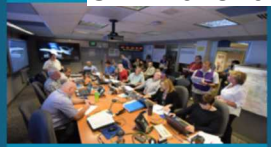
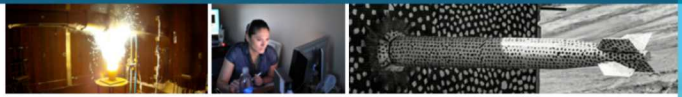
SAND2018-6960C

Multigrid-in-time methods for optimization with nonlinear PDE/DAE constraints





Denis Ridzal, Eric Cyr

23rd International Symposium on Mathematical Programming
Bordeaux, France

ional Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wh
iary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA00
July 1, 2018
DEFENSE SCIENCES OFFICE
Administration under contract DE-NA0003525.

Motivation

Optimization algorithm

Time-domain decomposition

Coarse-grid correction in time

Future work

Motivation

Optimization algorithm

Time-domain decomposition

Coarse-grid correction in time

Future work

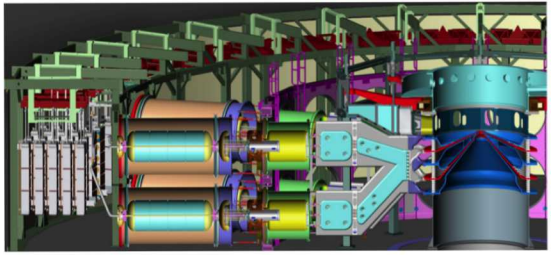
Design for Sandia's Z-Machine

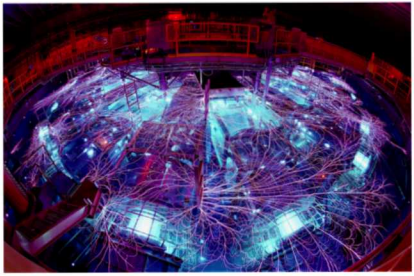
Pulsed power is important:

- Science;
- Fusion energy;
- National security.

Power losses have plagued current designs.

- Can numerical optimization help?





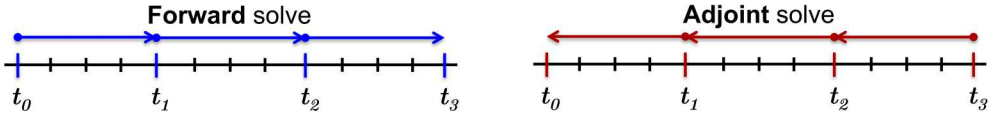
PDE constraints: Plasma models.

- Many-fluid models coupled with Maxwell's equation.
- Strongly nonlinear dynamics.
- A single transient simulation **takes a day to run**!
- Optimization: $\times$**100–1000**.

Similar challenges: optimal control of power grids, trajectory planning for atmospheric reentry, etc.

Exascale computing

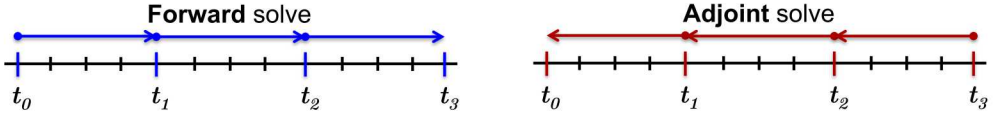
- The promise of **million-way** parallelism.
- **Hierarchies in hardware** mapping to **hierarchies in software**.
- Conventional optimization approaches inherit the **serial bottleneck** from forward simulation, namely the **time discretization**.



- Exascale may speed up forward and adjoint solves in some cases, but can we do *fundamentally better* in the context of optimization?

Exascale computing

- The promise of **million-way** parallelism.
- **Hierarchies in hardware** mapping to **hierarchies in software**.
- Conventional optimization approaches inherit the **serial bottleneck** from forward simulation, namely the **time discretization**.



- Exascale may speed up forward and adjoint solves in some cases, but can we do *fundamentally better* in the context of optimization?
- **Exploit time parallelism at the level of the optimization algorithm.**

Example: Control of viscous Burgers' equation

$$\underset{u,z}{\text{Minimize}} \quad \frac{1}{2}\int_0^T \int_0^1 (u(x,t) - \widehat{u}(x,t))^2 + \alpha z^2(x,t)\, dx\, dt$$

subject to

$$\frac{\partial}{\partial t}u(x,t) - \nu\frac{\partial^2}{\partial x^2}u(x,t) + \frac{\partial}{\partial x}u(x,t)u(x,t) = z(x,t)$$

$$u(0,t) = u(1,t) = 0; \quad u(x,0) = u_0(x)$$

where $x \in (0,1)$, $t \in (0,T)$, $u_0(x) = \{1 : x \in (0,1/2]; 0 : x \in (1/2,1)\}$, and $\widehat{u}(x,t) = \{1 : (x,t) \in (0,1/2] \times (0,1); 0 : (x,t) \in (1/2,1) \times (0,1)\}$.

- Nonlinear transient problem with advective and diffusive features.
- Stepping stone to Navier-Stokes and other fluid models.
- Good start for a parallel-in-time method.

Discretization in space and time

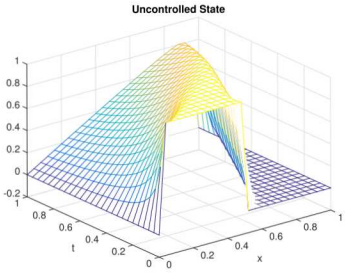
- Space: FE matrices M, A, Q, B; nonlinear operator N; 100 intervals.
- Time: θ-method on $0 = t_0 < t_1 < \cdots < t_{N+1} = t_F$; $\theta = 0.5$.
- Other details: viscosity $\nu = 0.01$, control penalty $\alpha = 0.05$, etc., see Heinkenschloss (2008), Rice University CAAM Tech. Rep. TR08-05.

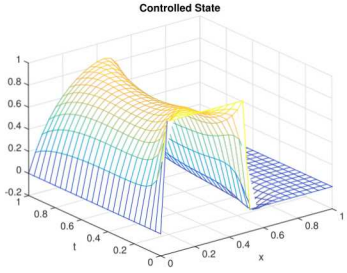
$$\underset{u,z}{\text{Minimize}} \quad \sum_{i=1}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} u_i^T M u_i + g(t_i)^T u_i \right) + \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{\alpha}{2} z_i^T Q z_i \right)$$

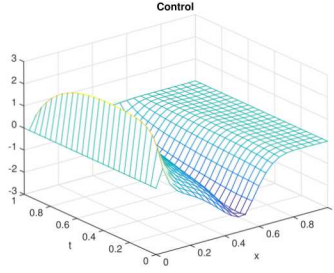
subject to

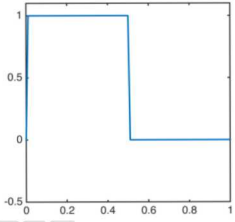
$$\begin{aligned} (M + \theta \Delta t_i A) u_{i+1} \quad &+ \theta \Delta t_i N(u_{i+1}) \quad + \theta \Delta t_i B z_{i+1} + \\ (-M + (1-\theta) \Delta t_i A) u_i + (1-\theta) \Delta t_i N(u_i) &+ (1-\theta) \Delta t_i B z_i \quad = 0 \end{aligned}$$

Reference solution









Sequential Quadratic Programming (SQP)

Solve equality-constrained optimization problem:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & J(x) \\ \text{s.t.} \quad & c(x) = 0 \end{aligned}$$

where $J : \mathcal{X} \to \mathbb{R}$ and $c : \mathcal{X} \to \mathcal{C}$, for some Hilbert spaces $\mathcal{X}$ and $\mathcal{C}$, and J and c are twice continuously Fréchet differentiable. We identify the spaces $\mathcal{X}$ and $\mathcal{C}$ with their duals.

For time-dependent PDEs, $\mathcal{X} = \mathcal{U} \times \mathcal{Z} = (\mathcal{U}_\Omega \times \mathcal{U}_t) \times (\mathcal{Z}_\Omega \times \mathcal{Z}_t)$.

Define Lagrangian functional $L : \mathcal{X} \times \mathcal{C} \to \mathbb{R}$:

$$L(x, \lambda) = J(x) + \langle \lambda, c(x) \rangle_{\mathcal{C}}$$

If *regular* point x_* is a local solution of the NLP, then there exists a $\lambda_* \in \mathcal{C}$ satisfying the *first-order necessary optimality conditions*:

$$\begin{aligned} \nabla_x J(x_*) + c_x(x_*)^* \lambda_* &= 0 \\ c(x_*) &= 0 \end{aligned}$$

Sequential Quadratic Programming (SQP)

Newton's method applied to optimality conditions:

$$\begin{pmatrix} \nabla_{xx} L(x_k, \lambda_k) & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} s \\ z \end{pmatrix} = - \begin{pmatrix} \nabla_x J(x_k) + c_x(x_k)^* \lambda_k \\ c(x_k) \end{pmatrix}$$

If $\nabla_{xx} L(x_k, \lambda_k)$ is positive definite on the null space of $c_x(x_k)$, the above KKT system is necessary and sufficient for solving the QP:

$$\begin{aligned} \min_{s \in \mathcal{X}} \quad & \frac{1}{2} \langle \nabla_{xx} L(x_k, \lambda_k) s, s \rangle_{\mathcal{X}} + \langle \nabla_x L(x_k, \lambda_k), s \rangle_{\mathcal{X}} + L(x_k, \lambda_k) \\ \text{s.t.} \quad & c_x(x_k) s + c(x_k) = 0 \end{aligned}$$

Sequential Quadratic Programming (SQP)

Newton's method applied to optimality conditions:

$$\begin{pmatrix} \nabla_{xx} L(x_k, \lambda_k) & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} s \\ z \end{pmatrix} = - \begin{pmatrix} \nabla_x J(x_k) + c_x(x_k)^* \lambda_k \\ c(x_k) \end{pmatrix}$$

If $\nabla_{xx} L(x_k, \lambda_k)$ is positive definite on the null space of $c_x(x_k)$, the above KKT system is necessary and sufficient for solving the QP:

$$\begin{aligned} \min_{s \in \mathcal{X}} \quad & \frac{1}{2} \langle \nabla_{xx} L(x_k, \lambda_k) s, s \rangle_{\mathcal{X}} + \langle \nabla_x L(x_k, \lambda_k), s \rangle_{\mathcal{X}} + L(x_k, \lambda_k) \\ \text{s.t.} \quad & c_x(x_k) s + c(x_k) = 0 \end{aligned}$$

Solve a sequence of *nonconvex* quadratic trust-region subproblems:

$$\begin{aligned} \min_{s \in \mathcal{X}} \quad & \frac{1}{2} \langle \nabla_{xx} L(x_k, \lambda_k) s, s \rangle_{\mathcal{X}} + \langle \nabla_x L(x_k, \lambda_k), s \rangle_{\mathcal{X}} + L(x_k, \lambda_k) \\ \text{s.t.} \quad & c_x(x_k) s + c(x_k) = 0\,, \quad \|s\|_{\mathcal{X}} \leq \Delta_k \end{aligned}$$

Possible incompatibility of constraints: Byrd-Omojokun composite-step approach.

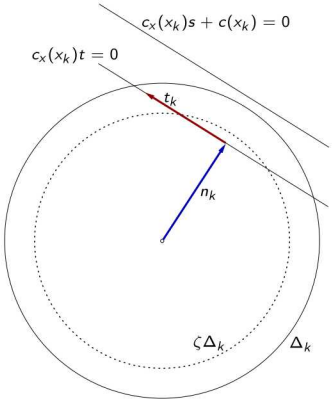
Composite-step method for trust-region subproblem

- **Trust-region step**: $s_k = n_k + t_k$

- **Quasi-normal step** n_k:

  reduces linear infeasibility

$$\begin{aligned} \min_{n \in \mathcal{X}} \quad & \|c_x(x_k)n + c(x_k)\|_C^2 \\ \text{s.t.} \quad & \|n\|_{\mathcal{X}} \leq \zeta \Delta_k \end{aligned}$$



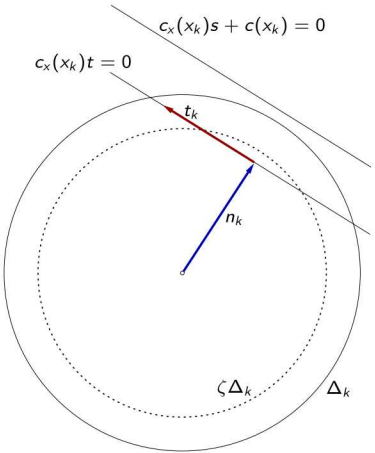
- **Tangential step** t_k:

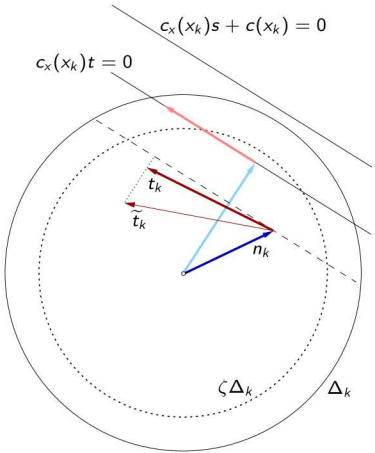
  improves optimality while staying in the null space of the linearized constraints

$$\begin{aligned} \min_{t \in \mathcal{X}} \quad & \frac{1}{2}\langle \nabla_{xx} L(x_k, \lambda_k)(t + n_k), t + n_k \rangle_{\mathcal{X}} + \langle \nabla_x L(x_k, \lambda_k), t + n_k \rangle_{\mathcal{X}} + L(x_k, \lambda_k) \\ \text{s.t.} \quad & c_x(x_k)t = 0\,, \quad \|t + n_k\|_{\mathcal{X}} \leq \Delta_k \end{aligned}$$

Omojokun (1989), Byrd, Hribar, Nocedal (1997), Dennis, El-Alem, Maciel (1997)

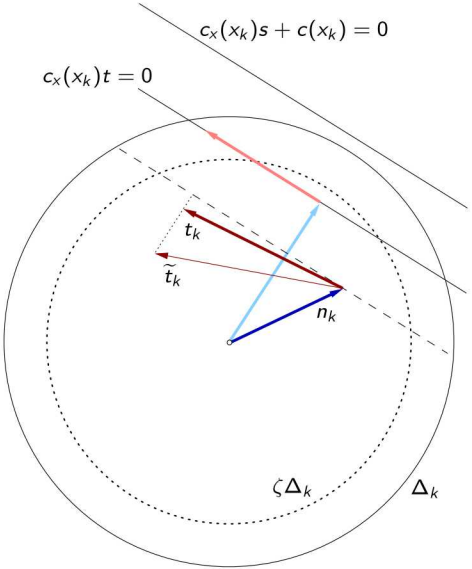
Composite step with iterative (matrix-free) solvers





With matrix-free linear algebra — such as iterative linear solvers — the quasi-normal and tangential steps are computed **inexactly**. This is an **opportunity**!

Matrix-free trust-region SQP algorithm



Composite step:
$s_k = n_k + t_k$

1. Compute quasi-normal step n_k using **inexact Powell dogleg**.
2. Solve tangential subproblem for $\widetilde{t}_k$ using **inexact projected Steihaug-Toint CG**.
3. Restore linearized feasibility, yielding tangential step t_k.
4. Update Lagrange multipliers λ_{k+1}.
5. Evaluate progress.

Ridzal, *Ph.D. Thesis, Rice University (2006)*
Heinkenschloss, Ridzal, *SIAM J. Opt. (2014)*

Linear systems are ALL augmented constraint systems

$$\begin{pmatrix} I & c_x(x_k)^* \\ c_x(x_k) & 0 \end{pmatrix} \begin{pmatrix} y^1 \\ y^2 \end{pmatrix} = \begin{pmatrix} b^1 \\ b^2 \end{pmatrix} + \begin{pmatrix} e^1 \\ e^2 \end{pmatrix}$$

- The size of $(e^1 \; e^2)$ is governed by various model reduction conditions, i.e., the progress of the optimization algorithm:

$$\|e^1\|_{\mathcal{X}} + \|e^2\|_{\mathcal{C}} \leq \text{func}\left(\|b^1\|_{\mathcal{X}}, \|b^2\|_{\mathcal{C}}, \|y^1\|_{\mathcal{X}}, \Delta_k, \xi\right)$$

- These are KKT systems for the **convex quadratic problems**

$$\begin{aligned} \min \quad & \frac{1}{2}\langle y^1, y^1\rangle_{\mathcal{X}} - \langle b^1, y^1\rangle_{\mathcal{X}} \\ \text{s.t.} \quad & c_x(x_k)y^1 = b^2. \end{aligned}$$

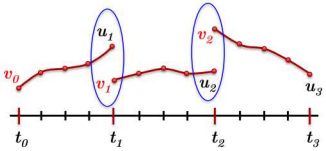
- True even if the original trust-region subproblems are not convex!

Time-domain decomposition

- Our approach is heavily influenced by:
  - Heinkenschloss (2005), J. Comp. Appl. Math., 173:169-198.
  - Comas (2006), Ph. D. Thesis, Rice University.
- Related to multiple shooting methods.
- Some potential differences:
  - Take a purely **algebraic** view of the optimization problem.
  - Introduce **virtual state variables**, at the discrete level, to weaken the coupling in time and define parallelizable smoothers.
  - Use a **coarse-grid** strategy for the **full optimality system**.
  - Embed the approach in our **nonlinear** (SQP) optimization scheme.
  - Make heavy use of **inexact linear solvers**.
  - Ensure applicability to **general (PDE, DAE, etc.)** nonlinear systems.
  - Enable reasonable **software interfaces** for large-scale applications.

Time-domain decomposition

$$\begin{aligned} \min_{u_i, z_i} \quad & \sum_{i=1}^{N} J_i(u_i, z_i) \\ \text{subject to} \quad & u_i = F(u_{i-1}, z_{i-1}, t_{i-1}, t_i) \\ & \text{for } i = 1, \ldots, N \end{aligned}$$



$$\begin{aligned} \min_{u_i, v_i, z_i} \quad & \sum_{i=1}^{N} J_i(u_i, v_i, z_i) \\ \text{subject to} \quad & \text{(I)} \;\; u_i = F(v_{i-1}, z_{i-1}, t_{i-1}, t_i) \\ & \text{(II)} \; u_i = v_i; \quad \text{for } i = 1, \ldots, N \end{aligned}$$

Discretization in space and time for Burgers

$$\underset{u,z}{\text{Minimize}} \quad \sum_{i=1}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} u_i^T M u_i + g(t_i)^T u_i \right) + \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{\alpha}{2} z_i^T Q z_i \right)$$

subject to

$$\begin{aligned} (M + \theta \Delta t_i A) u_{i+1} \quad &+ \theta \Delta t_i N(u_{i+1}) \quad + \theta \Delta t_i B z_{i+1} + \\ (-M + (1-\theta) \Delta t_i A) u_i + (1-\theta) \Delta t_i N(u_i) &+ (1-\theta) \Delta t_i B z_i \quad = 0 \end{aligned}$$

Discontinuous-in-time, decoupled control

$$\underset{u,z}{\text{Minimize}} \quad \sum_{i=1}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} u_i^T M u_i + g(t_i)^T u_i \right) + \sum_{i=0}^{N} \Delta t_i \left(\frac{\alpha}{2} z_i^T Q z_i \right)$$

subject to

$$\begin{aligned} (M + \theta \Delta t_i A) u_{i+1} \quad &+ \theta \Delta t_i N(u_{i+1}) \quad + \\ (-M + (1-\theta)\Delta t_i A) u_i + (1-\theta)\Delta t_i N(u_i) + \Delta t_i B z_i \quad &= 0 \end{aligned}$$

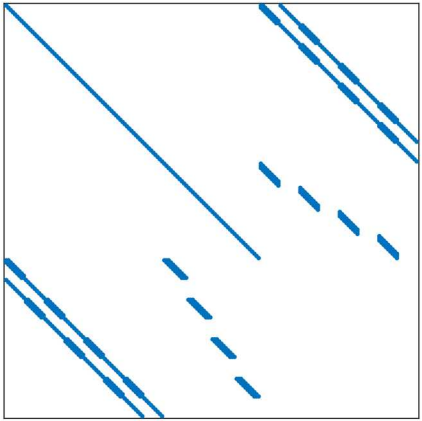
Virtual state variables + coupling constraint

$$\underset{u,z}{\text{Minimize}} \quad \sum_{i=1}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} (u_i + v_i)^T M (u_i + v_i) + g(t_i)^T (u_i + v_i) \right) + \sum_{i=0}^{N} \Delta t_i \left(\alpha z_i^T Q z_i \right)$$

subject to

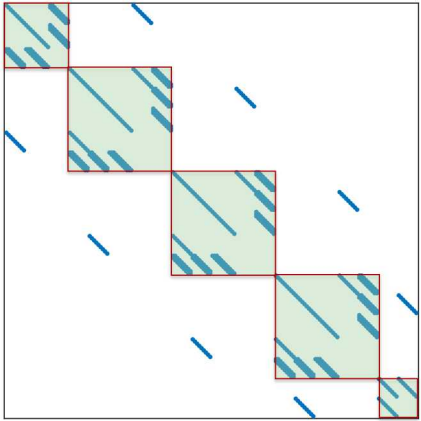
$$\begin{aligned} (M + \theta \Delta t_i A) u_{i+1} \quad + \theta \Delta t_i N(u_{i+1}) \quad + \\ (-M + (1-\theta) \Delta t_i A) v_i + (1-\theta) \Delta t_i N(v_i) + \Delta t_i B z_i \quad &= 0 \\ W(u_i - v_i) \quad &= 0 \end{aligned}$$

The augmented system



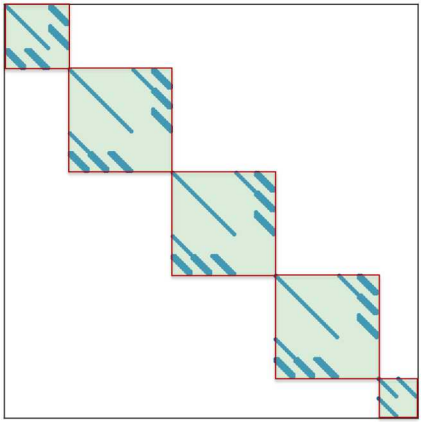
Ordering: $u_1\ v_1 \quad u_2\ v_2 \quad \ldots \quad z_0\ z_1 \quad \ldots \quad \lambda_1\ \mu_1 \quad \lambda_2\ \mu_2 \quad \ldots$

Weakly coupled optimality systems

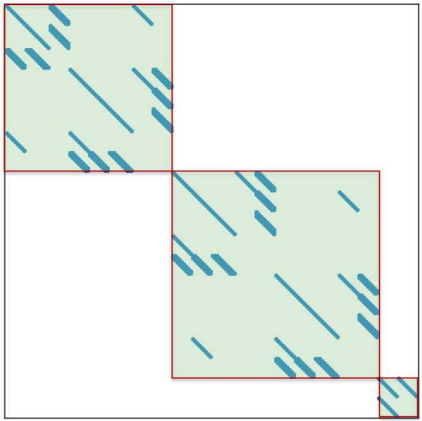


Transform: $u_1\ z_0\ \lambda_1 \quad v_1\ u_2\ z_1\ \mu_1\ \lambda_2 \quad \ldots \quad v_2\ u_3\ z_2\ \mu_2\ \lambda_3 \quad \ldots$

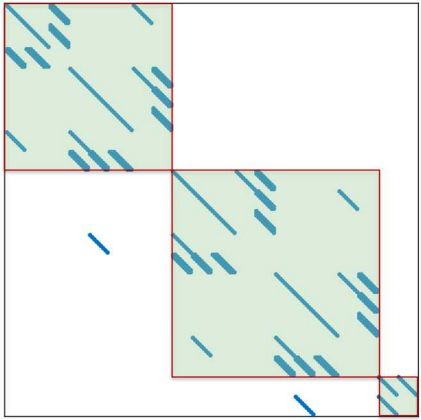
One-timestep Jacobi



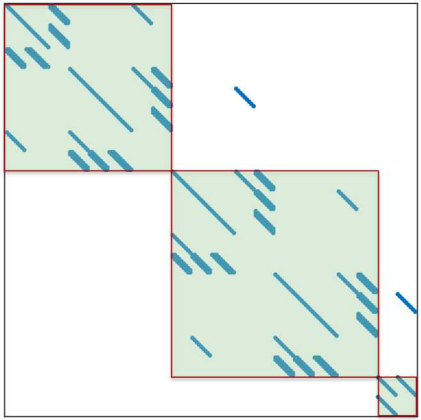
Two-timestep Jacobi



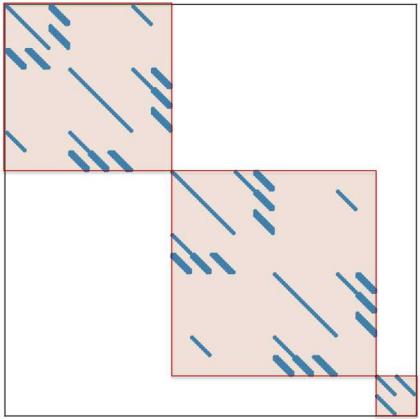
Two-timestep backward Gauss-Seidel



Two-timestep forward Gauss-Seidel



Some properties



Theorem

- *The diagonal blocks can be interpreted as optimality systems for optimization problems on the time subdomains.*
- *The subdomain optimization problems are convex QPs, regardless of the constraint equation or the objective function.*
- *If the constraint Jacobian is surjective, the subdomain QPs are strictly convex.*
- *Under this (weak) assumption, the diagonal blocks are invertible.*

Performance as preconditioners for GMRES

- Jacobi

#ts/subdomain	1	2	4	8
LS Iter	153.2	72.7	37.3	20.3
CG Iter	22	21	22	16
Opt Iter	10	10	10	5
Speeedup	0.65	0.69	0.66	0.62

- Backward Gauss-Seidel*

#ts/subdomain	1	2	4	8
LS Iter	134.3	59.3	31.3	16.8
CG Iter	18	17	14	14
Opt Iter	3	3	3	3
Speeedup	0.74	0.84	0.80	0.74

- Forward Gauss-Seidel*

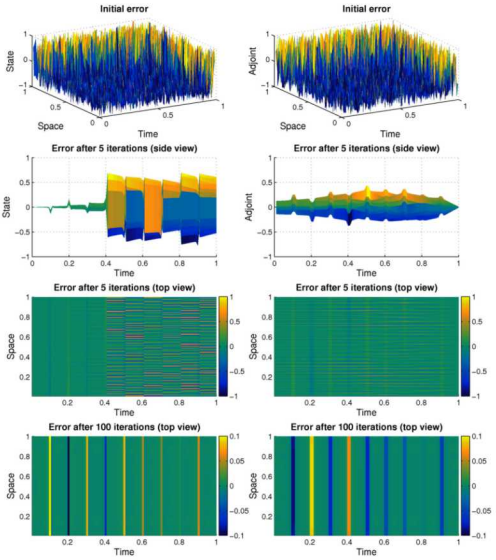
#ts/subdomain	1	2	4	8
LS Iter	69.7	33.1	17.4	8.3
CG Iter	14	10	9	16
Opt Iter	7	5	5	5
Speeedup	1.43	1.51	1.44	1.50

– #ts = 100, opt tol = 10^{-4}

– LS Iters averaged over the entire SQP run

- In principle, the approach works, however, ...
- There is no significant speedup based on fixed- point iterations alone.
- Coarse-grid correction?
- I.e., are these methods good smoothers for a multigrid scheme?

Numerical evidence for good time-domain smoothers



- Start with random right-hand side.
- Apply smoother iteratively and compute error.
- Observation: The error is confined to the time subdomain interfaces!

- Hajghassem, Cyr, Ridzal, A Time-Parallel Method for the Solution of PDE-Constrained Optimization Problems, in *Center for Computing Research Summer Proceedings 2015*, Technical Report SAND2016-0830R, Sandia National Laboratories, 2015, pp. 79–92.

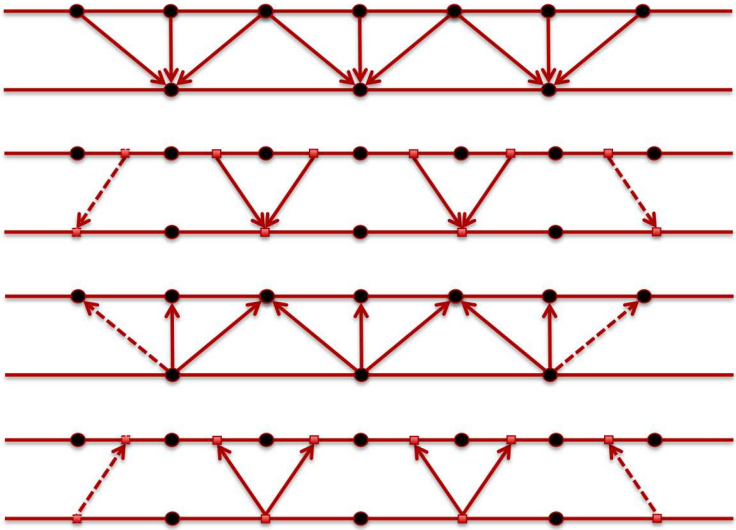
Two-grid approach

- Applying as a preconditioner for GMRES.
- For a fine grid in time (t), we want to approximate the action of K_t^{-1} on a vector v, using:
  - The solution with a coarse grid in time (T) operator K_T;
  - A smoother J^{-1}; and
  - Appropriate restriction and prolongation operators, R and P, resp.

1. Pre-smooth: $x_1 = J^{-1}v$.
2. Solve: $K_T e_T = R(v - K_t x_1)$.
3. Correct: $x_2 = x_1 + Pe_T$.
4. Post-smooth: $x_3 = x_2 + J^{-1}(v - K_t x_2)$.
5. Return x_3.

Restriction and prolongation operators

- For states and adjoints, we define R as a 3-point average.
- For controls, we define R as a 2-interval average.
- For states and adjoints, we define P via linear interpolation.
- For controls, we define P via injection (copy).



Two-grid results

- Using one-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	13.6	14.8	14.1	15.9
CG Iter	41	30	30	28
Opt Iter	6	5	6	6
Speeedup	**7.35**	**13.5**	**28.4**	**50.3**

- Using two-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	8.7	9.4	10.6	12.7
CG Iter	27	50	32	9
Opt Iter	4	6	7	4
Speeedup	**5.75**	**10.6**	**18.9**	**31.5**

- Using four-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	9.2	10.4	10.9	11.3
CG Iter	67	15	9	10
Opt Iter	7	4	3	4
Speeedup	**2.72**	**4.85**	**9.17**	**17.7**

Speedup =
#timesteps /
#ts-per-subdomain /
#LS-iters /
2 (pre/post Jacobi)

Two-grid results

- Using one-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	13.6	14.8	14.1	15.9
CG Iter	41	30	30	28
Opt Iter	6	5	6	6
Speeedup	**7.35**	**13.5**	**28.4**	**50.3**

- Using two-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	8.7	9.4	10.6	12.7
CG Iter	27	50	32	9
Opt Iter	4	6	7	4
Speeedup	**5.75**	**10.6**	**18.9**	**31.5**

- Using four-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	9.2	10.4	10.9	11.3
CG Iter	67	15	9	10
Opt Iter	7	4	3	4
Speeedup	**2.72**	**4.85**	**9.17**	**17.7**

Speedup =
#timesteps /
#ts-per-subdomain /
#LS-iters /
2 (pre/post Jacobi)

- **Raw speedup:**
  50x or better!

Two-grid results

- Using one-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	13.6	14.8	14.1	15.9
CG Iter	41	30	30	28
Opt Iter	6	5	6	6
Speeedup	**7.35**	**13.5**	**28.4**	**50.3**

- Using two-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	8.7	9.4	10.6	12.7
CG Iter	27	50	32	9
Opt Iter	4	6	7	4
Speeedup	**5.75**	**10.6**	**18.9**	**31.5**

- Using four-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	9.2	10.4	10.9	11.3
CG Iter	67	15	9	10
Opt Iter	7	4	3	4
Speeedup	**2.72**	**4.85**	**9.17**	**17.7**

Speedup =
#timesteps /
#ts-per-subdomain /
#LS-iters /
2 (pre/post Jacobi)

- **Raw speedup: 50x or better!**
- **Fix 200 subdomains: About 10x, scales.**

Two-grid results

- Using one-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	13.6	14.8	14.1	15.9
CG Iter	41	30	30	28
Opt Iter	6	5	6	6
Speeedup	**7.35**	**13.5**	**28.4**	**50.3**

- Using two-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	8.7	9.4	10.6	12.7
CG Iter	27	50	32	9
Opt Iter	4	6	7	4
Speeedup	**5.75**	**10.6**	**18.9**	**31.5**

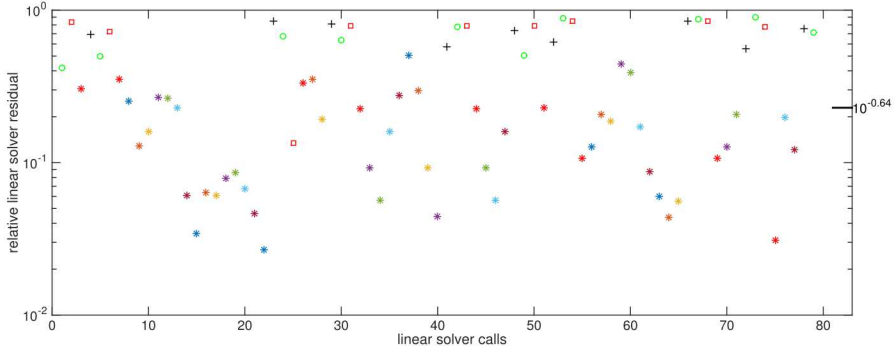
- Using four-timestep Jacobi as the smoother:

# time steps	200	400	800	1600
LS Iter	9.2	10.4	10.9	11.3
CG Iter	67	15	9	10
Opt Iter	7	4	3	4
Speeedup	**2.72**	**4.85**	**9.17**	**17.7**

Speedup =
#timesteps /
#ts-per-subdomain /
#LS-iters /
2 (pre/post Jacobi)

- **Raw speedup: 50x or better!**
- **Fix 200 subdomains: About 10x, scales.**
- **Fix 400 subdomains: About 20x, scales.**

Very coarse solver tolerances



- □ red squares denote quasi-normal step computations
- * red asterisks correspond to the first STCG iterations
- * blue asterisks denote the subsequent STCG iterations
- \+ black crosses correspond to tangential step computations
- ○ green circles denote Lagrange multiplier computations

Very coarse solver tolerances are important!

- Using two-timestep Jacobi, max tol $= 0.9$:

# time steps	200	400	800	1600
LS Iter	8.7	9.4	10.6	12.7
CG Iter	27	50	32	9
Opt Iter	4	6	7	4
Speeedup	**5.75**	**10.6**	**18.9**	**31.5**

- Using two-timestep Jacobi, max tol $= 10^{-2}$:

# time steps	200	400	800	1600
LS Iter	21.0	22.2	22.1	22.0
CG Iter	14	14	14	14
Opt Iter	3	3	3	3
Speeedup	**2.38**	**4.50**	**9.05**	**18.2**

- Using two-timestep Jacobi, max tol $= 10^{-8}$:

# time steps	200	400	800	1600
LS Iter	53.8	58.6	64.0	64.0
CG Iter	14	14	14	13
Opt Iter	3	3	3	3
Speeedup	**0.93**	**1.71**	**3.13**	**6.16**

- Tighter tolerances may give slightly better convergence of the optimization loop.
- Typically not needed.
- We observe fast superlinear convergence even with max tol $= 0.9$.
- Keep the linear solver tolerances coarse!

Future work

- Improve two-grid scheme; extend two-grid to multigrid.
- **Research 1:** Theoretical understanding of the scaling.
- **Research 2:** Prolongation and restriction operators.
- **Research 3:** Smoother improvements, inexact smoothers.
- **Research 4:** Subdomain / coarse-grid KKT solvers.
  - Serial Schur-complement approaches (Rees/Dollar/Wathen/Stoll) appear to work well; a handful of iterations of forward/adjoint solves.
- **Research 5:** Balancing storage requirements with raw speedup.
  - Amortizing precomputations; memory hierarchies, memory access; compression and checkpointing.
  - Developing a comprehensive performance model.
- Implementation in the Rapid Optimization Library (ROL).
- More interesting physics: Navier-Stokes, coupled fluid/Maxwell's systems, DAE systems such as power networks and electric circuits.