin_theta_i = sin(theta_i);
		const double cos_theta_i = cos(theta_i);
		const double tan_theta_i = sin_theta_i/cos_theta_i;
		const double a = 1.0 / tan_theta_i;
		const double erf_a = erf(a);
		const double exp_a2 = exp(-a*a);
		const double SQRT_PI_INV = 0.56418958354;
		const double Lambda = 0.5*(erf_a-1) + 0.5*SQRT_PI_INV*exp_a2/a;
		const double G1 = 1.0 / (1.0 + Lambda); // masking
		const double C= 1.0 - G1 * erf_a;

		// sample slope X
		if(U1 < C)
		{
			// rescale U1
			U1 = U1 / C;

			const double w_1 = 0.5 * SQRT_PI_INV * sin_theta_i * exp_a2;
			const double w_2 = cos_theta_i * (0.5 - 0.5*erf_a);
			const double p = w_1 / (w_1+w_2);

			if(U1 < p)
			{
				U1 = U1 / p;
				slope_x = -sqrt(-log(U1*exp_a2));
			}
			else
			{
				U1 = (U1-p) / (1.0-p);
				slope_x = erfinv(U1-1.0-U1*erf_a);
			}
		}
		else
		{
			// rescale U1
			U1 = (U1-C) / (1.0-C);

			slope_x = erfinv((-1.0+2.0*U1)*erf_a);

			const double p = (-slope_x*sin_theta_i + cos_theta_i) / (2.0*cos_theta_i);
			if (U2 > p)
			{
				slope_x = -slope_x;
				U2 = (U2-p) / (1.0-p);
			}
			else
				U2 = U2 / p;
		}

		// sample slope Y
		slope_y = erfinv(2.0*U2-1.0);
	}
};
```

3.6 Results

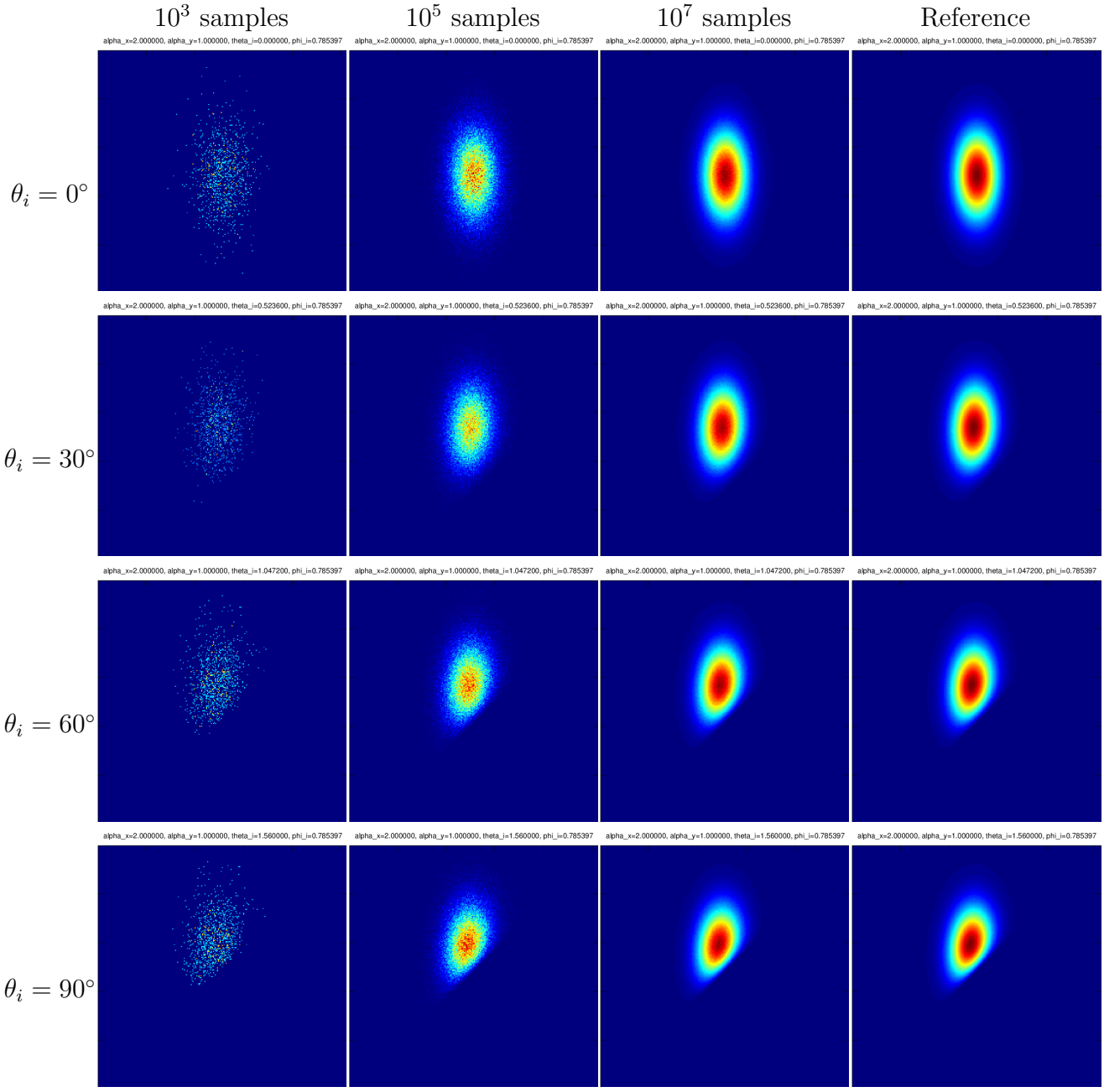

Figure 1: Sampling the distribution of visible slopes $P^{22}_{\omega_i}$ using our C++ implementation, where P^{22} is a Beckmann distribution with anisotropic roughness $\alpha_x = 2$ and $\alpha_y = 1$. The azimuthal angle of incidence is $\phi_i = 45°$.

4 Sampling GGX Distributions

4.1 The GGX Distribution of Slopes

The GGX distributions of slopes with $\alpha = 1$ is

$$P^{22}(x_{\tilde{m}}, y_{\tilde{m}}) = \frac{1}{\pi} \frac{1}{(1 + x_{\tilde{m}}^2 + y_{\tilde{m}}^2)^2}.$$

4.2 Sample $P_{\omega_i}^{2-}(x_{\tilde{m}})$

The Marginalized Distribution of Slopes is

$$\begin{aligned} P^{2-}(x_{\tilde{m}}) &= \int_{-\infty}^{\infty} P^{22}(x_{\tilde{m}}, y_{\tilde{m}})\, dy_{\tilde{m}} \\ &= \int_{-\infty}^{\infty} \frac{1}{\pi} \frac{1}{(1 + x_{\tilde{m}}^2 + y_{\tilde{m}}^2)^2}\, dy_{\tilde{m}} \\ &= \frac{1}{2\,(x_{\tilde{m}}^2 + 1)^{3/2}}. \end{aligned}$$

The Marginalized Distribution of Visible Slopes is

$$\begin{aligned} P_{\omega_i}^{2-}(x_{\tilde{m}}) &= \frac{(-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}})}{\int_{-\infty}^{+\infty} (-x'_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x'_{\tilde{m}} \sin\theta_i + \cos\theta_i)\, P^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}}} \\ &= \frac{G_1(\omega_i)}{\cos\theta_i} (-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\, P^{2-}(x_{\tilde{m}}) \\ &= \frac{G_1(\omega_i)}{\cos\theta_i} (-x_{\tilde{m}} \sin\theta_i + \cos\theta_i)\ \chi^+(-x_{\tilde{m}} \sin\theta_i + \cos\theta_i) \frac{1}{2\,(x_{\tilde{m}}^2 + 1)^{3/2}}, \end{aligned}$$

and the associated CDF is:

$$\begin{aligned} C_{\omega_i}^{2-}(x_{\tilde{m}}) &= \int_{-\infty}^{x_{\tilde{m}}} P_{\omega_i}^{2-}(x'_{\tilde{m}})\, dx'_{\tilde{m}} \\ &= \frac{G_1(\omega_i)}{2\cos\theta_i} \left(\frac{\sin\theta_i + x_{\tilde{m}}\ \cos\theta_i}{\sqrt{1 + x_{\tilde{m}}^2}} + \cos\theta_i \right). \end{aligned}$$

Inversion of the CDF. We solve

$$\begin{aligned} \mathcal{U} = C_{\omega_i}^{2-}(x_{\tilde{m}}) &= \frac{G_1(\omega_i)}{2\cos\theta_i} \left(\frac{\sin\theta_i + x_{\tilde{m}}\ \cos\theta_i}{\sqrt{1 + x_{\tilde{m}}^2}} + \cos\theta_i \right) \\ \Rightarrow \frac{2\mathcal{U}}{G_1} - 1 &= \frac{\tan\theta_i + x_{\tilde{m}}}{\sqrt{1 + x_{\tilde{m}}^2}} \\ \Rightarrow A &= \frac{B + x_{\tilde{m}}}{\sqrt{1 + x_{\tilde{m}}^2}}, \end{aligned}$$

with $A = \frac{2\mathcal{U}}{G_1} - 1$ and $B = \tan\theta_i$. The solution is

$$x_{\tilde{m}} = \begin{cases} x_{\tilde{m}1} & \text{if } A < 0 \text{ or } x_{\tilde{m}2} > \cot\theta_i \\ x_{\tilde{m}2} & \text{else,} \end{cases}$$

where

$$x_{\tilde{m}1} = \frac{B}{A^2-1} - \sqrt{\frac{B^2}{(A^2-1)^2} - \frac{A^2-B^2}{A^2-1}},$$
$$x_{\tilde{m}2} = \frac{B}{A^2-1} + \sqrt{\frac{B^2}{(A^2-1)^2} - \frac{A^2-B^2}{A^2-1}}.$$

The algorithm we use to sample $P^{2-}_{\omega_i}(x_{\tilde{m}})$ is thus

---

$A \leftarrow \frac{2\mathcal{U}}{G_1(\omega_i)} - 1$
$B = \tan\theta_i$
$x_{\tilde{m}1} \leftarrow \frac{B}{A^2-1} - \sqrt{\frac{B^2}{(A^2-1)^2} - \frac{A^2-B^2}{A^2-1}}$
$x_{\tilde{m}2} \leftarrow \frac{B}{A^2-1} + \sqrt{\frac{B^2}{(A^2-1)^2} - \frac{A^2-B^2}{A^2-1}}$
if $A < 0$ or $x_{\tilde{m}2} > \cot\theta_i$ **then**
    $x_{\tilde{m}} \leftarrow x_{\tilde{m}1}$
else
    $x_{\tilde{m}} \leftarrow x_{\tilde{m}2}$
end if

4.3 Sample $P^{2|2}(y_{\tilde{m}}|x_{\tilde{m}})$

The conditional PDF is given by:

$$P^{2|2}(y_{\tilde{m}}|x_{\tilde{m}}) = \frac{\frac{1}{(1+x_{\tilde{m}}^2+y_{\tilde{m}}^2)^2}}{\int_{-\infty}^{\infty} \frac{1}{(1+x_{\tilde{m}}^2+{y'_{\tilde{m}}}^2)^2}\, dy'_{\tilde{m}}},$$

and the associated conditional CDF is:

$$C^{2|2}(y_{\tilde{m}}|x_{\tilde{m}}) = \frac{\int_{-\infty}^{y_{\tilde{m}}} \frac{1}{(1+x_{\tilde{m}}^2+{y'_{\tilde{m}}}^2)^2} dy'_{\tilde{m}}}{\int_{-\infty}^{\infty} \frac{1}{(1+x_{\tilde{m}}^2+{y'_{\tilde{m}}}^2)^2}\, dy'_{\tilde{m}}}.$$

Unfortunately, this CDF is not directly invertible.

Change of variable. With a change of variable we make the formulation independent of $x_{\tilde{m}}$:

$$\begin{aligned} C^{22}(y_{\tilde{m}}|x_{\tilde{m}}) &= \frac{\int_{-\infty}^{y_{\tilde{m}}} \frac{1}{\left(1+x_{\tilde{m}}^2+{y'_{\tilde{m}}}^2\right)^2}\, dy'_{\tilde{m}}}{\int_{-\infty}^{\infty} \frac{1}{\left(1+x_{\tilde{m}}^2+{y'_{\tilde{m}}}^2\right)^2}\, dy'_{\tilde{m}}} \\ &= \frac{\int_{-\infty}^{y_{\tilde{m}}} \frac{1}{\left(1+\frac{{y'_{\tilde{m}}}^2}{a}\right)^2}\, dy'_{\tilde{m}}}{\int_{-\infty}^{\infty} \frac{1}{\left(1+\frac{{y'_{\tilde{m}}}^2}{a}\right)^2}\, dy'_{\tilde{m}}} \end{aligned}$$

with $a = 1 + x_{\tilde{m}}^2$. Let be $z' = y'_{\tilde{m}}/\sqrt{a}$ and $z = y_{\tilde{m}}/\sqrt{a}$, and we get

$$\begin{aligned} C^{22}(y_{\tilde{m}}|x_{\tilde{m}}) &= C_z(z) \\ &= \frac{\int_{-\infty}^{z} \frac{1}{(1+z'^2)^2}\, dz'}{\int_{-\infty}^{\infty} \frac{1}{(1+z'^2)^2}\, dz'}. \end{aligned}$$

The sampling of z does not depend on $x_{\tilde{m}}$ anymore. We fit the inverse CDF C_z^{-1} of z and we solve

$$\begin{aligned} \mathcal{U} = C^{22}(y_{\tilde{m}}|x_{\tilde{m}}) &= C_z(z) \\ \Rightarrow z &= C_z^{-1}(\mathcal{U}) \\ \Rightarrow y_{\tilde{m}} &= C_z^{-1}(\mathcal{U})\sqrt{1+x_{\tilde{m}}^2} \end{aligned}$$

We use a rational polynomial to fit C_z^{-1}:

$$C_z^{-1}(\mathcal{U}) = \frac{0.46341\,\mathcal{U} - 0.73369\,\mathcal{U}^2 + 0.27385\,\mathcal{U}^3}{0.597999 - \mathcal{U} + 0.309420\,\mathcal{U}^2 + 0.093073\,\mathcal{U}^3},$$

in the positive interval $[0, +\infty[$. To use this fit in the interval $]-\infty, +\infty[$, if $\mathcal{U} < \frac{1}{2}$, we compute $z = C_z^{-1}\left(2\left(\mathcal{U} - \frac{1}{2}\right)\right)$, else if $\mathcal{U} \geq \frac{1}{2}$ we compute $z = C_z^{-1}\left(2\left(\frac{1}{2} - \mathcal{U}\right)\right)$. In this way, we can sample $z \in]-\infty, \infty[$. This fit is plotted in Figure 2.

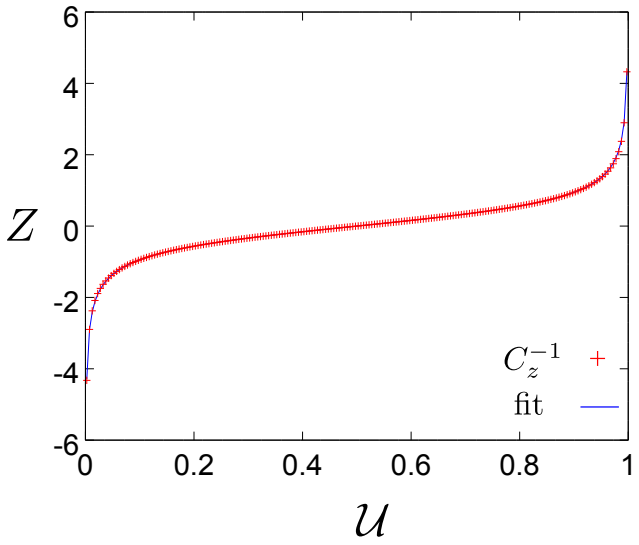

Figure 2: Plot of our fit of the inverse CDF C_z^{-1}.

The algorithm we use to sample $P^{2|2}(y_{\tilde{m}}|x_{\tilde{m}})$ is thus

---

if $\mathcal{U} \leq \frac{1}{2}$ **then**
  $s = 1$
  $\mathcal{U} \leftarrow 2\left(\mathcal{U} - \frac{1}{2}\right)$
else
  $s = -1$
  $\mathcal{U} \leftarrow 2\left(\frac{1}{2} - \mathcal{U}\right)$
end if
$z \leftarrow s\,\frac{0.46341\,\mathcal{U} - 0.73369\,\mathcal{U}^2 + 0.27385\,\mathcal{U}^3}{0.597999 - \mathcal{U} + 0.309420\,\mathcal{U}^2 + 0.093073\,\mathcal{U}^3}$
$y_{\tilde{m}} \leftarrow z\sqrt{1 + x_{\tilde{m}}^2}$

4.4 Sample $P^{22}_{\omega_i}(x_{\tilde{m}}, y_{\tilde{m}}, 1, 1)$ for GGX Distributions

Algorithm 3 Sample $P^{22}_{\omega_i}(x_{\tilde{m}}, y_{\tilde{m}}, 1, 1)$ for GGX distributions

$A \leftarrow \frac{2\mathcal{U}_1}{G_1(\omega_i)} - 1$
$B = \tan\theta_i$
$x_{\tilde{m}1} \leftarrow \frac{B}{A^2-1} - \sqrt{\frac{B^2}{(A^2-1)^2} - \frac{A^2-B^2}{A^2-1}}$
$x_{\tilde{m}2} \leftarrow \frac{B}{A^2-1} + \sqrt{\frac{B^2}{(A^2-1)^2} - \frac{A^2-B^2}{A^2-1}}$
if $A < 0$ or $X_2 > \cot\theta_i$ **then**
    $x_{\tilde{m}} \leftarrow x_{\tilde{m}1}$
else
    $x_{\tilde{m}} \leftarrow x_{\tilde{m}2}$
end if

if $\mathcal{U}_2 \leq \frac{1}{2}$ **then**
    $s = 1$
    $\mathcal{U}_2 \leftarrow 2\ \left(\mathcal{U}_2 - \frac{1}{2}\right)$
else
    $s = -1$
    $\mathcal{U}_2 \leftarrow 2\ \left(\frac{1}{2} - \mathcal{U}_2\right)$
end if
$z \leftarrow s\ \frac{0.46341\mathcal{U}_2 - 0.73369\mathcal{U}_2^2 + 0.27385\mathcal{U}_2^3}{0.597999 - \mathcal{U}_2 + 0.309420\mathcal{U}_2^2 + 0.093073\mathcal{U}_2^3}$
$y_{\tilde{m}} \leftarrow z\sqrt{1 + x^2_{\tilde{m}}}$

4.5 C++ Implementation

The C++ implementation of Algorithm 3 is

```
class VNDFSamplerGGX : public VNDFSampler
{
protected:
        virtual void sample11(
                        // input
                        const double theta_i, // incident direction
                        double U1, double U2, // random numbers
                        // output
                        double& slope_x, double& slope_y // slopes
                        )
        {
                // special case (normal incidence)
                if(theta_i < 0.0001)
                {
                        const double r = sqrt(U1/(1-U1));
                        const double phi = 6.28318530718 * U2;
                        slope_x = r * cos(phi);
                        slope_y = r * sin(phi);
                        return;
                }

                // precomputations
                const double tan_theta_i = tan(theta_i);
                const double a = 1 / (tan_theta_i);
                const double G1 = 2 / (1 + sqrt(1.0+1.0/(a*a)));

                // sample slope_x
                const double A = 2.0*U1/G1 - 1.0;
                const double tmp = 1.0 / (A*A-1.0);
                const double B = tan_theta_i;
                const double D = sqrt(B*B*tmp*tmp - (A*A-B*B)*tmp);
                const double slope_x_1 = B*tmp - D;
                const double slope_x_2 = B*tmp + D;
                slope_x = (A < 0 || slope_x_2 > 1.0/tan_theta_i) ? slope_x_1 : slope_x_2;

                // sample slope_y
                double S;
                if(U2 > 0.5)
                {
                        S = 1.0;
                        U2 = 2.0*(U2-0.5);
                }
                else
                {
                        S = -1.0;
                        U2 = 2.0*(0.5-U2);
                }
                const double z = (U2*(U2*(U2*0.27385-0.73369)+0.46341)) / (U2*(U2*(U2*0.093073+0.309420)-1.000000)+0.597999);
                slope_y = S * z * sqrt(1.0+slope_x*slope_x);
        }
};
```

4.6 Results

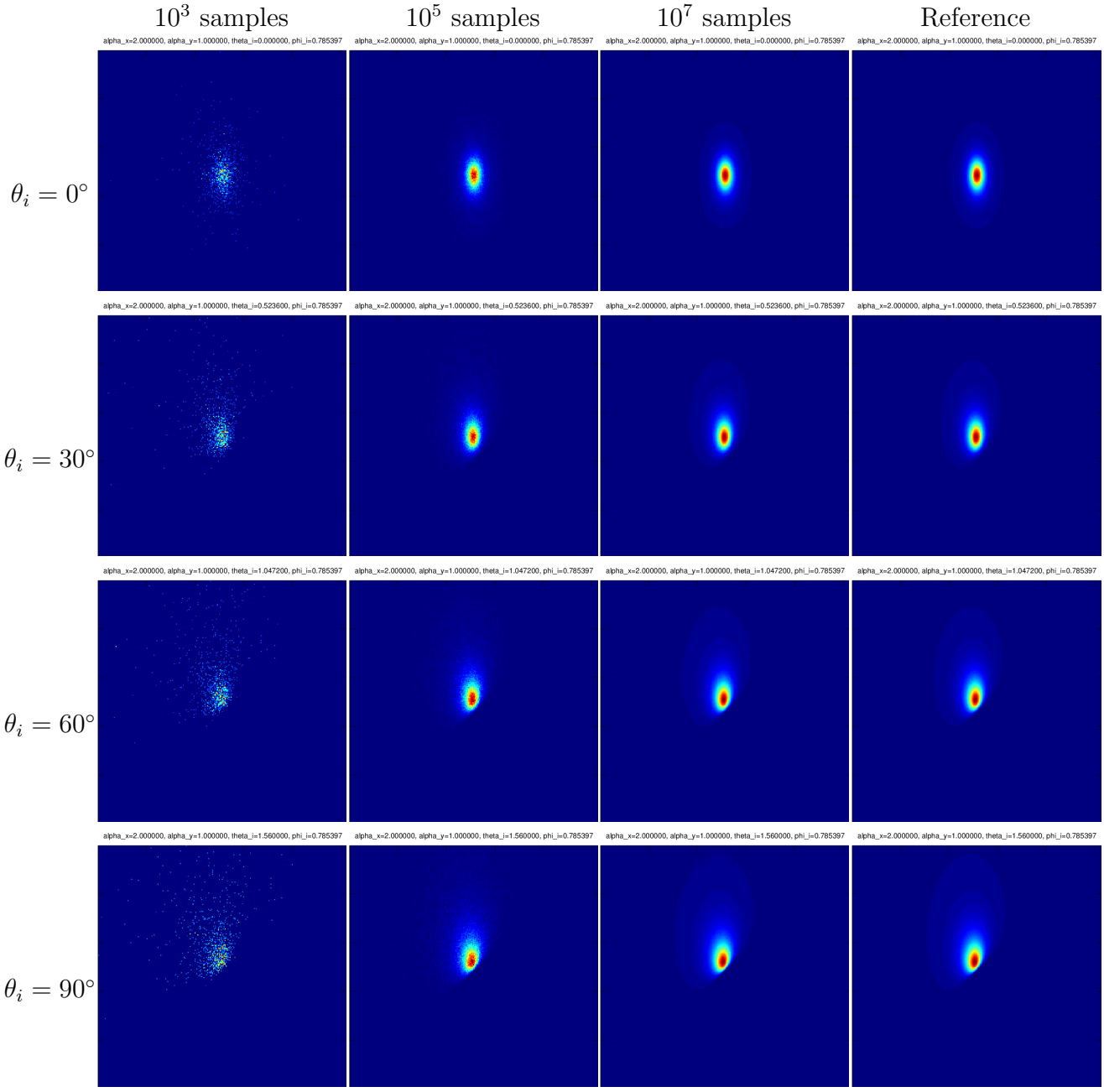

Figure 3: Sampling the distribution of visible slopes $P^{22}_{\omega_i}$ using our C++ implementation, where P^{22} is a GGX distribution with anisotropic roughness $\alpha_x = 2$ and $\alpha_y = 1$. The azimuthal angle of incidence is $\phi_i = 45°$.