ECS231 Handout

Subspace Projection Methods for Solving Large-Scale Linear System

Part I. Basics

1. The landscape of solvers for linear systems of equations

$$Ax = b,$$

where A is an $n \times n$ nonsingular matrix and b is an n-vector, $x \in \mathrm{R}^n$ is the unknown.

more robust $\leftarrow - - - - \rightarrow$ less storage

	Direct	Iterative ($u = Av$)
Nonsymmetric A	LU	GMRES
Symmetric positive definite A	Cholesky	CG

more general ↑ | | | ↓ more robust

2. A framework for subspace projection methods.

   The basic idea of subspace projection technique is to extract an approximate solution $\widetilde{x}$ from a subspace of R^n. It is a technique of ***dimension reduction***. Let $\mathcal{W}$ and $\mathcal{V}$ be two m-dimensional subspaces of R^n, and x_0 is an initial guess of the solution, then the *subspace projection technique* is to

$$\text{find } \widetilde{x} \in x_0 + z,\ z \in \mathcal{W} \text{ such that } b - A\widetilde{x} \perp \mathcal{V}. \tag{1}$$

   Define initial residual $r_0 = b - Ax_0$, we have $b - A\widetilde{x} = b - A(x_0 + z) = r_0 - Az$. Therefore, the formulation (1) is equivalent to

$$\text{find } z \in \mathcal{W} \text{ such that } r_0 - Az \perp \mathcal{V}. \tag{1a}$$

   If $\mathcal{W} = \mathcal{V}$, then it is called an **orthogonal projection** method and the corresponding orthogonality constraints in (1a) is known as the Galerkin condition. Otherwise, if $\mathcal{W} \neq \mathcal{V}$, it is called an **oblique projection** method and the corresponding orthogonality constraints in (1a) is known as the Petrov-Galerkin condition.

3. In matrix notation, let $V = [v_1, v_2, \ldots, v_m]$ be an $n \times m$ matrix whose columns form a basis of $\mathcal{V}$, and similarly $W = [w_1, w_2, \ldots, w_m]$ an $n \times m$ matrix whose columns form a basis of $\mathcal{W}$. Then any approximation solutions in $x_0 + \mathcal{W}$ can be written as

$$\widetilde{x} = x_0 + z = x_0 + Wy$$

   and the orthogonality condition (1a) implies

$$V^T(r_0 - Az) = 0.$$

   Thus we have

$$V^T AWy = V^T r_0.$$

Consequently, if V^TAW is invertible, then

$$y = (V^TAW)^{-1}V^Tr_0.$$

Putting all together, a new approximate solution $\widetilde{x}$ is then given by

$$\widetilde{x} = x_0 + W(V^TAW)^{-1}V^Tr_0.$$

4. What we described here is a basic one projection step. The practical implementations (and algorithms we will learn) use a succession of such projections. Typically, a new projection step uses a "new" pair of subspaces $\mathcal{W}$ and $\mathcal{V}$ (updated from the previous step) and an initial guess x_0 equal to the most recent approximation. This leads to an **iterative** (refinement) procedure, and is a common technique in scientific computing.

5. Now, we have a ***prototype iterative subspace projection technique***:

   PROTOTYPE PROJECTION METHOD:
   0. Let x_0 be an initial approximation
   1. Iterate until convergence:
   2. Select a pair of subspaces $\mathcal{V}$ and $\mathcal{W}$ of R^n
   3. Generate basis matrices V and W for $\mathcal{V}$ and $\mathcal{W}$
   4. $r_0 \leftarrow b - Ax_0$
   5. $y \leftarrow (V^TAW)^{-1}V^Tr_0$
   6. $x_0 \leftarrow x_0 + Wy$

   Two remarks are in order:

   1. In many practical algorithms, the matrix V^TAW does not have to be formed explicitly. It is available as a by-product of Steps 2 and 3.
   2. The method is defined only when V^TAW is nonsingular, which is not guaranteed to be true. There are two important special cases where the nonsingularity of V^TAW is guaranteed:
      (a) If A is symmetric positive definite (SPD) and $\mathcal{W} = \mathcal{V}$, then $V^TAW = W^TAW$ is also SPD (and nonsingular).
      (b) If A is nonsingular, and $\mathcal{V} = A\mathcal{W}$, then $V^TAW = W^TA^TAW$, which is SPD (and nonsingular).

6. A one-dimensional subspace projection process is defined when

$$\mathcal{W} = \mathrm{span}\{w\} \quad \text{and} \quad \mathcal{V} = \mathrm{span}\{v\},$$

where w and v are two vectors. In this case, the new approximation takes form

$$x_0 \leftarrow x_0 + z = x_0 + \alpha w$$

and the orthogonality condition (1a) implies $v^T(r_0 - Az) = v^T(r_0 - \alpha Aw) = 0$, and thus

$$\alpha = \frac{v^Tr_0}{v^TAw}.$$

7. Steepest Descent (SD) method

   When A is SPD, at each step, take

$$v = w = r_0 = b - Ax_0$$

   This yields

Steepest Descent (SD) Algorithm:

1. Pick an initial guess x_0
2. For $k = 0, 1, 2, \ldots$ until convergence do
3. $\quad r_k = b - Ax_k$
4. $\quad \alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}$
5. $\quad x_{k+1} = x_k + \alpha_k r_k$

Remarks:

(1) Since A is SPD, $r_k^T A r_k > 0$ except $r_k = 0$. Therefore, SD does not breakdown.

(2) Let $x_* = A^{-1}b$, then k-th step of the SD iteration minimizes

$$f(x) \stackrel{\text{def}}{=} \frac{1}{2}\|x_* - x\|_A^2 = \frac{1}{2}(x_* - x)^T A(x_* - x), \quad x_* = A^{-1}b$$

over all vectors of the form $x_k - \alpha(\nabla f(x_k))$, known as *line search*. This is equivalent to the optimization problem

$$\alpha_k = \text{argmin}_\alpha f\left(x_{k-1} - \alpha \cdot \nabla f(x_k)\right),$$

where $\nabla f(x_k) = b - Ax_k$ is the gradient of f at x_k. Recall that from Calculus, we learned that the negative of the gradient direction is locally the direction that yields the fastest rate of decrease for f.

8. Minimal Residual (MR) Iteration.

For a general nonsingular matrix A, at each step, let

$$w = r_0 \quad \text{and} \quad v = Ar_0,$$

It yields

Minimal Residual (MR) Algorithm:

1. Pick an initial guess x_0
2. For $k = 0, 1, 2, \ldots$ until convergence do
3. $\quad r_k = b - Ax_k$
4. $\quad \alpha_k = \frac{r_k^T A^T r_k}{r_k^T A^T A r_k}$
5. $\quad x_{k+1} = x_k + \alpha_k r_k$

Remark: each step of the MR iteration minimizes

$$f(x) \stackrel{\text{def}}{=} \|r\|_2^2 = \|b - Ax\|_2^2$$

over all vectors of the form $x_k - \alpha r_k$, namely *line search*, which is equivalent to solve the least squares problem

$$\min_\alpha \|b - A(x_k - \alpha r_k)\|_2.$$

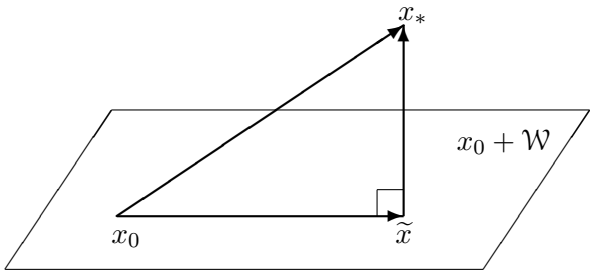
9. Further reading: theoretical results

(a) Optimality for orthogonal projection.

Theorem 1. *Assume that A is SPD and that $\mathcal{V} = \mathcal{W}$. Then a vector $\widetilde{x}$ is the result of (1) if and only if*

$$\|x_* - \widetilde{x}\|_A = \min_{x \in x_0 + \mathcal{W}} \|x_* - x\|_A,$$

where $\|x_* - x\|_A = \sqrt{(x_* - x)^T A(x_* - x)}$, and x_* is the exact solution to $Ax = b$.



PROOF: Notice that $(A(\cdot), \cdot)$ is an inner product on R^n. Thus $\|x_* - x\|_A$ over all possible $x \in x_0 + \mathcal{W}$ is minimized at $\widetilde{x}$ if and only if $x_* - \widetilde{x} \perp_A \mathcal{W}$, i.e.,

$$(A(x_* - \widetilde{x}), w) = (b - A\widetilde{x}, w) = 0 \quad \text{for any } w \in \mathcal{W} = \mathcal{V}.$$

This is (1). ∎

(b) The *steepest descent (SD) method* and *conjugate gradient (CG) method* (to be discussed later) are the corresponding implementations for solving large scale symmetric definite linear system of equations.

(c) Convergence of the SD algorithm: Let A be SPD, and let $\lambda_{\min}$ and $\lambda_{\max}$ be its smallest and largest eigenvalues respectively. Then for the SD ALGORITHM

$$\|x_* - x_{k+1}\|_A \leq \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right) \|x_* - x_k\|_A = \left(\frac{\kappa(A) - 1}{\kappa(A) + 1}\right) \|x_* - x_k\|_A,$$

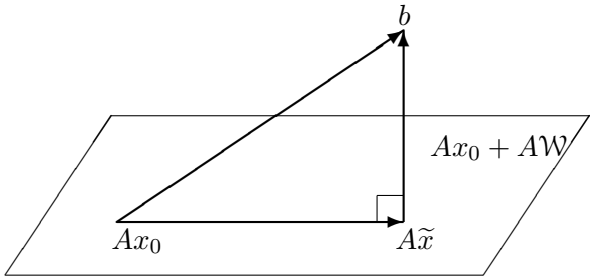
where x_* is the exact solution to $Ax = b$. $\kappa(A) = \lambda_{\max}/\lambda_{\min}$ is the condition number of A.[1]

(d) The SD converges for any initial guess. However, if $\kappa(A)$ is large and $\frac{\kappa(A)-1}{\kappa(A)+1} \approx 1$, the convergence could be extremely slow. The simple SD becomes impractical.

(e) **Optimality for oblique projection.**

Theorem 2. *Let A be an arbitrary square matrix and assume $\mathcal{V} = A\mathcal{W}$. Then a vector $\widetilde{x}$ is the result of (1) if and only if*

$$\|b - A\widetilde{x}\|_2 = \min_{x \in x_0 + \mathcal{W}} \|b - Ax\|_2.$$



PROOF: $\|b - Ax\|_2$ over all possible $x \in x_0 + \mathcal{W}$ is minimized at $\widetilde{x}$ if and only if $b - A\widetilde{x} \perp A\mathcal{W}$, i.e.,

$$(b - A\widetilde{x}, v) = 0 \quad \text{for any } v \in A\mathcal{W} = \mathcal{V}.$$

This is (1). ∎

(f) The *minimal residual residual (MR) method* and *generalized minimal residual (GMRES) method* (to be studied next) are the corresponding implementations for solving large scale nonsymmetric linear systems of equations.

[1] For a proof, see [Y. Saad, Iterative methods for sparse linear systems, Second Edition, SIAM, 2003]

(g) Convergence theorem of MR algorithm: Assume that $A + A^T$ is *SPD*,[2], and let $\mu = \lambda_{\min}\left(\frac{A+A^T}{2}\right)$, and $\sigma = \|A\|_2$. Then for the MR iteration[3]

$$\|r_{k+1}\|_2 \leq \left(1 - \frac{\mu^2}{\sigma^2}\right)^{1/2} \|r_k\|_2.$$

(h) For any positive definite (not necessarily symmetric) linear systems, the MR iteration converges for any initial guess. However, if $\frac{\mu}{\sigma} \approx 0$, the convergence becomes extremely slow and the MR is not a practical method.

Part II: Krylov subspace and GMRES

1. *Krylov subspace* is defined as

$$\mathcal{K}_m(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{m-1}v\},$$

where A is an $n \times n$ matrix, and v is a column vector of length n.

Note that if $x \in \mathcal{K}_m(A, v)$, then $x = p(A)v$, where $p(A)$ is a polynomial of degree not exceeding $m - 1$.

2. *Arnoldi procedure* is an algorithm for building an orthogonal basis $\{v_1, v_2, \ldots, v_m\}$ of the Krylov subspace $\mathcal{K}_m(A, v)$ using a modified Gram-Schmidt orthogonalization process.

```
1.  v_1 = v/||v||_2
2.  for j = 1, 2, ..., m
3.      compute w = A v_j
4.      for i = 1, 2, ..., j
5.          h_ij = v_i^T w
6.          w := w - h_ij v_i
7.      end for
8.      h_{j+1,j} = ||w||_2
9.      If h_{j+1,j} = 0, stop
10.     v_{j+1} = w/h_{j+1,j}
11. endfor
```

Proposition 1. *Assume that $h_{j+1,j} \neq 0$ for $j = 1, 2, \ldots, m$, then the vectors $\{v_1, v_2, \ldots, v_m\}$ form an orthonormal basis of the Krylov subspace $\mathcal{K}_m(A, v)$:*

$$\textit{span}\{v_1, v_2, \ldots, v_m\} = \mathcal{K}_m(A, v).$$

Proof. By induction.

3. Let

$$V_m = [v_1, v_2, \ldots, v_m] \quad \text{and} \quad H_m = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1,m-1} & h_{1m} \\ h_{21} & h_{22} & \cdots & h_{2,m-1} & h_{2m} \\ & h_{32} & \ddots & h_{3,m-1} & h_{3m} \\ & & \ddots & \vdots & \vdots \\ & & & h_{m,m-1} & h_{m,m} \end{bmatrix},$$

[2]This is equivalent to say that A is positive definite. A real matrix A said to be positive definite if $u^T Au > 0$ for any $0 \neq u \in \mathcal{R}$. It can be shown that if A is real positive definite, then A is nonsingular, in addition, $u^T Au \geq \lambda_{\min}\left(\frac{1}{2}(A + A^T)\right) u^T u$.

[3]For a proof, see [Y. Saad, Iterative methods for sparse linear systems, Second Edition, SIAM, 2003]

where H_m is called an upper Hessenberg matrix. Then in the matrix form, the Arnoldi procedure can be expressed by the following relations:

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T \tag{2}$$

and $V_m^T V_m = I_m$, $V_m^T v_{m+1} = 0$ and $\|v_{m+1}\|_2 = 1$.

If we denote

$$V_{m+1} = [V_m \ v_{m+1}] \quad \textit{and} \quad \widehat{H}_m = \begin{bmatrix} H_m \\ h_{m+1,m} e_m^T \end{bmatrix},$$

where $\widehat{H}_m$ is a $m+1$ by m upper triangular matrix, then the order-m Arnoldi decomposition (2) can also be written in the following compact form

$$AV_m = V_{m+1} \widehat{H}_m. \tag{3}$$

The expressions (2) (and (3)) is referred to as an *order-m Arnoldi decomposition.*

4. Remarks:

   - Note that the matrix A is only referenced via the matrix-vector multiplication Av_j. Therefore, it is ideal for large sparse or dense structure matrices. Any sparsity or structure of a matrix can be exploited in the matrix-vector multiplication.
   - The main storage requirement is $n(m+1)$ for storing Arnoldi vectors $\{v_i\}$ plus the storage requirement for the matrix A in question or the required matrix-vector multiplication.
   - The primary arithmetic cost of the procedure is the cost of m matrix-vector products plus $2m^2 n$ for the rest. It is common that the matrix-vector multiplication is the dominant cost.
   - The Arnoldi procedure breaks down when $h_{j+1,j} = 0$ for some j. It is easy to see that if the Arnoldi procedure breaks down at step j (i.e. $h_{j+1,j} = 0$), we have

     $$AV_j = V_j H_j.$$

     This indicates that $\mathcal{K}_j$ is an invariant subspace of A.
   - Some care must be taken to insure that the vectors v_j remain orthogonal to working accuracy in the presence of rounding error. The usual technique is called *reorthogonalization.*

5. The Generalized Minimum Residual (GMRES) method[4] is a generalization of the one-dimensional MR iteration. GMRES uses the following pair of Krylov subspaces as pair of projection subspaces:

   $$\mathcal{W} = \mathcal{K}_m(A, r_0) \quad \text{and} \quad \mathcal{V} = A\mathcal{W} = A\mathcal{K}_m(A, r_0).$$

   and can be derived under the framework of the subspace projection technique.

   Specifically, let

   $$x \in x_0 + \mathcal{W} = x_0 + V_m y.$$

   Then by the orthogonality condition (1), we have

   $$V_m^T A^T (b - Ax) = 0.$$

   i.e.,

   $$V_m^T A^T (r_0 - AV_m y) = 0.$$

[4] Y. Saad and M. H. Schultz. GMRES: a Generalized Minimal RESidual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, Vol.7, pp.856–869, 1986.

which is equivalent to

$$V_m^T A^T A V_m y = V_m^T A^T r_0$$

Then by *order-m Arnoldi decomposition*, we have

$$\widehat{H}_m^T \widehat{H}_m y = \widehat{H}_m^T V_{m+1}^T r_0 = \widehat{H}_m^T (\beta e_1)$$

This is equivalent to solve the least squares problem

$$\min_y \|\beta e_1 - \widehat{H}_m y\|_2.$$

for y.

6. Alternatively, we can also derive the GMRES method by exploitng the optimality property. Note that any vector x in $x_0 + \mathcal{K}_m$ can be written as $x = x_0 + V_m y$, where y is an m-vector. Define

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2 \tag{4}$$

Then using the Arnoldi decomposition (3), we have

$$\begin{aligned} b - Ax &= b - A(x_0 + V_m y) = r_0 - AV_m y \\ &= \beta v_1 - V_{m+1}\widehat{H}_m y = V_{m+1}(\beta e_1 - \widehat{H}_m y). \end{aligned}$$

Since the column vectors of V_{m+1} are orthonormal, then

$$J(y) = \|b - A(x_0 + V_m y)\|_2 = \|\beta e_1 - \widehat{H}_m y\|_2.$$

Therefore, the GMRES approximation x_m is the unique vector

$$x_m = x_0 + V_m y,$$

where y the solution of the least squares problem

$$\min_y \|\beta e_1 - \widehat{H}_m y\|_2.$$

This least squares problem is inexpensive to compute since m is typically small.

7. Restarting GMRES method. As m increases, the computational cost increases at least as $O(m^2 n)$. The memory cost increases as $O(mn)$. For large n this limits the largest value of m that can be used. The popular remedy is to restart the algorithm periodically for a fixed m.

   Restarted GMRES:
   1. compute $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$
   2. call Arnoldi procedure with A, v_1 and m
   3. solve $\min_y \|\beta e_1 - \widehat{H}_m y\|_2$
   4. $x_m = x_0 + V_m y_m$
   5. test for convergence, if satisfied, then *stop*
   6. set $x_0 := x_m$ and go to 1.

8. Breakdown of GMRES: Since the least squares problem always has solution, the only possibility of the breakdown of the GMRES is in the Arnoldi procedure when $h_{j+1,j}$ at some step j. However, in this case, the residual norm of x_j is zero, $b - Ax_j = 0$. x_j is the exact solution of the linear system $Ax = b$. This is called *lucky breakdown*. In fact, we have

Proposition 2. *Let A be a nonsingular matrix. Then the GMRES algorithm breaks down at step j, i.e., $h_{j+1,j} = 0$, if and only if x_j is an exact solution of $Ax = b$.*

9. Further reading: Convergence of GMRES.

We wish to establish a result to provide an upper bound on the convergence rate of the GMRES iterates. Unfortunately, because of the complication of non-Hermitian matrices and their spectral distribution, it is not possible to prove a simple result, but can get pretty close for practical use. First, we have the following lemma to characterize the approximate solution by the GMRES method:

Proposition 3. *Let x_m be the approximate solution obtained from the m-th step of the GMRES algorithm, and let $r_m = b - Ax_m$. Then x_m is of the form*

$$x_m = x_0 + q_m(A)r_0$$

and

$$\|r_m\|_2 = \|(I - Aq_m(A))r_0\|_2 = \min_{q \in \mathcal{P}_{m-1}} \|(I - Aq(A))r_0\|_2.$$

PROOF: This is true because x_m minimizes the 2-norm of the residual in the affine subspace $x_0 + \mathcal{K}_m$, the optimality property of the projection technique. Recall that $\mathcal{K}_m$ is the set of all vectors of the form $x_0 + q(A)r_0$, where q is a polynomial of degree $\leq m - 1$. ∎

Proposition 4. *Assume that A is diagonalizable matrix and let $A = V\Lambda V^{-1}$ where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the diagonal matrix of eigenvalues. Define*

$$\epsilon^{(m)} = \min_{p \in \mathcal{P}_m, p(0)=1} \max_{1 \leq i \leq n} |p(\lambda_i)|.$$

Then the residual norm satisfies the inequality

$$\|r_m\|_2 \leq \kappa_2(V)\epsilon^{(m)}\|r_0\|_2.$$

where $\kappa_2(V) = \|V\|_2\|V^{-1}\|_2$.

PROOF: see [Y. Saad, Iterative methods for sparse linear systems, Second Edition, SIAM, 2003]

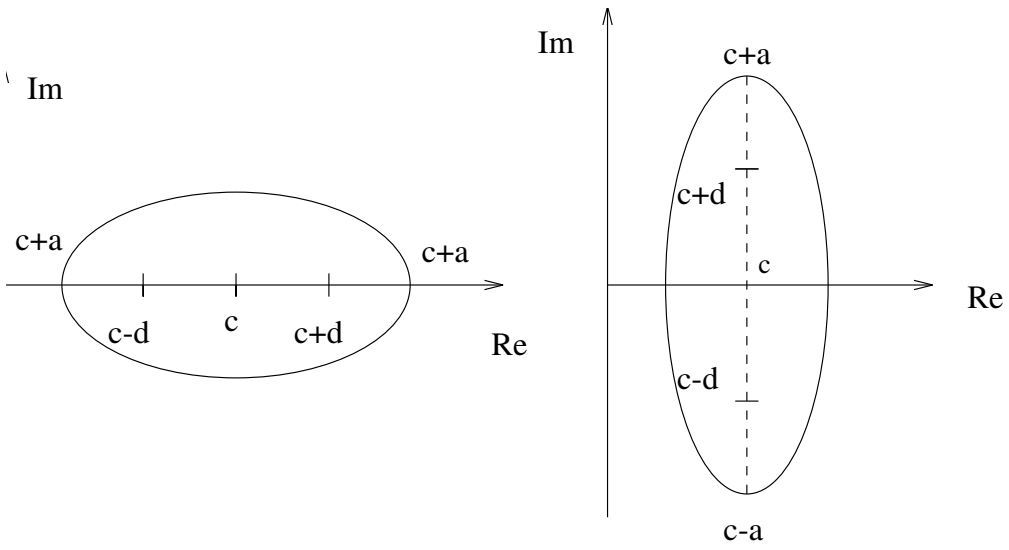
The results of approximation theory on near-optimal Chebyshev polynomials in the complex plane can now be used to obtain an upper bound for $\epsilon^{(m)}$. This is stated in the following corollary.

Corollary 1. *Assume that all the eigenvalues of A are located in the ellipse $E(c, d, a)$ which excludes the origin. Then*

$$\|r_m\|_2 \lesssim \kappa_2(V) \left(\frac{a + \sqrt{a^2 - d^2}}{c + \sqrt{c^2 - d^2}} \right)^m \|r_0\|_2.$$

PROOF: see [Y. Saad, Iterative methods for sparse linear systems, Second Edition, SIAM, 2003]

The follow plots show the spectrum of A is contained in the ellipses $E(c, d, a)$ with center c, focal distance d and major semi axis a. The left plot is for the case of real d and the right plot is for the case of purely imaginary d.



Since the condition number $\kappa_2(V)$ is typically not known and can be very large, results are of limited practical interest. They can be useful one when it is known that the matrix is nearly normal, in which case, $\kappa_2(V) \approx 1$.

Part III. Lanczos process, Conjugate Gradient method

1. The *Lanczos procedure* can be regarded as a simplification of Arnoldi's procedure when A is symmetric.

   By an order-m Arnoldi decomposition, we know that

   $$H_m = V_m^T A V_m.$$

   If A is symmetric, then H_m becomes symmetric tridiagonal. This simple observation leads to the following procedure to compute an orthonormal basis V_m of Krylov subspace $\mathcal{K}_m(A, v)$ when A is symmetric[5]:

   1. $v_1 = v/\|v\|_2$, set $\beta_1 = 0$, $v_0 = 0$
   2. for $j = 1, 2, \ldots, m$
   3. $\quad w = Av_j - \beta_j v_{j-1}$
   4. $\quad \alpha_j = v_j^T w$
   5. $\quad w := w - \alpha_j v_j$
   8. $\quad \beta_{j+1} = \|w\|_2$
   9. $\quad$ If $\beta_{j+1} = 0$, then *stop*
   10. $\quad v_{j+1} = w/\beta_{j+1}$
   11. endfor

   Remarks:

   - Only three vectors must be saved in the inner loop of the procedure. This is referred to as a *three-term recurrence*.
   - The computed Lanczos vectors $\{v_i\}$ are orthogonal in exact arithmetic. In the presence of finite precision, it could start losing such orthogonality rapidly with the increase of j. (The same phenomenon is also observed in the Arnoldi procedure, but it's not as severe

[5]Note that we change the notation $\alpha_j = h_{jj}$ and $\beta_{j+1} = h_{j-1,j}$, comparing with the Arnoldi procedure.

as in the Lanczos procedure). There has been much research devoted to understanding the effect of loss of the orthogonality, and finding ways to either recover the orthogonality, or to at last diminish its effects. An excellent reference on the subject is [B. N. Parlett, The Symmetric Eigenvalue Problem, SIAM Press, 1998].

2. In the matrix form, the Lanczos procedure can be expressed in the following governing equations, referred to as an *order−m Lanczos decomposition*:

$$\begin{aligned} AV_m &= V_m T_m + \beta_{m+1} v_{m+1} e_m^T \\ &= V_{m+1} \widehat{T}_m \end{aligned}$$

where $V_m = [v_1, v_2, \ldots, v_m]$, $V_{m+1} = [V_m, v_{m+1}]$, and

$$T_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \ddots & & \\ & \beta_3 & \ddots & \beta_{m-1} & \\ & & \ddots & \alpha_{m-1} & \beta_m \\ & & & \beta_m & \alpha_m \end{bmatrix} \quad \text{and} \quad \widehat{T}_m = \begin{bmatrix} T_m \\ \beta_{m+1} e_m^T \end{bmatrix}.$$

By the orthogonlity properties $V_m^T V_m = I$ and $V_m^T v_{m+1} = 0$, we have $V_m^T A V_m = T_m$.

3. The **Conjugate Gradient (CG) method** is the best known *iterative* technique for solving large scale SPD linear systems $Ax = b$, first published in 1952 by Hestenes and Stiefel.[6] There are several ways to derive the CG method. In terms of our familiar subspace projection technique, we can describe the CG method in one sentence:

   The CG method is a realization of an orthogonal projection technique onto the Krylov subspace $\mathcal{K}_m(A, r_0)$, where $r_0 = b - Ax_0$ with initial guess x_0.

   In this note, we provide a derivation of the CG method under this algorithmic framework.

   There are many different derivations of the CG method, for example, see the following paper

   - Jonathan Shewchuk, *An Introduction to Conjugate Gradient Method Without the Agonizing Pain*. 1994 (64 pages) (pdf file is available at the class website)

4. Before we derive the CG method, we first derive a so-called **direct Lanczos method**.

   Using the subspace projection technique, with an initial guess x_0, the approximate solution obtained from an orthogonal projection method onto $x_0 + \mathcal{K}_m(A, r_0)$ is given by

$$x_m = x_0 + V_m y_m, \tag{5}$$

   where y_m is the solution of the tridiagonal system

$$T_m y_m = \|r_0\|_2 e_1. \tag{6}$$

5. Now, let's try to compute the solution of the tridiagonal system (6) *progressively* along with the Lanczos procedure. For doing so, let's write the LU factorization of T_m as

$$T_m = L_m U_m,$$

[6] M. R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Standards, 49:409–436, 1952.

i.e. the Gaussian elimination without pivoting:

$$T_m = L_m U_m = \begin{bmatrix} 1 & & & & \\ \lambda_2 & 1 & & & \\ & \lambda_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & \lambda_m & 1 \end{bmatrix} \begin{bmatrix} \eta_1 & \beta_2 & & & \\ & \eta_2 & \beta_3 & & \\ & & \ddots & \ddots & \\ & & & \eta_{m-1} & \beta_m \\ & & & & \eta_m \end{bmatrix},$$

where $\eta_1 = \alpha_1$, and for $j = 2, 3, \ldots, m$,

$$\lambda_j = \beta_j/\eta_{j-1}, \qquad \eta_j = \alpha_j - \lambda_j \beta_j.$$

Then x_m is given by

$$\begin{aligned} x_m &= x_0 + V_m U_m^{-1} L_m^{-1}(\|r_0\|_2 e_1) \\ &\equiv x_0 + P_m z_m. \end{aligned}$$

where $P_m = V_m U_m^{-1}$ and $z_m = L_m^{-1}(\|r_0\|_2 e_1)$.

The following two observations connect P_m and z_m of the mth step with P_{m-1} and z_{m-1} of the previous step.

Observation A. Let us write $P_m = [P_{m-1}\ p_m]$, where p_m is the last column of P_m, then we have

$$\begin{aligned} P_m = [P_{m-1}\ p_m] &= V_m U_m^{-1} = \begin{bmatrix} V_{m-1} & v_m \end{bmatrix} \begin{bmatrix} U_{m-1} & \beta_m e_{m-1} \\ & \eta_m \end{bmatrix}^{-1} \\ &= \begin{bmatrix} V_{m-1} & v_m \end{bmatrix} \left[\begin{array}{c|c} U_{m-1}^{-1} & -U_{m-1}^{-1}(\beta_m e_{m-1})\eta_m^{-1} \\ \hline & \eta_m^{-1} \end{array}\right] \\ &= \begin{bmatrix} V_{m-1}U_{m-1}^{-1} & -V_{m-1}U_{m-1}^{-1}(\beta_m e_{m-1})\eta_m^{-1} + v_m\eta_m^{-1} \end{bmatrix} \\ &= \begin{bmatrix} P_{m-1} & -P_{m-1}(\beta_m e_{m-1})\eta_m^{-1} + v_m\eta_m^{-1} \end{bmatrix} \\ &= \begin{bmatrix} P_{m-1} & \eta_m^{-1}(v_m - \beta_m p_{m-1}) \end{bmatrix} \end{aligned}$$

Therefore, we see that the vector p_m can be computed from previous p_{m-1} and v_m by the simple update

$$p_m = \eta_m^{-1}(v_m - \beta_m p_{m-1}), \tag{7}$$

Observation B. By the definition of the vector z_m, we have

$$\begin{aligned} z_m &= L_m^{-1}(\|r_0\|_2 e_1) = \left[\begin{array}{c|c} L_{m-1}^{-1} & \\ \hline -\lambda_m e_{m-1}^T L_{m-1}^{-1} & 1 \end{array}\right] \begin{bmatrix} \|r_0\|_2 e_1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} L_{m-1}^{-1}(\|r_0\|_2 e_1) \\ -\lambda_m e_{m-1}^T L_{m-1}^{-1}(\|r_0\|_2 e_1) \end{bmatrix} \equiv \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} \end{aligned}$$

where $\zeta_m = -\lambda_m \zeta_{m-1}$ (here we use the definition that ζ_m is the last component of z_m).

As a result of these two observations, x_m can be written in an updated form

$$\begin{aligned} x_m &= x_0 + P_m z_m \\ &= x_0 + [P_{m-1}\ p_m] \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} \\ &= x_0 + P_{m-1} z_{m-1} + \zeta_m p_m \\ &= x_{m-1} + \zeta_m p_m. \end{aligned} \tag{8}$$

This gives the following direct Lanczos algorithm:

DIRECT LANCZOS METHOD

1. compute $r_0 = b - Ax_0$, $\zeta_1 = \|r_0\|_2$, and $v_{=}r_0/\zeta_1$
2. set $\lambda_1 = \beta_1 = 0$, $p_0 = 0$
3. for $m = 1, 2, \ldots,$
4. $\quad w := Av_m - \beta_m v_{m-1}$ and $\alpha_m = v_m^T w$
5. $\quad$ If $m > 1$ then compute $\lambda_m = \beta_m/\eta_{m-1}$ and $\zeta_m = -\lambda_m \zeta_{m-1}$
6. $\quad \eta_m = \alpha_m - \lambda_m \beta_m$
7. $\quad p_m = \eta_m^{-1}(v_m - \beta_m p_{m-1})$
8. $\quad x_m = x_{m-1} + \zeta_m p_m$
9. $\quad$ If x_m has converged, then Stop
10. $\quad w := w - \alpha_m v_m$
11. $\quad \beta_{m+1} = \|w\|_2$ and $v_{m+1} = w/\beta_{m+1}$
12. endfor

6. Toward the CG method, let us examine the residual vector r_m of the approximate solution x_m,

$$\begin{aligned} r_m &= b - Ax_m = b - A(x_0 + V_m y_m) = r_0 - AV_m y_m \\ &= r_0 - (V_m T_m + \beta_{m+1} v_{m+1} e_m^T) y_m \\ &= r_0 - V_m T_m y_m - \beta_{m+1} v_{m+1}(e_m^T y_m) \\ &= -\beta_{m+1}(e_m^T y_m) v_{m+1}. \end{aligned}$$

Therefore, we see that the residual vector r_m is in the direction of v_{m+1}. Since $\{v_i\}$ are orthogonal, we conclude that

$$\text{the residual vectors } \{r_i\} \text{ are orthogonal, i.e., } r_j^T r_i = 0 \text{ for } i \neq j. \tag{9}$$

7. Next we note that $P_m^T A P_m$ is a diagonal matrix. In fact,

$$P_m^T A P_m = U_m^{-T} V_m^T A V_m U_m^{-1} = U_m^{-T} T_m U_m^{-1} = U_m^{-T} L_m U_m U_m^{-1} = U_m^{-T} L_m.$$

Note that $U^{-T} L_m$ is a lower triangular which is also symmetric. Therefore it must be a diagonal matrix. By the fact that $P_m^T A P_m$ is diagonal, we conclude that

$$\text{the vectors } \{p_i\} \text{ are } A\text{-conjugate, i.e., } p_j^T A p_i = 0 \text{ for } i \neq j. \tag{10}$$

8. A consequence of the orthogonality condition (9) and conjugacy condition (10) is that a version of the algorithm can be derived by directly imposing the conditions (9) and (10). This gives us the **Conjugate Gradient (CG) algorithm**.

We now drive this. By the relation (8), let us express the $j+1$-th approximate vector x_{j+1} as

$$x_{j+1} = x_j + \theta_j p_j,$$

Then the corresponding residual vector satisfies

$$r_{j+1} = b - Ax_{j+1} = b - A(x_j + \theta_j p_j) = r_j - \theta_j A p_j. \tag{11}$$

Since the r_j's are orthogonal, i.e., $r_j^T r_{j+1} = 0$, then it gives

$$\theta_j = \frac{r_j^T r_j}{r_j^T A p_j} \tag{12}$$

By the relation (7) and noting that v_j is in the direction of r_{j+1}, it is known that the next search direction p_{j+1} is a linear combination of r_{j+1} and p_j. Therefore, we can write

$$p_{j+1} = r_{j+1} + \tau_j p_j.$$

Thus a first consequence of the above relation is that

$$r_j^T A p_j = (p_j - \tau_{j-1} p_{j-1})^T A p_j = p_j^T A p_j.$$

Therefore the scalar θ_j in (12) can be rewritten as

$$\theta_j = \frac{r_j^T r_j}{p_j^T A p_j}.$$

The second consequence is that by imposing A-conjugacy $p_{j+1}^T A p_j = 0$, we have

$$\tau_j = -\frac{p_j^T A r_{j+1}}{p_j^T A p_j}$$

Note that from (11), $Ap_j = -\frac{1}{\theta_j}(r_{j+1} - r_j)$ and therefore we have the following simplified expression for the scalar τ_j:

$$\tau_j = \frac{1}{\theta_j} \frac{(r_{j+1} - r_j)^T r_{j+1}}{p_j^T A p_j} = \frac{r_{j+1}^T r_{j+1}}{r_j^T r_j}$$

Putting these relations together gives the following CG algorithm

Conjugate Gradient (CG) Method
1. select initial approximation x_0, compute $r_0 = b - Ax_0$ and set $p_0 := r_0$
2. for $j = 0, 1, 2, \ldots,$ until convergence do
3. $\quad \theta_j = r_j^T r_j / (p_j^T A p_j)$
4. $\quad x_{j+1} = x_j + \theta_j p_j$
5. $\quad r_{j+1} = r_j - \theta_j A p_j$
6. $\quad \tau_j = r_{j+1}^T r_{j+1} / (r_j^T r_j)$
7. $\quad p_{j+1} = r_{j+1} + \tau_j p_j$
8. endfor

Note that in addition to the matrix A, only four vectors of storage (workspace) are required: x, p, Ap and r.

9. Further reading on convergence analysis of the CG method

(a) From the optimality of the projection technique, we know that the approximate solution obtained from the m-th step of the CG algorithm minimizes the A-norm of the error in the affine subspace $x_0 + \mathcal{K}_m(A, r_0)$. Since $\mathcal{K}_m$ is the set of all vectors of the form $x_0 + q(A)r_0$, where q is a polynomial of degree $\leq m-1$, we conclude the following lemma which characterizes the approximate solution x_m:

Lemma 1. *Let x_m be the approximate solution obtained from the m-th step of the CG algorithm, and let $d_m = x_* - x_m$ where x_* is the exact solution of $Ax = b$. Then x_m is of the form*

$$x_m = x_0 + q_m(A) r_0$$

where q_m is a polynomial of degree $m-1$ such that

$$\|(I - A q_m(A)) d_0\|_A = \min_{q \in \mathcal{P}_{m-1}} \|(I - Aq(A)) d_0\|_A$$

(b) From Lemma 1, we have the following theorem.

Theorem 3. *Let x_m be the approximate solution obtained from the m-th step of the CG algorithm, and x_* is the exact solution of $Ax = b$. Then,*

$$\|x_* - x_m\|_A \leq \frac{1}{T_m(1+2\eta)} \|x_* - x_0\|_A, \tag{13}$$

where T_m is the Chebyshev polynomial of degree m, and $\eta = \lambda_{\min}/(\lambda_{\max} - \lambda_{\min})$. $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalues of A.

A slightly different formulation of inequality can be derived. Using the relation

$$T_m(t) = \frac{1}{2}\left[\left(t + \sqrt{t^2-1}\right)^m + \left(t + \sqrt{t^2-1}\right)^{-m}\right] \geq \frac{1}{2}\left(t + \sqrt{t^2-1}\right)^m.$$

Then

$$T_m(1+2\eta) \geq \frac{1}{2}\left(1 + 2\eta + \sqrt{(1+2\eta)^2 - 1}\right)^m = \frac{1}{2}\left(1 + 2\eta + 2\sqrt{\eta(\eta+1)}\right)^m.$$

Now notice that

$$\begin{aligned} 1 + 2\eta + 2\sqrt{\eta(\eta+1)} &= (\sqrt{\eta} + \sqrt{\eta+1})^2 = \frac{(\sqrt{\lambda_{\min}} + \sqrt{\lambda_{\max}})^2}{\lambda_{\max} - \lambda_{\min}} \\ &= \frac{\sqrt{\lambda_{\max}} + \sqrt{\lambda_{\min}}}{\sqrt{\lambda_{\max}} - \sqrt{\lambda_{\min}}} = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \end{aligned}$$

where κ is the condition number of A, $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$. Substituting into the inequality (13) yields

$$\|x_* - x_m\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|x_* - x_0\|_A.$$

This bound is similar to that of the steepest descent algorithm except that the condition number of A is now replaced by its square root. CG method could be of order of magnitudes faster than the steepest descent algorithm. For example, let $\kappa = 10^3$, if one wants

$$\left(\frac{k-1}{k+1}\right)^{m_1} = \left(\frac{\sqrt{k}-1}{\sqrt{k}+1}\right)^{m_2} = 10^{-2}$$

then it means that the steepest descent algorithm needs to take $m_1 \approx 2300$ iterations to reach the same level of accuracy as $m_2 \approx 73$ iterations of the CG method.

(c) The above analysis using the condition number may not explain all the convergence behavior of CG. In fact, the entire distribution of eigenvalues of A is important, not just the ratio of the largest to the smallest one. If the largest and smallest eigenvalues of A are few in number (or clustered closely together), then CG will converge much more quickly than the above analysis based just on A's condition number would indicate. Any important fact is that the behavior of CG in floating point arithmetic can differ significantly from its behavior in exact arithmetic[7].

[7] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, 1997.
H. van der Vorst, Iterative methods for large linear systems, Cambridge University Press, 2003

Part IV. Preconditioning techniques

1. By the convergence analysis of CG and GMRES algorithms, we learn that the convergence rate strongly depends on the condition number of the coefficient matrix A of the linear system $Ax = b$, and the distribution of A's eigenvalues. Other Krylov subspace methods share the similar property.

2. *Preconditioning* means replacing the system $Ax = b$ with the modified systems

$$M^{-1}Ax = M^{-1}b. \tag{14}$$

or

$$AM^{-1}\hat{x} = b, \quad x = M^{-1}\hat{x}. \tag{15}$$

These are referred to as left and right preconditioning, respectively.

If the preconditioner M is SPD, then one can precondition symmetrically and solve the modified linear system

$$L^{-1}AL^{-T}y = L^{-1}b, \quad x = L^{-T}y, \tag{16}$$

where $M = LL^T$. The matrix L could be the Cholesky factor of M or any other matrix satisfying $M = LL^T$.

3. The desired *preconditioner* M should be chosen so that

   (a) $M^{-1}A$ or $L^{-1}AL^{-T}$ is "well-conditioned" or approximates "the identity matrix",

   (b) linear systems with coefficient matrix M are easy to solve.

   A careful choice of M can often make the condition number of the modified system much smaller than the condition number of the original one, and thus accelerate convergence dramatically. Indeed, a good preconditioner is often necessary for an iterative method to converge at all, and much past and current research in iterative methods is directed at finding better preconditioners.

4. We now show that a preconditioner can be easily incorporated into the CG method, and lead to a Preconditioned Conjugate Gradient method, PCG for short.

   If the CG algorithm is applied directly to the symmetric preconditioned system (16), the iterative kernels satisfy

$$\begin{aligned} y_{j+1} &= y_j + \hat{\alpha}_j \hat{p}_j \\ \hat{r}_{j+1} &= \hat{r}_j - \hat{\alpha}_j L^{-1}AL^{-T}\hat{p}_j \\ \hat{p}_{j+1} &= \hat{r}_{j+1} + \hat{\beta}_j \hat{p}_j \end{aligned}$$

   with

$$\hat{\alpha}_j = \frac{\hat{r}_j^T \hat{r}_j}{\hat{p}_j^T L^{-1}AL^{-T}\hat{p}_j} \quad \text{and} \quad \hat{\beta}_j = \frac{\hat{r}_{j+1}^T \hat{r}_{j+1}}{\hat{r}_j^T \hat{r}_j}.$$

   Defining

$$x_j = L^{-T}y_j, \quad r_j = L\hat{r}_j, \quad p_j = L^{-T}\hat{p}_j.$$

   The iterative kernels become

$$\begin{aligned} x_{j+1} &= x_j + \alpha_j p_j \\ r_{j+1} &= r_j - \alpha_j A p_j \\ p_{j+1} &= M^{-1}r_{j+1} + \beta_j p_j \end{aligned}$$

with

$$\alpha_j = \frac{r_j^T M^{-1} r_j}{p_j^T A p_j} \quad \text{and} \quad \beta_j = \frac{r_{j+1}^T M^{-1} r_{j+1}}{r_j^T M^{-1} r_j}.$$

We obtained the following preconditioned CG algorithm for solving $Ax = b$ using the preconditioner $M = LL^T$.

PRECONDITIONED CONJUGATE GRADIENT (PCG)

1. compute $r_0 = b - Ax_0$, solve $Mz_0 = r_0$ and $p_0 := z_0$
2. for $j = 0, 1, 2, \ldots,$ until convergence do
3. $\quad \alpha_j = (r_j^T z_j)/(p_j^T A p_j)$
4. $\quad x_{j+1} = x_j + \alpha_j p_j$
5. $\quad r_{j+1} = r_j - \alpha_j A p_j$
6. $\quad$ solve $Mz_{j+1} = r_{j+1}$
7. $\quad \beta_j = (r_{j+1}^T z_{j+1})/(r_j^T z_j)$
8. $\quad p_{j+1} = z_{j+1} + \beta_j p_j$
9. endfor

5. Similarly, a preconditioner can be easily incorporated into the GMRES method, and lead to a Preconditioned GMRES method, PGMRES for short.

PRECONDITIONED GMRES

1. compute $r_0 = M^{-1}(b - Ax_0)$, $\beta = \|r_0\|_2$ and $v_1 := r_0/\beta$
2. for $j = 0, 1, 2, \ldots, m$ do
3. $\quad$ solve $Mw = Av_j$
4. $\quad$ for $i = 1, 2, \ldots, j$ do
5. $\quad\quad h_{ij} = v_i^T w$
6. $\quad\quad w := w - h_{ij} w_i$
7. $\quad$ end do
8. $\quad$ compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
9. end do
10. let y_m be the solution of $\min_y \|\beta e_1 - \widehat{H}_m y\|_2$
11. $x_m = x_0 + V_m y_m$
12. If satisfied, **Stop**, else set $x_0 := x_m$ and go to 1.

Note that in the above algorithm, $V_m = [v_1, v_2, \ldots, v_m]$ and $\widehat{H}_m$ is a $(m+1) \times m$ upper triangular matrix with the entries h_{ij} computed at steps 4 and 8.

6. Commonly used preconditioners

The reliability and robustness of iterative techniques, when dealing with various applications, often depends much more on the quality of the preconditioner than on the particular Krylov subspace methods used. Finding a good preconditioner to solve a given sparse linear system is oftne viewed as a combination of art and science. Preconditioners can be divided roughly into three categories:

I. Preconditioners designed for general classes of matrices; e.g. Jacobi, Gauss-Seidel, SOR, incomplete LU factorization, incomplete Cholesky decomposition, approximate inverse.

II. Preconditioners designed for broad classes of underlying problems; e.g. elliptic partial differential equations (such as Poisson equation). Examples are multigrid and domain decomposition preconditioners.

III. Preconditioners designed for a specific matrix or underlying problem; e.g. for the transport equation.

The best choice of a preconditioner is generally application problem-dependent, and also depends on the iterative method being used.

- For CG and related methods to solve a symmetric positive definite system, one would like the condition number of the symmetrically preconditioned matrix $L^{-1}AL^{-T}$ to be close to one, in order for the error bound based on the Chebyshev polynomial to be small, or alternatively, has few extreme eigenvalues.
- For GMRES, a preconditioned matrix that is close to normal and whose eigenvalues are tightly clustered around some point away from the origin would be good, but other properties might also suffice to define a good preconditioner.

7. ILU Factorization Preconditioners.

Except for diagonal matrices, the solution of the linear system with coefficient matrix M requires that we have a suitable decomposition of M. In many instances this will be an LU decomposition. The idea of an incomplete LU preconditioner is to perform an abbreviated (sparse) form of Gaussian elimination of A and to declare the production of the resulting factors to be M. Since M is by construction already factorized, system involving M will be easy to solve.

Let us first introduce a *sparsity set* $\mathcal{Z}$ to control the patterns of zeros. Specifically, let $\mathcal{Z}$ be a set of ordered pairs of integers from $\{1, 2, \ldots, n\}$ containing no pairs of the form (i, i). An incomplete LU factorization of A is a decomposition of the form

$$A = LU + E, \tag{17}$$

where L is unit lower triangular, and U is upper triangular, and L, U and E have the following properties

(a) If $(i, j) \in \mathcal{Z}$ with $i > j$, then $\ell_{ij} = 0$,

(b) If $(i, j) \in \mathcal{Z}$ with $i < j$, then $u_{ij} = 0$,

(c) If $(i, j) \notin \mathcal{Z}$, then $e_{ij} = 0$.

In other words, the elements of L and U are zero on the sparsity set $\mathcal{Z}$, and off the sparsity set the decomposition reproduces A.

It is instructive to consider two extreme cases. (1) If the sparsity $\mathcal{Z}$ set is empty, we get the LU decomposition of A, i.e., we are using A as a preconditioner. (2) If $\mathcal{Z}$ is everything except diagonal pairs of the form (i, i), then we are effectively using the diagonal of A as a preconditioner.

Let us consider an ILU algorithm to generate L and U rowwise. Suppose we have computed the first $k-1$ rows of L and U, and we wish to compute the kth row. Write the first k rows of (17) in the form

$$\begin{bmatrix} A_{11} & A_{1k} \\ a_{k1}^T & a_{kk}^T \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ l_{1k}^T & 1 \end{bmatrix} \begin{bmatrix} U_{11} & U_{1k} \\ 0 & u_{kk}^T \end{bmatrix} + \begin{bmatrix} E_{11} & E_{1k} \\ e_{k1}^T & e_{kk}^T \end{bmatrix}.$$

we need to compute l_{1k}^T and u_{kk}^T. Multiplying out, we find that

$$l_{1k}^T U_{11} + e_{k1}^T = a_{k1}^T \tag{18}$$

and

$$u_{kk}^T + e_{kk}^T = a_{kk}^T - l_{1k}^T U_{1k}$$

We then can solve these two systems in order:

$$\underbrace{\ell_{k1}, \ell_{k2}, \ldots, \ell_{k,k-1}}_{l_{1k}^T}, \underbrace{\nu_{kk}, \nu_{k,k+1}, \ldots, \nu_{k,n}}_{u_{kk}^T}.$$

Suppose that we have computed $\ell_{k1}, \ell_{k2}, \ldots, \ell_{k,j-1}$. If $(k,j) \in \mathcal{Z}$, then set $\ell_{kj} = 0$. If $(k,j) \notin \mathcal{Z}$, then $e_{kj} = 0$, and the equation (18) gives

$$\alpha_{kj} = \sum_{i=1}^{k-1} \ell_{ki}\nu_{ij} + \ell_{kj}\nu_{jj},$$

from which we get

$$\ell_{kj} = \frac{\alpha_{kj} - \sum_{i=1}^{k-1} \ell_{ki}\nu_{ij}}{\nu_{jj}}.$$

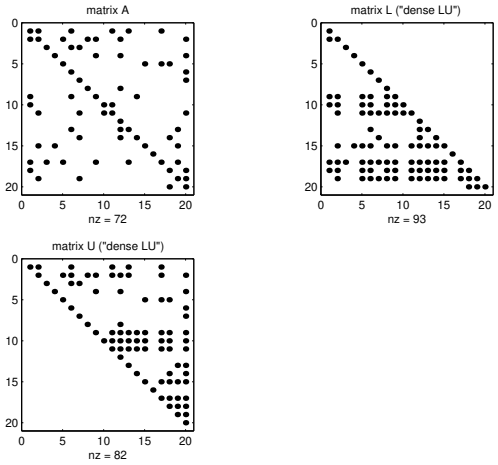
The key observation here is that it does not matter how the values of the preceding ℓ's and ν's were determined. If ℓ_{kj} is defined in this way, then when we compute LU, its (k,j)-element will be α_{kj}. Thus we set ℓ's and ν's to zero on the sparsity set without interfering with the values of LU off the sparsity set. A similar procedure applies to the determination of $\nu_{kk}, \nu_{k,k+1}, \ldots, \nu_{k,n}$.

```
INCOMPLETE_LU_FACTORIZATION(A, Z)
1.  for k = 1 to n
2.      for j = 1 to k − 1
3.          if ((k, j) ∈ Z)
4.              L(k, j) = 0
5.          else
6.              L(k, j) = (A(k, j) − L(k, 1 : j − 1) ∗ U(1 : j − 1, j))/U(j, j)
7.          end if
8.      end for j
9.      for j = k to n
10.         if ((k, j) ∈ Z)
11.             U(k, j) = 0
12.         else
13.             U(k, j) = (A(k, j) − L(k, 1 : k − 1) ∗ U(1 : k − 1, j)
14.         end if
15.     end for j
16. end for k
```

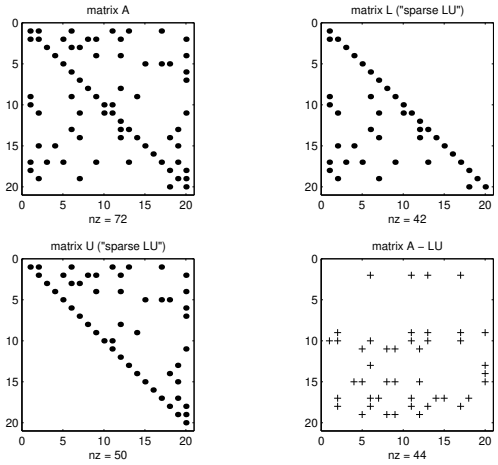
The algorithm can be carried to completion provided the quantities $U(j,j)$ are all nonzero, in which case the decomposition is unique. Whether or not the $U(j,j)$ are nonzero will depend on the matrix in question.

The following figure compares the sparsity of LU and ILU factorizations of a sparse 20 by 20 matrix

The sparsity of LU



The sparsity of ILU and E-factor



8. Not all matrices can have an ILU factorization. The following two classes of matrices, the algorithm always works.

   (a) If A is nonsingular diagonally dominant matrix, then A has an incomplete LU factorization for any sparsity set $\mathcal{Z}$.

   Note: A matrix A of order n is *diagonally dominant* if

$$|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|, \qquad \text{for } i = 1, 2, \ldots, n.$$

   It is strictly diagonally dominant if strictly inequality holds for all j.

   It can be shown that *A strictly diagonally dominant matrix is nonsingular*. Be aware that diagonal dominance alone does not imply either nonsingularity or singularity. For examples, let

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

   Then A is nonsingular. On the other hand, B is singular.

   (b) The incomplete LU factorization also exists for any M-matrix.

   Note: A matrix is said to be an M-matrix if it satisfies the following properties:

   (1) $a_{ii} > 0$ for $i = 1, \ldots n$,

   (2) $a_{ij} \leq 0$ for $i \neq j, i, j = 1, \ldots n$,

   (3) A is nonsingular and

   (4) A^{-1} is a nonnegative matrix (all entries are nonnegative).

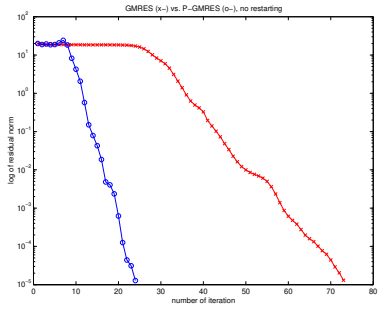
9. Block preconditioner is a popular technique for block-tridiagonal matrices arising from the discretization of elliptic problems, such as Poisson's equation. It can be also be generalized to other sparse matrices. For example, the matrix arises in the solution of 2D Poisson's equation has the form

$$A = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix}$$

where T is a symmetric tridiagonal matrix, with diagonal entres all 4, and off diagonal entries all -1. In this case, a natural preconditioner is

$$M = \text{diag}(T, T, \ldots, T).$$

10. The following figure shows the convergence history of GMRES with and without preconditioning for solving a linear system of equations arising from a discretization of a model convection-diffusion equation. The preconditioner used here is ILU(0), i.e., ILU factorization with the same sparsity pattern of A.



11. Iterative methods in Matlab

functions	methods
`pcg`	Preconditioned Conjugate Gradients Method.
`gmres`	Generalized Minimum Residual Method.
`bicg`	BiConjugate Gradients Method.
`bicgstab`	BiConjugate Gradients Stabilized Method.
`cgs`	Conjugate Gradients Squared Method.
`minres`	Minimum Residual Method.
`qmr`	Quasi-Minimal Residual Method.
`symmlq`	Symmetric LQ Method.

Preconditioners

functions	preconditioners
`luinc`	Incomplete LU factorization.
`cholinc`	Incomplete Cholesky factorization.

12. Further Reading

- Yousef Saad, Iterative Methods for Sparse Linear Systems, 2nd Edition, SIAM, 2003
- H. van der Vorst, Iterative Krylov Methods for Large Linear Systems, Cambridge Univ. Press, 2003
- R. Barrett *et al*, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, 1994