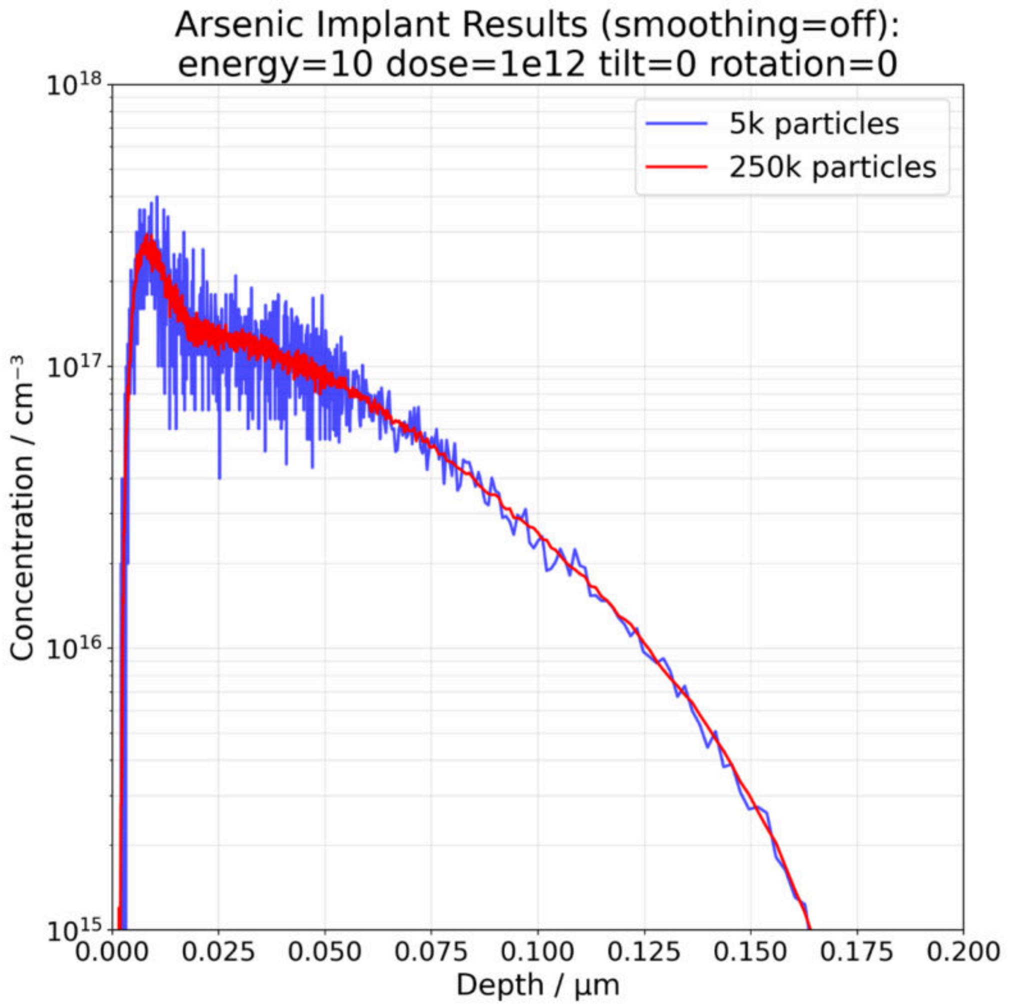
SILVACO

Denoising the Results of Monte-Carlo Ion Implantation

Daniel Chen
TCAD Intern

Wednesday 27 August 2025

Introduction: Noisy Monte Carlo Results

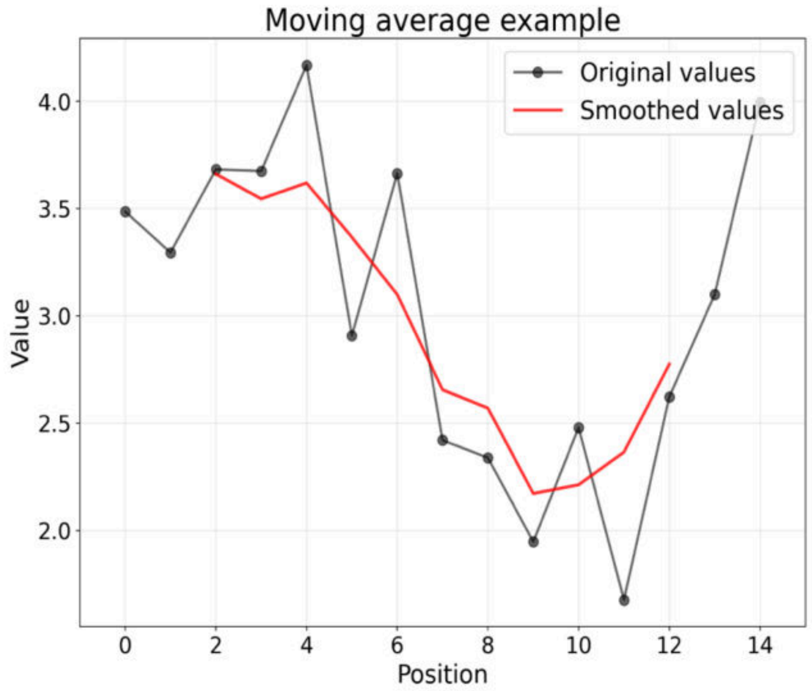


- 250k result takes a long time to simulate – instead, apply smoothing to the 5k results
- Free lunch "paradox": we improve the "sample average" (blue line) solely by post-processing it
- Answer: think like a Bayesian not a Frequentist – implantation profiles tend to be smooth (our prior)
- Bias-variance tradeoff (Stein's paradox, shrinkage)
- Plugin mean-squared error estimate:

$$\frac{1}{N}\sum_i (f_{5k}[i] - f_{250k}[i])^2$$

Convolutions as Smoothers

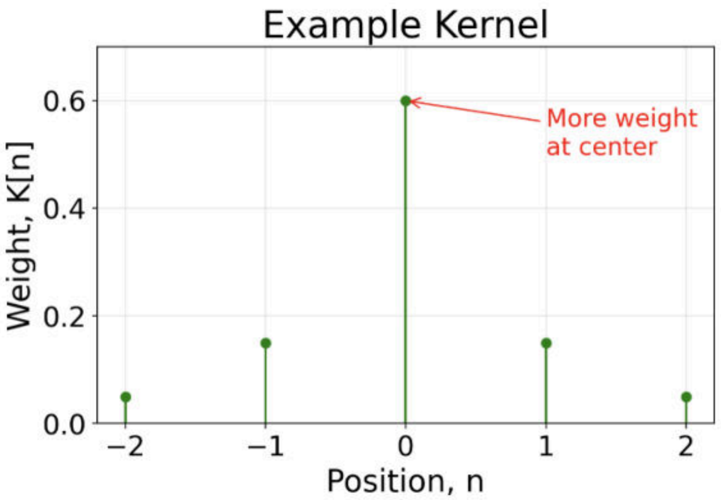
- Moving average: replace each value with the average of the values around it



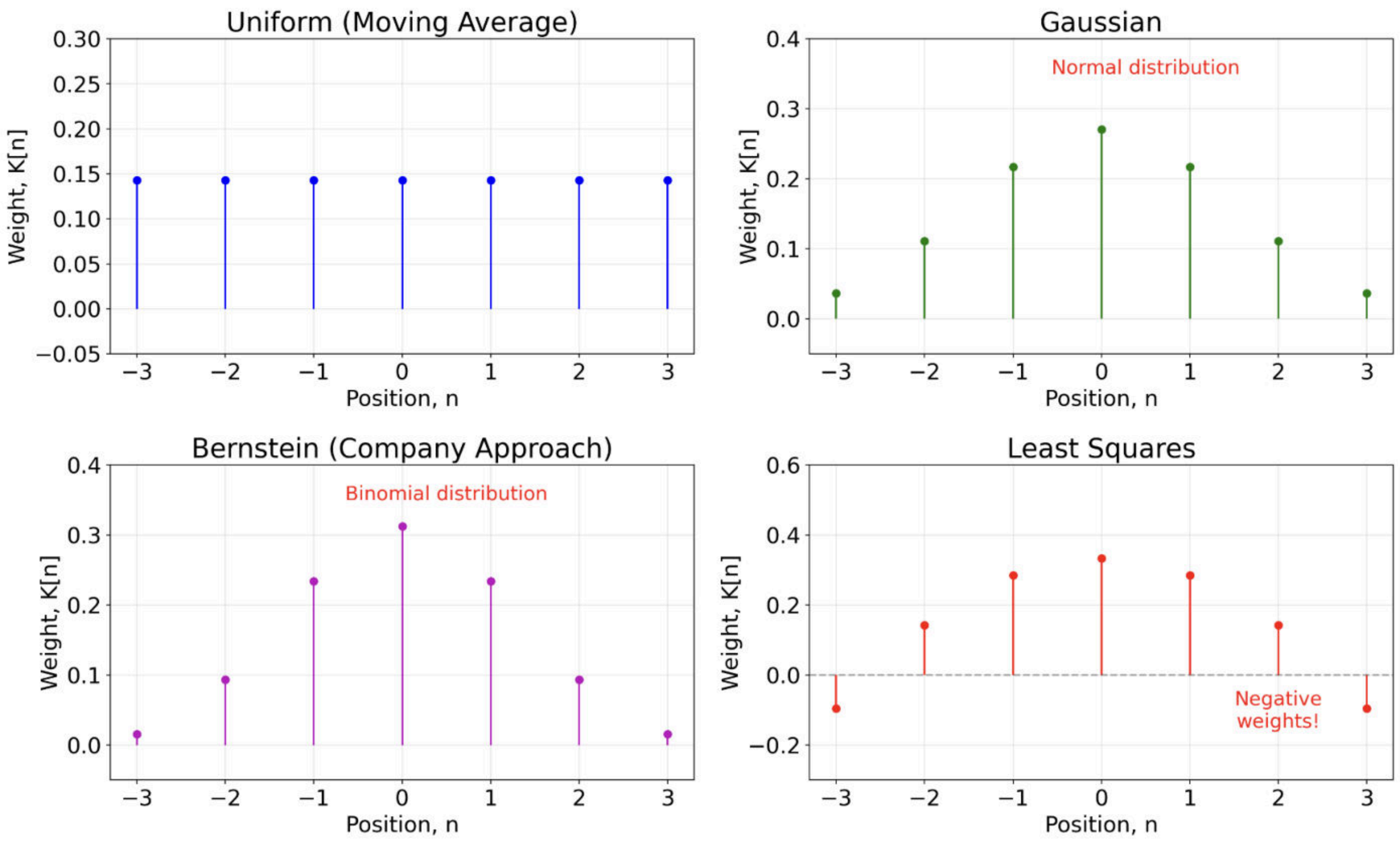
$$\hat{f}_i = \frac{f_{i-2} + f_{i-1} + f_i + f_{i+1} + f_{i+2}}{5}$$

- More generally: collection of weights K[n]

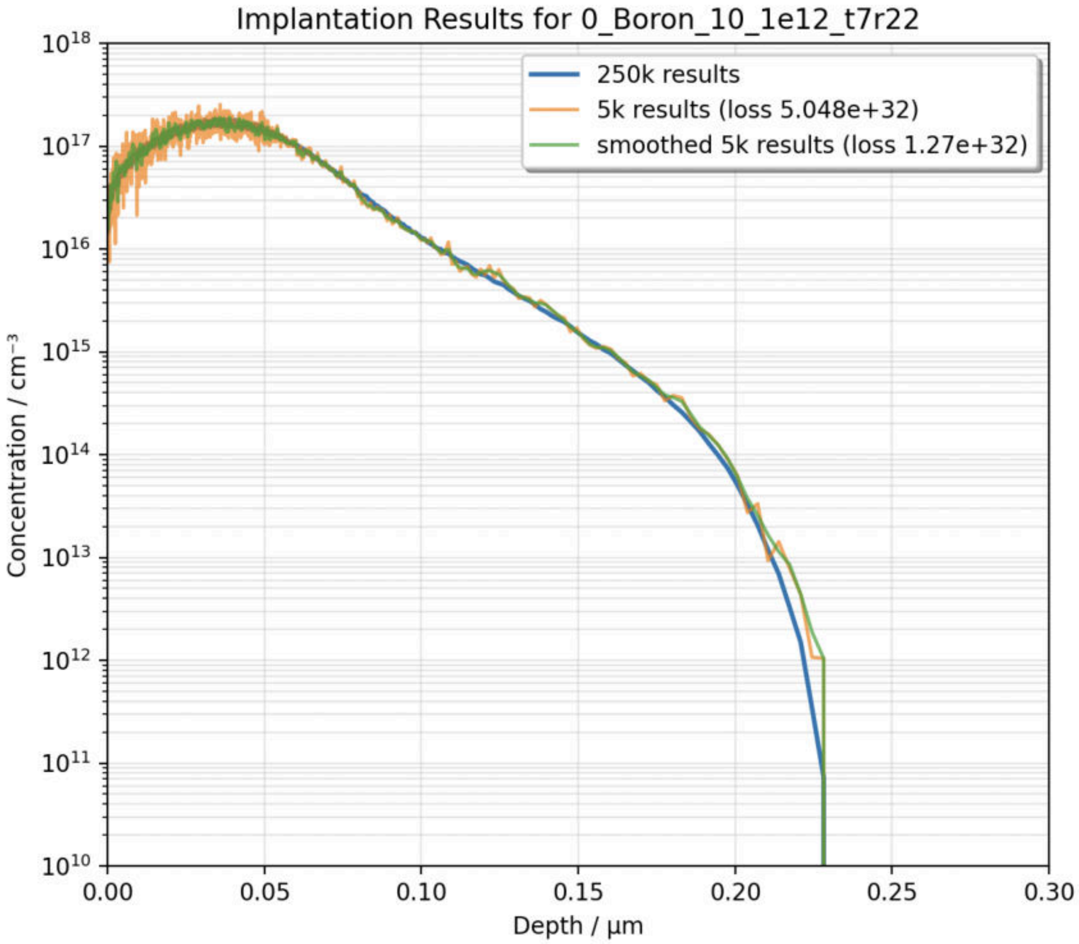
$$\hat{f}_i = \sum_{n=-w}^{w} K[n] \cdot f[i-n]$$

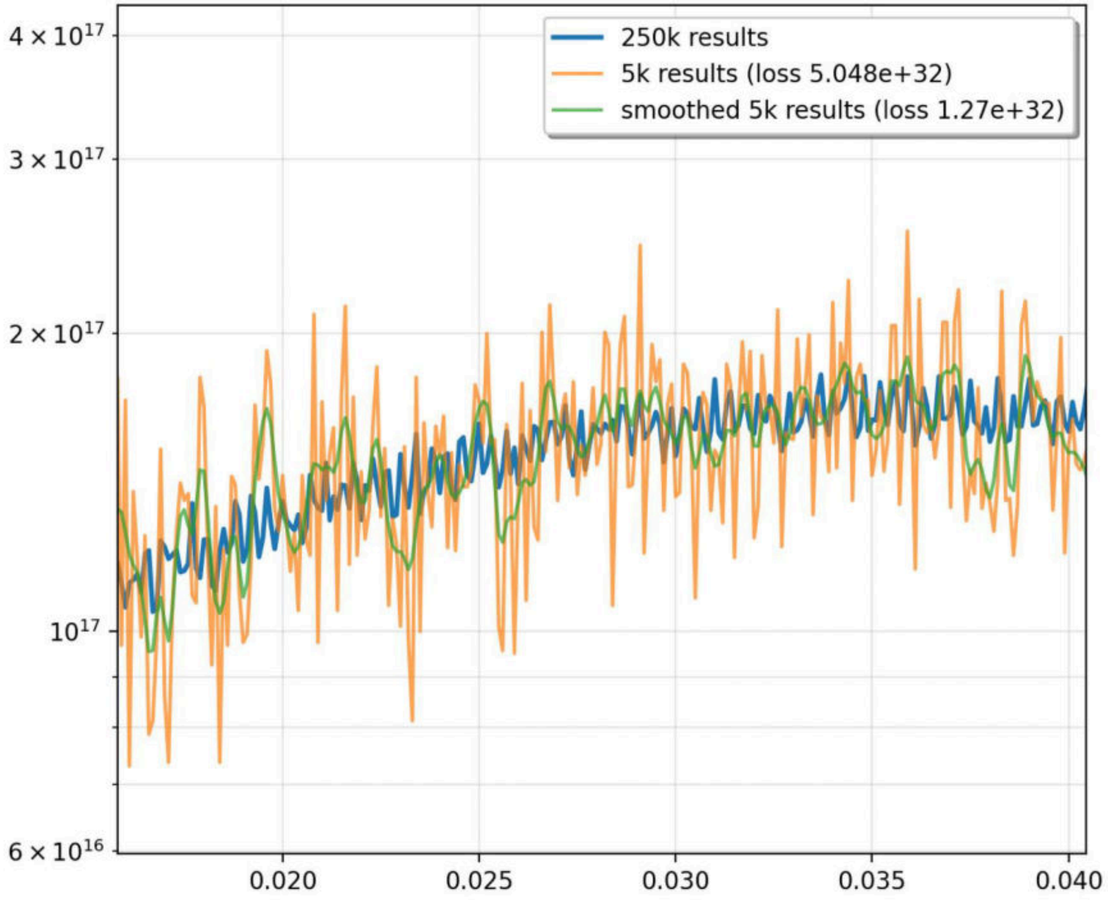


Some Different Kernels (window size = 7)



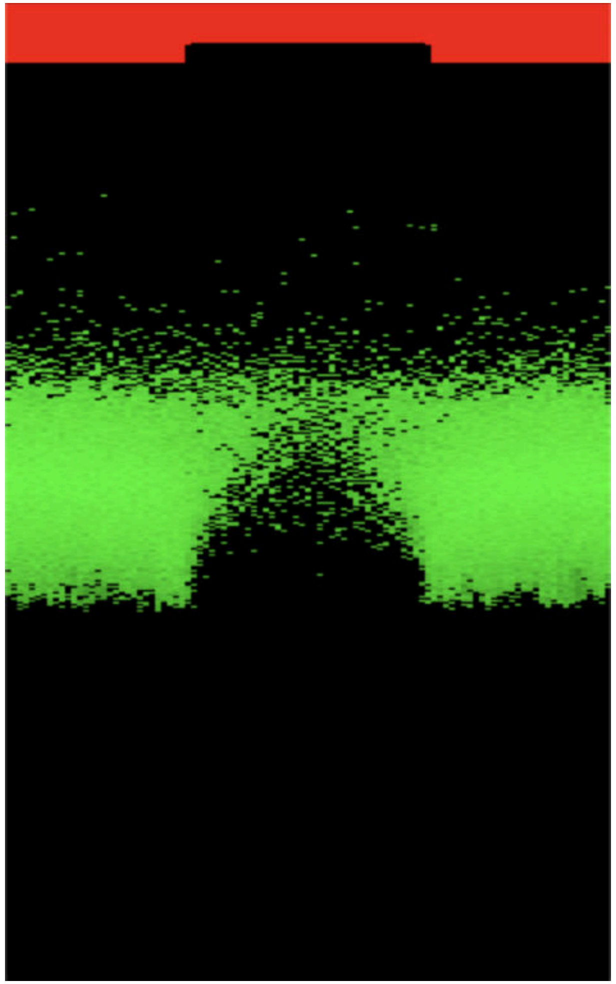
Example 1D Implant – Boron, energy = 10keV



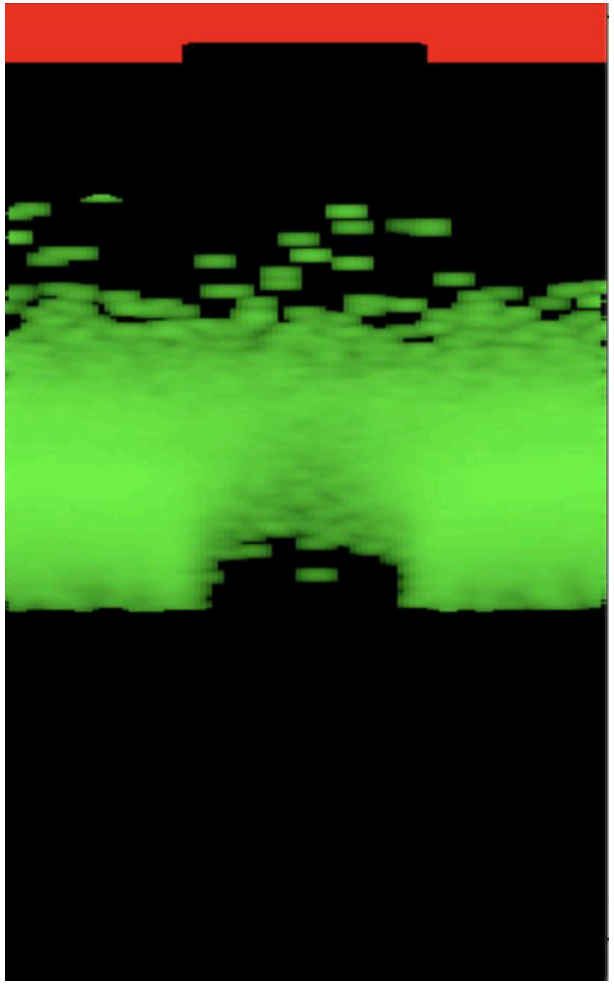


Example 2D Implant – Boron, energy = 1MeV

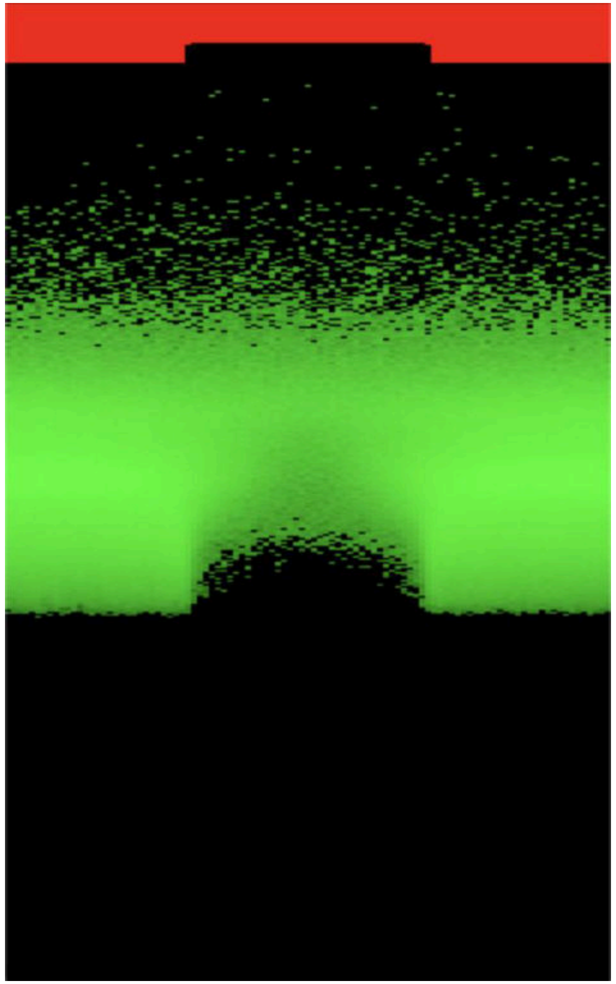
15k raw



15k with Bernstein smoothing



500k reference



Continuous Convolution and the Heat Equation

- Recall: discrete convolution $\sum_n K[n]\, f[i-n]$
- Continuous analogy: define the convolution of K and f to be

$$(K * f)(x) := \int_{-\infty}^{\infty} K(y) f(x-y)\, dy$$

- When $K(x)$ is a Gaussian, $K * f$ will be a "smoothed out" version of f
- **Key insight: "to smooth a graph, run the heat equation"**
- The solution to the heat equation

$$\begin{cases} u_t = D u_{xx} \\ u(x,0) = f(x) \end{cases}$$

is $u(x,t) = (K_t * f)(x)$ where $K_t(x)$ is a Gaussian with mean 0 and variance $2Dt$.

- Mathematically equivalent to smoothing f using a wider and wider kernel!

Modifying the Heat Equation

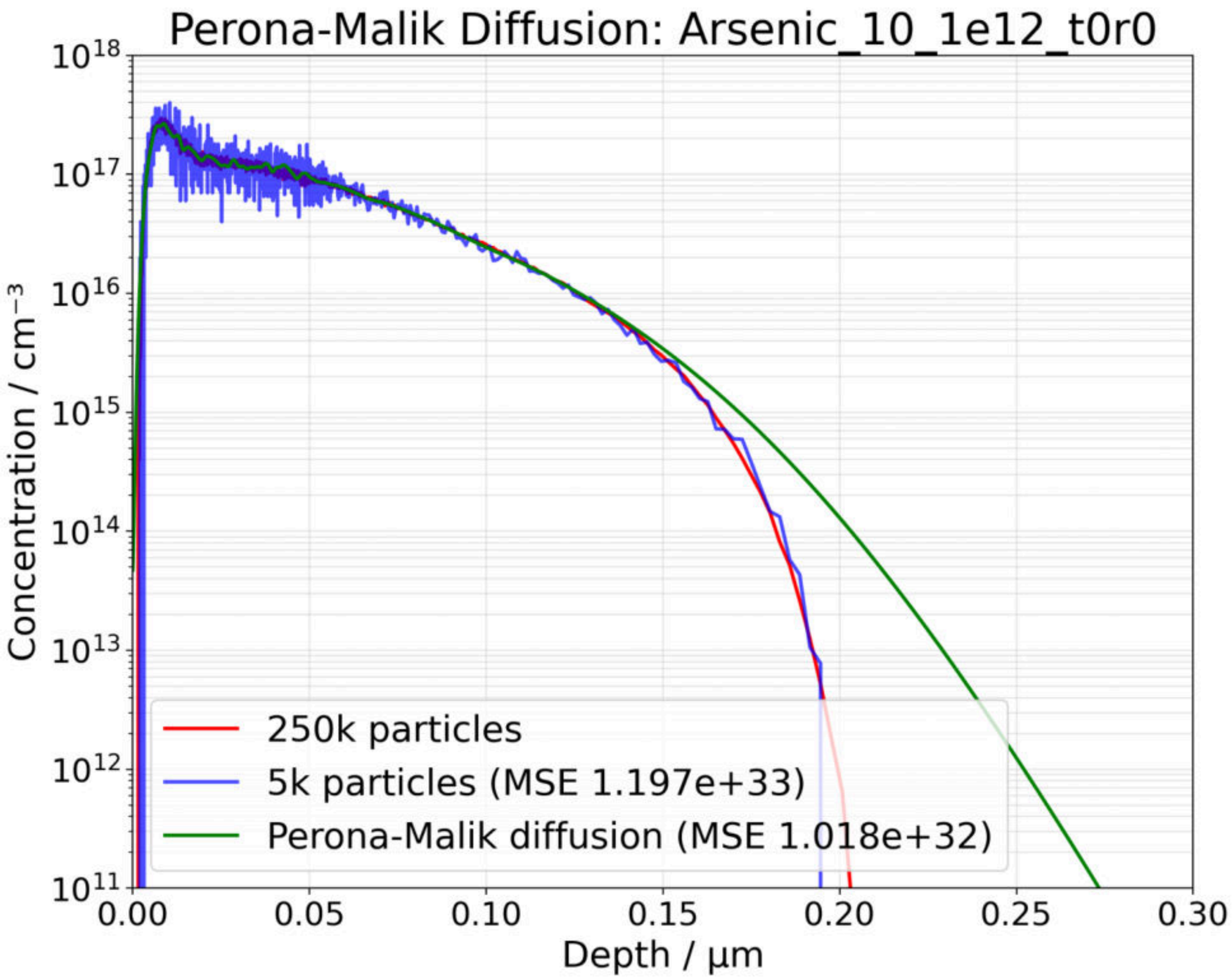
- Standard heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(D u_x)$$

- Bad at preserving max depth
- Perona-Malik: D is variable

$$D\left(u_x^2\right) = \frac{1}{1 + u_x^2}$$

- Better than Gaussian Blur but still bad



Screening Function Modification

- Perona-Malik $$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}(D(u_x^2)u_x)$$

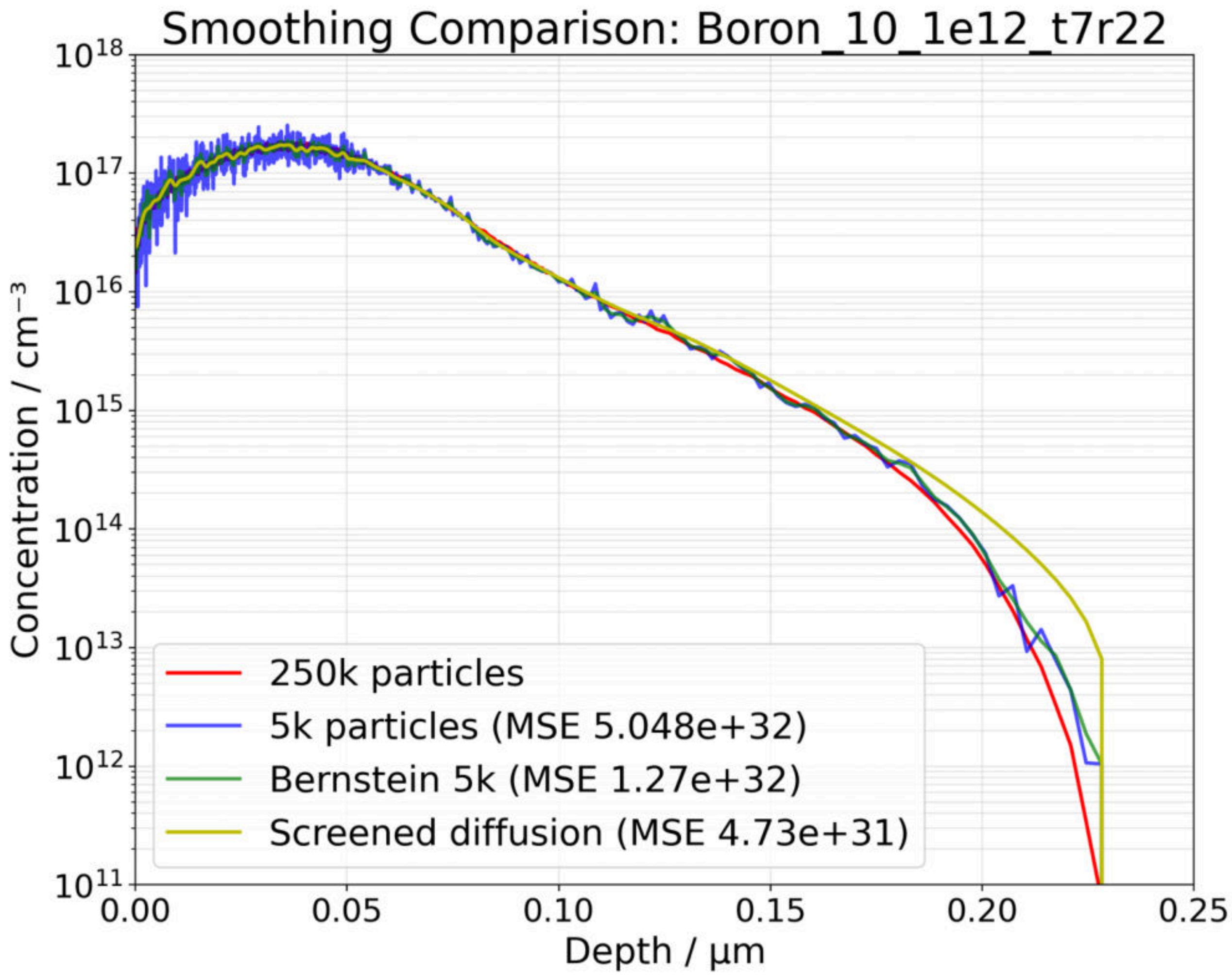
- Now multiply by a screening function

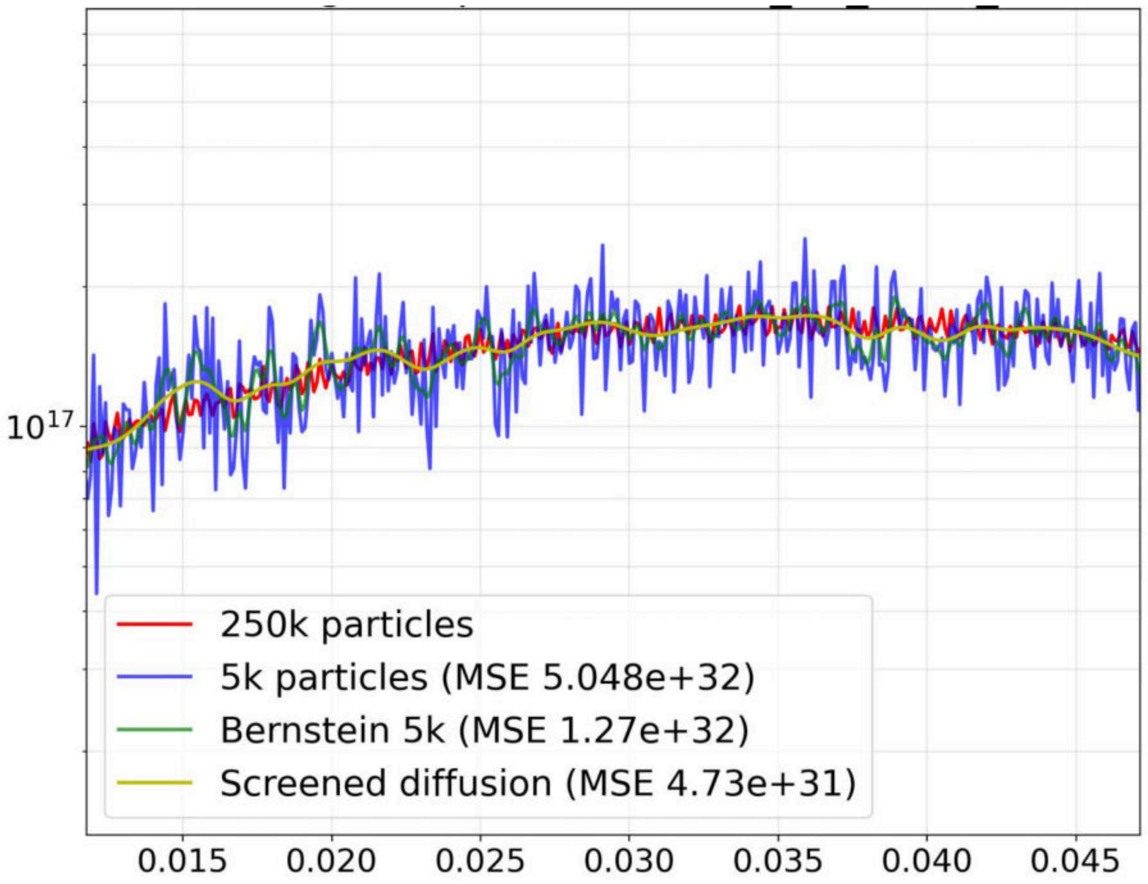
$$\frac{\partial u}{\partial t} = W(u)\frac{\partial}{\partial x}(D(u_x^2)x_x)$$

$$W(u) = \frac{\log(1+u)}{\log(1+P)}$$

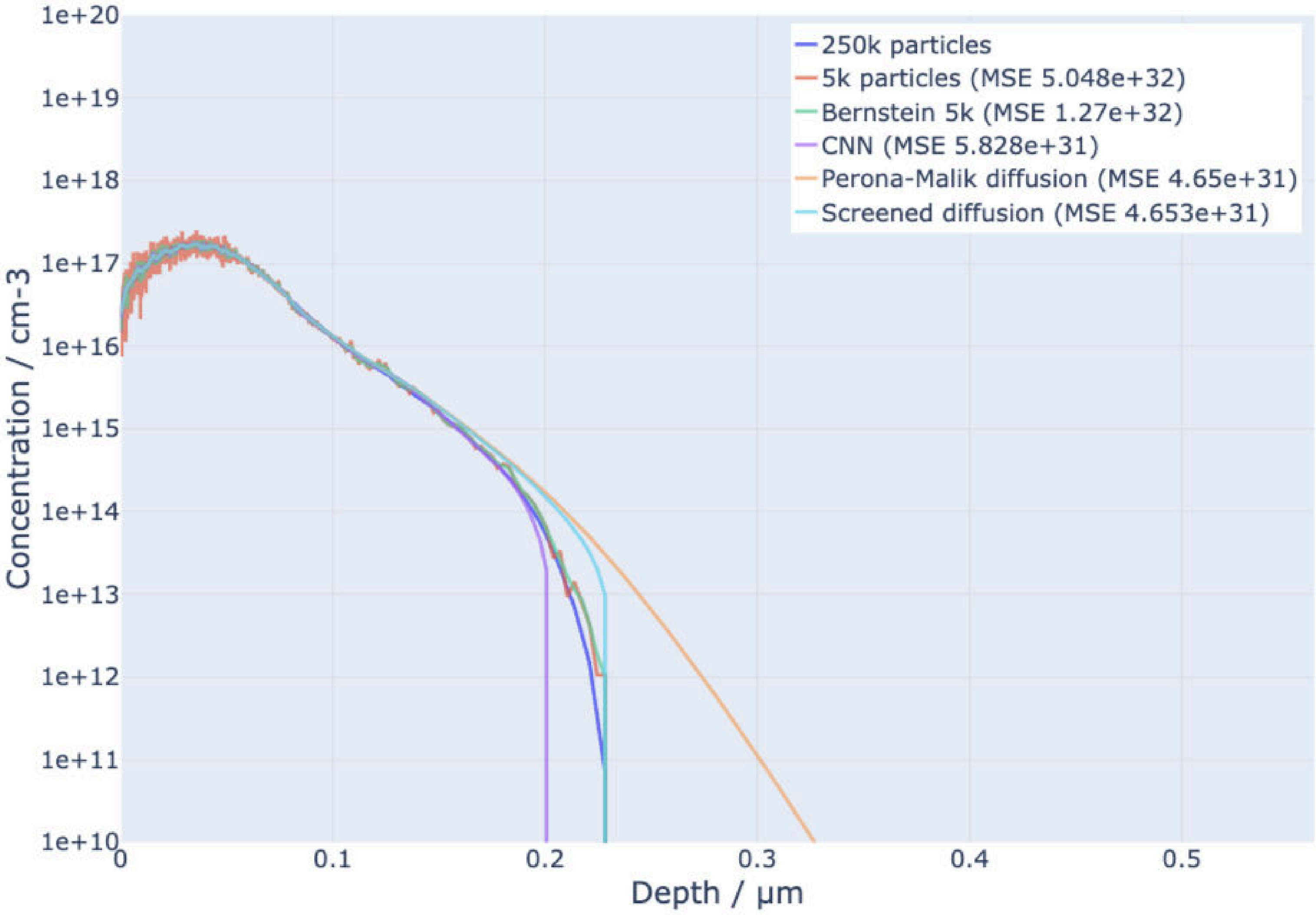
- Derived from a tweak in the staggered-grid numerical solver to encourage area loss (omitted)

1D Screened Diffusion Example

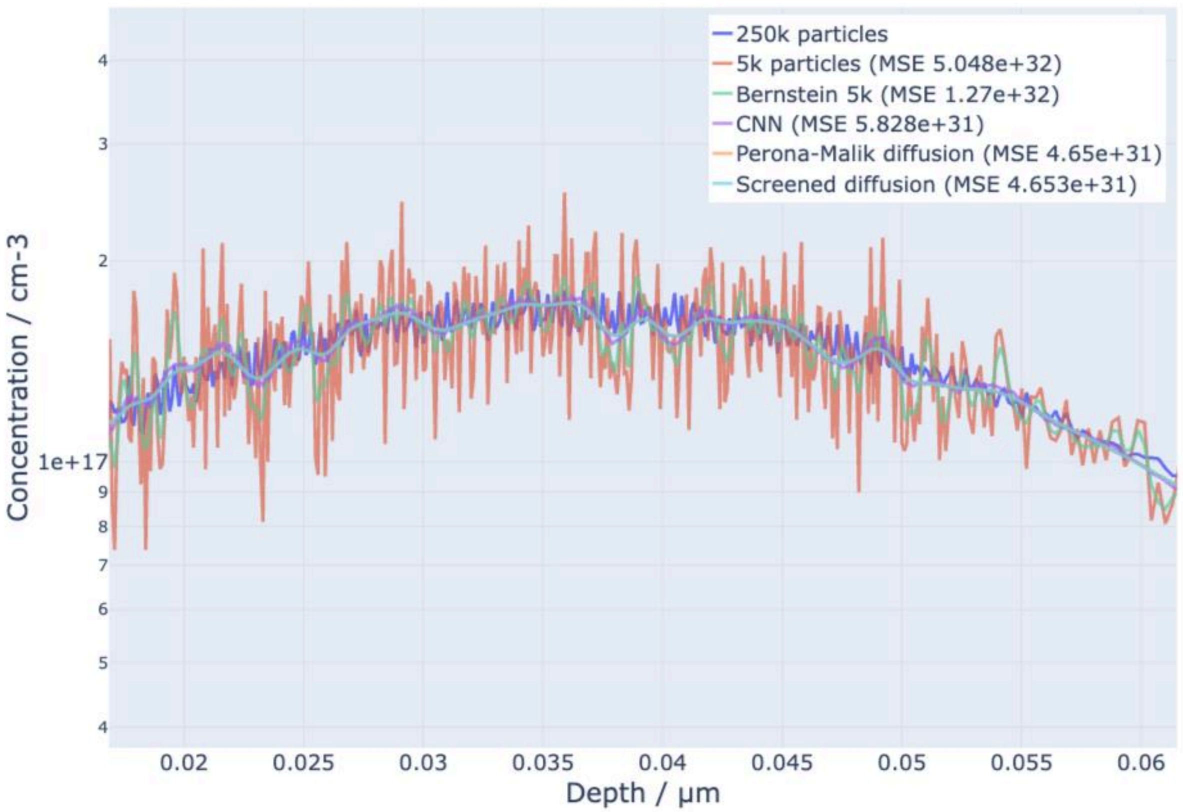


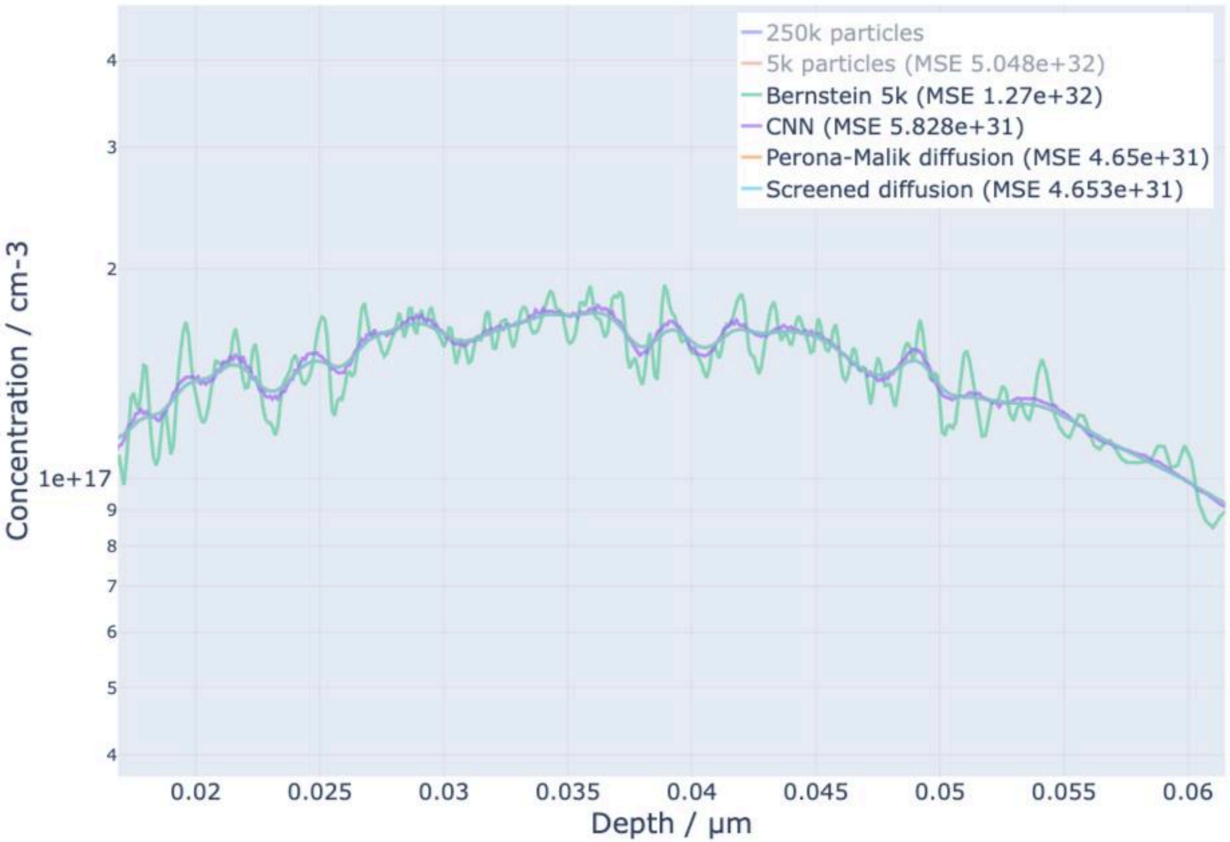


1D Comparison of All Methods – Boron 10keV



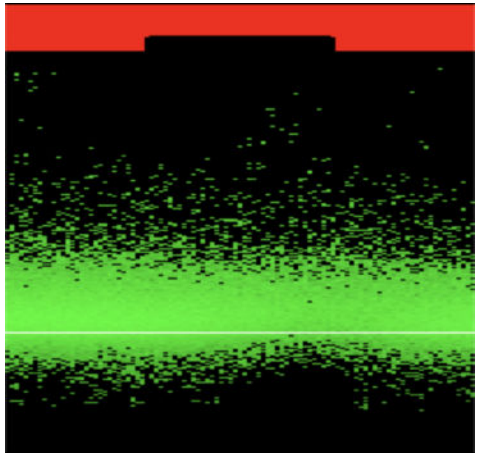
1D Comparison of All Methods – Boron 10keV



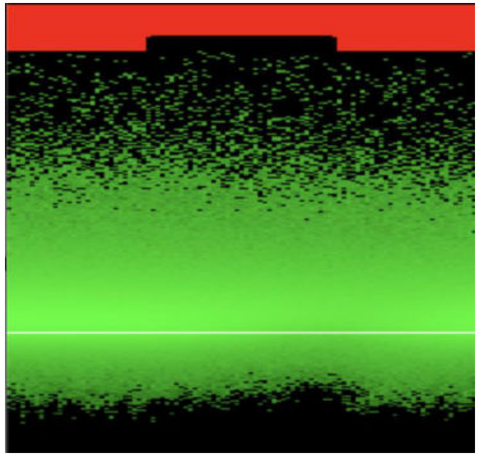


2D Example – Boron 1MeV

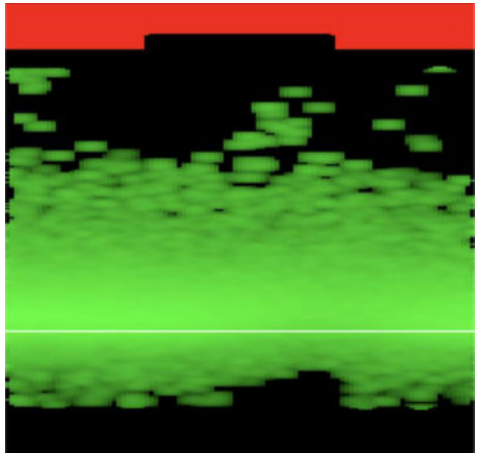
15k raw



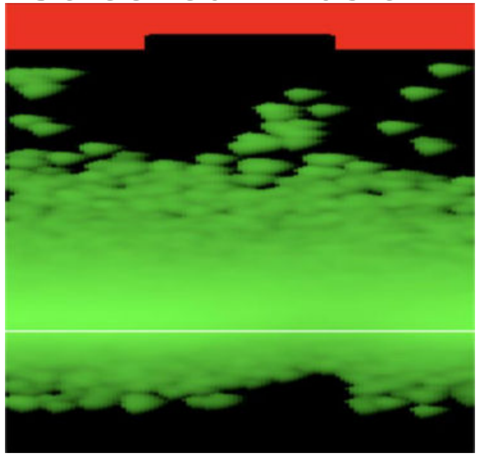
500k reference

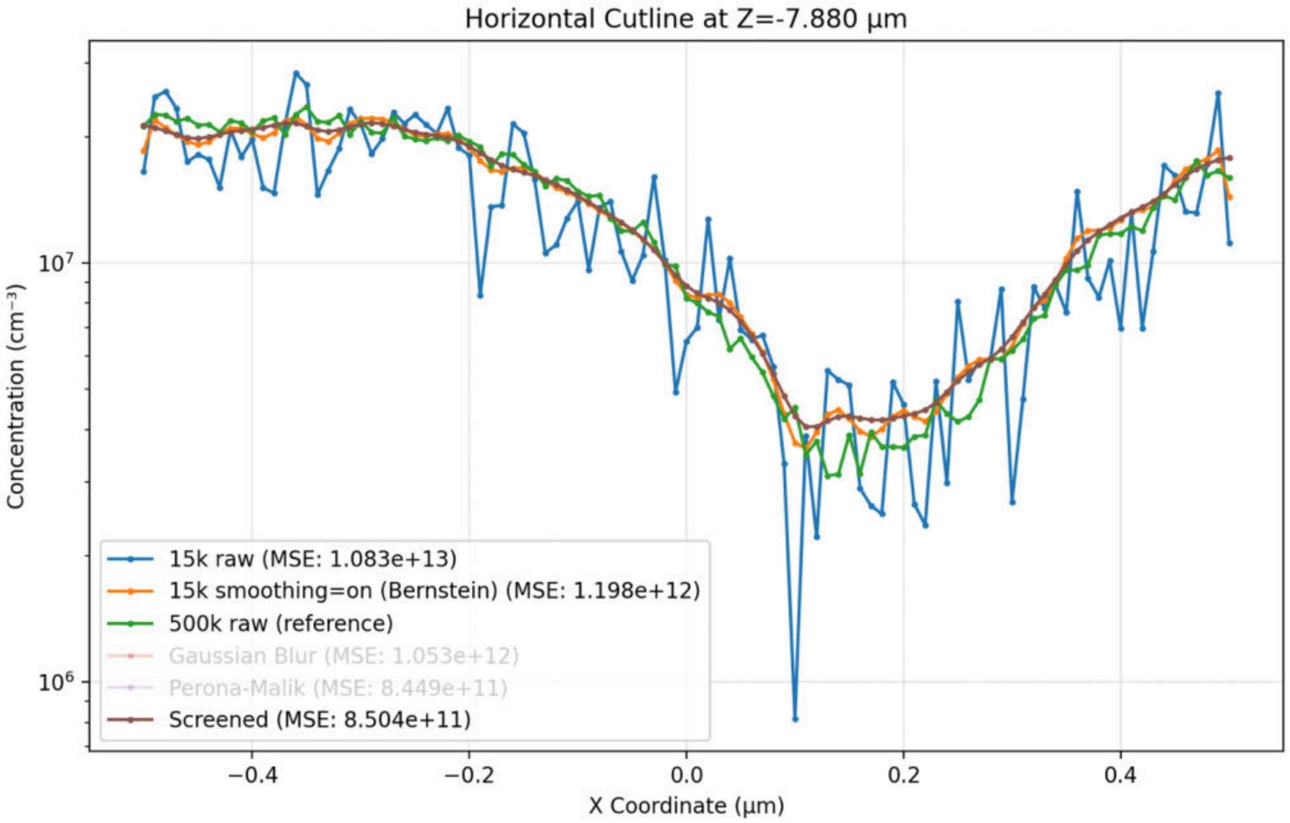


Bernstein



Screened Diffusion





Conclusion

Average MSE loss for smoothing methods applied to the 5k validation data

Method	MSE / 1e32
Raw 5k	13.29
Bernstein (current)	3.63
CNN	1.91
Screened Diffusion	1.66
Perona-Malik Diffusion	1.63

- N.B. In the graphs shown, current Bernstein algorithm is the best at preserving max depth, but it is cheating (chops off data past max depth) – this is not applicable to higher dimensions than 1D so I didn't implement it
- Screened diffusion is sacrificing a bit of MSE to preserve max depth. Sometimes it doesn't sufficiently smooth the peak
- Omitted from this talk – CNN, gradient descent of jaggedness metric, 3D data

Thank you for listening