Sparse linear solvers: iterative methods

L. Grigori

ALPINES
INRIA and LJLL, Sorbonne Université

December 2018
Plan

Sparse linear solvers
  Sparse matrices and graphs
  Classes of linear solvers

Krylov subspace methods
  Conjugate gradient method

Enlarged Krylov methods
  Definition and properties
  Numerical and parallel performance results

Preconditioned Krylov subspace methods
  One level Additive Schwarz methods
  Two level preconditioners
Plan

Sparse linear solvers
  Sparse matrices and graphs
  Classes of linear solvers

Krylov subspace methods

Enlarged Krylov methods

Preconditioned Krylov subspace 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, \( n = 100196 \), \( \text{nnz}(A) = 544688 \)

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, $n = 105676$, $\text{nnz}(A) = 513072$

![Nonzero structure of the matrix](image1.png)

![Its undirected graph](image2.png)

**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 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
Plan

Sparse linear solvers

**Krylov subspace methods**
  Conjugate gradient method

Enlarged Krylov methods

Preconditioned Krylov subspace methods
Krylov subspace methods

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

$$x_0 + K_i(A, r_0), \quad i = 1, ..., k$$

They are defined by two conditions:

1. Subspace condition: $x_k \in x_0 + K_k(A, r_0)$
2. Petrov-Galerkin condition: $r_k \perp L_k$

$$\iff (r_k)^t y = 0, \quad \forall \ y \in L_k$$

where

- $x_0$ is the initial iterate, $r_0$ is the initial residual,
- $K_k(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, ..., A^{k-1}r_0\}$ is the Krylov subspace of dimension $k$,
- $L_k$ is a well-defined subspace of dimension $k$. 
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
A Krylov projection method for SPD matrices where \( \mathcal{L}_k = \mathcal{K}_k(A, r_0) \).

Finds \( x^* = A^{-1}b \) by minimizing the quadratic function

\[
\phi(x) = \frac{1}{2} (x)^t Ax - b^t x \\
\nabla \phi(x) = Ax - b = 0
\]

After \( j \) iterations of CG,

\[
||x^* - x_j||_A \leq 2||x - x_0||_A \left( \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^j,
\]

where \( x_0 \) is starting vector, \( ||x||_A = \sqrt{x^T Ax} \) and \( \kappa(A) = |\lambda_{max}(A)|/|\lambda_{min}(A)|. \)
Conjugate gradient

- Computes A-orthogonal search directions by conjugation of the residuals

\[
\begin{align*}
    p_1 &= r_0 = -\nabla \phi(x_0) \\
    p_k &= r_{k-1} + \beta_k p_{k-1}
\end{align*}
\]  

(1)

- At k-th iteration,

\[
x_k = x_{k-1} + \alpha_k p_k = \text{argmin}_{x \in x_0 + K_k(A, r_0)} \phi(x)
\]

where \( \alpha_k \) is the step along \( p_k \).

- CG algorithm obtained by imposing the orthogonality and the conjugacy conditions

\[
\begin{align*}
    r_k^T r_i &= 0, \text{ for all } i \neq k, \\
    p_k^T A p_i &= 0, \text{ for all } i \neq k.
\end{align*}
\]
Since we have \( x_k = x_{k-1} + \alpha_k p_k \) we obtain

\[
r_k = r_{k-1} - \alpha_k Ap_k \quad \text{and} \quad (r_k, r_{k-1}) = 0 \quad \text{hence}
\]

\[
r_{k-1}^T r_{k-1} - \alpha_k r_{k-1}^T Ap_k = 0 \quad \implies \quad \alpha_k = \frac{(r_{k-1}, r_{k-1})}{(Ap_k, r_{k-1})}
\]

Since we have \( p_k = r_{k-1} + \beta_k p_{k-1} \),

\[
(Ap_k, r_{k-1}) = (Ap_k, p_k - \beta_k p_{k-1}) = (Ap_k, p_k) \quad \implies \quad \alpha_k = \frac{(r_{k-1}, r_{k-1})}{(Ap_k, p_k)}
\]

Since \( p_k = r_{k-1} + \beta_k p_{k-1} \) is A-orthogonal to \( p_{k-1} \) we obtain

\[
\beta_k = -\frac{(r_{k-1}, Ap_{k-1})}{(p_{k-1}, Ap_{k-1})} \quad \text{and} \quad Ap_{k-1} = \frac{1}{\alpha_{k-1}} (r_{k-1} - r_k) \quad \implies \quad \beta_k = \frac{(r_{k-1}, r_{k-1})}{(r_{k-2}, r_{k-2})}
\]
Algorithm 1 The CG Algorithm

1: \( r_0 = b - Ax_0, \rho_0 = \|r_0\|_2^2, p_1 = r_0, k = 1 \)
2: \( \textbf{while} \left( \sqrt{\rho_k} > \epsilon \|b\|_2 \text{ and } k < k_{\text{max}} \right) \textbf{do} \)
3: \( \textbf{if } (k \neq 1) \textbf{ then} \)
4: \( \beta_k = (r_{k-1}, r_{k-1})/(r_{k-2}, r_{k-2}) \)
5: \( p_k = r_{k-1} + \beta_k p_{k-1} \)
6: \( \textbf{end if} \)
7: \( \alpha_k = (r_{k-1}, r_{k-1})/(Ap_k, p_k) \)
8: \( x_k = x_{k-1} + \alpha_k p_k \)
9: \( r_k = r_{k-1} - \alpha_k Ap_k \)
10: \( \rho_k = \|r_k\|_2^2 \)
11: \( k = k + 1 \)
12: \( \textbf{end while} \)
Challenge in getting efficient and scalable solvers

- A Krylov solver finds $x_{k+1}$ from $x_0 + \mathcal{K}_{k+1}(A, r_0)$ where

$$\mathcal{K}_{k+1}(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \ldots, A^kr_0\},$$

such that the Petrov-Galerkin condition $b - Ax_{k+1} \perp \mathcal{L}_{k+1}$ is satisfied.

- Does a sequence of $k$ SpMVs to get vectors $[x_1, \ldots, x_k]$

- Finds best solution $x_{k+1}$ as linear combination of $[x_1, \ldots, x_k]$

Typically, each iteration requires

- Sparse matrix vector product $\rightarrow$ point-to-point communication

- Dot products for orthogonalization $\rightarrow$ global communication

![Graph showing overall runtime, communication, and computation]

Map making, with R. Stompor, M. Szydlarski
Results obtained on Hopper, Cray XE6, NERSC
A Krylov solver finds $x_{k+1}$ from $x_0 + \mathcal{K}_{k+1}(A, r_0)$ where

$$\mathcal{K}_{k+1}(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^k r_0\},$$

such that the Petrov-Galerkin condition $b - Ax_{k+1} \perp \mathcal{L}_{k+1}$ is satisfied.

- Does a sequence of $k$ SpMVs to get vectors $[x_1, \ldots, x_k]$
- Finds best solution $x_{k+1}$ as linear combination of $[x_1, \ldots, x_k]$

Typically, each iteration requires

- Sparse matrix vector product \rightarrow point-to-point communication
- Dot products for orthogonalization \rightarrow global communication
Ways to improve performance

■ Improve the performance of sparse matrix-vector product.

■ Improve the performance of collective 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

Krylov subspace methods

**Enlarged Krylov methods**
- Definition and properties
- Numerical and parallel performance results

Preconditioned Krylov subspace methods
Enlarged Krylov methods [LG, Moufawad, Nataf, 14]

- Partition the matrix into $N$ domains
- Split the residual $r_0$ into $t$ vectors corresponding to the $N$ domains, obtain $R_0^e$.

- Generate $t$ new basis vectors, obtain an enlarged Krylov subspace

$$\mathcal{K}_{t,k}(A, r_0) = \text{span}\{R_0^e, AR_0^e, A^2R_0^e, ..., A^{k-1}R_0^e\}$$

- Search for the solution of the system $Ax = b$ in $\mathcal{K}_{t,k}(A, r_0)$
Properties of enlarged Krylov subspaces

- The Krylov subspace $\mathcal{K}_k(A, r_0)$ is a subset of the enlarged one
  \[ \mathcal{K}_k(A, r_0) \subset \mathcal{K}_{t,k}(A, r_0) \]

- For all $k < k_{max}$ the dimensions of $\mathcal{K}_{t,k}$ and $\mathcal{K}_{t,k+1}$ are strictly increasing by some number $i_k$ and $i_{k+1}$ respectively, where
  \[ t \geq i_k \geq i_{k+1} \geq 1. \]

- The enlarged subspaces are increasing subspaces, yet bounded.
  \[ \mathcal{K}_{t,1}(A, r_0) \subsetneq \ldots \subsetneq \mathcal{K}_{t,k_{max}-1}(A, r_0) \subsetneq \mathcal{K}_{t,k_{max}}(A, r_0) = \mathcal{K}_{t,k_{max}+q}(A, r_0), \forall q > 0 \]

- The solution of the system $Ax = b$ belongs to the subspace $x_0 + \mathcal{K}_{t,k_{max}}$. 
Enlarged Krylov subspace methods based on CG

Defined by the subspace $\mathcal{K}_{t,k}$ and the following two conditions:

1. Subspace condition: $x_k \in x_0 + \mathcal{K}_{t,k}$
2. Orthogonality condition: $r_k \perp \mathcal{K}_{t,k}$

At each iteration, the new approximate solution $x_k$ is found by minimizing $\phi(x) = \frac{1}{2}(x)^tAx - b^tx$ over $x_0 + \mathcal{K}_{t,k}$:

$$\phi(x_k) = \min\{\phi(x), \forall x \in x_0 + \mathcal{K}_{t,k}(A, r_0)\}$$

Can be seen as a particular case of a block Krylov method

- $AX = S(b)$, such that $S(b)\text{ones}(t, 1) = b$; $R_0^e = AX_0 - S(b)$
- Orthogonality condition involves the block residual $R_k \perp \mathcal{K}_{t,k}$
Enlarged Krylov subspace methods based on CG

Defined by the subspace $\mathcal{K}_{t,k}$ and the following two conditions:

1. **Subspace condition:** $x_k \in x_0 + \mathcal{K}_{t,k}$
2. **Orthogonality condition:** $r_k \perp \mathcal{K}_{t,k}$

- At each iteration, the new approximate solution $x_k$ is found by minimizing $\phi(x) = \frac{1}{2} (x)^t A x - b^t x$ over $x_0 + \mathcal{K}_{t,k}$:

$$\phi(x_k) = \min \{ \phi(x), \forall x \in x_0 + \mathcal{K}_{t,k}(A, r_0) \}$$

- Can be seen as a particular case of a block Krylov method
  - $AX = S(b)$, such that $S(b)\text{ones}(t, 1) = b$; $R_0^e = AX_0 - S(b)$
  - Orthogonality condition involves the block residual $R_k \perp \mathcal{K}_{t,k}$
Related work

- **Block Krylov methods** [O’Leary, 1980]: solve systems with multiple rhs

\[ AX = B, \]

by searching for an approximate solution \( X_k \in X_0 + \mathcal{K}_{k}(A, R_0), \)

\[ \mathcal{K}_{k}(A, R_0) = \text{block-span}\{ R_0, AR_0, A^2R_0, ..., A^{k-1}R_0 \}. \]

- coopCG (Bhaya et al, 2012): solve one system by starting with \( t \) different initial guesses
- BRRHS-CG [Nikishin and Yeremin, 1995]: use a block method with \( t-1 \) random right hand sides

- **Multiple preconditioners**
  - GMRES with multiple preconditioners [Greif, Rees, Szyld, 2011]
  - AMPFETI [Rixen, 97], [Gosselet et al, 2015]

- And to reduce communication: s-step methods, pipelined methods
Convergence analysis

Given

- $A$ is an SPD matrix, $x^*$ is the solution of $Ax = b$
- $||x^* - x_k||_A$ is the $k^{th}$ error of CG
- $||x^* - x_k||_A$ is the $k^{th}$ error of ECG

Result

\[
\begin{align*}
\text{CG} & \quad ||x^* - x_k||_A \leq 2||x^* - x_0||_A \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \\
\text{ECG} & \quad ||x^* - x_k||_A \leq C||x^* - x_0||_A \left( \frac{\sqrt{\kappa_t} - 1}{\sqrt{\kappa_t} + 1} \right)^k
\end{align*}
\]

where $\kappa = \frac{\lambda_{\text{max}}(A)}{\lambda_{\text{min}}(A)}$

From here on, results on ECG obtained with O. Tissot

\[
C \text{ is a const indpdt. of } k, \text{ dpdt. of } t
\]
Algorithm 2 Classical CG

1: \( p_1 = r_0 (r_0^T A r_0)^{-1/2} \)
2: while \( \| r_{k-1} \|_2 > \varepsilon \| b \|_2 \) do
3: \( \alpha_k = p_k^T r_{k-1} \)
4: \( x_k = x_{k-1} + p_k \alpha_k \)
5: \( r_k = r_{k-1} - A p_k \alpha_k \)
6: \( z_{k+1} = r_k - p_k (p_k^T A r_k) \)
7: \( p_{k+1} = z_{k+1} (z_{k+1}^T A z_{k+1})^{-1/2} \)
8: end while

Cost per iteration
\# flops = \( O\left(\frac{n^2}{P}\right) \) ← BLAS 1 & 2
\# words = \( O(1) \)
\# messages = \( O(1) \) from SpMV + \( O(\log P) \) from dot prod + norm

Algorithm 3 ECG

1: \( P_1 = R_0^e (R_0^e)^T A R_0^e \)^{-1/2}
2: while \( \| \sum_{i=1}^{T} R_{k}^{(i)} \|_2 < \varepsilon \| b \|_2 \) do
3: \( \alpha_k = P_k^T R_{k-1} \) \( \text{\textarrow}{t \times t \text{matrix}} \)
4: \( X_k = X_{k-1} + P_k \alpha_k \)
5: \( R_k = R_{k-1} - A P_k \alpha_k \)
6: Construct \( Z_{k+1} \) s.t. \( Z_{k+1}^T A P_i = 0, \forall i \leq k \)
7: \( P_{k+1} = Z_{k+1} (Z_{k+1}^T A Z_{k+1})^{-1/2} \)
8: end while
9: \( x = \sum_{i=1}^{T} X_k^{(i)} \)

Cost per iteration
\# flops = \( O\left(\frac{nt^2}{P}\right) \) ← BLAS 3
\# words = \( O(t^2) \) ← Fit in the buffer
\# messages = \( O(1) \) from SpMV + \( O(\log P) \) from A-ortho
Construction of the search directions $P_{k+1}$

1. Construct $Z_{k+1}$ s.t. $Z_{k+1}^T A P_i = 0$, $\forall i \leq k$ by using:
   1.a Orthomin as in block CG [OLeary., 1980] and original CG method [Hestenes and Stiefel., 1952]:
      \[
      Z_{k+1} = R_k - P_k (P_k^T A R_k)
      \]
   1.b or Orthodir as in ECG [Grigori et al., 2016], based on Lanczos formula [Ashby et al., 1990]:
      \[
      Z_{k+1} = A P_k - P_k (P_k^T A A P_k) - P_{k-1} (P_{k-1}^T A A P_k)
      \]

2. A-orthonormalize $P_{k+1}$, using e.g. A Cholesky QR:
   \[
   P_{k+1} = Z_{k+1} (Z_{k+1}^T A Z_{k+1})^{-1/2}
   \]

Orthomin (Omin) \quad Orthodir (Odir)

$\rightarrow$ Cheaper \quad $\rightarrow$ More expensive
$\rightarrow$ In practice breakdowns \quad $\rightarrow$ In practice no breakdowns
Orthomin (Omin) versus Orthodir (Odir)

Both rely on same projection process

- $\hat{X}_k = \tilde{X}_k$ and $\hat{R}_k = \tilde{R}_k$
- $\hat{P}_k \neq \tilde{P}_k$ and $\hat{Z}_k \neq \tilde{Z}_k$

With a tilde $\to$ Omin variables
With a hat $\to$ Odir variables

Proposition
Assume no breakdown occurred, then there exists orthogonal matrix $\delta_k$ st:

$$\tilde{P}_k = \hat{P}_k \delta_k$$
$$\tilde{Z}_{k+1} = -\hat{Z}_{k+1} \delta_k \tilde{\alpha}_k,$$ where $\tilde{\alpha}_k = \tilde{P}_k^T \tilde{R}_{k-1}$

- Generalization of result in [Ashby et al., 1990]; explicit link between Lanczos and CG
- When $k$ is large, $||\tilde{\alpha}_k||_2$ becomes small, hence $||\tilde{Z}_{k+1}||_2 < ||\hat{Z}_{k+1}||_2$
- The conditioning of $\tilde{Z}_{k+1}^T A \tilde{Z}_{k+1}$ can be worse than that of $\hat{Z}_{k+1}^T A \hat{Z}_{k+1}$!
Orthomin (Omin) versus Orthodir (Odir)

Both rely on same projection process

- $\hat{X}_k = \tilde{X}_k$ and $\hat{R}_k = \tilde{R}_k$

- With a tilde $\rightarrow$ Omin variables

- With a hat $\rightarrow$ Odir variables

Proposition

Assume no breakdown occurred, then there exists orthogonal matrix $\delta_k$ st:

$$\tilde{P}_k = \hat{P}_k \delta_k$$

$$\tilde{Z}_{k+1} = -\hat{Z}_{k+1} \delta_k \tilde{\alpha}_k, \text{ where } \tilde{\alpha}_k = \tilde{P}_k^T \tilde{R}_{k-1}$$

- Generalization of result in [Ashby et al., 1990]; explicit link between Lanczos and CG

- When $k$ is large, $||\tilde{\alpha}_k||_2$ becomes small, hence $||\tilde{Z}_{k+1}||_2 < ||\hat{Z}_{k+1}||_2$

- The conditioning of $\tilde{Z}_{k+1}^T A \tilde{Z}_{k+1}$ can be worse than that of $\hat{Z}_{k+1}^T A \hat{Z}_{k+1}$
Orthomin (Omin) versus Orthodir (Odir)

Both rely on same projection process
- \( \hat{X}_k = \tilde{X}_k \) and \( \hat{R}_k = \tilde{R}_k \)
- !! \( \hat{P}_k \neq \tilde{P}_k \) and \( \hat{Z}_k \neq \tilde{Z}_k \)
- With a tilde → Omin variables
- With a hat → Odir variables

Proposition
Assume no breakdown occurred, then there exists orthogonal matrix \( \delta_k \) st:

\[
\tilde{P}_k = \hat{P}_k \delta_k \\
\tilde{Z}_{k+1} = -\hat{Z}_{k+1} \delta_k \tilde{\alpha}_k, \text{ where } \tilde{\alpha}_k = \tilde{P}_k^T \tilde{R}_{k-1}
\]

- Generalization of result in [Ashby et al., 1990]; explicit link between Lanczos and CG
- When \( k \) is large, \( \|\tilde{\alpha}_k\|_2 \) becomes small, hence \( \|\tilde{Z}_{k+1}\|_2 < \|\hat{Z}_{k+1}\|_2 \)
- The conditioning of \( \tilde{Z}_{k+1}^T A \tilde{Z}_{k+1} \) can be worse than that of \( \hat{Z}_{k+1}^T A \hat{Z}_{k+1} \)!
Test cases: boundary value problem

2D and 3D Skyscraper Problem - SKY2D,3D

\[-\text{div}(\kappa(x) \nabla u) = f \text{ in } \Omega \]
\[u = 0 \text{ on } \partial \Omega_D \]
\[\frac{\partial u}{\partial n} = 0 \text{ on } \partial \Omega_N \]

discretized on a 3D grid, where

\[\kappa(x) = \begin{cases} 
10^3 \times ([10 \times x_2] + 1), & \text{if } [10 \times x_i] = 0 \text{ mod}(2), \ i = 1, 2, 3, \\
1, & \text{otherwise}.
\end{cases} \]
Test cases (contd)

Linear elasticity 3D problem

\[
\begin{align*}
\text{div}(\sigma(u)) + f &= 0 \quad \text{on } \Omega, \\
u &= u_D \quad \text{on } \partial\Omega_D, \\
\sigma(u) \cdot n &= g \quad \text{on } \partial\Omega_N,
\end{align*}
\]

- \( u \in \mathbb{R}^d \) is the unknown displacement field, \( f \) is some body force.
- Young’s modulus \( E \) and Poisson’s ratio \( \nu \) take two values, \((E_1, \nu_1) = (2 \cdot 10^{11}, 0.25), \) and \((E_2, \nu_2) = (10^7, 0.45). \)
- Cauchy stress tensor \( \sigma(u) \) is given by Hooke’s law, defined by \( E \) and \( \nu \).

Matrices Generated with FreeFem++ (F. Hecht, Sorbonne Université)
Linear Elasticity discretized using \( \mathbb{P}_1 \) FE, 1600 \( \times \) \( Y \) \( \times \) \( Y \) grid
Enlarged CG: numerical results

- Block Jacobi preconditioner (1024 blocks)
- Stopping criterion $10^{-6}$, initial block size 32
- BRRHS-CG: block method with $t - 1$ random rhs

<table>
<thead>
<tr>
<th>matrix</th>
<th>$n(A)$</th>
<th>$nnz(A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKY2D</td>
<td>10,000</td>
<td>49,600</td>
</tr>
<tr>
<td>Ela3D100</td>
<td>36,663</td>
<td>1,231,497</td>
</tr>
<tr>
<td>Ela2D200</td>
<td>80,802</td>
<td>964,800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>red. size</th>
<th>PCG iter</th>
<th>BRRHS-CG iter</th>
<th>dim($K_{32,k}$)</th>
<th>ECG iter</th>
<th>dim($K_{32,k}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKY2D</td>
<td>×</td>
<td>655</td>
<td>61</td>
<td>1952</td>
<td>57</td>
<td>1824</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>655</td>
<td>61</td>
<td>1739</td>
<td>59</td>
<td>1546</td>
</tr>
<tr>
<td>Ela3D100</td>
<td>×</td>
<td>955</td>
<td>102</td>
<td>3264</td>
<td>109</td>
<td>3488</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>955</td>
<td>102</td>
<td>3093</td>
<td>116</td>
<td>2384</td>
</tr>
<tr>
<td>Ela2D200</td>
<td>×</td>
<td>4551</td>
<td>255</td>
<td>8160</td>
<td>253</td>
<td>8096</td>
</tr>
<tr>
<td></td>
<td>✓</td>
<td>4551</td>
<td>258</td>
<td>7331</td>
<td>266</td>
<td>6553</td>
</tr>
</tbody>
</table>
Enlarged CG: dynamic reduction of search directions

- Block Jacobi preconditioner (112 blocks)
- Stopping criterion $10^{-5}$, initial block size $t = 24$
- Odir-0/1: orthodir without/with dynamic reduction of search directions (difference of 4 iterations to converge)
- Ela_20: 3D elasticity, $n = 2, 118, 123$, $nnz = 74, 735, 397$

![Graph showing convergence with and without dynamic reduction](image)
Enlarged CG: parallel performance

- Stopping criterion $10^{-5}$, blocks Jacobi = #MPI
- Performance study on:
  - Kebnekaise (Suede), Intel Xeon (Broadwell), 28 MPI process/node
  - Cori NERSC, Intel KNL, 68 cores each

<table>
<thead>
<tr>
<th># MPI</th>
<th># iter</th>
<th>res</th>
<th># iter</th>
<th>res</th>
</tr>
</thead>
<tbody>
<tr>
<td>252</td>
<td>513</td>
<td>1.3E-4</td>
<td>13,626</td>
<td>1.3E-4</td>
</tr>
<tr>
<td>504</td>
<td>531</td>
<td>1.9E-4</td>
<td>15,819</td>
<td>1.9E-4</td>
</tr>
<tr>
<td>1,008</td>
<td>606</td>
<td>2.6E-4</td>
<td>17,023</td>
<td>2.7E-4</td>
</tr>
<tr>
<td>2,016</td>
<td>696</td>
<td>2.6E-4</td>
<td>19,047</td>
<td>2.7E-4</td>
</tr>
</tbody>
</table>

Time (s)

- Ideal
- ECG D-Odir(24)
- PETSc
Plan

Sparse linear solvers

Krylov subspace methods

Enlarged Krylov methods

Preconditioned Krylov subspace methods
  One level Additive Schwarz methods
  Two level preconditioners
Preconditioned Krylov subspace methods

- Solve by using iterative methods

\[ Ax = b. \]

- Convergence depends on \( \kappa(A) \) and the eigenvalue distribution (for SPD matrices).

- To accelerate convergence, solve

\[ M^{-1}Ax = M^{-1}b, \]

where

- \( M \) approximates well the inverse of \( A \) and/or
- improves \( \kappa(A) \), the condition number of \( A \).

- Ideally, we would like to bound \( \kappa(A) \), independently of the size of the matrix \( A \).
Additive Schwarz: notations

Solve $M^{-1}Ax = M^{-1}b$, where $A \in \mathbb{R}^{n \times n}$ is SPD

For $\mathcal{N} = \{1, \ldots, n\}$, let $\mathcal{N}_i \subset \mathcal{N}$ for $i = 1 \ldots N$ be the subset of DOF of subdomain $i$, referred to as $\Omega_i$, possibly with overlap. We define:

- The restriction operator $R_i \in \mathbb{R}^{n_i \times n}$, $R_i = I_n (\mathcal{N}_i, :)$.
- The prolongation operator, $R_i^T \in \mathbb{R}^{n \times n_i}$

- The matrix associated to domain $i$, $A_i \in \mathbb{R}^{n_i \times n_i}$,

\[
A_i = R_i AR_i^T
\]

- The algebraic partition of unity $(D_i)_{1 \leq i \leq N}$,

\[
I_n = \sum_{i=1}^{N} R_i^T D_i R_i
\]
Additive and Restrictive Additive Schwarz methods

- Original idea from Schwarz algorithm at the continuous level (Schwarz 1870)
- Restricted Additive Schwarz (Cai & Sarkis 1999) defined as
  \[ M_{\text{RAS}}^{-1} := \sum_{i=1}^{N} R_i^T D_i A_i^{-1} R_i \]
- Symmetric formulation, Additive Schwarz (1989) defined as
  \[ M_{\text{AS}}^{-1} := \sum_{i=1}^{N} R_i^T A_i^{-1} R_i \]
- In practice, RAS more efficient than AS
Two level preconditioners

Given a coarse subspace $V_0 \in \mathbb{R}^{n \times n_0}$ and $Z$ its basis, $V_0 = \text{span } Z$, let $R_0 = Z^T$, the coarse grid $R_0 A R_0^T$.

The two level AS preconditioner is

$$M_{AS,2}^{-1} := R_0^T (R_0 A R_0^T)^{-1} R_0 + \sum_{i=1}^{N} R_i^T (A_i)^{-1} R_i$$

Let $k_c$ be minimum number of distinct colors so that $(\text{span}\{R_i^T\})_{1 \leq i \leq N}$ of the same color are mutually $A$-orthogonal. The following holds (e.g. Chan and Mathew 1994)

$$\lambda_{\text{max}}(M_{AS,2}^{-1}) \leq k_c + 1$$
Convergence theory

Results from e.g. [Chan and Mathew, 1994, Dolean et al., 2015].

\[ M_{AS,2}^{-1}A := \sum_{i=0}^{N} R_i^T (A_i)^{-1} R_i A = \sum_{i=0}^{N} P_i, \] where \( P_i = R_i^T (A_i)^{-1} R_i A \)

\( P_i \) are orthogonal projection matrices in the \( A \) inner product since

\[ P_i P_i = R_i^T (A_i)^{-1} R_i A R_i^T (A_i)^{-1} R_i A = R_i^T (A_i)^{-1} R_i A = P_i \]

\[ A P_i = A R_i^T (A_i)^{-1} R_i A = P_i^T A \]

Recall that \( a(u, v) = v^T A u \) and \( \|P_i\| \leq 1 \).

\[ \lambda_{\max}(M_{AS,2}^{-1}A) = \sup_{u \in \mathbb{R}^n} \frac{a(M_{AS,2}^{-1}A u, u)}{a(u, u)} \]

\[ = \sup_{u \in \mathbb{R}^n} \sum_{i=0}^{N} \frac{a(P_i u, u)}{\|u\|_a^2} = \sup_{u \in \mathbb{R}^n} \sum_{i=0}^{N} \frac{a(P_i u, P_i u)}{\|u\|_a^2} \]

\[ \leq \sum_{i=0}^{N} \sup_{u \in \mathbb{R}^n} \frac{a(P_i u, P_i u)}{\|u\|_a^2} \leq N + 1 \]
If we define a-orthogonal projectors

\[ \tilde{P}_i = \sum_{j \in \Theta_i} P_j, \text{ for } i = 1, \ldots k_c \]

where \( \Theta_i \) is a set of indices with the same color (that is the indices denoting disjoint subdomains). We can apply the same reasoning and obtain

\[ \lambda_{\text{max}}(M_{AS,2}^{-1}A) \leq k_c + 1 \]
How to compute the coarse subspace $V_0 = \text{span } Z$

- Nicolaides 87 (CG): kernel of the operator (constant vectors) for a Poisson like problem works well

$$Z := (R_i^T D_i R_i 1)_{i=1:N}$$

$Z$ defined as in (Nicolaides 1987):

$$Z = \begin{pmatrix} 1_{\Omega_1} \\ 1_{\Omega_2} \\ \vdots \\ 1_{\Omega_N} \end{pmatrix}$$
How to compute the coarse subspace $V_0 = \text{span } Z$

- Nicolaides 87 (CG): kernel of the operator (constant vectors)
  \[ Z := (R_i^T D_i R_i 1)_{i=1:N} \]

- Other early references: [Morgan 92] (GMRES), [Chapman, Saad 92], [Kharchenko, Yeremin 92], [Burrage, Ehrel, and Pohl, 93]

- Estimations of eigenvectors corresponding to smallest eigenvalues / knowledge from the physics

- Geneo [Nataf, Spillane et al]: through solving local Gen EVPs, bounds smallest eigenvalue for standard FE and bilinear forms, SPD input matrix

<table>
<thead>
<tr>
<th>subd</th>
<th>dofs</th>
<th>AS</th>
<th>AS-ZEM $(V_0)$</th>
<th>GenEO $(V_0)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1452</td>
<td>79</td>
<td>54 (24)</td>
<td>16 (46)</td>
</tr>
<tr>
<td>8</td>
<td>29040</td>
<td>177</td>
<td>87 (48)</td>
<td>16 (102)</td>
</tr>
<tr>
<td>16</td>
<td>58080</td>
<td>378</td>
<td>145 (96)</td>
<td>16 (214)</td>
</tr>
</tbody>
</table>

$V_0$: size of the coarse space
AS-ZEM Nicolaides with rigid body motions, 6 per subd
Results for 3D elasticity problem provided by F. Nataf
How to compute the coarse subspace $V_0 = \text{span } Z$

- Nicolaides 87 (CG): kernel of the operator (constant vectors)

$$Z := (R_i^T D_i R_i 1)_{i=1:N}$$

- Other early references: [Morgan 92] (GMRES), [Chapman, Saad 92], [Kharchenko, Yeremin 92], [Burrage, Ehrel, and Pohl, 93]

- Estimations of eigenvectors corresponding to smallest eigenvalues / knowledge from the physics

- Geneo [Nataf, Spillane et al]: through solving local Gen EVPs, bounds smallest eigenvalue for standard FE and bilinear forms, SPD input matrix

<table>
<thead>
<tr>
<th>subd</th>
<th>dofs</th>
<th>AS</th>
<th>AS-ZEM ($V_0$)</th>
<th>GenEO ($V_0$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1452</td>
<td>79</td>
<td>54 (24)</td>
<td>16 (46)</td>
</tr>
<tr>
<td>8</td>
<td>29040</td>
<td>177</td>
<td>87 (48)</td>
<td>16 (102)</td>
</tr>
<tr>
<td>16</td>
<td>58080</td>
<td>378</td>
<td>145 (96)</td>
<td>16 (214)</td>
</tr>
</tbody>
</table>

$V_0$: size of the coarse space
AS-ZEM Nicolaides with rigid body motions, 6 per subd
Results for 3D elasticity problem provided by F. Nataf
The need for two level preconditioners

- When solving complex linear systems arising, e.g. from large discretized systems of PDEs with strongly heterogeneous coefficients (high contrast, multiscale).

- Flow in porous media
- Elasticity problems
- CMB data analysis (no PDE)

- Most of the existing preconditioners lack robustness
  - wrt jumps in coefficients / partitioning into irregular subdomains, e.g. one level DDM methods (block Jacobi, RAS), incomplete LU
  - A few small eigenvalues hinder the convergence of iterative methods
Using deflation to deal with low frequency modes

In the unified framework of [Tang et al., 2009], let:

\[ P := I - AZE^{-1}Z^T, \quad E := Z^T AZ \]

where
- \( Z \) is the deflation subspace matrix of full rank
- \( E \) is the coarse grid correction, a small dense invertible matrix
- \( P \) is the deflation matrix, \( PAZ = 0 \)

Usage in different classes of preconditioners
- DDM - \( Z \) and \( Z^T \) are the restriction and prolongation operators based on subdomains, \( E \) is a coarse grid, \( P \) is a subspace correction
- Deflation - \( Z \) contains the vectors to be deflated
- Multigrid - interpretation possible
Using deflation to deal with low frequency modes

In the unified framework of [Tang et al., 2009], let:

\[ P := I - AZE^{-1}Z^T, \quad E := Z^TAZ \]

where
- \( Z \) is the deflation subspace matrix of full rank
- \( E \) is the coarse grid correction, a small dense invertible matrix
- \( P \) is the deflation matrix, \( PAZ = 0 \)

Usage in different classes of preconditioners
- DDM - \( Z \) and \( Z^T \) are the restriction and prolongation operators based on subdomains, \( E \) is a coarse grid, \( P \) is a subspace correction
- Deflation - \( Z \) contains the vectors to be deflated
- Multigrid - interpretation possible
Using deflation to deal with low frequency modes

In the unified framework of [Tang et al., 2009], let:

\[ P := I - AZE^{-1}Z^T, \quad E := Z^T AZ \]

where

- \( Z \) is the deflation subspace matrix of full rank
- \( E \) is the coarse grid correction, a small dense invertible matrix
- \( P \) is the deflation matrix, \( PAZ = 0 \)

Example of preconditioner

\[ P_{2/lvl}^{-1} = M^{-1}P + ZE^{-1}Z^T, \]

where \( M \) is the first level preconditioner (eg based on block Jacobi).

- \( P_{2/lvl}^{-1}AZ = Z \)
- The small eigenvalues are shifted to 1.
- \( P_{2/lvl} \) is not SPD, even when \( A \) is, better choices available, but more expensive.
Using deflation to deal with low frequency modes

In the unified framework of [Tang et al., 2009], let:

\[ P := I - AZE^{-1}Z^T, \quad E := Z^TAZ \]

where

- \( Z \) is the deflation subspace matrix of full rank
- \( E \) is the coarse grid correction, a small dense invertible matrix
- \( P \) is the deflation matrix, \( PAZ = 0 \)

**Example of preconditioner**

\[ P_{2lvl}^{-1} = M^{-1}P + ZE^{-1}Z^T, \]

where \( M \) is the first level preconditioner (eg based on block Jacobi).

- \( P_{2lvl}^{-1}AZ = Z \)
- The small eigenvalues are shifted to 1.
- \( P_{2lvl} \) is not SPD, even when \( A \) is, better choices available, but more expensive.
Two level preconditioners (contd)

Computing the preconditioner requires

- Deflation subspace $Z$, which can be formed by
  - Eigenvectors corresponding to smallest eigenvalues - from previous linear systems solved with different right hand sides, etc.
  - Using knowledge from the physics, partition of the unity, etc.

- Computing $AZ$ and $E = Z^T AZ$.

Applying the preconditioner at each iteration requires

- Computing $y = ZE^{-1}Z^T (Ax_i) = ZE^{-1}Z^Tv$
  \[ \Rightarrow \text{involves collective communication when computing } Z^Tv, \]
  \[ \text{and solving a linear system with } E. \]
Two level preconditioners (contd)

Computing the preconditioner requires

- Deflation subspace $Z$, which can be formed by
  - Eigenvectors corresponding to smallest eigenvalues - from previous linear systems solved with different right hand sides, etc.
  - Using knowledge from the physics, partition of the unity, etc.

- Computing $AZ$ and $E = Z^T AZ$.

Applying the preconditioner at each iteration requires

- Computing $y = ZE^{-1}Z^T(Ax_i) = ZE^{-1}Z^Tv$
  $\Rightarrow$ involves collective communication when computing $Z^Tv$, and solving a linear system with $E$. 

\[
\begin{align*}
\text{Z} & \quad \text{E}^{-1} & \quad \text{Z}^T & \quad (Ax_i) & \quad \text{Z} & \quad \text{E}^{-1}(Z^T Ax_i)
\end{align*}
\]
A taxonomy for conjugate gradient methods.

Domain decomposition algorithms.

*An introduction to domain decomposition methods.*
Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.
Algorithms, theory, and parallel implementation.

Enlarged Krylov Subspace Conjugate Gradient Methods for Reducing Communication.
Also as INRIA TR 8266.

Methods of conjugate gradients for solving linear systems.

The block conjugate gradient algorithm and related methods.
*Linear Algebra and Its Applications*, 29:293–322.

Exact and inexact breakdowns in the block gmres method.