Outline

1. Motivating Applications
2. General Problem Formulation
3. Randomize Sketching for State Compression
4. Fixed Rank Analysis
5. Adaptive Rank Algorithm
6. Numerical Results
1. **Goal:** Determine injection rates that maximize the net present value of the reservoir.

2. **Problem Size:** The reservoir can span 10+ km per side \( \implies \) 1000km\(^3\)
   A regular grid with 5m\(\times\)5m\(\times\)5m cells \(\implies\) 8 billion cells!

3. **Similar Applications:** Gas pipeline and energy network operations, financial portfolio optimization, optimal control of hypersonic vehicles, etc.

\[
\begin{align*}
\text{max } & \text{ NPV}(v, p, s, q) \\
\text{subject to } & v = -K\lambda(s)(\nabla p - \rho G) \\
& \nabla \cdot v = q/\rho \\
& \phi \partial_t s + \nabla \cdot (f(s)v) = q
\end{align*}
\]
**Optimal Design**

**Convolute Design for the Z-Machine (Inertial Confinement Fusion)**

**Goal:** Minimize power loss – current design experiences 20% power loss

**Similar Applications:** Designing elastic structures with topology optimization, shape optimization for airplane wing design, etc.
1. **Goal:** Determine subsurface rock properties that minimize data misfit.

2. **Problem Size:** Typical marine surveys require many TBs of memory!

3. **Similar Applications:** Parameter calibration, data assimilation, supervised learning for neural networks, etc.

\[
\min_{u,\theta} \frac{1}{2} \|u - d\|^2 + \alpha \varphi(\theta)
\]

subject to

\[
\partial_t (\varphi(\theta) \partial_t u) - \nabla \cdot (K(\theta) : \varepsilon) = s
\]

\[
\varepsilon = \frac{1}{2} (\nabla u + \nabla u^\top)
\]
Discrete Dynamic Optimization

**Full-Space Formulation**

\[
\min_{z_n \in \mathbb{R}^m, \ u_n \in \mathbb{R}^M} \sum_{n=1}^{N} f_n(u_{n-1}, u_n, z_n)
\]

subject to \( c_n(u_{n-1}, u_n, z_n) = 0, \ n = 1, \ldots, N \)

where \( f_n : \mathbb{R}^M \times \mathbb{R}^M \times \mathbb{R}^m \rightarrow \mathbb{R} \) is the _cost function at time_ \( t_n \),

\( c_n : \mathbb{R}^M \times \mathbb{R}^M \times \mathbb{R}^m \rightarrow \mathbb{R}^M \) is the _dynamic constraint at time_ \( t_n \),

\( u_n \in \mathbb{R}^M \) is the _state/trajecotry at time_ \( t_n \), and

\( z_n \in \mathbb{R}^m \) is the _control/design/parameters at time_ \( t_n \).

E.g., continuous dynamic problem discretized in time (and space).

Our methods also work for time-independent control/design/parameters.
Discrete Dynamic Optimization

Optimality Conditions

The KKT conditions for the full-space dynamic optimization problem are

\[ c_n(u_{n-1}, u_n, z_n) = 0 \]

\[ (d_2 c_N(u_{N-1}, u_N, z_N))^\top \lambda_N = -d_2 f_N(u_{N-1}, u_N, z_N) \]

\[ (d_2 c_n(u_{n-1}, u_n, z_n))^\top \lambda_n + (d_1 c_{n+1}(u_n, u_{n+1}, z_{n+1}))^\top \lambda_{n+1} = -d_2 f_N(u_{n-1}, u_n, z_n) - d_1 f_{n+1}(u_n, u_{n+1}, z_{n+1}) \]

\[ d_3 f_n(u_{n-1}, u_n, z_n) + (d_3 c_n(u_{n-1}, u_n, z_n))^\top \lambda_n = 0, \quad n = 1, \ldots, N. \]

The new variables \( \lambda_1, \ldots, \lambda_N \) are Lagrange multipliers for the dynamic constraint.

Most algorithms (e.g., SQP, AL, etc.) require storage of \( \{u_n\}_{n=1}^N \), \( \{\lambda_n\}_{n=1}^N \) and \( \{z_n\}_{n=1}^N \).

Storage Requirement: \( \mathcal{O}(N(2M + m)) \) floating point numbers

\[ \Longrightarrow \text{Can be huge (e.g., TBs of memory)!} \]
Discrete Dynamic Optimization

Reduced-Space Formulation

If \( c_n(u_{n-1}, u_n, z_n) = 0 \) has a unique solution \( u_n = S_n(u_{n-1}, z_n) \) for each \( z \in Z_{ad} \) and fixed initial state \( u_0 \in \mathbb{R}^M, n = 1, \ldots, N \), then we can equivalently solve

\[
\min_{Z = \{z_n\}_{n=1}^N \in \mathbb{R}^{mN}} F(Z)
\]

where the reduced objective function \( F \) is given by

\[
F(Z) := f_1(u_0, S_1(u_0, z_1), z_1) + \sum_{n=2}^{N} f_n(S_{n-1}(u_{n-2}, z_{n-1}), S_n(u_{n-1}, z_n), z_n)
\]

Storage Requirement: \( \mathcal{O}(Nm) \) floating point numbers — typically, \( m \ll M \).
Gradient Computation:

\[
(\nabla F(Z))_n = d_3f_n(u_{n-1}, u_n, z_n) + (d_3c_n(u_{n-1}, u_n, z_n))^\top \lambda_n
\]

where \( \lambda_n \in \mathbb{R}^M \), \( n = 1, \ldots, N \) solves the \textit{backward-in-time} adjoint equation

\[
(d_2c_N(u_{N-1}, u_N, z_N))^\top \lambda_N = -d_2f_N(u_{N-1}, u_N, z_N)
\]

\[
(d_2c_n(u_{n-1}, u_n, z_n))^\top \lambda_n = -d_2f_N(u_{n-1}, u_n, z_n) - d_1f_{n+1}(u_n, u_{n+1}, z_{n+1})
- (d_1c_{n+1}(u_n, u_{n+1}, z_{n+1}))^\top \lambda_{n+1}
\]

Gradient computations requires entire state trajectory to solve adjoint equation!
1. **Checkpointing**: Store a handful of state snapshots in memory or to hard disk. Recompute state from snapshots during adjoint computation.


   **Significant increase in computational cost!**
1. **Checkpointing:** Store a handful of state snapshots in memory or to hard disk. Recompute state from snapshots during adjoint computation.


   **Significant increase in computational cost!**

2. **Model Reduction:** Project dynamic equation onto a low-order basis of the solution space (e.g., balanced truncation, POD, reduced basis, etc.).


   **Difficult for nonlinear dynamics and requires significant code modification!**
1. **Checkpointing:** Store a handful of state snapshots in memory or to hard disk. Recompute state from snapshots during adjoint computation.


**Significant increase in computational cost!**

2. **Model Reduction:** Project dynamic equation onto a low-order basis of the solution space (e.g., balanced truncation, POD, reduced basis, etc.).


**Difficult for nonlinear dynamics and requires significant code modification!**

3. **State Compression:** Compress dynamic trajectory using numerical compression techniques including nested meshes, PCA, Gramm-Schmidt and DFT.


**Can be physics-dependent and often compresses time/space independently!**
Randomized Sketching

We apply randomized sketching to compress the $M \times N$ state trajectory matrix

$$U = (u_1 \mid \cdots \mid u_N) \in \mathbb{R}^{M \times N}.$$ 

**Goal:** Generate an accurate rank-$r$ approximation of $U$ using $O(r(M + N))$ storage.

**Matrix Sketching:** Given the four random linear dimension reduction maps

$$\Upsilon \in \mathbb{R}^{k \times M}, \quad \Omega \in \mathbb{R}^{k \times N}, \quad \Phi \in \mathbb{R}^{s \times M}, \quad \text{and} \quad \Psi \in \mathbb{R}^{s \times N},$$

where $k := 2r + 1$ and $s = 2k + 1$, we define the sketch as the following three matrices

$$X := \Upsilon U \in \mathbb{R}^{k \times N} \quad \text{the co-range sketch;}$$

$$Y := U \Omega^T \in \mathbb{R}^{M \times k} \quad \text{the range sketch;}$$

$$Z := \Phi U \Psi^T \in \mathbb{R}^{s \times s} \quad \text{the core sketch.}$$

Randomized Sketching

Details and Intuition

Choice of Random Matrices: Standard normal, scrambled subsampled randomized Fourier transform, sparse sign, . . .

Intuition:

- The range sketch $Y = U\Omega^\top$ consists of columns $U\omega_i$, which are independent random samples from range($U$).

  **Roughly speaking**, $Y$ captures the action of the top $k$ left singular vectors.

- Similarly, the co-range sketch $X = \Upsilon U$ consists of rows $v_i^\top U$, which are independent random samples from range($U^\top$).

  **Roughly speaking**, $X$ captures the action of the top $k$ right singular vectors.

- **Roughly speaking**, the core sketch $Z = \Phi U\Psi^\top$ captures the top $k$ singular values.
Online State Sketching:

**Require:** $X = 0, Y = 0, Z = 0$

1. for $n=1, \ldots, N$ do
2. Given $u_{n-1}$ and $z_n$, solve $c_n(u_{n-1}, u_n, z_n) = 0$ for $u_n$
3. Update $X \leftarrow X + \gamma u_n e_n^T$
4. Update $Y \leftarrow Y + u_n (\Omega e_n)^T$
5. Update $Z \leftarrow Z + (\Phi u_n)(\Psi e_n)^T$
6. end for

**Storage Requirement:** Sketching requires $k(M + N) + s^2$ floating point numbers.
Randomized Sketching

Online State Sketching:

Require: \( X = 0, Y = 0, Z = 0 \)

1. **for** \( n=1, \ldots, N \) **do**
2. Given \( u_{n-1} \) and \( z_n \), solve \( c_n(u_{n-1}, u_n, z_n) = 0 \) for \( u_n \)
3. Update \( X \leftarrow X + \gamma u_ne_n^T \)
4. Update \( Y \leftarrow Y + u_n(\Omega e_n)^T \)
5. Update \( Z \leftarrow Z + (\Phi u_n)(\Psi e_n)^T \)
6. **end for**

**Storage Requirement:** Sketching requires \( k(M + N) + s^2 \) floating point numbers.

Recovery:

1. \( (Q, R_2) \leftarrow qr(Y, 0) \)
2. \( (P, R_1) \leftarrow qr(X^T, 0) \)
3. \( C \leftarrow (\Phi Q)^\dagger Z((\Psi P)^\dagger)^T \)
4. \( W \leftarrow CP^T \)
5. \( u_n \approx Q We_n \)

**Storage Requirement:** Recovery requires \( k(M + N) + k^2 \) floating point numbers.
**Theorem: Sketching Error Bound**

Let $\{\{U\}\}_r$ denote the sketch of $U$ associated with the rank parameter $r$. Then,

$$
\mathbb{E}_{\gamma, \Omega, \Phi, \Psi} \|U - \{\{U\}\}_r\|_F \leq \sqrt{6} \left( \sum_{i \geq r+1} \sigma_i^2(U) \right)^{\frac{1}{2}}.
$$

The error is *expected* to be slightly larger than the best rank-$r$ approximation!

Similar results exist for the probability of large deviation.

Solve Adjoint Equation and Compute Gradient:

Require: Sketched state \( \{\{U\}\}_r \).

1: Reconstruct \( \tilde{u}_{N-1} \) and \( \tilde{u}_N \) from \( \{\{U\}\}_r \).
2: Compute \( \tilde{\lambda}_N \) that solves \((d_2c_N(\tilde{u}_{N-1}, \tilde{u}_N, z_N))^\top \tilde{\lambda}_N = -d_2f_N(\tilde{u}_{N-1}, \tilde{u}_N, z_N)\).
3: Set \( g_N = d_3f_N(\tilde{u}_{N-1}, \tilde{u}_N, z_N) + (d_3c_N(\tilde{u}_{N-1}, \tilde{u}_N, z_N))^\top \tilde{\lambda}_N \).
4: for \( n = N - 1, \ldots, 1 \) do
5: Reconstruct \( \tilde{u}_{n-1} \) from \( \{\{U\}\}_r \).
6: Compute \( \tilde{\lambda}_n \) that solves

\[
(d_2c_n(\tilde{u}_{n-1}, \tilde{u}_n, z_n))^\top \tilde{\lambda}_n = -d_2f_n(\tilde{u}_{n-1}, \tilde{u}_n, z_n) - d_1f_{n+1}(\tilde{u}_n, \tilde{u}_{n+1}, z_{n+1}) \\
- (d_1c_{n+1}(\tilde{u}_n, \tilde{u}_{n+1}, z_{n+1}))^\top \tilde{\lambda}_{n+1}.
\]
7: Set \( g_n = d_3f_n(\tilde{u}_{n-1}, \tilde{u}_n, z_n) + (d_3c_n(\tilde{u}_{n-1}, \tilde{u}_n, z_n))^\top \tilde{\lambda}_n \).
8: end for
Randomized Sketching

Define the combined objective function and constraint by

\[ f(U, Z) = \sum_{i=1}^{N} f_i(u_{i-1}, u_i, z_i) \quad \text{and} \quad c(U, Z) = \begin{pmatrix} c_1(u_0, u_1, z_1) \\ \vdots \\ c_N(u_{N-1}, u_N, z_N) \end{pmatrix}. \]

Assumptions:

1. For any bounded set \( Z \subseteq \mathbb{R}^{Nm} \), the set \( U = \{u \in \mathbb{R}^{Nm} : c(U, Z) = 0\} \) is bounded;
2. There exist \( 0 < \beta < 1 \) such that for all \( U, Z \)

\[ \min \{d_1c(U, Z)\} \leq \beta \max \{d_1c(U, Z)\} < 1; \]
3. The functions \( d_1c(U, Z) \), \( d_2c(U, Z) \), \( d_1f(U, Z) \) and \( d_2f(U, Z) \) are Lipschitz continuous on \( UZ \) with respect to \( U \) and the Lipschitz modulii are independent of \( Z \).

Under these assumptions, we can show that

\[ \|g_f(U, Z)\| \leq \|c(U, Z)\| \text{ and } \|E_g\| \leq X. \]
Randomized Sketching

Define the combined objective function and constraint by

$$f(U, Z) = \sum_{i=1}^{N} f_i(u_{i-1}, u_i, z_i) \quad \text{and} \quad c(U, Z) = \begin{pmatrix} c_1(u_0, u_1, z_1) \\ \vdots \\ c_N(u_{N-1}, u_N, z_N) \end{pmatrix}.$$ 

Assumptions:

1. For any bounded set $\mathcal{Z} \subset \mathbb{R}^{Nm}$, the set $\mathcal{U} = \{U \mid c(U, Z) = 0, Z \in \mathcal{Z}\}$ is bounded;
Randomized Sketching  Assumptions and Gradient Error

Define the combined objective function and constraint by

\[ f(U, Z) = \sum_{i=1}^{N} f_i(u_{i-1}, u_i, z_i) \quad \text{and} \quad c(U, Z) = \begin{pmatrix}
  c_1(u_0, u_1, z_1) \\
  \vdots \\
  c_N(u_{N-1}, u_N, z_N)
\end{pmatrix}. \]

**Assumptions:**

1. For any bounded set \( Z \subset \mathbb{R}^{Nm} \), the set \( U = \{ U \mid c(U, Z) = 0, \ Z \in Z \} \) is bounded;

2. There exists \( 0 < \sigma_0 \leq \sigma_1 < \infty \) such that for all \( U \in U \) and \( Z \in Z \)

\[ \sigma_0 \leq \sigma_{\min}(d_1 c(U, Z)) \leq \sigma_{\max}(d_1 c(U, Z)) \leq \sigma_1; \]
Define the combined objective function and constraint by

\[ f(U, Z) = \sum_{i=1}^{N} f_i(u_{i-1}, u_i, z_i) \quad \text{and} \quad c(U, Z) = \begin{pmatrix} c_1(u_0, u_1, z_1) \\ \vdots \\ c_N(u_{N-1}, u_N, z_N) \end{pmatrix}. \]

**Assumptions:**

1. For any bounded set \( Z \subset \mathbb{R}^{Nm} \), the set \( U = \{ U \mid c(U, Z) = 0, \ Z \in Z \} \) is bounded;

2. There exists \( 0 < \sigma_0 \leq \sigma_1 < \infty \) such that for all \( U \in U \) and \( Z \in Z \)

\[ \sigma_0 \leq \sigma_{\min}(d_1 c(U, Z)) \leq \sigma_{\max}(d_1 c(U, Z)) \leq \sigma_1; \]

3. The functions \( d_1 c(U, Z) \), \( d_2 c(U, Z) \), \( d_1 f(U, Z) \) and \( d_2 f(U, Z) \) are Lipschitz continuous on \( U \times Z \) with respect to \( U \) and the Lipschitz modulii are independent of \( Z \in Z \).
Define the combined objective function and constraint by

\[ f(U, Z) = \sum_{i=1}^{N} f_i(u_{i-1}, u_i, z_i) \quad \text{and} \quad c(U, Z) = \begin{pmatrix} c_1(u_0, u_1, z_1) \\ \vdots \\ c_N(u_{N-1}, u_N, z_N) \end{pmatrix}. \]

**Assumptions:**

1. For any bounded set \( Z \subset \mathbb{R}^{Nm} \), the set \( U = \{ U \mid c(U, Z) = 0, \ Z \in Z \} \) is bounded;
2. There exists \( 0 < \sigma_0 \leq \sigma_1 < \infty \) such that for all \( U \in \mathcal{U} \) and \( Z \in Z \)
   \[ \sigma_0 \leq \sigma_{\min}(d_1 c(U, Z)) \leq \sigma_{\max}(d_1 c(U, Z)) \leq \sigma_1; \]
3. The functions \( d_1 c(U, Z) \), \( d_2 c(U, Z) \), \( d_1 f(U, Z) \) and \( d_2 f(U, Z) \) are Lipschitz continuous on \( \mathcal{U} \times Z \) with respect to \( U \) and the Lipschitz modulii are independent of \( Z \in Z \).

**Under these assumptions, we can show that**

\[ \| g - \nabla F(Z) \| \leq \kappa_1 \| c(\{\{U\}\}_r, Z) \| \quad \text{and} \quad \mathbb{E}\| g - \nabla F(Z) \| \leq \sqrt{6\kappa} \left( \sum_{i \geq r+1} \sigma_i^2(U) \right)^{\frac{1}{2}}. \]
Fixed Rank Approach

- **Idea:** Use approximate gradient within a derivative-based optimization algorithm. For example, gradient descent, nonlinear CG, Newton–Krylov method, etc.

- If the rank parameter $r$ is sufficiently large, then gradient error may be negligible.

- For Newton–Krylov, applying the Hessian to a vector requires a linearized forward-in-time solve and a linearized backward-in-time solve.

- Can approximately apply the Hessian to a vector by sketching the adjoint as well as the trajectories from the additional solves.
Fixed Rank Approach

Basic Trust-Region Method

Require: Initial guess $Z^{(0)} \in \mathbb{R}^{Nm}$ and target rank $r$. Algorithmic parameters: $\Delta^{(0)} > 0$ and $0 < \eta_0 \leq \eta_1 < \eta_2 < 1$.

1: for $\ell = 0, 1, 2, \ldots$ do
2:    Solve state equation and sketch solution $\{U^{(\ell)}\}_r$.
3:    Compute gradient approximation $g^{(\ell)} \approx \nabla F(Z^{(\ell)})$ using sketched state $\{U^{(\ell)}\}_r$.
4:    Choose $B^{(\ell)} \in \mathbb{R}^{Nm \times Nm}$ to be an approximation of the Hessian $\nabla^2 F(Z^{(\ell)})$.
5:    Compute $S^{(\ell)} \in \mathbb{R}^{Nm}$ that approximately solves the trust-region subproblem
       $$\min_{S \in \mathbb{R}^{Nm}} \frac{1}{2} \langle S, B^{(\ell)} S \rangle + \langle S, g^{(\ell)} \rangle \quad \text{subject to} \quad \|S\| \leq \Delta^{(\ell)}.$$ 
6:    Compute the ratio of actual and predicted reduction
       $$\rho^{(\ell)} = \frac{F(Z^{(\ell)}) - F(Z^{(\ell)} + S^{(\ell)})}{-\frac{1}{2} \langle S^{(\ell)}, B^{(\ell)} S^{(\ell)} \rangle - \langle S^{(\ell)}, g^{(\ell)} \rangle}.$$ 
7:    Set $Z^{(\ell+1)} = \begin{cases} 
Z^{(\ell)} & \text{if } \rho^{(\ell)} \leq \eta_0 \\
Z^{(\ell)} + S^{(\ell)} & \text{otherwise}. 
\end{cases}$
8:    Choose $\Delta^{(\ell+1)} \in \begin{cases} 
(0, \Delta^{(\ell)}) & \text{if } \rho^{(\ell)} \leq \eta_1 \\
\{\Delta^{(\ell)}\} & \text{if } \rho^{(\ell)} \in (\eta_1, \eta_2) \\
(\Delta^{(\ell)}, \infty) & \text{otherwise}. 
\end{cases}$
9: end for
Fixed Rank Approach

Basic Trust-Region Method

**Require:** Initial guess $Z^{(0)} \in \mathbb{R}^{Nm}$ and target rank $r$. Algorithmic parameters: $\Delta^{(0)} > 0$ and $0 < \eta_0 \leq \eta_1 < \eta_2 < 1$.

1: \textbf{for } $\ell = 0, 1, 2, \ldots$ \textbf{do}
2: \hspace{1em} Solve state equation and sketch solution $\{U^{(\ell)}\}_r$.
3: \hspace{1em} **Compute gradient approximation** $g^{(\ell)} \approx \nabla F(Z^{(\ell)})$ using sketched state $\{U^{(\ell)}\}_r$.
4: \hspace{1em} Choose $B^{(\ell)} \in \mathbb{R}^{Nm \times Nm}$ to be an approximation of the Hessian $\nabla^2 F(Z^{(\ell)})$.
5: \hspace{1em} Compute $S^{(\ell)} \in \mathbb{R}^{Nm}$ that approximately solves the trust-region subproblem

$$\min_{S \in \mathbb{R}^{Nm}} \frac{1}{2} \langle S, B^{(\ell)} S \rangle + \langle S, g^{(\ell)} \rangle \quad \text{subject to} \quad ||S|| \leq \Delta^{(\ell)}.$$  

6: \hspace{1em} Compute the ratio of actual and predicted reduction

$$\rho^{(\ell)} = \frac{F(Z^{(\ell)}) - F(Z^{(\ell)} + S^{(\ell)})}{-\frac{1}{2} \langle S^{(\ell)}, B^{(\ell)} S^{(\ell)} \rangle - \langle S^{(\ell)}, g^{(\ell)} \rangle}.$$  

7: \hspace{1em} Set $Z^{(\ell+1)} = \begin{cases} Z^{(\ell)} & \text{if } \rho^{(\ell)} \leq \eta_0 \\ Z^{(\ell)} + S^{(\ell)} & \text{otherwise.} \end{cases}$

8: \hspace{1em} Choose $\Delta^{(\ell+1)} \in \begin{cases} (0, \Delta^{(\ell)}) & \text{if } \rho^{(\ell)} \leq \eta_1 \\ \{\Delta^{(\ell)}\} & \text{if } \rho^{(\ell)} \in (\eta_1, \eta_2) \\ (\Delta^{(\ell)}, \infty) & \text{otherwise.} \end{cases}$

9: \textbf{end for}

**Algorithm may not converge because gradient is inexact!**
Adaptive Rank Algorithm

1. Fixed-rank algorithm may not converge unless the rank $r$ is chosen sufficiently large. **A priori choice of $r$ is often difficult for practical applications!**

Inexact Trust-Region Method

...
Adaptive Rank Algorithm

1. Fixed-rank algorithm may not converge unless the rank $r$ is chosen sufficiently large. **A priori choice of $r$ is often difficult for practical applications!**

2. To ensure convergence of the TR method, gradient inexactness must be controlled

$$
\|g^{(\ell)} - \nabla F(Z^{(\ell)})\| \leq \kappa_0 \min\{\|g^{(\ell)}\|, \Delta^{(\ell)}\}.
$$

1. Fixed-rank algorithm may not converge unless the rank $r$ is chosen sufficiently large. 
   **A priori choice of $r$ is often difficult for practical applications!**

2. To ensure convergence of the TR method, gradient inexactness must be controlled

   $$
   \|g^{(\ell)} - \nabla F(Z^{(\ell)})\| \leq \kappa_0 \min\{\|g^{(\ell)}\|, \Delta^{(\ell)}\}.
   $$


3. Satisfy gradient inexactness condition by monitoring state residual

   $$
   \|g^{(\ell)} - \nabla F(Z^{(\ell)})\| \leq \kappa_1 c(\tilde{U}^{(\ell)}, Z^{(\ell)}) \quad \Rightarrow \quad c(\tilde{U}^{(\ell)}, Z^{(\ell)}) \leq \kappa'_0 \min\{\|g^{(\ell)}\|, \Delta^{(\ell)}\}.
   $$
Adaptive Rank Algorithm

1. Fixed-rank algorithm may not converge unless the rank $r$ is chosen sufficiently large.
   **A priori choice of $r$ is often difficult for practical applications!**

2. To ensure convergence of the TR method, gradient inexactness must be controlled

\[
\|g^{(\ell)} - \nabla F(Z^{(\ell)})\| \leq \kappa_0 \min\{\|g^{(\ell)}\|, \Delta^{(\ell)}\}.
\]


3. Satisfy gradient inexactness condition by monitoring state residual

\[
\|g^{(\ell)} - \nabla F(Z^{(\ell)})\| \leq \kappa_1 \|c(\tilde{U}^{(\ell)}, Z^{(\ell)})\| \quad \Rightarrow \quad \|c(\tilde{U}^{(\ell)}, Z^{(\ell)})\| \leq \kappa'_0 \min\{\|g^{(\ell)}\|, \Delta^{(\ell)}\}.
\]

4. Evaluate state residual while solving adjoint equation.
   **If bound is violated, increase $r$, re-solve state equation and re-sketch state!**
Adaptive Rank Algorithm

1. Fixed-rank algorithm may not converge unless the rank $r$ is chosen sufficiently large. A priori choice of $r$ is often difficult for practical applications!

2. To ensure convergence of the TR method, gradient inexactness must be controlled

$$\|g^{(\ell)} - \nabla F(Z^{(\ell)})\| \leq \kappa_0 \min\{\|g^{(\ell)}\|, \Delta^{(\ell)}\}.$$


3. Satisfy gradient inexactness condition by monitoring state residual

$$\left\|g^{(\ell)} - \nabla F(Z^{(\ell)})\right\| \leq \kappa_1 \left\|c(\tilde{U}^{(\ell)}, Z^{(\ell)})\right\| \quad \Rightarrow \quad \left\|c(\tilde{U}^{(\ell)}, Z^{(\ell)})\right\| \leq \kappa_0' \min\{\|g^{(\ell)}\|, \Delta^{(\ell)}\}.$$

4. Evaluate state residual while solving adjoint equation. If bound is violated, increase $r$, re-solve state equation and re-sketch state!

5. Hopefully sufficiently large $r$ is determined after a handful of updates. May run out of memory if rank of state is too large!
Theorem: Convergence of Adaptive-Rank Trust-Region Method

Suppose there exists a bounded, open, convex set $\mathcal{Z} \subset \mathbb{R}^{Nm}$ such that $\mathcal{Z}^{(\ell)} \in \mathcal{Z}$ for all $\ell$. Suppose $F$ is bounded below and twice continuously differentiable on $\mathcal{Z}$. Moreover, assume that $\nabla^2 F(\cdot)$ is uniformly bounded on $\mathcal{Z}$ and that $B^{(\ell)}$ is bounded independently of $\ell$. Then,

$$\liminf_{\ell \to \infty} ||g^{(\ell)}|| = \liminf_{\ell \to \infty} ||\nabla F(Z^{(\ell)})|| = 0.$$

For detailed analysis of inexact trust-region methods see:


Adaptive Rank Algorithm

"Simple" Control Constraints: Constraints with *easy-to-compute* projections such as

\[ a_0 \leq CZ \leq a_1 \quad \text{and} \quad DZ = b. \]

Only need to modify trust-region subproblem to account for constraints in the set \( Z_{ad} \), i.e.,

\[
\min_{S \in \mathbb{R}^{Nm}} \frac{1}{2} \langle S, B^{(\ell)} S \rangle + \langle S, g^{(\ell)} \rangle \quad \text{subject to} \quad Z^{(\ell)} + S \in Z_{ad}, \quad \|S\| \leq \Delta^{(\ell)}.
\]

Adaptive Rank Algorithm

"Simple" Control Constraints: Constraints with easy-to-compute projections such as

\[ a_0 \leq CZ \leq a_1 \quad \text{and} \quad DZ = b. \]

Only need to modify trust-region subproblem to account for constraints in the set \( Z_{\text{ad}} \), i.e.,

\[
\min_{S \in \mathbb{R}^{Nm}} \frac{1}{2} \langle S, B^{(\ell)} S \rangle + \langle S, g^{(\ell)} \rangle \quad \text{subject to} \quad Z^{(\ell)} + S \in Z_{\text{ad}}, \quad \|S\| \leq \Delta^{(\ell)}.
\]


General Constraints: Use augmented Lagrangian, log barrier, etc. to penalize general nonlinear constraints such as

\[ g(U, Z) = 0 \quad \text{and} \quad h(U, Z) \leq 0. \]

Apply the adaptive rank algorithm to solve the penalized subproblems

\[
\min_{Z \in \mathbb{R}^{Nm}} F(Z) + \phi^{(\ell)}(Z).
\]
Example Application  Linear Parabolic Control

Let $L$ be a uniformly bounded and coercive linear operator, $B$ a bounded linear operator and $\beta$ a continuous linear functional. Consider the optimal control problem

$$\min_{u, z} \frac{1}{2} \int_0^T \|u - w\|^2 \, dt + \frac{\alpha}{2} \int_0^T \|z\|^2 \, dt$$

subject to

$$\partial_t u + L(t)u = \beta(t) + Bz$$

$$u(0) = u_0.$$  

Discretization

1. Continuous finite elements in space and implicit Euler in time.
2. Similar results hold for trapezoidal (Crank-Nicolson) and explicit Euler.

Stability Estimates: $C_n^1$, $C_n^2$ are positive constants that depend on the time step $\delta t_n$

$$\|\tilde{u}_n - u_n\| \leq C_n^1 \sum_{i=1}^n \|c_i(\tilde{u}_{i-1}, \tilde{u}_i, z_i)\| \quad \text{and} \quad \|\tilde{\lambda}_n - \lambda_n\| \leq C_n^2 \sum_{i=1}^n \|c_i(\tilde{u}_{i-1}, \tilde{u}_i, z_i)\|.$$

Using these estimates, we can prove convergence of adaptive rank algorithm!
Numerical Results

Linear Parabolic Control

Let $D = (0, 0.5) \times (0, 0.2)$ with boundary $\partial D$ and consider the optimal control problem

$$\min_{u, z} \frac{1}{2} \int_0^T \int_D |u - 1|^2 \, dx \, dt + \frac{\alpha}{2} \int_0^T \int_D |z|^2 \, dx \, dt$$

subject to

$$\partial_t u - \gamma \Delta u + \mathbf{v} \cdot \nabla u + u = \beta + z$$

in $(0, T) \times D$

$$\gamma \nabla u \cdot n = 0$$

on $(0, T) \times \partial D$

$$u(0) = 0$$

in $D$,

where $\alpha = 10^{-4}$, $\gamma = 0.1$, $\mathbf{v}(x) = (7.5 - 2.5x_1, 2.5x_2)^T$, and

$$\beta(x) = \begin{cases} 
1 & \text{if } \|x - (0.1, 0.1)^T\| \leq 0.07 \\
0 & \text{otherwise.}
\end{cases}$$

Spatial Discretization: Q1 finite elements on a uniform mesh of $60 \times 20$ quadrilaterals.

Temporal Discretization: Implicit Euler with 500 equal time steps.
Sketching error averaged over 20 realizations and tail energy for the uncontrolled state (left) and the optimal state (right). Recall that the rank of the sketch is $k = 2r + 1$. 
Termination Criterion: $\|g^{(\ell)}\| \leq 10^{-7}$ or $\ell > 20$.

Rank Increase Function: $r \leftarrow \max\{r + 2, \lceil (b - \log \tau)/a \rceil \}$ with $a = 2.6125$, $b = 2.4841$.

<table>
<thead>
<tr>
<th>rank</th>
<th>objective</th>
<th>iter</th>
<th>nstate</th>
<th>nadjoint</th>
<th>itercG</th>
<th>compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>*1</td>
<td>5.544040e-4</td>
<td>20</td>
<td>21</td>
<td>11</td>
<td>196</td>
<td>118.79</td>
</tr>
<tr>
<td>2</td>
<td>5.528490e-4</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>151</td>
<td>70.96</td>
</tr>
<tr>
<td>3</td>
<td>5.528490e-4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>78</td>
<td>50.46</td>
</tr>
<tr>
<td>4</td>
<td>5.528490e-4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>67</td>
<td>38.08</td>
</tr>
<tr>
<td>5</td>
<td>5.528490e-4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>59</td>
<td>31.83</td>
</tr>
<tr>
<td>Adaptive</td>
<td>5.528490e-4</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>65</td>
<td>23.14</td>
</tr>
<tr>
<td>Full</td>
<td>5.528490e-4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>53</td>
<td>1.00</td>
</tr>
</tbody>
</table>

*The rank 1 experiment exceeded the maximum number of iterations.*
### Numerical Results
#### Linear Parabolic Control

<table>
<thead>
<tr>
<th></th>
<th>iter</th>
<th>value</th>
<th>gnorm</th>
<th>snorm</th>
<th>delta</th>
<th>iterCG</th>
<th>rank</th>
<th>rnorm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive</td>
<td>0</td>
<td>5.446e-2</td>
<td>5.990e-3</td>
<td>---</td>
<td>1.000e+1</td>
<td>---</td>
<td>1</td>
<td>2.707e-4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.375e-2</td>
<td>2.205e-3</td>
<td>1.000e+1</td>
<td>2.500e+1</td>
<td>1</td>
<td>1</td>
<td>1.990e-4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.475e-3</td>
<td>1.408e-4</td>
<td>2.499e+1</td>
<td>6.250e+1</td>
<td>5</td>
<td>5</td>
<td>6.700e-7</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5.531e-4</td>
<td>6.431e-7</td>
<td>4.077e+1</td>
<td>1.563e+1</td>
<td>27</td>
<td>7</td>
<td>3.893e-9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5.528e-4</td>
<td>5.039e-9</td>
<td>1.059e+0</td>
<td>3.906e+2</td>
<td>32</td>
<td>7</td>
<td>1.508e-9</td>
</tr>
<tr>
<td>Full</td>
<td>0</td>
<td>5.446e-2</td>
<td>5.989e-3</td>
<td>---</td>
<td>1.000e+1</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.375e-2</td>
<td>2.201e-3</td>
<td>1.000e+1</td>
<td>2.500e+1</td>
<td>1</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1.472e-3</td>
<td>1.401e-4</td>
<td>2.500e+1</td>
<td>6.250e+1</td>
<td>5</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5.538e-4</td>
<td>1.361e-6</td>
<td>4.051e+1</td>
<td>1.563e+1</td>
<td>19</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5.528e-4</td>
<td>8.416e-9</td>
<td>2.178e+0</td>
<td>3.906e+2</td>
<td>28</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

Adaptive-rank algorithm reduced memory requirement 23.14-fold!
Numerical Results  
Flow Control

We consider the optimal flow control problem

$$\min_z \int_0^T \left\{ \int_{\partial C} \left( \frac{1}{\text{Re}} \frac{\partial \mathbf{v}}{\partial n} - pn \right) \cdot (z\tau - \mathbf{v}_\infty) \, dx + \frac{\alpha}{2} z(t)^2 \right\} \, dt,$$

where the velocity/pressure \((\mathbf{v}, p) : [0, T] \times D \rightarrow \mathbb{R}^2 \times \mathbb{R}\) solves the Navier-Stokes equations

$$\frac{\partial \mathbf{v}}{\partial t} - \frac{1}{\text{Re}} \nabla \mathbf{v} + (\mathbf{v} \cdot \nabla) \mathbf{v} + \nabla p = 0 \quad \text{in } (0, T) \times D \setminus C$$

$$\nabla \cdot \mathbf{v} = 0 \quad \text{in } (0, T) \times D \setminus C$$

$$\frac{1}{\text{Re}} \frac{\partial \mathbf{v}}{\partial n} - pn = 0 \quad \text{on } (0, T) \times \Gamma_{\text{out}}$$

$$\mathbf{v} = \mathbf{v}_\infty \quad \text{on } (0, T) \times \partial D \setminus \Gamma_{\text{out}}$$

$$\mathbf{v} = z\tau \quad \text{on } (0, T) \times \partial C$$

**Goal:** Minimize the **power required to overcome the drag** on \(C\) by rotating \(C\).
Numerical Results

Flow Control

\[ \mathbf{v} = (0, 1)^T \]

\[ \mathbf{v}_2 = 0 \]

\[ \Gamma_{\text{out}} \]

Problem Data: \( D = (-15, 45) \times (-15, 15) \), \( C = B_{1/2} \), \( \text{Re} = 200 \), \( T = 20 \), and tangent vector

\[ \tau(x) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} (x - x_0), \quad x \in \partial C. \]

Spatial Discretization: Q2–Q1 finite elements.

Temporal Discretization: Implicit Euler with 800 equal time steps.
Numerical Results

Flow Control

Initial Vorticity ($t = 0$)

Controlled Vorticity ($t = 20$)

Initial Velocity ($t = 0$)

Controlled Velocity ($t = 20$)

Drew Kouri  Sandia National Laboratories

Sketching for Dynamic Optimization
Numerical Results

Flow Control

Sketching error averaged over 20 realizations and tail energy for the uncontrolled state (left) and the optimal state (right). Recall that the rank of the sketch is $k = 2r + 1$. 
**Termination Criterion:** $\|g^{(\ell)}\| \leq 10^{-5}$ or $\ell > 40$.

**Rank Increase Function:** $r \leftarrow 2r$.

<table>
<thead>
<tr>
<th>rank</th>
<th>objective</th>
<th>iter</th>
<th>nstate</th>
<th>nadjoint</th>
<th>iterCG</th>
<th>compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>*8</td>
<td>18.35919</td>
<td>40</td>
<td>41</td>
<td>15</td>
<td>136</td>
<td>45.44</td>
</tr>
<tr>
<td>*16</td>
<td>18.20003</td>
<td>40</td>
<td>41</td>
<td>33</td>
<td>897</td>
<td>23.35</td>
</tr>
<tr>
<td>*32</td>
<td>18.19779</td>
<td>40</td>
<td>41</td>
<td>31</td>
<td>236</td>
<td>11.80</td>
</tr>
<tr>
<td>64</td>
<td>18.19779</td>
<td>29</td>
<td>41</td>
<td>34</td>
<td>110</td>
<td>5.88</td>
</tr>
<tr>
<td>Adaptive</td>
<td>18.19779</td>
<td>23</td>
<td>27</td>
<td>24</td>
<td>121</td>
<td>5.88</td>
</tr>
<tr>
<td>Full</td>
<td>18.19779</td>
<td>29</td>
<td>30</td>
<td>24</td>
<td>107</td>
<td>---</td>
</tr>
</tbody>
</table>

*The rank 8, 16, and 32 experiments exceeded the maximum number of iterations.*
**Numerical Results**

Flow Control

**Left:** Sketch rank as a function of iteration of adaptive-rank algorithm. **Right:** Required gradient inexactness tolerance and computed residual norm as functions of iteration.

**Adaptive-rank algorithm reduced memory requirement 5.88-fold!**
Conclusions

- **Numerical solution** of dynamic optimization is **expensive** and requires **careful memory management**.

- We **compress** the dynamic state using **randomized matrix sketching**.

- State compression using sketching is **lossy** and results in **inexact gradients**.

- Introduced provably convergent algorithm to handle gradient inexactness. Algorithm adatively learns the rank of the optimal state — **May run out of memory for non-low-rank states!**

- Numerical examples suggest **$\sim 5-70$-fold reduction** in memory requirement.