Introduction to Krylov Subspace Methods

Matthias Bollhöfer (TU Braunschweig)

INRIA Sophia Antipolis
08. September 2010

Outline

Model Problems

1D elliptic boundary value problem

$$\begin{aligned} -u''(x) &= f(x),\ x \in [0,1] \\ u(0) &= g_0,\ u(1) = g_1 \end{aligned}$$

2D elliptic boundary value problem

$$\begin{aligned} \Omega &= [0,1] \times [0,1] \\ \overbrace{-u_{xx}(x,y) - u_{yy}(x,y)}^{-\Delta u(x,y)} &= f(x,y),\ (x,y)\text{ß}n\Omega \\ u(x,y) &= g(x,y),\ (x,y) \in \partial\Omega \end{aligned}$$

3D elliptic boundary value problem

$$-\Delta u = f \text{ in } [0,1]^3 + \text{b.c.}$$

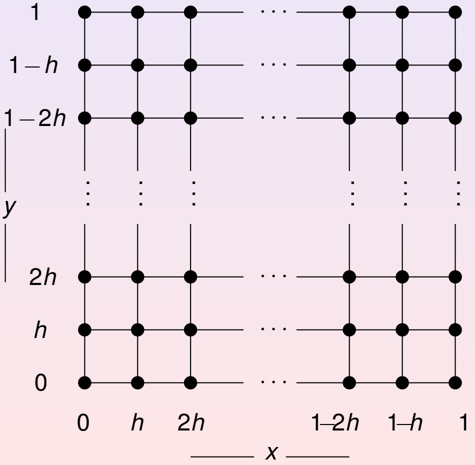
Discretization using Finite Differences

1D problem: $h = \frac{1}{N+1}$. $[0,1] \to \Omega_h = \{0, h, 2h, \dots, 1-h, 1\}$

2D problem:

$\Omega = [0,1]^2 \to \Omega_h =$

$\{(k,l)h : \ k,l = 0,\dots N+1\}$



Discretization using Finite Differences

1D boundary value problem

Differential equation

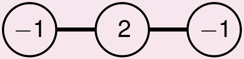
$$-y''(x) = f(x),\ y(0) = g_0, y(1) = g_1$$

$$\downarrow$$

Difference equation

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i, u_0 = g_0, u_{N+1} = g_1,\ \text{where } u_i \equiv u(ih)\ \forall i$$

3-point stencil



Linear System

$$T_h = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \longrightarrow T_h u = f.$$

Discretization using Finite Differences

2D boundary value problem

Differential equation

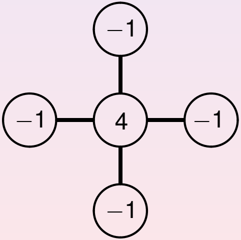
$$-\Delta y(x,y) = f(x,y), \text{ in } \Omega = [0,1]^2, \; y = g \text{ on } \partial\Omega$$

$$\downarrow$$

Difference equation

$$\frac{-u_{i-1,j} - u_{i,j-1} + 4u_{ij} - u_{i+1,j} - u_{i,j+1}}{h^2} = f_{ij}$$

5-point-difference stencil



Linear system

$$A_h u = f$$

Direct Solvers — Introduction

$$A = \begin{pmatrix} * & * & * & \cdots & * \\ * & * & & & \\ * & & * & & \\ \vdots & & & \ddots & \\ * & & & & * \end{pmatrix}$$

Gaussian elimination (*LU* decomposition) without pivoting yields

$$A = \begin{pmatrix} * & & & & 0 \\ * & * & & & \\ * & * & * & & \\ \vdots & \vdots & \ddots & \ddots & \\ * & * & * & \cdots & * \end{pmatrix} \begin{pmatrix} * & * & * & \cdots & * \\ & * & * & \cdots & * \\ & & * & \ddots & * \\ & & & \ddots & \vdots \\ 0 & & & & * \end{pmatrix}$$

Factors are dense! Giant memory consumption and computation time!

Direct Solvers — Introduction

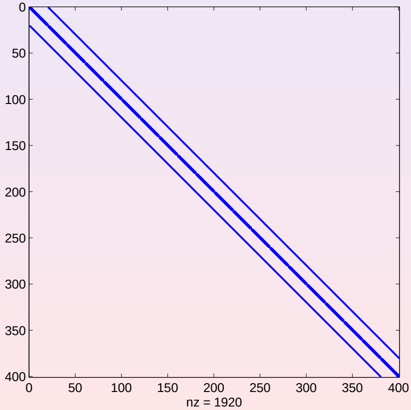
Here. problem solvable by reordering

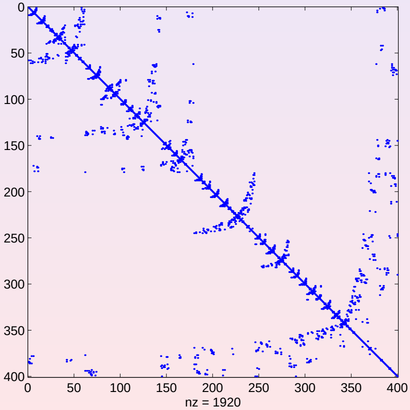
$$A \longrightarrow \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & \ddots & & \vdots \\ & & & * & * \\ * & * & \cdots & * & * \end{pmatrix}, \quad A = \begin{pmatrix} * & & & & 0 \\ & * & & & \\ & & * & & \\ & & & \ddots & \\ * & * & * & \cdots & * \end{pmatrix} \begin{pmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & \ddots & \vdots \\ 0 & & & & * \end{pmatrix}$$

Unfortunately: Reordering does not always help!

Direct Solvers

1D/2D/3D BVP: matrix is sparse, symmetric positive definite
Adapted direct solver $A = LL^\top$, Cholesky decomposition
Reorder matrix entries in advance (“(approximate) minimum degree”)

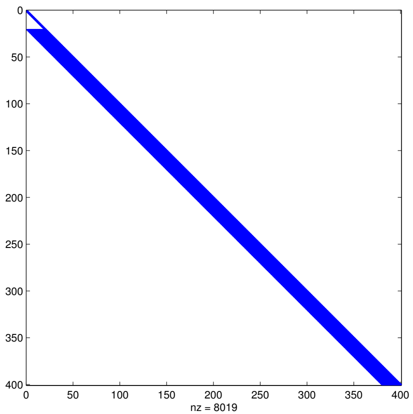


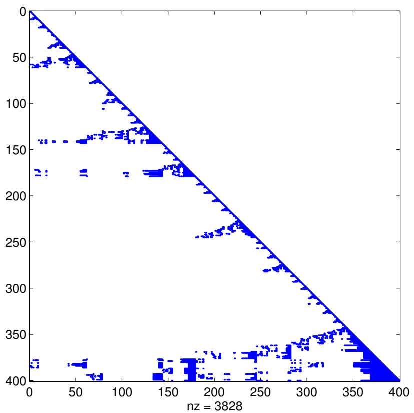


Direct Solver

2D boundary value problem

triangular factors L from Cholesky decomposition





Direct Solver (MATLAB)

2D boundary value problem

h	problem size	comp. time[sec]	$nnz(L+L^T)/nnz(A)$
$\frac{1}{31}$	961	0.01	4.4
$\frac{1}{63}$	3969	0.01	6.2
$\frac{1}{127}$	$1.6 \cdot 10^4$	0.12	8.3
$\frac{1}{255}$	$6.5 \cdot 10^4$	0.45	11.1
$\frac{1}{511}$	$2.6 \cdot 10^5$	2.4	14.4
$\frac{1}{1023}$	$1.0 \cdot 10^6$	11.5	18.8
$\frac{1}{2047}$	$4.2 \cdot 10^6$	54.0	23.2

- computation time slightly worse than linear
- memory requirement also a little bit worse than linear
- $h = \frac{1}{2047}$ extremely small for 2D!

Direct Solver (MATLAB)

3D boundary value problem

h	problem size	comp. time[sec]	$nnz(L+L^T)/nnz(A)$
$\frac{1}{31}$	$3.0 \cdot 10^4$	1.5	67.4
$\frac{1}{63}$	$2.5 \cdot 10^5$	38.5	212.0
$\frac{1}{127}$	$2.0 \cdot 10^6$	crash, integer overflow	

- computation time drastically grows
- memory requirement is even worse
- $h = \frac{1}{63}$ is “normal” size for 3D!

Iterative Solvers — Introduction

Idea:

$$Ax = b$$

1. $x_0, r_0 = b - Ax_0$, build $v_1 = r_0$, choose $x_1 \in x_0 + \mathrm{span}\{r_0\} = x_0 + \mathrm{span}\{v_1\}$,
2. build $v_2 = Av_1$, choose $x_2 \in x_0 + \mathrm{span}\{r_0, Ar_0\} = x_0 + \mathrm{span}\{v_1, v_2\}$,
3. build $v_3 = Av_2$, choose $x_3 \in x_0 + \mathrm{span}\{r_0, Ar_0, A^2 r_0\} = x_0 + \mathrm{span}\{v_1, v_2, v_3\}$,
4. ...
5. build $v_k = Av_{k-1}$, choose $x_k \in x_0 + \mathrm{span}\{r_0, \dots, A^{k-1} r_0\} = x_0 + \mathrm{span}\{v_1, \dots, v_k\}$,

Questions

- How well can we approximate x by elements x_k from $x_0 + \mathrm{span}\{r_0, \dots, A^{k-1} r_0\}$?
- How can we efficiently compute a suitable approximation?

Technische Universität Braunschweig

Polynomial Approximation

$$K_k(A, r_0) = \mathrm{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\} \text{ (Krylov subspace)}$$

$$x_k \in x_0 + K_k(A, r_0)$$

$$\Rightarrow r_k = b - Ax_k = b - A\left(x_0 + \sum_{l=0}^{k-1} \alpha_l A^l r_0\right) = (I + \sum_{l=1}^{k} \alpha_l A^l) r_0$$

$$I + \sum_{l=1}^{k} \alpha_l A^l = p_k(A)$$

is a polynomial of degree k w.r.t. A such that $p_k(0) = 1$.

Suppose that x_k is the “best” solution such that $\|r_k\|$ is minimal.

$$\Rightarrow \|r_k\| = \|p_k(A)r_0\| = \min_{q_k(0)=1} \|q_k(A)r_0\|$$

Polynomial Approximation

Normal Matrix Case

Suppose for simplicity that A is normal, i.e., we have a unitary matrix U such that

$$A = U \Lambda U^H, \text{ where } \Lambda = \mathrm{diag}_i(\lambda_i)$$

$$\begin{aligned}
\Rightarrow \|r_k\|_2 &\leqslant \min_{q_k(0)=1} \|q_k(A)\|_2 \cdot \|r_0\|_2 \\
&= \min_{q_k(0)=1} \|U q_k(\Lambda) U^H\|_2 \cdot \|r_0\|_2 \\
&= \min_{q_k(0)=1} \|q_k(\Lambda)\|_2 \cdot \|r_0\|_2 \\
&= \min_{q_k(0)=1} \max_i |q_k(\lambda_i)| \cdot \|r_0\|_2
\end{aligned}$$

$$\Rightarrow \frac{\|r_k\|_2}{\|r_0\|_2} \leqslant \min_{q_k(0)=1} \max_i |q_k(\lambda_i)|.$$

In particular, if $A = A^H$, then A is normal and this bound applies.

Polynomial Approximation

Positive Definite Case

Suppose that $A = A^H$ is positive definite. We have a unitary matrix U such that

$$A = U\Lambda U^H, \text{ where } \Lambda \text{ is diagonal}, \lambda_i > 0$$

$\|x\|_A = \sqrt{x^H A x} = \|A^{1/2}x\|_2$ is a norm

$$\|C\|_A = \sup_{x\neq 0} \frac{\|Cx\|_A}{\|x\|_A} = \sup_{x\neq 0} \frac{\|A^{1/2}Cx\|_2}{\|A^{1/2}x\|_2} = \sup_{y\neq 0} \frac{\|A^{1/2}CA^{-1/2}y\|_2}{\|y\|_2}$$

$$\Rightarrow \|C\|_A = \|A^{1/2}CA^{-1/2}\|_2.$$

Note that

$$(x - x_k)^H A(x - x_k) = (x - x_k)^H AA^{-1}A(x - x_k) = (b - Ax_k)^H A^{-1}(b - Ax_k)$$

$$\Rightarrow \|x - x_k\|_A = \|r_k\|_{A^{-1}},$$

Suppose that x_k is the "best" solution such that the error $\|x - x_k\|_A$ is minimal.

$$\Rightarrow \|x - x_k\|_A = \|r_k\|_{A^{-1}} = \|p_k(A)r_0\|_{A^{-1}} = \min_{q_k(0)=1} \|q_k(A)r_0\|_{A^{-1}}$$

It follows that

$$
\begin{aligned}
\Rightarrow \|x - x_k\|_A = \|r_k\|_{A^{-1}} &= \min_{q_k(0)=1} \|q_k(A) r_0\|_{A^{-1}} \\
&= \min_{q_k(0)=1} \|A^{-1/2} q_k(A) r_0\|_2 \\
&= \min_{q_k(0)=1} \|q_k(A) A^{-1/2} r_0\|_2 \\
&\leqslant \min_{q_k(0)=1} \|q_k(A)\|_2 \cdot \|A^{-1/2} r_0\|_2 \\
&= \min_{q_k(0)=1} \|U q_k(\Lambda) U^H\|_2 \cdot \|r_0\|_{A^{-1}} \\
&= \min_{q_k(0)=1} \max_i |q_k(\lambda_i)| \cdot \|x - x_0\|_A
\end{aligned}
$$

$$
\frac{\|x - x_k\|_A}{\|x - x_0\|_A} \leqslant \min_{q_k(0)=1} \max_i |q_k(\lambda_i)|.
$$

Polynomial approximation

$$x_k \in x_0 + K_k(A, r)$$

Summary so far:

- $A = U\Lambda U^H$ normal, $\|b - Ax_k\|_2$ minimal

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leqslant \min_{q_k(0)=1} \max_i |q_k(\lambda_i)|.$$

- $A = U\Lambda U^H$ s.p.d., $\|x - x_k\|_A$ minimal

$$\frac{\|x - x_k\|_A}{\|x - x_0\|_A} \leqslant \min_{q_k(0)=1} \max_i |q_k(\lambda_i)|.$$

- $A = V\Lambda V^{-1}$ simple, $\|b - Ax_k\|_2$ minimal

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leqslant \|V\|_2 \|V^{-1}\|_2 \cdot \min_{q_k(0)=1} \max_i |q_k(\lambda_i)|.$$

- General case, $\|b - Ax_k\|_2$ minimal. Not more than

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leqslant \min_{q_k(0)=1} \|q_k(A)\|_2.$$

Optimal Polynomials

If the eigenvalues are contained in a line segment $[\alpha, \beta]$ in $\mathbb{C}$ that does not include the origin, then (transformed and normalized) Chebyshev polynomials are optimal.

$$c_k(x) = \cos(k \arccos x)$$

$$\text{Choose } q_k(x) = \frac{c_k(Lx)}{c_k(L0)}, \text{ where } L : [\alpha, \beta] \to [-1, 1].$$

- $A = U\Lambda U^H$ normal, $\lambda_1, \ldots, \lambda_n \in [\alpha, \beta]$ not including 0, $\kappa = \max |\alpha|, |\beta| / \min |\alpha|, |\beta|$

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leqslant 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k .$$

- $A = U\Lambda U^H$ s.p.d., $\kappa = \max_i \lambda_i / \min_i \lambda_i$

$$\frac{\|x - x_k\|_A}{\|x - x_0\|_A} \leqslant 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k .$$

Optimal Polynomials

- A general

  $$\mathcal{F}(A)=\{x^HAx:\|x\|_2=1\} \text{ (field of values)}$$

  eigenvalues are always inside $\mathcal{F}(A)$

  $\mathcal{F}(A)$ is convex, but generally not the convex hull of the eigenvalues.

  Suppose that $\mathcal{F}(A)\subset D(c,\rho)$ not including the origin

  $$\frac{\|r_k\|_2}{\|r_0\|_2}\leqslant 2\left(\frac{\rho}{|c|}\right)^k.$$

- Further approximation results based on ellipses, Faber polynomials, ϵ-pseudospectra.

Optimal Polynomials

Even in the case of Chebyshev polynomials the bounds are not optimal, isolated eigenvalues or clusters of isolated eigenvalues can be treated separately leading to an improved bound.

Example. A s.p.d. $\lambda_{n-p+1} = \cdots = \lambda_n$

$$\begin{aligned}\frac{\|x-x_k\|_A}{\|x-x_0\|_A} &\leqslant \min_{q_k(0)=1} \max_{i=1,\dots,n} |q_k(\lambda_i)| \leqslant \max_{i=1,\dots,n} \frac{|c_{k-p}(L\lambda_i)|}{|c_{k-p}(L0)|} \cdot \frac{|\lambda_n-\lambda_i|^p}{|\lambda_n|^p} \\ &\leqslant \max_{i=1,\dots,n-p} \frac{|c_{k-p}(L\lambda_i)|}{|c_{k-p}(L0)|} \cdot 1,\end{aligned}$$

where $c_{k-p}(Lx)$ is the optimal Chebyshev polynomial with respect to $L : [\lambda_1, \lambda_{n-p}] \to [-1, 1]$.

$$\Rightarrow \frac{\|x-x_k\|_A}{\|x-x_0\|_A} \leqslant 2\left(\frac{\sqrt{\hat\kappa}-1}{\sqrt{\hat\kappa}+1}\right)^{k-p}, \text{ where } \hat\kappa = \frac{\lambda_{n-p}}{\lambda_1}.$$

Suppose we wish to build $r_0, Ar_0, A^2r_0, \ldots$ to form $\text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$. Then we preferably would like to have an orthonormal basis of this space.

Method by Gram-Schmidt:

0. $h_{10} = \|r_0\|_2$, $v_1 = r_0/h_{10}$
1. build $q = Av_1$ and orthogonalize q against v_1

$$q := q - v_1 h_{11}, \text{ where } h_{11} = v_1^H q.$$

   $h_{21} = \|q\|_2$, $v_2 = q/h_{21}$

2. build $q = Av_2$ and orthogonalize q against v_1, v_2

$$q := q - v_1 h_{12} - v_2 h_{22}, \text{ where } h_{12} = v_1^H q, h_{22} = v_2^H q.$$

   $h_{32} = \|q\|_2$, $v_3 = q/h_{32}$

3. ...

k. build $q = Av_{k-1}$ and orthogonalize q against $v_1, \dots, v_{k-1}$

$$q := q - v_1 h_{1,k-1} - \cdots v_{k-1} h_{k-1,k-1}, \text{ where } h_{1,k-1} = v_1^H q, \dots, h_{k-1,k-1} = v_{k-1}^H q.$$

$$h_{k,k-1} = \|q\|_2, \; v_k = q/h_{k,k-1}$$

- Gram-Schmidt's method successively forms an orthonormal basis $v_1, \dots, v_k$ of $\mathrm{span}\{r_0, Ar_0, \dots, A^{k-1} r_0\}$.
- Numerically it is more advisable to overwrite q after each update step by its projected update (modified Gram-Schmidt)

Arnoldi's Method

Arnoldi's method = modified Gram-Schmidt applied to $\{r_0, Ar_0, A^2 r_0, \dots\}$

r_0 given, $h_{10} = \|r_0\|_2$, $v_1 = r_0/h_{10}$
for $k = 1, 2, 3, \dots$
    $q = Av_k$
    for $l = 1, \dots, k$
        $h_{lk} = v_l^H q$
        $q := q - v_l h_{lk}$
    end
    $h_{k+1,k} = \|q\|_2$
    $v_{k+1} = q/h_{k+1,k}$
end

Arnoldi's Method

A different look at Arnoldi's method. Recall that

$$Av_k - v_1 h_{1k} - \cdots v_k h_{kk} = q_{final} = v_{k+1} h_{k+1,k}.$$

$$\Rightarrow Av_k = [v_1, \ldots, v_k, v_{k+1}] \begin{bmatrix} h_{1k} \\ \vdots \\ h_{kk} \\ h_{k+1,k} \end{bmatrix}.$$

Successively applied, column by column this yields

$$A \underbrace{[v_1, \ldots, v_k]}_{V_k} = \underbrace{[v_1, \ldots, v_k, v_{k+1}]}_{V_{k+1}} \left[\begin{array}{cccc} h_{11} & \cdots & \cdots & h_{1k} \\ h_{21} & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ & & h_{k,k-1} & h_{kk} \\ \hline 0 & & & h_{k+1,k} \end{array}\right]$$

$$\Leftrightarrow AV_k = V_{k+1} \left[\begin{array}{cccc} & H_k & & \\ \hline 0 & \cdots & 0 & h_{k+1,k} \end{array}\right] \Leftrightarrow AV_k = V_{k+1} \underline{H_k}$$

Arnoldi's Method

- This method is known as Arnoldi's method when computing eigenvalue and invariant subspace information from H_k to obtain approximate eigenvalues (Ritz values) and approximate invariant subspaces for A.

$$H_k x = \mu x \Rightarrow A\,(V_k x) = \mu(V_k x) \;+\; v_{k+1} h_{k+1,k} x_k$$

- H_k, $\underline{H}_k$ are called upper Hessenberg matrices

- Arnoldi's method can be used to compute the minimum residual solution. This leads to the GMRES method.

GMRES Method

Suppose that $r_0 = b - Ax_0$ is given for Arnoldi's method.
Then $x_k \in x_0 + \mathrm{span}\{r_0, \dots, A^{k-1} r_0\}$ is equivalent to

$$x_k = x_0 + V_k d \text{ for some suitable } d$$

$$\Rightarrow r_k = b - Ax_k = b - Ax_0 - AV_k d = r_0 - V_{k+1} \underline{H_k} d = V_{k+1} \left(h_{10} e_1 - \underline{H_k} d \right)$$

$$\Rightarrow \|r_k\|_2 = \|h_{10} e_1 - \underline{H_k} d\|_2 .$$

- we can minimize $\|r_k\|_2$ by minimizing $\|h_{10} e_1 - \underline{H_k} d\|_2$
- The latter problem is easily solved by a QR decomposition since $\underline{H_k}$ is almost upper triangular
- When the QR decomposition is performed, $h_{10} e_1$ is updated and its $(k+1)$-st component (in modulus) refers to $\|r_k\|_2$

GMRES Method

x_0 $r_0 = b - Ax_0$ given, $h_{10} = \|r_0\|_2$, $v_1 = r_0/h_{10}$, $z = h_{10}e_1$.
for $k = 1, 2, 3, \dots$
  $q = Av_k$
  for $l = 1, \dots, k$
    $h_{lk} = v_l^H q$
    $q := q - v_l h_{lk}$
  end
  $h_{k+1,k} = \|q\|_2$
  $v_{k+1} = q/h_{k+1,k}$
  proceed with the QR decomposition of $\|z - \underline{H_k} d\|_2$:
  1) Apply plane rotations from previous steps to the most recent column k of H_k
  2) Compute a plane rotation to eliminate $h_{k+1,k}$ from the pair $(h_{kk}, h_{k+1,k})^T$
  3) apply this rotation also to $(z_k, z_{k+1})^T$
end
Compute d from $\|z - \underline{H_k} d\|_2$, where $\underline{H_k}$ is upper triangular
$x_{final} = x_0 + V_k d$

GMRES Method

- Typically GMRES is only run for a limited number of steps
- $v_1, \ldots, v_k$ need to be stored
- After, say m steps GMRES is stopped, x_{final} is computed and GMRES is restarted with $x_0 = x_{final}$ (restarted GMRES(m))

In theory we could update x_k simultaneously during Arnoldi's method

Let $\underline{H}_k = Q_k R_k$ be the QR decomposition,

$$\Rightarrow x_k = x_0 + V_k \underbrace{\underline{H}_k^+ h_{10} e_1}_{d} = x_0 + (V_k R_k^{-1}) \underbrace{(Q_k^H h_{10} e_1)}_{z}$$

If we compute the last column $p_k = V_k R_k^{-1} e_k$ successively, then

$$x_k = x_0 + V_k R_k^{-1} z = x_{k-1} + p_k z_k.$$

Hermitian Case — MINRES

$$AV_k = V_{k+1}\underline{H}_k \Rightarrow V_k^H AV_k = V_k^H V_{k+1}\underline{H}_k = H_k$$

$$H_k = V_k^H AV_k,\ A = A^H \Rightarrow H_k = H_k^H$$

- If H_k is Hermitian, then H_k is already tridiagonal (Arnoldi is then called Lanczos)
- If $\underline{H}_k = Q_k R_k$, then the upper triangular part of R_k only consists of three bands in total

$$R_k = \begin{bmatrix} * & * & * & \\ & \ddots & \ddots & \ddots \\ & & * & * \\ 0 & & & * \end{bmatrix}$$

- The Arnoldi method only selectively has to apply reorthogonalization
- MINRES does not require more than v_k, v_{k-1}, v_{k-2}.
- x_k can be updated using the additional search direction p_k .
- $p_k = V_k R_k^{-1} e_k$ requires a back substitution with R_k, only v_{k-2}, v_{k-1}, v_k are needed

$$x_k = x_0 + V_k R_k^{-1} z = x_{k-1} + p_k z_k.$$

Hermitian Case — MINRES

x_0, $r_0 = b - Ax_0$ given, $h_{10} = \|r_0\|_2$, $v_1 = r_0/h_{10}$, $z = h_{10}e_1$.
for $k = 1, 2, 3, \dots$
$q = Av_k$
$q := q - v_{k-1}\overline{h}_{k,k-1}$
$h_{kk} = v_k^H q$
$q := q - v_k h_{kk}$
$h_{k+1,k} = \|q\|_2$
$v_{k+1} = q/h_{k+1,k}$
proceed with the QR decomposition of $\|z - \underline{H_k} d\|_2$:
1) Apply plane rotations from step $k-1, k-2$ to the most recent column k of H_k
2) Compute a plane rotation to eliminate $h_{k+1,k}$ from the pair $(h_{kk}, h_{k+1,k})^T$
3) apply this rotation also to $(z_k, z_{k+1})^T$
4) Compute p_k
5) update x_k from x_{k-1}
end

Positive Definite Case — CG

$$A = A^H \text{ positive definite } \Rightarrow H_k = H_k^T \text{ positive definite.}$$

- Instead of computing the minimal solution from $\|h_{10}e_1 - \underline{H_k}d\|_2$, we directly compute d from
$$h_{10}e_1 = H_k d$$
using Cholesky decomposition $L_k D_k L_k^H$ of H_k.
- Formally
$$x_k = x_0 + V_k \underbrace{H_k^{-1} h_{10} e_1}_{d} = x_0 + (V_k L_k^{-H})\,(D_k^{-1} L_k^{-1} h_{10} e_1)$$
- The entries of $L_k^{-1} h_{10} e_1$ can be computed successively by forward substitution
- The columns $p_k = V_k L_k^{-H} e_k$ can be successively computed using back substitution. This only requires v_k, v_{k-1}, since L_k is bidiagonal.
- If x_{k-1} is already computed, then x_k is obtained from x_{k-1} via $x_k = x_{k-1} + \alpha_k p_k$, where α_k is the last row of $D_k^{-1} L_k^{-1} h_{10} e_1$

Positive Definite Case — CG

- the search directions p_k are A-orthogonal, since $L_k^{-1} V_k^H A V_k L_k^{-H} = L_k^{-1} H_k L_k^{-H} = D_k$.
- Solving $h_{10} e_1 = H_k d$ can be shown to minimize $\|x - x_k\|_A$.
- CG is usually stated differently based on minimizing a quadratic functional $\Phi(x) = \frac{1}{2} x^H A x - b^H x$.

x_0 initial guess, $r_0 = b - Ax_0$, $p_1 = r_0$, $\rho_0 = r_0^* r_0$
for $k = 1, 2, 3, \ldots$
$z = Ap_k$
$\alpha_k = \rho_{k-1}/(p_k^* z)$
$x_k = x_{k-1} + \alpha_k p_k$
$r_k = r_{k-1} - \alpha_k z$
$\rho_k = r_k^* r_k$
$\beta_k = \rho_k / \rho_{k-1}$
$p_{k+1} = r_k + \beta_k p_k$
end

Two-Sided Iterative Methods — Introduction

- Iterative methods so far have dealt with $\mathrm{span}\{r_0, Ar_0, \ldots, A^{k-1} r_0\}$
- In the general unsymmetric case, GMRES requires to store the whole basis $v_1, \ldots, v_k$ of this Krylov subspace
- Only in the Hermitian case the this sequence simplifies (MINRES, CG)
- Alternatively we can use a left Krylov subspace

$$\mathrm{span}\{s_0, A^H s_0, \ldots, (A^H)^{k-1} s_0\}.$$

- The use of two subspace will allow to use short recurrences also in the unsymmetric case
- Unfortunately unstable
- Basis for these methods is the two-sided Lanczos method

Two-Sided Lanczos

- Main idea, successively compute two sequences $V_k = [v_1, v_2, \ldots, v_k]$ and $W_k = [w_1, w_2, \ldots, w_k]$ such that

$$W_k^H V_k = I, \; AV_k \approx V_k T_k, \; A^H W_k \approx W_k T_k^H,$$

  where T_k is tridiagonal.
- The computation of these sequences is interlaced

0. r_0, s_0 are given, set $t_{10} = \|r_0\|_2$, $v_1 := r_0/t_{10}$, $t_{01} = v_1^H s_0$, $w_1 := s_0/t_{01}$.
1. $q := Av_1$, bi-orthogonalize q against v_1, i.e.,

$$q := q - v_1 t_{11}, \text{ where } t_{11} = w_1^H q$$

   After that we have $w_1^H q = 0$.
   $z := A^H w_1$, bi-orthogonalize z against w_1, i.e.,

$$z := z - w_1 \bar{t}_{11}$$

   After that we have $z^H v_1 = 0$.
   $t_{21} = \|q\|_2$, $v_2 := q/t_{21}$, $t_{12} = v_2^H z$, $w_2 := z/t_{12}$.

Two-Sided Lanczos

2. $q := Av_2$, bi-orthogonalize q against v_1, v_2, i.e.,

$$q := q - v_1 t_{12} - v_2 t_{22}, \text{ where } t_{22} = w_2^H q$$

After that we have $w_1^H q = w_2^H q = 0$.
$z := A^H w_2$, bi-orthogonalize z against w_1, w_2, i.e.,

$$z := z - w_1 \bar{t}_{21} - w_2 \bar{t}_{22}$$

After that we have $z^H v_1 = z^H v_2 = 0$.
$t_{32} = \|q\|_2$, $v_3 := q/t_{32}$, $t_{23} = v_3^H z$, $w_3 := z/t_{23}$.

k. At step k: compute v_{k+1}, w_{k+1} and $t_{k+1,k}, t_{k,k+1}$ from

$$Av_k = [v_{k-1}, v_k, v_{k+1}] \begin{bmatrix} t_{k-1,k} \\ t_{k,k} \\ t_{k+1,k} \end{bmatrix}, \; A^H w_k = [w_{k-1}, w_k, w_{k+1}] \begin{bmatrix} \bar{t}_{k,k-1} \\ \bar{t}_{k,k} \\ \bar{t}_{k,k+1} \end{bmatrix},$$

where $w_i^H v_j = \delta_{ij}$.

Iterative Methods Based on the Two-Sided Lanczos

- Choose $x_k \in x_0 + \mathrm{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}$, i.e., $x_k = x_0 + V_k d$ and set $r_0 = b - Ax_0$.
- $r_k = b - Ax_k = b - Ax_0 - AV_k d = r_0 - V_{k+1}\underline{T}_k = V_{k+1}\left(t_{10}e_1 - \underline{T}_k d\right)$

Iterative methods derived from this choice:

(a) BiCG: Define d by solving

$$T_k d = t_{10}e_1 \Rightarrow x_k = x_0 + V_k T_k^{-1} W_k^H r_0.$$

For the solution of the tridiagonal system, LU decomposition is used.

(b) QMR: Define d by solving

$$\|t_{10}e_1 - \underline{T}_k d\|_2 = \min \Rightarrow x_k = x_0 + V_k \underline{T}_k^{+} W_{k+1}^H r_0.$$

For the solution of the least squares system, QR decomposition is used.

(c) CGS: variant of BiCG that does not require A^H

(d) TFQMR: variant of QMR that does not require A^H

(e) BiCGstab: variant of BiCG/CGS that does not require A^H and performs local minimization

Iterative Methods Based on the Two-Sided Lanczos

- Although unstable, methods like QMR and BiCGstab work relatively well in practice
- All two-sided version allow for short recurrences
- Two sided versions can be used to set up structure-preserving iterative solvers

Example. Suppose that $A^H J = JA$ for some non-singular matrix J.

$$\Rightarrow (A^H)^k Jv = JA^k v \Rightarrow \mathrm{span}\{Jv, A^H Jv, \ldots, (A^H)^k Jv\} = J\mathrm{span}\{v, Av, \ldots, A^k v\}.$$

If we start with the some initial guess r_0 for the right Krylov subspace $\mathrm{span}\{r_0, Ar_0, \ldots, A^k r_0\}$ and choose $s_0 = Jr_0$, then the left Krylov space $\mathrm{span}\{s_0, A^H s_0, \ldots, (A^H)^k s_0\}$ is computed from the right one by multiplying with J.

$$\Rightarrow AV_k = V_{k+1}\underline{T}_k,\ A^H JV_k = JV_{k+1}\begin{bmatrix} T_k^H \\ (0 \cdots \bar{t}_{k,k+1}) \end{bmatrix},$$

where

$$V_k^H JV_k = I.$$

J-symmetry saves half of the work, since only the right Krylov subspace is needed

Technische Universität Braunschweig

Summary

- Iterative methods presented here are

  either based on Arnoldi's method and $\mathrm{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$

  or on the two-sided Lanczos method and $\mathrm{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$, $\mathrm{span}\{s_0, A^H r_0, \ldots, (A^H)^{k-1} r_0\}$.

- Numerical methods

  either minimize $\|e_1\|r_0\|_2 - \underline{H}_k d\|_2$, resp. $\|e_1\|r_0\|_2 - \underline{T}_k d\|_2$,

  or solve $e_1\|r_0\|_2 = H_k d$, resp. $e_1\|r_0\|_2 = T_k d$.

- Approximation partially explained by polynomials and eigenvalue distribution

- Nothing is said about how the eigenvalue distribution could be improved

  $\longrightarrow$ Preconditioning (next time)