MAX PLANCK INSTITUTE
FOR DYNAMICS OF COMPLEX
TECHNICAL SYSTEMS
MAGDEBURG

CSC

COMPUTATIONAL METHODS IN
SYSTEMS AND CONTROL THEORY

Parallel Algorithms for Tensor Train Arithmetic

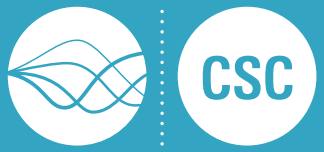
Peter Benner

joint work with:
Hussam Al Daas (STFC/RAL, UK)
Grey Ballard (Wake Forest U., USA)

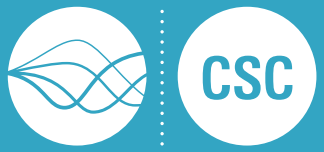
IPAM Long Program
"Tensor Methods and Emerging Applications
to the Physical and Data Sciences"

Workshop I:
Tensor Methods and their Applications in the Physical and Data Sciences

March 29 – April 1, 2021
(virtual)



Outline



Outline

Main Goal

Design a software library for

- iteratively solving linear systems

$$\mathbf{A}\mathcal{X} = \mathcal{B},$$

Main Goal

Design a software library for

- iteratively solving linear systems

$$\mathbf{A}\mathcal{X} = \mathcal{B},$$

- with data and solution in low-rank tensor, specifically, Tensor Train (TT) format,

Main Goal

Design a software library for

- iteratively solving linear systems
$$\mathbf{A}\mathcal{X} = \mathcal{B},$$
- with data and solution in low-rank tensor, specifically, Tensor Train (TT) format,
- on distributed-memory parallel computers with high scalability.

Main Goal

Design a software library for

- iteratively solving linear systems
$$\mathbf{A}\mathcal{X} = \mathcal{B},$$
- with data and solution in low-rank tensor, specifically, Tensor Train (TT) format,
- on distributed-memory parallel computers with high scalability.

This Talk (based on [Al Daas/Ballard/B. arXiv:2011:06532])

Scalable parallel algorithms for TT arithmetic:

- scaling and addition
- Hadamard (element-wise) and (Frobenius) inner product/norm
- TT rounding/compression

Optimization under Uncertainty

Consider the problem:

$$\min_{y \in \mathcal{Y}, u \in \mathcal{U}} \mathcal{J}(y,u) \quad \text{subject to} \quad c(y,u) = 0,$$

where

- $c(y,u) = 0$ represents a (linear or nonlinear) PDE (system) with uncertain coefficient(s).
- The state y and control u are random fields.
- The cost functional $\mathcal{J}$ is a real-valued Fréchet-differentiable functional on $\mathcal{Y} \times \mathcal{U}$.

Optimization under Uncertainty

Consider the problem:

$$\min_{y \in \mathcal{Y}, u \in \mathcal{U}} \mathcal{J}(y,u) \quad \text{subject to} \quad c(y,u) = 0,$$

where

- $c(y,u) = 0$ represents a (linear or nonlinear) PDE (system) with uncertain coefficient(s).
- The state y and control u are random fields.
- The cost functional $\mathcal{J}$ is a real-valued Fréchet-differentiable functional on $\mathcal{Y} \times \mathcal{U}$.

Solution $y = y(u)$ depends on

Optimization under Uncertainty

Consider the problem:

$$\min_{y \in \mathcal{Y}, u \in \mathcal{U}} \mathcal{J}(y,u) \quad \text{subject to} \quad c(y,u) = 0,$$

where

- $c(y,u) = 0$ represents a (linear or nonlinear) PDE (system) with uncertain coefficient(s).
- The state y and control u are random fields.
- The cost functional $\mathcal{J}$ is a real-valued Fréchet-differentiable functional on $\mathcal{Y} \times \mathcal{U}$.

Solution $y = y(u)$ depends on

- time ($\rightsquigarrow$ 1 dimension),

Optimization under Uncertainty

Consider the problem:

$$\min_{y \in \mathcal{Y}, u \in \mathcal{U}} \mathcal{J}(y,u) \quad \text{subject to} \quad c(y,u) = 0,$$

where

- $c(y,u) = 0$ represents a (linear or nonlinear) PDE (system) with uncertain coefficient(s).
- The state y and control u are random fields.
- The cost functional $\mathcal{J}$ is a real-valued Fréchet-differentiable functional on $\mathcal{Y} \times \mathcal{U}$.

Solution $y = y(u)$ depends on

- time ($\rightsquigarrow$ 1 dimension),
- spatial coordinates ($\rightsquigarrow$ $d \in \{1,2,3\}$ dimensions),

Optimization under Uncertainty

Consider the problem:

$$\min_{y \in \mathcal{Y}, u \in \mathcal{U}} \mathcal{J}(y,u) \quad \text{subject to} \quad c(y,u) = 0,$$

where

- $c(y,u) = 0$ represents a (linear or nonlinear) PDE (system) with uncertain coefficient(s).
- The state y and control u are random fields.
- The cost functional $\mathcal{J}$ is a real-valued Fréchet-differentiable functional on $\mathcal{Y} \times \mathcal{U}$.

Solution $y = y(u)$ depends on

- time ($\rightsquigarrow$ 1 dimension),
- spatial coordinates ($\rightsquigarrow$ $d \in \{1,2,3\}$ dimensions),
- parameters, e.g. Karhunen-Loève coefficients ($\rightsquigarrow$ $p \in \{1,\ldots\}$ dimensions).

Optimization under Uncertainty

Consider the problem:

$$\min_{y \in \mathcal{Y}, u \in \mathcal{U}} \mathcal{J}(y,u) \quad \text{subject to} \quad c(y,u) = 0,$$

where

- $c(y,u) = 0$ represents a (linear or nonlinear) PDE (system) with uncertain coefficient(s).
- The state y and control u are random fields.
- The cost functional $\mathcal{J}$ is a real-valued Fréchet-differentiable functional on $\mathcal{Y} \times \mathcal{U}$.

Solution $y = y(u)$ depends on

- time ($\rightsquigarrow$ 1 dimension/mode),
- spatial coordinates ($\rightsquigarrow$ $d \in \{1,2,3\}$ dimensions/modes),
- parameters, e.g. Karhunen-Loève coefficients ($\rightsquigarrow$ $p \in \{1,\ldots\}$ dimensions/modes).

Optimization under Uncertainty

Consider the problem:

$$\min_{y \in \mathcal{Y}, u \in \mathcal{U}} \mathcal{J}(y,u) \quad \text{subject to} \quad c(y,u) = 0,$$

where

- $c(y,u) = 0$ represents a (linear or nonlinear) PDE (system) with uncertain coefficient(s).
- The state y and control u are random fields.
- The cost functional $\mathcal{J}$ is a real-valued Fréchet-differentiable functional on $\mathcal{Y} \times \mathcal{U}$.

Solution $y = y(u)$ depends on

- time ($\rightsquigarrow$ 1 dimension/mode),
- spatial coordinates ($\rightsquigarrow$ $d \in \{1,2,3\}$ dimensions/modes),
- parameters, e.g. Karhunen-Loève coefficients ($\rightsquigarrow$ $p \in \{1,\ldots\}$ dimensions/modes).

$\Rightarrow$ Solution can be considered as $(1+d+p)$-way tensor!

Curse of Dimensionality [Bellman 1957]

Increase of matrix size of discretized differential operator for $h \to \frac{h}{2}$ by factor 2^D.

$\rightsquigarrow$ **Rapid Increase of Dimensionality**, called **Curse of Dimensionality** $(D = (1 + d + p) > 3)$.

Curse of Dimensionality [Bellman 1957]

Increase of matrix size of discretized differential operator for $h \to \frac{h}{2}$ by factor 2^D.

$\rightsquigarrow$ **Rapid Increase of Dimensionality**, called **Curse of Dimensionality** $(D = (1 + d + p) > 3)$.

Solution technique for PDE-constrained optimization under uncertainty

- Low-rank iterative solvers for optimality systems related to PDE-contraints defined by
  - unsteady heat equation,
  - Stokes-Brinkman problem (flow through a porous medium),
  - Navier-Stokes equations.

Curse of Dimensionality [Bellman 1957]

Increase of matrix size of discretized differential operator for $h \to \frac{h}{2}$ by factor 2^D.

$\rightsquigarrow$ **Rapid Increase of Dimensionality**, called **Curse of Dimensionality** $(D = (1 + d + p) > 3)$.

Solution technique for PDE-constrained optimization under uncertainty

- Low-rank iterative solvers for optimality systems related to PDE-contraints defined by
  - unsteady heat equation,
  - Stokes-Brinkman problem (flow through a porous medium),
  - Navier-Stokes equations.
- Discretization using tensor product stochastic Galerkin FEM (polynomial chaos).

Curse of Dimensionality [Bellman 1957]

Increase of matrix size of discretized differential operator for $h \to \frac{h}{2}$ by factor 2^D.

$\rightsquigarrow$ **Rapid Increase of Dimensionality**, called **Curse of Dimensionality** $(D = (1 + d + p) > 3)$.

Solution technique for PDE-constrained optimization under uncertainty

- Low-rank iterative solvers for optimality systems related to PDE-contraints defined by
  - unsteady heat equation,
  - Stokes-Brinkman problem (flow through a porous medium),
  - Navier-Stokes equations.
- Discretization using tensor product stochastic Galerkin FEM (polynomial chaos).
- Developed appropriate preconditioned TT-MINRES and block-ALS/AMEn solvers for indefinite saddle-point systems.

Curse of Dimensionality [Bellman 1957]

Increase of matrix size of discretized differential operator for $h \to \frac{h}{2}$ by factor 2^D.

$\rightsquigarrow$ **Rapid Increase of Dimensionality**, called **Curse of Dimensionality** $(D = (1 + d + p) > 3)$.

Solution technique for PDE-constrained optimization under uncertainty

- Low-rank iterative solvers for optimality systems related to PDE-contraints defined by
  - unsteady heat equation,
  - Stokes-Brinkman problem (flow through a porous medium),
  - Navier-Stokes equations.
- Discretization using tensor product stochastic Galerkin FEM (polynomial chaos).
- Developed appropriate preconditioned TT-MINRES and block-ALS/AMEn solvers for indefinite saddle-point systems.

Achievements

Biggest problem solved so far has $n = 1.29 \cdot 10^{15}$ unknowns (optimality system for unsteady incompressible Navier-Stokes control problem with uncertain viscosity).

Would require ≈ 10 **petabytes (PB)** $= 10,000$ **TB** to store the solution vector!

With low-rank TT techniques, solution of optimality system requires $\approx 7 \cdot 10^7$ **bytes** $= 70$ **GB.**

This explores the limits of shared-memory computations. . . already for very small p!

Besides PDEs with (uncertain) parameters, low-rank tensor techniques can deal with high-order data efficiently in many applications such as

- Molecular Simulation
- Uncertainty Quantification (UQ)
- Machine Learning
- ...

Besides PDEs with (uncertain) parameters, low-rank tensor techniques can deal with high-order data efficiently in many applications such as

- Molecular Simulation
- Uncertainty Quantification (UQ)
- Machine Learning
- ...

In Molecular Simulation and UQ (N-point correlations)

- $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$,
- with a large number of modes (N can be of order 100), and
- mode size can be as large as $I_i \approx 10^6$,

while in parametric PDE numerics, mode sizes in the solution tensor often vary significantly, between a few tens and millions.

Runtime estimation of an algorithm

$$\text{Computation complexity} = \#\text{flops}\ \gamma,$$

where γ time/flop

Runtime estimation of an algorithm

$$\begin{aligned} \text{Computation complexity} &= \#\text{flops } \gamma, \\ \text{Communication complexity} &= \#\text{words } \beta + \alpha, \end{aligned}$$

where γ time/flop, β time/word, α time to send a message.

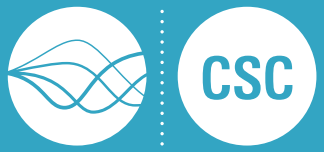
Runtime estimation of an algorithm

$$\begin{aligned} \text{Computation complexity} &= \#\text{flops } \gamma, \\ \text{Communication complexity} &= \#\text{words } \beta + \alpha, \end{aligned}$$

where γ time/flop, β time/word, α time to send a message.

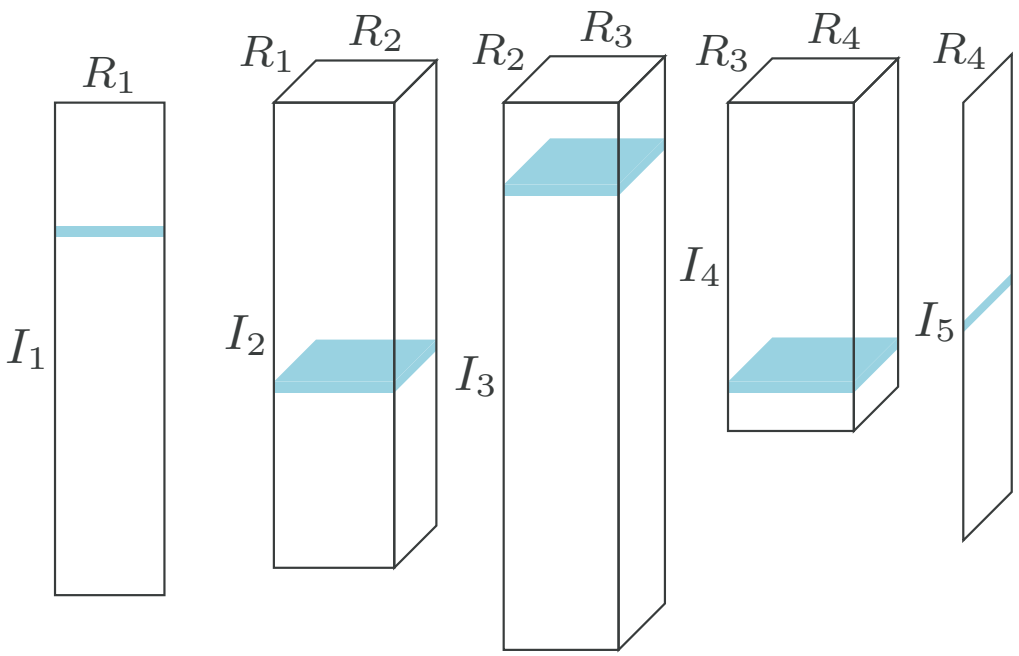
γ		β	α
59 %	Network	26%	15%
	DRAM	23%	5%

Table: Annual improvement of the parameters of the *latency-bandwidth model*. Network stands for the intraconnection between processors on distributed memory architectures, DRAM is the reading access memory.



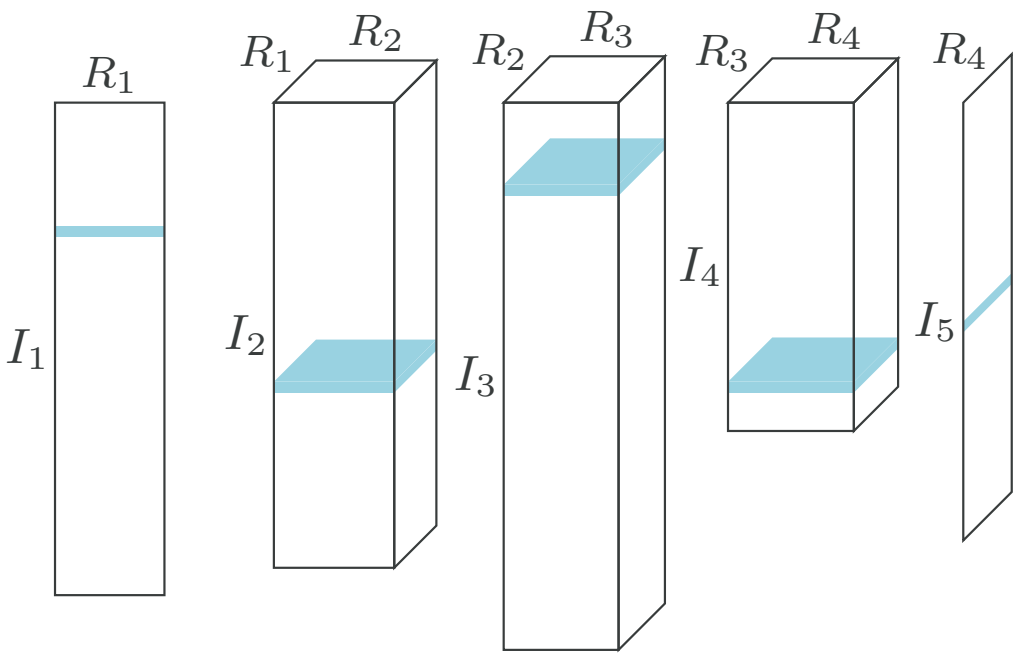
Outline

Tensor Train Notation
($N = 5$ for illustration)



$\mathcal{X} = \{\mathcal{T}_{\mathcal{X},k}\}_{k=1}^{5}$, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$, $\mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}$ are the TT cores

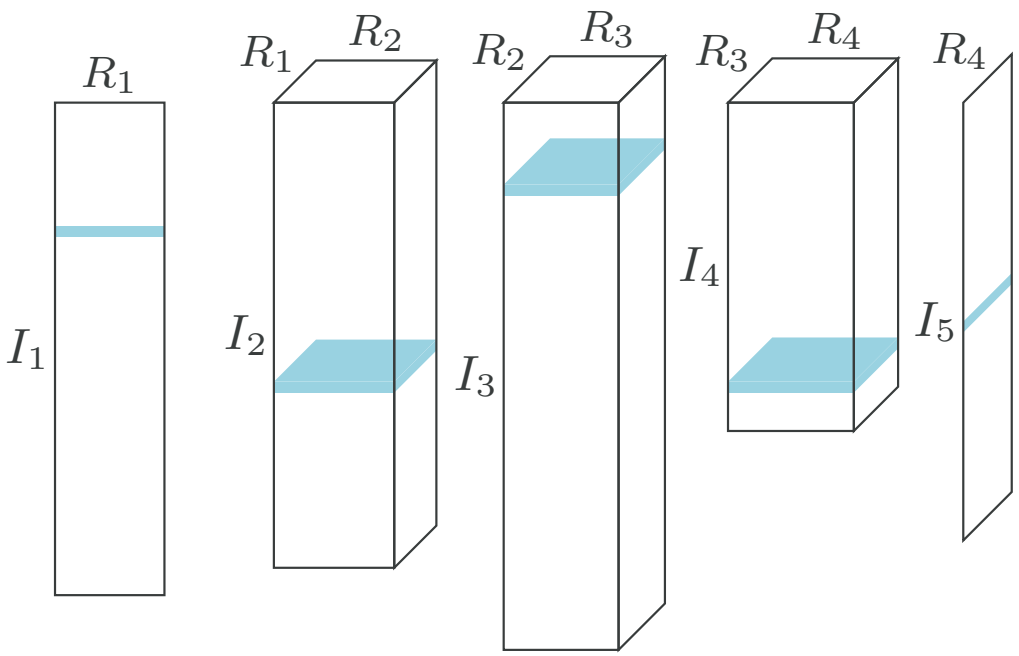
Tensor Train Notation
($N = 5$ for illustration)



$\mathcal{X} = \{\mathcal{T}_{\mathcal{X},k}\}_{k=1}^{5}$, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}$, $\mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}$ are the TT cores

$$x_{i_1,\ldots,i_5} = \sum_{\alpha=1}^{R_2} \sum_{\beta=1}^{R_3} \sum_{\gamma=1}^{R_4} \sum_{\delta=1}^{R_5} \mathcal{T}_{\mathcal{X},1}(i_1,\alpha)\mathcal{T}_{\mathcal{X},2}(\alpha,i_2,\beta)\mathcal{T}_{\mathcal{X},3}(\beta,i_3,\gamma)\mathcal{T}_{\mathcal{X},4}(\gamma,i_4,\delta)\mathcal{T}_{\mathcal{X},5}(\delta,i_5)$$

Tensor Train Notation
($N = 5$ for illustration)



$\mathcal{X} = \{\mathcal{T}_{\mathcal{X},k}\}_{k=1}^{5},\ \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4 \times I_5}, \qquad \mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}$ are the TT cores

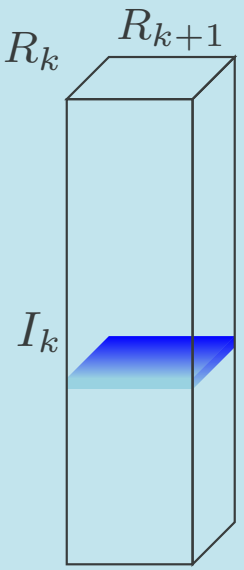
$$x_{i_1,\ldots,i_5} = \sum_{\alpha=1}^{R_2} \sum_{\beta=1}^{R_3} \sum_{\gamma=1}^{R_4} \sum_{\delta=1}^{R_5} \mathcal{T}_{\mathcal{X},1}(i_1,\alpha)\mathcal{T}_{\mathcal{X},2}(\alpha,i_2,\beta)\mathcal{T}_{\mathcal{X},3}(\beta,i_3,\gamma)\mathcal{T}_{\mathcal{X},4}(\gamma,i_4,\delta)\mathcal{T}_{\mathcal{X},5}(\delta,i_5)$$

$$x_{i_1,\ldots,i_5} = \mathbf{T}_{\mathcal{X},1}(i_1)\cdots\mathbf{T}_{\mathcal{X},5}(i_5)$$

Unfoldings (Matricizations)

TT cores

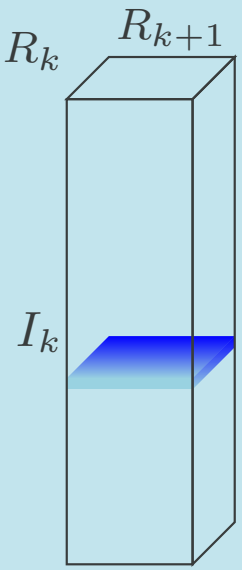
$$\mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}$$



Unfoldings (Matricizations)

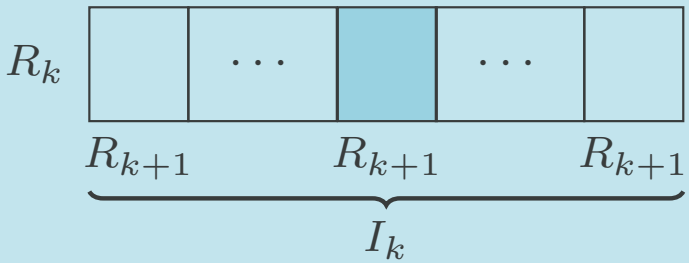
TT cores

$$\mathcal{T}_{\mathbf{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}$$



Horizontal Unfolding

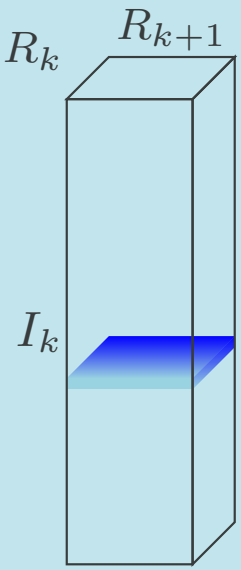
$$\mathcal{H}(\mathcal{T}_{\mathbf{X},k}) \in \mathbb{R}^{R_k \times I_k R_{k+1}}$$



Unfoldings (Matricizations)

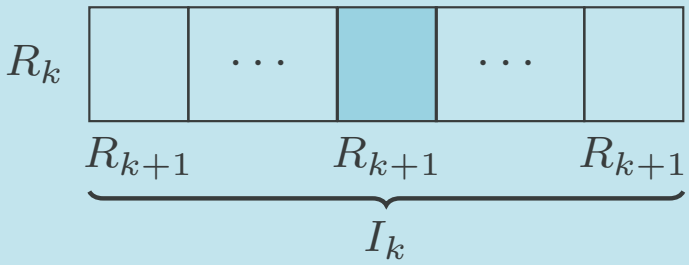
TT cores

$$\mathcal{T}_{\mathbf{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}$$



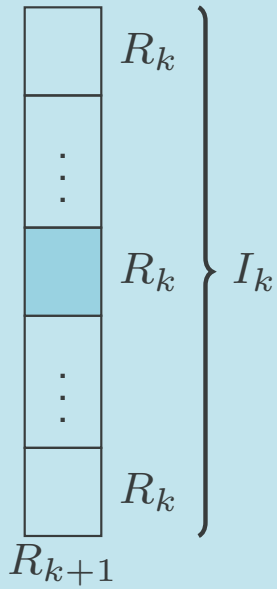
Horizontal Unfolding

$$\mathcal{H}(\mathcal{T}_{\mathbf{X},k}) \in \mathbb{R}^{R_k \times I_k R_{k+1}}$$



Vertical Unfolding

$$\mathcal{V}(\mathcal{T}_{\mathbf{X},k}) \in \mathbb{R}^{R_k I_k \times R_{k+1}}$$



Algebraic Operations

Scaling

$$\mathcal{X} = \{\mathcal{T}_{\mathcal{X},k}\}, \quad \mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}, \quad \mathcal{Z} = \lambda \mathcal{X}$$

$\mathcal{Z}$ in TT format:

Algebraic Operations

Scaling

$$\mathcal{X} = \{\mathcal{T}_{\mathcal{X},k}\}, \quad \mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}}, \quad \mathcal{Z} = \lambda \mathcal{X}$$

$\mathcal{Z}$ in TT format:

$$z_{i_1,\ldots,i_N} = (\lambda \mathbf{T}_{\mathcal{X},1}(i_1)) \cdots \mathbf{T}_{\mathcal{X},k}(i_k) \cdots \mathbf{T}_{\mathcal{X},N}(i_N),$$

then

$$\mathcal{T}_{\mathcal{Z},1} = \lambda \mathcal{T}_{\mathcal{X},1}, \quad \mathcal{T}_{\mathcal{Z},k} = \mathcal{T}_{\mathcal{X},k} \quad \text{for} \quad k = 2, \ldots, N.$$

Algebraic Operations

Summation

$$\mathcal{X} = \{\mathcal{T}_{\mathcal{X},k}\}, \quad \mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}},$$
$$\mathcal{Y} = \{\mathcal{T}_{\mathcal{Y},k}\}, \quad \mathcal{T}_{\mathcal{Y},k} \in \mathbb{R}^{L_k \times I_k \times L_{k+1}},$$
$$\mathcal{Z} = \mathcal{X} + \mathcal{Y}$$

$\mathcal{Z}$ in TT format:

Algebraic Operations

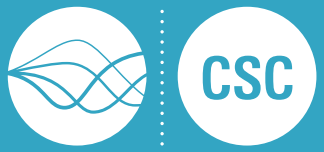
Summation

$$\mathcal{X} = \{\mathcal{T}_{\mathcal{X},k}\}, \quad \mathcal{T}_{\mathcal{X},k} \in \mathbb{R}^{R_k \times I_k \times R_{k+1}},$$
$$\mathcal{Y} = \{\mathcal{T}_{\mathcal{Y},k}\}, \quad \mathcal{T}_{\mathcal{Y},k} \in \mathbb{R}^{L_k \times I_k \times L_{k+1}},$$
$$\mathcal{Z} = \mathcal{X} + \mathcal{Y}$$

$\mathcal{Z}$ in TT format:

$$\begin{aligned} z_{i_1,\ldots,i_N} &= \mathbf{T}_{\mathcal{X},1}(i_1) \cdots \mathbf{T}_{\mathcal{X},k}(i_k) \cdots \mathbf{T}_{\mathcal{X},N}(i_N) \\ &\quad + \mathbf{T}_{\mathcal{Y},1}(i_1) \cdots \mathbf{T}_{\mathcal{Y},k}(i_k) \cdots \mathbf{T}_{\mathcal{Y},N}(i_N) \\ &= \begin{pmatrix} \mathbf{T}_{\mathcal{X},1}(i_1) & \mathbf{T}_{\mathcal{Y},1}(i_1) \end{pmatrix} \cdots \begin{pmatrix} \mathbf{T}_{\mathcal{X},k}(i_k) & \\ & \mathbf{T}_{\mathcal{Y},k}(i_k) \end{pmatrix} \cdots \begin{pmatrix} \mathbf{T}_{\mathcal{X},N}(i_N) \\ \mathbf{T}_{\mathcal{Y},N}(i_N) \end{pmatrix} \end{aligned}$$

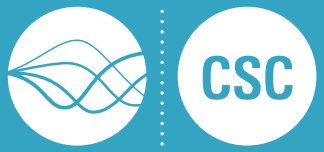
TT-ranks of $\mathcal{Z}$ are formally increased, but often can be compressed, e.g., when $\mathcal{X} = \mathcal{Y}$, TT-ranks of $\mathcal{Z}$ and $\mathcal{X}$ are equal!



TT Rounding

Compressing a tensor in TT format by reducing its TT ranks is relevant

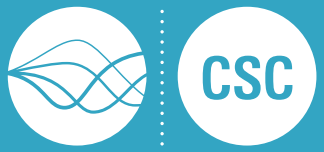
- for removing redundancy resulting from algebraic operations (e.g., summation);



TT Rounding

Compressing a tensor in TT format by reducing its TT ranks is relevant

- for removing redundancy resulting from algebraic operations (e.g., summation);
- for data compression w.r.t. given error threshold.



TT Rounding

Compressing a tensor in TT format by reducing its TT ranks is relevant

- for removing redundancy resulting from algebraic operations (e.g., summation);
- for data compression w.r.t. given error threshold.

Rounding done in two phases: orthogonalization and truncation:

- Orthogonalization is performed sequentially core-by-core.
- Truncation is performed in reverse direction core-by-core.

TT Rounding

Compressing a tensor in TT format by reducing its TT ranks is relevant

- for removing redundancy resulting from algebraic operations (e.g., summation);
- for data compression w.r.t. given error threshold.

Rounding done in two phases: orthogonalization and truncation:

- Orthogonalization is performed sequentially core-by-core.
- Truncation is performed in reverse direction core-by-core.

for $k = N$ down to 2 **do**
$L \cdot \mathcal{H}(\mathcal{Q}) = \mathcal{H}(\mathcal{T}_{\mathcal{X},k})$ ▷ LQ factorization of short-fat matrix
$\mathcal{T}_{\mathcal{X},k} = \mathcal{Q}$
$\mathcal{V}(\mathcal{T}_{\mathcal{X},k-1}) = \mathcal{V}(\mathcal{T}_{\mathcal{X},k-1})L$

for $k = 1$ to $N-1$ **do**
$\mathcal{V}(\mathcal{U}_k) \cdot \Sigma_k \cdot V_k^T = \mathcal{V}(\mathcal{T}_{\mathcal{X},k})$ ▷ truncated SVD of tall-skinny matrix
$\mathcal{T}_{\mathcal{X},k} = \mathcal{U}_k$
$\mathcal{H}(\mathcal{T}_{\mathcal{X},k+1}) = \Sigma_k V_k^\top \mathcal{H}(\mathcal{T}_{\mathcal{X},k+1})$

TT Rounding

Matrix Case, e.g., [Benner, 2004] or [Oseledets, 2011]

$$\mathbf{A} = \mathbf{X}\mathbf{Y}^T$$

TT Rounding

Matrix Case, e.g., [Benner, 2004] or [Oseledets, 2011]

$$\mathbf{A} = \mathbf{X}\mathbf{Y}^T$$

Right orthogonalization:

QR factorization $\mathbf{Y} = \mathbf{Q}\mathbf{R}$ $(\equiv$ LQ factorization of $Y^T)$

TT Rounding

Matrix Case, e.g., [Benner, 2004] or [Oseledets, 2011]

$$\mathbf{A} = \mathbf{X}\mathbf{Y}^T$$

Right orthogonalization:

$$\text{QR factorization} \quad \mathbf{Y} = \mathbf{Q}\mathbf{R} \qquad (\equiv \text{ LQ factorization of } Y^T)$$

$$\rightsquigarrow \quad \mathbf{A} = (\mathbf{X}\mathbf{R}^T)\mathbf{Q}^T$$

TT Rounding

Matrix Case, e.g., [Benner, 2004] or [Oseledets, 2011]

$$\mathbf{A} = \mathbf{X}\mathbf{Y}^T$$

Right orthogonalization:

$$\text{QR factorization} \quad \mathbf{Y} = \mathbf{Q}\mathbf{R} \qquad (\equiv \text{ LQ factorization of } Y^T)$$

$$\rightsquigarrow \quad \mathbf{A} = (\mathbf{X}\mathbf{R}^T)\mathbf{Q}^T$$

Left to right truncation:

$$\text{Truncated SVD} \quad \mathbf{X}\mathbf{R}^T = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T + \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T \approx \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T$$

TT Rounding

Matrix Case, e.g., [Benner, 2004] or [Oseledets, 2011]

$$\mathbf{A} = \mathbf{X}\mathbf{Y}^T$$

Right orthogonalization:

$$\text{QR factorization} \quad \mathbf{Y} = \mathbf{Q}\mathbf{R} \qquad (\equiv \text{ LQ factorization of } Y^T)$$

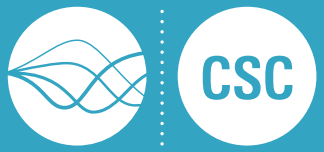
$$\rightsquigarrow \quad \mathbf{A} = (\mathbf{X}\mathbf{R}^T)\mathbf{Q}^T$$

Left to right truncation:

$$\text{Truncated SVD} \quad \mathbf{X}\mathbf{R}^T = \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T + \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T \approx \mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T$$

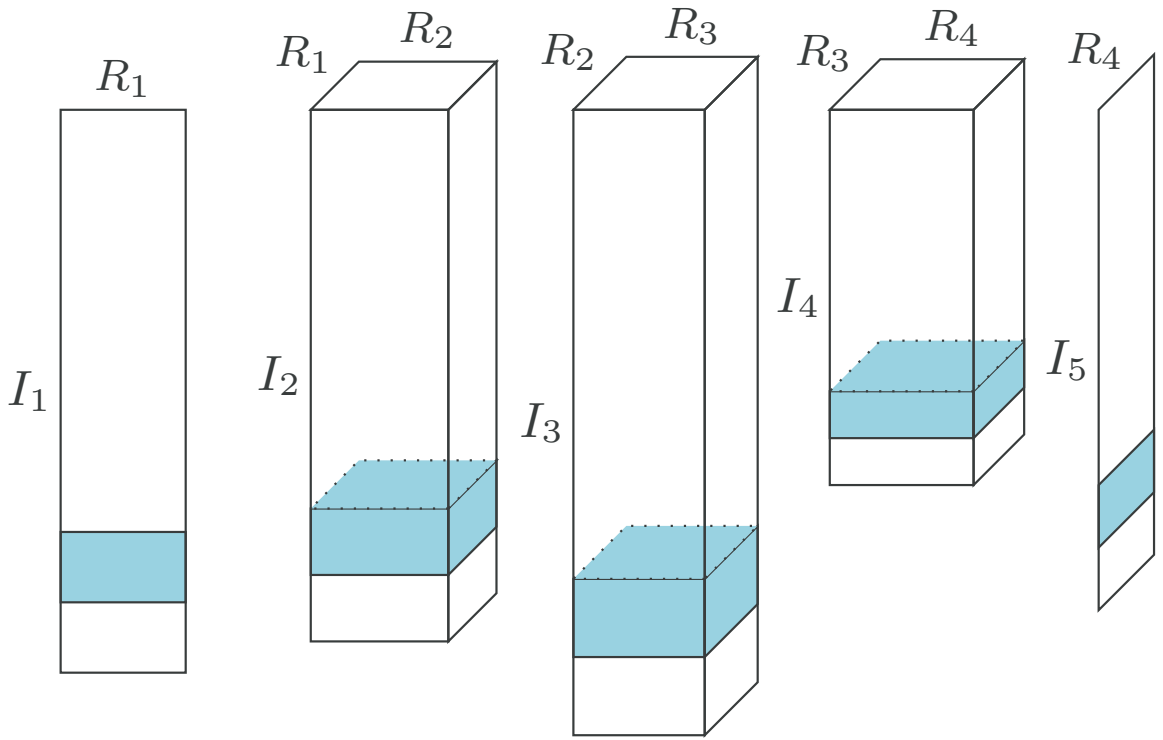
$$\rightsquigarrow \quad \mathbf{A} = \left(\mathbf{U}_1\mathbf{\Sigma}_1\mathbf{V}_1^T + \mathbf{U}_2\mathbf{\Sigma}_2\mathbf{V}_2^T\right)\mathbf{Q}^T,$$

$$\approx \mathbf{U}_1\left(\mathbf{Q}\mathbf{V}_1\mathbf{\Sigma}_1\right)^T$$

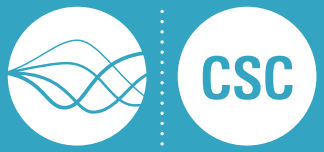


Outline

Data Distribution



- Each core is distributed across across all P processors by "block slices".
- Local kth core dimensions are $R_k \times \frac{I_k}{P} \times R_{k+1}$.
- Vertical and horizontal unfoldings are 1D-distributed, i.e., block row-wise and block column-wise, respectively.



Parallel TT Arithmetic [Al Daas et al., 2020]

Embarassingly parallel:

- Scaling
- Summation

Parallel TT Arithmetic [Al Daas et al., 2020]

Embarassingly parallel:

- Scaling
- Summation

Classic parallel:

- Linear operator applied to TT tensor $\equiv$ matrix-vector product:

$$(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)\,\mathcal{X}.$$

Parallel TT Arithmetic [Al Daas et al., 2020]

Embarassingly parallel:

- Scaling
- Summation

Classic parallel:

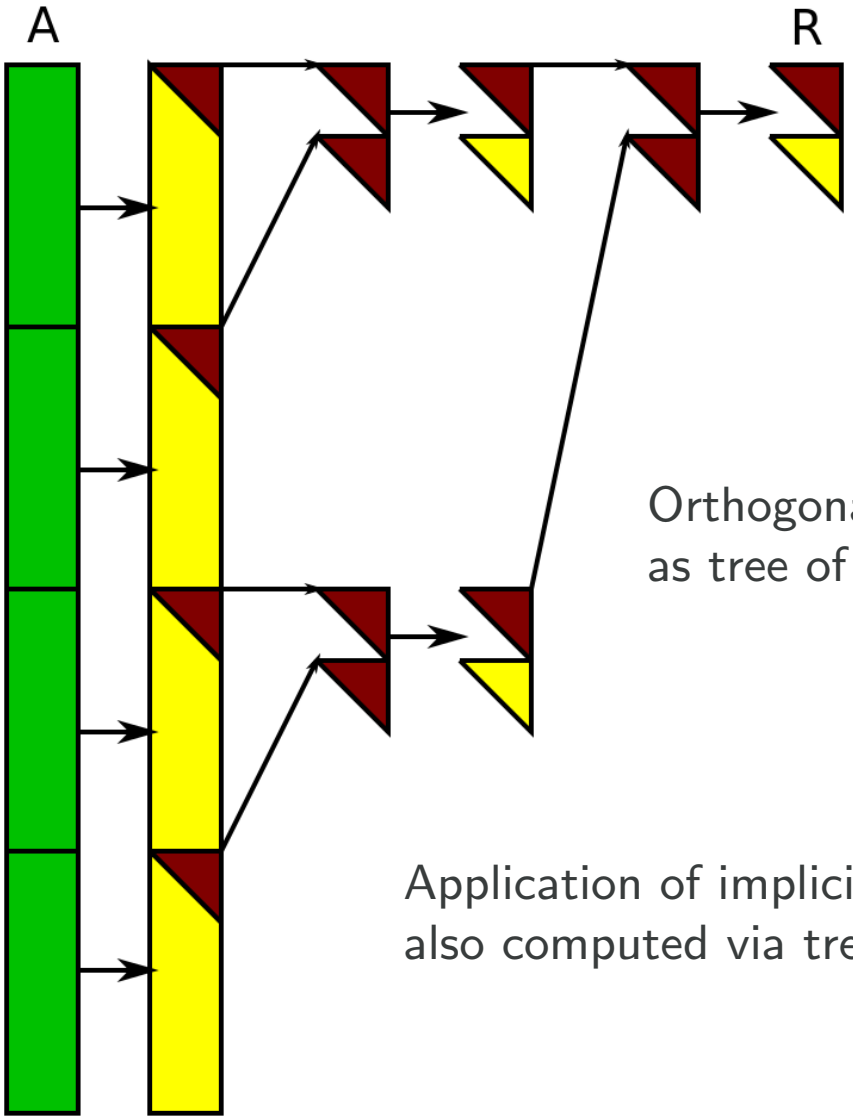
- Linear operator applied to TT tensor $\equiv$ matrix-vector product:

$$(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)\,\mathcal{X}.$$

Sequential in modes:

- (Frobenius) inner product $\langle \mathcal{X}, \mathcal{Y} \rangle$.
- TT Rounding!

Work Horse: Tall-Skinny QR (TSQR) Algorithm [Demmel et al., 2012]



Key benefit of TSQR:
one parallel reduction

Orthogonal factor stored implicitly
as tree of Householder vectors

Application of implicit orthogonal factor (TSQR-Apply-Q)
also computed via tree (backwards)

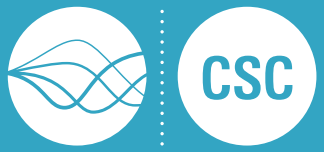
Parallel TT Rounding Algorithm

function $\{\mathcal{T}_{\mathcal{Y},k}^{(p)}\} = \text{PAR-TT-ROUNDING}(\{\mathcal{T}_{\mathcal{X},k}^{(p)}\})$

for $k = N$ down to 2 **do**
- $[\{Y_p^{(\ell)}\}_k, R_k] = \textbf{TSQR}(\mathcal{H}(\mathcal{T}_{\mathcal{X},k}^{(p)})^T)$ ▷ QR factorization
- $R^{(p)} = \textbf{Broadcast}(R_k, \text{root})$ ▷ Broadcast R to all procs
- $\mathcal{V}(\mathcal{T}_{\mathcal{X},k-1}^{(p)}) = \textbf{Mult}(\mathcal{V}(\mathcal{T}_{\mathcal{X},k-1}^{(p)}), R^{(p)})$ ▷ Apply R to previous core

$\mathcal{Y} = \mathcal{X}$

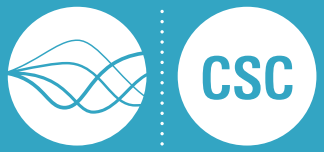
for $k = 1$ to $N - 1$ **do**
- $[\{Y_p^{(\ell)}\}_k, R_k] = \textbf{TSQR}(\mathcal{V}(\mathcal{T}_{\mathcal{Y},k}^{(p)}))$ ▷ QR factorization
- **if** $p = \text{root}$ **then**
  - $[\hat{U}_R, \hat{\Sigma}, \hat{V}] = \textbf{tSVD}(R_k)$ ▷ Truncated SVD of R_k
- $\mathcal{V}(\mathcal{T}_{\mathcal{Y},k}^{(p)}) = \textbf{TSQR-Apply-Q}(\{Y_p^{(\ell)}\}_k, \hat{U}_R)$ ▷ Form explicit $\hat{U}$
- $\mathcal{H}(\mathcal{T}_{\mathcal{X},k+1}^{(p)})^T = \textbf{TSQR-Apply-Q}(\{Y_p^{(\ell)}\}_{k+1}, \hat{V})$ ▷ Apply $\hat{V}$ to next core



Cost Analysis of TT-Rounding

Assumptions:

- $I_k \equiv I$ and $R_k \equiv R$;
- R is reduced by factor of 2.



Cost Analysis of TT-Rounding

Assumptions:

- $I_k \equiv I$ and $R_k \equiv R$;
- R is reduced by factor of 2.

Results:

- Computational cost is

$$7\frac{NIR^3}{P} + O(NR^3 \log P) \text{ flops } .$$

- Communication cost is

$$O(NR^2 \log P) \text{ words} \quad \text{and} \quad O(N \log P) \text{ messages.}$$

Cost Analysis of TT-Rounding

Assumptions:

- $I_k \equiv I$ and $R_k \equiv R$;
- R is reduced by factor of 2.

Results:

- Computational cost is

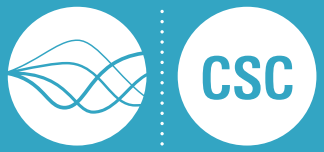
$$7\frac{NIR^3}{P} + O(NR^3 \log P) \text{ flops } .$$

- Communication cost is

$$O(NR^2 \log P) \text{ words} \quad \text{and} \quad O(N \log P) \text{ messages.}$$

Interpretation:

- Computational cost is linear (and not exponential) in N (property of TT).
- Communication cost is independent of I (✓).
- Communication cost increases slightly with P and N (−).



Outline

MPI-ATTAC

- Programming language: C
- Dependencies: MKL, (MPI for parallel computation)
- Shared-memory test runs performed on COBRA (MPCDF), MKL version 2019 and MATLAB 2019
- Parallel test runs performed on COBRA (MPCDF), MKL version 2019, Intel-MPI 2019

MPG Supercomputer Cobra

Based on Intel Xeon Skylake-SP processors and Nvidia GPUs (V100, RTX5000):

- 3424 compute nodes, 136,960 CPU-cores
- 128 Tesla V100-32 GPUs, 240 Quadro RTX 5000 GPUs
- 529 TB CPU RAM (DDR4), 7.9 TB GPU RAM HBM2
- 11.4 PFlop/s peak (FP64)

Numerical Experiments
Test Examples

Model	# Modes	Dimensions	Ranks	Memory
1	50	$2K \times \cdots \times 2K$	50	2 GB
2	16	$100M \times 50K \times \cdots \times 50K \times 1M$	30	28 GB
3	30	$2M \times \cdots \times 2M$	30	385 GB

Table: Synthetic TT models used for performance experiments. In each case the formal ranks are all the same and are cut in half by the TT rounding procedure.

All data are synthetic and mimic typical situations in

- compressing the TT tensor resulting from a cross approximation of High-Order Correlation Functions (Model 1);
- Parameter-dependent PDEs, arising, e.g., in UQ (Model 2): large dimensions of "spatial" modes, small dimension in "parameter" modes;
- Molecular Simulations (Model 3).

Numerical Experiments

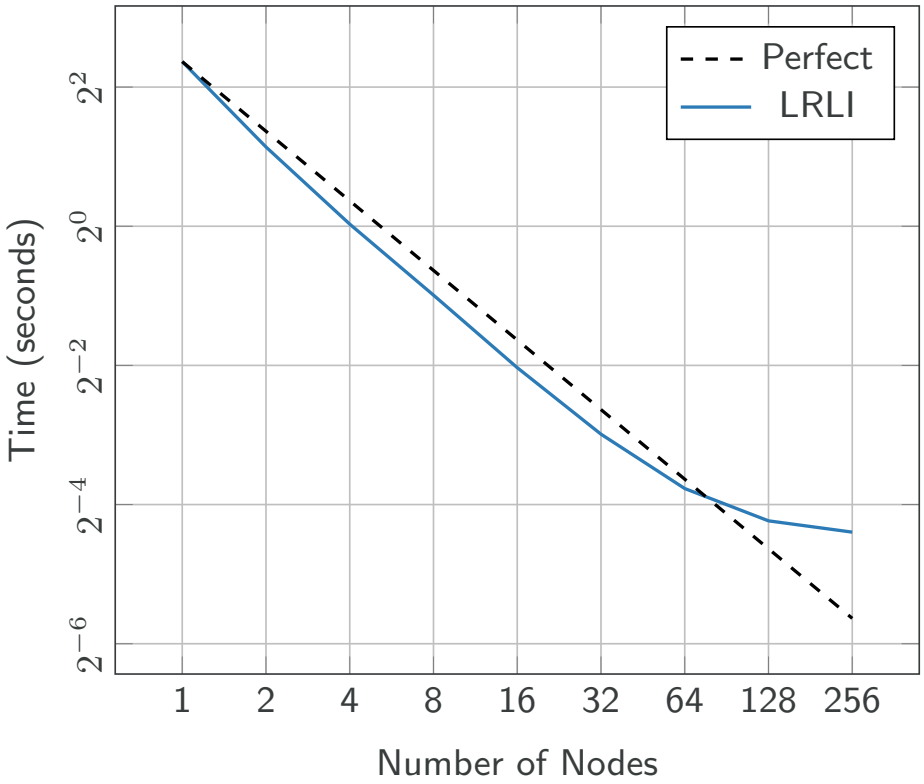
Comparison: TT-Toolbox (MATLAB 2019) vs. ATTAC (sequential)

.

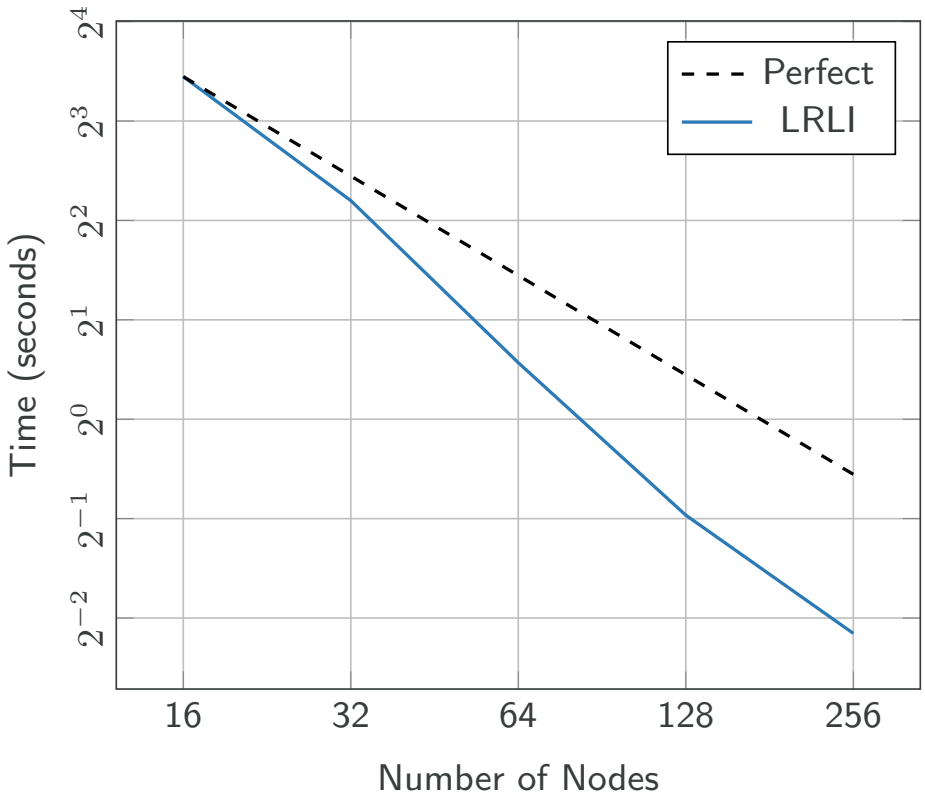
	1 core	20 cores	Par. Speedup	40 cores	Par. Speedup
TT-Toolbox	15.68	8.34	$\mathbf{1.9}\times$	8.752	$\mathbf{1.8}\times$
MPI-ATTAC	9.2	0.44	$\mathbf{20.9}\times$	0.27	$\mathbf{33.9}\times$
Speedup	$\mathbf{1.7}\times$	$\mathbf{18.95}\times$		$\mathbf{32.2}\times$	

Table: Single-node performance results on TT Model 1 and comparison with the MATLAB TT-Toolbox.

Numerical Experiments
Parallel (Strong) Scaling Experiments



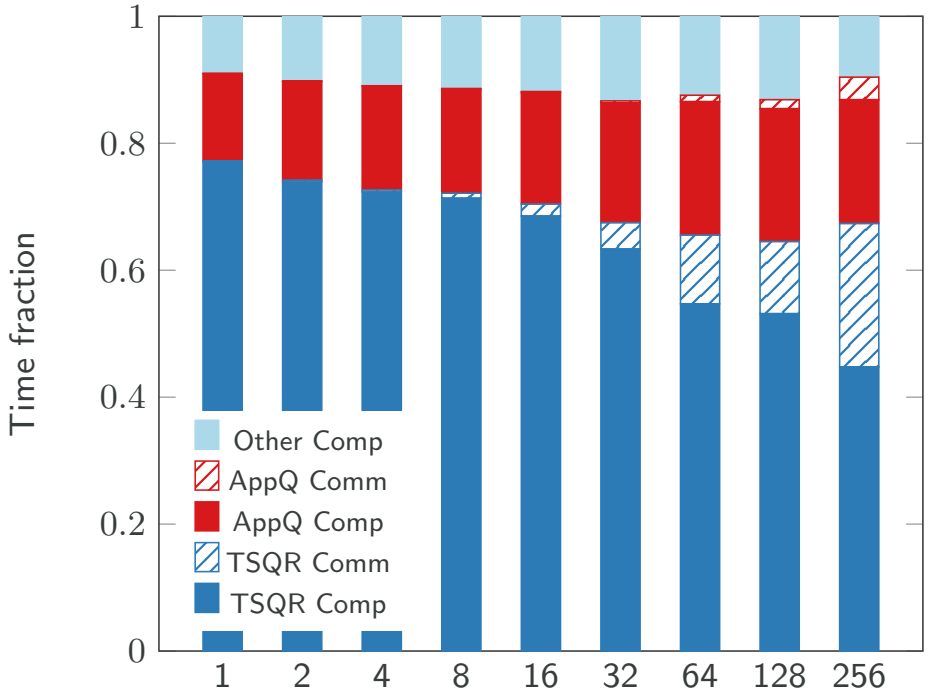
(a) Parallel scaling for Model 2



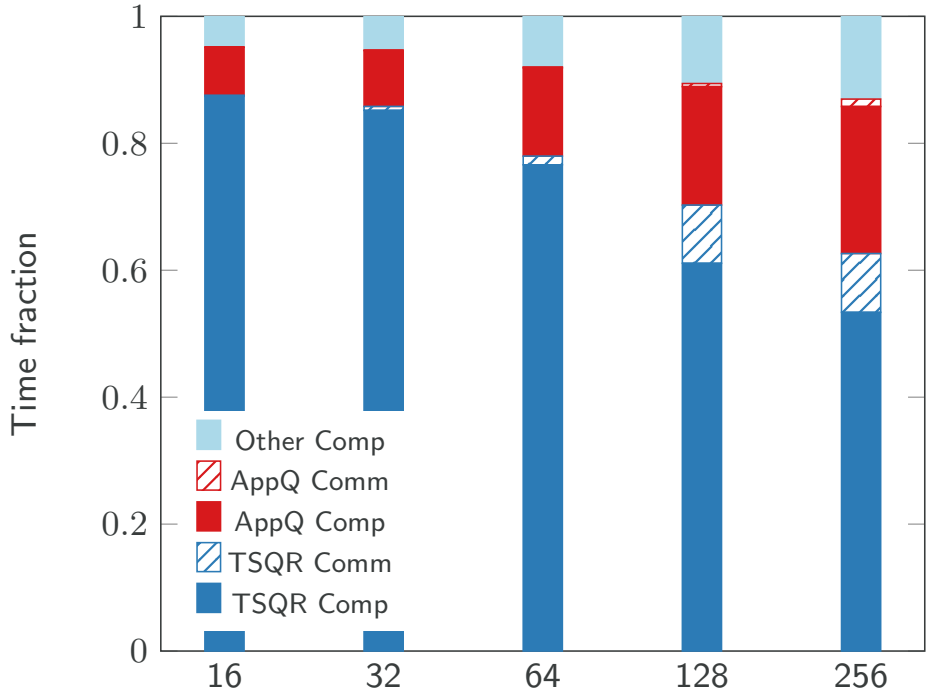
(b) Parallel scaling for Model 3

Figure: Time breakdown and and parallel scaling of LRLI variant of TT rounding.

Performance Breakdown
Computation vs. Communication

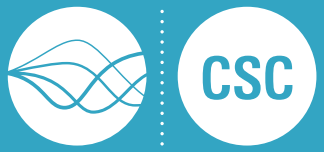


(a) Time breakdown for Model 2



(b) Time breakdown for Model 3

- As expected, communication cost increases with number of processors.
- Recall: $O(NR^2 \log P)$ words and $O(N \log P)$ messages.



Outline

Conclusions and Outlook

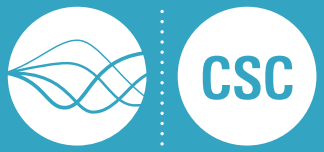
So far:

- Developed distributed-memory parallel algorithms for basic TT tensor operations (scaling, addition, Hadamard and inner products, norm, rounding/truncation).
- Algorithms for Tensor Train Arithmetic and Computation.
- Scalable MPI implementation (BSD-2Clauses).

Conclusions and Outlook

So far:

- Developed distributed-memory parallel algorithms for basic TT tensor operations (scaling, addition, Hadamard and inner products, norm, rounding/truncation).
- Algorithms for Tensor Train Arithmetic and Computation.
- Scalable MPI implementation (BSD-2Clauses).



Conclusions and Outlook

So far:

- Developed distributed-memory parallel algorithms for basic TT tensor operations (scaling, addition, Hadamard and inner products, norm, rounding/truncation).
- Algorithms for Tensor Train Arithmetic and Computation.
- Scalable MPI implementation (BSD-2Clauses).

Next steps:

- Publish `gitlab` repo!
- Linear solvers (TT-GMRES, AMEn).
- Randomized rounding.

References

Al Daas, H., Ballard, G., and Benner, P. (2020).
Parallel algorithms for tensor train arithmetic.
arXiv, 2011.06532.

Benner, P. (2004).
Factorized solution of Sylvester equations with applications in control.
In *Proc. Intl. Symp. Math. Theory Networks and Syst. MTNS 2004.*

Benner, P., Dolgov, S., Onwunta, A., and Stoll, M. (2016).
Low-rank solvers for unsteady Stokes-Brinkman optimal control problem with random data.
Comp. Meth. Appl. Mech. Eng., 304:26–54.

Benner, P., Dolgov, S., Onwunta, A., and Stoll, M. (2020).
Low-rank solution of an optimal control problem constrained by random Navier-Stokes equations.
Internat. J. Numer. Methods Fluids, 92(11):1653–1678.

Benner, P., Onwunta, A., and Stoll, M. (2015).
Low-rank solution of unsteady diffusion equations with stochastic coefficients.
SIAM/ASA J. Uncertain. Quantif., 3(1):622–649.

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

Oseledets, I. (2011).
Tensor-train decomposition.
SIAM J. Sci. Comput., 33(5):2295–2317.