Krylov subspace methods: GMRES

L. Grigori

EPFL and PSI

November 26, 2024

EPFL

PAUL SCHERRER INSTITUT PSI

Plan

Sparse linear solvers
- Sparse matrices and graphs
- Classes of linear solvers

Orthogonalization techniques: deterministic and randomized
- Gram-Schmidt, CholeskyQR and their randomized versions

Krylov subspace methods
- Arnoldi process and application to GMRES

Communication avoiding in iterative methods

Plan

Sparse linear solvers
- Sparse matrices and graphs
- Classes of linear solvers

Orthogonalization techniques: deterministic and randomized

Krylov subspace methods

Communication avoiding in iterative methods

Sparse matrices and graphs

- Most matrices arising from real applications are sparse.
- A 1M-by-1M submatrix of the web connectivity graph, constructed from an archive at the Stanford WebBase.

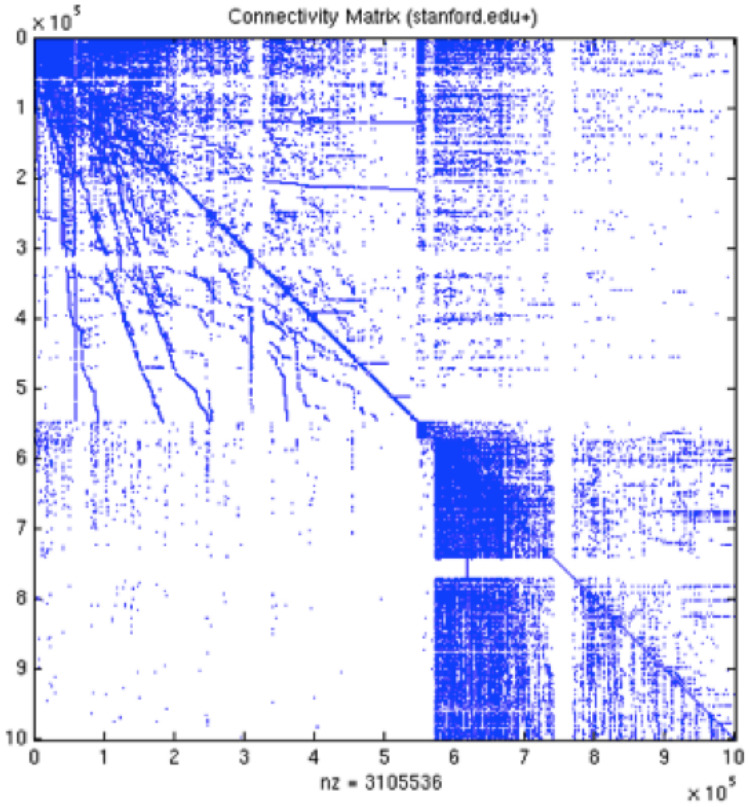


Figure: Nonzero structure of the matrix

Sparse matrices and graphs

- Most matrices arising from real applications are sparse.
- GHS class: Car surface mesh, $n = 100196, nnz(A) = 544688$

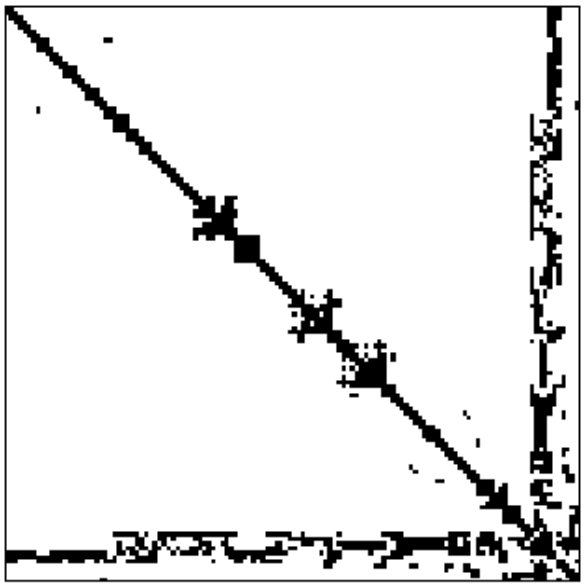


Figure: Nonzero structure of the matrix

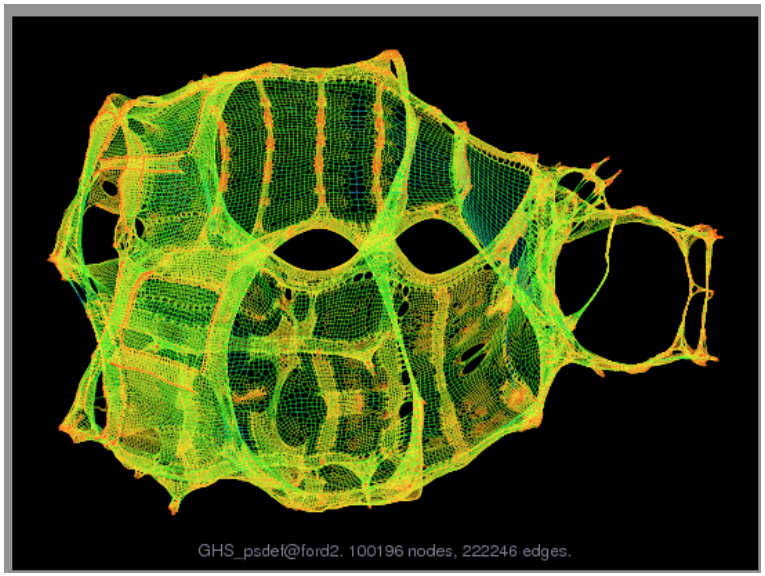


Figure: Its undirected graph

Examples from Tim Davis's Sparse Matrix Collection,
`http://www.cise.ufl.edu/research/sparse/matrices/`

Sparse matrices and graphs

- Semiconductor simulation matrix from Steve Hamm, Motorola, Inc. circuit with no parasitics, $n = 105676$, $nnz(A) = 513072$

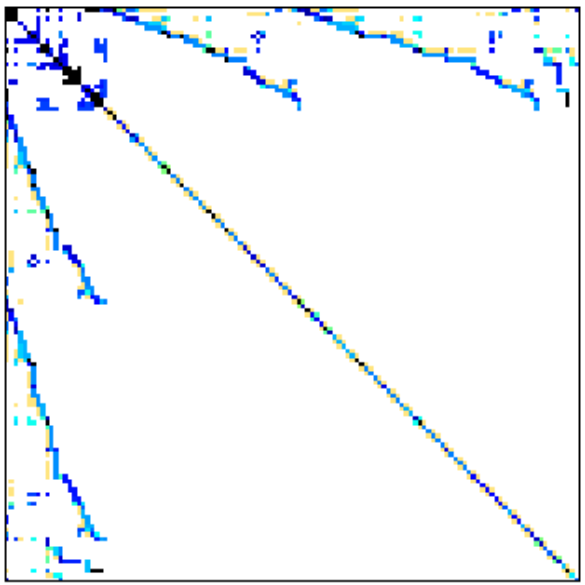


Figure: Nonzero structure of the matrix

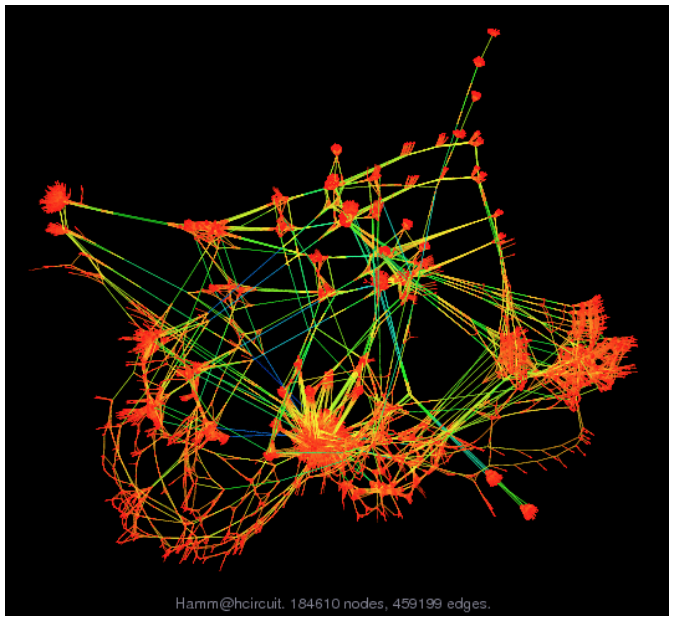


Figure: Its undirected graph

Examples from Tim Davis's Sparse Matrix Collection,
`http://www.cise.ufl.edu/research/sparse/matrices/`

Computing with sparse matrices - guiding principles

- Store nonzero elements

$$A = \begin{pmatrix} 1.1 & & 1.3 & & \\ & 2.2 & & 2.4 & \\ & 3.2 & & & 3.5 \\ 4.1 & & & 4.4 & \\ 5.1 & & 5.3 & & 5.5 \end{pmatrix}$$

  - Compressed sparse formats by columns (CSC), rows (CSR), or coordinates
  - For A of size $n \times n$, CSC uses one aray of size $n+1$ and two arrays of size $nnz(A)$:

$$\begin{array}{rc} \mathit{ColPtr} & \begin{pmatrix} 1 & 4 & 6 & 8 & 10 & 12 \end{pmatrix} \\ \mathit{RowInd} & \left(\begin{array}{ccc|cc|cc|cc|cc} 1 & 4 & 5 & 2 & 3 & 1 & 5 & 2 & 4 & 3 & 5 \\ \mathit{Vals} & 1.1 & 4.1 & 5.1 & 2.2 & 3.2 & 1.3 & 5.3 & 2.4 & 4.4 & 3.5 & 5.5 \end{array}\right) \end{array}$$

- Compute flops only on nonzeros elements
- Identify and exploit parallelism due to the sparsity of the matrix

Sparse linear solvers

Direct methods of factorization

- For solving $Ax = b$, least squares problems
  - Cholesky, LU, QR, LDL^T factorizations
- Limited by fill-in/memory consumption and scalability

Iterative solvers

- For solving $Ax = b$, least squares, $Ax = \lambda x$, SVD
- When only multiplying A by a vector is possible
- Limited by accuracy/convergence

Hybrid methods

As domain decomposition methods

Krylov subspace methods

Solve $Ax = b$ by finding a sequence $x_1, x_2, ..., x_m$ that minimizes some measure of error over the corresponding spaces

$$x_0 + \mathcal{K}_i(A, r_0), \quad i = 1, ..., m$$

.

They are defined by two conditions:

1. Subspace condition: $x_m \in x_0 + \mathcal{K}_m(A, r_0)$
2. Petrov-Galerkin condition: $r_m \perp \mathscr{L}_m$

$$\iff (r_m)^t y = 0, \quad \forall \quad y \in \mathscr{L}_m$$

where

- x_0 is the initial iterate, r_0 is the initial residual, $r_m = b - Ax_m$
- $\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2 r_0, ..., A^{m-1} r_0\}$ is the Krylov subspace of dimension m
- $\mathscr{L}_m$ is a well-defined subspace of dimension m
- For stability, the vectors of the Krylov basis are orthogonalized

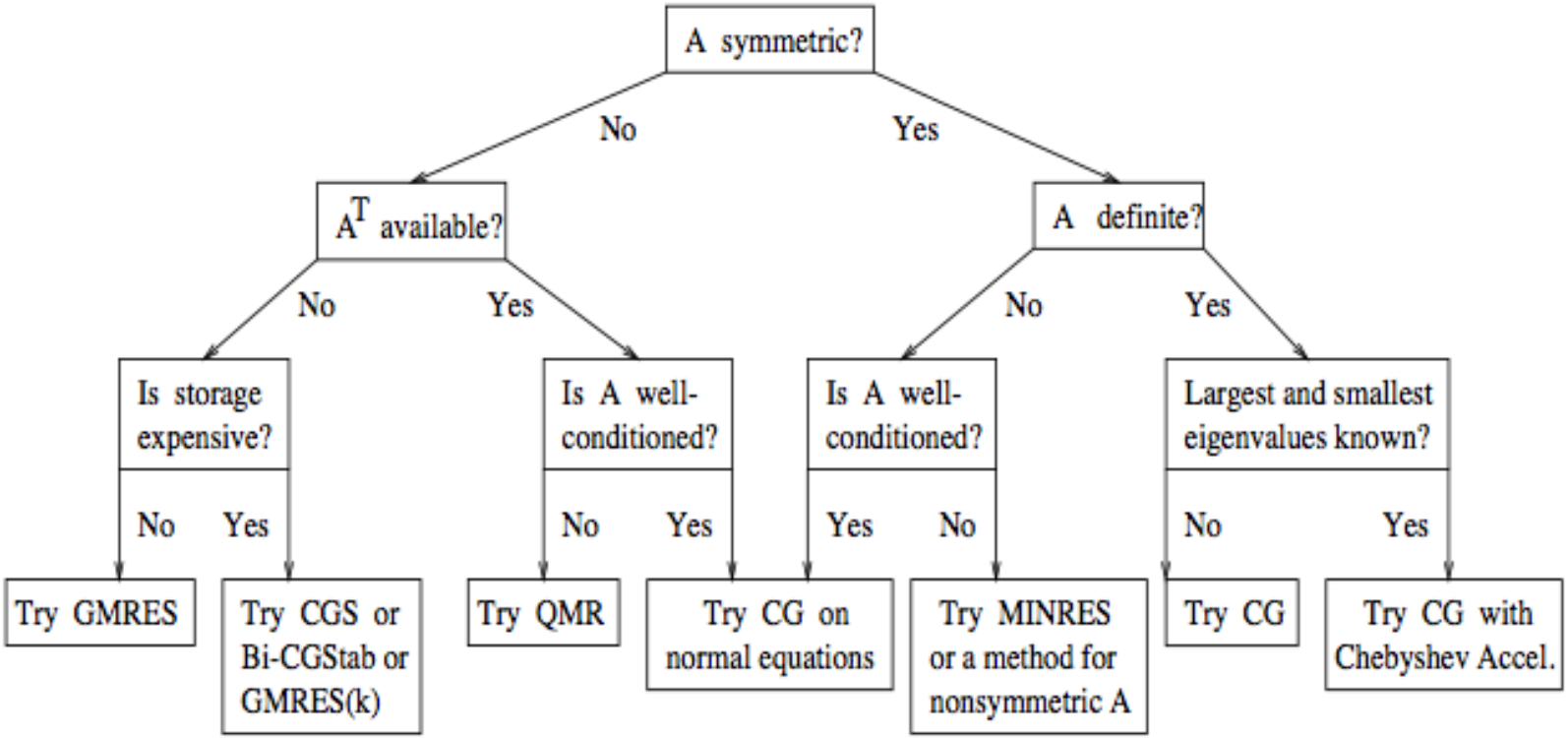
One of Top Ten Algorithms of the 20th Century

From SIAM News, Volume 33, Number 4:
Magnus Hestenes, Eduard Stiefel, and Cornelius Lanczos, all from the Institute for Numerical Analysis at the National Bureau of Standards, initiate the development of Krylov subspace iteration methods.

- Russian mathematician Alexei Krylov writes first paper, 1931.
- Lanczos - introduced an algorithm to generate an orthogonal basis for such a subspace when the matrix is symmetric.
- Hestenes and Stiefel - introduced CG for SPD matrices.

Other Top Ten Algorithms: Monte Carlo method, decompositional approach to matrix computations (Householder), Quicksort, Fast multipole, FFT.

Choosing a Krylov method



All methods (GMRES, CGS,CG…) depend on SpMV (or variations…)

See www.netlib.org/templates/Templates.html for details

Source slide: J. Demmel

Challenge in getting efficient and scalable solvers

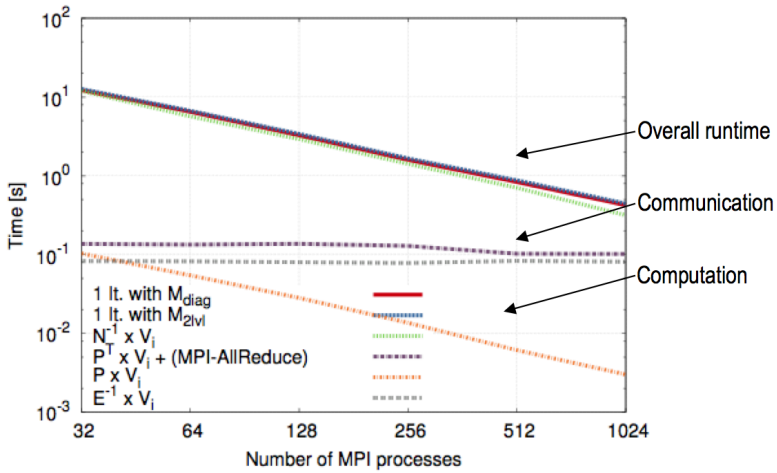
- A Krylov solver finds x_m from $x_0 + \mathcal{K}_m(A, r_0)$ where

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2 r_0, ..., A^{m-1} r_0\},$$

  such that the Petrov-Galerkin condition $b - Ax_m \perp \mathscr{L}_m$ is satisfied.
- Does a sequence of sparse matrix vector products to get vectors $[x_1, ..., x_m]$
- Finds best solution x_m as linear combination of $[x_1, ..., x_{m-1}]$
- For numerical stability, an (A)-orthonormal basis $\{q_1, q_2, ..., q_m\}$ for $\mathcal{K}_m(A, r_0)$ is computed (CG, GMRES, BiCGstab,...).

Typically, each iteration requires

- Sparse matrix vector product
  $\rightarrow$ point-to-point communication
- Dot products for orthogonalization
  $\rightarrow$ global communication



Map making, with R. Stompor, M. Szydlarski
Results obtained on Hopper, Cray XE6, NERSC

Challenge in getting efficient and scalable solvers

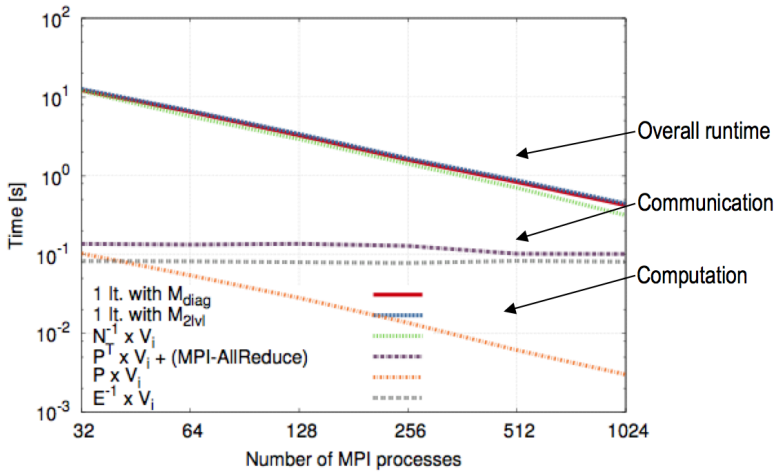
- A Krylov solver finds x_m from $x_0 + \mathcal{K}_m(A, r_0)$ where

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2 r_0, ..., A^{m-1} r_0\},$$

  such that the Petrov-Galerkin condition $b - Ax_m \perp \mathscr{L}_m$ is satisfied.
- Does a sequence of sparse matrix vector products to get vectors $[x_1, ..., x_m]$
- Finds best solution x_m as linear combination of $[x_1, ..., x_{m-1}]$
- For numerical stability, an (A)-orthonormal basis $\{q_1, q_2, ..., q_m\}$ for $\mathcal{K}_m(A, r_0)$ is computed (CG, GMRES, BiCGstab,...).

Typically, each iteration requires

- Sparse matrix vector product
  $\rightarrow$ point-to-point communication
- Dot products for orthogonalization
  $\rightarrow$ global communication



Map making, with R. Stompor, M. Szydlarski
Results obtained on Hopper, Cray XE6, NERSC

Ways to improve performance

- Improve the performance of sparse matrix-vector product
- Improve the performance of MPI communication
- Change numerics - reformulate or introduce Krylov subspace algorithms to:
  - reduce communication,
  - increase arithmetic intensity - compute sparse matrix-set of vectors product
- Use preconditioners to decrease the number of iterations till convergence.

Plan

Sparse linear solvers

Orthogonalization techniques: deterministic and randomized

- Gram-Schmidt, CholeskyQR and their randomized versions

Krylov subspace methods

Communication avoiding in iterative methods

Gram-Schmidt (GS) orthogonalization process

Given set of linearly independent vectors $W = [w_1, \ldots, w_m]$, $W \in \mathbb{R}^{n \times m}$.
Construct an orthogonal basis $Q = [q_1, \ldots, q_m]$ such that $W = QR$.

For each w_j **do**

1. Given P_{j-1} projector on $\mathrm{span}(Q_{j-1})^{\perp}$, compute $q_j \perp q_1, \ldots, q_{j-1}$ as

$$q_j = P_{j-1} w_j$$

2. Normalize q_j

End For

Projectors P_j used in Gram-Schmidt

- **Classical Gram-Schmidt (CGS)**: BLAS-2 matrix-vector operations
  - one synchronization, assume $c_2(m,n)\kappa^2(W)u < 1$,

$$P_{j-1} = I - Q_{j-1}Q_{j-1}^T, \quad \|I - Q^TQ\| = c_1(m,n)\kappa^2(W)u, \;\; c_1, c_2 = \mathcal{O}(nm^2)$$

  $\rightarrow$ good efficiency, but stability issues

- **Modified Gram-Schmidt (MGS)** : BLAS-1 vector-vector operations
  - $(j-1)$ synchronizations, assume $c_3(m,n)\kappa(W)u < 1$,

$$P_{j-1} = (I - q_{j-1}q_{j-1}^T)\ldots(I - q_1q_1^T), \; \|I - Q^TQ\| = c_3(m,n)\kappa(W)u, c_3 = \mathcal{O}(nm)$$

  $\rightarrow$ better numerical stability, but poor efficiency

Note: u machine precision, $W \in \mathbb{R}^{n\times m}$, bounds from [Giraud et al., 2005].

Projectors P_j used in Gram-Schmidt

- **Classical Gram-Schmidt (CGS)**: BLAS-2 matrix-vector operations
  - one synchronization, assume $c_2(m,n)\kappa^2(W)u < 1$,

$$P_{j-1} = I - Q_{j-1}Q_{j-1}^T, \quad \|I - Q^TQ\| = c_1(m,n)\kappa^2(W)u, \;\; c_1, c_2 = \mathcal{O}(nm^2)$$

$\rightarrow$ good efficiency, but stability issues

- **Modified Gram-Schmidt (MGS)** : BLAS-1 vector-vector operations
  - $(j-1)$ synchronizations, assume $c_3(m,n)\kappa(W)u < 1$,

$$P_{j-1} = (I - q_{j-1}q_{j-1}^T)\ldots(I - q_1q_1^T), \; \|I - Q^TQ\| = c_3(m,n)\kappa(W)u, c_3 = \mathcal{O}(nm)$$

$\rightarrow$ better numerical stability, but poor efficiency

Note: u machine precision, $W \in \mathbb{R}^{n\times m}$, bounds from [Giraud et al., 2005].

Cholesky QR

- **CholeskyQR** : BLAS-3 matrix-matrix operations
  Compute $W = \tilde{Q}\tilde{R}$ as:
  1. Compute the Cholesky factorization of $W^T W$ as $W^T W = \tilde{R}^T \tilde{R}$
  2. Compute the orthogonal factor as $\tilde{Q} = W\tilde{R}^{-1}$
- Numerical stability: assume $O(u)\kappa^2(W) < 1$, then

$$\|I - \tilde{Q}^T \tilde{Q}\| = O(u)\kappa^2(W)$$

Recap on ε-subspace embedding property

For a given subspace $\mathcal{V} \subset \mathbb{R}^m$ and $\varepsilon \in (0,1)$, a sketching matrix $\Omega \in \mathbb{R}^{l \times m}$ is an ε-embedding for $\mathcal{V}$ if for all $x_i, x_j \in \mathcal{V}$, we have

$$|\langle \Omega x_i, \Omega x_j \rangle - \langle x_i, x_j \rangle| \leq \epsilon \|x_i\|_2 \|x_j\|_2 \tag{1}$$

Let W be a matrix whose columns form a basis for $\mathcal{V}$. For simplicity, we refer to an ε-subspace embedding for $\mathcal{V}$ as an ε-embedding for W.

Corollary (Corollary 2.2 in [Balabanov and Grigori, 2022])

If $\Omega \in \mathbb{R}^{l \times m}$ is an ε-embedding for W, then the singular values of W are bounded by

$$(1+\varepsilon)^{-1/2}\sigma_{min}(\Omega W) \ \leq \sigma_{min}(W) \leq \sigma_{max}(W) \leq (1-\varepsilon)^{-1/2}\sigma_{max}(\Omega W).$$

Randomized orthogonalization processes

Main underlying idea:

If sketching matrix Ω is OSE(m, ϵ, δ), hence preserves geometry (inner products) for $\mathrm{span}(Q)$ with high probability (w.h.p.), then

$$\kappa(Q) \leq \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}\kappa(\Omega Q)$$

$\rightarrow$ orthogonalize ΩQ instead of Q.

Randomized Cholesky QR

For simplicity, assume $W \in \mathbb{R}^{n \times m}$ is nonsingular

Randomized Cholesky QR

- Compute $W_s = \Omega W$, $W_s \in \mathbb{R}^{l \times m}$, $\Omega \in \mathbb{R}^{l \times n}$
- Compute the QR factorization $W_s = Q_s R_s$ where $Q_s \in \mathbb{R}^{l \times m}$ has orthonormal columns (if W is ill-conditioned, one could use the rank revealing QR factorization)
- Compute $Q = W R_s^{-1}$

Randomized Cholesky QR

In infinite precision we have:

$$\begin{aligned} Q &= WR_s^{-1} \\ \Omega Q &= \Omega W R_s^{-1} = W_s R_s^{-1} = Q_s \end{aligned}$$

and hence, if sketching matrix Ω is OSE(m, ϵ, δ) for $\mathrm{span}(Q)$, with h.p.:

$$\kappa(Q) \leq \sqrt{\frac{1+\varepsilon}{1-\varepsilon}} \kappa(\Omega Q) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}$$

Randomized Gram-Schmidt (RGS) orthogonalization

For a more stable procedure, avoid using R_s^{-1}, instead use the previously computed $j-1$ columns of Q to compute a new column j

- P_{j-1}^r is an orthogonal projector wrt $\langle \Omega \cdot, \Omega \cdot \rangle$

$$P_{j-1}^r = I - Q_{j-1}(\Omega Q_{j-1})^\dagger \Omega$$

- $\Omega W = SR$, with $S = \Omega Q$ ℓ_2-orthonormal
- Obtain $W = QR$ factorization such that Q is unit-orthogonal wrt sketched inner product $\langle \Omega \cdot, \Omega \cdot \rangle$

Reference: [Balabanov and Grigori, 2022, Balabanov and Grigori, 2020]

Randomized Gram-Schmidt orthogonalization (contd)

- Compute $W = QR$ using P^r_{j-1} orth. projector with respect to $\langle \Omega \cdot, \Omega \cdot \rangle$.

$$P^r_{j-1} = I - Q_{j-1}(\Omega Q_{j-1})^\dagger \Omega$$

For each w_j **do**

Given $W_{j-1} = Q_{j-1}R_{j-1}, j > 1$, compute $[W_{j-1}, w_j] = [Q_{j-1}, \quad q_j] \begin{bmatrix} R_{j-1} & r_{1:j-1,j} \\ & r_{jj} \end{bmatrix}$

1. Compute the sketch Ωw_j
2. Compute $q_j = P^r_{j-1} w_j$ by solving a small least-squares problem:

$$\begin{aligned} r_{1:j-1,j} &= [S_{j-1}]^\dagger \Omega w_j, \text{ where } S_{j-1} = \Omega Q_{j-1} \\ q_j &= w_j - Q_{j-1} r_{1:j-1,j} \end{aligned}$$

2. Sketch the new vector $s_j = \Omega q_j$
3. Normalize $r_{j,j} = \|s_j\|_2$ and $q_j = q_j / r_{j,j}$, $s_j = s_j / r_{j,j}$

End For

Randomized Gram-Schmidt orthogonalization (contd)

- Compute $W = QR$ using P^r_{j-1} orth. projector with respect to $\langle \Omega \cdot, \Omega \cdot \rangle$.

$$P^r_{j-1} = I - Q_{j-1}(\Omega Q_{j-1})^\dagger \Omega$$

For each w_j **do**

Given $W_{j-1} = Q_{j-1}R_{j-1}, j > 1$, compute $[W_{j-1}, w_j] = \begin{bmatrix} Q_{j-1}, & q_j \end{bmatrix} \begin{bmatrix} R_{j-1} & r_{1:j-1,j} \\ & r_{jj} \end{bmatrix}$

1. Compute the sketch Ωw_j
2. Compute $q_j = P^r_{j-1} w_j$ by solving a small least-squares problem:

$$\begin{aligned} r_{1:j-1,j} &= [S_{j-1}]^\dagger \Omega w_j, \text{ where } S_{j-1} = \Omega Q_{j-1} \\ q_j &= w_j - Q_{j-1} r_{1:j-1,j} \end{aligned}$$

2. Sketch the new vector $s_j = \Omega q_j$
3. Normalize $r_{j,j} = \|s_j\|_2$ and $q_j = q_j / r_{j,j}$, $s_j = s_j / r_{j,j}$

End For

Randomized Gram-Schmidt orthogonalization (contd)

- Compute $W = QR$ using P^r_{j-1} orth. projector with respect to $\langle \Omega \cdot, \Omega \cdot \rangle$.

$$P^r_{j-1} = I - Q_{j-1}(\Omega Q_{j-1})^\dagger \Omega$$

For each w_j **do**

Given $W_{j-1} = Q_{j-1}R_{j-1}, j > 1$, compute $[W_{j-1}, w_j] = \begin{bmatrix} Q_{j-1}, & q_j \end{bmatrix} \begin{bmatrix} R_{j-1} & r_{1:j-1,j} \\ & r_{jj} \end{bmatrix}$

1. Compute the sketch Ωw_j
2. Compute $q_j = P^r_{j-1} w_j$ by solving a small least-squares problem:

$$\begin{aligned} r_{1:j-1,j} &= [S_{j-1}]^\dagger \Omega w_j, \text{ where } S_{j-1} = \Omega Q_{j-1} \\ q_j &= w_j - Q_{j-1} r_{1:j-1,j} \end{aligned}$$

2. Sketch the new vector $s_j = \Omega q_j$
3. Normalize $r_{j,j} = \|s_j\|_2$ and $q_j = q_j / r_{j,j}$, $s_j = s_j / r_{j,j}$

End For

Stability of randomized Gram-Schmidt

Assumptions:

- Sketching Ω is OSE(m, ϵ, δ), and preserves inner products for $\mathrm{span}(W)$ with h.p.
- Backward-stable least-squares solver for solving

$$(\hat{S}_{j-1})^{\dagger}\hat{p}_j = \arg\min_{\hat{x}} \|\hat{S}_{j-1}x - \hat{p}_j\|, \text{ where } \hat{p}_j = \Omega w_j$$

- Mixed precision model with $u_{fine}/c_4(n,m) < u_{crs} < 1/c_5(m)$, $c_4(n,m) = \mathcal{O}(m^{1/2}n^{3/2})$, $c_5(m) = \mathcal{O}(m^2\kappa(W))$ (in practice better constants)
  - Fine precision u_{fine} for low dim. operations and random projections
  - Coarse precision u_{crs} for expensive high-dim. operations

Stability result with probabilistic rounding and mixed precision

Let $\hat{Q}$, $\hat{R}$ be the computed QR factors of $W \in \mathbb{R}^{n\times m}$ in finite precision. With high probability:

$$\|W - \hat{Q}\hat{R}\|_F \leq \mathcal{O}(u_{crs}m^{3/2})\|W\|_F,$$

$$\kappa(\hat{Q}) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}(1 + u_{crs}c_5(m)),\ c_5(m) = \mathcal{O}(m^2\kappa(W))$$

Stability of randomized Gram-Schmidt

Assumptions:

- Sketching Ω is OSE(m, ϵ, δ), and preserves inner products for $\mathrm{span}(W)$ with h.p.
- Backward-stable least-squares solver for solving

$$(\hat{S}_{j-1})^{\dagger}\hat{p}_j = \arg\min_{\hat{x}} \|\hat{S}_{j-1}x - \hat{p}_j\|, \text{ where } \hat{p}_j = \Omega w_j$$

- Mixed precision model with $u_{fine}/c_4(n,m) < u_{crs} < 1/c_5(m)$, $c_4(n,m) = \mathcal{O}(m^{1/2}n^{3/2})$, $c_5(m) = \mathcal{O}(m^2\kappa(W))$ (in practice better constants)
  - Fine precision u_{fine} for low dim. operations and random projections
  - Coarse precision u_{crs} for expensive high-dim. operations

Stability result with probabilistic rounding and mixed precision

Let $\hat{Q}$, $\hat{R}$ be the computed QR factors of $W \in \mathbb{R}^{n\times m}$ in finite precision. With high probability:

$$\|W - \hat{Q}\hat{R}\|_F \leq \mathcal{O}(u_{crs}m^{3/2})\|W\|_F,$$

$$\kappa(\hat{Q}) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}(1 + u_{crs}c_5(m)),\ c_5(m) = \mathcal{O}(m^2\kappa(W))$$

Stability of randomized Gram-Schmidt

Assumptions:

- Sketching Ω is OSE(m, ϵ, δ), and preserves inner products for $\mathrm{span}(W)$ with h.p.
- Backward-stable least-squares solver for solving

$$(\hat{S}_{j-1})^{\dagger}\hat{p}_j = \arg\min_{\hat{x}} \|\hat{S}_{j-1}x - \hat{p}_j\|, \text{ where } \hat{p}_j = \Omega w_j$$

- Mixed precision model with $u_{fine}/c_4(n,m) < u_{crs} < 1/c_5(m)$, $c_4(n,m) = \mathcal{O}(m^{1/2}n^{3/2})$, $c_5(m) = \mathcal{O}(m^2\kappa(W))$ (in practice better constants)
  - Fine precision u_{fine} for low dim. operations and random projections
  - Coarse precision u_{crs} for expensive high-dim. operations

Stability result with probabilistic rounding and mixed precision

Let $\hat{Q}$, $\hat{R}$ be the computed QR factors of $W \in \mathbb{R}^{n\times m}$ in finite precision. With high probability:

$$\|W - \hat{Q}\hat{R}\|_F \leq \mathcal{O}(u_{crs}m^{3/2})\|W\|_F,$$

$$\kappa(\hat{Q}) = \sqrt{\frac{1+\varepsilon}{1-\varepsilon}}(1 + u_{crs}c_5(m)),\ c_5(m) = \mathcal{O}(m^2\kappa(W))$$

Randomized Gram-Schmidt vs. CGS and MGS

Classical Gram-Schmidt (CGS):

- BLAS-2 matrix-vector operations
- One synchronization

→ good efficiency, but stability issues

Modified Gram-Schmidt (MGS) :

- BLAS-1 vector-vector operations
- $(j-1)$ synchronizations

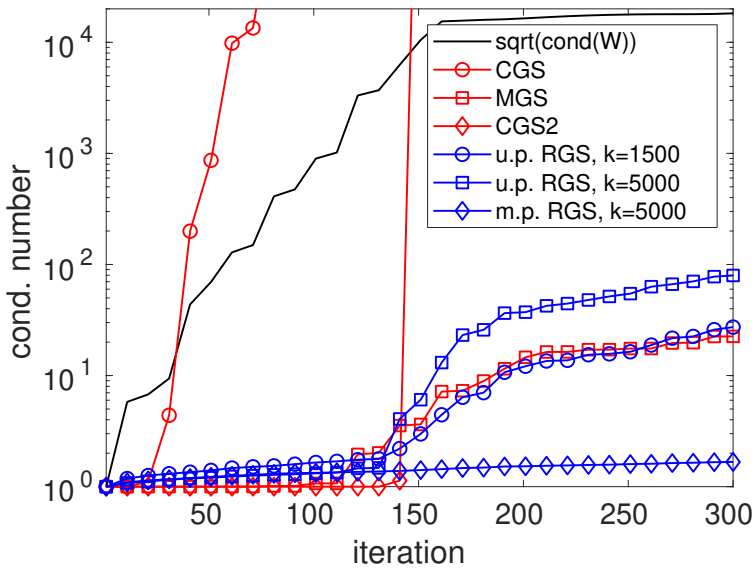
→ better numerical stability, but poor efficiency

Randomized Gram-Schmidt (RGS):

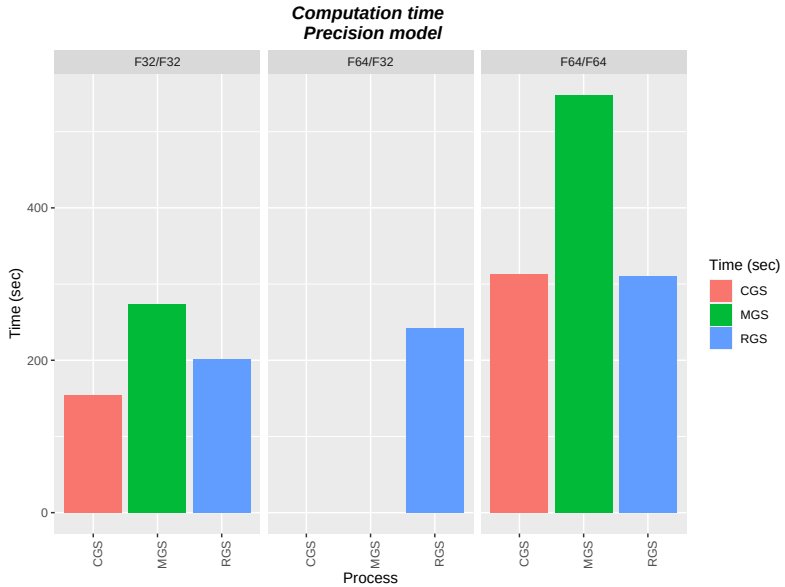
- BLAS-2 matrix-vector operations
- Twice less flops than CGS and MGS (could be computed in lower precision)
- One synchronization
- Numerical stability similar (or better) to MGS

Randomized GS for synthetic function

Cond number of Q_i (Matlab)



Results in Julia (with V. Lederer)



- Unique precision (u.p.): fp32 for all arithmetic operations, i.e., $u_{crs} = u_{fine} \approx 10^{-8}$
- Mixed precision (m.p.): fp32/fp64 formats, i.e. $u_{crs} \approx 10^{-8}$ and $u_{fine} \approx 10^{-16}$
- Synthetic function:

$$f_\mu(x) = \frac{\sin(10(\mu + x))}{\cos(100(\mu - x)) + 1.1}, \; x \in [0,1], \; \mu \in [0,1],$$

$$W(i,j) = f_{\mu_j}(x_i), 1 \le i \le 10^6, 1 \le m \le 300$$

Plan

Sparse linear solvers

Orthogonalization techniques: deterministic and randomized

Krylov subspace methods

- Arnoldi process and application to GMRES

Communication avoiding in iterative methods

Krylov subspace methods for solving $Ax = b$

Finds a sequence $x_1, x_2, ..., x_m$ that minimizes some measure of error over the spaces $x_0 + \mathcal{K}_i(A, r_0),\ i = 1, ..., m$, by satisfying two conditions:

1. Subspace condition: $x_m \in x_0 + \mathcal{K}_m(A, r_0)$
2. Petrov-Galerkin condition: $r_m \perp \mathscr{L}_m \iff (r_m)^t y = 0,\ \forall\ y \in \mathscr{L}_m$

where

- x_0 initial iterate, r_0 initial residual, $\mathscr{L}_m$ well-defined subspace of dimension m
- $\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2 r_0, ..., A^{m-1} r_0\}$ is the Krylov subspace of dimension m.

Two instances of Krylov projection method:

Conjugate gradient [Hestenes, Stieffel, 52], A is SPD, $\mathscr{L}_m = \mathcal{K}_m(A, r_0)$,

- finds x_m by minimizing $\|x - x_m\|_A$ over $x_0 + \mathcal{K}_m(A, r_0)$

GMRES [Saad, Schultz, 86], A is unsymmetric, $\mathscr{L}_m = A\mathcal{K}_m(A, r_0)$,

- finds x_m by minimizing $\|Ax - b\|_2$ over $x_0 + \mathcal{K}_m(A, r_0)$.

Krylov subspace methods for solving $Ax = b$

Finds a sequence $x_1, x_2, ..., x_m$ that minimizes some measure of error over the spaces $x_0 + \mathcal{K}_i(A, r_0),\ i = 1, ..., m$, by satisfying two conditions:

1. Subspace condition: $x_m \in x_0 + \mathcal{K}_m(A, r_0)$
2. Petrov-Galerkin condition: $r_m \perp \mathscr{L}_m \iff (r_m)^t y = 0,\ \forall\ y \in \mathscr{L}_m$

where

- x_0 initial iterate, r_0 initial residual, $\mathscr{L}_m$ well-defined subspace of dimension m
- $\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, A^2 r_0, ..., A^{m-1} r_0\}$ is the Krylov subspace of dimension m.

Two instances of Krylov projection method:

Conjugate gradient [Hestenes, Stieffel, 52], A is SPD, $\mathscr{L}_m = \mathcal{K}_m(A, r_0)$,

- finds x_m by minimizing $\|x - x_m\|_A$ over $x_0 + \mathcal{K}_m(A, r_0)$

GMRES [Saad, Schultz, 86], A is unsymmetric, $\mathscr{L}_m = A\mathcal{K}_m(A, r_0)$,

- finds x_m by minimizing $\|Ax - b\|_2$ over $x_0 + \mathcal{K}_m(A, r_0)$.

Arnoldi process

- Basis $Q_m = \{q_1, \ldots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through Gram-Schmidt (CGS or MGS), that is by computing the QR fact. of Arnoldi matrix $[r_0, AQ_m]$ [Paige et al, 06]
- Produces a Hessenberg matrix $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ and $H_m \in \mathbb{R}^{m\times m}$ obtained from $\bar{H}_m$ by deleting its last row:

$$\bar{H}_m = \begin{pmatrix} h_{11} & h_{12} & \ldots & h_{1n} \\ h_{21} & h_{22} & \ldots & h_{2n} \\ 0 & h_{32} & \ldots & h_{3n} \\ & & \ddots & \vdots \\ 0 & 0 & & \end{pmatrix} \tag{2}$$

Algorithm 1 Arnoldi with CGS

1: $r_0 = b - Ax_0, \beta = \|r_0\|_2, q_1 = r_0/\beta$
2: Define $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$, $\bar{H}_m = \{h_{ij}\}_{1\le i\le m+1, 1\le j\le m}$. Set $\bar{H}_m = 0$
3: **for** $j = 1 : m$ **do**
4: $\quad w_{j+1} = Aq_j$
5: $\quad$ Compute $h_{ij} = \langle w_{j+1}, q_i\rangle$ for $i = 1, 2, \ldots, j$
6: $\quad \bar{w}_{j+1} = w_{j+1} - \sum_{i=1}^{j} h_{ij} q_i$ // CGS to orthogonalize w_{j+1} against $\{q_1, \ldots, q_j\}$,
7: $\quad h_{j+1,j} = \|\bar{w}_{j+1}\|_2$
8: $\quad$ If $h_{j+1,j} = 0$ then Stop
9: $\quad q_{j+1} = \bar{w}_{j+1}/h_{j+1,j}$
10: **end for**

Properties of Arnoldi process

Proposition (see Proposition 6.4 in [Saad, 2003]) Assume Algorithm 1 does not stop before m-th step. Then the vectors $\{q_1, \ldots, q_m\}$ form an orthonormal basis for the Krylov subspace

$$\mathcal{K}_m(A, r_0) = span\{q_1, Aq_1, A^2q_1, ..., A^{m-1}q_1\}$$

Sketch of the proof: Show by induction that each vector $q_j = p_{j-1}(A)q_1$, where p_{j-1} is a polynomial of degree $j-1$.
True for $j = 1$ since $q_1 = p_0(A)q_1$. Suppose true for all $q_2, \ldots q_j$ and consider q_{j+1},

$$h_{j+1,j}q_{j+1} = Aq_j - \sum_{i=1}^{j} h_{ij}q_i = Ap_{j-1}(A)q_1 - \sum_{i=1}^{j} h_{ij}p_{i-1}(A)q_1 = p_j(A)q_1$$

Properties of Arnoldi process (contd)

Proposition (see Proposition 6.5 in [Saad, 2003]) Let H_m be formed from $\bar{H}_m$ by deleting its last row. Then we have:

$$\begin{aligned} AQ_m &= Q_m H_m + \bar{w}_{m+1} e_m^T && (3) \\ &= Q_{m+1} \bar{H}_m, && (4) \\ Q_m^T A Q_m &= H_m && (5) \end{aligned}$$

Eq. (4) is obtained since from Arnoldi, we have that:

$$Aq_j = \sum_{i=1}^{j+1} h_{ij} q_i, \; j = 1, 2, \ldots, m \qquad (6)$$

The other equations follow since Q_m is formed by orthonormal columns.

GMRES: derivation of the algorithm

By the subspace condition, we look for a vector x in $x_0 + \mathcal{K}_m(A, r_0)$, which can be written as:

$$x = x_0 + Q_m y, \quad y \in \mathbb{R}^m \tag{7}$$

Define (where $\beta = \|r_0\|_2$):

$$\begin{aligned} J(y) &= \|b - Ax\|_2 = \|b - A(x_0 + Q_m y)\|_2 && (8)\\ &= \|r_0 - AQ_m y\|_2 = \|\beta q_1 - Q_{m+1}\bar{H}_m y\|_2 && (9)\\ &= \|Q_{m+1}(\beta e_1 - \bar{H}_m y)\|_2 = \|\beta e_1 - \bar{H}_m y\|_2 && (10) \end{aligned}$$

GMRES obtains a new solution by minimizing (8) that is:

$$\begin{aligned} x_m &= x_0 + Q_m y_m, \text{ where} && (11)\\ y_m &= \mathit{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2 && (12) \end{aligned}$$

which requires solving an $(m+1) \times m$ least squares problem

GMRES

- Basis $Q_m = \{q_1, \ldots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through GS, QR fact. of Arnoldi matrix $[r_0, AQ_m]$ [Paige et al, 06]
- Produces orthonormal Q_m satisfying

$$AQ_m = Q_{m+1}\bar{H}_m$$

- At iteration j of GMRES, GS computes: $[r_0, AQ_j] = Q_{j+1}[\|r_0\|_2 e_1, \bar{H}_j]$

Algorithm 2 GMRES with MGS

1: $r_0 = b - Ax_0, \beta = \|r_0\|_2,\ q_1 = r_0/\beta$
2: **for** $j = 1 : m - 1$ **do**
3: $\quad w_{j+1} = Aq_j$
4: $\quad$ MGS to orthogonalize w_{j+1} against $\{q_1, \ldots, q_j\}$
5: $\quad$ Obtain $[r_0, AQ_j] = Q_{j+1}[\|r_0\|_2 e_1, \bar{H}_j]$
6: **end for**
7: Solve $y_m = argmin_y \|\beta e_1 - \bar{H}_m y\|_2$
8: $x = x_0 + Q_m y_m$

Arnoldi process and GMRES

- Basis $Q_m = \{q_1, \dots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through Gram-Schmidt (GS) QR fact. of Arnoldi matrix $[r_0, AQ_m]$
- Produces orthonormal Q_m and upper Hessenberg H_m satisfying

$$AQ_m = Q_{m+1}\bar{H}_m$$

Randomized Arnoldi

- Basis $Q_m = \{q_1, \dots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through randomized Gram-Schmidt (RGS) QR fact. of Arnoldi matrix $[r_0, AQ_m]$
- Q orthonormal with respect to the sketched product $\langle \Omega \cdot, \Omega \cdot \rangle$ and upper Hessenberg H_m satisfy

$$AQ_m = Q_{m+1}\bar{H}_m$$

Arnoldi process and GMRES

- Basis $Q_m = \{q_1, \dots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through Gram-Schmidt (GS) QR fact. of Arnoldi matrix $[r_0, AQ_m]$
- Produces orthonormal Q_m and upper Hessenberg H_m satisfying

$$AQ_m = Q_{m+1}\bar{H}_m$$

Randomized Arnoldi

- Basis $Q_m = \{q_1, \dots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through randomized Gram-Schmidt (RGS) QR fact. of Arnoldi matrix $[r_0, AQ_m]$
- Q orthonormal with respect to the sketched product $\langle \Omega \cdot, \Omega \cdot \rangle$ and upper Hessenberg H_m satisfy

$$AQ_m = Q_{m+1}\bar{H}_m$$

GMRES

- Basis $Q_m = \{q_1, \dots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through (randomized) GS, QR fact. of Arnoldi matrix $[r_0, AQ_m]$ [Paige et al, 06]
- Produces (sketch) orthonormal Q_m satisfying

$$AQ_m = Q_{m+1}\bar{H}_m$$

Algorithm 3 GMRES with GS

1: $r_0 = b - Ax_0, \beta = \|r_0\|_2, q_1 = r_0/\beta$
2: **for** $j = 1 : m$ **do**
3: $w_{j+1} = Aq_j$
4: MGS to orthogonalize w_{j+1} against $\{q_1, \dots, q_j\}$
5: Obtain $[r_0, AQ_j] = Q_{j+1}[\|r_0\|_2 e_1, \bar{H}_j]$
6: **end for**
7: Solve $y_m = argmin_y \|\beta e_1 - \bar{H}_m y\|_2$
8: $x = x_0 + Q_m y_m$

Algorithm 4 GMRES with randomized GS

1: $r_0 = b - Ax_0, \beta = \|r_0\|_2, q_1 = r_0/\beta$
2: **for** $j = 1 : m$ **do**
3: $w_{j+1} = Aq_j$
4: Randomized GS to orthogonalize w_{j+1} against $\{q_1, \dots, q_j\}$
5: Obtain $[r_0, AQ_j] = Q_{j+1}[\|r_0\|_2 e_1, \bar{H}_j]$
6: **end for**
7: Solve $y_m = argmin_y \|\beta e_1 - \bar{H}_m y\|_2$
8: $x = x_0 + Q_m y_m$

Cost per iteration MGS

$\#$ flops $= 4mn$

$\#$ synchronizations $= j - 1$

Cost per iteration randomized GS

$\#$ flops $= 2mn + 4nlog_2 n$

$\#$ synchronizations $= 1$

GMRES

- Basis $Q_m = \{q_1, \dots q_m\}$ of $\mathcal{K}_m(A, r_0)$ obtained through (randomized) GS, QR fact. of Arnoldi matrix $[r_0, AQ_m]$ [Paige et al, 06]
- Produces (sketch) orthonormal Q_m satisfying

$$AQ_m = Q_{m+1}\bar{H}_m$$

Algorithm 5 GMRES with GS

1: $r_0 = b - Ax_0, \beta = \|r_0\|_2, q_1 = r_0/\beta$
2: **for** $j = 1 : m$ **do**
3: $w_{j+1} = Aq_j$
4: MGS to orthogonalize w_{j+1} against $\{q_1, \dots, q_j\}$
5: Obtain $[r_0, AQ_j] = Q_{j+1}[\|r_0\|_2 e_1, \bar{H}_j]$
6: **end for**
7: Solve $y_m = argmin_y \|\beta e_1 - \bar{H}_m y\|_2$
8: $x = x_0 + Q_m y_m$

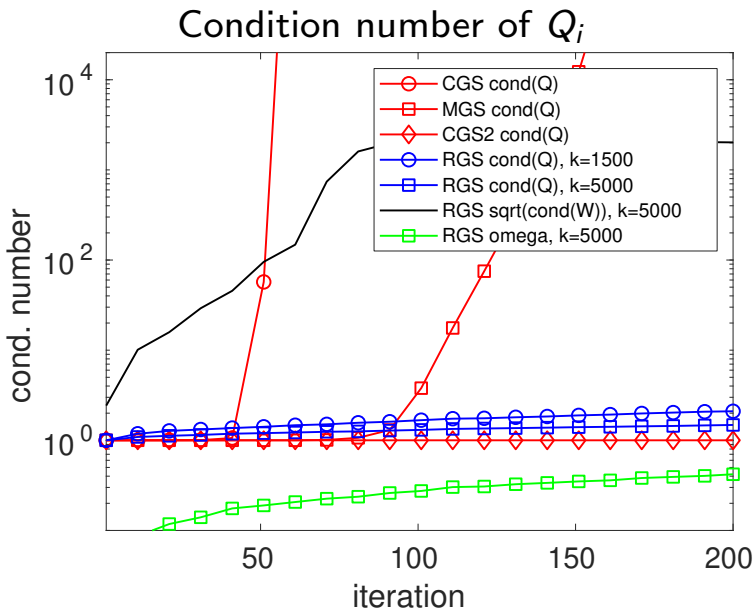
Solution y_m (step 7) minimizes $\|A(x_0 + Q_m y) - b\|_2$

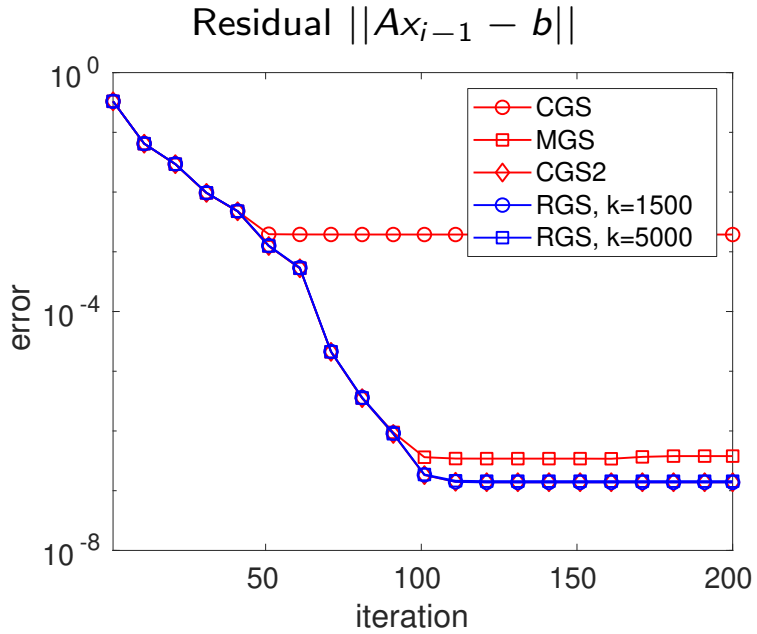
Algorithm 6 GMRES with randomized GS

1: $r_0 = b - Ax_0, \beta = \|r_0\|_2, q_1 = r_0/\beta$
2: **for** $j = 1 : m$ **do**
3: $w_{j+1} = Aq_j$
4: Randomized GS to orthogonalize w_{j+1} against $\{q_1, \dots, q_j\}$
5: Obtain $[r_0, AQ_j] = Q_{j+1}[\|r_0\|_2 e_1, \bar{H}_j]$
6: **end for**
7: Solve $y_m = argmin_y \|\beta e_1 - \bar{H}_m y\|_2$
8: $x = x_0 + Q_m y_m$

Solution y_m (step 7) minimizes $\|\Omega(A(x_0 + Q_m y) - b)\|_2$

GMRES numerical experiments





- Matrix SiO2 (pseudopotential method) from Tim Davis's collection, dimensions 155331×155331, preconditioned with ILU(0)
- Results obtained in Matlab, $W_i = [AQ_{i-1}, b]$
- Hessenberg least-squares problem and products with A in fp64
- GS iterations in fp32

Plan

Sparse linear solvers

Orthogonalization techniques: deterministic and randomized

Krylov subspace methods

Communication avoiding in iterative methods

CA solvers based on s-step methods: main idea

To avoid communication, unroll k-steps, ghost necessary data,

- generate a set of vectors W for the Krylov subspace $\mathcal{K}_k(A, r_0)$,
- (A)-orthogonalize the vectors using a communication avoiding orthogonalization algorithm (e.g. TSQR(W)).

References

- Van Rosendale '83, Walker '85, Chronopoulous and Gear '89, Erhel '93, Toledo '95, Bai, Hu, Reichel '91 (Newton basis), Joubert and Carey '92 (Chebyshev basis), etc.
- CA references: J. Demmel, M. Hoemmen, M. Mohiyuddin, K. Yellick '2009, Carson, Demmel, Knight (2013, 2015other Krylov solvers, preconditioners)

CA-GMRES

GMRES: find x in $span\{v, Av, ..., A^k v\}$ minimizing $||Ax - b||_2$
Cost of k steps of standard GMRES vs new GMRES

Standard GMRES
 for i=1 to k
  w = A · v(i-1)
  MGS(w, v(0),...,v(i-1))
  update v(i), H
 endfor
 solve LSQ problem with H

Sequential: #words_moved =
  O(k·nnz) from SpMV
 + O(k^2·n) from MGS
Parallel: #messages =
  O(k) from SpMV
 + O(k^2 · log p) from MGS

Slide source: J. Demmel

CA-GMRES

GMRES: find x in $span\{v, Av, ..., A^k v\}$ minimizing $||Ax - b||_2$
Cost of k steps of standard GMRES vs new GMRES

Standard GMRES
for i=1 to k
w = A · v(i-1)
MGS(w, v(0),...,v(i-1))
update v(i), H
endfor
solve LSQ problem with H

Sequential: #words_moved =
O(k·nnz) from SpMV
+ O(k^2·n) from MGS
Parallel: #messages =
O(k) from SpMV
+ O(k^2 · log p) from MGS

Communication-avoiding GMRES
W = [v, Av, A^2v, ... , A^kv]
[Q,R] = TSQR(W) ... "Tall Skinny QR"
Build H from R, solve LSQ problem

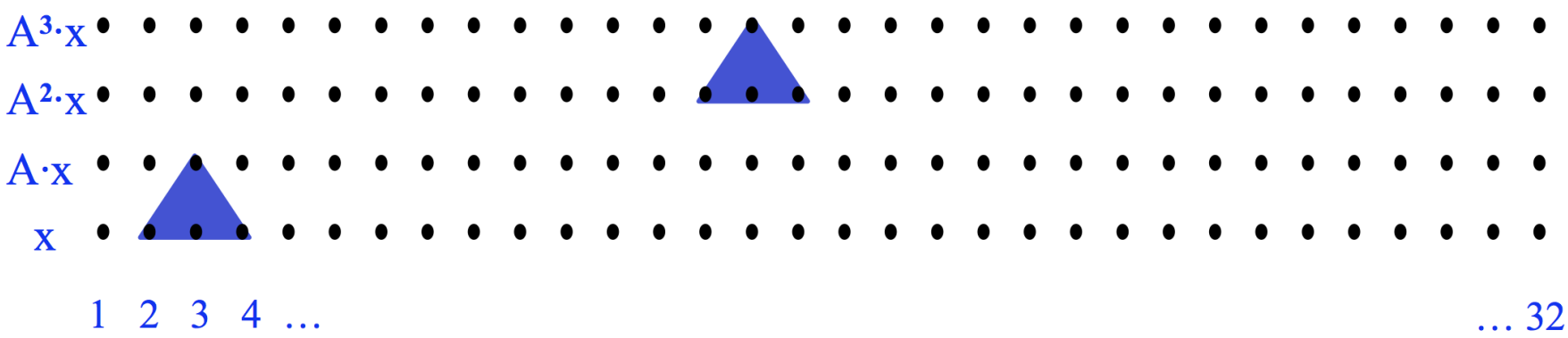
Sequential: #words_moved =
O(nnz) from SpMV
+ O(k·n) from TSQR
Parallel: #messages =
O(1) from computing W
+ O(log p) from TSQR

Slide source: J. Demmel

Matrix Powers Kernel

- Generate the set of vectors $\{Ax, A^2x, \ldots A^kx\}$ in parallel
- Ghost necessary data to avoid communication
- Example: A tridiagonal, $n = 32$, $s = 3$
- Shaded triangles represent data computed redundantly

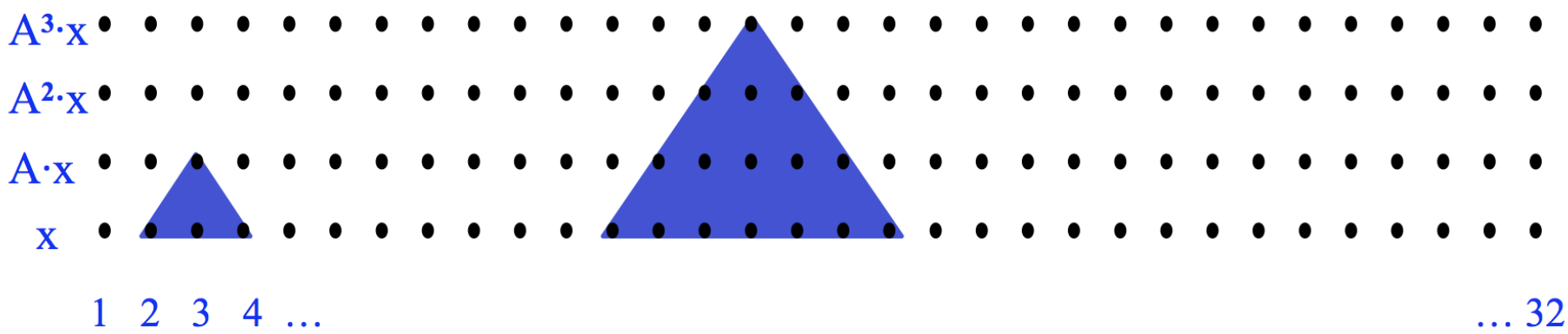
$$Ax = \begin{pmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & \ddots & \ddots & \ddots & \end{pmatrix} \cdot \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix}$$



Matrix Powers Kernel

- Generate the set of vectors $\{Ax, A^2x, \ldots A^kx\}$ in parallel
- Ghost necessary data to avoid communication
- Example: A tridiagonal, $n = 32$, $s = 3$
- Shaded triangles represent data computed redundantly

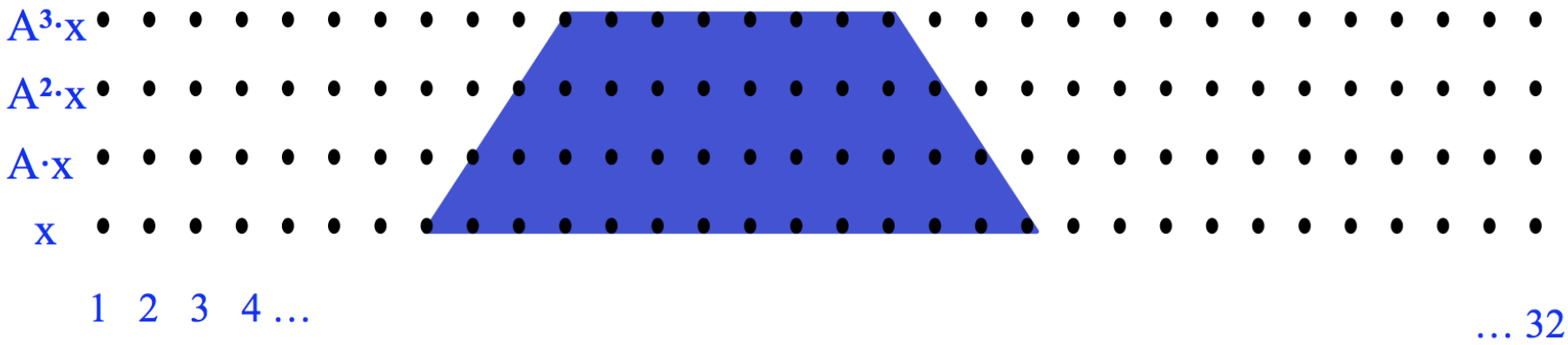
$$Ax = \begin{pmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \cdot \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix}$$



Matrix Powers Kernel

- Generate the set of vectors $\{Ax, A^2x, \ldots A^kx\}$ in parallel
- Ghost necessary data to avoid communication
- Example: A tridiagonal, $n = 32$, $s = 3$
- Shaded triangles represent data computed redundantly

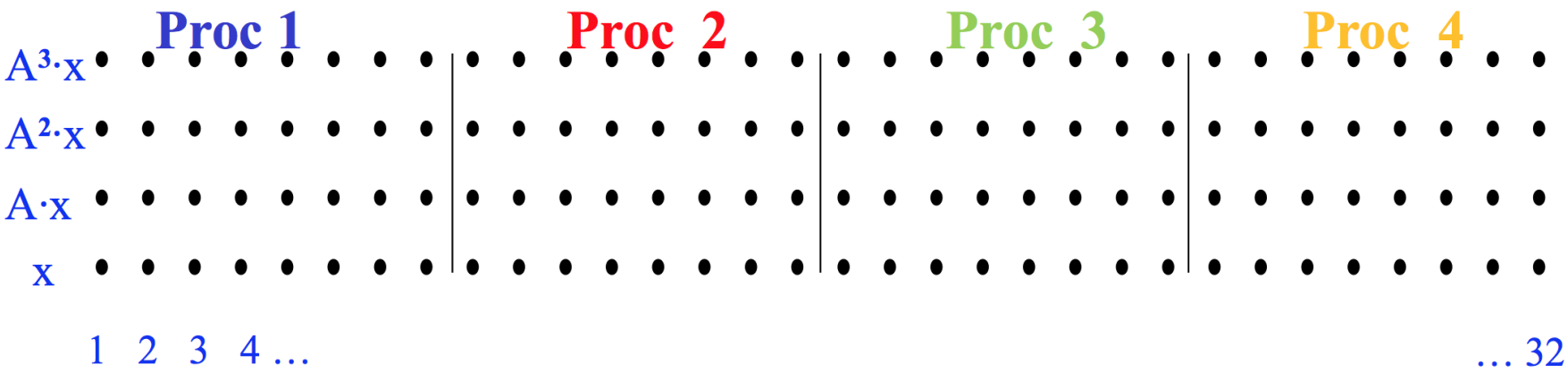
$$Ax = \begin{pmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \cdot \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix}$$



Matrix Powers Kernel

- Generate the set of vectors $\{Ax, A^2x, \ldots A^kx\}$ in parallel
- Ghost necessary data to avoid communication
- Example: A tridiagonal, $n = 32$, $s = 3$
- Shaded triangles represent data computed redundantly

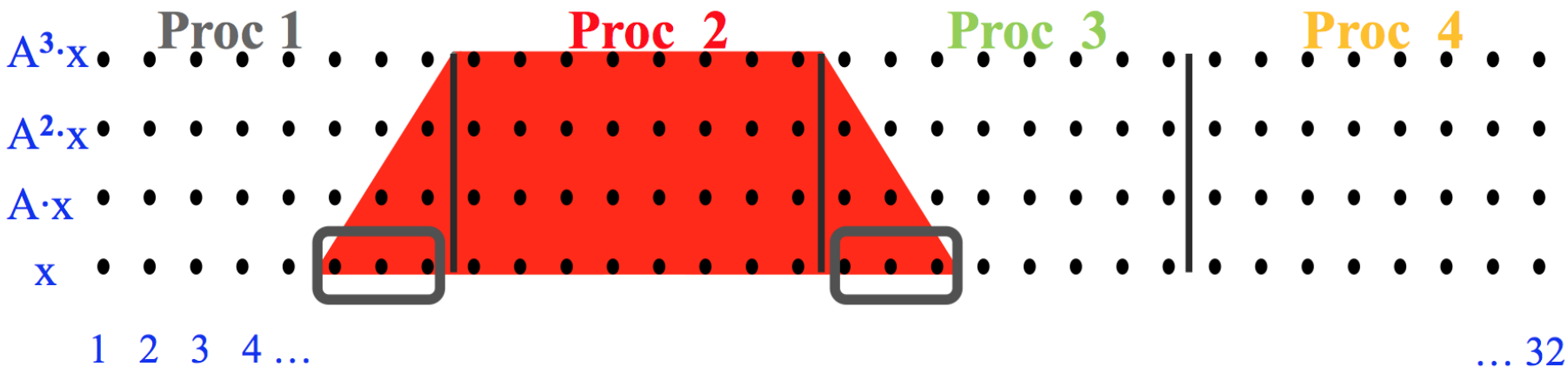
$$Ax = \begin{pmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \cdot \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix}$$



Matrix Powers Kernel

- Generate the set of vectors $\{Ax, A^2x, \ldots A^kx\}$ in parallel
- Ghost necessary data to avoid communication
- Example: A tridiagonal, $n = 32$, $s = 3$
- Shaded triangles represent data computed redundantly

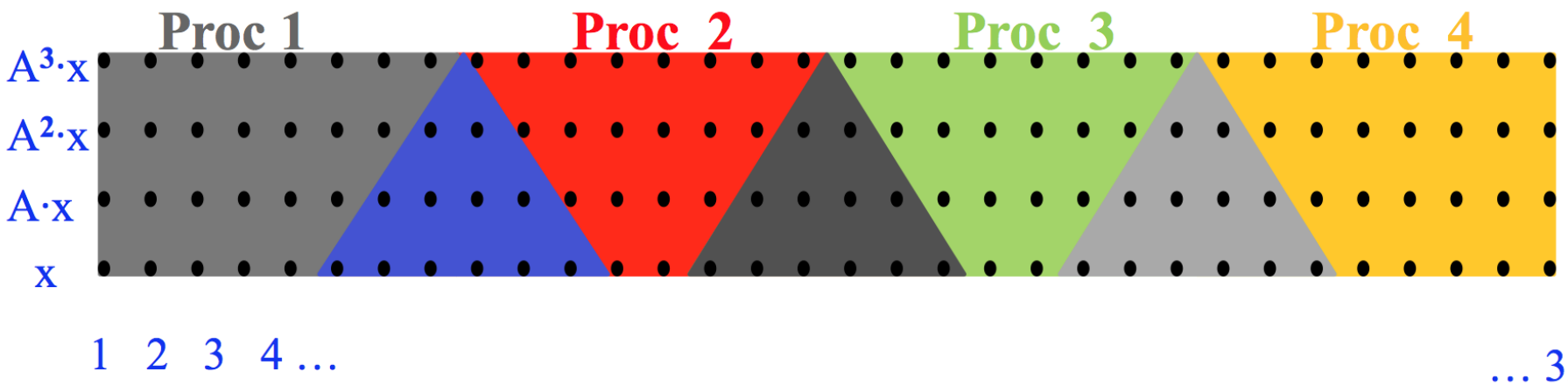
$$Ax = \begin{pmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & \ddots & \ddots & \ddots & \end{pmatrix} \cdot \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix}$$



Matrix Powers Kernel

- Generate the set of vectors $\{Ax, A^2x, \ldots A^kx\}$ in parallel
- Ghost necessary data to avoid communication
- Example: A tridiagonal, $n = 32$, $s = 3$
- Shaded triangles represent data computed redundantly

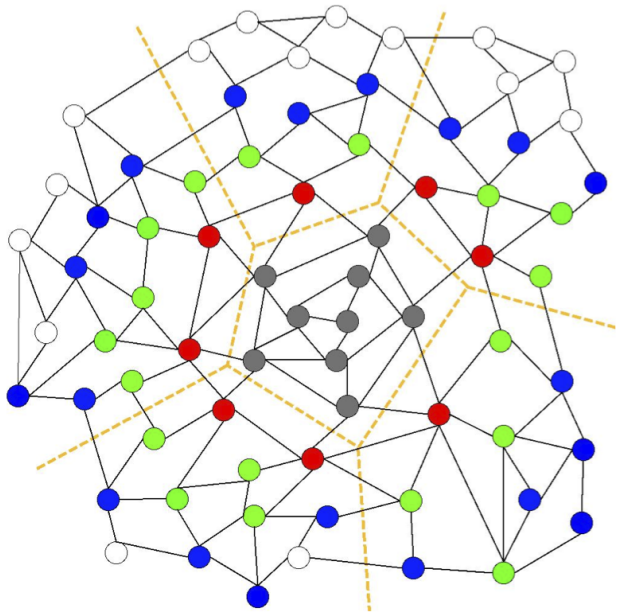
$$Ax = \begin{pmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & \ddots & \ddots & \ddots \end{pmatrix} \cdot \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix} = \begin{pmatrix} * \\ * \\ * \\ * \\ \vdots \end{pmatrix}$$



Matrix Powers Kernel (contd)

Ghosting works for structured or well-partitioned unstructured matrices, with modest surface-to-volume ratio.

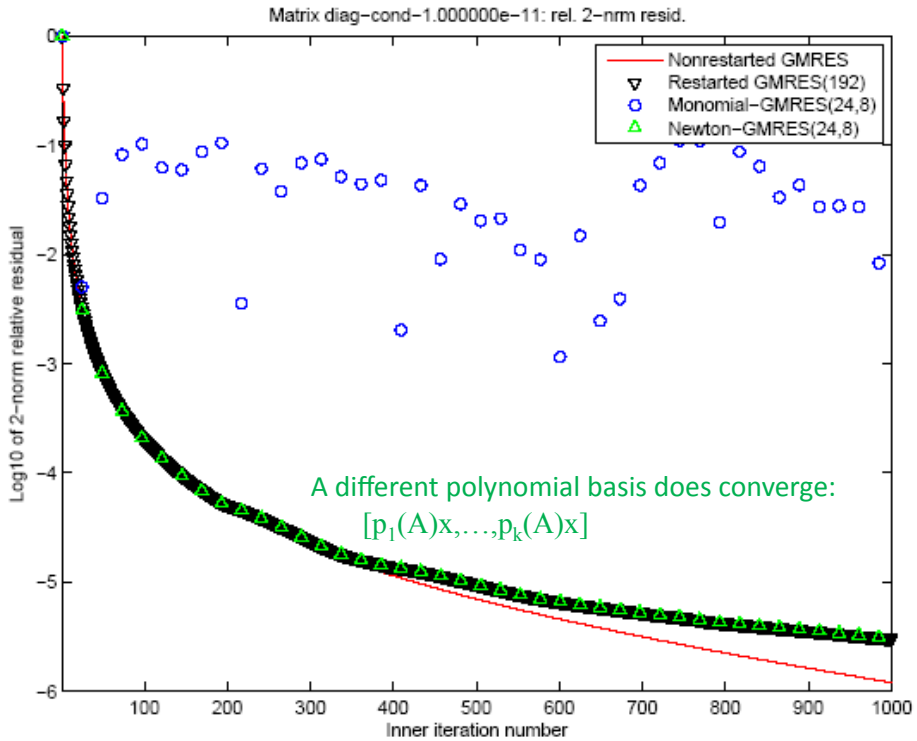
- Parallel: block-row partitioning based on (hyper)graph partitioning,
- Sequential: top-to-bottom processing based on traveling salesman problem.



Challenges and research opportunities

Length of the basis k is limited by

- Size of ghost data
- Loss of precision
  - replace W with $x, p_1(A)x, \ldots p_k(A)x$



References (1)

Balabanov, O. and Grigori, L. (2020).

Randomized gram-schmidt process with application to gmres.

Balabanov, O. and Grigori, L. (2022).

Randomized gram–schmidt process with application to gmres.
SIAM Journal on Scientific Computing, 44(3):A1450–A1474.

Demmel, J., Hoemmen, M., Mohiyuddin, M., and Yelick, K. (2009).

Minimizing communication in sparse matrix solvers.
In *Proceedings of the ACM/IEEE Supercomputing SC9 Conference*.

Giraud, L., Langou, J., Rozložník, M., and Eshof, J. v. d. (2005).

Rounding error analysis of the classical gram-schmidt orthogonalization process.
Numerische Mathematik, 101(1):87–100.

Saad, Y. (2003).

Iterative Methods for Sparse Linear Systems.
Society for Industrial and Applied Mathematics, second edition.