Volume Estimation of High-Dimensional Convex Bodies

MCMC Approach

Jonathan, Silvan, Manuel, Emanuel

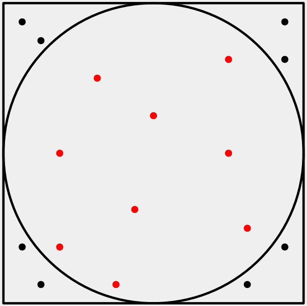
23. 05. 2020

Problem: Calculating the volume of a convex body is NP-hard.

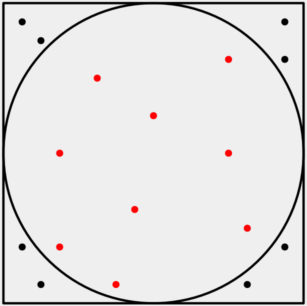
Problem: Calculating the volume of a convex body is NP-hard.

Solution: Use randomized approximation algorithm (MCMC).

Idea: Sample points randomly to estimate volume

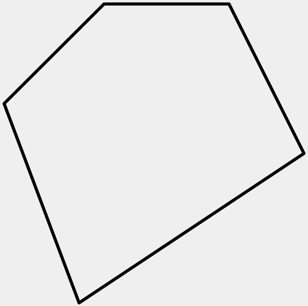


Idea: Sample points randomly to estimate volume

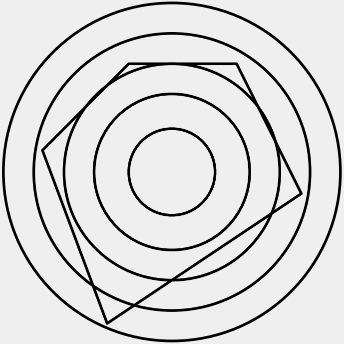


New problem: $\frac{Vol(\text{Unit ball})}{Vol(\text{Unit cube})} = O(2^{-d})$, where d is the dimension

Consider the following convex body

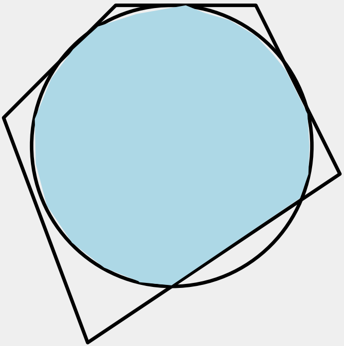


Solution: Subdivide the body into zones to keep ratio constant



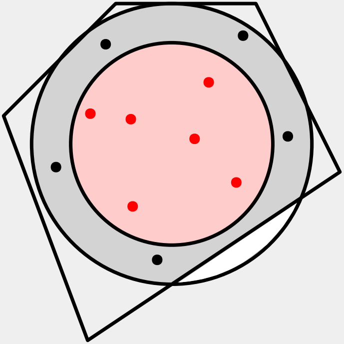
Note: The smallest ball is fully contained and the biggest ball contains the body.

Solution: Subdivide the body into zones to keep ratio constant

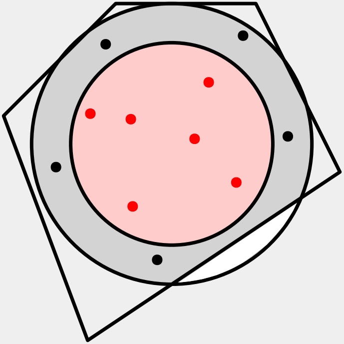


Each zone is the intersection of a ball with the body

Then sample in one zone and count how many points fall into the smaller zone



Then sample in one zone and count how many points fall into the smaller zone



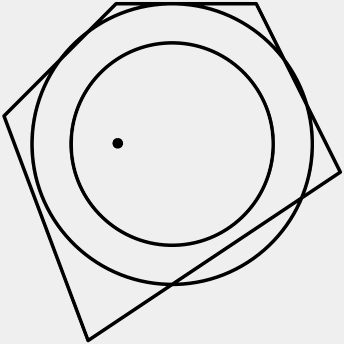
This gives us the ratio $\frac{\#\text{Total samples}}{\#\text{Samples in } \text{Zone}_\text{i}} \approx \frac{Vol(\text{Zone}_{\text{i}+1})}{Vol(\text{Zone}_\text{i})}$

Doing this for all zones we can calculate the volume of the body:

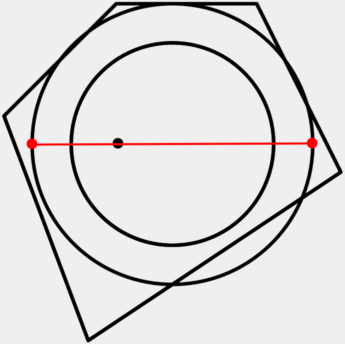
$$Vol(\text{Body}) = \frac{Vol(\text{Zone}_n)}{Vol(\text{Zone}_{n-1})} \cdot ... \cdot \frac{Vol(\text{Zone}_2)}{Vol(\text{Zone}_1)} \cdot Vol(\text{Zone}_1)$$

where $Vol(\text{Zone}_1)$ is just the volume of the innermost ball.

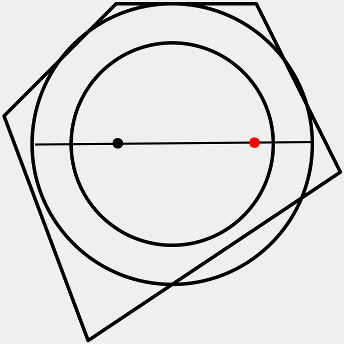
How to sample uniformly at random in a zone:



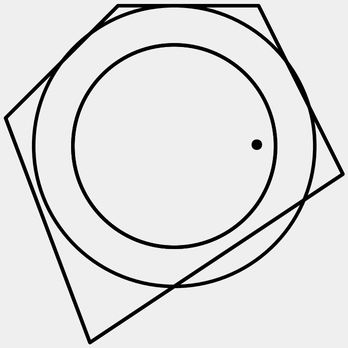
How to sample uniformly at random in a zone:



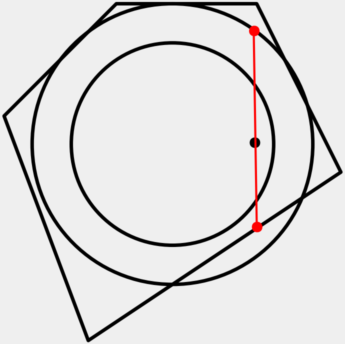
How to sample uniformly at random in a zone:



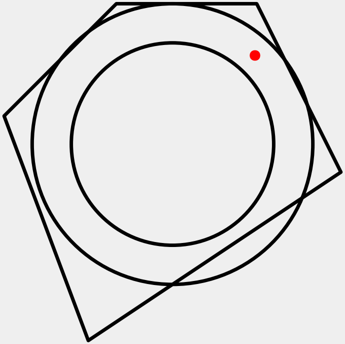
How to sample uniformly at random in a zone:

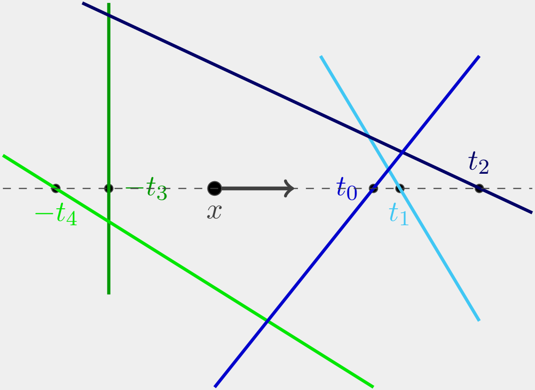


How to sample uniformly at random in a zone:

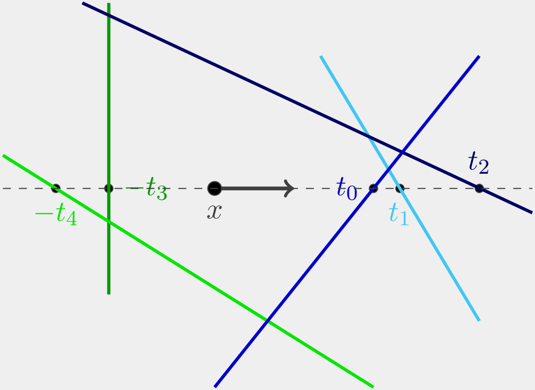


How to sample uniformly at random in a zone:





$$\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \\ a_{2,0} & a_{2,1} \\ a_{3,0} & a_{3,1} \\ a_{4,0} & a_{4,1} \end{pmatrix} \cdot \begin{pmatrix} x_1 + t \\ x_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

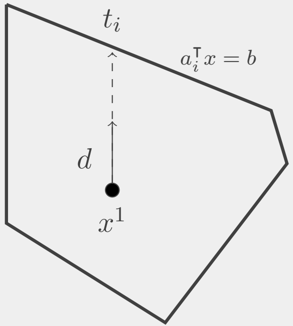


$$\begin{pmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \\ a_{2,0} & a_{2,1} \\ a_{3,0} & a_{3,1} \\ a_{4,0} & a_{4,1} \end{pmatrix} \cdot \begin{pmatrix} x_1 + t \\ x_2 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

solve for t_i for each constraint a_i

$$\min\{t_0, t_1, t_2\}$$

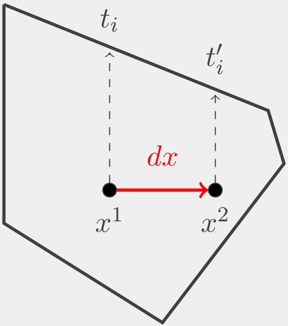
$$\max\{-t_3, -t_4\}$$



$$x^1 = \begin{pmatrix} x_1^1 \\ \vdots \\ x_k^1 \\ \vdots \\ x_n^1 \end{pmatrix}$$

$$t_i = \frac{b_i - a_i^\intercal x^1}{a_i^\intercal d} = \frac{b_i - a_i^\intercal x^1}{a_{i,j}}$$

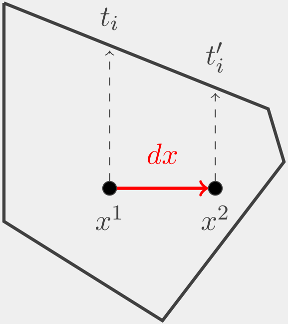
Compute intersections



$$x^1 = \begin{pmatrix} x_1^1 \\ \vdots \\ \boxed{x_k^1} \\ \vdots \\ x_n^1 \end{pmatrix}, \qquad x^2 = \begin{pmatrix} x_1^1 \\ \vdots \\ \boxed{x_k^1 + \|dx\|} \\ \vdots \\ x_n^1 \end{pmatrix}$$

$$t_i = \frac{b_i - a_i^\intercal x^1}{a_i^\intercal d} = \frac{b_i - a_i^\intercal x^1}{a_{i,j}}$$

$$t_i' = \frac{b_i - a_i^\intercal x^1 - a_{i,k} \cdot \|dx\|}{a_{i,j}}$$



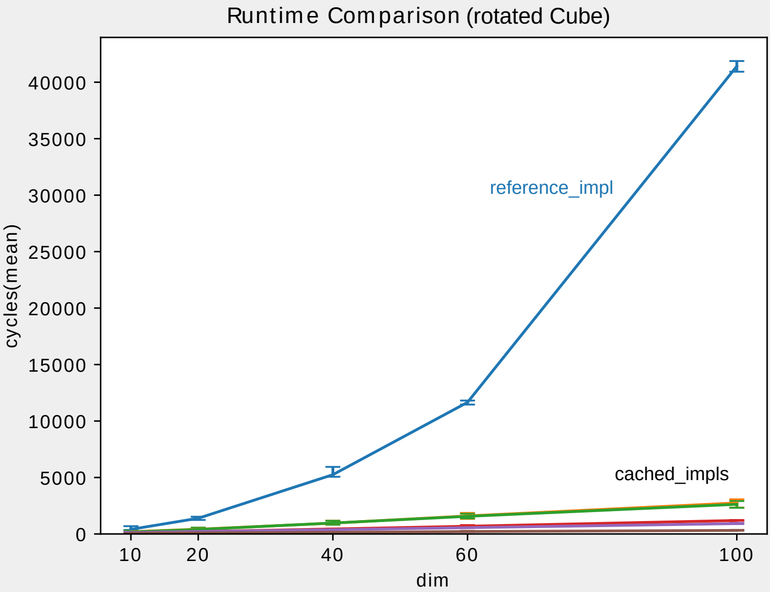
$$x^1 = \begin{pmatrix} x_1^1 \\ \vdots \\ \boxed{x_k^1} \\ \vdots \\ x_n^1 \end{pmatrix}, \qquad x^2 = \begin{pmatrix} x_1^1 \\ \vdots \\ \boxed{x_k^1 + \|dx\|} \\ \vdots \\ x_n^1 \end{pmatrix}$$

$$t_i = \frac{b_i - a_i^\intercal x^1}{a_i^\intercal d} = \frac{b_i - a_i^\intercal x^1}{a_{i,j}}$$

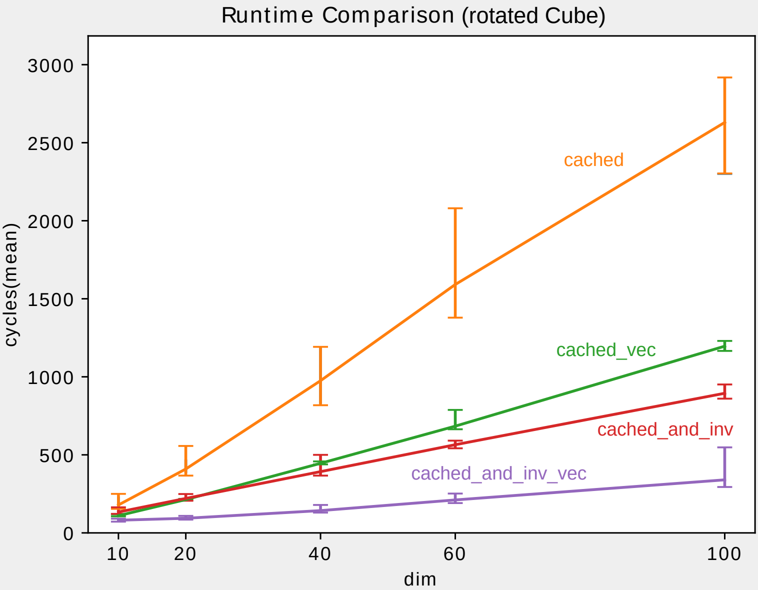
$$t'_i = \frac{b_i - a_i^\intercal x^1 - a_{i,k} \cdot \|dx\|}{a_{i,j}}$$

- Platform: Intel Haswell i7-4870HQ
- Compiler: gcc version 9.3.0
- Compilation flags: -march=native -mfma -ffast-math -O3
- L1d cache: 128kB
- L1i cache: 128kB

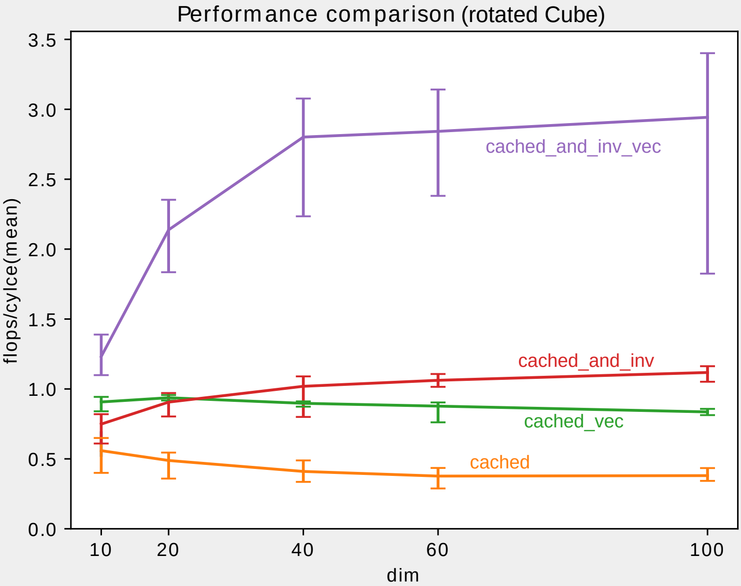
Benchmark: Intersect



Benchmark: Intersect



Benchmark: Intersect



Sparse Scenario

- So far, we considered dense constraint matrices

Sparse Scenario

- So far, we considered dense constraint matrices
- Now assume each constraint contains only a few variables (still NP-hard)

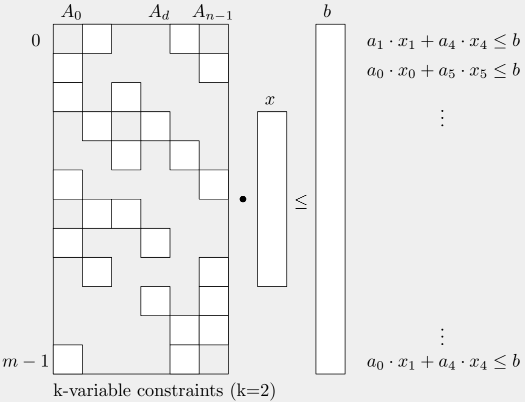
Sparse Scenario

- So far, we considered dense constraint matrices
- Now assume each constraint contains only a few variables (still NP-hard)

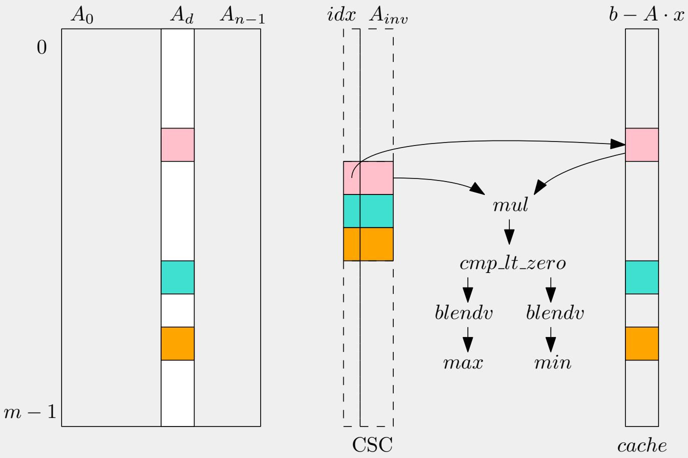
$$a_1 \cdot x_1 + a_4 \cdot x_4 \leq b$$
$$a_0 \cdot x_0 + a_5 \cdot x_5 \leq b$$
$$\vdots$$

$$\vdots$$
$$a_0 \cdot x_1 + a_4 \cdot x_4 \leq b$$

- So far, we considered dense constraint matrices
- Now assume each constraint contains only a few variables (still NP-hard)

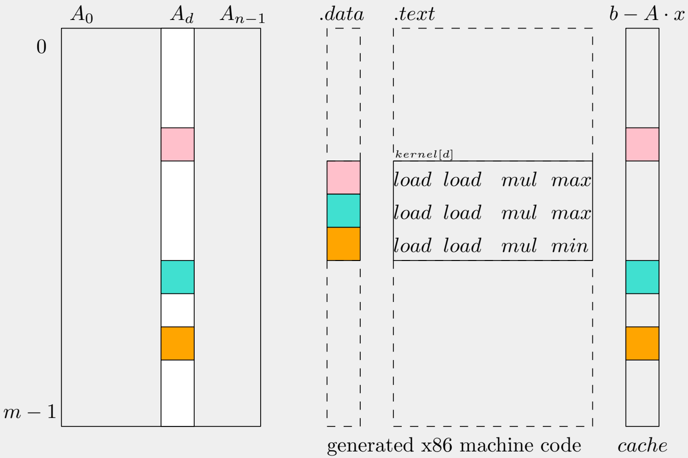


- Sparse matrix format, column major.

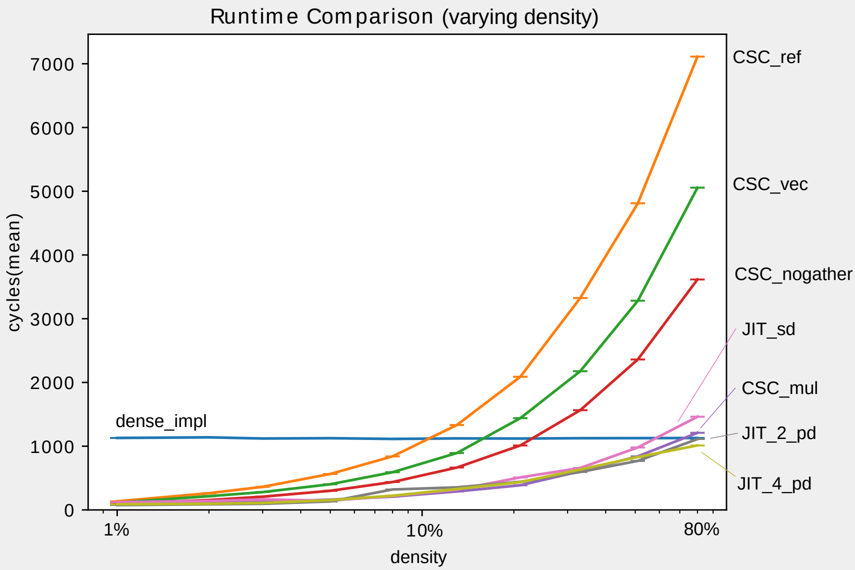


Sparsity JIT

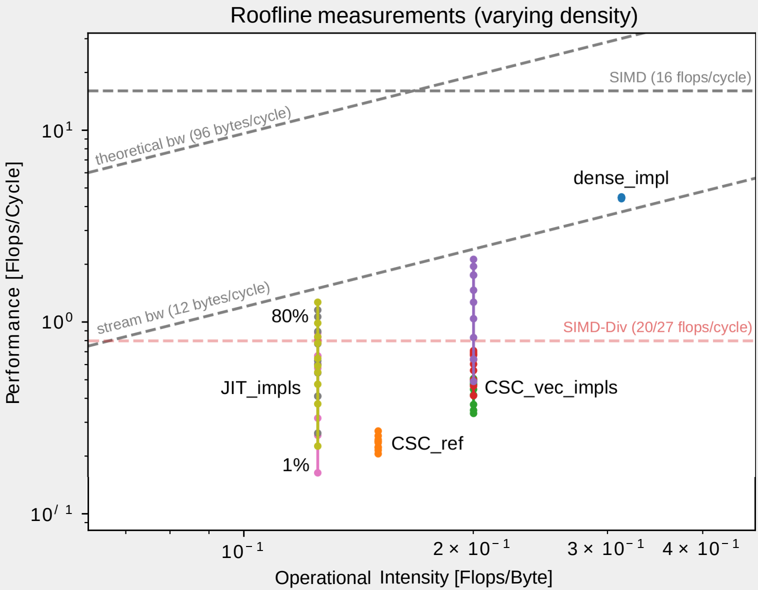
- Run same matrix many times: kernel?
- Generate code at run time (just-in-time). No blendv, no cmp.



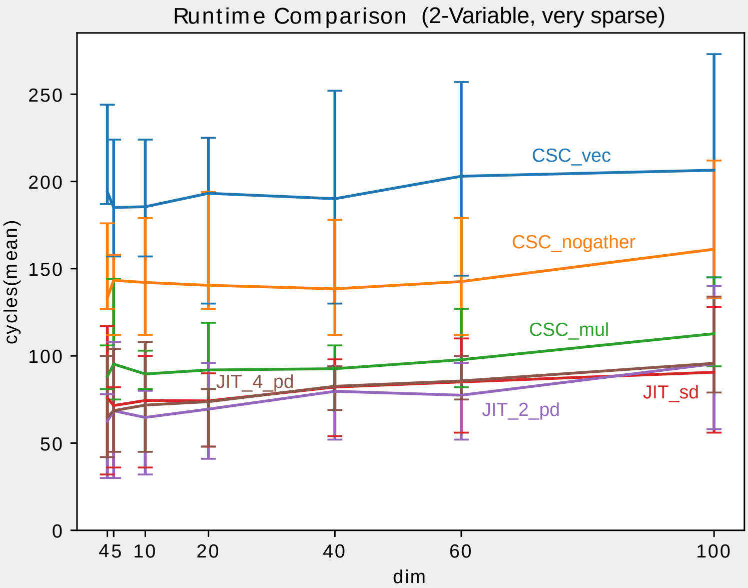
Benchmark: Runtime Density



Benchmark: Roofline Density



Benchmark: Runtime Latency



Thank you very much!

Benchmark: Dense Update