Void-and-Cluster Sampling of Large Scattered Data and Trajectories: *Supplementary Material*

Tobias Rapp, Christoph Peters, and Carsten Dachsbacher

1 Void-and-Cluster Sampling

Our void-and-cluster sampling algorithm for scattered data is shown in detail in algorithm 1. We split the sampling algorithm into the initial random sampling, optimization, and the void filling steps.

Algorithm 1 Void-and-cluster sampling algorithm

```
procedure VoidAndCluster(P ⊂ ℝ^d, v : P → V, h_P, q ∈ (0,1])
    h ← (h_P / q)^(1/d)                          ▷ Kernel size for samples
    n ← q|P|                                     ▷ Number of samples
    φ ← Importance(v, h)                         ▷ From entropy or const.
    S, r, ρ_P, λ_S ← InitialRandomSampling(P, φ, h)
    OptimizeSamples(S, r, λ_S, ρ_P, h)
    VoidFilling(S, r, λ_S, ρ_P, h, n)
    return S, 1/φ, r                             ▷ Return samples, weights, and rank
end procedure

procedure InitialRandomSampling(P, φ, h)
    for all p ∈ P do                             ▷ Compute point density ρ_P
        ρ_P ← AddDensity(φ(p), h)
    end for
    S, r ← RandomSampling(φ)
    for all s ∈ S do                             ▷ Compute sample density λ_S
        λ_S ← AddDensity(ρ_P(s)^(-1), h)
    end for
    return S, r, ρ_P, λ_S
end procedure

procedure OptimizeSamples(S, r, λ_S, ρ_P, h)
    while true do
        s_max ← arg max_{s∈S} {λ_S(s)}           ▷ Find tightest cluster
        λ_S ← AddDensity(−ρ_P(s_max)^(-1), h)
        p_min ← arg min_{p∈P\S} {λ_S(p)}         ▷ Find largest void
        λ_S ← AddDensity(ρ_P(p_min)^(-1), h)
        r[p_min] ← r[s_max], r[s_max] ← ∞        ▷ Exchange rank
        if p_min = s_max then
            break
        end if
    end while
end procedure

procedure VoidFilling(S, r, λ_S, ρ_P, h, n)
    for i ← |S|, n do
        p_min ← arg min_{p∈P\S} {λ_S(p)}         ▷ Find largest void
        λ_S ← AddDensity(ρ_P(p_min)^(-1), h)
        S ← S ∪ p_min                            ▷ Add sample
        r[p_min] ← i
    end for
end procedure
```

2 Indexing a Lower Tridiagonal Matrix

We derive how to index the non-zero elements of a lower tridiagonal matrix $A \in \mathbb{R}^{n\times n}$ given a linear index $k \in \{0,\dots,\frac{n(n-1)}{2}\}$. We define i as the i-th column and j as the j-th row of the lower tridiagonal matrix A, i.e. the non-zero elements. Note that going from i and j to the linear index k is easier: $k = \frac{(i-1)i}{2} + j$. We re-order and solve the equation for i:

$$\begin{aligned} &\frac{(i-1)i}{2} + j = k \\ \Rightarrow\ &(i-1)i = 2(k-j) \\ \Rightarrow\ &i^2 - i = 2(k-j) \\ \Rightarrow\ &i^2 - i - 2(k-j) = 0 \\ \Rightarrow\ &i = \frac{1}{2} \pm \sqrt{\frac{1}{4} - 2(k-j)}. \end{aligned}$$

Only the positive solution is of interest to us. Now, we can compute i by setting $j = 0$ and using the floor function since i must be a natural number:

$$i = \left\lfloor \frac{1}{2} + \sqrt{\frac{1}{4} - 2k} \right\rfloor. \tag{1}$$

Finally, we insert i in the equation we started from to compute j:

$$j = k - \frac{(i-1)i}{2}. \tag{2}$$

Due to floating point inaccuracies for large k, we evaluate the square root in double-precision.