Non-Blocking Collective Operations

- Use to hide communication!
- Allows overlap of numerical computations and communcations.
- In MPI-1/MPI-2 only possible for point-to-point communication: `MPI_Isend` and `MPI_Irecv`.

  Additional libraries necessary for collective operations!
  Example: LibNBC (non-blocking collectives)
- Are included in new MPI-3 standard.

Example: Pseudocode for Nonblock. Reduct.

```
MPI_Request req;
int sbuf1[SIZE], rbuf1[SIZE], buf2[SIZE];
// compute sbuf1
compute(sbuf1, SIZE);

// start non-blocking allreduce of subf1
// computation and communication overlap
MPI_Iallreduce(sbuf1,rbuf1,SIZE,MPI_INT,MPI_SUM,MPI_COMM, &req);

// compute buf2 (independent of buf1)
compute(buf2, SIZE);

// synchronization
MPI_WAIT(&req, &stat);

// use data in rbuf1; final computation
evaluate(rbuf1, buf2, SIZE);
```

Iter. Methods for general (nonsymmetric) A: BiCGSTAB

- Applicable if little memory at hand and A^T not available.
- Computational costs per iteration similar to BiCG and CGS.
- Alternative for CGS that avoids irregular convergence patterns of CGS maintaining similar convergence speed.
- Less loss of accuracy in the updated residual.

Iter. Methods for general (nonsymmetric) A: GMRES

- Leads to smallest residual for fixed number of iteration steps, but these steps become increasingly expensive.
- To limit increasing storage requirments and work per iteration step, restarting is necessary. When to do so depends on A and b; it requires skill and experience.
- Requires only matrix-vector products with the coeff. matrix.
- Number of inner products grows linearly with iteration number (up to restart point).
- Implementation based on Gram-Schmidt $\rightarrow$ inner products independent $\rightarrow$ only one synchronization point.
  Using modified Gram-Schmidt $\rightarrow$ one synchronization point per inner product.

We consider GMRES in the following.

5.2.3. GMRES

- Iterative solution method for general A
- Consider small subspace U_m and determine optimal approximate solution for $Ax = b$ in U_m. Hence x is of the form $x := U_m y$

$$\min_{x \in U_m} \|Ax - b\|_2 = \min_{y} \|A(U_m y) - b\|_2$$

- Can be solved by normal equations: $(U_m^T A^T A U_m) y = U_m^T A^T b$
- Try to find sequence of ”good“ subspaces $U_1 \to U_2 \to U_3 \to \ldots$ such that iteratively we can update the optimal solutions

$$x_1 \to x_2 \to x_3 \to \ldots \to A^{-1} b$$

using mainly matrix-vector products.

GMRES: Subspace

What subspace for fast convergence and easy computations?

$$U_m := K_m(A, b) = \text{span}(b, Ab, \ldots, A^{m-1}b) \quad \text{(Krylov space)}$$

Problem: b, Ab, A^2b, ... is bad basis for this subspace!

So we need a first step to compute a good basis for U_m:

Start with $u_1 := \frac{b}{\|b\|_2}$ and do for $j = 2 : m$:

$$\widetilde{u}_j := Au_{j-1} - \sum_{k=1}^{j-1}(u_k^T Au_{j-1})u_k = Au_{j-1} - \sum_{k=1}^{j-1} h_{k,j-1}u_k$$

$$u_j := \frac{\widetilde{u}_j}{\|\widetilde{u}_j\|_2} = \frac{\widetilde{u}_j}{h_{j,j-1}}$$

which is the standard orthogonalization method (Arnoldi method)

Matrix Form of Arnoldi ONB

$$U_m = \text{span}(b, Ab, \ldots, A^{m-1}b) = \text{span}(u_1, u_2, \ldots, u_m) \quad \text{(ONB)}$$

Write this orthogonalization method in matrix form

$$Au_{j-1} = \sum_{k=1}^{j-1} h_{k,j-1} u_k + \widetilde{u}_j = \sum_{k=1}^{j} h_{k,j-1} u_k$$

$$AU_m = A(u_1 \quad \ldots \quad u_m) = (u_1 \quad \ldots \quad u_{m+1})\widetilde{H}_{m+1,m} = U_{m+1}\widetilde{H}_{m+1,m}$$

$$\widetilde{H}_{m+1,m} = \begin{pmatrix} h_{11} & \cdots & \cdots & h_{1m} \\ h_{21} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ & & \ddots & h_{mm} \\ 0 & & 0 & h_{m+1,m} \end{pmatrix}$$

GMRES: Minimization

This leads to minimization problem

$$\begin{aligned}
\min_{x \in U_m} \|Ax - b\| &= \min_{y} \|AU_m y - b\| \\
&= \min_{y} \left\| U_{m+1} \widetilde{H}_{m+1,m} y - \|b\| u_1 \right\| \\
&= \min_{y} \left\| U_{m+1} (\widetilde{H}_{m+1,m} y - \|b\| e_1) \right\| \\
&= \min_{y} \left\| \widetilde{H}_{m+1,m} y - \|b\| e_1 \right\|
\end{aligned}$$

Because U_{m+1} is part of an orthogonal matrix.

GMRES: QR

Use Givens matrices to compute a QR-factorization of the upper Hessenberg matrix $\widetilde{H}_{m+1,m}$.

$$G_1 \cdot \begin{pmatrix} * & \cdots & * \\ * & \ddots & \vdots \\ & \ddots & * \\ & & * \end{pmatrix} = \begin{pmatrix} * & \cdots & * \\ 0 & \ddots & \vdots \\ & \ddots & * \\ & & * \end{pmatrix} =$$

$$G_m \cdots G_2 G_1 \cdot \widetilde{H}_{m+1,m} = Q \cdot \widetilde{H}_{m+1,m} = \begin{pmatrix} R_m \\ 0 \end{pmatrix}$$

QR via Givens matrices is a simplified version of QR via Householder matrices.

GMRES

$$
\begin{aligned}
\min_{x \in K_m} \|Ax - b\| &= \min_y \left\| \widetilde{H}_{m+1,m} y - \|b\| e_1 \right\| \\
&= \min_y \left\| Q^T R y - \|b\| e_1 \right\| = \min_y \|Ry - \|b\| Q e_1\| \\
&= \min_y \left\| \begin{pmatrix} R_m \\ 0 \end{pmatrix} y - \widetilde{b} \right\| = \min_y \left\| \begin{pmatrix} R_m y - \widetilde{b}_m \\ -\widetilde{\beta}_m \end{pmatrix} \right\|
\end{aligned}
$$

Solution:

$$R_m y = \widetilde{b}_m, \quad x_m = U_m y$$

GMRES Algorithm

- GMRES is a clever implementation of this sequence of least squares problems.
- Computing step by step for increasing m
  - next Au_m
  - enlarged $H_{m+1,m}$ by Arnoldi orthogonalization (gives new u_{m+1})
  - next Givens matrix G_m
  - update triangular solves to get next y_m and x_m.
- Disadvantage: large Hessenberg matrices!
- Therefore, use restarted GMRES, e.g. GMRES(20).

GMRES Algorithm: First Step

Start: $b,\ u_1 := \frac{b}{\|b\|}$

$$h_{2,1}u_2 = Au_1 - h_{1,1}u_1, \quad H_{2,1} = \begin{pmatrix} h_{1,1} \\ h_{2,1} \end{pmatrix}, \quad G_1 H_{2,1} = \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix},$$

$$\begin{aligned} \|Ax - b\| &= \|AU_1 y_1 - b\| = \|H_{2,1} y_1 - \|b\| e_1\| \\ &= \left\| G_1^T \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix} y_1 - \|b\| e_1 \right\| = \left\| \begin{pmatrix} \nu_{1,1} \\ 0 \end{pmatrix} y_1 - \|b\| G_1 e_1 \right\| \end{aligned}$$

$$y_1 = \frac{(G_1 e_1 \|b\|)_1}{\nu_{1,1}}, \qquad x_1 = u_1 y_1 = U_1 y_1$$

GMRES Algorithm: Second Step

$$h_{3,2}u_3 = Au_2 - h_{2,2}u_2 - h_{1,2}u_1, \qquad H_{3,2} = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \\ 0 & h_{32} \end{pmatrix},$$

$$G_1 H_{3,2} = \begin{pmatrix} \nu_1 & * \\ 0 & * \\ 0 & * \end{pmatrix}, \qquad G_2 G_1 H_{3,2} = \begin{pmatrix} \nu_{1,1} & \nu_{1,2} \\ 0 & \nu_{2,2} \\ 0 & 0 \end{pmatrix}$$

$$\|Ax - b\| = \|AU_2 y_2 - b\| = \|H_{3,2} y_2 - \|b\| e_1\|$$

$$= \left\| G_1^T G_2^T \begin{pmatrix} \nu_{1,1} & \nu_{1,2} \\ 0 & \nu_{2,2} \\ 0 & 0 \end{pmatrix} y_2 - \|b\| e_1 \right\| = \left\| \begin{pmatrix} \nu_{1,1} & \nu_{1,2} \\ 0 & \nu_{2,2} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_{2,1} \\ y_{2,2} \end{pmatrix} - \|b\| G_2 G_1 e_1 \right\|$$

$$y_{2,2} = \frac{(G_2 G_1 e_1 \|b\|)_2}{\nu_{2,2}}, \qquad y_{2,1} = \frac{(G_2 G_1 e_1 \|b\|)_1 - \nu_{1,2} y_{2,2}}{\nu_{1,1}}$$

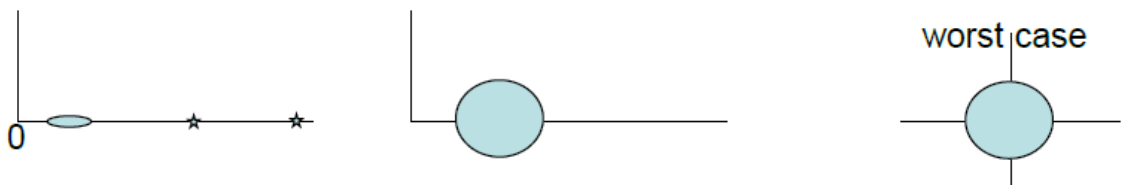
$x_2 = U_2 y_2 = u_1 y_{2,1} + u_2 y_{2,2}$ and so forth...

GMRES Error (A diagonalisable)

$$
\begin{aligned}
\|r_m\| &= \|Ax_m - b\| = \min_{x \in U_m} \|Ax - b\| \\
&= \min_{\alpha_1,\ldots,\alpha_m} \left\| A\left(\sum_{j=0}^{m-1} \alpha_j A^j b\right) - b \right\| = \min_{\mathcal{P}^{(m-1)}} \left\| A\mathcal{P}^{(m-1)}(A)b - b \right\| \\
&= \min_{\mathcal{Q}^{(m)}(0)=1} \left\| \mathcal{Q}^{(m)}(A)b \right\| = \min_{\mathcal{Q}^{(m)}(0)=1} \left\| V\mathcal{Q}^{(m)}(\Lambda)V^{-1}b \right\| \\
&= \min_{\mathcal{Q}^{(m)}(0)=1} \left\| V \operatorname{diag}(\mathcal{Q}^{(m)}(\lambda_j))V^{-1}b \right\| \leq \\
&\leq \kappa_2(V) \cdot \min_{\mathcal{Q}^{(m)}(0)=1} \left[\max_{j=1,\ldots,n} \left| \mathcal{Q}^{(m)}(\lambda_j) \right| \right] \|r_0\|
\end{aligned}
$$

Iterative Methods: Survey

- Basic iterative methods (not considering multigrid!)
- SPD A: PCG (preconditioned conjugate gradient method)
- symmetric indefinite A: MINRES, SYMMLQ, (PCG on normal equations)
- general A:
  - GMRES (optimal like CG, but expensive)
  - BiCG, BiCGSTAB, CGS (not optimal, but cheap, instable?)
  - QMR (quasi optimal, cheap)
- Essential for convergence is position of eigenvalues (singular values)
- Fast convergence for well conditioned problems or matrices with clustered spectrum



5.3. Preconditioning

- Direct solvers: sequential, loosing sparsity
- Iterative solvers: easy parallel and sparse, but possibly slowly convergent
- Combination of both methods (2 variants):
  1. Include preconditioner $M \approx A$ in the form $M^{-1}Ax = M^{-1}b$, such that
     - M is easy to deal with in parallel (reduced approximate direct solver)
     - spectrum of $M^{-1}A$ is much better clustered
  2. Or include preconditioner $M \approx A^{-1}$ in the form $MAx = Mb$, such that
     - M is easy to deal with in parallel (reduced approximate inverse)
     - Spectrum of MA is much better clustered
- General both-sided preconditioning:

$$Ax = b \Leftrightarrow MA(Ky) = Mb \text{ and } Ky = x$$

Overview of Following Preconditioners

- Stationary preconditioners
- ILU preconditioner
- Polynomial preconditioners
- Sparse approximate inverse preconditioners:
  - SPAI for the general nonsymmetric case
  - FSPAI for the SPD case
  - MSPAI using SPAI with probing

5.3.1. Stationary Preconditioners

Consider preconditioner $M \approx A$ in the form $M^{-1}Ax = M^{-1}b$.

- Stationary iteration to splitting $A = M - N$. Convergence depends on
$$\|I - M^{-1}A\| < 1$$
- That is exactly a condition for a good preconditioner: spectrum clustered around 1, $M^{-1}A \approx I$
- If splitting leads to fast convergence, than M is also a good preconditioner.
- In this sense, PCG with stationary preconditioner M can be seen as an accelaration of the stationary method with splitting $M - N$.

Stationary Preconditioners (cont.)

- Jacobi splitting with $D = \mathrm{diag}(A)$ gives Jacobi preconditioner $M := D$.
- Gauss-Seidel splitting $M = D - L$ leads to GS preconditioner.
- Relaxation:

$$x^{(k,\mathrm{new})} := (1-\omega)x^{(k-1)} + \omega x^{(k)}$$

  As convex combination of old and new iterate (Jacobi or GS)
- Symmetrization: first iteration with preconditioner M, second iteration with M^T.

$$M_{\mathrm{new}} := M + M^T - M^T A M$$

- Special case: damped GS $\rightarrow$ SSOR:

$$M_{\mathrm{new}} = \frac{1}{2-\omega}\left(\frac{1}{\omega}D - L\right)\left(\frac{1}{\omega}D\right)^{-1}\left(\frac{1}{\omega}D - L\right)^T$$

5.3.2. ILU Preconditioner

- Apply Gaussian elimination algorithm, but only on allowed pattern $\Rightarrow$ incomplete LU factorization called ILU.
- Reduce in the `for`-loops the indices to the indices with
  - allowed pattern, e.g. ILU(0) for pattern of A
  - values that are not too small, ILUT for ILU with treshold
- Leads to approximate LU factorization

$$A = LU + R, \quad \text{preconditioner} \quad M = LU$$

  with all ignored fill-in entries collected in R.
- MILU (modified ILU): collect all ignored fill-in entries on the related main diagonal elements $\rightarrow$ maintains the row sum or the action on $(1, 1, \ldots, 1)^T$.

Overview Explicit Preconditioners

- ILU and stationary methods use approximations on A itself.
- The resulting preconditioners are given by triangular matrices L, that have to be solved in each iteration step: $L^{-1}x^{(k)}$! Strongly sequential!
- Jacobi easy to parallelize, but slow convergence.
- Question: How to derive preconditioners that lead to fast convergence and are easy to parallelize?
- Until now considered variant 1.; In the following variant 2.:

  Find approximations on $M \approx A^{-1}$. Then the solution of the linear system given by the preconditioner, is only $Mx^{(k)}$, a matrix vector product!

Parallel Preconditioning

- Find preconditioner M, that satisfies three conditions:
  - **(i)** The computation of M is fast in parallel
  - **(ii)** The application Mx in each iteration step is easy in parallel
  - **(iii)** The spectrum AM or MA is clustered $\rightarrow$ fast convergence
- Examples:

  For GS is (i) ok, but not (ii)

  For Jacobi (i) and (ii) ok, but not (iii)

  For ILU (iii) ok, but not (i) and (ii)
- Note that preconditioners also have to be well defined! Zero on diagonals?

5.3.3. Polynomial Preconditioners

- Characteristic polynomial for A:

$$0 = q(A) = \gamma_n A^n + \gamma_{n-1} A^{n-1} + \ldots + \gamma_1 A + \gamma_0 I$$

  Gives polynomial representation for A^{-1} ($\gamma_0 \neq 0$):

$$A^{-1} = \frac{1}{\gamma_0}\left(-\gamma_n A^{n-1} - \gamma_{n-1} A^{n-2} - \ldots - \gamma_1 I\right) = p(A)$$

- Therefore, it makes sense to approximate A^{-1} by a polynomial.
- Better approximation by finding region $S \in \mathbb{R}$ or $\mathbb{C}$ that contains nearly all eigenvalues, and then find polynomial p that is near the inverse in S:

$$\min_{p_n} \|P(A)A - I\|, \quad \min_{p_n(x)} \left(\max_{\lambda \text{ eigenvalue}} |p(\lambda)\lambda - 1| \right)$$

- Solution: Normalized Chebyshev polynomials.

\+ Advant. of pol. prec.: better in parallel (mtx-vec product, BLAS 2)

\- Disadvantage: non-optimal approximation in Krylov subspace

5.3.4. Sparse Approximate Inverses

- Other approach for approximating A^{-1} by norm minimization

$$\min_{M \in \mathcal{P}} \|AM - I\| \qquad \text{over some sparsity pattern } \mathcal{P}.$$

- Choice of the norm?
  - analytic (to allow the explicit solution of this problem)
  - easy to compute (in parallel)

- Optimal norm is Frobenius norm:

$$\|A\|_F^2 := \sum_{i,j=1}^{n} a_{i,j}^2 = \sum_{j=1}^{n} \|A_{\bullet,j}\|^2 = \text{trace}(A^T A)$$

SPAI in Parallel

- First, we choose pattern $\mathcal{P}$ in a static way a priori, e.g. as the pattern of A

$$\min_{M \in \mathcal{P}} \|AM - I\|_F^2 = \min_{M \in \mathcal{P}} \sum_{k=1}^{n} \|(AM - I)e_k\|_2^2 = \sum_{k=1}^{n} \min_{M_k \in \mathcal{P}_k} \|AM_k - e_k\|_2^2$$

- Hence, to minimize the Frobenius norm, we have to solve n least squares problems in the sparse columns of M!
- This can be done fully in parallel!
- But costs for least squares problems?

SPAI and Least Squares

$$\min_{M_k \in \mathcal{P}_k} \|AM_k - e_k\|_2^2 = \min_{M_k \in \mathcal{P}_k} \left\| A \begin{pmatrix} 0 \\ * \\ 0 \\ * \\ 0 \\ 0 \end{pmatrix} - e_k \right\|_2^2 = \min_{M_k \in \mathcal{P}_k} \|A(:, \mathcal{J}_k) M_k(\mathcal{J}_k) - e_k\|_2^2$$

Denote by $\mathcal{J}_k$ the set of allowed indices in the kth column of M.