ECS 231

Subspace projection methods for LS

Part I. Basics

The landscape of solvers for linear systems of equations

$$Ax = b,$$

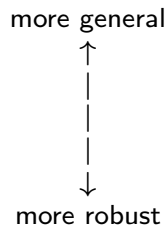
Part I. Basics

The landscape of solvers for linear systems of equations

$$Ax = b,$$

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

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



Part I. Basics

A framework for subspace projection methods.

- The basic idea:
  - extract an approximate solution $\widetilde{x}$ from a subspace of $\mathbb{R}^n$.
  - a technique of *dimension reduction*.

Part I. Basics

A framework for subspace projection methods.

- The basic idea:
  - extract an approximate solution $\widetilde{x}$ from a subspace of $\mathbb{R}^n$.
  - a technique of *dimension reduction*.
- Mathematically, let $\mathcal{W}, \mathcal{V} \subseteq \mathbb{R}^n$, and x_0 is an initial guess of the solution, then the *subspace projection technique* is to

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

Part I. Basics

A framework for subspace projection methods.

- The basic idea:
  - extract an approximate solution $\widetilde{x}$ from a subspace of $\mathbb{R}^n$.
  - a technique of *dimension reduction*.
- Mathematically, let $\mathcal{W}, \mathcal{V} \subseteq \mathbb{R}^n$, and x_0 is an initial guess of the solution, then the *subspace projection technique* is to

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

In other words, let $r_0 = b - Ax_0$, then

$$b - A\widetilde{x} = b - A(x_0 + z) = r_0 - Az.$$

(1) is equivalent to

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

Part I. Basics

A framework for subspace projection methods.

- The basic idea:
  - extract an approximate solution $\widetilde{x}$ from a subspace of $\mathbb{R}^n$.
  - a technique of *dimension reduction*.
- Mathematically, let $\mathcal{W}, \mathcal{V} \subseteq \mathbb{R}^n$, and x_0 is an initial guess of the solution, then the *subspace projection technique* is to

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

  In other words, let $r_0 = b - Ax_0$, then

$$b - A\widetilde{x} = b - A(x_0 + z) = r_0 - Az.$$

  (1) is equivalent to

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

- *Orthogonal projection*: $\mathcal{W} = \mathcal{V}$,
- *Oblique projection*: $\mathcal{W} \neq \mathcal{V}$,

Part I. Basics

In matrix notation, let $V = [v_1, v_2, \ldots, v_m]$ be a basis of $\mathcal{V}$, and $W = [w_1, w_2, \ldots, w_m]$ be a basis of $\mathcal{W}$. Then any approximation solution

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

and the orthogonality condition (1a) implies

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

Part I. Basics

In matrix notation, let $V = [v_1, v_2, \ldots, v_m]$ be a basis of $\mathcal{V}$, and $W = [w_1, w_2, \ldots, w_m]$ be a basis of $\mathcal{W}$. Then any approximation solution

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

and the orthogonality condition (1a) implies

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

Thus we have

$$V^T AW\, y = V^T r_0.$$

Thus assuming $V^T AW$ is invertible, a new approximate solution $\widetilde{x}$:

$$\widetilde{x} = x_0 + W(V^T AW)^{-1} V^T r_0.$$

Part I. Basics

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 $\mathbb{R}^n$
3. Generate basis matrices V and W for $\mathcal{V}$ and $\mathcal{W}$
4. $r_0 \leftarrow b - Ax_0$
5. $y \leftarrow (V^T AW)^{-1} V^T r_0$
6. $x_0 \leftarrow x_0 + Wy$

Prototype iterative subspace projection technique, *cont'd*

Remarks:

1. The matrix $V^T AW$ does not have to be formed explicitly, typically a by-product of Steps 2 and 3.

Prototype iterative subspace projection technique, *cont'd*

Remarks:

1. The matrix $V^T AW$ does not have to be formed explicitly, typically a by-product of Steps 2 and 3.
2. There are two important cases where the nonsingularity of $V^T AW$ is guaranteed:

Prototype iterative subspace projection technique, *cont'd*

Remarks:

1. The matrix $V^T AW$ does not have to be formed explicitly, typically a by-product of Steps 2 and 3.
2. There are two important cases where the nonsingularity of $V^T AW$ is guaranteed:
    1. If A is symmetric positive definite (SPD) and $\mathcal{W} = \mathcal{V}$, then $V^T AW = W^T AW$ is also SPD (and nonsingular).

Part I. Basics

Prototype iterative subspace projection technique, *cont'd*

Remarks:

1. The matrix $V^T AW$ does not have to be formed explicitly, typically a by-product of Steps 2 and 3.
2. There are two important cases where the nonsingularity of $V^T AW$ is guaranteed:
    1. If A is symmetric positive definite (SPD) and $\mathcal{W} = \mathcal{V}$, then $V^T AW = W^T AW$ is also SPD (and nonsingular).
    2. If A is nonsingular, and $\mathcal{V} = A\mathcal{W}$, then $V^T AW = W^T A^T AW$, which is SPD (and nonsingular).

Prototype iterative subspace projection technique in one-dimension:

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

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

$$\alpha = \frac{v^T r_0}{v^T Aw}.$$

Part I. Basics

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. $\quad x_{k+1} = x_k + \alpha_k r_k$

Steepest Descent (SD) method, *cont'd*

Remarks:

- Since A is SPD, $r_k^T A r_k > 0$ except $r_k = 0$.

Steepest Descent (SD) method, *cont'd*

Remarks:

- Since A is SPD, $r_k^T A r_k > 0$ except $r_k = 0$.
- Let $x_* = A^{-1}b$, then k-th step of the SD iteration minimizes

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

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

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

where $\nabla f(x_k) = b - Ax_k$ is the gradient of f at x_k.

Part I. Basics

Steepest Descent (SD) method, *cont'd*

Remarks:

- Since A is SPD, $r_k^T A r_k > 0$ except $r_k = 0$.
- Let $x_* = A^{-1}b$, then k-th step of the SD iteration minimizes

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

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

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

where $\nabla f(x_k) = b - Ax_k$ is the gradient of f at x_k.

- Recall that from Calculus, the negative of the gradient direction is locally the direction that yields the fastest rate of decrease for f.

Minimal Residual (MR) Iteration.

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

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

Part I. Basics

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. $\quad x_{k+1} = x_k + \alpha_k r_k$

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:

Part I. Basics

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 iteration minimizes

$$f(x) \equiv \|r\|_2^2 = \|b - Ax\|_2^2$$

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

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

Part II: Krylov subspace and GMRES

Krylov subspace is defined as

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

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 $m - 1$.

Part II: Krylov subspace and GMRES

Arnoldi procedure is an algorithm for building an orthonormal 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 $j = 1, 2, \ldots, m$
3. compute $w = Av_j$
4. for $i = 1, 2, \ldots, j$
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

Part II: Krylov subspace and GMRES

Proposition. 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 $\mathcal{K}_m(A, v)$:

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

Proof. By induction.

Part II: Krylov subspace and GMRES

Let

$$V_m = [v_1, v_2, \ldots, v_m] \quad \text{and} \quad H_m = (h_{ij}) = \text{Upper Hessenberg.}$$

Then in the matrix form, the Arnoldi procedure can be expressed by the following *order-m Arnoldi decompositions*:

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

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

In addition, we denote

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

Part II: Krylov subspace and GMRES

Remarks:

1. the matrix A is only referenced via the matvec 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 matvec.

Part II: Krylov subspace and GMRES

Remarks:

1. the matrix A is only referenced via the matvec 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 matvec.
2. The main storage requirement is $n(m+1)$ for storing Arnoldi vectors $\{v_i\}$ plus the storage requirements for A or the required matvec.

Part II: Krylov subspace and GMRES

Remarks:

1. the matrix A is only referenced via the matvec 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 matvec.
2. The main storage requirement is $n(m+1)$ for storing Arnoldi vectors $\{v_i\}$ plus the storage requirements for A or the required matvec.
3. The primary arithmetic cost is the cost of m matvecs plus $2m^2n$ for the rest. It is common that the matvec is the dominant cost.

Part II: Krylov subspace and GMRES

Remarks:

1. the matrix A is only referenced via the matvec 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 matvec.
2. The main storage requirement is $n(m+1)$ for storing Arnoldi vectors $\{v_i\}$ plus the storage requirements for A or the required matvec.
3. The primary arithmetic cost is the cost of m matvecs plus $2m^2n$ for the rest. It is common that the matvec is the dominant cost.
4. The procedure breaks down when $h_{j+1,j} = 0$ for some j. If it 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.

Part II: Krylov subspace and GMRES

Remarks:

1. the matrix A is only referenced via the matvec 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 matvec.
2. The main storage requirement is $n(m+1)$ for storing Arnoldi vectors $\{v_i\}$ plus the storage requirements for A or the required matvec.
3. The primary arithmetic cost is the cost of m matvecs plus $2m^2n$ for the rest. It is common that the matvec is the dominant cost.
4. The procedure breaks down when $h_{j+1,j} = 0$ for some j. If it 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.
5. 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*.

Part II: Krylov subspace and GMRES

- The Generalized Minimum Residual (GMRES)[1] 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

[1] 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.

Part II: Krylov subspace and GMRES

Derivation of GMRES using subspace projection

- Let

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

Then by the orthogonality condition, we have

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

i.e.,

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

which is equivalent to

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

- By *order-m Arnoldi decompositions*, 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 LS problem

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

for y

Part II: Krylov subspace and GMRES

An alternative derivation of GMRES by exploitng the optimality property.

- A vector x in $x_0 + \mathcal{K}_m$ can be written as $x = x_0 + V_m y$
- Define $J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y)\|_2$
- Using the Arnoldi decomposition (2), 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 (LS) problem

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

- The LS problem is inexpensive to compute since m is small.

Part II: Krylov subspace and GMRES

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.

Part II: Krylov subspace and GMRES

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.

Proposition. 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 $Ax = b$.

Part III. Lanczos process, Conjugate Gradient method

- 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

Part III. Lanczos process, Conjugate Gradient method

Lanczos procedure[2]:

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

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

Part III. Lanczos process, Conjugate Gradient method

Remarks:

- Only three vectors must be saved in the inner loop of the procedure. This is sometimes referred to as a *three-term recurrence*.
- In the presence of finite precision, it could start losing such orthogonality of v_j rapidly with the increase of j. 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[3]

[3] An excellent reference on the subject is [B. N. Parlett, The Symmetric Eigenvalue Problem, SIAM Press, 1998].

Part III. Lanczos process, Conjugate Gradient method

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

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

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 $V_m^T A V_m = T_m$.

Part III. Lanczos process, Conjugate Gradient method

- The **Conjugate Gradient (CG) method** is the best known *iterative* technique for solving large scale SPD linear system $Ax = b$[4]
- 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.
- An alternative derivation is given by Shewchuk[5]

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

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

Part III. Lanczos process, Conjugate Gradient method

- 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

$$x_m = x_0 + V_m y_m, \tag{3}$$

where y_m is the solution of the tridiagonal system

$$T_m y_m = \|r_0\|_2 e_1. \tag{4}$$

Part III. Lanczos process, Conjugate Gradient method

- Now, let's try to solve the tridiagonal system (4) *progressively* along with the Lanczos procedure.
- Let's write the LU factorization of T_m as

$$T_m = L_m U_m,$$

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

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

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

Part III. Lanczos process, Conjugate Gradient method

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

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

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

$$p_m = \eta_m^{-1}(v_m - \beta_m p_{m-1}), \tag{5}$$

Part III. Lanczos process, Conjugate Gradient method

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

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

where $\zeta_m = -\lambda_m \zeta_{m-1}$.

Part III. Lanczos process, Conjugate Gradient method

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

$$\begin{aligned} x_m &= x_0 + P_m z_m \\ &= x_0 + [P_{m-1}\ p_m] \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} \\ &= x_0 + P_{m-1} z_{m-1} + \zeta_m p_m \\ &= x_{m-1} + \zeta_m p_m. \end{aligned} \tag{6}$$

Part III. Lanczos process, Conjugate Gradient method

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

Part III. Lanczos process, Conjugate Gradient method

Toward the CG method

- The residual vector r_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 the residual vector r_m is in the direction of v_{m+1}.

- Since $\{v_i\}$ are orthogonal, we conclude that the residual vectors $\{r_i\}$ are orthogonal, i.e.,

$$r_j^T r_i = 0 \quad \text{for} \quad i \neq j. \tag{7}$$

Part III. Lanczos process, Conjugate Gradient method

Toward the CG method, cont'd,

- Note that $P_m^T A P_m$ is a diagonal matrix:

$$\begin{aligned} P_m^T A P_m &= U_m^{-T} V_m^T A V_m U_m^{-1} \\ &= U_m^{-T} T_m U_m^{-1} \\ &= U_m^{-T} L_m U_m U_m^{-1} \\ &= U_m^{-T} L_m. \end{aligned}$$

  Since $U^{-T} L_m$ is a lower triangular which is also symmetric. Therefore $P_m^T A P_m$ i must be a diagonal matrix.

- By the fact that $P_m^T A P_m$ is diagonal, we conclude that the vectors $\{p_i\}$ are A-conjugate, i.e.,

$$p_j^T A p_i = 0 \quad \text{and} \quad i \neq j. \tag{8}$$

Part III. Lanczos process, Conjugate Gradient method

Toward the CG method, cont'd,

A consequence of the orthogonality condition (7) and conjugacy condition (8) is that a version of the algorithm can be derived by directly imposing the conditions (7) and (8). This gives us the **Conjugate Gradient (CG) algorithm**.

Part III. Lanczos process, Conjugate Gradient method

The CG method

- By the relation (6), 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

$$r_{j+1} = b - Ax_{j+1} = b - A(x_j + \theta_j p_j) = r_j - \theta_j A p_j. \tag{9}$$

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

$$\theta_j = \frac{r_j^T r_j}{r_j^T A p_j} \tag{10}$$

Part III. Lanczos process, Conjugate Gradient method

The CG method, cont'd

- By the relation (5) 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

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

- first consequence

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

Therefore the scalar θ_j in (10) can be rewritten as

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

Part III. Lanczos process, Conjugate Gradient method

The CG method, cont'd

- second consequence: 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 (9),

$$Ap_j = -\frac{1}{\theta_j}(r_{j+1} - r_j)$$

and therefore we have the following simplified expression for the scalar τ_j:

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

Part III. Lanczos process, Conjugate Gradient method

The CG method, cont'd

CONJUGATE GRADIENT (CG) METHOD

1. select initial 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

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