Krylov Subspace Methods

Harris Enniss

University of California Santa Barbara

henniss@math.ucsb.edu

2014-10-29
Consider a linear system $Ax = b$, $A \in M^{n \times n}$.

- $n$ very large, $A$ sparse.
- Finite element and finite difference schemes for PDEs tend to produce such systems.
Motivation

Example: Triangular discretization for the Poisson problem on the square.

\[ h = \frac{1}{N+1} \]

In this case the linear system (1.21) reads as follows:

\[
\begin{bmatrix}
4 & -1 & 0 & -1 & 0 & \cdots & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & \cdots \\
0 & -1 & 4 & -1 & 0 & -1 & \cdots \\
-1 & 0 & -1 & 4 & -1 & 0 & \cdots \\
0 & \cdots & \cdots & \cdots & 0 & -1 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & \cdots & \cdots & -1 & 0 & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \cdots & \cdots & 0 & \cdots & \cdots & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\end{bmatrix}
\begin{bmatrix}
\xi_1 \\
\xi_2 \\
\xi_3 \\
\xi_4 \\
\xi_5 \\
\vdots \\
\xi_M \\
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
b_2 \\
b_3 \\
b_4 \\
b_5 \\
\vdots \\
b_M \\
\end{bmatrix}
\]

[Joh87]
Krylov Subspace methods are a class of iterative methods. 
\(x_0, \ldots, x_m\) lie in subspaces \(x_0 + K_m(A, v)\). 
\(K_m(A, v)\) is a Krylov subspace:

\[ K_m(A, v) = \text{span}\{v, Av, \ldots, A^{m-1}v\}. \]
What is a Krylov subspace method?

Each $x_j$ is chosen from $x_0 + \mathcal{K}_j(A, r_0)$ to satisfy

$$r_j = b - Ax_j \perp \mathcal{L},$$

where $r_0 = b - Ax_0$.

The choice of $\mathcal{L}$ will depend on method.
Historically, stability of these methods has been a large obstacle to adoption. Typically Krylov subspace methods will be applied with a preconditioner to help ensure convergence.
Conjugate Gradient (CG)

- Probably the best known Krylov Subspace method. Discovered by Hestenes and Stiefel (1952).
- CG requires $A$ be symmetric and positive definite.
- Makes use of a very short recurrence relation.
- Behind the scenes, it constructs an orthogonal basis for $K_m(A, x_0)$.
- For general non-symmetric matrices, we will be able to choose one or the other, but not both.
- To be continued...
Arnoldi’s method for linear systems

Originally developed by Arnoldi in 1951 to solve Eigenvalue problems. Unlike CG, it works for nonsymmetric matrices.

Construct an orthonormal basis for $\mathcal{K}_m(A, r_0)$ using Gram-Schmidt (or Householder) orthogonalization.

Project onto $\mathcal{L} = \mathcal{K}$.
1: \( r_0 = b - Ax_0, \beta = \|r_0\|_2, v_1 = r_0 / \beta. \)
2: Define an \( m \times m \) matrix \( H_m = \{h_{ij}\}_{i,j=1,...,m} \); set \( H_m = 0. \)
3: \textbf{for} \( j = 1, \ldots, m \) \textbf{do}
4: \hspace{1em} \( w_j = Av_j. \)
5: \hspace{1em} \textbf{for} \( j = 1, \ldots, j \) \textbf{do}
6: \hspace{2em} \( h_{ij} = (w_j, v_i). \)
7: \hspace{2em} \( w_j = w_j - h_{ij}v_i. \)
8: \hspace{1em} \textbf{end for}
9: \hspace{1em} \( h_{j+1,j} = \|w_j\|_2. \)
10: \hspace{1em} \textbf{if} \( h_{j+1,j} = 0 \) \textbf{then}
11: \hspace{2em} \( m = j, \textbf{go to} \ 15. \)
12: \hspace{1em} \textbf{end if}
13: \hspace{1em} \( v_{j+1} = w_j / h_{j+1,j}. \)
14: \hspace{1em} \textbf{end for}
15: \( y_m = H_m^{-1}(\beta e_1) \) and \( x_m = x_0 + V_my_m. \)
$H_m$ is the $m \times m$ Hessenberg matrix with entries \{ $h_{ij}$ \}. Let $V_m$ be the $n \times m$ orthogonal matrix with columns $v_i$. The construction ensures they satisfy the relation:

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

15: \( y_m = H_m^{-1}(\beta e_1) \) and \( x_m = x_0 + V_m y_m \).

\[ L_m = K_m(A, r_0), \text{ i.e. want } x_m \text{ such that} \]
\[ r_m = b - Ax_m \perp L_m = K_m. \]

Suppose \( x_m = x_0 + V_m y_m \), for some \( y_m \).

\[ 0 = V_m^T (b - Ax_m) \]
\[ = V_m^T (r_0 - AV_m y_m) \]
\[ = V_m^T (\beta v_1) - V_m^T AV_m y_m \]
\[ = \beta e_1 - H_m y_m \]
10: if $h_{j+1,.j} = 0$ then
11:     $m = j$, go to 15.
12: end if

What we call a “lucky breakdown”, signaling early convergence.
Arnoldi’s method is simple, but with GMRES, we can actually minimize the $\|r_m\|_2$ over $x_0 + K_m$.

We compute one extra column of $V_m$ to yield $V_{m+1}$, and one extra row of $H_m$ to yield $\bar{H}_m$.

We want to minimize

$$\|b - A(x_0 + V_m y)\|_2,$$

and

$$b - A(x_0 + V_m y) = r_0 - AV_m y = \beta v_1 - V_{m+1} \bar{H}_m y = V_{m+1}(\beta e_1 - \bar{H}_m y).$$

The columns of $V_{m+1}$ are orthonormal, so

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

Minimizing this turns out to equivalent to choosing $r_m \perp \mathcal{L} = AK_m$. 

Generalized Minimal RESidual (GMRES)
1: \( r_0 = b - Ax_0, \beta = \|r_0\|_2, \nu_1 = r_0/\beta. \)
2: Define an \( m \times m \) matrix \( H_m = \{h_{ij}\}_{i,j=1,...,m} \); set \( H_m = 0. \)
3: \textbf{for} \( j = 1, \ldots, m \) \textbf{do}
4: \( w_j = Av_j. \)
5: \textbf{for} \( i = 1, \ldots, j \) \textbf{do}
6: \( h_{ij} = (w_j, v_i). \)
7: \( w_j = w_j - h_{ij}v_i. \)
8: \textbf{end for}
9: \( h_{j+1,j} = \|w_j\|_2. \)
10: \textbf{if} \( h_{j+1,j} = 0 \) \textbf{then}
11: \( m = j, \text{ go to } 15. \)
12: \textbf{end if}
13: \( v_{j+1} = w_j/h_{j+1,j}. \)
14: \textbf{end for}
15: Define \( \bar{H}_m = \{h_{ij}\}_{1 \leq i \leq m+1, 1 \leq j \leq m} \)
16: Compute \( y_m \) which minimizes \( \|\beta e_1 - \bar{H}_m y\|_2 \), and \( x_m = x_0 + V_m y_m. \)
Both Arnoldi’s method and GMRES compute $j$ inner products at each step, each requiring $n$ multiplications. They also perform one matrix-vector multiplication.

The number of multiplications performed is thus $O(m^2 n + N(A)m)$, where $N(A)$ is the number of nonzero entries in $A$.

Great if $m$ is small; otherwise prohibitive.

If convergence is slow, we may need to use GMRES($m$): we compute $x_m$, then restart using $x_0 = x_m$ as our initial guess.
Overall, GMRES works well if convergence happens early in the iteration.
It’s also easier to analyze than many other Krylov subspace methods.
The residuals are guaranteed, at the very least, to be monotonic.
But it can be slow if we wind up needing to use a high dimensional subspace.
Lanczos Iteration

Let's go back to Arnoldi's method, and suppose $A$ is symmetric and positive definite. 

$H_m = V_m^T AV_m$, so $H_m$ is symmetric, i.e. tridiagonal.

Thus each step needs only two inner products.

---

1: $w_j = Av_j - \beta_j v_{j-1}$
2: $\alpha_j = (w_j, v_j)$
3: $w_j = w_j - \alpha v_j$
4: $\beta_j = ||w_j||_2$
5: $v_{j+1} = w_j / \beta_j$.

[Saa03]
But there’s still a problem. How to store $V_m$? What we’d really like is to update $x_m$ progressively, so we never need to store all of $V_m$. 
Lanczos Iteration

Previously, we computed

\[ x_m = x_0 + V_m(H_m^{-1}(\beta_0 e_1)). \]

Now \( H_m \) is symmetric and tridiagonal. Write

\[
H_m = \begin{bmatrix}
\alpha_1 & \beta_2 & 0 & \cdots & 0 \\
\beta_2 & \alpha_2 & \beta_3 \\
0 & \beta_3 & \alpha_3 & \ddots \\
\vdots & \vdots & \ddots & \ddots \\
0 & \cdots & \cdots & \cdots & \alpha_m
\end{bmatrix}
\]

\[
= \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
\lambda_2 & 1 & 0 \\
0 & \lambda_3 & 1 & \ddots \\
\vdots & \vdots & \ddots & \ddots \\
0 & \cdots & \cdots & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
\eta_1 & \beta_2 & 0 & \cdots & 0 \\
0 & \eta_2 & \beta_3 \\
0 & 0 & \eta_3 & \ddots \\
\vdots & \vdots & \ddots & \ddots \\
0 & \cdots & \cdots & \cdots & \eta_m
\end{bmatrix}
= L_m U_m
\]
Previously, we computed

\[ x_m = x_0 + V_m(H_m^{-1}(\beta_0 e_1)). \]

Now \( H_m \) is symmetric and tridiagonal. Write

\[ H_m = \begin{bmatrix}
\alpha_1 & \beta_2 & 0 & \cdots & 0 \\
\beta_2 & \alpha_2 & \beta_3 \\
0 & \beta_3 & \alpha_3 & \ddots \\
\vdots & \ddots & \ddots & \ddots \\
0 & \cdots & \cdots & \cdots & \alpha_m
\end{bmatrix} \]

\[ = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 \\
\lambda_2 & 1 & 0 & \cdots & 0 \\
0 & \lambda_3 & 1 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & \cdots & 1
\end{bmatrix} \begin{bmatrix}
\eta_1 & \beta_2 & 0 & \cdots & 0 \\
0 & \eta_2 & \beta_3 \\
0 & 0 & \eta_3 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & \cdots & \cdots & \eta_m
\end{bmatrix} = L_m U_m \]
Previously, we computed

\[ x_m = x_0 + V_m(H_m^{-1}(\beta_0 e_1)) \].

Now \( H_m \) is symmetric and tridiagonal. Write

\[
H_m = \begin{bmatrix}
\alpha_1 & \beta_2 & 0 & \ldots & 0 \\
\beta_2 & \alpha_2 & \beta_3 \\
0 & \beta_3 & \alpha_3 & \ddots \\
\vdots & \ddots & \ddots & \ddots \\
0 & \ldots & \ldots & \ldots & \alpha_m
\end{bmatrix}
\]

\[
= \begin{bmatrix}
1 & 0 & 0 & \ldots & 0 \\
\lambda_2 & 1 & 0 & \ldots & 0 \\
0 & \lambda_3 & 1 & \ldots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
0 & \ldots & \ldots & 1
\end{bmatrix}
\begin{bmatrix}
\eta_1 & \beta_2 & 0 & \ldots & 0 \\
0 & \eta_2 & \beta_3 \\
0 & 0 & \eta_3 & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots \\
0 & \ldots & \ldots & \ldots & \eta_m
\end{bmatrix}
= L_m U_m
Lanczos Iteration

\[ \lambda_m = \frac{\beta_m}{\eta_{m-1}}, \]
\[ \eta_m = \alpha_m - \lambda_m \beta_m. \]

If we define \( P_m = V_m U_m^{-1} \), we see that

\[ v_m = \eta_m p_m + \beta_m p_{m-1}. \]

Also, if \( z_m = L_m^{-1}(\beta e_1) \), then

\[ z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} \]

where \( \zeta_m = -\lambda_m \zeta_{m-1} \).
So

\[ x_m = x_0 + V_m(H_m^{-1}(\beta_0 e_1)) \]

\[ = x_0 + P_m \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} \]

\[ = x_0 + P_{m-1} z_{m-1} + p_m \zeta_m \]

\[ = x_{m-1} + p_m \zeta_m. \]
Lanczos Iteration leads us directly to the Conjugate Gradient method. Three critical properties of Lanczos iteration:

\[
(r_i, r_j) = \delta_{ij} \\
(Ap_i, p_j) = \delta_{ij} \\
r_i = \gamma_m \nu_{m+1}
\]
Lanczos Iteration gives us:

\[ p_m = \frac{1}{\eta_m} (v_m - \beta_m p_{m-1}), \]
\[ x_m = x_{m-1} + p_m \zeta_m, \]
\[ r_m = b - Ax_m \]

We can rescale \( p_i \), and after some manipulation obtain

\[ x_i = x_{i-1} + \alpha_{i-1} p_i, \]
\[ r_i = r_{i-1} - \alpha_{i-1} A p_i, \]
\[ p_i = r_{i-1} + \beta_{i-1} p_{i-1}. \]

If we use orthogonality of \( r_i \) and conjugacy of \( p_i \), we can find \( \alpha_i, \beta_i \).
1: Compute $r_0 = b - Ax_0$, $p_1 = r_0$.
2: for $j = 0, \ldots$ do
3: \hspace{1em} $\alpha_j = (r_j, r_j)/(Ap_{j+1}, p_{j+1})$
4: \hspace{1em} $x_{j+1} = x_j + \alpha_j p_{j+1}$
5: \hspace{1em} $r_{j+1} = r_j - \alpha_j Ap_{j+1}$
6: \hspace{1em} $\beta_j = (r_{j+1}, r_{j+1})/(r_j, r_j)$
7: \hspace{1em} $p_{j+2} = r_{j+1} + \beta_j p_{j+1}$
8: end for

[Saa03]
Biorthogonal methods

- Let's return to the *non-symmetric* case.
- With CG, we get a short recurrence relation which allows us to generate an (orthogonal) basis for the Krylov subspace $K(A, r_0)$.
- With Arnoldi’s method, we started out by constructing an orthogonal basis for $K(A, r_0)$.
- If we give up trying to achieve an orthogonal basis, can we obtain a short recurrence relation instead?
Here we construct \( \{v_j\}, \{w_j\}_{j=1 \ldots m} \) bases of \( \mathcal{K}_m(A, r_0) \) and \( \mathcal{K}_m(A^T, r_0) \) respectively such that \((v_i, w_j) = \delta_{ij}\).
Lanczos Biorthogonalization

1: Compute $r_0 = b - Ax_0$ and $\beta = ||r_0||_2$.
2: Set $v_1 = r_0/\beta$, and choose $w_1$ such that $(v_1, w_1) = 1$.
3: Set $\beta_1 = \delta_1 = 0$, $w_0 = v_0 = 0$.
4: for $j = 1, \ldots, m$ do
   5: $\alpha_j = (Av_j, w_j)$
   6: $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
   7: $\hat{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
   8: $\delta_{j+1} = |(\hat{v}_j, \hat{w}_j)|^{1/2}$. If $\delta_{j+1} = 0$, stop.
   9: $\beta_{j+1} = (\hat{v}_j, \hat{w}_j)/\delta_{j+1}$.
   10: $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$
   11: $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$
5: end for
13: Define $T_m$ as the tridiagonal matrix with $T_{j,j} = \alpha_j$, $T_{j,j-1} = \delta_j$, $T_{j,j+1} = \beta_{j+1}$.
14: Compute $y_m = T_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$. 
14: Compute $y_m = T_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$.

Much like Arnoldi’s method, we get the identity

$$W_m^T A V_m = T_m.$$ 

- We can thus choose $x_m$ just like in Arnoldi’s method.
- We multiply our residual by $W_m^T$ rather than $V_m^T$.
- This gives us $r_m = b - A x_m \perp K_m(A^T, r_0) = \mathcal{L}_m$. 
What about keeping track of $V_m$?
It turns out we can derive the Biconjugate Gradient method from Lanczos’ Biorthogonalization in exactly the same way as CG was derived from Lanczos’ method for symmetric matrices.

- Find an $LU$ decomposition of $T_m$
- Define $P_m = V_m U_m^{-1}$, $P_m^* = W_M L_m^{-T}$.
- Update $p_j, p_j^*, r_j, r_j^*, x_j$ at every step.
BiCG - convergence

Unfortunately, there’s no guarantee that convergence is even monotonic.

[TB97]
BiCG - other challenges

- GMRES can suffer “lucky breakdowns”, where $v_m = 0$.
- BiCG can suffer more serious breakdowns, where $(v_m, w_m) = 0$.
- The work to computing the vectors $w_i$ and $p_i^*$ doesn’t contribute directly to our solution unless we also happen to need to solve the system $A^T x^* = b^*$.


