An Introduction to Iterative Solvers and Preconditioning Techniques

Alison Ramage
Department of Mathematics
University of Strathclyde

ar@maths.strath.ac.uk
http://www.maths.strath.ac.uk/~caas63/

Overview

- revision and motivation
- stationary iterative methods
- nonstationary (Krylov subspace) methods for
  - symmetric positive definite systems
  - symmetric indefinite systems
  - nonsymmetric systems
- preconditioning
- multigrid
- further information

Revision of Linear Algebra Ideas (1)

- real square $N \times N$ matrix A
- A is symmetric if A=A^T.
- The trace of A is the sum of its diagonal entries.
- If $A\mathbf{v} = \lambda\mathbf{v}$ for $\lambda \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^N$, then λ is called the eigenvalue of A with corresponding eigenvector $\mathbf{v}$.
- A is called
  - positive definite if all of its eigenvalues are positive;
  - negative definite if all of its eigenvalues are negative;
  - indefinite if it has positive and negative eigenvalues.

Revision of Linear Algebra Ideas (2)

- Common vector norms:

$$\|\mathbf{v}\|_\infty = \max_i |v_i| \qquad \text{infinity norm}$$

$$\|\mathbf{v}\|_1 = \sum_{i=1}^{N} |v_i| \qquad \text{1-norm}$$

$$\|\mathbf{v}\|_2 = \sqrt{v_1^2 + v_2^2 + ... + v_n^2} = \sqrt{\mathbf{v}^T\mathbf{v}} \qquad \text{2-norm}$$

- Common matrix norms:

$$\|A\|_\infty = \max_i \sum_{j=1}^{N} |a_{ij}| \qquad \text{max abs row sum}$$

$$\|A\|_1 = \max_j \sum_{i=1}^{N} |a_{ij}| \qquad \text{max abs column sum}$$

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)} \qquad \text{spectral norm}$$

$$\|A\|_F = \sqrt{tr(AA^T)} \qquad \text{Frobenius norm}$$

Revision of Linear Algebra Ideas (3)

- Vectors $\mathbf{v}, \mathbf{w}$ in a vector space V over $\mathbb{R}$ satisfy

$$\mathbf{v}, \mathbf{w} \in V \Rightarrow \mathbf{v} + \mathbf{w} \in V, \qquad \mathbf{v} \in V, \alpha \in \mathbb{R} \Rightarrow \alpha \mathbf{v} \in V$$

- Vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m \in V$ are linearly independent if

$$\sum_{j=1}^{m} a_j \mathbf{v}_j = a_1 \mathbf{v}_1 + \ldots + a_m \mathbf{v}_m = 0 \Leftrightarrow a_1, a_2, \ldots, a_m = 0$$

- V is of dimension d if there exist d LI basis vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_d$ such that any vector in V can be written as a linear combination of the basis vectors

$$\mathbf{v} = \sum_{j=1}^{d} b_j \mathbf{y}_j$$

Motivation: PDE-based Problems

- large-scale simulations in e.g. fluid/solid mechanics, structural engineering, elasticity, medical applications, meteorology . . .
- linear algebra costs often dominate
- finite element, finite difference, finite volume discretisations
- only local connections between unknowns on the computational grid
- very large, very sparse linear systems

Model Finite Element BVP

- Laplace's equation
- d-dimensional uniform grid, discretisation parameter h
- $n = \frac{1}{h}$ nodes in each dimension
- coefficient matrix A is $N \times N$ where $N = O(n^d) = O(h^{-d})$
- e.g. bilinear finite elements
  - $d = 2$: 9 nonzeros per row
  - $d = 3$: 27 nonzeros per row

Solving Linear Systems

PROBLEM: solve $A\mathbf{u} = \mathbf{f}$ where A is very large and sparse

- Direct methods: e.g. Gaussian elimination
  - Factorise A as L (lower triangular matrix) and U (upper triangular matrix).
  - Solve for $\mathbf{u}$ via back-substitution.
  - Direct computation of exact solution $\hat{\mathbf{u}}$.

- Iterative methods:
  - Choose an initial guess $\mathbf{u}_0$.
  - Generate a sequence of iterates $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \ldots$
  - Stop when converged in some sense to $\hat{\mathbf{u}}$.
  - Usual to iterate until $\|\mathbf{u}_k - \mathbf{u}_{k-1}\| \leq \epsilon$ for some given tolerance ϵ.

Direct or Iterative Solvers?

- direct methods
  - efficient for full matrices
  - good for lots of RHS vectors
  - sparse matrices may lead to fill-in
  - node ordering often important
  - storage and CPU restrictions

- iterative methods
  - data structures predetermined
  - no need for special node ordering
  - efficient for extremely large sparse problems
  - last iterate can give a good starting vector
  - some expertise needed
  - lack of robustness

Asymptotic Work Estimates

Iterative method: Conjugate Gradients

Direct method: Gaussian Elimination with band-minimising node ordering

Computational Work

	$d=2$	$d=3$
CG	$O(N^{\frac{3}{2}})$	$O(N^{\frac{4}{3}})$
GE factorise	$O(N^2)$	$O(N^{\frac{7}{3}})$
GE solve	$O(N^{\frac{3}{2}})$	$O(N^{\frac{5}{3}})$

Storage

	$d=2$	$d=3$
CG	$O(N)$	$O(N)$
GE factorise	$O(N^{\frac{3}{2}})$	$O(N^{\frac{5}{3}})$
GE solve	$O(N^{\frac{3}{2}})$	$O(N^{\frac{5}{3}})$

Stationary Iterative Methods

- matrix splitting $A = M - N$ (M invertible)

$$A\mathbf{u} = \mathbf{f} \Rightarrow (M - N)\mathbf{u} = \mathbf{f} \Rightarrow \quad M\mathbf{u} = N\mathbf{u} + \mathbf{f}$$

- generate iterates via $M\mathbf{u}_k = N\mathbf{u}_{k-1} + \mathbf{f} \Rightarrow$

$$\mathbf{u}_k = M^{-1}N\mathbf{u}_{k-1} + M^{-1}\mathbf{f}$$

iteration matrix $R = M^{-1}N$

- splittings combine D, L, and U where $A = D + L + U$
- stationary methods: M and N do not depend on k

Common Matrix Splittings

$$A = M - N, \qquad A = D + L + U$$

Richardson: $A = I - (I - A)$

Jacobi: $A = D - [-(L + U)]$

Gauss-Seidel: $A = (D + L) - (-U)$

SOR:
$$\begin{aligned} A &= M_\omega - N_\omega \\ &= \frac{1}{\omega}(D + \omega L) - \frac{1}{\omega}[(1 - \omega)D - \omega U] \end{aligned}$$

SSOR: $A = \frac{\omega}{2 - \omega}(M_\omega D^{-1} M_\omega^T - N_\omega D^{-1} N_\omega^T)$

Nonstationary Methods: SPD Systems

Solve $A\mathbf{u} = \mathbf{f}$ where

- A is symmetric and positive definite
- A has s distinct (positive) eigenvalues

Minimal polynomial:

$$A^s + m_1 A^{s-1} + \cdots + m_{s-1} A + m_s I = 0$$

so

$$A^{-1} = -\frac{1}{m_s} A^{s-1} - \frac{m_1}{m_s} A^{s-2} - \cdots - \frac{m_{s-1}}{m_s} I$$

$$\hat{\mathbf{u}} = A^{-1}\mathbf{f} \in \mathcal{K}(A, \mathbf{f}, s) \equiv \text{span}\{\mathbf{f}, A\mathbf{f}, A^2\mathbf{f}, \ldots, A^{s-1}\mathbf{f}\}$$

Krylov Subspace

Three equivalent problems

1. solve $A\mathbf{u} = \mathbf{f}$

2. minimise $\Phi(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T A\mathbf{u} - \mathbf{u}^T\mathbf{f}$

$$\nabla\Phi(\mathbf{u}) = A\mathbf{u} - \mathbf{f} = \mathbf{0}$$

3. minimise $\|\mathbf{u} - \hat{\mathbf{u}}\|_A$

$$\|\mathbf{v}\|_A = \left\{\mathbf{v}^T A\mathbf{v}\right\}^{\frac{1}{2}}$$

$$(\mathbf{u} - \hat{\mathbf{u}})^T A(\mathbf{u} - \hat{\mathbf{u}}) = \mathbf{f}^T A^{-1}\mathbf{f} + 2\Phi(\mathbf{u})$$

Steepest Descent Method

$$\Phi(\mathbf{u}) = \frac{1}{2}\mathbf{u}^T A\mathbf{u} - \mathbf{u}^T\mathbf{f}$$

At a point $\mathbf{u}_k$, Φ decreases most rapidly in direction

$$-\nabla\Phi(\mathbf{u}_k) = \mathbf{f} - A\mathbf{u}_k = \mathbf{r}_k$$

Value of Φ at point $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k\mathbf{r}_k$ minimised when

$$\alpha_k = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_k^T A\mathbf{r}_k}$$

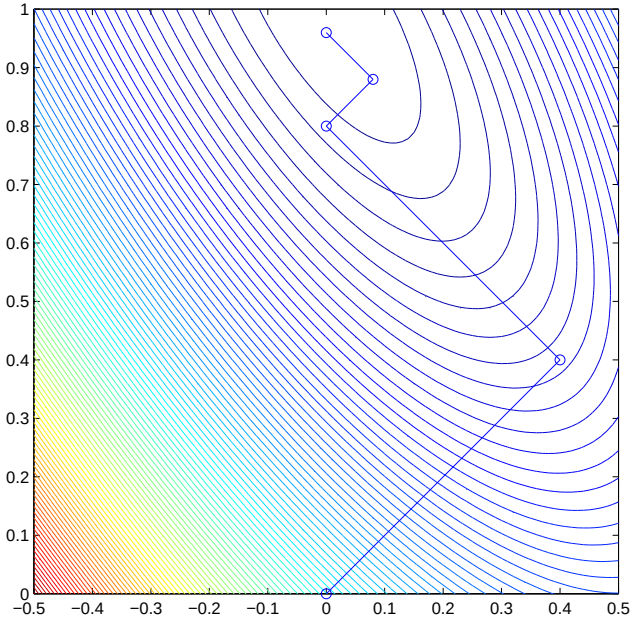
and

$$\Phi(\mathbf{u}_{k+1}) < \Phi(\mathbf{u}_k)$$

is guaranteed

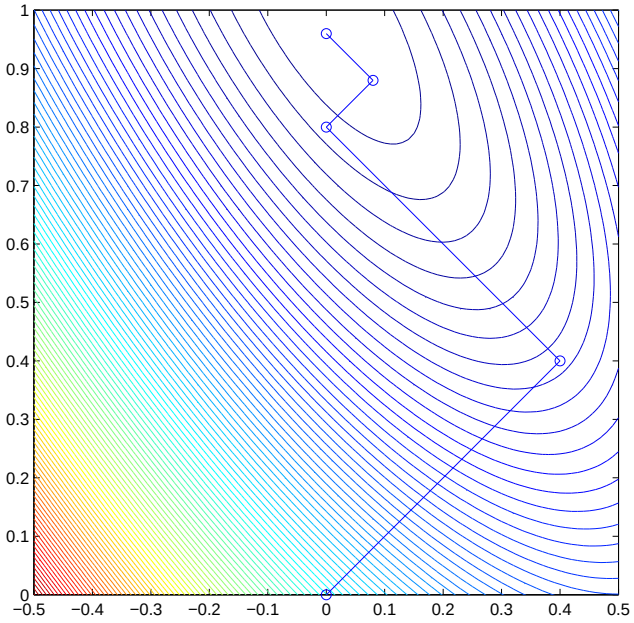
A Practical SD Example

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

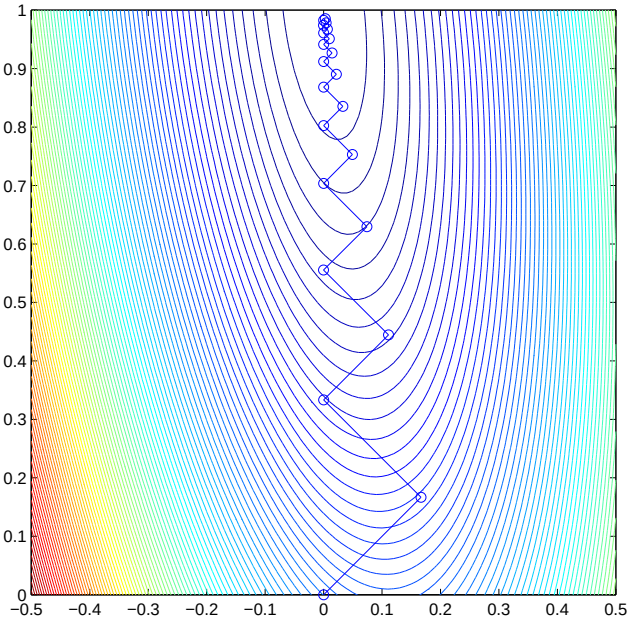


A Practical SD Example

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$



$$A = \begin{bmatrix} 9 & 1 \\ 1 & 1 \end{bmatrix}$$



Conjugate Gradient Method

Choose new search directions $\{\mathbf{p}_0, \mathbf{p}_1, \ldots\}$ with iterates

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha \mathbf{p}_k$$

Writing $P_{k+1} \equiv \mathtt{span}\{\mathbf{p}_0, \ldots, \mathbf{p}_k\}$, we require

(i) $$\mathbf{u}_{k+1} = \min_{\mathbf{u} \in P_{k+1}} \Phi(\mathbf{u})$$

(ii) $$\min_{\alpha} \Phi(\mathbf{u}_k + \alpha \mathbf{p}_k)$$

QUESTION: can we choose $\mathbf{p}_k$ so that $\mathbf{u}_{k+1}$ satisfies (i) and (ii) simultaneously?

Conjugate Gradient Method

Choose new search directions $\{\mathbf{p}_0, \mathbf{p}_1, \ldots\}$ with iterates

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha \mathbf{p}_k$$

Writing $P_{k+1} \equiv \mathtt{span}\{\mathbf{p}_0, \ldots, \mathbf{p}_k\}$, we require

(i) $$\mathbf{u}_{k+1} = \min_{\mathbf{u} \in P_{k+1}} \Phi(\mathbf{u})$$

(ii) $$\min_{\alpha} \Phi(\mathbf{u}_k + \alpha \mathbf{p}_k)$$

QUESTION: can we choose $\mathbf{p}_k$ so that $\mathbf{u}_{k+1}$ satisfies (i) and (ii) simultaneously?

ANSWER: yes, choose the vectors $\mathbf{p}_k$ to be A-conjugate, i.e.

$$\mathbf{p}_j^T A \mathbf{p}_k = 0, \quad j < k$$

Conjugate Gradient Method

Hestenes & Stiefel (1952)

choose $\mathbf{u}_0$
compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{u}_0$
set $\mathbf{p}_0 = \mathbf{r}_0$
for $k = 0$ until convergence do

$$\alpha_k = \mathbf{r}_k^T \mathbf{r}_k / \mathbf{p}_k^T A\mathbf{p}_k$$
$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{p}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$$
$$\beta_k = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} / \mathbf{r}_k^T \mathbf{r}_k$$
$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$

end do

can be implemented with one MVM per iteration

Finite Termination

CG method constructs iterates

$$\mathbf{u}_k \in \mathbf{u}_0 + \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\}$$

with properties

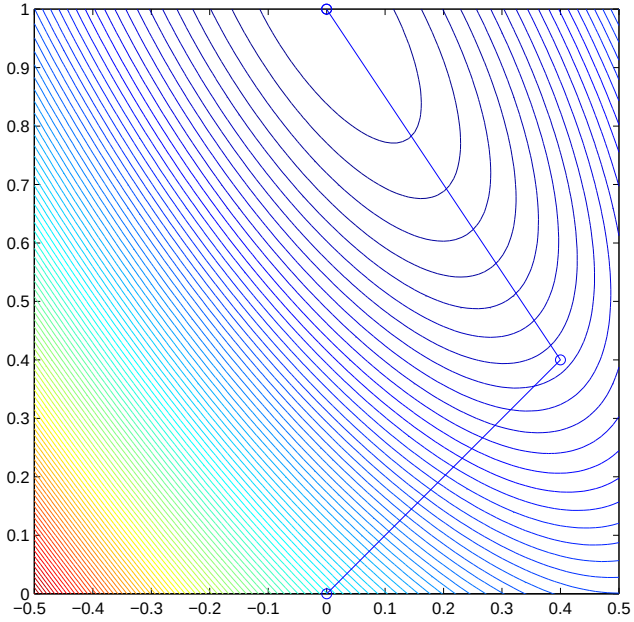
- $\mathbf{u}_k$ minimises $\|\mathbf{u}_k - \hat{\mathbf{u}}\|_A$
- uses a three-term recurrence relation

Theorem: The CG method finds $\hat{\mathbf{u}}$ in s steps.

- In exact arithmetic, CG is a direct method!

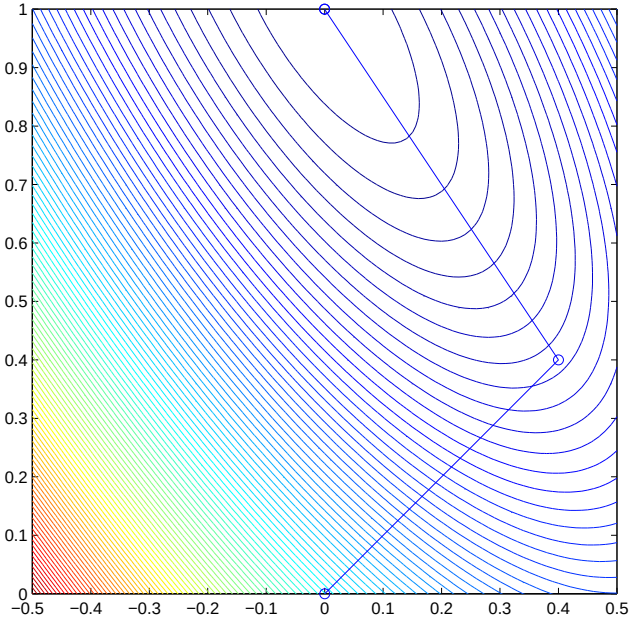
A Practical CG Example

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

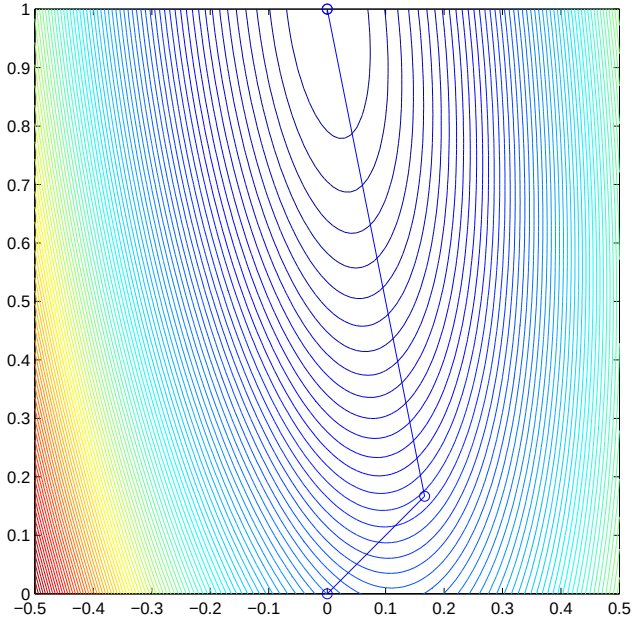


A Practical CG Example

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$



$$A = \begin{bmatrix} 9 & 1 \\ 1 & 1 \end{bmatrix}$$



CG Convergence

$$\mathbf{u}_k \in \mathbf{u}_0 + \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\}$$

$$\mathbf{r}_k = \mathbf{f} - A\mathbf{u}_k = \mathbf{f} - A\left(\mathbf{u}_0 + \sum_{i=1}^{k} \gamma_i A^{i-1}\mathbf{r}_0\right) = \mathbf{r}_0 - \sum_{i=1}^{k} \gamma_i A^{i}\mathbf{r}_0$$

i.e.

$$\mathbf{r}_k = \hat{P}_k(A)\mathbf{r}_0$$

$\hat{P}_k \in \Pi_k^1 \equiv$ polynomials of degree k with constant term 1

$$\|\mathbf{u}_k - \hat{\mathbf{u}}\|_A = \|\mathbf{r}_k\|_{A^{-1}} = \min_{P_k \in \Pi_k^1} \|P_k(A)\mathbf{r}_0\|_{A^{-1}}$$

Expand $\mathbf{r}_0$ in terms of orthonormal eigenvectors:

$$\mathbf{r}_0 = \sum_{i=1}^{n} \rho_i \mathbf{v}_i, \qquad \rho_i = \mathbf{v}_i^T \mathbf{r}_0, \qquad A\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

$$\begin{aligned}
\|\mathbf{u}_k - \hat{\mathbf{u}}\|_A &= \min_{P_k \in \Pi_k^1} \|P_k(A) \sum_{i=1}^{n} \rho_i \mathbf{v}_i\|_{A^{-1}} \\
&= \left\{ \min_{P_k \in \Pi_k^1} \sum_{i=1}^{n} P_k(\lambda_i)^2 (\rho_i \mathbf{v}_i)^T A^{-1} (\rho_i \mathbf{v}_i) \right\}^{\frac{1}{2}} \\
&\leq \min_{P_k \in \Pi_k^1} \max_i |P_k(\lambda_i)| \left\{ \mathbf{r}_0^T A^{-1} \mathbf{r}_0 \right\}^{\frac{1}{2}} \\
&= \min_{P_k \in \Pi_k^1} \max_i |P_k(\lambda_i)| \|\mathbf{r}_0\|_{A^{-1}}
\end{aligned}$$

$$\|\mathbf{u}_k - \hat{\mathbf{u}}\|_A \leq \min_{P_k \in \Pi_k^1} \max_i |P_k(\lambda_i)| \|\mathbf{u}_0 - \hat{\mathbf{u}}\|_A$$

Ideal Bound

$P_{\min}$ is such that

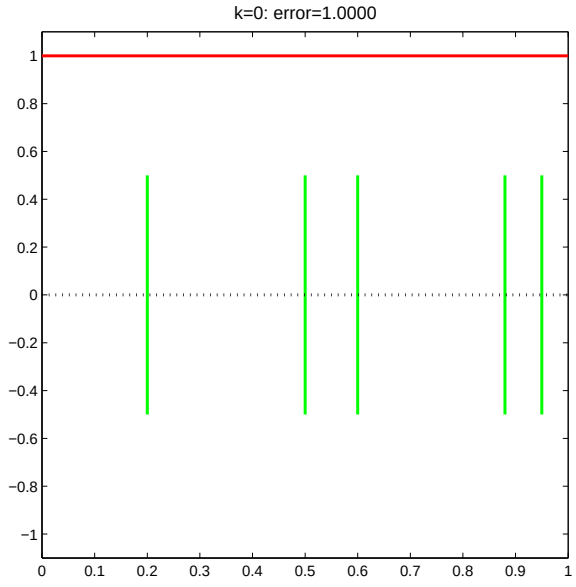
$$M = \max_i |P_{\min}(\lambda_i)| = \min_{P_k \in \Pi_k^1} \max_i |P_k(\lambda_i)|$$

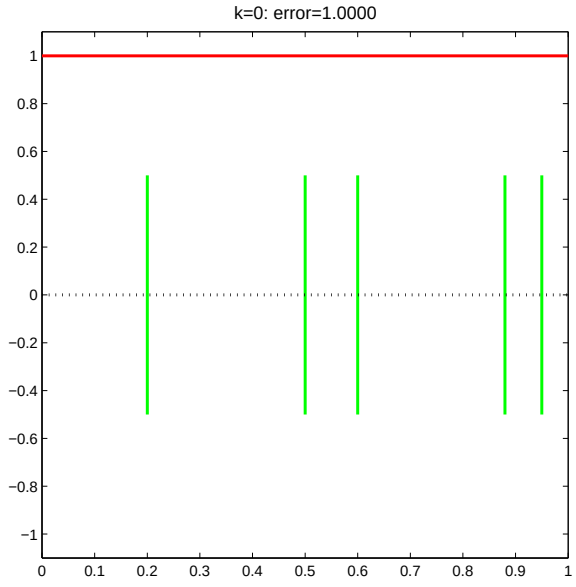
MINIMAX APPROXIMATION

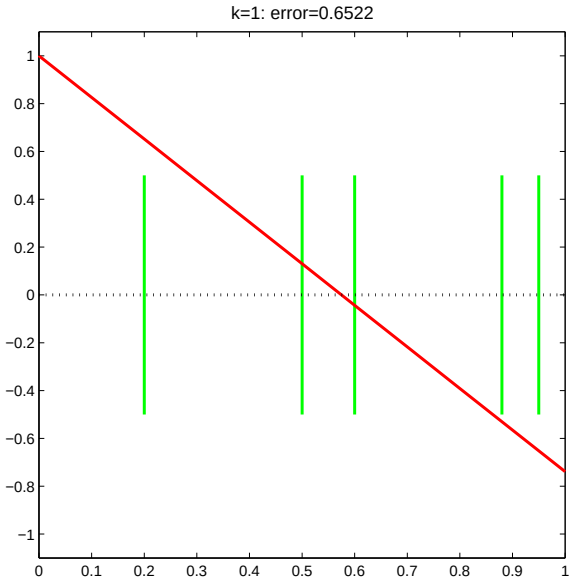
Theorem: Greenbaum (1979)
This error bound is sharp, i.e. there is always some $\mathbf{u}_0$ such that the discrete minimax bound

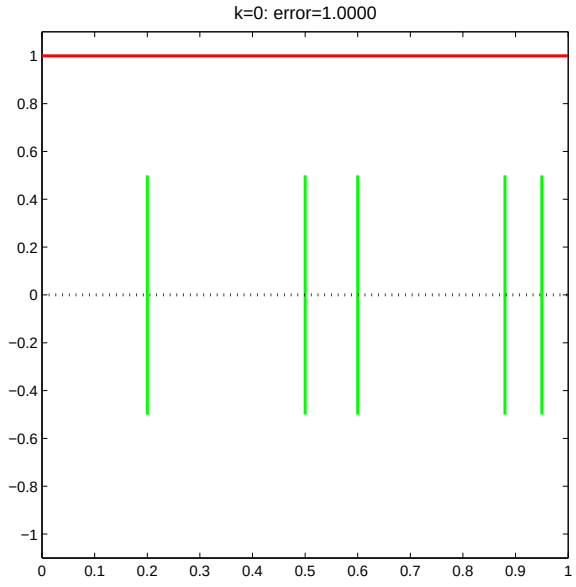
$$\|\mathbf{u}_k - \hat{\mathbf{u}}\|_A \leq M \|\mathbf{u}_0 - \hat{\mathbf{u}}\|_A$$

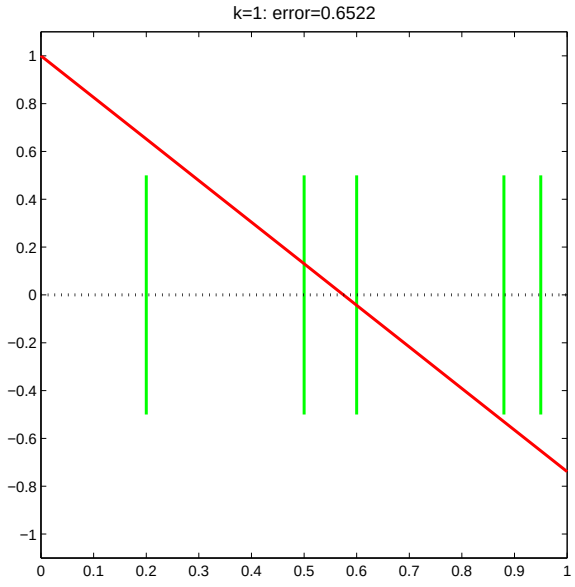
is attained.

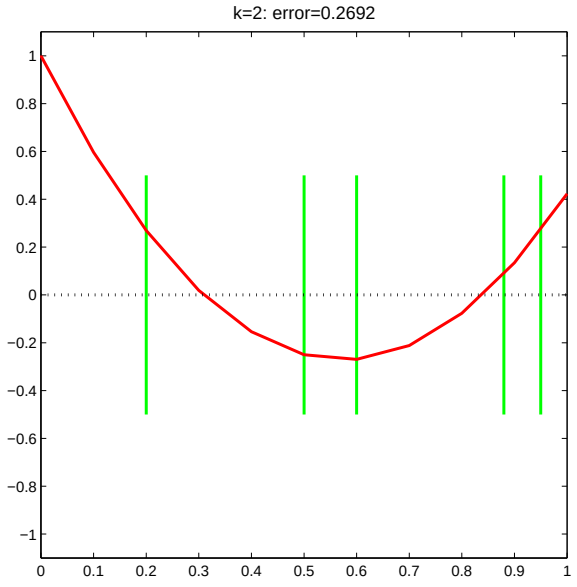

k=0: error=1.0000
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

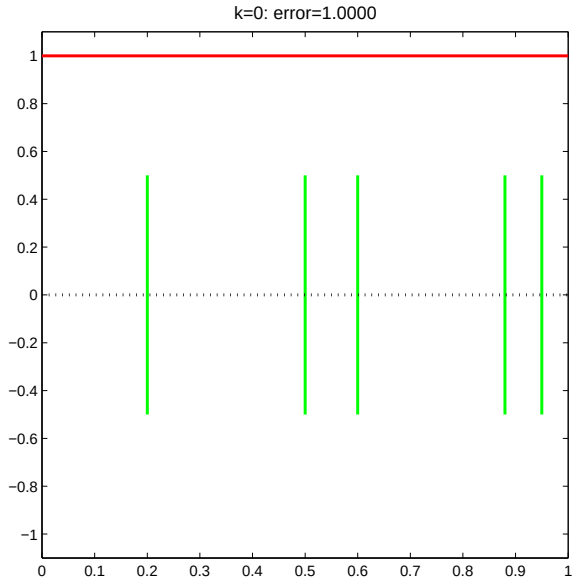

k=0: error=1.0000
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

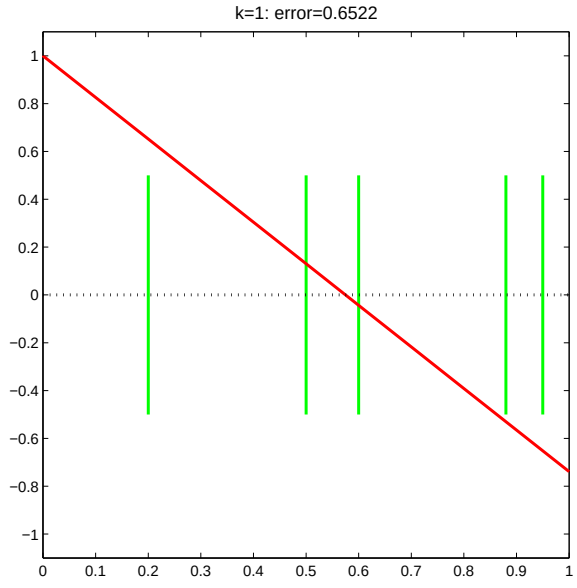

k=1: error=0.6522
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

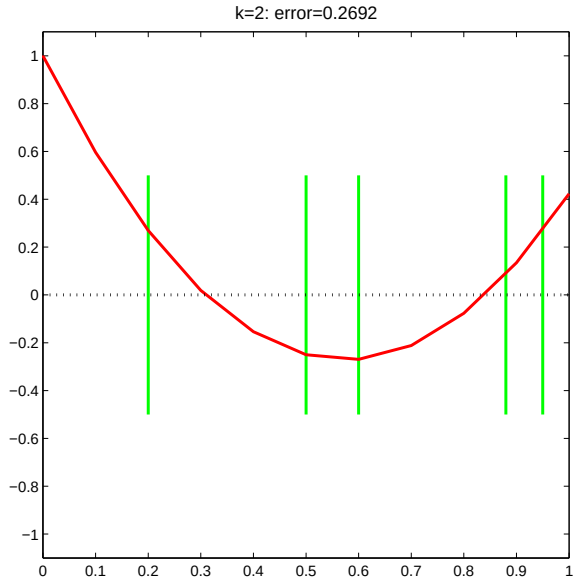

k=0: error=1.0000
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

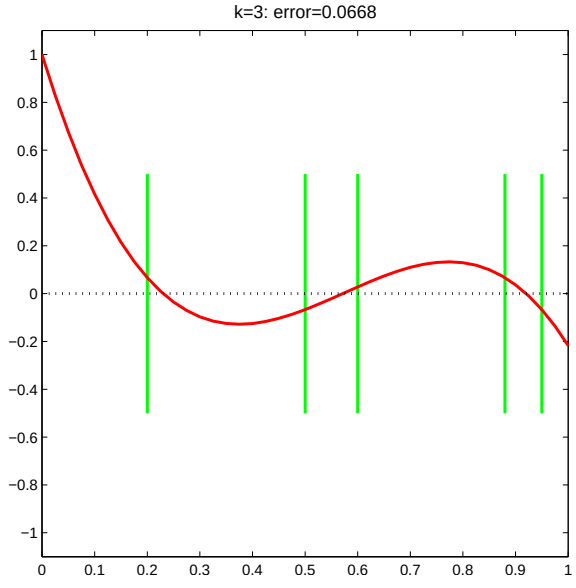

k=1: error=0.6522
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

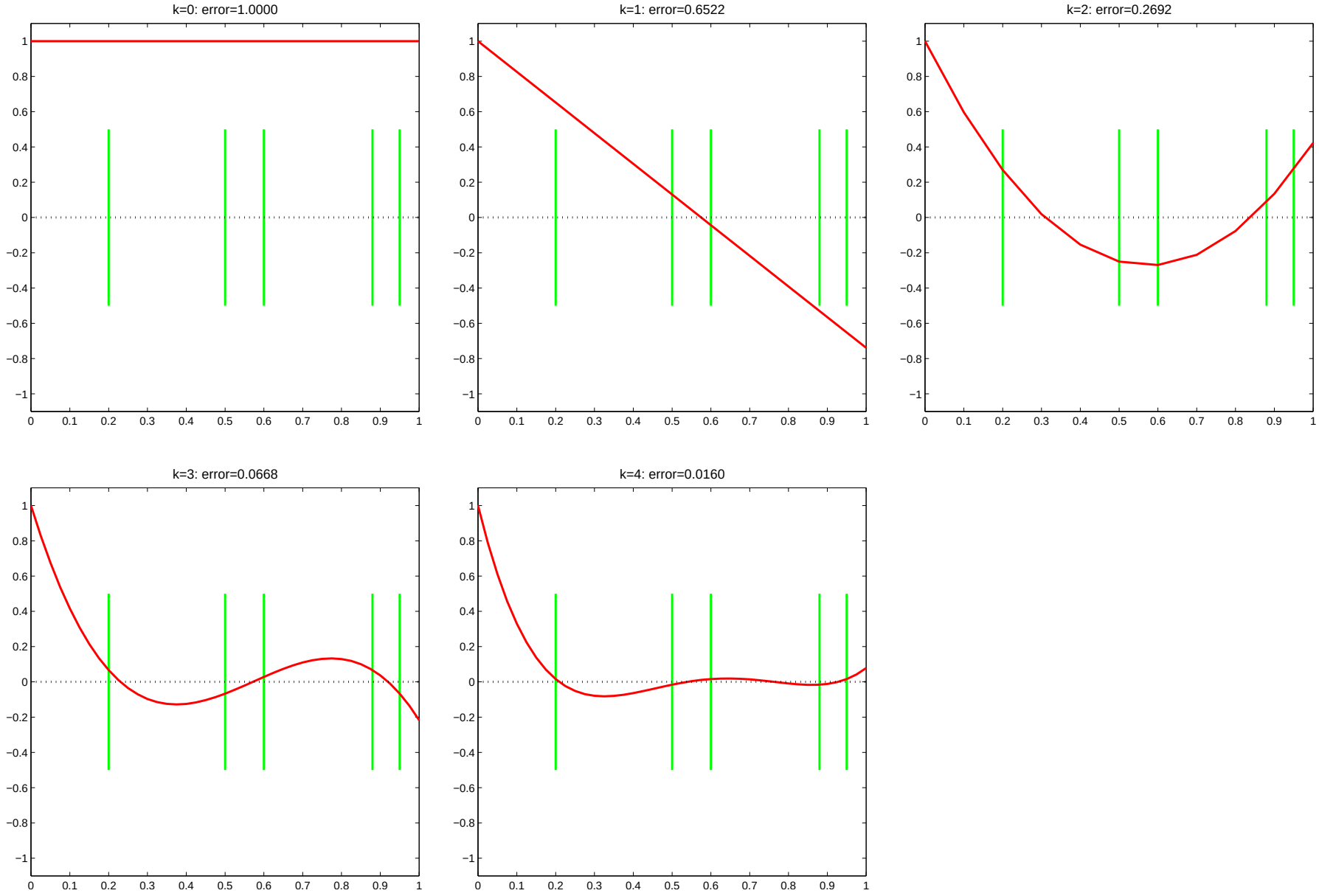

k=2: error=0.2692
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

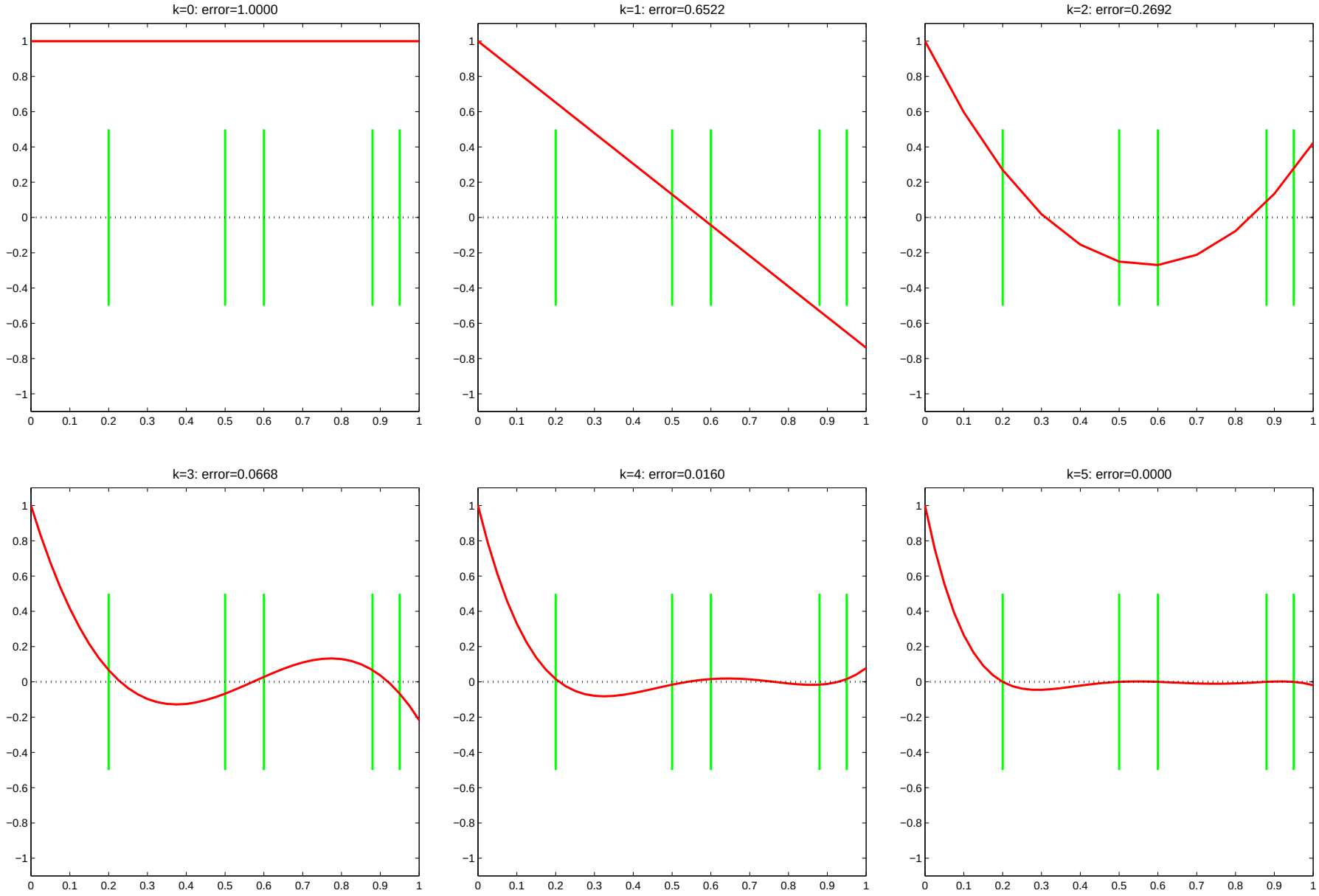

k=0: error=1.0000


k=1: error=0.6522


k=2: error=0.2692


k=3: error=0.0668


k=0: error=1.0000
k=1: error=0.6522
k=2: error=0.2692
k=3: error=0.0668
k=4: error=0.0160


k=0: error=1.0000
k=1: error=0.6522
k=2: error=0.2692
k=3: error=0.0668
k=4: error=0.0160
k=5: error=0.0000

Practical Bound

Based on knowledge of $\lambda_{\max}$ and $\lambda_{\min}$ alone, bound involves

$$\hat{T}_k(\lambda) = \frac{T_k\left[\frac{\lambda_{\max}+\lambda_{\min}-2\lambda}{\lambda_{\max}-\lambda_{\min}}\right]}{T_k\left[\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right]}, \qquad \text{condition number} \quad \kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$$

$$M = \max_i |\hat{T}_k(\lambda_i)| = \frac{1}{T_k\left[\frac{\lambda_{\max}+\lambda_{\min}}{\lambda_{\max}-\lambda_{\min}}\right]} = \frac{1}{T_k\left[\frac{\kappa+1}{\kappa-1}\right]}$$

TCHEBYSHEV APPROXIMATION

Number of iterations required for convergence is

$$k \simeq \frac{1}{2}\ln\frac{2}{\epsilon}\sqrt{\kappa}$$

PDE examples

3D uniform grid with n nodes per dimension

$$r, s, t = 1, \ldots, n$$

7 point Finite Difference Stencil

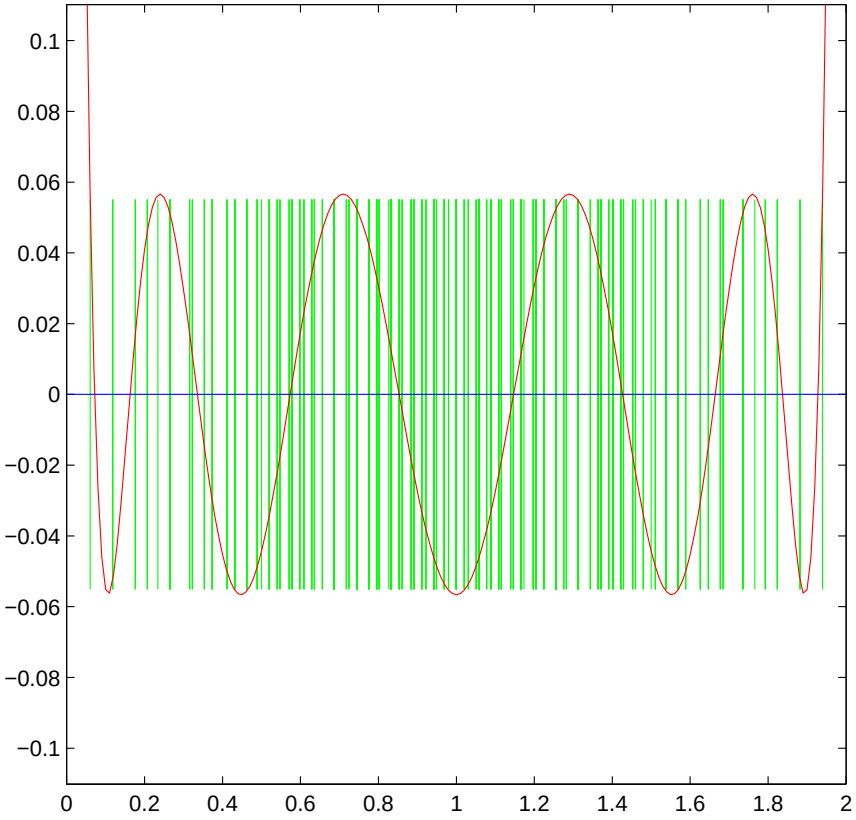
$$\lambda^{rst} = 1 - \frac{1}{3}\cos\frac{r\pi}{n+1} - \frac{1}{3}\cos\frac{s\pi}{n+1} - \frac{1}{3}\cos\frac{t\pi}{n+1}$$

27 point Finite Element Stencil

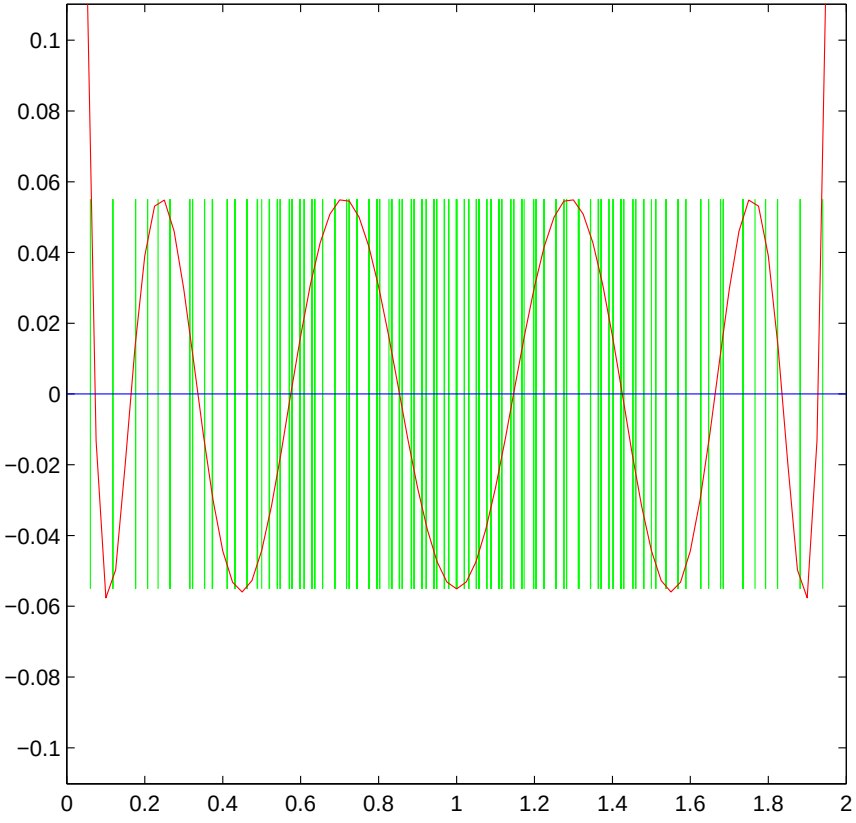
$$\lambda^{rst} = 1 - \frac{1}{4}\cos\frac{r\pi}{n+1}\cos\frac{s\pi}{n+1} - \frac{1}{4}\cos\frac{r\pi}{n+1}\cos\frac{t\pi}{n+1}$$
$$-\frac{1}{4}\cos\frac{s\pi}{n+1}\cos\frac{t\pi}{n+1} - \frac{1}{4}\cos\frac{r\pi}{n+1}\cos\frac{s\pi}{n+1}\cos\frac{t\pi}{n+1}$$

7 point Finite Difference Stencil

Tchebyshev error: 0.5662e-1

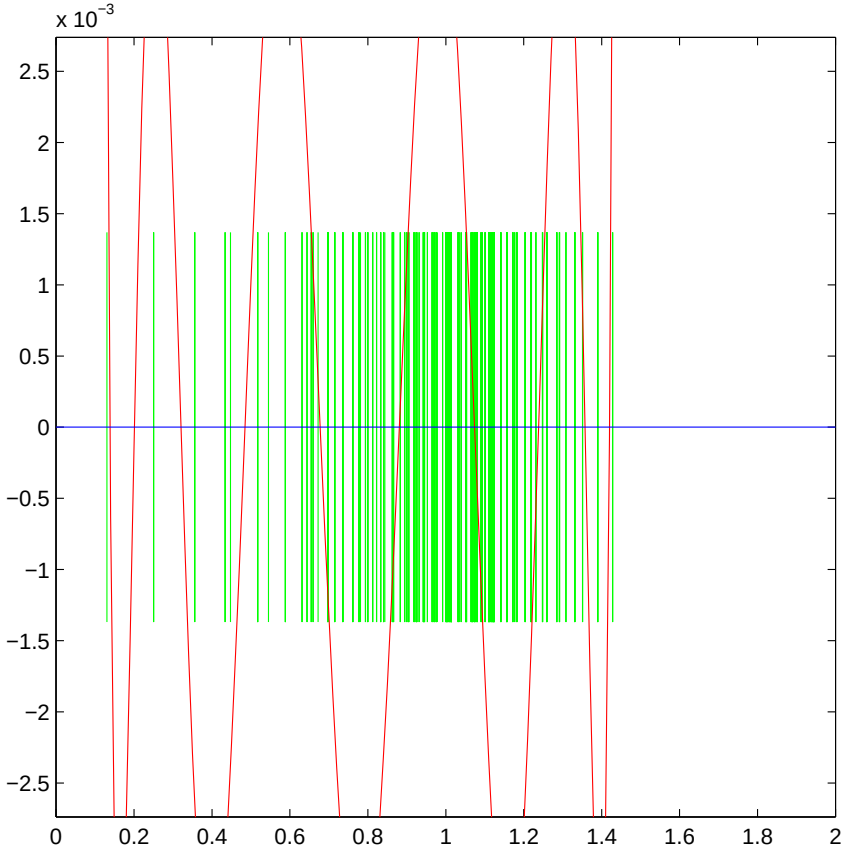


Minimax error: 0.5507e-1

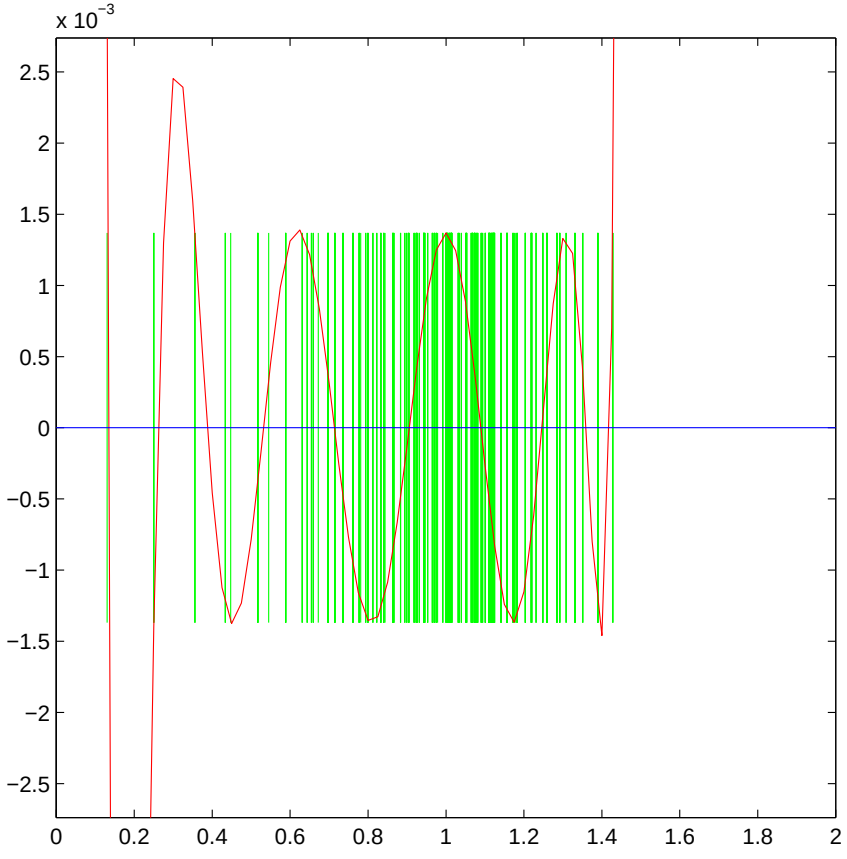


27 point Finite Element Stencil

Tchebyshev error: 0.3918e-2

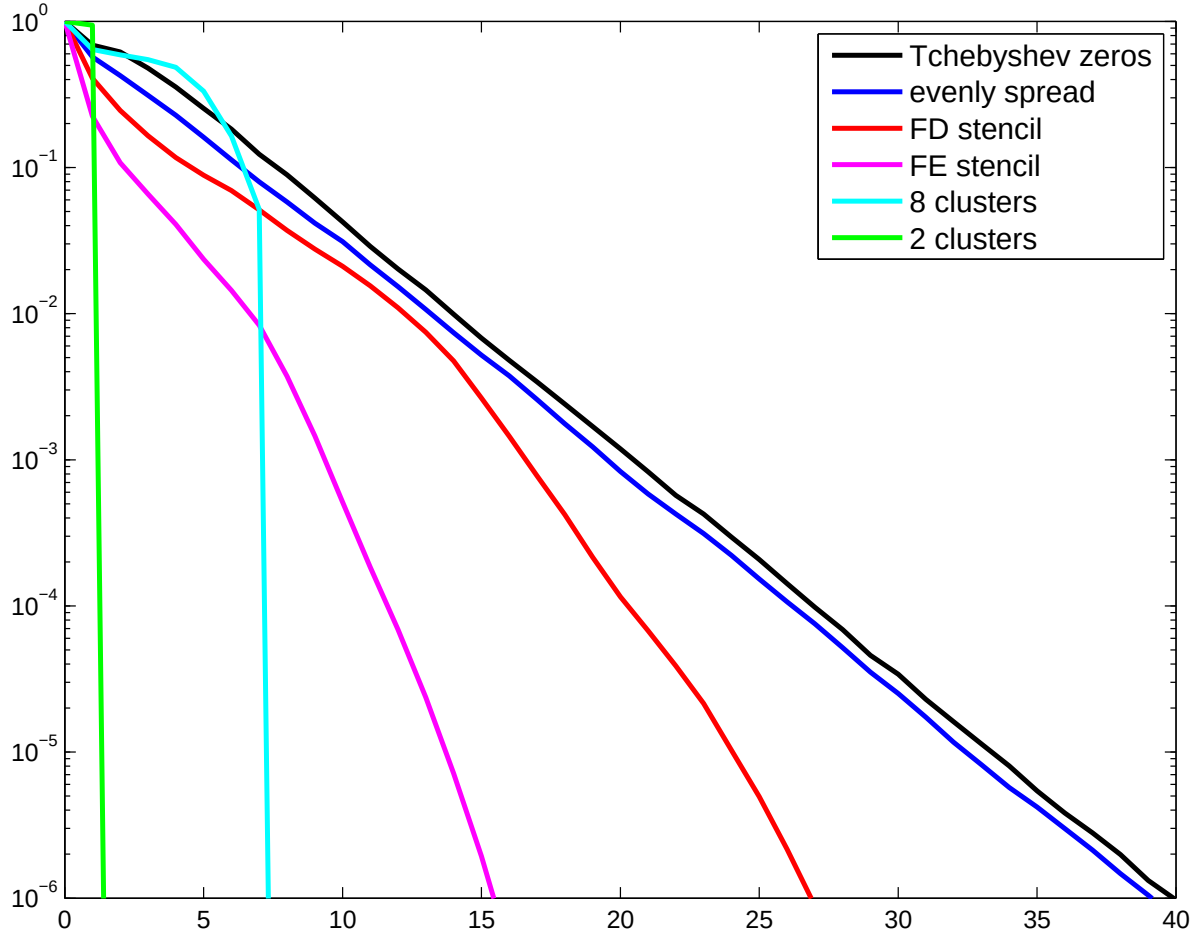


Minimax error: 0.1369e-2



CG residual reduction

512×512 matrices, zero initial guess, random RHS
different eigenvalue spectra, same condition number



CG Method In Practice

- Advantages:
  - involves only matrix-vector and dot products
  - exact solution obtained in at most s iterations
- Problems:
  - s may be very large
  - rounding error means theoretical properties lost
- Solution:
  - reduce the number of CG steps required by applying PRECONDITIONING (more later . . .)

Symmetric Indefinite Systems

If A is symmetric indefinite:

- A has both positive and negative (nonzero) eigenvalues
- $\mathbf{v}^T A\mathbf{v}$ may equal zero for some N-vector $\mathbf{v} \neq 0$

Potential problems with CG:

- A can no longer be used to define a norm
- breakdown may occur: denominator of α_k could be zero (or close to zero)

Conjugate Residual Method

Stiefel (1955)

Solve $A^2\mathbf{u} = A\mathbf{f}$ by CG method

CR method constructs iterates

$$\mathbf{u}_k \in \mathbf{u}_0 + \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\}$$

with properties

- $\mathbf{u}_k$ minimises $\|\mathbf{u}_k - \hat{\mathbf{u}}\|_{A^2} = \|\mathbf{r}_k\|_2$
- uses a three-term recurrence relation

symmetric eigenvalue intervals: convergence bound gives

$$k \propto \sqrt{\kappa(A^2)} = \kappa(A)$$

CR Algorithm

choose $\mathbf{u}_0$
compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{u}_0$
set $\mathbf{p}_0 = \mathbf{r}_0$
compute $A\mathbf{p}_0$
for $k = 0$ until convergence do

$$\alpha_k = \mathbf{r}_k^T \mathbf{r}_k / (A\mathbf{p}_k)^T A\mathbf{p}_k$$
$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{p}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$$
$$\beta_k = \mathbf{r}_{k+1}^T \mathbf{r}_{k+1} / \mathbf{r}_k^T A\mathbf{r}_k$$
$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$
$$A\mathbf{p}_{k+1} = A\mathbf{r}_{k+1} + \beta_k A\mathbf{p}_k$$

end do

can be implemented with one MVM per iteration

Alternative Methods

Potential problems with CR:

- breakdown may occur: denominator of α_k could be zero (or close to zero)
- CR algorithm is unstable in this form

Possible solution:

- generate an orthonormal basis for $\kappa(A, \mathbf{r}_0, k)$ in a more stable way, retaining the cheap three-term recurrence

$\Rightarrow$ mathematically equivalent but stable method ...

MINRES

Paige and Saunders (1975)

Construct iterates $\mathbf{u}_k = \mathbf{u}_0 + V_k \mathbf{y}_k$ with properties

- $\mathbf{u}_k$ minimises $\|\mathbf{r}_k\|_2$
- uses three-term recurrence relation

$$V_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

$\mathbf{v}_k$ form an orthonormal basis for $\kappa(A, \mathbf{r}_0, k)$

- use Lanczos method to find $\mathbf{v}_k$
- solve resulting least squares problem for $\mathbf{y}_k$ using Givens rotations and QR factorisation

MINRES Algorithm

Fischer (1996)

choose $\mathbf{u}_0$
compute $\hat{\mathbf{v}}_0 = \mathbf{f} - A\mathbf{u}_0$ — initialise
set $\beta_0 = \|\hat{\mathbf{v}}_0\|_2$, $\eta_0 = \beta_0$
set $c_0 = 1$, $c_{-1} = 1$, $s_0 = 0$, $s_{-1} = 0$

for $k = 0$ until convergence do

$$\mathbf{v}_{k+1} = \hat{\mathbf{v}}_k / \beta_k$$
$$\alpha_{k+1} = \mathbf{v}_{k+1}^T A \mathbf{v}_{k+1}$$ — Lanczos
$$\hat{\mathbf{v}}_{k+1} = (A - \alpha_{k+1} I)\mathbf{v}_{k+1} - \beta_k \mathbf{v}_k$$
$$\beta_{k+1} = \|\hat{\mathbf{v}}_0\|_2$$

$$\hat{r_1} = c_k \alpha_{k+1} - c_{k-1} s_k \beta_k$$
$$r_1 = \sqrt{\hat{r_1}^2 + \beta_{k+1}^2}$$
$$r_2 = s_k \alpha_{k+1} + c_{k-1} c_k \beta_k$$
$$r_3 = s_{k-1} \beta_k$$

QR

$$c_{k+1} = \hat{r_1} / r_1$$
$$s_{k+1} = \beta_{k+1} / r_1$$

Givens

$$\mathbf{w}_{k+1} = \left(\mathbf{v}_{k+1} - r_2 \mathbf{w}_k - r_3 \mathbf{w}_{k-1}\right) / r_1$$
$$\mathbf{u}_{k+1} = \mathbf{u}_k + c \eta_k \mathbf{w}$$
$$\eta_{k+1} = -s \eta_k$$

update

end do

Alternative methods

SYMMLQ

Paige and Saunders (1975)

Also has a strong Lanczos connection, but minimises the 2-norm of the error rather than the residual.

ORTHODIR

Fletcher (1976)

ORTHOMIN/ORTHODIR

Chandra et al. (1977)

Equivalent to MINRES, closer in implementation to CG/CR.

Nonsymmetric Systems

Faber and Manteuffel (1984 & 1987): there is no Krylov type method which retains both

(i) minimisation property

(ii) fixed length recurrence

- Normal Equations
  solve $A^T A\mathbf{u} = A^T\mathbf{f}$ using CG

- Minimum Residual Methods
  retain (i), sacrifice (ii)

- Biorthogonalisation Methods
  retain (ii), sacrifice (i)

CGNR

Hestenes and Stiefel (1952)

Apply CG to normal equations $A^T A\mathbf{u} = A^T\mathbf{f}$

Construct iterates

$$\mathbf{u}_k \in \mathbf{u}_0 + \mathtt{span}\{A^T\mathbf{r}_0, (A^TA)A^T\mathbf{r}_0, \ldots, (A^TA)^{k-1}A^T\mathbf{r}_0\}$$

satisfying

- $\mathbf{u}_k$ minimises $\|\mathbf{u}_k - \hat{\mathbf{u}}\|_{A^TA} = \|\mathbf{r}_k\|_2$
- uses three-term recurrence relation

convergence analysis gives

$$k \propto \sqrt{\kappa(A^TA)} = \kappa(A)$$

Generalised Minimal Residual Method (GMRES)

Saad and Schultz (1986)

Construct iterates $\mathbf{u}_k = \mathbf{u}_0 + V_k \mathbf{y}_k$ with properties

- $\mathbf{u}_k$ minimises $\|\mathbf{r}_k\|_2$
- no short-term recurrence

$$V_k = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k]$$

$\mathbf{v}_k$ form an orthonormal basis for $\kappa(A, \mathbf{r}_0, k)$

- Use the Arnoldi method to find $\mathbf{v}_k$
- solve resulting least squares problem for $\mathbf{y}_k$ using Givens rotations and QR factorisation

Some Observations

- some convergence analysis based on eigenvalues and vectors, singular values, pseudo-eigenvalues
- alternative implementations available e.g. based on Householder orthogonalisation: extra work but better numerical properties
- Restarted GMRES
  - restart GMRES every m steps
  - no simple rule for choosing m: convergence speed may vary drastically with different values
  - some convergence analysis available
- Simpler GMRES Walker and Zhou (1994)
  - calculate orthonormal basis for $A\kappa(A, \mathbf{f}, k)$ directly
  - may be useful for restarting with small m

Biorthogonalisation Methods

BiCG method

Construct iterates

$$\mathbf{u}_k = \mathbf{u}_0 + \mathtt{span}\{\mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{k-1}\mathbf{r}_0\}$$

with properties

- $\mathbf{r}_k \perp \mathtt{span}\{\hat{\mathbf{r}}_0, A\hat{\mathbf{r}}_0, \ldots, A^{k-1}\hat{\mathbf{r}}_0\}$
- uses three-term recurrence relation

- nonsymmetric Lanczos: generate two sets of biorthogonal vectors $\mathbf{v}_i$, $\mathbf{w}_i$
- potential problems:
  - wild oscillations in $\|\mathbf{r}_k\|_2$
  - possible breakdowns: $\hat{\mathbf{p}}_{k-1}^T A\mathbf{p}_{k-1} = 0$, $\hat{\mathbf{r}}_{k-1}^T \mathbf{r}_{k-1} = 0$
  - possible solution: look-ahead Lanczos

Biconjugate Gradient Method (BiCG)

Lanczos (1952), Fletcher (1976)

choose $\mathbf{u}_0$, compute $\mathbf{p}_0 = \mathbf{r}_0 = \mathbf{f} - A\mathbf{u}_0$
choose $\hat{\mathbf{r}}_0$
set $\hat{\mathbf{p}}_0 = \hat{\mathbf{r}}_0, \rho_0 = \hat{\mathbf{r}}_0^T \mathbf{r}_0$
for $k = 1, 2, \ldots$ until convergence do

$$\sigma_{k-1} = \hat{\mathbf{p}}_{k-1}^T A\mathbf{p}_{k-1}$$
$$\alpha_{k-1} = \rho_{k-1} / \sigma_{k-1}$$
$$\mathbf{u}_k = \mathbf{u}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}$$
$$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_{k-1} A\mathbf{p}_{k-1}$$
$$\hat{\mathbf{r}}_k = \hat{\mathbf{r}}_{k-1} - \alpha_{k-1} A^T \hat{\mathbf{p}}_{k-1}$$
$$\rho_k = \hat{\mathbf{r}}_k^T \mathbf{r}_k$$
$$\beta_{k-1} = \rho_k / \rho_{k-1}$$
$$\mathbf{p}_k = \mathbf{r}_k + \beta_k \mathbf{p}_{k-1}$$
$$\hat{\mathbf{p}}_k = \hat{\mathbf{r}}_k + \beta_k \hat{\mathbf{p}}_{k-1}$$

end

Quasi-Minimal Residual Method (QMR)

Freund and Nachtigal (1991)

- based on GMRES using nonsymmetric Lanczos
- too expensive to minimise $\|\mathbf{r}_k\|_2$: minimise "nearby" quantity quasi-minimal
- avoid Lanczos breakdown: do l steps of look-ahead Lanczos
- incurable breakdown (unlikely due to round-off)
- if A is symmetric, QMR $\equiv$ MINRES
- some convergence results are available

Transpose-free Methods

Transpose-Free QMR (TFQMR)

Freund (1991), Chan et al. (1991), Freund and Szeto (1991)

- A^T can be eliminated by choosing a suitable starting vector

Conjugate Gradients Squared (CGS)

Sonneveld (1989)

- construct iterates $\mathbf{u}_{2k} = \mathbf{u}_0 + \kappa(A, \mathbf{r}_0, 2k)$
- residual polynomials of CG are squared
- magnifies erratic convergence of BiCG
- may diverge when BiCG converges

BiCGSTAB

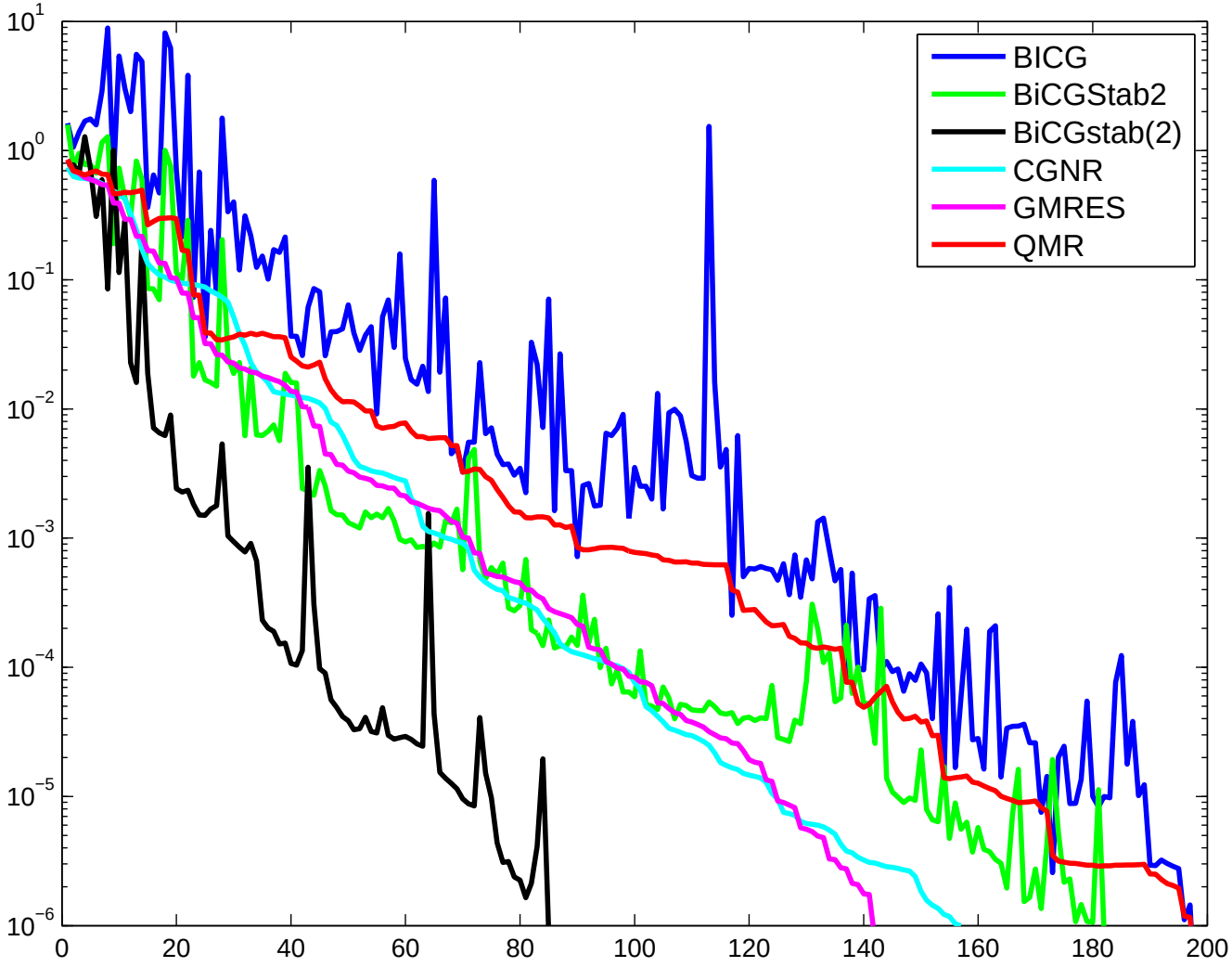
van der Vorst (1990)

- construct iterates $\mathbf{u}_{2k} = \mathbf{u}_0 + \kappa(A, \mathbf{r}_0, 2k)$
- residual polynomial updated with a linear factor at each step
- free parameter μ_k determined via a local steepest descents problem
- convergence typically much smoother than CGS

BiCGStab2, Gutnecht (1993)
BiCGstab(l), Sleijpen and Fokkema (1993)

Example: Calculation of Invariant Tori

$$a(X,Y)\frac{\partial s}{\partial X} + b(X,Y)\frac{\partial s}{\partial Y} + c(X,Y)s = \psi$$



Summary

- symmetric positive definite CG
- symmetric indefinite CR, MINRES, SYMMLQ
- nonsymmetric
  - normal equations
    CGNR
  - minimisation
    GMRES, GMRES(m)
  - biorthogonalisation
    BiCG, CGS, BiCGSTAB, BiCGstab(l), QMR

Preconditioning

Idea: instead of solving $A\mathbf{u} = \mathbf{f}$, solve

$$M^{-1}A\mathbf{u} = M^{-1}\mathbf{f}$$

for some preconditioner M

Choose M so that

(i) eigenvalues of $M^{-1}A$ are well clustered
(ii) $M\mathbf{u} = \mathbf{r}$ is easily solved

Extreme cases:

- $M = A$: good for (i), bad for (ii)
- $M = I$: good for (ii), bad for (i)

Preconditioned Conjugate Gradient Method

Concus, Golub & O'Leary (1976)

choose $\mathbf{u}_0$
compute $\mathbf{r}_0 = \mathbf{f} - A\mathbf{u}_0$
solve $M\hat{\mathbf{r}}_0 = \mathbf{r}_0$
set $\mathbf{p}_0 = \mathbf{r}_0$
for $k = 0$ until convergence do

$$\alpha_k = \mathbf{r}_k^T \hat{\mathbf{r}}_k / \mathbf{p}_k^T A \mathbf{p}_k$$
$$\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{p}_k$$
$$\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A \mathbf{p}_k$$

solve $M\hat{\mathbf{r}}_{k+1} = \mathbf{r}_{k+1}$

$$\beta_k = \mathbf{r}_{k+1}^T \hat{\mathbf{r}}_{k+1} / \mathbf{r}_k^T \hat{\mathbf{r}}_k$$
$$\mathbf{p}_{k+1} = \hat{\mathbf{r}}_{k+1} + \beta_k \mathbf{p}_k$$

end do

Practical Implementation

- preconditioner $M = M_1 M_2$
- solve $M\mathbf{u} = \mathbf{r}$, $M^T\mathbf{u} = \mathbf{r}$
- new system

$$[M_1^{-1} A M_2^{-1}][M_2\mathbf{u}] = M_1^{-1}\mathbf{f} \Rightarrow \tilde{A}\tilde{\mathbf{u}} = \tilde{\mathbf{f}}$$

- $\mathbf{u}_k = M_2^{-1}\tilde{\mathbf{u}}_k$, $\mathbf{r}_k = M_1\tilde{\mathbf{r}}_k$

central : as above
left : $M_2 = I$
right : $M_1 = I$

symmetric positive definite: if $M_2 = M_1^T$, resulting system is also symmetric positive definite

symmetric indefinite: M must be symmetric positive definite for MINRES; M can be indefinite with QMR

nonsymmetric:

- central: analysis may be easier
- left: if $M^{-1}A \simeq I$, $\tilde{\mathbf{r}}_k = M^{-1}A(\mathbf{u}_k - \hat{\mathbf{u}}) \simeq \mathbf{u}_k - \hat{\mathbf{u}}$, i.e.

$$\|\tilde{\mathbf{r}}_k\|_2 \simeq \|\mathbf{u}_k - \hat{\mathbf{u}}\|_2$$

- right: minimise in correct norm, i.e.

$$\|\tilde{\mathbf{r}}_k\|_2 = \|\mathbf{r}_k\|_2$$

Connection with Stationary Methods

matrix splitting $A = M - N$

Iterates

$$\mathbf{u}_{k+1} = M^{-1}N\mathbf{u}_k + M^{-1}\mathbf{f} = \mathbf{u}_k + M^{-1}\mathbf{r}_k$$

where the error satisfies

$$\mathbf{u}_k - \hat{\mathbf{u}} = (I - M^{-1}A)^k(\mathbf{u}_0 - \hat{\mathbf{u}}).$$

$(I - M^{-1}A)$ is small $\Rightarrow$ rapid convergence

good preconditioner $\equiv$ good splitting operator

Stationary Methods as Preconditioners

- Jacobi (diagonal scaling)
  - very simple to implement, minimal storage requirements
  - scales condition number
  - still competitive for extremely large 3D problems: may be better to do more cheaper iterations than fewer expensive ones
- Gauss-Seidel, SOR
  - for CG, M must be symmetric
  - applying the method twice per iteration (once *forward* then once *backward*)
- preconditioners based on stationary methods are widely applicable

Incomplete LU Factorisation

Step 1: select set $J = \{(i,j) : 1 \leq i,j \leq N\}$ of index pairs (including all (i,i))

Step 2: perform LU factorisation and restrict all non-zeros to entries in J

$$A = LU - R = M - R$$

$$r_{ij} = 0, \ (i,j) \in J, \qquad r_{ii} = \alpha \sum_{i \neq j} r_{ij}$$

- ILU factorisations do not always exist
- very sequential in nature
- block matrix analogues

Parameterised Incomplete Factorisation

for $i = 1, \ldots, n$ do
  for $j = 1, \ldots, n$ do
    $s_{ij} = a_{ij} - \sum_{k=1}^{\min(i,j)-1} l_{ik} u_{kj}$
    if $(i, j) \in J$ then
      if $(i \geq j)$ then $l_{ij} = s_{ij}$
      if $(i < j)$ then $u_{ij} = s_{ij}$
    else
      $l_{ii} = l_{ii} + \alpha s_{ij}$
    endif
  enddo
  $u_{ii} = 1$
  for $j = i + 1, \ldots, n$ do
    $u_{ij} = u_{ij}/l_{ii}$
  enddo
enddo

Some Variations on ILU

- $J \equiv$ nonzero entries in A

  $\alpha = 0$: ILU, Meijerink and van der Vorst (1977)
  $\alpha = 1$: MILU, Gustafsson (1978)

- ILU(N), MILU(N)
  J includes N extra diagonals

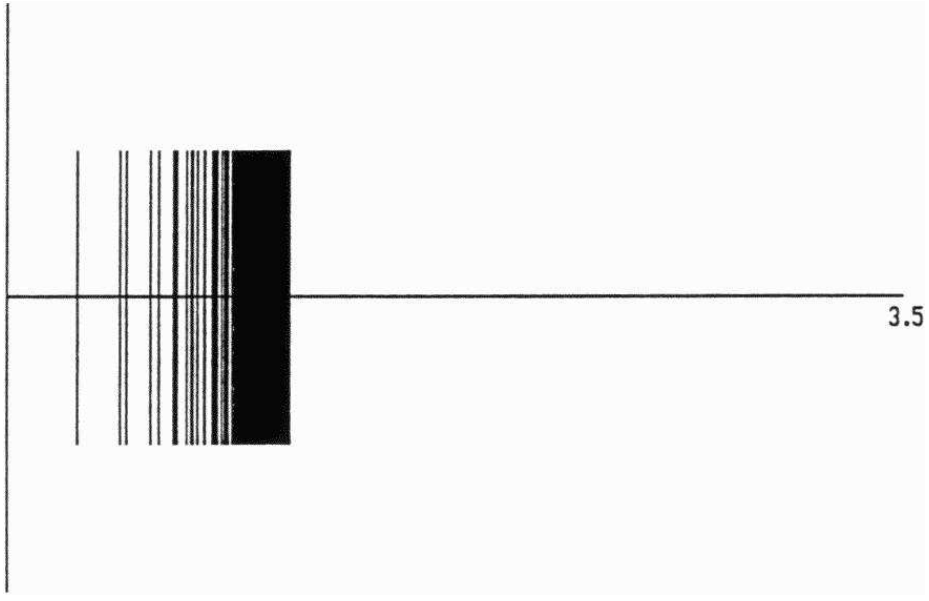
- ILU with Drop Tolerance, Munksgaard (1980)
  Drop all entries of fill-in with absolute value less than $\tau \in [10^{-4}, 10^{-2}]$.

- Shifted ILU, Manteuffel (1978,1980)
  Make A more diagonally dominant by factorising

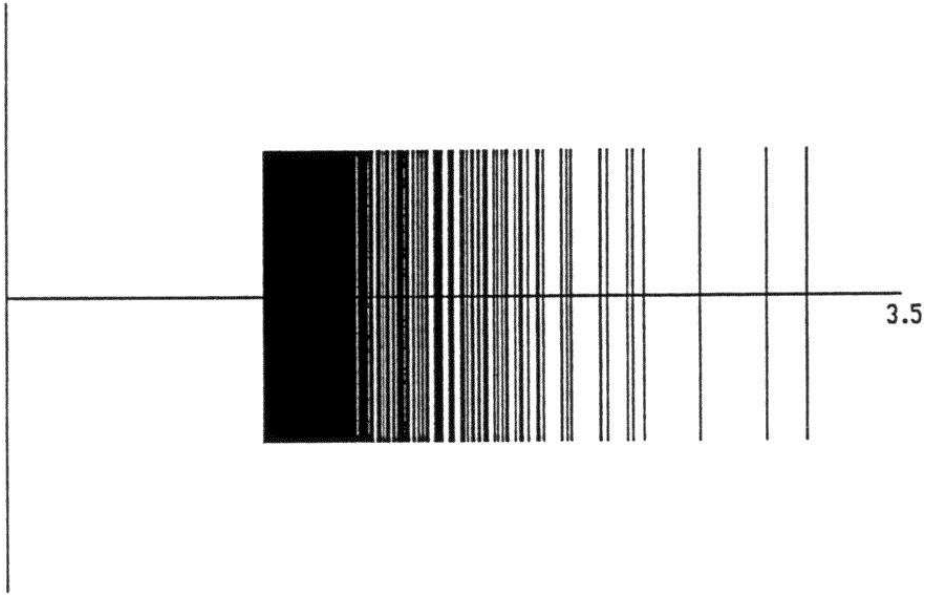
$$\bar{A} = D + \frac{1}{1+\gamma} C.$$

Sample Eigenvalue Plots

- seven point finite difference stencil



Incomplete Cholesky



Modified Incomplete Cholesky

SParse Approximate Inverse (SPAI)

- minimise $\|AM - I\|$ in the Frobenius norm
- approximate inverse computed explicitly so can be applied as a preconditioner
- sparsity pattern of approximate inverse calculated dynamically
- user controls the quality and cost
  - if M is sparse, it is cheap to compute but may not improve things much
  - as M becomes dense, it becomes more expensive to compute
  - optimal preconditioner lies between these two extremes and is problem and computer architecture dependent

Polynomial Preconditioning

- apply CG to $p(B^{-1}A)B^{-1}A\mathbf{u} = p(B^{-1}A)B^{-1}\mathbf{f}$ i.e. use

$$M^{-1} = p(B^{-1}A)B^{-1}$$

  so that $\mathbf{u} = M^{-1}\mathbf{r}$ is easily solved

- choose B to be a matrix splitting from stationary methods e.g.
  - B from SSOR gives m-step CG method (Adams (1985))
  - $B = I$ from Richardson (Ashby (1987))
  - applying preconditioner involves only MVMs so may be good for vector/parallel machines

Domain Decomposition

- break down underlying elliptic PDE problem into individual parts that can be solved separately
- piece results together to get solution to whole problem
- use different preconditioners for different parts of the grid
- two main types:
  - overlapping methods: additive/multiplicative Schwarz
  - nonoverlapping methods: substructuring

Element-By-Element Method

Hughes, Levit & Winget (1983)

- assemble one element matrix A_e into G_e
- form $\bar{G}_e = I + D^{-\frac{1}{2}}(G_e - D_e)D^{-\frac{1}{2}}$
- factorise $\bar{G}_e = \mathcal{L}_e \mathcal{D}_e \mathcal{L}_e^T$
  ($\mathcal{L}_e$ is the assembly of lower $\triangle$ factor of related $\bar{A}_e$)
- preconditioner is

$$M = D^{\frac{1}{2}} \left[\prod_{e=1}^{E} \mathcal{L}_e \right] \left[\prod_{e=1}^{E} \mathcal{D}_e \right] \left[\prod_{e=E}^{1} \mathcal{L}_e^T \right] D^{\frac{1}{2}}$$

Element Factorisation Method

Kaasschieter (1989)

- factor element matrices $A_e = (D_e + L_e)D_e^+(D_e + L_e)^T$ (D_e^+ is the generalised inverse of D_e)
- number without maximal global node numbers (using e.g. Reverse Cuthill-McKee numbering)
- form $\mathcal{L} = L^T[L_e]L$, $\mathcal{D} = L^T[D_e]L$
- preconditioner is

$$M = (\mathcal{D} + \mathcal{L})\mathcal{D}^{-1}(\mathcal{D} + \mathcal{L}^T)$$

Multigrid Methods

- developed for the solution of boundary value problems
- geometric multigrid (GMG)
  - based on sequence of physical grids associated with the problem
- algebraic multigrid (AMG)
  - no need for a physical grid, based on sparse matrix properties
- both methods very powerful when used either as preconditioners or as solvers in their own right
- multigrid tutorial by Briggs et al. (2000)

MG Relaxation

- stationary method: splitting $A = M - N$
- stationary iteration (relaxation)

$$\mathbf{u}_k = M^{-1}N\mathbf{u}_{k-1} + M^{-1}\mathbf{f} \quad \Rightarrow \qquad \mathbf{u}_k = R\mathbf{u}_{k-1} + M^{-1}\mathbf{f}$$

$$A\hat{\mathbf{u}} = \mathbf{f} \Rightarrow [M - N]\hat{\mathbf{u}} = \mathbf{f} \quad \Rightarrow$$

$$M\hat{\mathbf{u}} = N\hat{\mathbf{u}} + \mathbf{f} \quad \Rightarrow \qquad \hat{\mathbf{u}} = R\hat{\mathbf{u}} + M^{-1}\mathbf{f}$$

iteration matrix R

- subtracting: $\hat{\mathbf{u}} - \mathbf{u}_k = R[\hat{\mathbf{u}} - \mathbf{u}_{k-1}] \quad \Rightarrow \mathbf{e}_k = R\mathbf{e}_{k-1}$

error $\mathbf{e}_k = \hat{\mathbf{u}} - \mathbf{u}_k$

Error Bound

- bound on error after k iterations:

$$\|\mathbf{e}_k\| \leq \|R\|^k \|\mathbf{e}_0\|$$

- if $\|R\| < 1$, error will tend to zero

- can show

$$\lim_{k \to \infty} R^k = 0 \quad \Leftrightarrow \quad \rho(R) < 1$$

spectral radius $\rho(R) \equiv |\lambda_{\max}(R)|$

- method converges for any $\mathbf{u}_0$ if and only if $\rho(R) < 1$

Weighted Jacobi Method

- treat Jacobi iterate as intermediate value, $\mathbf{u}^*$

$$\mathbf{u}^* = D^{-1}(L+U)\mathbf{u}_{k-1} + D^{-1}\mathbf{f}$$

- take 'final' iterate as a weighted average of $\mathbf{u}_{k-1}$ and $\mathbf{u}^*$

$$\begin{aligned} \mathbf{u}_k &= (1-\omega)\mathbf{u}_{k-1} + \omega\mathbf{u}^* \\ &= [(1-\omega)I + \omega R_J]\hat{\mathbf{u}} + \omega D^{-1}\mathbf{f} \\ \Rightarrow \mathbf{u}_k &= R_\omega \mathbf{u}_{k-1} + \omega D^{-1}\mathbf{f} \end{aligned}$$

weighting factor $\omega \in \mathbb{R}$, identity matrix I

iteration matrix $R_\omega = [(1-\omega)I + \omega R_J]$

1D Model Problem

- one-dimensional model Poisson problem

$$-u''(x) = f(x), \quad 0 < x < 1, \quad u(0) = u(1) = 0$$

- linear finite elements, uniform grid Ω^h, $h = 1/N$
- discrete system

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h} = hf_i, \qquad 1 \leq i \leq N-1, \quad u_0 = u_N = 0$$

- solve $A^h \mathbf{u} = \mathbf{f}$ where

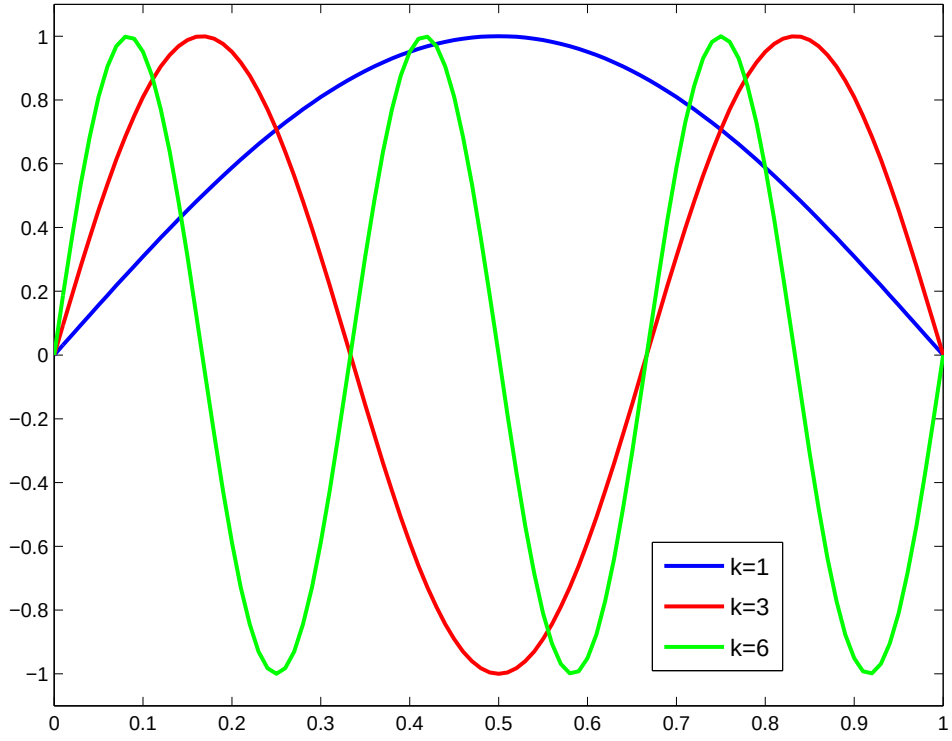
$$A^h = \text{tridiag}(-1, 2, -1)$$

Fourier Modes

- test weighted Jacobi with Fourier modes

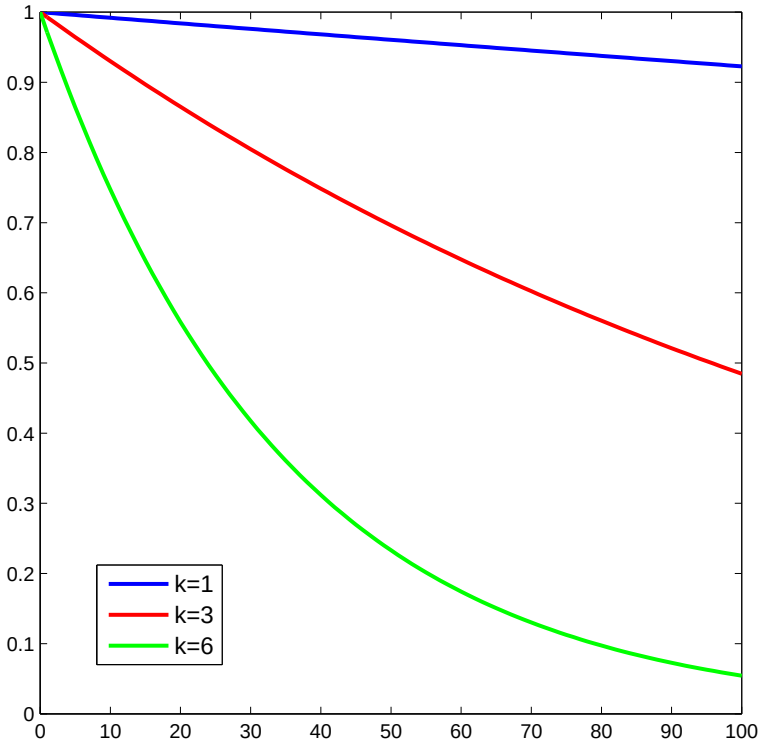
$$\mathbf{v}^s = \{v_j^s\}, \quad v_j^s = \sin\left(\frac{js\pi}{N}\right), \quad 1 \le j \le N-1,\ 1 \le s \le N-1$$

- wavenumber s gives the number of half sine waves that make up the vector $\mathbf{u}$ on the problem domain



Convergence of Weighted Jacobi (1)

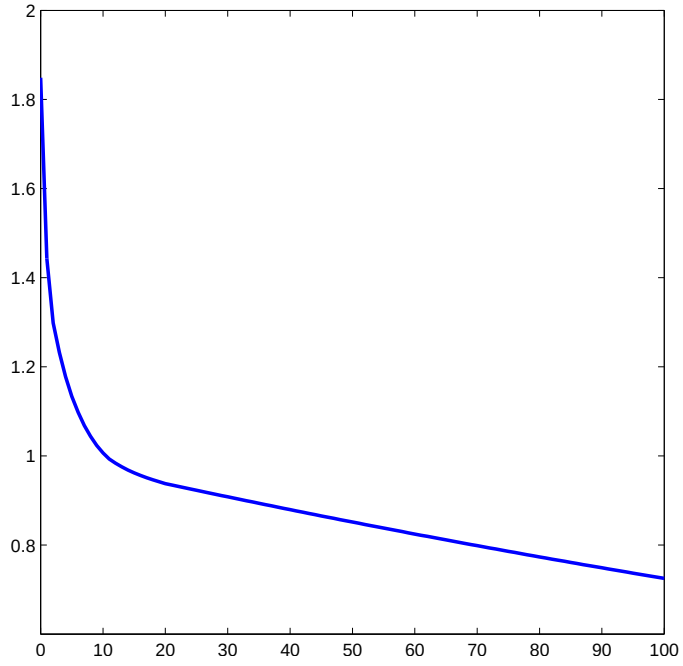
- weighted Jacobi method, $\omega = 2/3$, $N = 64$
- three initial iterates $\mathbf{v}^1$, $\mathbf{v}^3$ and $\mathbf{v}^6$



- error is decreased at each step
- rate of convergence increases with larger wavenumber

Convergence of Weighted Jacobi (2)

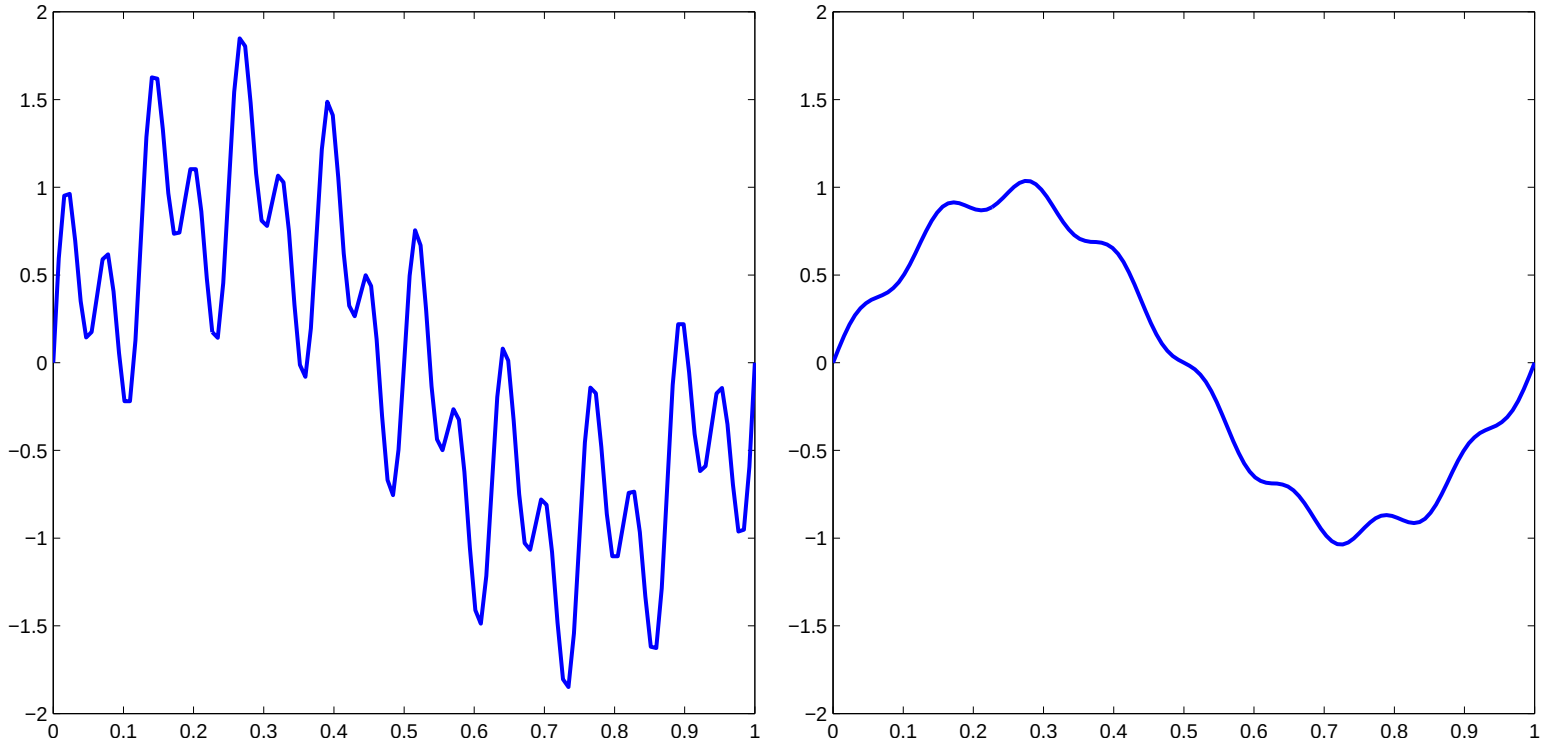
- initial guess will usually consist of a sum of Fourier modes, e.g. $\mathbf{v}^* = \mathbf{v}^2 + \mathbf{v}^{16}/2 + \mathbf{v}^{32}/2$



- initial fast convergence corresponds to the elimination of the high frequency mode
- subsequent rate much slower decrease due to the low frequency

Convergence of Weighted Jacobi (3)

- $\mathbf{e}_0$ highly oscillatory, $\mathbf{e}_{35}$ much smoother



- smoothing property: oscillatory modes in the error are eliminated effectively, smooth modes are damped very slowly

Nested Iteration

AIM: calculate an improved initial guess for relaxation on Ω^h

IDEA: use a coarse grid Ω^{2h}

- relaxation is cheaper on coarse grid (half, quarter, eighth of fine grid points)
- slightly better convergence rate.

nested iteration

start from some coarsest grid level

$\vdots$

relax on Ω^{4h} to get initial guess $\mathbf{u}^{2h}$

relax on Ω^{2h} to get initial guess $\mathbf{u}^{h}$

relax on Ω^{h} to get a final solution

Fourier Modes on Coarse Grids

- $s = 1, 2, \ldots, N/2$: $\quad v_{2j}^{s,h} = v_j^{s,2h}$
  - now s^{th} mode out of $N/2$
  - relatively higher wavenumber, more effectively treated by relaxation

- $s = N/2, \ldots, N-1$: $\quad v_{2j}^{s,h} = -v_j^{N-s,2h}$
  - s^{th} mode on the fine grid appears as the $(N-s)$th mode on the coarse grid:
  - aliasing: these modes are essentially invisible on the coarse grid

smooth error will be more oscillatory on a coarse grid

Residual Correction

residual $\mathbf{r}_k = \mathbf{f} - A^h\mathbf{u}_k = A^h\hat{\mathbf{u}} - A^h\mathbf{u}_k$:

residual equation $A^h\mathbf{e}_k = \mathbf{r}_k$

residual correction

1. Given $\mathbf{u}_{k-1}$, compute $\mathbf{r}_{k-1} = \mathbf{f} - A^h\mathbf{u}_{k-1}$.
2. Solve $A^h\mathbf{e}_{k-1} = \mathbf{r}_{k-1}$.
3. Compute $\mathbf{u}_k = \mathbf{u}_{k-1} + \mathbf{e}_{k-1}$.

- relax on $A^h\mathbf{u} = \mathbf{f}$ to smooth error
- move to coarser grid (so error will be more oscillatory)
- relax on residual equation with $\mathbf{e}_0 = \mathbf{0}$

Coarse Grid Correction

do ν_1 smoothing iterations on $A^h\mathbf{y} = \mathbf{f}^h$, initial guess $\mathbf{y} = \mathbf{u}_k$

compute residual $\mathbf{r}^h = \mathbf{f} - A^h\mathbf{y}_m$

transfer $\mathbf{r}^h$ to coarse grid ($\rightarrow \mathbf{r}^{2h}$)

solve residual equation $A^{2h}\mathbf{e}^{2h} = \mathbf{r}^{2h}$ for the error

transfer $\mathbf{e}^{2h}$ to fine grid ($\rightarrow \mathbf{e}^h$)

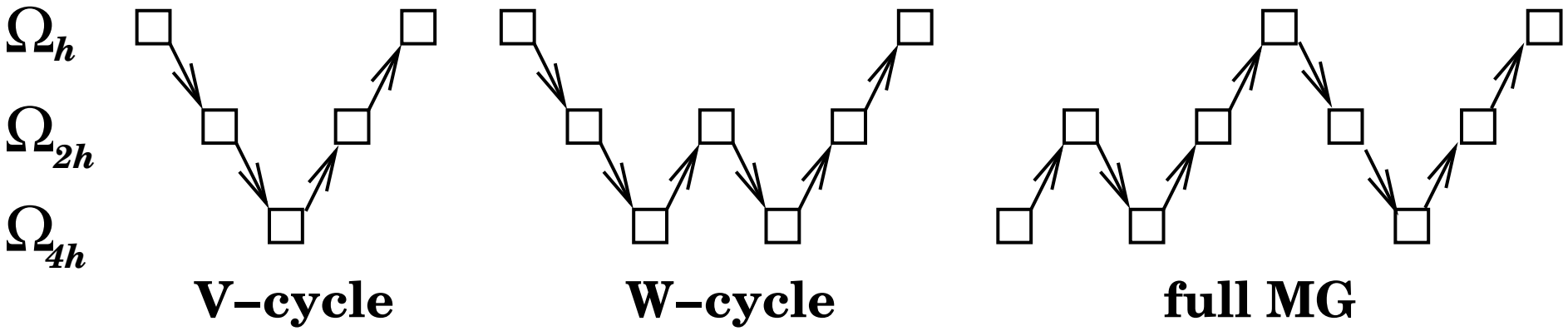
correct approximation: $\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{e}^h$

do ν_2 smoothing iterations on $A^h\mathbf{y} = \mathbf{f}^h$, initial guess $\mathbf{y} = \mathbf{u}_{k+1}$

Points to Note

- ν_1 and ν_2 are fixed integers
- grid transfer operators
  - prolongation (coarse to fine): interpolation
  - restriction (fine to coarse): injection, full weighting
- coarse grid operator A^{2h}
  - discretisation coarse grid approximation (DCA): discretise directly on coarse grid
  - Galerkin coarse grid approximation (GCA): transform A^h using restriction and prolongation operators

Multigrid Cycles



- down arrows represent restriction
- up arrows represent prolongation
- smoothing steps are performed on each level
- patterns extend to as many grids as needed
- use direct method on coarsest grid

GMG Convergence

- for elliptic PDE problems on uniform grids, convergence analysis using Fourier modes
- general convergence analysis is tricky

With the right combination of components, the number of operations involved is of the order of the total number of unknowns in the linear system, i.e. work $\propto N$.

- best possible (takes N operations to write the solution down!)
- the number of iterations is independent of N and does not grow as the underlying finite element grid is refined
- MG is scalable and can be easily parallelised

Algebraic Multigrid (AMG)

- no grid needed, works on sparse matrix
- construct undirected adjacency graph with edge between i and j whenever either of a_{ij}, a_{ji} is nonzero
- coarse-grid unknowns are a subset of the variables, identified by numerical indices
- choose coarse grid set to represent smooth functions accurately
  - smooth functions defined algebraically as those functions which are not effectively dealt with by relaxation
  - technical requirements can be worked out using convergence analysis of relaxation methods

AMG (2)

- constraints on coarse grid variables:
  - need good representation of smooth errors
  - need interpolation of these errors onto fine grid
  - coarse grid must have substantially fewer grid points
- identify grid dependencies:
  - u_i strongly depends on u_j if

    $$|a_{ij}| \simeq |\text{ largest off-diagonal entry in row } i \text{ of } A|$$

  - points exerting strong influence on other points included in the coarse grid, other points neglected
  - initial partitioning refined as suitable interpolation and restriction operators are defined according to some given rules

AMG (3)

- coarse-grid operator A^{2h} constructed via Galerkin coarse grid approximation
- two-grid correction scheme exactly the same as for GMG
- recursive calls used to set up various patterns on a full series of grids
- broad range of applicability
- theory of convergence behaviour relatively undeveloped
- AMG replicates the attractive $O(N)$ behaviour shown by GMG
- particularly attractive for unstructured grid problems
- developments towards element-based versions for paralellising (e.g. AMGe)

Points to Note

- MG applications include elliptic,parabolic and hyperbolic PDEs, integral equations, evolution problems, . . .
- nonlinear, anisotropic versions
- MG methods used increasingly as preconditioners
  - often only one or two V-cycles required per CG/GMRES iteration
  - often more robust as MG tends to be less sensitive to the tuning of specifics such as grid transfer operators, type and amount of smoothing, etc.

Examples of Stopping Criteria

- standard tests: $\|\mathbf{r}_k\|_2 \leq \epsilon, \quad \frac{\|\mathbf{r}_k\|_2}{\|\mathbf{r}_0\|_2} \leq \epsilon$
- with left preconditioning: $\|M_1\mathbf{r}_k\|_2 \leq \epsilon, \quad \frac{\|M_1\mathbf{r}_k\|_2}{\|M_1\mathbf{r}_0\|_2} \leq \epsilon$
- condition number dependent (e.g. Ashby et al. (1990)):

$$\frac{\|\mathbf{u}_k - \hat{\mathbf{u}}\|_2}{\|\mathbf{u}_0 - \hat{\mathbf{u}}\|_2} \leq \kappa(A)\frac{\|\mathbf{r}_k\|_2}{\|\mathbf{r}_0\|_2} \leq \epsilon$$

$$\frac{\|\mathbf{u}_k - \hat{\mathbf{u}}\|_A}{\|\mathbf{u}_0 - \hat{\mathbf{u}}\|_A} \leq \left(\kappa_A(A)\left|\frac{\gamma_k}{\gamma_0}\right|\right)^{\frac{1}{2}} \leq \epsilon$$

- backward error analysis (e.g. Arioli et al. (1991)):

$$\frac{\|\mathbf{r}_k\|_\infty}{\|A\|_\infty\|\mathbf{u}_k\|_1 + \|\mathbf{r}_0\|_\infty} \leq \epsilon$$

Software Packages

Available from netlib:

- by WWW:

  http://www.netlib.org
  http://www.netlib.org/bib/mirrors.html

- by email: mail **netlib@netlib.org** with

  send index from linalg

- by anonymous ftp:

  ftp.netlib.org

- list of free linear algebra software:

  http://www.netlib.org/utk/people/JackDongarra/la-sw.html

Some Examples

- some examples of packages on `netlib`:
  - `ITPACK` (FORTRAN),
  - `SLAP` (FORTRAN),
  - `linalg/laspack` (C),
  - `linalg/qmrpack` (FORTRAN),
  - `linalg/templates` (C, FORTRAN),
  - `linalg/cg` (PVM).
- some other freely-available codes:
  - `HYPRE` (FORTRAN,C),
  - `AZTEC` (MPI),
  - `LAPACK` (FORTRAN),
  - `TRILINOS` (C++).

Some Relevant Books

- Templates for the Solution of Linear Systems. . . , Barrett et al. , SIAM (1994)
- Iterative Solution Methods, Axelsson, CUP (1996)
- Iterative Methods for Sparse Linear Systems, Saad, PWS (1996)
- Iterative Methods for Solving Linear Systems, Greenbaum, SIAM (1997)
- Computer Solution of Large Linear Systems, Meurant, North-Holland (1999)
- Iterative Krylov Methods for Large Linear Systems, van der Vorst, CUP (2003)

Some Summary Papers

- Iterative Solution of Linear Systems, Freund, Golub and Nachtigal, Acta Numerica (1991)
- Developments and Trends in the Parallel Solution of Linear Systems, Duff and van der Vorst, Parallel Computing 25 (1999)
- Numerical Progress in Eigenvalue Computation in the 20th Century, Golub and van der Vorst, J. Comp. and Appl. Math. 123 (2000)
- Iterative Solution of Linear Systems in the 20th Century, Saad and van der Vorst, J. Comp. and Appl. Math. 123 (2000)
- Preconditioning Techniques for Large Linear Systems: A Survey, Benzi, J. Comput. Phys. (2003)