Krylov Subspace Methods

Harris Enniss

University of California Santa Barbara

henniss@math.ucsb.edu

2014-10-29

Motivation

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.

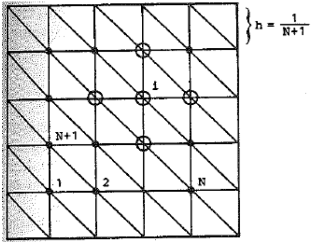


Fig 1.10

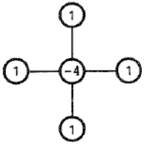


Fig 1.11

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

$$
(1.25)\qquad
\begin{array}{c}
\text{row } (N+1) \\ \\ \\ \\ \\ \\ \\ \\
\end{array}
\begin{bmatrix}
4 & -1 & 0 & -1 & 0 & \cdot & \cdot & 0 \\
-1 & 4 & -1 & 0 & -1 & 0 & \cdot & \cdot \\
0 & -1 & 4 & -1 & 0 & -1 & \cdot & \cdot \\
-1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\
0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & -1 \\
\cdot & 0 & \cdot & \cdot & \cdot & \cdot & -1 & 0 \\
\cdot & \cdot & \cdot & -1 & 0 & -1 & 4 & -1 \\
0 & \cdot & \cdot & 0 & -1 & 0 & -1 & 4
\end{bmatrix}
\begin{bmatrix} \xi_1 \\ \\ \cdot \\ \cdot \\ \cdot \\ \\ \\ \xi_M \end{bmatrix}
=
\begin{bmatrix} b_1 \\ \\ \cdot \\ \cdot \\ \cdot \\ \\ \\ b_M \end{bmatrix}.
$$

[Joh87]

What is a Krylov Subspace Method?

Krylov Subspace methods are a class of iterative methods.
$x_0, \ldots, x_m$ lie in subspaces $x_0 + \mathcal{K}_m(A, v)$.
$\mathcal{K}_m(A, v)$ is a *Krylov subspace*:

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

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.

Preconditioner

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 matricies, 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 matricies.
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 $H_m = \{h_{ij}\}_{i,j=1,\dots,m}$; set $H_m = 0$.
3: **for** $j = 1, \dots, m$ **do**
4: $\quad w_j = Av_j$.
5: $\quad$ **for** $j = 1, \dots, j$ **do**
6: $\quad\quad h_{ij} = (w_j, v_i)$.
7: $\quad\quad w_j = w_j - h_{ij} v_i$.
8: $\quad$ **end for**
9: $\quad h_{j+1,j} = ||w_j||_2$.
10: $\quad$ **if** $h_{j+1,j} = 0$ **then**
11: $\quad\quad m = j$, **go to** 15.
12: $\quad$ **end if**
13: $\quad v_{j+1} = w_j / h_{j+1,j}$.
14: **end for**
15: $y_m = H_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$.

Arnoldi

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 $x_m = x_0 + V_m y_m$.

$\mathcal{L}_m = \mathcal{K}_m(A, r_0)$, i.e. want x_m such that

$$r_m = b - Ax_m \perp \mathcal{L}_m = \mathcal{K}_m.$$

Suppose $x_m = x_0 + V_m y_m$, for some y_m.

$$\begin{aligned} 0 =& V_m^T(b - Ax_m) \\ =& V_m^T(r_0 - AV_m y_m) \\ =& V_m^T(\beta v_1) - V_m^T AV_m y_m \\ =& \beta e_1 - H_m y_m \end{aligned}$$

Arnoldi Breakdowns

```
if $h_{j+1,j} = 0$ then
    $m = j$, go to 15.
end if
```

What we call a "lucky breakdown", signaling early convergence.

Generalized Minimal RESidual (GMRES)

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

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

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 chosing $r_m \perp \mathcal{L} = A\mathcal{K}_m$.

GMRES

1: $r_0 = b - Ax_0$, $\beta = ||r_0||_2$, $v_1 = r_0/\beta$.
2: Define an $m \times m$ matrix $H_m = \{h_{ij}\}_{i,j=1,\ldots,m}$; set $H_m = 0$.
3: **for** $j = 1, \ldots, m$ **do**
4: $\quad w_j = Av_j$.
5: $\quad$ **for** $i = 1, \ldots, j$ **do**
6: $\quad\quad h_{ij} = (w_j, v_i)$.
7: $\quad\quad w_j = w_j - h_{ij}v_i$.
8: $\quad$ **end for**
9: $\quad h_{j+1,j} = ||w_j||_2$.
10: $\quad$ **if** $h_{j+1,j} = 0$ **then**
11: $\quad\quad m = j$, **go to** 15.
12: $\quad$ **end if**
13: $\quad v_{j+1} = w_j/h_{j+1,j}$.
14: **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 $x_m = x_0 + V_m y_m$.

Computational Cost

- 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.

GMRES, final remarks

- 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

Lets go back to Arnoldi's method, and suppose A is *symmetric* and positive definite.
$H_m = V_m^T A V_m$, 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]

Lanczos Iteration

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.

Previously, we computed

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

Now H_m is symmetric and tridiagonal. Write

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

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

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

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

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

Previously, we computed

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

Now H_m is symmetric and tridiagonal. Write

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

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

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

Lanczos Iteration

So

$$\begin{aligned} x_m &= x_0 + V_m(H_m^{-1}(\beta_0 e_1)) \\ &= x_0 + P_m(\begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix}) \\ &= x_0 + P_{m-1} z_{m-1} + p_m \zeta_m \\ &= x_{m-1} + p_m \zeta_m . \end{aligned}$$

Conjugate Gradient (CG)

Lanczos Iteration leads us directly to the Conjugate Gradient method.
Three critical properties of Lanczos iteration:

$$(r_i, r_j) = \delta_{ij}$$
$$(Ap_i, p_j) = \delta_{ij}$$
$$r_i = \gamma_m v_{m+1}$$

CG

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 α_i, β_i.

CG

Compute $r_0 = b - Ax_0$, $p_1 = r_0$.
for $j = 0, \ldots$ **do**
    $\alpha_j = (r_j, r_j)/(Ap_{j+1}, p_{j+1})$
    $x_{j+1} = x_j + \alpha_j p_{j+1}$
    $r_{j+1} = r_j - \alpha_j A p_{+1}$
    $\beta_j = (r_{j+1}, r_{j+1})/(r_j, r_j)$
    $p_{j+2} = r_{j+1} + \beta_j p_{j+1}$
end for

[Saa03]

Biorthogonal methods

- Lets 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?

Lanczos Biorthoganalization

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 Biorthoganalization

Compute $r_0 = b - Ax_0$ and $\beta = ||r_0||_2$.
Set $v_1 = r_0/\beta$, and choose w_1 such that $(v_1, w_w) = 1$.
Set $\beta_1 = \delta_1 = 0$, $w_0 = v_0 = 0$.
for $j = 1, \ldots, m$ **do**
    $\alpha_j = (Av_j, w_j)$
    $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
    $\hat{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
    $\delta_{j+1} = |(\hat{v}_j, \hat{w}_j)|^{1/2}$. If $\delta_{j+1} = 0$, stop.
    $\beta_{j+1} = (\hat{v}_j, \hat{w}_j)/\delta_{j+1}$.
    $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$
    $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$
end for
Define T_m as the tridiagonal matrix with $T_{j,j} = \alpha_j$, $T_{j,j-1} = \delta_j$, $T_{j,j+1} = \beta_{j+1}$.
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 $r_m = b - Ax_m \perp \mathcal{K}_m(A^T, r_0) = \mathcal{L}_m$.

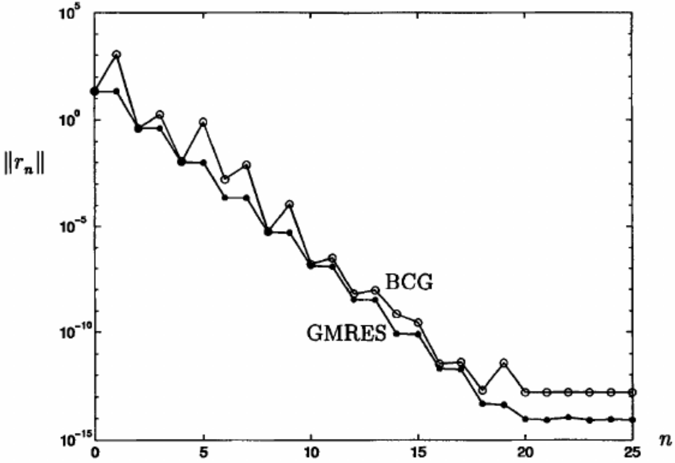
Biconjugate Gradient(BiCG)

What about keeping track of V_m?
It turns out we can derive the Biconjugate Gradient method from Lanczos' Biorthoganalization 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 $A^T x^* = b^*$.

References I

- Martin Gutknecht, *A brief introduction to krylov space methods for solving linear systems*, Frontiers of Computational Science (2006), 53–62.
- Claes Johnson, *Numerical solutions of partial differential equations by the finite element method*, Dover, 1987.
- Yousef Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.
- Yousef Saad and Henk van der Vorst, *Iterative solution of linear systems in the 20th century*, Journal of Computational and Applied Mathematics (2000), 1–33.
- Lloyd Trefethen and David Bau, *Numerical linear algebra*, SIAM, 1997.
- Henk van der Vorst, *Krylov subspace iteration*, Computing in Science and Engineering (2000), 32–37.