Preconditioning

- Introduction to preconditioning
- Preconditioned iterations
- Preconditioned CG and GMRES.
- Basic preconditioners.
- $ILU(0)$, $ILU(p)$, $ILUT$ preconditioners
- See Chapters 9, 10 of text for details.
Preconditioning – Basic principles

**Basic idea:**
Use Krylov subspace method on a modified system such as:

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

- The matrix \( M^{-1}A \) need not be formed explicitly; only need to solve \( Mw = v \) whenever needed.

- Consequence: fundamental requirement is that it should be easy to compute \( M^{-1}v \) for an arbitrary vector \( v \).

- We want: \( M \) close to \( A \) (system easier to solve) but operation \( v \rightarrow M^{-1}v \) inexpensive (added cost not too high).
Left, Right, and Split preconditioning

Left preconditioning

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

Right preconditioning

\[ AM^{-1}u = b, \text{ with } x = M^{-1}u \]

Split preconditioning: \( M \) is factored as \( M = M_L M_R \).

\[ M_L^{-1} AM_R^{-1}u = M_L^{-1}b, \text{ with } x = M_R^{-1}u \]
Preconditioned CG (PCG)

- Assume: \( A \) and \( M \) are both SPD.
- Can apply CG directly to systems
  \( M^{-1}Ax = M^{-1}b \) or \( AM^{-1}u = b \)
- Problem: loss of symmetry
- Alternative: when \( M = LL^T \) use split preconditioner option
- Second alternative: Observe that \( M^{-1}A \) is self-adjoint with respect to \( M \) inner product:

\[
(M^{-1}Ax, y)_M = (Ax, y) = (x, Ay) = (x, M^{-1}Ay)_M
\]
**Preconditioned CG (PCG)**

**ALGORITHM 1: Preconditioned CG**

1. Compute $r_0 := b - Ax_0$, $z_0 = M^{-1}r_0$, and $p_0 := z_0$
2. For $j = 0, 1, \ldots$, until convergence Do:
   3. $\alpha_j := (r_j, z_j)/(Ap_j, p_j)$
   4. $x_{j+1} := x_j + \alpha_j p_j$
   5. $r_{j+1} := r_j - \alpha_j Ap_j$
   6. $z_{j+1} := M^{-1}r_{j+1}$
   7. $\beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
   8. $p_{j+1} := z_{j+1} + \beta_j p_j$
   9. EndDo
Note $M^{-1}A$ is also self-adjoint with respect to $(.,.)_A$:

$$(M^{-1}Ax, y)_A = (AM^{-1}Ax, y) = (x, AM^{-1}Ay) = (x, M^{-1}Ay)_A$$

- Can obtain a similar algorithm
- Assume that $M = \text{Cholesky product } M = LL^T$.

Then, another possibility: Split preconditioning option, which applies CG to the system

$$L^{-1}AL^{-T}u = L^{-1}b, \text{ with } x = L^Tu$$

- Notation: $\hat{A} = L^{-1}AL^{-T}$. All quantities related to the preconditioned system are indicated by $\hat{\cdot}$. 
ALGORITHM 2. *CG with Split Preconditioner*

1. **Compute** \( r_0 := b - Ax_0; \) \( \hat{r}_0 = L^{-1}r_0; \) \( p_0 := L^{-T}\hat{r}_0. \)
2. **For** \( j = 0, 1, \ldots, \) **until convergence** **Do:**
   3. \( \alpha_j := (\hat{r}_j, \hat{r}_j)/(Ap_j, p_j) \)
   4. \( x_{j+1} := x_j + \alpha_j p_j \)
   5. \( \hat{r}_{j+1} := \hat{r}_j - \alpha_j L^{-1}Ap_j \)
   6. \( \beta_j := (\hat{r}_{j+1}, \hat{r}_{j+1})/(\hat{r}_j, \hat{r}_j) \)
   7. \( p_{j+1} := L^{-T}\hat{r}_{j+1} + \beta_j p_j \)
   8. **EndDo**

▶ The \( x_j \)'s produced by the above algorithm and PCG are identical (if same initial guess is used).

Show this
1. **Start**: Choose $x_0$ and a dimension $m$

2. **Arnoldi process**:
   
   - Compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$.
   
   - For $j = 1, \ldots, m$ do
     - Compute $z_j := M^{-1}v_j$
     - Compute $w := Az_j$
     - for $i = 1, \ldots, j$, do: \( h_{i,j} := (w, v_i) \)
     - \( w := w - h_{i,j}v_i \)
     - \( h_{j+1,1} = \|w\|_2; v_{j+1} = w/h_{j+1,1} \)
   
   - Define $V_m := [v_1, \ldots, v_m]$ and $\bar{H}_m = \{h_{i,j}\}$. 

*ALGORITHM: 3. GMRES – (right) Preconditioned*
3. Form the approximate solution: \[ x_m = x_0 + M^{-1}V_m y_m \] where 
\[ y_m = \arg\min_y \| \beta e_1 - \bar{H}_m y \|_2 \] and \( e_1 = [1, 0, \ldots, 0]^T \).

4. Restart: If satisfied stop, else set \( x_0 \leftarrow x_m \) and goto 2.

Remark: \( M \) is assumed to be the same at each step \( j \). Situations may arise where \( M \) varies: \( M \rightarrow M_j \). We need a ‘Flexible’ accelerator that allows this. Changes needed:

1) Save each \( z_j \) into matrix \( Z_m = [z_1, \ldots, z_m] \).

2) Replace \( M^{-1}V_m \) by \( Z_m \) to form solution in step 3.

What optimality property is satisfied with (1) Left Preconditioned GMRES, (2) Right Preconditioned GMRES; (3) Flexible GMRES?
Standard preconditioners

• Simplest preconditioner: $M = \text{Diag}(A)$ ➤ poor convergence.
• Next to simplest: SSOR. $M = (D - \omega E)D^{-1}(D - \omega F')$
• Still simple but often more efficient: ILU(0).
• ILU(p) – ILU with level of fill p – more complex.
• Class of ILU preconditioners with threshold
• Class of approximate inverse preconditioners
• Class of Multilevel ILU preconditioners
• Algebraic Multigrid Preconditioners
The SOR/SSOR preconditioner

**SOR preconditining**

\[ M_{\text{SOR}} = (D - \omega E) \]

**SSOR preconditining**

\[ M_{\text{SSOR}} = (D - \omega E)D^{-1}(D - \omega F) \]

\[ M_{\text{SSOR}} = LU, \; L = \text{lower unit matrix}, \; U = \text{upper triangular}. \; \text{One solve with } M_{\text{SSOR}} \approx \text{same cost as a MAT-VEC.} \]
- $k$-step SOR (resp. SSOR) preconditioning:
  
  $k$ steps of SOR (resp. SSOR)

- Questions: Best $\omega$? For preconditioning can take $\omega = 1$

  $$M = (D - E)D^{-1}(D - F)$$

Observe: $M = LU + R$ with $R = ED^{-1}F$.

- Best $k$? $k = 1$ is rarely the best. Substantial difference in performance.
Iteration times versus $k$ for SOR($k$) preconditioned GMRES
Notation: $\text{NZ}(X) = \{(i,j) \mid X_{i,j} \neq 0\}$

Formal definition of ILU(0):

\[
A = LU + R
\]

$\text{NZ}(L) \cup \text{NZ}(U) = \text{NZ}(A)$

$r_{ij} = 0$ for $(i,j) \in \text{NZ}(A)$

Constructive definition: Compute the LU factorization of $A$ but drop any fill-in in $L$ and $U$ outside of $\text{Struct}(A)$.

ILU factorizations are often based on $i,k,j$ version of GE.
What is the IKJ version of GE?

**Algorithm : 4. Gaussian Elimination – IKJ Variant**

1. For $i = 2, \ldots, n$ Do:
2. For $k = 1, \ldots, i - 1$ Do:
3. $a_{ik} := a_{ik} / a_{kk}$
4. For $j = k + 1, \ldots, n$ Do:
5. $a_{ij} := a_{ij} - a_{ik} * a_{kj}$
6. EndDo
7. EndDo
8. EndDo
Accessed but not modified
Accessed and modified
Not accessed
**ALGORITHM : 5**  

**ILU(0)**

For $i = 1, \ldots, N$ Do:

For $k = 1, \ldots, i - 1$ and if $(i, k) \in NZ(A)$ Do:

Compute $a_{ik} := a_{ik}/a_{kj}$

For $j = k + 1, \ldots$ and if $(i, j) \in NZ(A)$, Do:

compute $a_{ij} := a_{ij} - a_{ik}a_{k,j}$.

EndFor

EndFor

➤ When $A$ is SPD then the ILU factorization = Incomplete Choleski factorization – IC(0). Meijerink and Van der Vorst [1977].
Pattern of ILU(0) for 5-point matrix. ’Stencil’ viewpoint
Stencil: local connectivity for a graph with a regular pattern.

Example: For 5-point matrix $A$ each node is coupled with its East, West, North, South neighbors (when they exist).

Interpret fill-ins in the ILU(0) and ILU(1) preconditioners using only stencils/
More than anything else, what determines the convergence of an iterative method is the distribution of the eigenvalues of the matrix.

Need to consider eigenvalues of preconditioned matrix $M^{-1}A$

Clustering around 1 results in fast convergence

If $A$ is SPD with only $k$ distinct eigenvalues, what is the minimal polynomial $p$ of $A$? Show that $p(0) \neq 0$. How many steps will it take CG to converge for any linear system $Ax = b$?
Higher order ILU factorization

- Higher accuracy incomplete Choleski: for regularly structured problems, IC(\(p\)) allows \(p\) additional diagonals in \(L\).
- Can be generalized to irregular sparse matrices using the notion of level of fill-in [Watts III, 1979]

Initially \(\text{Lev}_{ij} = \begin{cases} 0 & \text{for } a_{ij} \neq 0 \\ \infty & \text{for } a_{ij} = 0 \end{cases} \)

At a given step \(i\) of Gaussian elimination:

\[
\text{Lev}_{ij} = \min\{\text{Lev}_{ij}; \text{Lev}_{ik} + \text{Lev}_{kj} + 1\}
\]
Algorithm 6: ILU($p$)

For $i = 2, N$ Do
  For each $k = 1, \ldots, i - 1$ and if $a_{ij} \neq 0$ do
    Compute $a_{ik} := a_{ik} / a_{jj}$
    Compute $a_{i,*} := a_{i,*} - a_{ik} a_{k,*}$.
    Update the levels of $a_{i,*}$
    In row $i$: if $\text{lev}(a_{ij}) > p$ set $a_{ij} = 0$
  EndFor
EndFor

- Algorithm can be split into symbolic and a numerical phase.
- Higher level of fill-in $\rightarrow$ typically fewer iterations - but more expensive set-up cost
Augmented pattern used for $\text{ILU}(1) = \text{pattern of } L U \text{ from } \text{ILU}(0)$
**ILU with threshold: ILUT($k, \epsilon$)**

ILU(p) factorizations are based on structure only and not numerical values

- potential problems for non M-matrices.

Alternative: ILU with Threshold, ILUT

- During each i-th step in GE ($i, k, j$ version), discard pivots or fill-ins whose value is below $\epsilon \|row_i(A)\|$.

- Once the $i$-th row of $L + U$, (L-part + U-part) is computed retain only the $k$ largest elements in both parts.


- Easy to implement and can be made quite inexpensive.
Other preconditioners

Many other techniques have been developed:

- Approximate inverse methods
- Polynomial preconditioners
- Multigrid - type methods
- Incomplete LU based on Crout factorization
- Multi-elimination and multilevel ILU (ARMS)