Parallel Tensor Train Rounding using Gram SVD

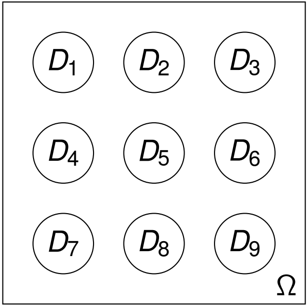
Hussam Al Daas[1], **Grey Ballard**[2], and Lawton Manning[2]

[1] Rutherford Appleton Laboratories
[2] Wake Forest University

SIAM Parallel Processing
February 26, 2022

WAKE FOREST
UNIVERSITY

Tensor Train (TT) can make very high dimensional problems tractable



Consider the parameter-dependent PDE:

$$\begin{aligned} -\mathrm{div}(\sigma(x,y;\boldsymbol{\rho})\nabla(u(x,y;\boldsymbol{\rho}))) &= f(x,y) && \text{in } \Omega, \\ u(x,y;\boldsymbol{\rho}) &= 0 && \text{on } \partial\Omega, \end{aligned}$$

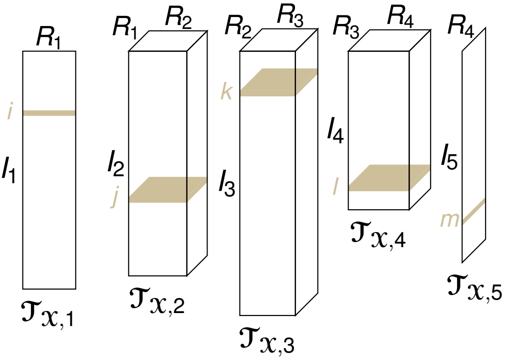
where σ is defined as:

$$\sigma(x,y;\boldsymbol{\rho}) = \begin{cases} 1+\rho_i & \text{if } (x,y) \in D_i \\ 1 & \text{elsewhere} \end{cases}$$

known as *cookies* problem [Tob12]

- Solving for all parameter values simultaneously, u is 11-D
- With mild assumptions, solution u has low TT ranks
- TT-based iterative linear solver exploits low-rank structure
  - can solve problem for high resolution [Dol13]

Tensor Train (TT) Notation



$\mathfrak{X} \approx \{\mathfrak{T}_{\mathfrak{X},n}\}, \mathfrak{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$

$\mathfrak{T}_{\mathfrak{X},n} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ are *TT cores*

$$x_{ijklm} \approx \sum_{\alpha=1}^{R_1} \sum_{\beta=1}^{R_2} \sum_{\gamma=1}^{R_3} \sum_{\delta=1}^{R_4} \mathfrak{T}_{\mathfrak{X},1}(i,\alpha)\mathfrak{T}_{\mathfrak{X},2}(\alpha,j,\beta)\mathfrak{T}_{\mathfrak{X},3}(\beta,k,\gamma)\mathfrak{T}_{\mathfrak{X},4}(\gamma,l,\delta)\mathfrak{T}_{\mathfrak{X},5}(\delta,m)$$

TT-Rounding

Given a tensor in TT format, often need to compress the ranks

- algebraic operations on TT formats over-extend ranks
- recompression (rank truncation) subject to error threshold
  - or subject to target ranks
- analogous to floating point rounding

TT-Rounding

Given a tensor in TT format, often need to compress the ranks

- algebraic operations on TT formats over-extend ranks
- recompression (rank truncation) subject to error threshold
  - or subject to target ranks
- analogous to floating point rounding

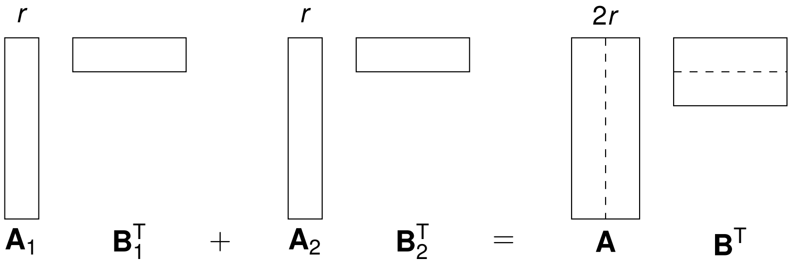
Goal: compute truncated SVDs of matricized TT-format tensors

- TT-ranks are ranks of unfoldings $\mathbf{X}_{(1:n)}$ for $1 \leq n \leq N$
- if $\mathcal{X}$ is $I_1 \times I_2 \times \cdots \times I_N$, then $\mathbf{X}_{(1:n)}$ is $(I_1 \cdots I_n) \times (I_{n+1} \cdots I_N)$
- $\mathbf{X}_{(1:n)}$ is generally a huge matrix, but it is highly structured

2D TT-Rounding (Matrix Case)

Low-rank matrix addition example

- consider $\mathbf{A}_1\mathbf{B}_1^T + \mathbf{A}_2\mathbf{B}_2^T$, where each factor has r columns
- can represent this in low-rank format $\begin{bmatrix}\mathbf{A}_1 & \mathbf{A}_2\end{bmatrix}\begin{bmatrix}\mathbf{B}_1 & \mathbf{B}_2\end{bmatrix}^T$ which now has rank $2r$
- goal is to compute low-rank approximation with rank $k < 2r$



Matrix Rounding via QR Decompositions

QR-based algorithm for rounding rank-r matrix $\mathbf{X} = \mathbf{A}\mathbf{B}^T$:

function $[\mathbf{U}, \mathbf{V}] = \text{QR-ROUNDING}(\mathbf{A}, \mathbf{B}, k)$

$[\mathbf{Q}_A, \mathbf{R}_A] = \text{QR}(\mathbf{A})$ ▷ (tall-skinny) QR decomposition

$[\mathbf{Q}_B, \mathbf{R}_B] = \text{QR}(\mathbf{B})$ ▷ (tall-skinny) QR decomposition

$[\hat{\mathbf{U}}_R, \hat{\Sigma}_R, \hat{\mathbf{V}}_R] = \text{TSVD}(\mathbf{R}_A \mathbf{R}_B^T, k)$ ▷ kth truncated SVD

$\mathbf{U} = \mathbf{Q}_A \hat{\mathbf{U}}_R$

$\mathbf{V} = \mathbf{Q}_B (\hat{\mathbf{V}}_R \hat{\Sigma}_R)$ ▷ $\mathbf{A}\mathbf{B}^T \approx \mathbf{U}\mathbf{V}^T$

Here's the algebra:

$$\mathbf{A}\mathbf{B}^\mathsf{T} = \underbrace{\mathbf{Q}_A \mathbf{R}_A}_{\mathbf{A}} \underbrace{\mathbf{R}_B^\mathsf{T} \mathbf{Q}_B^\mathsf{T}}_{\mathbf{B}^\mathsf{T}} \approx \mathbf{Q}_A \underbrace{\hat{\mathbf{U}}_R \hat{\Sigma}_R \hat{\mathbf{V}}_R^\mathsf{T}}_{\mathbf{R}_A \mathbf{R}_B^\mathsf{T}} \mathbf{Q}_B^\mathsf{T} = \underbrace{\mathbf{U}}_{\mathbf{Q}_A \hat{\mathbf{U}}_R} \underbrace{\mathbf{V}^\mathsf{T}}_{\hat{\Sigma}_R \hat{\mathbf{V}}_R^\mathsf{T} \mathbf{Q}_B^\mathsf{T}}$$

Matrix Rounding via Gram (1st attempt)

Gram SVD is a well-known method for computing singular values of rectangular matrices

- sing. values of $\mathbf{X}$ are square roots of eigenvalues of $\mathbf{X}^\mathsf{T}\mathbf{X}$
- if $\mathbf{X}$ is tall and skinny then $\mathbf{X}^\mathsf{T}\mathbf{X}$ is much smaller
- sacrifices some accuracy of singular values and vectors

If $\mathbf{X} = \mathbf{A}\mathbf{B}^\mathsf{T}$, then we can try direct application:

- $\mathbf{X}^\mathsf{T}\mathbf{X} = \mathbf{B}(\mathbf{A}^\mathsf{T}\mathbf{A})\mathbf{B}^\mathsf{T}$
- $\mathbf{X}\mathbf{X}^\mathsf{T} = \mathbf{A}(\mathbf{B}^\mathsf{T}\mathbf{B})\mathbf{A}^\mathsf{T}$

but we still have to orthogonalize one of the two factors in order to compute the eigendecomposition

Matrix Rounding via Gram SVD

Gram-based algorithm for rounding rank-r matrix $\mathbf{X} = \mathbf{A}\mathbf{B}^T$:

function $[\mathbf{U}, \mathbf{V}] = \textsc{Gram-Rounding}(\mathbf{A}, \mathbf{B}, k)$

$\mathbf{G}_A = \mathbf{A}^\mathsf{T}\mathbf{A}$ ▷ symmetric matrix multiplication

$\mathbf{G}_B = \mathbf{B}^\mathsf{T}\mathbf{B}$ ▷ symmetric matrix multiplication

$[\mathbf{V}_A, \mathbf{\Lambda}_A] = \textsc{Eig}(\mathbf{G}_A)$

$[\mathbf{V}_B, \mathbf{\Lambda}_B] = \textsc{Eig}(\mathbf{G}_B)$

$[\hat{\mathbf{U}}, \hat{\mathbf{\Sigma}}, \hat{\mathbf{V}}] = \textsc{tSVD}(\mathbf{\Lambda}_A^{1/2}\mathbf{V}_A^\mathsf{T}\mathbf{V}_B\mathbf{\Lambda}_B^{1/2}, k)$ ▷ kth truncated SVD

$\mathbf{U} = \mathbf{A}\left(\mathbf{V}_A\mathbf{\Lambda}_A^{-1/2}\hat{\mathbf{U}}\right)$

$\mathbf{V} = \mathbf{B}\left(\mathbf{V}_B\mathbf{\Lambda}_B^{-1/2}\hat{\mathbf{V}}\hat{\mathbf{\Sigma}}\right)$ ▷ $\mathbf{A}\mathbf{B}^T \approx \mathbf{U}\mathbf{V}^T$

Matrix Rounding via Gram SVD (continued)

Here's the algebra:

$$
\begin{aligned}
\mathbf{A}\mathbf{B}^{\mathsf{T}} &= \underbrace{\mathbf{U}_A \boldsymbol{\Sigma}_A \mathbf{V}_A^{\mathsf{T}}}_{\mathbf{A}} \underbrace{\mathbf{V}_B \boldsymbol{\Sigma}_B \mathbf{U}_B^{\mathsf{T}}}_{\mathbf{B}^{\mathsf{T}}} \\
&= \underbrace{\mathbf{A}\mathbf{V}_A \boldsymbol{\Lambda}_A^{-1/2} \boldsymbol{\Lambda}_A^{1/2} \mathbf{V}_A^{\mathsf{T}}}_{\mathbf{A}} \underbrace{\mathbf{V}_B \boldsymbol{\Lambda}_B^{1/2} \boldsymbol{\Lambda}_B^{-1/2} \mathbf{V}_B^{\mathsf{T}} \mathbf{B}^{\mathsf{T}}}_{\mathbf{B}^{\mathsf{T}}} \\
&= \underbrace{(\mathbf{A}\mathbf{V}_A \boldsymbol{\Lambda}_A^{-1/2})}_{\mathbf{U}_A} \underbrace{\boldsymbol{\Lambda}_A^{1/2} \mathbf{V}_A^{\mathsf{T}} \mathbf{V}_B \boldsymbol{\Lambda}_B^{1/2}}_{\mathbf{M}} \underbrace{(\mathbf{B}\mathbf{V}_B \boldsymbol{\Lambda}_B^{-1/2})}_{\mathbf{U}_B}{}^{\mathsf{T}} \\
&\approx \underbrace{(\mathbf{A}\mathbf{V}_A \boldsymbol{\Lambda}_A^{-1/2})}_{\mathbf{U}_A} \hat{\mathbf{U}} \hat{\boldsymbol{\Sigma}} \hat{\mathbf{V}}^{\mathsf{T}} \underbrace{(\mathbf{B}\mathbf{V}_B \boldsymbol{\Lambda}_B^{-1/2})}_{\mathbf{U}_B}{}^{\mathsf{T}} \\
&= \underbrace{\mathbf{A}(\mathbf{V}_A \boldsymbol{\Lambda}_A^{-1/2} \hat{\mathbf{U}})}_{\mathbf{U}} \underbrace{(\hat{\boldsymbol{\Sigma}} \hat{\mathbf{V}}^{\mathsf{T}} \boldsymbol{\Lambda}_B^{-1/2} \mathbf{V}_B^{\mathsf{T}}) \mathbf{B}^{\mathsf{T}}}_{\mathbf{V}^{\mathsf{T}}}
\end{aligned}
$$

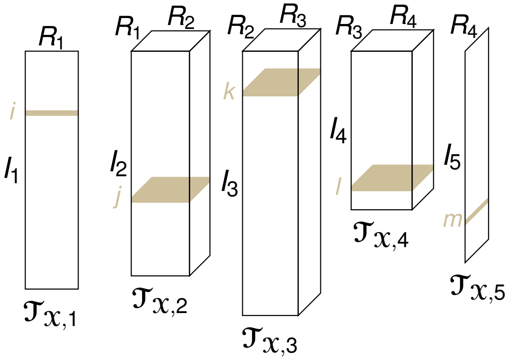
QR-Rounding vs Gram-Rounding

Gram-Rounding is at least twice as fast as QR-Rounding

- Gram-Rounding does half the flops of QR-Rounding
- Gram-Rounding dominated by matrix multiplication
- QR-Rounding dominated by computing QR and applying Q

QR-Rounding is more accurate than Gram-Rounding

- QR-Rounding computes singular values as small as $\sigma_1 \cdot \varepsilon$
- Gram-Rounding computes singular values only to $\sigma_1 \cdot \varepsilon^{1/2}$

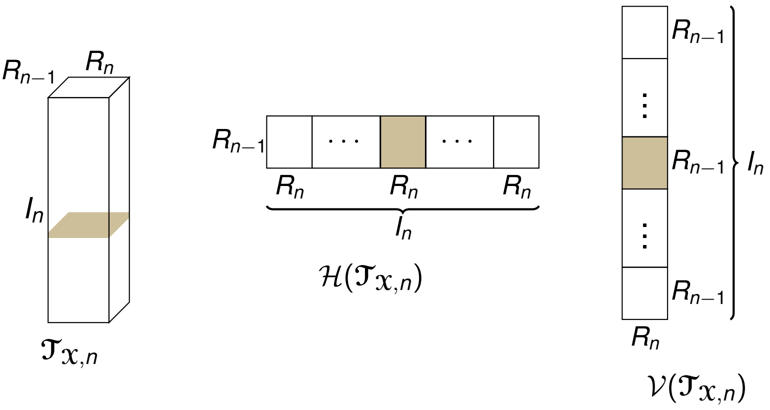


$\mathcal{X} \approx \{\mathcal{T}_{\mathcal{X},n}\}, \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$

$\mathcal{T}_{\mathcal{X},n} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$ are *TT cores*

$$x_{ijklm} \approx \sum_{\alpha=1}^{R_1} \sum_{\beta=1}^{R_2} \sum_{\gamma=1}^{R_3} \sum_{\delta=1}^{R_4} \mathcal{T}_{\mathcal{X},1}(i,\alpha)\mathcal{T}_{\mathcal{X},2}(\alpha,j,\beta)\mathcal{T}_{\mathcal{X},3}(\beta,k,\gamma)\mathcal{T}_{\mathcal{X},4}(\gamma,l,\delta)\mathcal{T}_{\mathcal{X},5}(\delta,m)$$

$\mathcal{H}(\mathfrak{T}_{\mathcal{X},n}) \in \mathbb{R}^{R_{n-1} \times I_n R_n}$ and $\mathcal{V}(\mathfrak{T}_{\mathcal{X},n}) \in \mathbb{R}^{R_{n-1} I_n \times R_n}$ are horizontal and vertical unfoldings of nth core



QR-Based TT Rounding Algorithm [Ose11]

function $\{\mathcal{T}_{\mathcal{Z},n}\} = \text{TT-ROUNDING}(\{\mathcal{T}_{\mathcal{X},n}\})$

$\triangleright$ Orthogonalization Phase

for $n = N$ down to 2 **do**

$[\mathbf{Y}_n, \mathbf{R}_n] = \text{QR}(\mathcal{H}(\mathcal{T}_{\mathcal{X},n})^T)$ $\triangleright$ (tall-skinny) QR factorization

$\mathcal{V}(\mathcal{T}_{\mathcal{X},n-1}) = \mathcal{V}(\mathcal{T}_{\mathcal{X},n-1}) \cdot \mathbf{R}^\mathsf{T}$ $\triangleright$ Apply **R** to previous core

$\triangleright$ Truncation Phase

$\mathcal{Z} = \mathcal{X}$

for $n = 1$ to $N - 1$ **do**

$[\mathbf{Y}_n, \mathbf{R}_n] = \text{QR}(\mathcal{V}(\mathcal{T}_{\mathcal{Z},n}))$ $\triangleright$ (tall-skinny) QR factorization

$[\hat{\mathbf{U}}_R, \hat{\Sigma}, \hat{\mathbf{V}}] \approx \text{TSVD}(\mathbf{R}_n)$ $\triangleright$ Truncated SVD of **R**

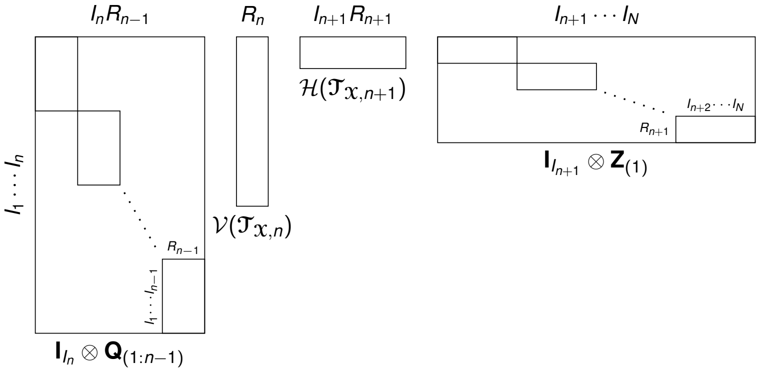
$\mathcal{V}(\mathcal{T}_{\mathcal{Z},n}) = \text{APPLY-Q}(\mathbf{Y}_n, \hat{\mathbf{U}}_R)$ $\triangleright$ Form explicit $\hat{\mathbf{U}}$

$\mathcal{H}(\mathcal{T}_{\mathcal{Z},n+1})^T = \text{APPLY-Q}(\mathbf{Y}_{n+1}, \hat{\mathbf{V}}\hat{\Sigma})$ $\triangleright$ Apply $\hat{\Sigma}\hat{\mathbf{V}}^\mathsf{T}$ to next core

We have parallelized this QR-based algorithm [DBB22] using the parallel TSQR algorithm [DGHL12]

TT-Rounding does truncated SVDs on $\mathbf{X}_{(1)}$, $\mathbf{X}_{(1:2)}$, $\mathbf{X}_{(1:3)}$, etc., and here is the matrix expression of each unfolding [DBB22]:

$$\mathbf{X}_{(1:n)} = (\mathbf{I}_{I_n} \otimes \mathbf{Q}_{(1:n-1)}) \cdot \mathcal{V}(\mathfrak{T}_{\mathcal{X},n}) \cdot \mathcal{H}(\mathfrak{T}_{\mathcal{X},n+1}) \cdot (\mathbf{I}_{I_{n+1}} \otimes \mathbf{Z}_{(1)})$$



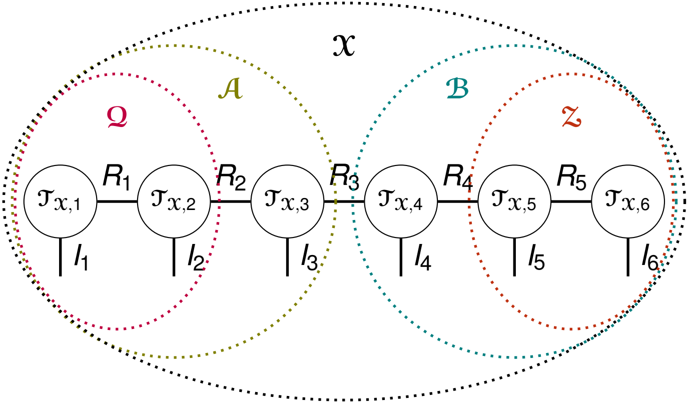
Main Ideas of Gram-Based TT-Rounding

Same matrix expression of nth unfolding:

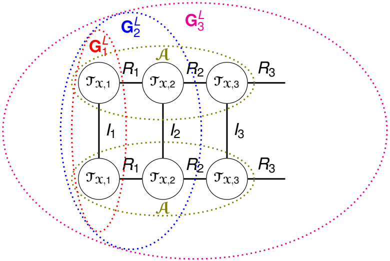
$$\mathbf{X}_{(1:n)} = \underbrace{(\mathbf{I}_{I_n} \otimes \mathbf{Q}_{(1:n-1)}) \cdot \mathcal{V}(\mathcal{T}_{\mathcal{X},n})}_{\mathbf{A}} \cdot \underbrace{\mathcal{H}(\mathcal{T}_{\mathcal{X},n+1}) \cdot (\mathbf{I}_{I_{n+1}} \otimes \mathbf{Z}_{(1)})}_{\mathbf{B}^{\mathsf{T}}}$$

- just like matrix case, we need to compute $\mathbf{A}^{\mathsf{T}}\mathbf{A}$ and $\mathbf{B}^{\mathsf{T}}\mathbf{B}$
- two key differences:
  - $\mathbf{A}$ and $\mathbf{B}$ are both highly structured
  - we need Gram matrices for all $1 \leq n \leq N$
- we obtain efficiency by exploiting structure and by exploiting the computational overlap across modes

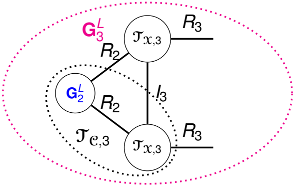
$$\mathbf{X}_{(1:n)} = \underbrace{(\mathbf{I}_{I_n} \otimes \mathbf{Q}_{(1:n-1)}) \cdot \mathcal{V}(\mathfrak{T}_{\mathfrak{X},n})}_{\mathbf{A}} \cdot \underbrace{\mathcal{H}(\mathfrak{T}_{\mathfrak{X},n+1}) \cdot (\mathbf{I}_{I_{n+1}} \otimes \mathbf{Z}_{(1)})}_{\mathbf{B}^\mathsf{T}}$$



Tensor Network Diagram for Gram Matrix Computation

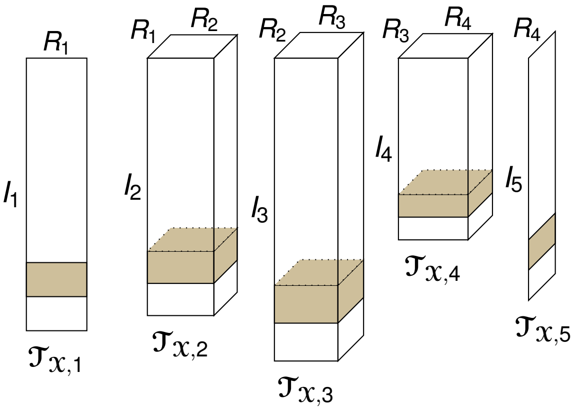


tensor network for $\mathbf{G}_3^L = \mathbf{A}^\mathsf{T}\mathbf{A}$



intermediate step,
computing $\mathbf{G}_3^L$ from $\mathbf{G}_2^L$

Parallel Distribution

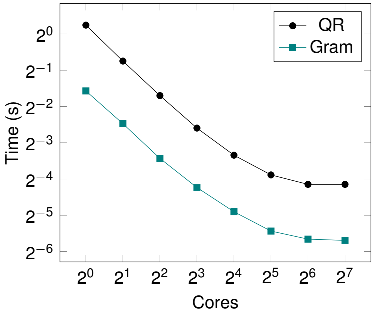


- Each core distributed across all P processors
- Local nth core dimensions are $R_{n-1} \times \frac{I_n}{P} \times R_n$
- Key: vertical and horizontal unfoldings are 1D-distributed

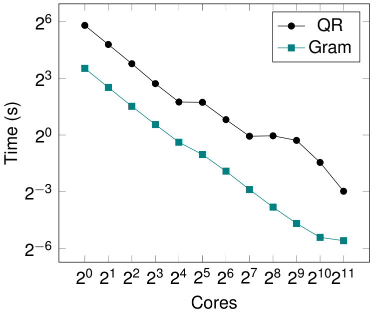
Strong Scaling Results for Synthetic Tensors

We round synthetic tensors with input TT-ranks $R = 20$ (all modes) down to $R = 10$, scaling up the number of processors

- results on Andes (ORNL), with 2 16-core AMD EPYC procs per node



N=50, $2K \times \cdots \times 2K$ (77 MB)

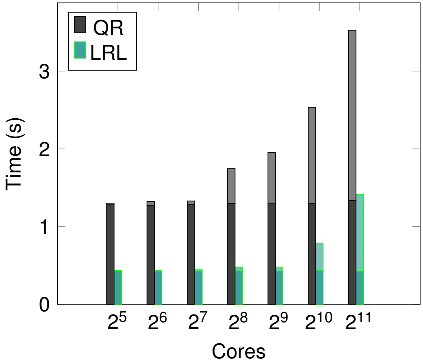


N=16, $100M \times 50K \times \cdots \times 50K \times 1M$ (8 GB)

Weak Scaling Results for Synthetic Tensor

We round the 50-mode synthetic tensor with input TT-ranks $R = 20$ down to $R = 10$, scaling up processors with fixed local data

- results on Andes (ORNL), with 2 16-core AMD EPYC procs per node

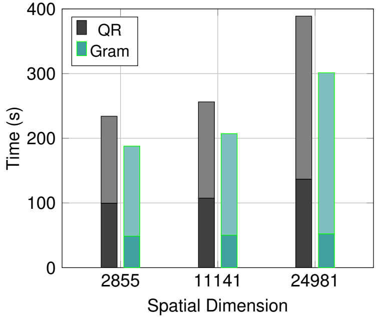


Dark signifies computation, light signifies communication

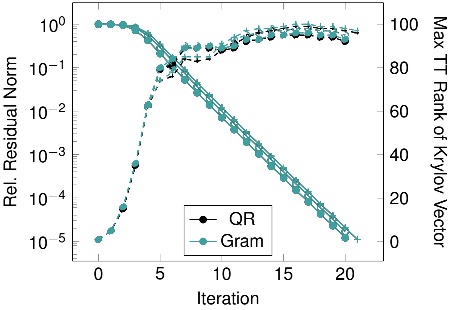
Results for Cookies Problem

We solve the Cookies problem with Matlab implementation of TT-GMRES [Dol13] and TT-Rounding (sequential)

- we use 4 parameters (cookies) and vary spatial discretization



Dark signifies time in TT-Rounding



Gram has negligible effect on GMRES behavior

Summary

- TT format efficiently approximates high-dim. tensors
  - TT arithmetic causes rank growth, TT-Rounding is key
  - enables solving problems like parameter-dependent PDEs
- TT-Rounding means truncated SVDs of unfoldings $\mathbf{X}_{(1:n)}$
  - QR-based approach most accurate but less efficient
  - Gram-based approach faster, can be accurate enough
- Gram-based computation exploits TT structure
  - efficient contraction of tensor network
  - par. distribution allows for comm.-efficient matrix multiplies

Thanks for your attention!

ballard@wfu.edu

MPI_ATTAC: Algorithms for Tensor Train Arithmetic and Computations

https://gitlab.com/aldaas/mpi_attac

Hussam Al Daas, Grey Ballard, and Lawton Manning.
Parallel Tensor Train Rounding using Gram SVD.
To appear in IPDPS 2022.

Hussam Al Daas, Grey Ballard, and Peter Benner.
Parallel Algorithms for Tensor Train Arithmetic.
SIAM Journal on Scientific Computing 2022.
https://dx.doi.org/10.1137/20M1387158

More details on TT-Rounding...

TT-Rounding does SVDs on $\mathbf{X}_{(1)}$, $\mathbf{X}_{(1:2)}$, $\mathbf{X}_{(1:3)}$, etc., so we seek similar matrix expressions of those unfoldings

The unfolding of $\mathcal{X}$ that maps the first n tensor dimensions to rows can be expressed as a product of four matrices:

$$\mathbf{X}_{(1:n)} = (\mathbf{I}_{I_n} \otimes \mathbf{Q}_{(1:n-1)}) \cdot \mathcal{V}(\mathfrak{T}_{\mathcal{X},n}) \cdot \mathcal{H}(\mathfrak{T}_{\mathcal{X},n+1}) \cdot (\mathbf{I}_{I_{n+1}} \otimes \mathbf{Z}_{(1)})$$

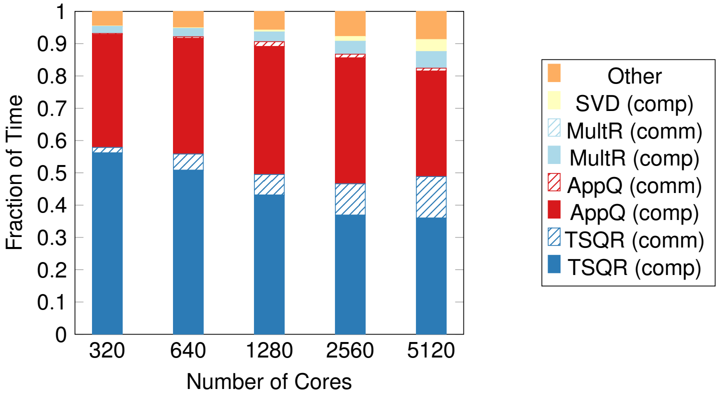
where $\mathfrak{Q}$ is $I_1 \times \cdots \times I_{n-1} \times R_{n-1}$ with

$$\mathfrak{Q}(i_1, \ldots, i_{n-1}, r_{n-1}) = \mathfrak{T}_{\mathcal{X},1}(i_1, :) \cdot \mathfrak{T}_{\mathcal{X},2}(:, i_2, :) \cdots \mathfrak{T}_{\mathcal{X},n-1}(:, i_{n-1}, r_{n-1}),$$

and $\mathfrak{Z}$ is $R_{n+1} \times I_{n+2} \times \cdots \times I_N$ with

$$\mathfrak{Z}(r_{n+1}, i_{n+2}, \ldots, i_N) = \mathfrak{T}_{\mathcal{X},n+2}(r_{n+1}, i_{n+2}, :) \cdot \mathfrak{T}_{\mathcal{X},n+3}(:, i_{n+3}, :) \cdots \mathfrak{T}_{\mathcal{X},N}(:, i_N).$$

Time Breakdown of Parallel QR-based TT-Rounding



- TT tensor: $I_n = 512K$, $R_n = 60 \to 30$, $N = 50$
- 70-80% of time spent in QR computations

References I

Hussam Al Daas, Grey Ballard, and Peter Benner.
Parallel algorithms for tensor train arithmetic.
SIAM Journal on Scientific Computing, 44(1):C25–C53, 2022.

Jim Demmel, Laura Grigori, Mark Hoemmen, and Julien Langou.
Communication-optimal parallel and sequential QR and LU factorizations.
SIAM Journal on Scientific Computing, 34(1):A206–A239, 2012.

Sergey V. Dolgov.
TT-GMRES: solution to a linear system in the structured tensor format.
Russian Journal of Numerical Analysis and Mathematical Modelling, 28(2):149–172, 2013.

Ivan Oseledets.
Tensor-train decomposition.
SIAM Journal on Scientific Computing, 33(5):2295–2317, 2011.

Christine Tobler.
Low-rank Tensor Methods for Linear Systems and Eigenvalue Problems.
PhD thesis, ETH Zurich, 2012.