MAX PLANCK INSTITUTE FOR DYNAMICS OF COMPLEX TECHNICAL SYSTEMS MAGDEBURG

CSC COMPUTATIONAL METHODS IN SYSTEMS AND CONTROL THEORY

Low-rank Tensor Methods for Optimal Control of Uncertain Flow Problems

Peter Benner

Joint work with
Sergey Dolgov (U Bath)
Martin Stoll (TU Chemnitz)
Akwum Onwunta (U Maryland)

ICIAM 2019
MS PDE-constrained optimization under uncertainty
July 18, 2019

Supported by:

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

CDS Forschungszentrum Dynamische Systeme

Outline

Introduction

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.

Introduction

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, . . .

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, ...
- **Gedankenexperiment:**

Introduction

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, ...
- **Gedankenexperiment:**
  - we use a grid with 10^6 nodes, for each node we store 4 real numbers: the velocity in x-, y-, and z-direction, plus the pressure;

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, ...
- **Gedankenexperiment:**
  - we use a grid with 10^6 nodes, for each node we store 4 real numbers: the velocity in x-, y-, and z-direction, plus the pressure;
  - we need $1,000$ time steps to reach steady-state;

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, ...
- **Gedankenexperiment:**
  - we use a grid with 10^6 nodes, for each node we store 4 real numbers: the velocity in x-, y-, and z-direction, plus the pressure;
  - we need $1,000$ time steps to reach steady-state;
  - we have 10 different design parameters, and want to test 10 values for each of them $\rightsquigarrow$ 10^{10} configurations, i.e., 10^{10} transient simulations.

Introduction

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, ...
- **Gedankenexperiment:**
  - we use a grid with 10^6 nodes, for each node we store 4 real numbers: the velocity in x-, y-, and z-direction, plus the pressure;
  - we need $1,000$ time steps to reach steady-state;
  - we have 10 different design parameters, and want to test 10 values for each of them $\rightsquigarrow 10^{10}$ configurations, i.e., 10^{10} transient simulations.
- Assuming you have a HPC cluster allowing you to do so in reasonable time, just storing all trajectories (using floats) requires

$$10^{10} \cdot 10^3 \cdot 10^6 \cdot 4 \cdot 4 = 1.6 \cdot 10^{20} \text{bytes} \approx 160 \text{ exabytes of memory!}$$

  Data can be compressed, but first it needs to be generated and stored...

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, ...
- **Gedankenexperiment:**
  - we use a grid with 10^6 nodes, for each node we store 4 real numbers: the velocity in x-, y-, and z-direction, plus the pressure;
  - we need $1,000$ time steps to reach steady-state;
  - we have 10 different design parameters, and want to test 10 values for each of them $\rightsquigarrow 10^{10}$ configurations, i.e., 10^{10} transient simulations.
- Assuming you have a HPC cluster allowing you to do so in reasonable time, just storing all trajectories (using floats) requires

  $$10^{10} \cdot 10^3 \cdot 10^6 \cdot 4 \cdot 4 = 1.6 \cdot 10^{20} \text{bytes} \approx 160 \text{ exabytes of memory!}$$

  Data can be compressed, but first it needs to be generated and stored...
- Why not starting with a compressed data format from the very beginning, and computing all trajectories all-at-once?

Introduction

A (big) data problem in flow simulation

- Assume you want to design an airfoil, wing, car, etc.
- This requires 3D flow simulations for varying configurations, and you want to save the data for visualization, control design, optimization, comparisons, ...
- **Gedankenexperiment:**
  - we use a grid with 10^6 nodes, for each node we store 4 real numbers: the velocity in x-, y-, and z-direction, plus the pressure;
  - we need $1,000$ time steps to reach steady-state;
  - we have 10 different design parameters, and want to test 10 values for each of them $\rightsquigarrow 10^{10}$ configurations, i.e., 10^{10} transient simulations.
- Assuming you have a HPC cluster allowing you to do so in reasonable time, just storing all trajectories (using floats) requires

  $$10^{10} \cdot 10^3 \cdot 10^6 \cdot 4 \cdot 4 = 1.6 \cdot 10^{20} \text{bytes} \approx 160 \text{ exabytes of memory!}$$

  Data can be compressed, but first it needs to be generated and stored...
- Why not starting with a compressed data format from the very beginning, and computing all trajectories all-at-once?
- This is the idea we pursue in this talk, where we store the data in a compressed (low-rank) $1 + 10 + 1(4)$–way tensor! (Above example $\rightsquigarrow$ terabyte-range.)

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.
- Models of these processes should account for these uncertainties.

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.
- Models of these processes should account for these uncertainties.
- PDEs governing the processes can involve uncertain coefficients, or uncertain sources, or uncertain geometry.

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.
- Models of these processes should account for these uncertainties.
- PDEs governing the processes can involve uncertain coefficients, or uncertain sources, or uncertain geometry.
- Uncertain parameters modeled as random variables $\rightsquigarrow$ random PDEs, potentially also containing uncertain inputs (controls) $\rightsquigarrow$ (generalized) polynomial chaos approach $\rightsquigarrow$ high-dimensional PDE!

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.
- Models of these processes should account for these uncertainties.
- PDEs governing the processes can involve uncertain coefficients, or uncertain sources, or uncertain geometry.
- Uncertain parameters modeled as random variables $\rightsquigarrow$ random PDEs, potentially also containing uncertain inputs (controls) $\rightsquigarrow$ (generalized) polynomial chaos approach $\rightsquigarrow$ high-dimensional PDE!
- Here: no stochastic PDEs in the sense of dynamics driven by Wiener or Lévy or ... processes!

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.
- Models of these processes should account for these uncertainties.
- PDEs governing the processes can involve uncertain coefficients, or uncertain sources, or uncertain geometry.
- Uncertain parameters modeled as random variables $\rightsquigarrow$ random PDEs, potentially also containing uncertain inputs (controls) $\rightsquigarrow$ (generalized) polynomial chaos approach $\rightsquigarrow$ high-dimensional PDE!
- Here: no stochastic PDEs in the sense of dynamics driven by Wiener or Lévy or ... processes!

Uncertainty arises because

- available data are incomplete;

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.
- Models of these processes should account for these uncertainties.
- PDEs governing the processes can involve uncertain coefficients, or uncertain sources, or uncertain geometry.
- Uncertain parameters modeled as random variables $\rightsquigarrow$ random PDEs, potentially also containing uncertain inputs (controls) $\rightsquigarrow$ (generalized) polynomial chaos approach $\rightsquigarrow$ high-dimensional PDE!
- Here: no stochastic PDEs in the sense of dynamics driven by Wiener or Lévy or . . . processes!

Uncertainty arises because

- available data are incomplete;
- data are predictable, but difficult to measure, e.g., porosity above oil reservoirs ("aleatoric uncertainty");

Introduction

Here: parameters result from stochastic modeling of coefficients

- Physical, biological, chemical, etc. processes involve uncertainties.
- Models of these processes should account for these uncertainties.
- PDEs governing the processes can involve uncertain coefficients, or uncertain sources, or uncertain geometry.
- Uncertain parameters modeled as random variables $\rightsquigarrow$ random PDEs, potentially also containing uncertain inputs (controls) $\rightsquigarrow$ (generalized) polynomial chaos approach $\rightsquigarrow$ high-dimensional PDE!
- Here: no stochastic PDEs in the sense of dynamics driven by Wiener or Lévy or ... processes!

Uncertainty arises because

- available data are incomplete;
- data are predictable, but difficult to measure, e.g., porosity above oil reservoirs ("aleatoric uncertainty");
- data are unpredictable, e.g, wind shear ("epistemic uncertainty").

Curse of Dimensionality [Bellman '57]

Increase in 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** ($d > 3$).

Curse of Dimensionality

[Bellman '57]

Increase in 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** $(d > 3)$.

Consider $-\Delta u = f$ in $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, uniformly discretized as

$$(I \otimes A + A \otimes I)\, x =: \mathcal{A}x = b \qquad \Longleftrightarrow \qquad AX + XA^T = B$$

with $x = \mathrm{vec}(X)$ and $b = \mathrm{vec}(B)$ with low-rank right hand side $B \approx b_1 b_2^T$.

Curse of Dimensionality [Bellman '57]

Increase in 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** ($d > 3$).

Consider $-\Delta u = f$ in $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, uniformly discretized as

$$(I \otimes A + A \otimes I)\, x =: \mathcal{A}x = b \qquad \Longleftrightarrow \qquad AX + XA^T = B$$

with $x = \mathrm{vec}\,(X)$ and $b = \mathrm{vec}\,(B)$ with low-rank right hand side $B \approx b_1 b_2^T$.

- Low-rankness of $\tilde{X} := VW^T \approx X$ follows from properties of A and B, and in particular (approximate) separability $u(x, y) \approx v(x)w(y)$, $f(x, y) \approx g(x)h(y)$; e.g., [Penzl '00, Grasedyck '04].

Curse of Dimensionality [Bellman '57]

Increase in 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** ($d > 3$).

Consider $-\Delta u = f$ in $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, uniformly discretized as

$$(I \otimes A + A \otimes I)\, x =: \mathcal{A}x = b \qquad \Longleftrightarrow \qquad AX + XA^T = B$$

with $x = \mathrm{vec}\,(X)$ and $b = \mathrm{vec}\,(B)$ with low-rank right hand side $B \approx b_1 b_2^T$.

- Low-rankness of $\tilde{X} := VW^T \approx X$ follows from properties of A and B, and in particular (approximate) separability $u(x, y) \approx v(x)w(y)$, $f(x, y) \approx g(x)h(y)$; e.g., [Penzl '00, Grasedyck '04].
- We solve this using low-rank Krylov subspace solvers. These essentially require matrix-vector multiplication and vector computations.

Curse of Dimensionality [Bellman '57]

Increase in 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** ($d > 3$).

Consider $-\Delta u = f$ in $[0,1] \times [0,1] \subset \mathbb{R}^2$, uniformly discretized as

$$(I \otimes A + A \otimes I)\, x =: \mathcal{A}x = b \qquad \Longleftrightarrow \qquad AX + XA^T = B$$

with $x = \mathrm{vec}(X)$ and $b = \mathrm{vec}(B)$ with low-rank right hand side $B \approx b_1 b_2^T$.

- Low-rankness of $\tilde{X} := VW^T \approx X$ follows from properties of A and B, and in particular (approximate) separability $u(x,y) \approx v(x)w(y)$, $f(x,y) \approx g(x)h(y)$; e.g., [Penzl '00, Grasedyck '04].
- We solve this using low-rank Krylov subspace solvers. These essentially require matrix-vector multiplication and vector computations.
- Hence, $\mathcal{A}\mathrm{vec}(X_k) = \mathcal{A}\mathrm{vec}(V_k W_k^T) = \mathrm{vec}\left([\,AV_k,\ V_k\,][\,W_k,\ AW_k\,]^T\right)$

Curse of Dimensionality [Bellman '57]

Increase in 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** ($d > 3$).

Consider $-\Delta u = f$ in $[0, 1] \times [0, 1] \subset \mathbb{R}^2$, uniformly discretized as

$$(I \otimes A + A \otimes I)\, x =: \mathcal{A}x = b \qquad \Longleftrightarrow \qquad AX + XA^T = B$$

with $x = \operatorname{vec}(X)$ and $b = \operatorname{vec}(B)$ with low-rank right hand side $B \approx b_1 b_2^T$.

- Low-rankness of $\tilde{X} := VW^T \approx X$ follows from properties of A and B, and in particular (approximate) separability $u(x, y) \approx v(x)w(y)$, $f(x, y) \approx g(x)h(y)$; e.g., [Penzl '00, Grasedyck '04].
- We solve this using low-rank Krylov subspace solvers. These essentially require matrix-vector multiplication and vector computations.
- Hence, $\mathcal{A}\operatorname{vec}(X_k) = \mathcal{A}\operatorname{vec}\left(V_k W_k^T\right) = \operatorname{vec}\left(\left[\, AV_k,\ V_k \,\right]\left[\, W_k,\ AW_k \,\right]^T\right)$
- The rank of $[AV_k \quad V_k] \in \mathbb{R}^{n,2r}$, $[W_k \quad AW_k] \in \mathbb{R}^{n_t,2r}$ increases but can be controlled using truncation. $\rightsquigarrow$ Low-rank Krylov subspace solvers. [Kressner/Tobler, B/Breiten, Savostyanov/Dolgov, ...].

We 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 we assume that

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

This Talk

Curse of Dimensionality [Bellman '57]

Increase 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** ($d > 3$).

Goal of this talk

Apply low-rank iterative solvers to discrete optimality systems resulting from

PDE-constrained optimization problems under uncertainty,

and go one step further applying low-rank tensor (instead of matrix) techniques.

This Talk

Curse of Dimensionality [Bellman '57]

Increase 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** ($d > 3$).

Goal of this talk

Apply low-rank iterative solvers to discrete optimality systems resulting from

PDE-constrained optimization problems under uncertainty,

and go one step further applying low-rank tensor (instead of matrix) techniques.

Take home message

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

This Talk

Curse of Dimensionality [Bellman '57]

Increase 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** ($d > 3$).

Goal of this talk

Apply low-rank iterative solvers to discrete optimality systems resulting from

PDE-constrained optimization problems under uncertainty,

and go one step further applying low-rank tensor (instead of matrix) techniques.

Take home message

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

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

This Talk

Curse of Dimensionality [Bellman '57]

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

$\rightsquigarrow$ **Rapid Increase of Dimensionality**, called **Curse of Dimensionality** ($d > 3$).

Goal of this talk

Apply low-rank iterative solvers to discrete optimality systems resulting from

PDE-constrained optimization problems under uncertainty,

and go one step further applying low-rank tensor (instead of matrix) techniques.

Take home message

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

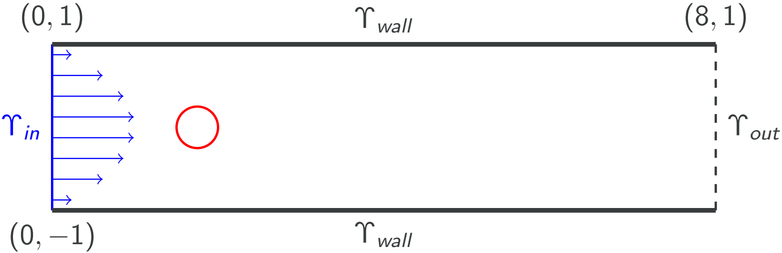
Would require ≈ 10 **petabytes (PB)** $= 10,000$ **TB** to store the solution vector!

Using low-rank tensor techniques, we need $\approx 7 \cdot 10^7$ **bytes** $= 70$ **GB** to solve the KKT system in MATLAB in less than one hour!

Outline

Model Problem 1: uncertain flow past a circular obstacle

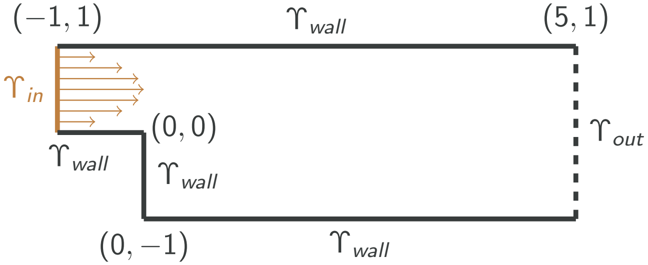
to Numerical Experiments



- We model this as a boundary control problem.
- Our constraint $c(y, u) = 0$ is given by the unsteady incompressible Navier-Stokes equations with uncertain viscosity $\nu := \nu(\omega)$/ inflow condition $\theta_2(t, x, \omega) = 0$ and

$$\theta_1(t, x, \omega) = \left((1 + x_2)(1 - x_2) + \sum_{k=1}^{m} k^{-\gamma-1/2} \cdot \sin(\pi k x_2) \cdot \xi_k(\omega) \right) (1 - \mathrm{e}^{-t}).$$

Model Problem 2: uncertain flow in a backward facing step domain



- We model this as a boundary control problem.
- Our constraint $c(y,u)=0$ is given by the unsteady incompressible Navier-Stokes equations with uncertain inflow condition $\theta_2(t,x,\omega)=0$ and

$$\theta_1(t,x,\omega) = \left(4x_2(1-x_2) + \frac{1}{2}\sum_{k=1}^{m} k^{-\gamma-1/2}\sin(2\pi k x_2)\xi_k(\omega)\right)(1-\mathrm{e}^{-t}).$$

Model Problem

Minimize:

$$\mathcal{J}(v,u) = \frac{1}{2}\|\mathrm{curl} v\|^2_{L^2(0,T;\mathcal{D})\otimes L^2(\Omega)} + \frac{\beta}{2}\|u\|^2_{L^2(0,T;\mathcal{D})\otimes L^2(\Omega)} \tag{1}$$

subject to

$$\begin{aligned}
\frac{\partial v}{\partial t} - \nu\Delta v + (v\cdot\nabla)v + \nabla p = 0, &\quad \text{in} \quad \mathcal{D},\\
-\nabla\cdot v = 0, &\quad \text{in} \quad \mathcal{D},\\
v = \theta, &\quad \text{on} \quad \Upsilon_{in},\\
v = 0, &\quad \text{on} \quad \Upsilon_{wall},\\
\frac{\partial v}{\partial n} = u, &\quad \text{on} \quad \Upsilon_{c},\\
\frac{\partial v}{\partial n} = 0, &\quad \text{on} \quad \Upsilon_{out},\\
v(\cdot,0,\cdot) = v_0, &\quad \text{in} \quad \mathcal{D}.
\end{aligned} \tag{2}$$

Model Problem Set-up (cf. [Powell/Silvester '12])

We assume

- $\nu(\omega) = \nu_0 + \nu_1\xi(\omega), \ \ \nu_0, \nu_1 \in \mathbb{R}^+, \ \ \xi \sim \mathcal{U}(-1, 1).$
- $\mathbb{P}\,(\omega \in \Omega : \nu(\omega) \in [\nu_{\min}, \nu_{\max}]) = 1$, for some $0 < \nu_{\min} < \nu_{\max} < +\infty$.
- $\Rightarrow$ velocity v, control u and pressure p are random fields on $L^2(\Omega)$.
- $L^2(\Omega) := L^2(\Omega, \mathcal{F}, \mathbb{P})$ is a complete probability space.
- $L^2(0, T; \mathcal{D}) := L^2(\mathcal{D}) \times L^2(\mathcal{T})$.

Model Problem Set-up (cf. [Powell/Silvester '12])

We assume

- $\nu(\omega) = \nu_0 + \nu_1 \xi(\omega), \;\; \nu_0, \nu_1 \in \mathbb{R}^+, \;\; \xi \sim \mathcal{U}(-1, 1).$
- $\mathbb{P}\left(\omega \in \Omega : \nu(\omega) \in [\nu_{\min}, \nu_{\max}]\right) = 1$, for some $0 < \nu_{\min} < \nu_{\max} < +\infty$.
- $\Rightarrow$ velocity v, control u and pressure p are random fields on $L^2(\Omega)$.
- $L^2(\Omega) := L^2(\Omega, \mathcal{F}, \mathbb{P})$ is a complete probability space.
- $L^2(0, T; \mathcal{D}) := L^2(\mathcal{D}) \times L^2(\mathcal{T})$.

Computational challenges

- Nonlinearity (due to the nonlinear convection term $(v \cdot \nabla)v$) .
- Uncertainty (due to random $\nu(\omega)$ and $\sigma(t, x, \omega)$).
- High dimensionality (of the resulting linear/optimality systems).

OTD Strategy and Picard (Oseen) Iteration $\rightsquigarrow$

state equation

$$v_t - \nu \Delta v + (\bar{v} \cdot \nabla) v + \triangledown p = 0$$
$$\nabla \cdot v = 0 + \text{ boundary conditions}$$

adjoint equation

$$-\chi_t - \Delta\chi - (\bar{v} \cdot \nabla)\chi + (\nabla \bar{v})^T \chi + \triangledown \mu = -\mathrm{curl}^2 v$$
$$\nabla \cdot \chi = 0$$
$$\text{on} \quad \Upsilon_{wall} \cup \Upsilon_{in} : \qquad \chi = 0$$
$$\text{on} \quad \Upsilon_{out} \cup \Upsilon_{c} : \qquad \frac{\partial \chi}{\partial n} = 0$$
$$\chi(\cdot, T, \cdot) = 0$$

gradient equation

$$\beta u + \chi|_{\Upsilon_c} = 0.$$

OTD Strategy and Picard (Oseen) Iteration $\rightsquigarrow$

state equation

$$v_t - \nu\Delta v + (\bar{v}\cdot\nabla)\, v + \bigtriangledown p = 0$$
$$\nabla\cdot v = 0 + \text{ boundary conditions}$$

adjoint equation

$$\begin{aligned} -\chi_t - \Delta\chi - (\bar{v}\cdot\nabla)\,\chi + (\nabla\bar{v})^T\,\chi + \bigtriangledown\mu &= -\mathrm{curl}^2 v \\ \nabla\cdot\chi &= 0 \\ \text{on} \quad \Upsilon_{wall}\cup\Upsilon_{in}: \qquad \chi &= 0 \\ \text{on} \quad \Upsilon_{out}\cup\Upsilon_{c}: \qquad \frac{\partial\chi}{\partial n} &= 0 \\ \chi(\cdot, T, \cdot) &= 0 \end{aligned}$$

gradient equation

$$\beta u + \chi|_{\Upsilon_c} = 0.$$

- $\bar{v}$ denotes the velocity from the previous Oseen iteration.
- Having solved this system, we update $\bar{v} = v$ until convergence.

Stochastic Galerkin Finite Element Method

- Velocity v and control u are of the form

$$z(t,x,\omega) = \sum_{k=0}^{P-1} \sum_{j=1}^{J_v} z_{jk}(t)\phi_j(x)\psi_k(\xi) = \sum_{k=0}^{P-1} z_k(t,x)\psi_k(\xi).$$

- Pressure p is of the form

$$p(t,x,\omega) = \sum_{k=0}^{P-1} \sum_{j=1}^{J_p} p_{jk}(t)\tilde{\phi}_j(x)\psi_k(\xi) = \sum_{k=0}^{P-1} p_k(t,x)\psi_k(\xi).$$

- Here,
  - $\{\phi_j\}_{j=1}^{J_v}$ and $\{\tilde{\phi}_j\}_{j=1}^{J_p}$ are Q2–Q1 finite elements (inf-sup stable);
  - $\{\psi_k\}_{k=0}^{P-1}$ are Legendre polynomials.
- Implicit Euler/dG(0) used for temporal discretization.

Discrete Stochastic Control Problem

Linearization and SGFEM discretization yields the following saddle point system

$$\underbrace{\begin{bmatrix} M_y & 0 & L^* \\ 0 & M_u & N^\top \\ L & N & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} y \\ u \\ \lambda \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} f \\ 0 \\ g \end{bmatrix}}_{b}.$$

Each of the block matrices in A is of the form

$$\sum_{\alpha=1}^{R} X_\alpha \otimes Y_\alpha \otimes Z_\alpha,$$

corresponding to temporal, stochastic, and spatial discretizations.

Discrete Stochastic Control Problem

Linearization and SGFEM discretization yields the following saddle point system

$$\underbrace{\begin{bmatrix} M_y & 0 & L^* \\ 0 & M_u & N^\top \\ L & N & 0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} y \\ u \\ \lambda \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} f \\ 0 \\ g \end{bmatrix}}_{b}.$$

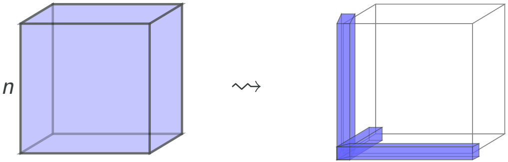
Each of the block matrices in A is of the form

$$\sum_{\alpha=1}^{R} X_\alpha \otimes Y_\alpha \otimes Z_\alpha,$$

corresponding to temporal, stochastic, and spatial discretizations.

Size: $\sim 3n_t P(J_v + J_p)$, e.g., for $P = 10$, $n_t = 2^{10}$, $J \approx 10^5 \rightsquigarrow \approx 10^9$ unknowns!

Separation of variables and low-rank approximation



- Approximate: $\underbrace{\mathbf{x}(i_1,\ldots,i_d)}_{\text{tensor}} \approx \underbrace{\sum_{\alpha} \mathbf{x}_{\alpha}^{(1)}(i_1)\mathbf{x}_{\alpha}^{(2)}(i_2)\cdots\mathbf{x}_{\alpha}^{(d)}(i_d)}_{\text{tensor product decomposition}}.$

Goals:

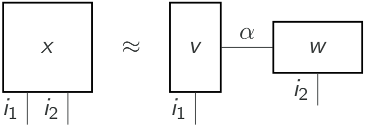
- Store and manipulate x — $\mathcal{O}(dn)$ cost *instead of* $\mathcal{O}(n^d)$.
- Solve equations $Ax = b$ — $\mathcal{O}(dn^2)$ cost *instead of* $\mathcal{O}(n^{2d})$.

Data Compression in 2D: Low-Rank Matrices

- Discrete *separation of variables*:

$$\begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & & \vdots \\ x_{n,1} & \cdots & x_{n,n} \end{bmatrix} = \sum_{\alpha=1}^{\color{blue}r} \begin{bmatrix} v_{1,\alpha} \\ \vdots \\ v_{n,\alpha} \end{bmatrix} \begin{bmatrix} w_{\alpha,1} & \cdots & w_{\alpha,n} \end{bmatrix} + \mathcal{O}({\color{red}\varepsilon}).$$

- Diagrams:



- *Rank* ${\color{blue}r} \ll n$.
- $\mathtt{mem}(v) + \mathtt{mem}(w) = {\color{blue}2nr} \ll n^2 = \mathtt{mem}(x)$.
- *Singular Value Decomposition (SVD)*
  $\Longrightarrow \quad {\color{red}\varepsilon}({\color{blue}r})$ optimal w.r.t. spectral/Frobenius norm.

Tensor Trains/Matrix Product States

[Wilson '75, White '93, Verstraete '04, Oseledets '09/'11]

For indices

$$\overline{i_p \dots i_q} = (i_p - 1)n_{p+1} \cdots n_q + (i_{p+1} - 1)n_{p+2} \cdots n_q + \cdots + (i_{q-1} - 1)n_q + i_q,$$

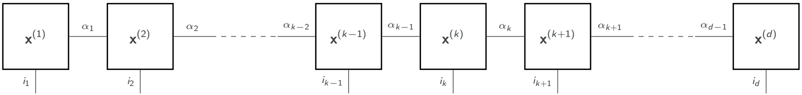
the TT format can be expressed as

$$x(\overline{i_1 \dots i_d}) = \sum_{\boldsymbol{\alpha}=\mathbf{1}}^{\mathbf{r}} \mathbf{x}^{(1)}_{\alpha_1}(i_1) \cdot \mathbf{x}^{(2)}_{\alpha_1,\alpha_2}(i_2) \cdot \mathbf{x}^{(3)}_{\alpha_2,\alpha_3}(i_3) \cdots \mathbf{x}^{(d)}_{\alpha_{d-1},\alpha_d}(i_d)$$

or

$$x(\overline{i_1 \dots i_d}) = \mathbf{x}^{(1)}(i_1) \cdots \mathbf{x}^{(d)}(i_d), \quad \mathbf{x}^{(k)}(i_k) \in \mathbb{R}^{r_{k-1} \times r_k} \text{ w/ } r_0, r_d = 1,$$

or



Storage: $\mathcal{O}(dnr^2)$ instead of $\mathcal{O}(n^d)$.

Overloading Tensor Operations

Always work with *factors* $\mathbf{x}^{(k)} \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ instead of full tensors.

- Sum $z = x + y \quad \rightsquigarrow \quad$ increase of tensor rank $\quad r_z = r_x + r_y$.
- TT format for a high-dimensional operator

$$A(\overline{i_1 \dots i_d}, \overline{j_1 \dots j_d}) = \mathbf{A}^{(1)}(i_1, j_1) \cdots \mathbf{A}^{(d)}(i_d, j_d)$$

- *Matrix-vector* multiplication $y = Ax$; $\quad \rightsquigarrow$ tensor rank $r_y = r_A \cdot r_x$.
- Additions and multiplications *increase* TT ranks.
- *Decrease* ranks quasi-optimally via QR and SVD.

Solving KKT System using TT Format

The dimensionality of the saddle point system is vast $\Rightarrow$ use tensor structure and low tensor ranks.

Solving KKT System using TT Format

The dimensionality of the saddle point system is vast $\Rightarrow$ use tensor structure and low tensor ranks.

Use tensor train format to approximate the solution as

$$\mathbf{y}(i_1,\ldots,i_d) \approx \sum_{\alpha_1\ldots\alpha_{d-1}=1}^{r_1\ldots r_{d-1}} \mathbf{y}^{(1)}_{\alpha_1}(i_1)\mathbf{y}^{(2)}_{\alpha_1,\alpha_2}(i_2)\cdots\mathbf{y}^{(d-1)}_{\alpha_{d-2},\alpha_{d-1}}(i_{d-1})\mathbf{y}^{(d)}_{\alpha_{d-1}}(i_d),$$

and represent the coefficient matrix as

$$\mathcal{A}(i_1\cdots i_d, j_1\cdots j_d) \approx \sum_{\beta_1\ldots\beta_{d-1}=1}^{R_1\ldots R_{d-1}} \mathbf{A}^{(1)}_{\beta_1}(i_1,j_1)\mathbf{A}^{(2)}_{\beta_1,\beta_2}(i_2,j_2)\cdots\mathbf{A}^{(d)}_{\beta_{d-1}}(i_d,j_d),$$

where the multi-index $\mathbf{i}=(i_1,\ldots,i_d)$ is implied by the parametrization of the approximate solutions of the form

$$\mathbf{z}(t,\xi_1,\ldots,\xi_N,\mathbf{x}), \quad \mathbf{z}=\mathbf{y},\mathbf{u},\mathbf{p},$$

i.e., solution vectors are represented by d-way tensor with $d=N+2$.

Linear Systems in TT Format

Central Question

How to solve $Ax = b$ if Krylov solvers become too expensive?

Linear Systems in TT Format

Central Question

How to solve $Ax = b$ if Krylov solvers become too expensive?

Data are given in TT format:

- $A(i,j) = \mathbf{A}^{(1)}(i_1, j_1) \cdots \mathbf{A}^{(d)}(i_d, j_d)$.
- $b(i) = \mathbf{b}^{(1)}(i_1) \cdots \mathbf{b}^{(d)}(i_d)$.

Seek the solution in the same format:

- $x(i) = \mathbf{x}^{(1)}(i_1) \cdots \mathbf{x}^{(d)}(i_d)$.

Linear Systems in TT Format

Central Question

How to solve $Ax = b$ if Krylov solvers become too expensive?

Data are given in TT format:

- $A(i,j) = \mathbf{A}^{(1)}(i_1, j_1) \cdots \mathbf{A}^{(d)}(i_d, j_d)$.
- $b(i) = \mathbf{b}^{(1)}(i_1) \cdots \mathbf{b}^{(d)}(i_d)$.

Seek the solution in the same format:

- $x(i) = \mathbf{x}^{(1)}(i_1) \cdots \mathbf{x}^{(d)}(i_d)$.

Use a new block-variant of *Alternating Least Squares* in a new block TT format to overcome difficulties with indefiniteness of KKT system matrix.

- If $A = A^\top > 0$: minimize $J(x) = x^\top A x - 2x^\top b$.

Alternating Least Squares (ALS):

- replace $\min_x J(x)$ by iteration — size n^d
- `for` $k = 1, \ldots, d$, — size $r^2 n$
  solve $\min_{\mathbf{x}^{(k)}} J\left(\mathbf{x}^{(1)}(i_1) \cdots \mathbf{x}^{(k)}(i_k) \cdots \mathbf{x}^{(d)}(i_d)\right)$.
  (all other blocks are fixed)

ALS for $d = 3$

1. $\hat{\mathbf{x}}^{(1)} = \arg\min_{\mathbf{x}^{(1)}} J\left(\mathbf{x}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$

1. $\hat{\mathbf{x}}^{(1)} = \arg\min_{\mathbf{x}^{(1)}} J\left(\mathbf{x}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
2. $\hat{\mathbf{x}}^{(2)} = \arg\min_{\mathbf{x}^{(2)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$

ALS for $d = 3$

1. $\hat{\mathbf{x}}^{(1)} = \arg\min_{\mathbf{x}^{(1)}} J\left(\mathbf{x}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
2. $\hat{\mathbf{x}}^{(2)} = \arg\min_{\mathbf{x}^{(2)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
3. $\hat{\mathbf{x}}^{(3)} = \arg\min_{\mathbf{x}^{(3)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\hat{\mathbf{x}}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$

1. $\hat{\mathbf{x}}^{(1)} = \arg\min_{\mathbf{x}^{(1)}} J\left(\mathbf{x}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
2. $\hat{\mathbf{x}}^{(2)} = \arg\min_{\mathbf{x}^{(2)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
3. $\hat{\mathbf{x}}^{(3)} = \arg\min_{\mathbf{x}^{(3)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\hat{\mathbf{x}}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
4. $\mathbf{x}^{(2)} = \arg\min_{\mathbf{x}^{(2)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\hat{\mathbf{x}}^{(3)}(i_3)\right)$

ALS for $d = 3$

1. $\hat{\mathbf{x}}^{(1)} = \arg\min_{\mathbf{x}^{(1)}} J\left(\mathbf{x}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
2. $\hat{\mathbf{x}}^{(2)} = \arg\min_{\mathbf{x}^{(2)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
3. $\hat{\mathbf{x}}^{(3)} = \arg\min_{\mathbf{x}^{(3)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\hat{\mathbf{x}}^{(2)}(i_2)\mathbf{x}^{(3)}(i_3)\right)$
4. $\mathbf{x}^{(2)} = \arg\min_{\mathbf{x}^{(2)}} J\left(\hat{\mathbf{x}}^{(1)}(i_1)\mathbf{x}^{(2)}(i_2)\hat{\mathbf{x}}^{(3)}(i_3)\right)$
5. repeat 1.–4. until convergence

ALS = Projection method

If we differentiate J w.r.t. TT blocks, we see that...

- ...each step means solving a *Galerkin linear system*

$$\left(X_{\neq k}^{\top} A X_{\neq k}\right) \hat{x}^{(k)} = \left(X_{\neq k}^{\top} b\right) \in \mathbb{R}^{nr^2}.$$

- $X_{\neq k} = \underbrace{\mathrm{TT}\left(\hat{\mathbf{x}}^{(1)} \dots \hat{\mathbf{x}}^{(k-1)}\right)}_{n^{k-1} \times r_{k-1}} \otimes \underbrace{I}_{n \times n} \otimes \underbrace{\mathrm{TT}\left(\mathbf{x}^{(k+1)} \dots \mathbf{x}^{(d)}\right)}_{n^{d-k} \times r_k}.$

ALS = Projection method

If we differentiate J w.r.t. TT blocks, we see that...

- ...each step means solving a *Galerkin linear system*

$$\left(X_{\neq k}^{\top} A X_{\neq k}\right) \hat{x}^{(k)} = \left(X_{\neq k}^{\top} b\right) \in \mathbb{R}^{nr^2}.$$

- $X_{\neq k} = \underbrace{\mathrm{TT}\left(\hat{\mathbf{x}}^{(1)} \dots \hat{\mathbf{x}}^{(k-1)}\right)}_{n^{k-1} \times r_{k-1}} \otimes \underbrace{I}_{n \times n} \otimes \underbrace{\mathrm{TT}\left(\mathbf{x}^{(k+1)} \dots \mathbf{x}^{(d)}\right)}_{n^{d-k} \times r_k}.$

Properties of ALS include:

+ Effectively 1D complexity in a prescribed format.
− Tensor format (ranks) is fixed and cannot be adapted.
− Convergence may be very slow, stagnation is likely.

ALS: Getting rid of "–"

- Density Matrix Renormalization Group (DMRG) [White '92]
  – updates *two* blocks $\mathbf{x}^{(k)}\mathbf{x}^{(k+1)}$ *simultaneously*.

- Alternating Minimal Energy (AMEn) [Dolgov/Savostyanov '13]
  – *augments* $X_{\neq k}$ by a TT block of the *residual* $\mathbf{z}^{(k)}$.

ALS: Getting rid of "–"

- Density Matrix Renormalization Group (DMRG) [White '92]
  – updates *two* blocks $\mathbf{x}^{(k)}\mathbf{x}^{(k+1)}$ *simultaneously*.
- Alternating Minimal Energy (AMEn) [Dolgov/Savostyanov '13]
  – *augments* $X_{\neq k}$ by a TT block of the *residual* $\mathbf{z}^{(k)}$.

But. . . , what about saddle point systems A?

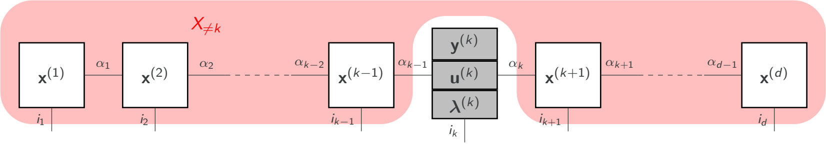
- Recall our KKT system:

$$\underbrace{\begin{bmatrix} M_y & 0 & L^* \\ 0 & M_u & N^\top \\ L & N & 0 \end{bmatrix}}_{A} \begin{bmatrix} y \\ u \\ \lambda \end{bmatrix} = \begin{bmatrix} f \\ 0 \\ g \end{bmatrix}.$$

- The whole matrix is indefinite $\Rightarrow X_{\neq k}^\top A X_{\neq k}$ can be degenerate.

- Work-around: Block TT representation

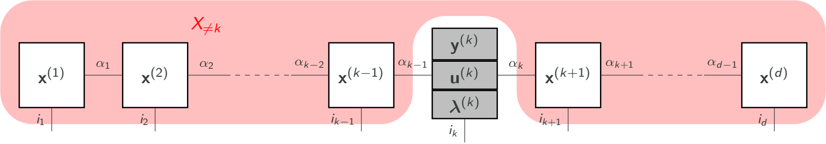
$$\begin{bmatrix} y \\ u \\ \lambda \end{bmatrix} = \mathbf{x}^{(1)}_{\alpha_1} \otimes \cdots \otimes \begin{bmatrix} \mathbf{y}^{(k)}_{\alpha_{k-1},\alpha_k} \\ \mathbf{u}^{(k)}_{\alpha_{k-1},\alpha_k} \\ \boldsymbol{\lambda}^{(k)}_{\alpha_{k-1},\alpha_k} \end{bmatrix} \otimes \cdots \otimes \mathbf{x}^{(d)}_{\alpha_{d-1}}.$$



- $X_{\neq k}$ is the *same* for y, u, λ.

- Work-around: Block TT representation

$$\begin{bmatrix} y \\ u \\ \lambda \end{bmatrix} = \mathbf{x}^{(1)}_{\alpha_1} \otimes \cdots \otimes \begin{bmatrix} \mathbf{y}^{(k)}_{\alpha_{k-1},\alpha_k} \\ \mathbf{u}^{(k)}_{\alpha_{k-1},\alpha_k} \\ \boldsymbol{\lambda}^{(k)}_{\alpha_{k-1},\alpha_k} \end{bmatrix} \otimes \cdots \otimes \mathbf{x}^{(d)}_{\alpha_{d-1}}.$$



- $X_{\neq k}$ is the *same* for y, u, λ.
- Project each *submatrix*:

$$\begin{bmatrix} \hat{M}_y & 0 & \hat{L}^* \\ 0 & \hat{M}_u & \hat{N}^\top \\ \hat{L} & \hat{N} & 0 \end{bmatrix} \begin{bmatrix} y^{(k)} \\ u^{(k)} \\ \lambda^{(k)} \end{bmatrix} = \begin{bmatrix} \hat{f} \\ 0 \\ \hat{g} \end{bmatrix}, \qquad \widehat{(\cdot)} = X_{\neq k}^\top (\cdot) X_{\neq k}.$$

Numerical Experiments

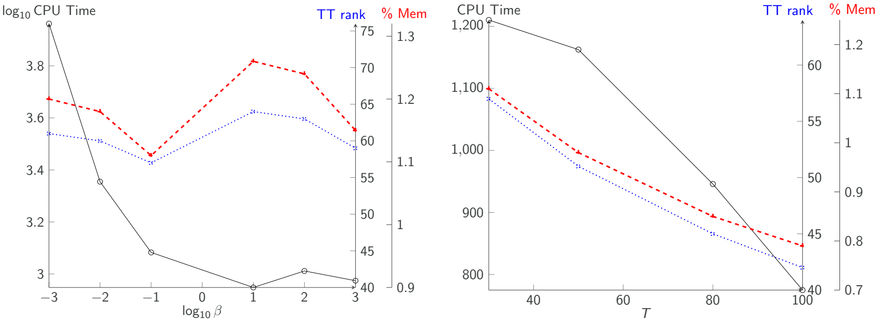
◂ Kármán vortex street

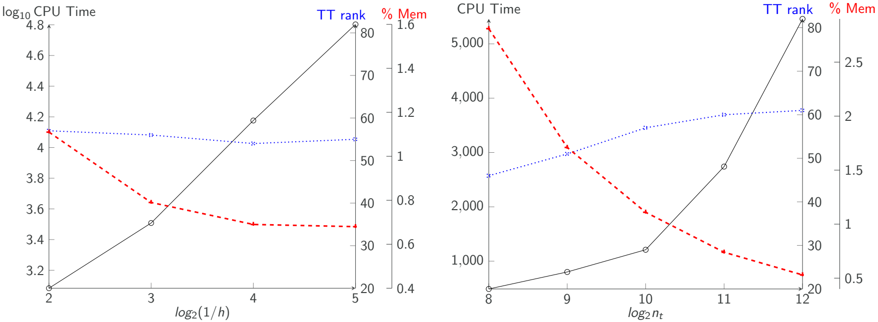
Vary one of the default parameters:

- TT truncation tolerance $\varepsilon = 10^{-4}$,
- mean viscosity $\nu_0 = 1/20$,
- uncertainty $\nu_1 = 1/80$,
- regularization/penalty parameter $\beta = 10^{-1}$,
- number of time steps: $n_t = 2^{10}$,
- time horizon $T = 30$,
- spatial grid size $h = 1/4 \rightsquigarrow J = 2488$,
- max. degree of Legendre polynomials: $P = 8$.

Solve projected linear systems using block-preconditioned GMRES using efficient approximation of Schur complement [B/Onwunta/Stoll 2016].

Varying regularization β (left) and time T (right)




log10 CPU Time
TT rank
% Mem
log2(1/h)
CPU Time
TT rank
% Mem
log2 n_t

Varying the Viscosity

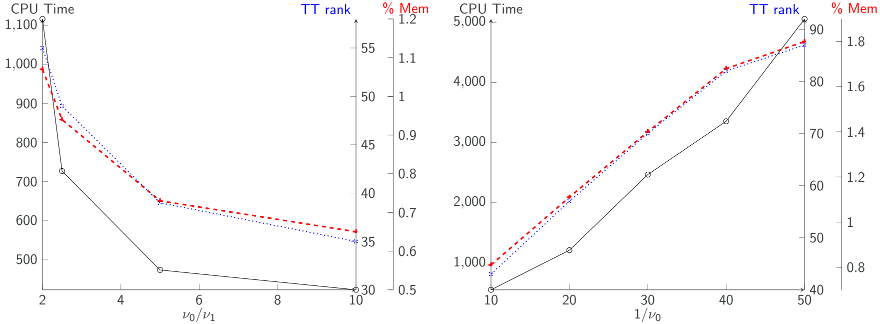
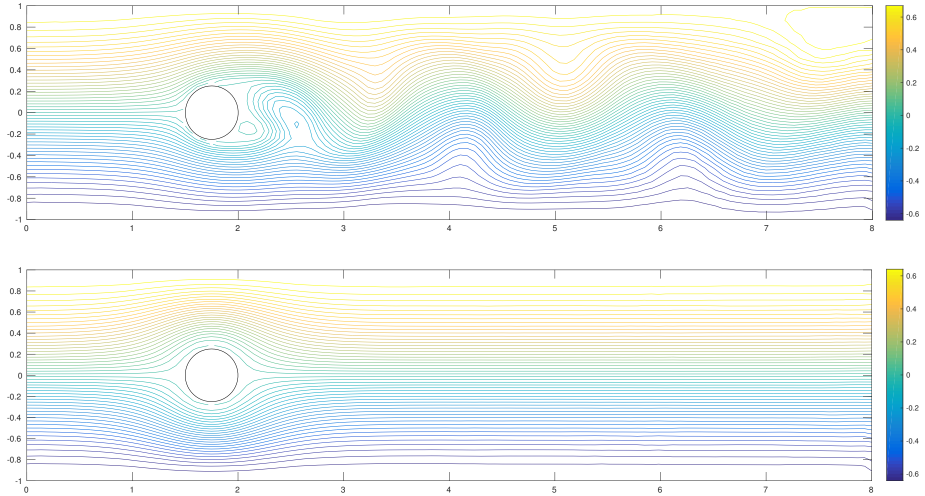
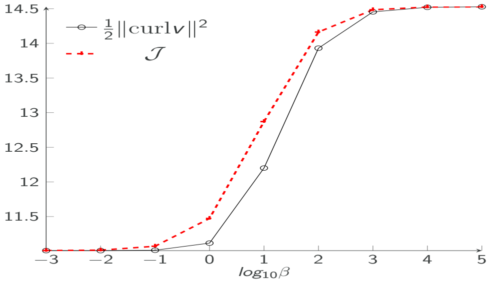


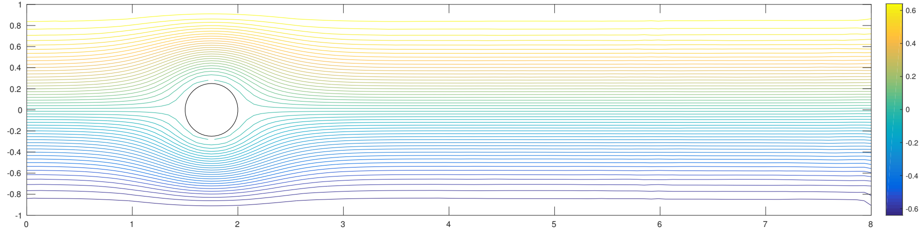
Figure: Left: $\nu_0 = 1/10$, ν_1 is varied. Right: ν_1 and ν_0 are varied together as $\nu_1 = 0.25\nu_0$

Controlled von Kármán Vortex Street



Controlled von Kármán Vortex Street



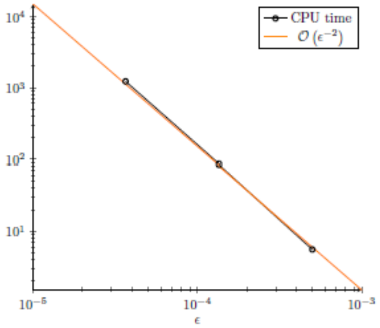


Emprical Complexity Study

Balance all errors w.r.t.

- time, spatial, and stochastic discretization;
- TT truncation.

Call the balanced error ε.

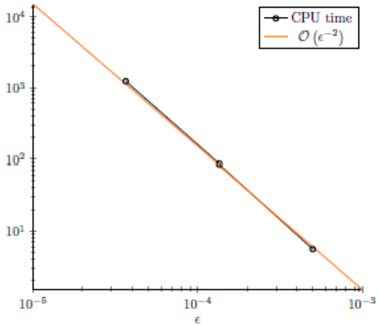


Emprical Complexity Study

Balance all errors w.r.t.

- time, spatial, and stochastic discretization;
- TT truncation.

Call the balanced error ε.

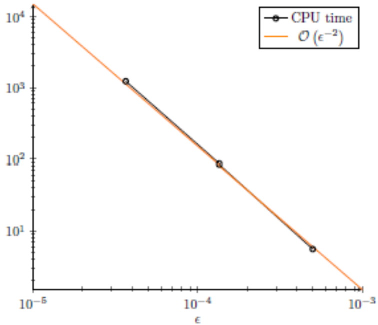


This indicates asymptotic complexity ε^{-2}, asymptotically equal complexity as for deterministic problem.

Balance all errors w.r.t.

- time, spatial, and stochastic discretization;
- TT truncation.

Call the balanced error ε.



This indicates asymptotic complexity ε^{-2}, asymptotically equal complexity as for deterministic problem.

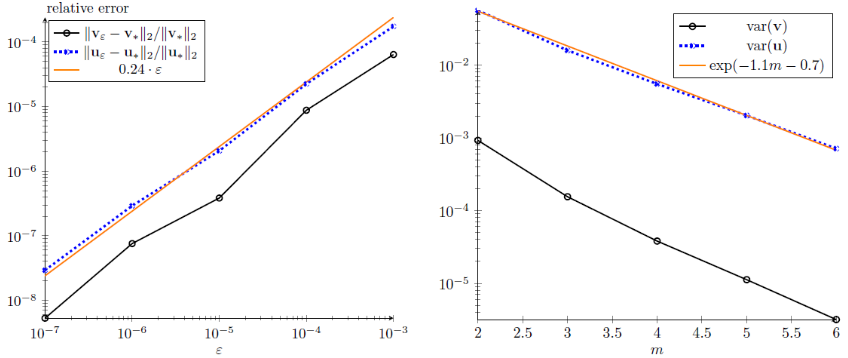
This compares favorably to

- Monte-Carlo $\mathcal{O}(\varepsilon^{-4})$,
- quasi Monte-Carlo / stochastic collocation $\mathcal{O}(\varepsilon^{-3})$.

Relative errors for different TT approximation thresholds (left) and varying m (right)

Parameters:

viscosity $\nu \equiv \nu_0 = 10^{-3}$, final time $T = 20$, regularization parameter $\beta = 10^{-2}$, KL decay rate $\gamma = 2.5$ for inflow condition.

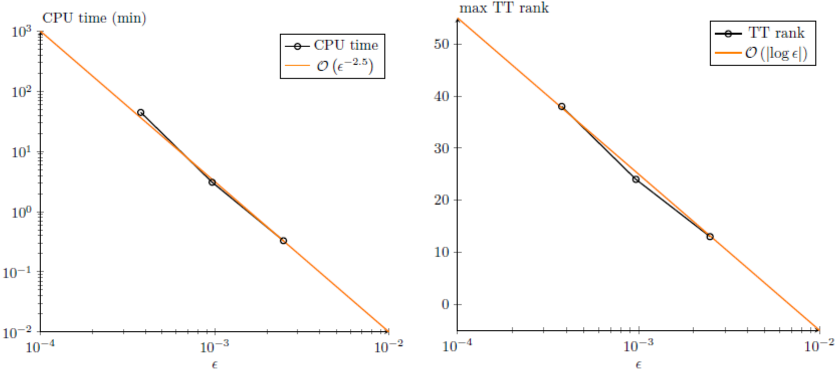


Results for Backward Facing Step

Total CPU time (left) and TT rank (right) versus total approximation error.

Parameters:

viscosity $\nu \equiv \nu_0 = 10^{-3}$, final time $T = 20$, regularization parameter $\beta = 10^{-2}$, KL decay rate $\gamma = 2.5$ for inflow condition.



Outline

1. Introduction
2. Optimal Control of Unsteady Navier-Stokes Equations under Uncertainty
3. Conclusions

Conclusions & Outlook

- Low-rank tensor solver for unsteady heat and Navier-Stokes equations with uncertain viscosity.
- Similar techniques already used for Stokes(-Brinkman) optimal control problems.
- Adapted AMEn (TT) solver to saddle point systems.
- With 1 stochastic parameter, the scheme reduces complexity by up to 2–3 orders of magnitude.
- To consider next:

Conclusions & Outlook

- Low-rank tensor solver for unsteady heat and Navier-Stokes equations with uncertain viscosity.
- Similar techniques already used for Stokes(-Brinkman) optimal control problems.
- Adapted AMEn (TT) solver to saddle point systems.
- With 1 stochastic parameter, the scheme reduces complexity by up to 2–3 orders of magnitude.
- To consider next:
  - many parameters coming from uncertain geometry or Karhunen-Loève expansion of random fields;
    Basic observation: the more parameters, the more significant is the complexity reduction w.r.t. memory — up to a factor of 10^9 for the control problem for a backward facing step set-up.
  - HPC implementation of AMEn-like solver to deal with even larger problems.

References

P. Benner, S. Dolgov, A. Onwunta, and M. Stoll.
Low-rank solvers for unsteady Stokes-Brinkman optimal control problem with random data.
Computer Methods in Applied Mechanics and Engineering, 304:26–54, 2016.

P. Benner, A. Onwunta, and M. Stoll.
Low rank solution of unsteady diffusion equations with stochastic coefficients.
SIAM/ASA Journal on Uncertainty Quantification, 3(1):622–649, 2015.

P. Benner, A. Onwunta, and M. Stoll.
Block-diagonal preconditioning for optimal control problems constrained by PDEs with uncertain inputs.
SIAM Journal on Matrix Analysis and Applications, 37(2):491–518, 2016.

P. Benner, S. Dolgov, A. Onwunta, and M. Stoll.
Solving optimal control problems governed by random Navier-Stokes equations using low-rank methods. arXiv:1703.06097, March 2017.