Low-Rank Approximation of Kernel Matrices

Leopold Cambier

ICME, Stanford University

February 21, 2017

Integral equations

$$\int_X K(x,y)u(x)\mathrm{d}x = f(y)$$

for given $K : X \times Y \to \mathbb{R}$, $f : Y \to \mathbb{R}$ and unknown $u : X \to \mathbb{R}$.
Discretization gives

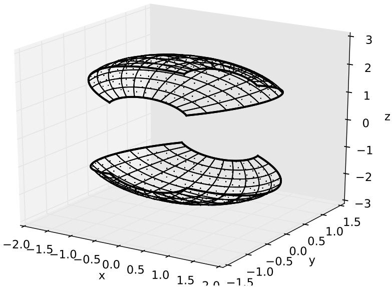
$$\sum_i K(x_i, y_j)u(x_i) = f(y_j)$$

Typical situation of BEM:

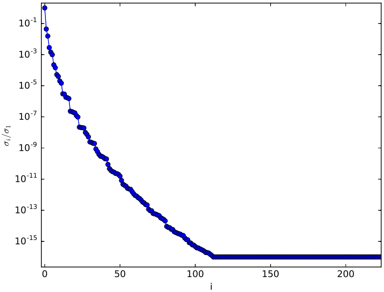
- X, Y are 2 parts of a 3D mesh
- May or may not be well-separated
- For instance,

$$K(x, y) = \frac{1}{\|x - y\|_2}$$

When X and Y are well-separated, $K(x,y) = \frac{1}{\|x-y\|}$ is smooth and hence low-rank.



1581 × 1608 points



The spectrum of K_{ij}

The problem

Given $X = \{x_1, \ldots, x_M\}$, $Y = \{y_1, \ldots, y_N\}$ and $K : X \times Y \to \mathbb{R}$, build a low-rank representation of $K_{ij} = K(x_i, y_j)$, i.e.

$$K \approx USV^\top$$

for $U \in \mathbb{R}^{M \times r}$,$V \in \mathbb{R}^{N \times r}$, $S \in \mathbb{R}^{r \times r}$.

Outline

Low-Rank Kernel Approximation

Naive Solution

Naive idea

- Compute $K_{ij} = K(x_i, y_j)$ - $\mathcal{O}(MN)$
- Rank-revealing QR - up to $\mathcal{O}(MNr)$

Bottleneck is $\mathcal{O}(MN)$ complexity.

Main Idea

Main idea

- Assume we are given

$$K(x,y) \approx \sum_{p=1}^{r} u_p(x) v_p(y)$$

- Then we immediately have

$$K_{ij} \approx \sum_{p=1}^{r} u_p(x_i) v_p(y_j) = \sum_{p=1}^{r} u_{ip} v_{jp}$$

- How to compute $u_p(x)$ and $v_p(y)$?

The Answer

Interpolation !

Quick Review

Given a function $f : X \subset \mathbb{R}^d \to \mathbb{R}$ we can write

$$f(x) \approx \bar{f}(x) = \sum_{k=1}^{K} f(\bar{x}_k) T_k(x)$$

where x_k are interpolation nodes and T_k Lagrange basis functions, i.e.

$$T_k(\bar{x}_k) = 1,\ T_k(\bar{x}_i) = 0 \text{ for } i \neq k.$$

This directly implies (interpolation)

$$f(\bar{x}_k) = \bar{f}(\bar{x}_k).$$

Low-Rank Kernel Approximation

Given this, write

$$\begin{aligned}
K(x,y) &\approx \sum_{k=1}^{K} R_k(x) K(\bar{x}_k, y) \\
&\approx \sum_{k=1}^{K} R_k(x) \sum_{l=1}^{L} T_l(y) K(\bar{x}_k, \bar{y}_l) \\
&= \sum_{k=1}^{K} \sum_{l=1}^{L} R_k(x) K(\bar{x}_k, \bar{y}_l) T_l(y) \\
&= \bar{K}(x,y)
\end{aligned}$$

We have an interpolation scheme on $X \times Y$ since

$$\bar{K}(\bar{x}_k, \bar{y}_l) = K(\bar{x}_k, \bar{y}_l).$$

Low-Rank Kernel Approximation

$$\bar{K}(x,y) = \sum_{k=1}^{K}\sum_{l=1}^{L} R_k(x) K(\bar{x}_k, \bar{y}_l) T_l(y)$$

$$= \begin{bmatrix} | & & | \\ R_1(x) & \dots & R_K(x) \\ | & & | \end{bmatrix} \begin{bmatrix} K(\bar{x}_1, \bar{y}_1) & \dots & K(\bar{x}_1, \bar{y}_L) \\ \vdots & & \vdots \\ K(\bar{x}_K, \bar{y}_1) & \dots & K(\bar{x}_K, \bar{y}_L) \end{bmatrix} \begin{bmatrix} — T_1(y)^\top — \\ \vdots \\ — T_L(y)^\top — \end{bmatrix}$$

$$\text{rank } r_0 = \min(K, L)$$

Given such representation, complexity becomes

$$\mathcal{O}\left(MK + KL + LN\right) \approx \mathcal{O}\left(r_0 n\right)$$

versus

$$\mathcal{O}\left(MKr\right) \approx \mathcal{O}\left(rn^2\right)$$

before.

Recompression

Given

$$\bar{K} = RKT^\top \quad \text{rank } \bar{K} = r_0$$

we further recompress $\bar{K}$ as

$$\begin{aligned} \bar{K} &= (Q_R R_R) K (Q_T R_T)^\top \\ &= Q_R (R_R K R_T^\top) Q_T^\top \\ &= Q_R U_K S_K V_K^\top Q_T^\top \\ &= (Q_R U_K) S_K (Q_T V_K)^\top \\ &= USV^\top \end{aligned}$$

where rank $\bar{K} = r_1$. This requires $\approx \mathcal{O}(n r_0 r_1) + \mathcal{O}(r_0^2 r_1)$ work.

One Thing Left Unanswered

Interpolation

How to obtain the (multivariate) interpolation ?

Multivariate Polynomial Interpolation

Univariate Polynomial Interpolation

- The best way to interpolate a smooth function f on $[a, b]$ using polynomials is to use Chebyshev-like type of nodes that cluster to the boundary
- E.g.

$$\bar{x}_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{2k-1}{2n}\pi\right) \quad k = 1, \ldots, n$$

- Using barycentric formula (for stability), this implies

$$f(x) \approx \bar{f}(x) = \sum_{k=1}^{n} f(\bar{x}_k) \underbrace{\frac{\frac{\bar{w}_k}{x-\bar{x}_k}}{\sum_{j=1}^{n} \frac{\bar{w}_j}{x-\bar{x}_j}}}_{=T_k(x)}$$

with

$$\bar{w}_j = \frac{1}{\prod_{k \neq j}(\bar{x}_j - \bar{x}_k)}$$

Multivariate Polynomial Interpolation

If $X = I_1 \times \cdots \times I_d$ where $I_i = [a_i, b_i]$ then use a sequence of univariate interpolation rules

$$f(x) \approx \bar{f}(x) = \sum_{k_{\mathbf{1}}=1}^{K_{\mathbf{1}}} \cdots \sum_{k_d=1}^{K_d} T_{k_{\mathbf{1}}}(x_1) \dots T_{k_d}(x_d) f(\bar{x}_1, \dots, \bar{x}_d)$$

Mappings

Mappings

Basic idea

Find a mapping $X \subset \mathbb{R}^d \to R$ such that

$$R = I_1 \times \cdots \times I_{d'}$$

and such that R is "small"

Mapping 1: Box

Find a box aligned with the data $X = \{x_1, \ldots, x_M\}$ using PCA:

- Translate points to the origin $\tilde{x}_i = x_i - c$ where c is the center
- Compute the axis of the box as the eigenvector of the covariance matrix $C_{ij} = \tilde{x}_i^\top \tilde{x}_j$
- Compute the length of each axis

Works well if points lie on planes ($d' < d$) or almost planar surfaces.

Mapping 2: Ellipsoid

Find an ellipsoid tightly fitting the data

- In general
$$f(x) = (x - x_c)^\top A(x - x_c) = 1$$
- Find x_c by taking the mean of the data
- Find A by
$$V = \text{argmin}_{A=A^\top} \sum_{i=1}^{n} \left((x_i - x_c)^\top A(x_i - x_c) - 1\right)^2$$
- $A = P\Lambda P^\top$ gives the axis (p_i) and the length ($\frac{1}{\sqrt{\lambda_i}}$) of the ellipsoid
- In $3d$, use this to build a polar coordinate representation of the data, minimizing the range of every coordinate

Numerical Results

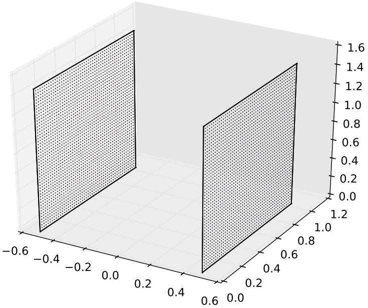
Algorithm

- Given
  - $K : X \times Y \to \mathbb{R}$,
  - $\{x_1, \ldots, x_M\} \subset X$ and $\{y_1, \ldots, y_N\} \subset Y$
  - Test points $\{\tilde{x}_1, \ldots, \tilde{x}_{M'}\} \subset X$ and $\{\tilde{y}_1, \ldots, \tilde{y}_{N'}\} \subset Y$
  - Mappings $M_x : X \to I_x^1 \times \cdots \times I_x^{d_x}$ and $M_y : Y \to I_y^1 \times \cdots \times I_y^{d_y}$
  - Tolerance δ
- Start with $n_x = [0, 0, \ldots, 0] \in \mathbb{N}^{d_x}$ and $n_y = [0, 0, \ldots, 0] \in \mathbb{N}^{d_y}$ and build $\bar{K}_{n_x,n_y}$ interpolating $K(M_x^{-1}(\cdot), M_y^{-1}(\cdot))$.
- While $\epsilon > \delta$,
  - For $z = \{x, y\}$ and for $i = 1, \ldots, d_z$
    - Increase $n_z[i]$ by 1 and build temporary $\bar{K}_{n_x,n_y}$
    - $\epsilon_{z,i} = \frac{\|\bar{K}_{n_x,n_y}(\tilde{x},\tilde{y}) - K(\tilde{x},\tilde{y})\|}{\|K(\tilde{x},\tilde{y})\|}$
  - Pick $(z, i) = \text{argmin}_{z,i} \epsilon_{z,i}$
  - $\epsilon = \epsilon_{z,i}$
  - $n_z[i] = n_z[i] + 1$, update $\bar{K}_{n_x,n_y}$
- This gives $\bar{K}_{n_x,n_y}$ of rank r_0
- Recompress to get $\bar{K}'_{n_x,n_y}$ of rank r_1.

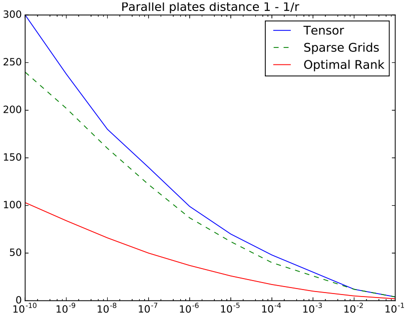
Experiments

- 2d and 3d geometries
- Multiple radial kernels
- Compare to optimal low-rank factorization (SVD)
- Re-compression (from rank r_0 to r_1) always brings the rank very close ($\sim 5 - 10\%$ max) to the optimal value (r) and is ommited in plots, where we show r_0 (before recompression, for usual method "Tensor" and more involved "Sparse Grids") and the optimal rank r
- Sparse Grids ideas (i.e. removing some well-choosen nodes from Tensor) also used

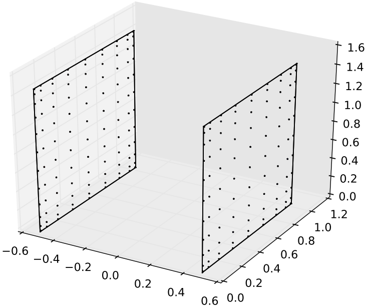
Parallel Plates: $1/r$



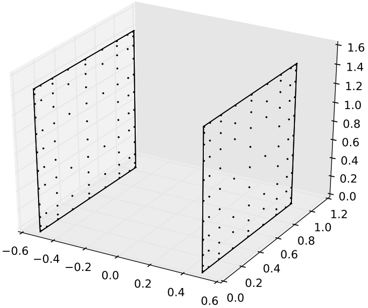
2500 x 2500 points



Parallel Plates: Sparse Grids

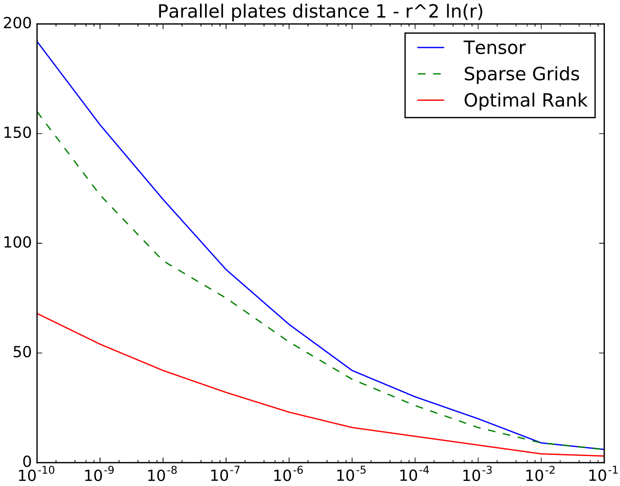


Tensor Grid

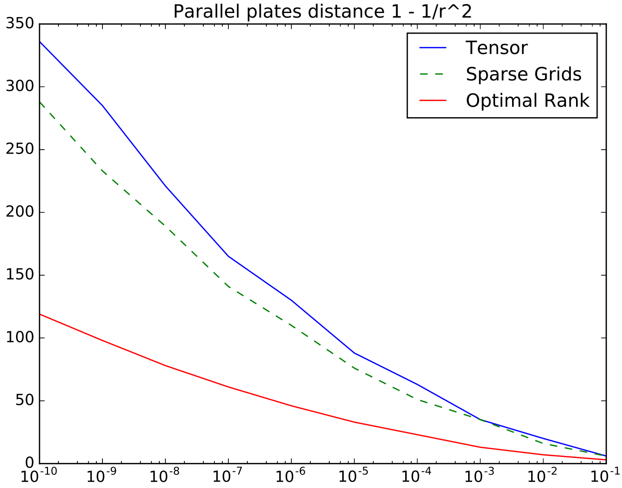


Sparse Grid

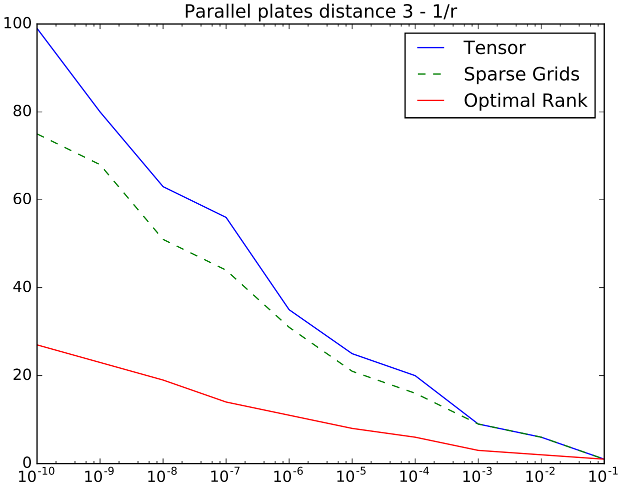
Parallel Plates: $r^2 \log(r)$



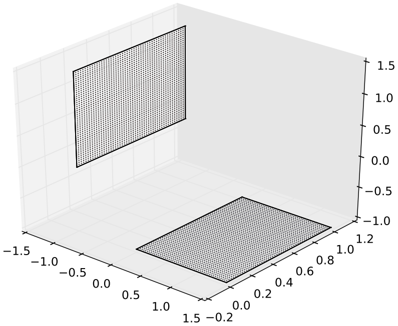
Parallel Plates: $1/r^2$

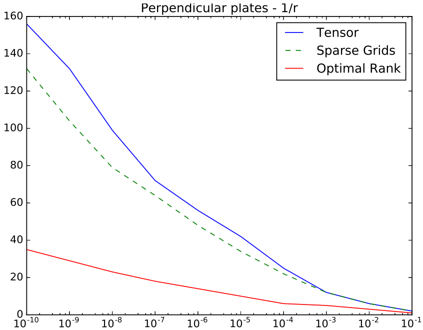


Parallel Plates - Increasing the Distance: $1/r$

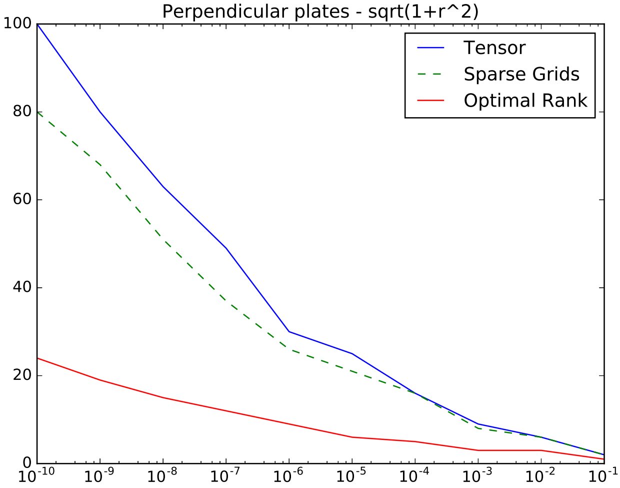


Perpendicular Plates: $1/r$

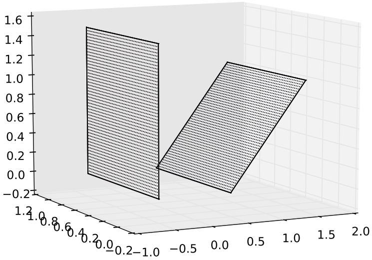


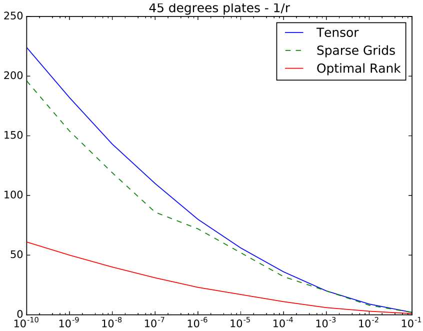


Perpendicular Plates: $\sqrt{1 + r^2}$

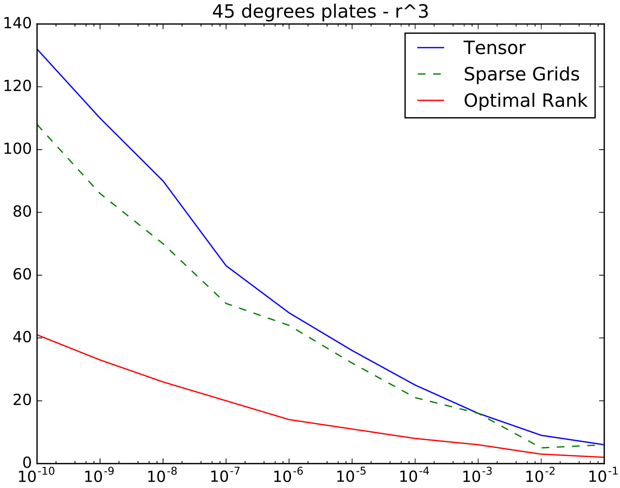


45° Plates: $1/r$

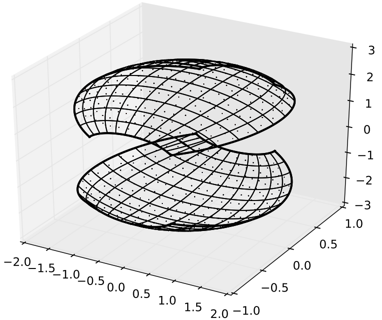

1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

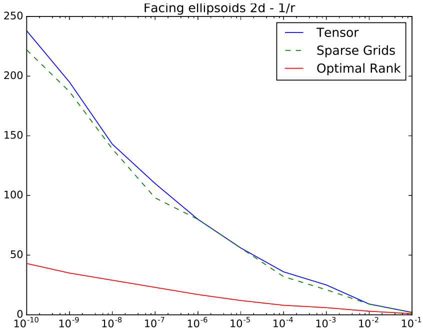

45 degrees plates - 1/r
Tensor
Sparse Grids
Optimal Rank
250
200
150
100
50
0
10^{-10}
10^{-9}
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}

45° Plates: r^3

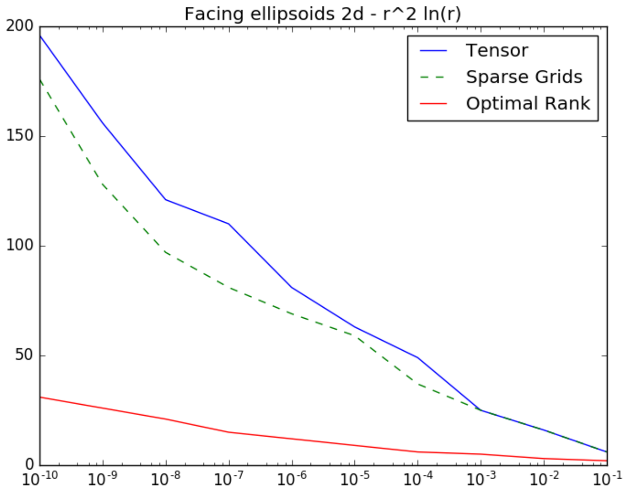

45 degrees plates - r^3
Tensor
Sparse Grids
Optimal Rank
140
120
100
80
60
40
20
0
10^-10
10^-9
10^-8
10^-7
10^-6
10^-5
10^-4
10^-3
10^-2
10^-1

Ellipsoid 2d: $1/r$

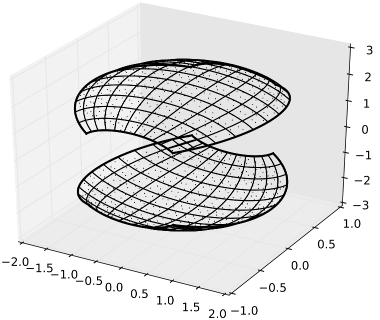


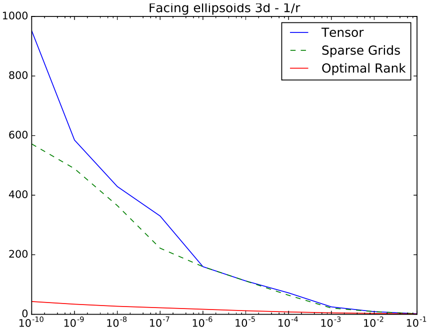


Ellipsoid 2d: $r^2 \log(r)$

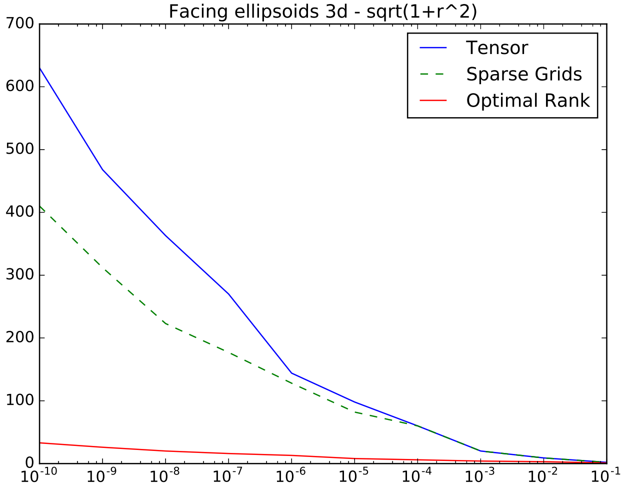


Ellipsoid 3d: $1/r$

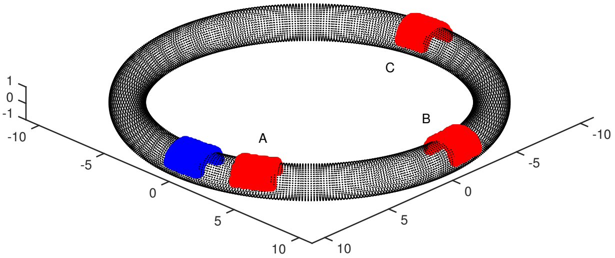




Ellipsoid 3d: $\sqrt{1 + r^2}$

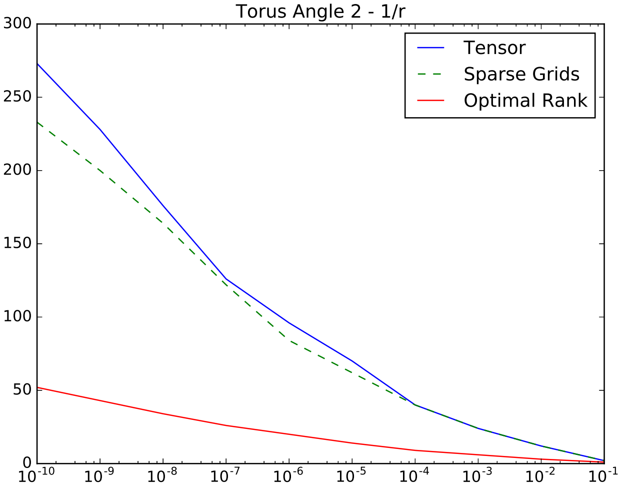


Torus 2d: $1/r$

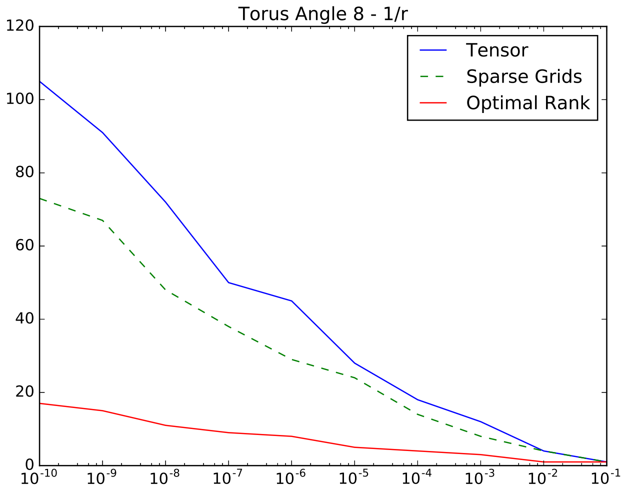


$50 \times 50 = 2500$ points per partition

Torus 2d - A: $1/r$

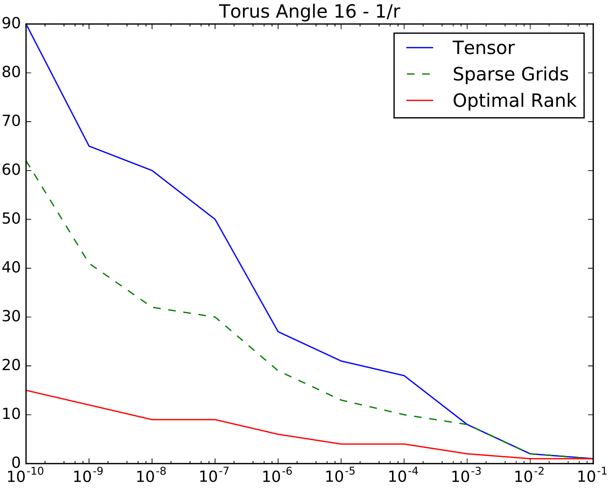

Torus Angle 2 - 1/r
300
250
200
150
100
50
0
10^-10
10^-9
10^-8
10^-7
10^-6
10^-5
10^-4
10^-3
10^-2
10^-1
Tensor
Sparse Grids
Optimal Rank

Torus 2d - B: $1/r$



Torus Angle 8 - 1/r
120
100
80
60
40
20
0
10^-10
10^-9
10^-8
10^-7
10^-6
10^-5
10^-4
10^-3
10^-2
10^-1
Tensor
Sparse Grids
Optimal Rank

Torus 2d - C: $1/r$


Torus Angle 16 - 1/r
Tensor
Sparse Grids
Optimal Rank
90
80
70
60
50
40
30
20
10
0
10^-10
10^-9
10^-8
10^-7
10^-6
10^-5
10^-4
10^-3
10^-2
10^-1

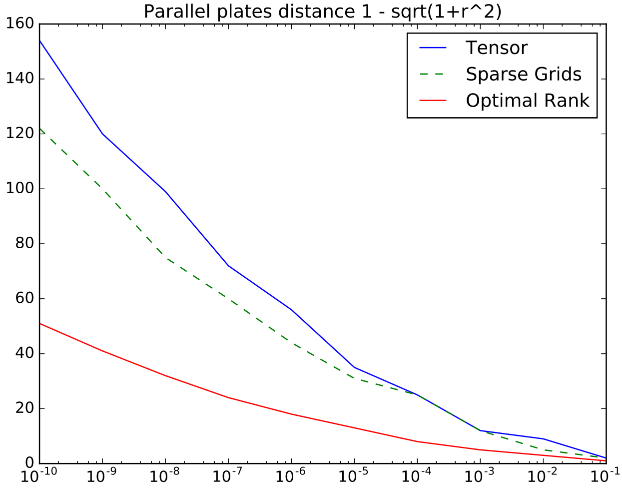
Low-Rank Kernel Matrix Approximation

- Method to approximate kernel matrices
- Independant of the size of the matrix
- Independant of the geometry, ...
- but requires a tight parametrization of the surface
- Can be improved by removing some well selected nodes ("Sparse Grids")

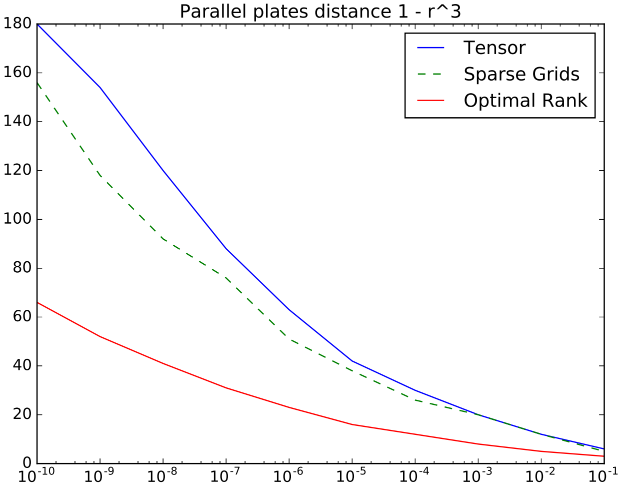
References

- Berrut, Jean-Paul, and Lloyd N. Trefethen. "Barycentric lagrange interpolation." SIAM review 46.3 (2004): 501-517.
- Kaarnioja, Vesa. "Smolyak quadrature." (2013). Master's Thesis

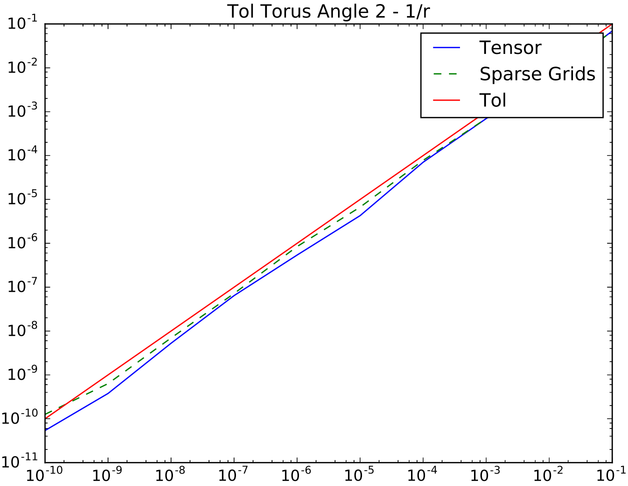
Parallel Plates: $\sqrt{1+r^2}$



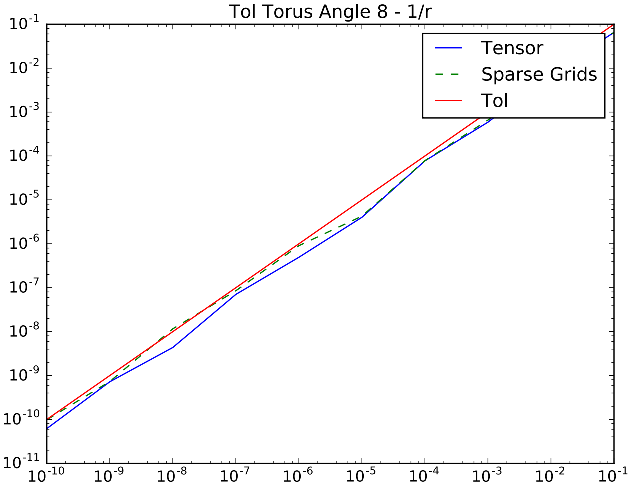
Parallel Plates: r^3



Torus 2d - A: Accuracy



Torus 2d - B: Accuracy



Torus 2d - C: Accuracy