Parallel Numerics, WT 2016/2017

5 Iterative Methods for Sparse Linear Systems of Equations
1 Introduction
  1.1 Computer Science Aspects
  1.2 Numerical Problems
  1.3 Graphs
  1.4 Loop Manipulations
2 Elementary Linear Algebra Problems
  2.1 BLAS: Basic Linear Algebra Subroutines
  2.2 Matrix-Vector Operations
  2.3 Matrix-Matrix-Product
3 Linear Systems of Equations with Dense Matrices
  3.1 Gaussian Elimination
  3.2 Parallelization
  3.3 QR-Decomposition with Householder matrices
4 Sparse Matrices
  4.1 General Properties, Storage
  4.2 Sparse Matrices and Graphs
  4.3 Reordering
  4.4 Gaussian Elimination for Sparse Matrices
5 Iterative Methods for Sparse Linear Systems of Equations
  5.1 Stationary Methods
  5.2 Nonstationary Methods
  5.3 Preconditioning
6 Domain Decomposition
  6.1 Overlapping Domain Decomposition
  6.2 Non-overlapping Domain Decomposition
  6.3 Schur Complements
- Disadvantages of direct methods (in parallel):
  - strongly sequential
  - may lead to dense matrices
  - sparsity pattern changes, additional entries necessary
  - indirect addressing
  - storage
  - computational effort

- Iterative solver:
  - choose initial guess = starting vector $x(0)$, e.g., $x(0) = 0$
  - iteration function $x(k+1) := \Phi(x(k))$

- Applied on solving a linear system:
  - Main part of $\Phi$ should be a matrix-vector multiplication $Ax$ (matrix-free!?)
  - Easy to parallelize, no change in the pattern of $A$.
  - $x(k) \to \infty \tooverrightarrow{\bar{x}} = A^{-1}b$

- Main problem: Fast convergence!
• Disadvantages of direct methods (in parallel):
  – strongly sequential
  – may lead to dense matrices
  – sparsity pattern changes, additional entries necessary
  – indirect addressing
  – storage
  – computational effort

• Iterative solver:
  – choose initial guess = starting vector \( x^{(0)} \), e.g., \( x^{(0)} = 0 \)
  – iteration function \( x^{(k+1)} := \Phi(x^{(k)}) \)
• Disadvantages of direct methods (in parallel):
  – strongly sequential
  – may lead to dense matrices
  – sparsity pattern changes, additional entries necessary
  – indirect addressing
  – storage
  – computational effort

• Iterative solver:
  – choose initial guess = starting vector \( x^{(0)} \), e.g., \( x^{(0)} = 0 \)
  – iteration function \( x^{(k+1)} := \Phi(x^{(k)}) \)

• Applied on solving a linear system:
  – Main part of \( \Phi \) should be a matrix-vector multiplication \( Ax \)
    (matrix-free!?)
  – Easy to parallelize, no change in the pattern of \( A \).
    \[ x^{(k)} \xrightarrow{k \to \infty} \bar{x} = A^{-1} b \]
  – Main problem: Fast convergence!
5.1. Stationary Methods

5.1.1. Richardson Iteration
• Construct from $Ax = b$ an iteration process:

\[
b = Ax = (A - I + I)x = x - (I - A)x \quad \Rightarrow \quad x = b + (I - A)x
\]

(artificial) splitting of $A$

\[
= b + Nx
\]
• Construct from $Ax = b$ an iteration process:

$$b = Ax = (A - I + I) x = x - (I - A)x \quad \Rightarrow \quad x = b + (I - A)x$$

(artificial) splitting of $A$

$$= b + Nx$$

• Leads to equation $x = \Phi(x)$ with $\Phi(x) := b + Nx$:

\begin{align*}
\text{start: } x^{(0)}; \\
\quad x^{(k+1)} := \Phi(x^{(k)}) = b + Nx^{(k)} = b + (I - A)x^{(k)}
\end{align*}
Richardson Iteration (cont.)

\[
\begin{align*}
\text{start: } & x^{(0)}; \\
& x^{(k+1)} := \Phi(x^{(k)}) = b + N x^{(k)} = b + (I - A) x^{(k)} \\
\text{If } x^{(k)} \text{ is convergent, } & x^{(k)} \to \tilde{x}, \\
\text{then} & \\
\tilde{x} = & \Phi(\tilde{x}) = b + N \tilde{x} = b + (I - A)\tilde{x} \Rightarrow A\tilde{x} = b \\
\text{and therefore it holds} & \\
& x^{(k)} \to \tilde{x} = \bar{x} := A^{-1}b
\end{align*}
\]
Richardson Iteration (cont.)

start: \( x^{(0)} \);

\[
x^{(k+1)} := \Phi(x^{(k)}) = b + Nx^{(k)} = b + (I - A)x^{(k)}
\]

If \( x^{(k)} \) is convergent, \( x^{(k)} \to \tilde{x} \), then

\[
\tilde{x} = \Phi(\tilde{x}) = b + N\tilde{x} = b + (I - A)\tilde{x} \Rightarrow A\tilde{x} = b
\]

and therefore it holds

\[
x^{(k)} \to \tilde{x} = \bar{x} := A^{-1}b
\]

Residual-based formulation:

\[
x^{(k+1)} = \Phi(x^{(k)}) = b + (I - A)x^{(k)} = b + x^{(k)} - Ax^{(k)} = x^{(k)} + (b - Ax^{(k)})
\]

\[
= x^{(k)} + r(x^{(k)}) = x^{(k)} + r(x)
\]

\( r(x) \) = residual
Convergence Analysis via Neumann Series

\[ x^{(k)} = b + N x^{(k-1)} = b + N \left( b + N x^{(k-2)} \right) = b + Nb + N^2 x^{(k-2)} = \]
\[ \cdots = b + Nb + N^2 b + \cdots + N^{k-1} b + N^k x^{(0)} = \]
\[ = \sum_{j=0}^{k-1} N^j b + N^k x^{(0)} = \left( \sum_{j=0}^{k-1} N^j \right) b + N^k x^{(0)} \]

Special case \( x^{(0)} = 0 \):

\[ x^{(k)} = \left( \sum_{j=0}^{k-1} N^j \right) b \]
Convergence Analysis via Neumann Series

\[ x^{(k)} = b + Nx^{(k-1)} = b + N(b + Nx^{(k-2)}) = b +Nb + N^2x^{(k-2)} = \]
\[ \ldots = b + Nb + N^2b + \cdots + N^{k-1}b + N^kx^{(0)} = \]
\[ = \sum_{j=0}^{k-1} N^j b + N^k x^{(0)} = \left( \sum_{j=0}^{k-1} N^j \right) b + N^k x^{(0)} \]

Special case \( x^{(0)} = 0 \):

\[ x^{(k)} = \left( \sum_{j=0}^{k-1} N^j \right) b \]

\[ \Rightarrow x^{(k)} \in \text{span}\{b, Nb, N^2b, \ldots, N^{k-1}b\} = \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\} = K_k(A, b) \]

which is called the Krylov space to \( A \) and \( b \).
Convergence Analysis via Neumann Series

\[ x^{(k)} = b + Nx^{(k-1)} = b + N(b + Nx^{(k-2)}) = b + Nb + N^2x^{(k-2)} = \]
\[ \vdots = b + Nb + N^2b + \cdots + N^{k-1}b + N^kx^{(0)} = \]
\[ = \sum_{j=0}^{k-1} N^j b + N^kx^{(0)} = \left( \sum_{j=0}^{k-1} N^j \right) b + N^kx^{(0)} \]

Special case \( x^{(0)} = 0 \):

\[ x^{(k)} = \left( \sum_{j=0}^{k-1} N^j \right) b \]

\[ \Rightarrow x^{(k)} \in \text{span}\{b, Nb, N^2b, \ldots, N^{k-1}b\} = \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\} = K_k(A, b) \]

which is called the Krylov space to \( A \) and \( b \).

For \( \|N\| < 1 \) holds:

\[ \sum_{j=0}^{k-1} N^j \rightarrow \sum_{j=0}^{\infty} N^j = (I - N)^{-1} = (I - (I - A))^{-1} = A^{-1} \]
Convergence Analysis via Neumann Series (cont.)

\[ x^{(k)} \to \left( \sum_{j=0}^{\infty} N^j \right) b = (I - N)^{-1} b = A^{-1} b = \bar{x} \]

Richardson iteration is convergent for \( \|N\| < 1 \) or \( A \approx I \).
Convergence Analysis via Neumann Series (cont.)

\[ x^{(k)} \rightarrow \left( \sum_{j=0}^{\infty} N^j \right) b = (I - N)^{-1} b = A^{-1} b = \bar{x} \]

Richardson iteration is convergent for \( \|N\| < 1 \) or \( A \approx I \).

Error analysis for \( e^{(k)} := x^{(k)} - \bar{x} \):

\[
\begin{align*}
  e^{(k+1)} &= x^{(k+1)} - \bar{x} = \Phi(x^{(k)}) - \Phi(\bar{x}) = (b + Nx^{(k)}) - (b + N\bar{x}) = \\
  &= N(x^{(k)} - \bar{x}) = Ne^{(k)}
\end{align*}
\]

\[ \|e^{(k)}\| \leq \|N\| \|e^{(k-1)}\| \leq \|N\|^2 \|e^{(k-2)}\| \leq \cdots \leq \|N\|^k \|e^{(0)}\| \]

\[ \|N\| < 1 \Rightarrow \|N\|^k \xrightarrow{k \to \infty} 0 \Rightarrow \|e^{(k)}\| \xrightarrow{k \to \infty} 0 \]
Convergence Analysis via Neumann Series (cont.)

\[ x^{(k)} \to \left( \sum_{j=0}^{\infty} N^j \right) b = (I - N)^{-1} b = A^{-1} b = \bar{x} \]

Richardson iteration is convergent for \( \|N\| < 1 \) or \( A \approx I \).

Error analysis for \( e^{(k)} := x^{(k)} - \bar{x} \):

\[
e^{(k+1)} = x^{(k+1)} - \bar{x} = \Phi(x^{(k)}) - \Phi(\bar{x}) = (b + Nx^{(k)}) - (b + N\bar{x}) = N(x^{(k)} - \bar{x}) = Ne^{(k)}
\]

\[
\| e^{(k)} \| \leq \| N \| \| e^{(k-1)} \| \leq \| N \|^2 \| e^{(k-2)} \| \leq \cdots \leq \| N \|^k \| e^{(0)} \|
\]

\[
\| N \| < 1 \Rightarrow \| N \|^k \xrightarrow{k \to \infty} 0 \Rightarrow \| e^{(k)} \| \xrightarrow{k \to \infty} 0
\]

- Convergence, if \( \rho(N) = \rho(I - A) < 1 \), where \( \rho \) is spectral radius

\[
\rho(N) = |\lambda_{\text{max}}| = \max_{i}(|\lambda_i|) \quad (\lambda_i \text{ is eigenvalue of } N)
\]

- Eigenvalues of \( A \) have to be all in circle around 1 with radius 1.
Splittings of $A$

- Convergence of Richardson only in very special cases!
  Try to improve the iteration for better convergence!
- Write $A$ in form $A := M - N$

\[ b = Ax = (M - N)x = Mx - Nx \Leftrightarrow x = M^{-1}b + M^{-1}Nx = \Phi(x) \]

\[ \Phi(x) = M^{-1}b + M^{-1}Nx = M^{-1}b + M^{-1}(M - A)x = M^{-1}(b - Ax) + x = x + M^{-1}r(x) \]
Splittings of $A$

- Convergence of Richardson only in very special cases! Try to improve the iteration for better convergence!
- Write $A$ in form $A := M - N$

$$
\begin{align*}
    b &= Ax = (M - N)x = Mx - Nx \iff x = M^{-1}b + M^{-1}Nx = \Phi(x)

    \Phi(x) &= M^{-1}b + M^{-1}Nx = M^{-1}b + M^{-1}(M - A)x = \\
    &= M^{-1}(b - Ax) + x = x + M^{-1}r(x)
\end{align*}
$$

- $N$ should be such that $Ny$ can be evaluated efficiently.
- $M$ should be such that $M^{-1}y$ can be evaluated efficiently.

$$
    x^{(k+1)} = x^{(k)} + M^{-1}r^{(k)}
$$

- Iteration with splitting $M - N$ is equivalent to Richardson on

$$
    M^{-1}Ax = M^{-1}b
$$
Convergence

• Iteration with splitting $A = M - N$ is convergent if

$$\rho(M^{-1}N) = \rho(I - M^{-1}A) < 1$$

• For fast convergence it should hold
  – $M^{-1}A \approx I$
  – $M^{-1}A$ should be better conditioned than $A$ itself
Convergence

• Iteration with splitting $A = M - N$ is convergent if

$$
\rho(M^{-1}N) = \rho(I - M^{-1}A) < 1
$$

• For fast convergence it should hold
  - $M^{-1}A \approx I$
  - $M^{-1}A$ should be better conditioned than $A$ itself

• Such a matrix $M$ is called a preconditioner for $A$.
  Is used in other iterative methods to accelerate convergence.

• Condition number:

$$
\kappa(A) = \|A^{-1}\| \|A\|, \quad \left| \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} \right|, \quad \text{or} \quad \frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}
$$
5.1.2. Jacobi (Diagonal) Splitting

Choose $A = M - N = D - (L + U)$ with $D = \text{diag}(A)$

$L$ the lower triangular part of $A$, and

$U$ the upper triangular part.

\[
\begin{align*}
A &= 
\begin{pmatrix}
D & -U \\
-L & D \\
\end{pmatrix} \\
x^{(k+1)} &= D^{-1}b + D^{-1}(L + U)x^{(k)} = \\
&= D^{-1}b + D^{-1}(D - A)x^{(k)} = x^{(k)} + D^{-1}r^{(k)}
\end{align*}
\]
5.1.3. Jacobi (Diagonal) Splitting

Choose $A = M - N = D - (L + U)$ with

$D = \text{diag}(A)$

$L$ the lower triangular part of $A$, and

$U$ the upper triangular part.

\[
x^{(k+1)} = D^{-1} b + D^{-1} (L + U) x^{(k)} = D^{-1} b + D^{-1} (D - A) x^{(k)} = x^{(k)} + D^{-1} r^{(k)}
\]

Convergent for $A \approx \text{diag}(A)$ or diagonal dominant matrices:

$\rho(D^{-1}N) = \rho(I - D^{-1}A) < 1$
Jacobi (Diagonal) Splitting (cont.)

Iteration process written elementwise:

\[ x^{(k+1)} = D^{-1} (b - (A - D)x^{(k)}) \Rightarrow x_j^{(k+1)} = \frac{1}{a_{jj}} \left( b_j - \sum_{m=1, m \neq j}^{n} a_{j,m}x_m^{(k)} \right) \]

\[ a_{jj}x_j^{(k+1)} = b_j - \sum_{m=1}^{j-1} a_{j,m}x_m^{(k)} - \sum_{m=j+1}^{n} a_{j,m}x_m^{(k)} \]
Jacobi (Diagonal) Splitting (cont.)

Iteration process written elementwise:

\[ x^{(k+1)} = D^{-1} (b - (A - D)x^{(k)}) \Rightarrow x_j^{(k+1)} = \frac{1}{a_{jj}} \left( b_j - \sum_{m=1, m \neq j}^{n} a_{j,m}x_m^{(k)} \right) \]

\[ a_{jj}x_j^{(k+1)} = b_j - \sum_{m=1}^{j-1} a_{j,m}x_m^{(k)} - \sum_{m=j+1}^{n} a_{j,m}x_m^{(k)} \]

- Damping or relaxation for improving convergence
- Idea: Iterative method as correction of last iterate in search direction.
Jacobi (Diagonal) Splitting (cont.)

Iteration process written elementwise:

\[ x^{(k+1)} = D^{-1} (b - (A - D)x^{(k)}) \Rightarrow x_j^{(k+1)} = \frac{1}{a_{jj}} \left( b_j - \sum_{m=1, m \neq j}^{n} a_{jm}x_m^{(k)} \right) \]

\[ a_{jj}x_j^{(k+1)} = b_j - \sum_{m=1}^{j-1} a_{jm}x_m^{(k)} - \sum_{m=j+1}^{n} a_{jm}x_m^{(k)} \]

- Damping or relaxation for improving convergence
- Idea: Iterative method as correction of last iterate in search direction.
- Introduce step length for this correction step:

\[ x^{(k+1)} = x^{(k)} + D^{-1} r^{(k)} \quad \rightarrow \quad x^{(k+1)} = x^{(k)} + \omega D^{-1} r^{(k)} \]

with additional damping parameter \( \omega \).

- Damped Jacobi iteration:

\[ x^{(k+1)}_{\text{damped}} = (\omega + 1 - \omega)x^{(k)} + \omega D^{-1} r^{(k)} = \omega x^{(k+1)} + (1 - \omega)x^{(k)} \]
**Damped Jacobi Iteration**

\[
\begin{align*}
    x^{(k+1)} & = x^{(k)} + \omega D^{-1} r^{(k)} = x^{(k)} + \omega D^{-1} (b - Ax^{(k)}) = \\
    & = \ldots \\
    & = \omega D^{-1} b + [(1 - \omega)I + \omega D^{-1} (L + U)] x^{(k)}
\end{align*}
\]

is convergent for

\[
\rho([[(1 - \omega)I + \omega D^{-1} (L + U)]]) < 1
\]

\[
\omega \rightarrow 0, \quad I
\]

Look for optimal \( \omega \) with best convergence (add. degree of freedom).
Parallelism in the Jacobi Iteration

- Jacobi method is easy to parallelize: only $Ax$ and $D^{-1}x$.
- But often too slow convergence!
- Improvement: block Jacobi iteration

![Matrix Diagram]
5.1.4. Gauss-Seidel Iteration

Always use newest information available!

Jacobi iteration:

\[ a_{jj}x_j^{(k+1)} = b_j - \sum_{m=1}^{j-1} a_{j,m}x_m^{(k)} - \sum_{m=j+1}^{n} a_{j,m}x_m^{(k)} \]

Gauss-Seidel iteration:

\[ a_{jj}x_j^{(k+1)} = b_j - \sum_{m=1}^{j-1} a_{j,m}x_m^{(k+1)} - \sum_{m=j+1}^{n} a_{j,m}x_m^{(k)} \]
Gauss-Seidel Iteration (cont.)

- Compare dependency graphs for general iterative algorithms. Here:

\[ x = f(x) = D^{-1}(b + (D - A)x) = D^{-1}(b - (L + U)x) \]

to splitting \( A = (D - L) - U = M - N \)

\[
\begin{align*}
    x^{(k+1)} &= (D - L)^{-1} b + (D - L)^{-1} U x^{(k)} = \\
              &= (D - L)^{-1} b + (D - L)^{-1}(D - L - A) x^{(k)} = \\
              &= x^{(k)} + (D - L)^{-1} r^{(k)}
\end{align*}
\]

- Convergence depends on spectral radius \( \rho(I - (D - L)^{-1} A) < 1 \)
Parallelism in the Gauss-Seidel Iteration

- Linear system in $D - L$ is easy to solve because $D - L$ is lower triangular but strongly sequential!
- Use red-black ordering or graph colouring for compromise: parallel $\leftrightarrow$ convergence
Successive Over Relaxation (SOR)

- Damping or relaxation:
  \[ x^{(k+1)} = x^{(k)} + \omega(D-L)^{-1} r^{(k)} = \omega(D-L)^{-1} b + [(1-\omega)+\omega(D-L)^{-1} U] x^{(k)} \]

- Convergence depends on spectral radius of iteration matrix
  \[ (1 - \omega) + \omega(D - L)^{-1} U \]

- Parallelization of SOR == parallelization of GS
Stationary Methods (in General)

- Can always be written in the two normal forms
  \[ x^{(k+1)} = c + Bx^{(k)} \]
  with convergence depending on \( \rho(B) \) of iteration matrix and
  \[ x^{(k+1)} = x^{(k)} + Fr^{(k)} \]
  with preconditioner \( F, B = I - FA \)
Stationary Methods (in General)

- Can always be written in the two normal forms
  \[ x^{(k+1)} = c + Bx^{(k)} \]
  with convergence depending on \( \rho(B) \) of iteration matrix and
  \[ x^{(k+1)} = x^{(k)} + Fr^{(k)} \]
  with preconditioner \( F \), \( B = I - FA \)

- For \( x^{(0)} = 0 \):
  \[ x^{(k+1)} \subseteq K_k(B, c) \]
  which is the Krylov space with respect to matrix \( B \) and vector \( c \).

- Slow convergence (but good smoothing properties! \( \rightarrow \) multigrid)
MATLAB Example

- clear; n=100; k=10; omega=1; stationary
- tridiag(-.5, 1, -.5):
  - Jacobi norm $|\cos(\pi/n)|$
  - GS norm $\cos(\pi/n)^2$
  - both $< 1 \rightarrow$ convergence, but slow
- To improve convergence $\rightarrow$ nonstationary methods (or multigrid)
Chair of Informatics V—SCCS

Efficient Numerical Algorithms—Parallel & HPC

- High-dimensional numerics (sparse grids)
- Fast iterative solvers (multi-level methods, preconditioners, eigenvalue solvers)
- Uncertainty Quantification
- Space-filling curves
- Numerical linear algebra
- Numerical algorithms for image processing
- HW-aware numerical programming

Fields of application in simulation

- CFD (incl. fluid-structure interaction)
- Plasma physics
- Molecular dynamics
- Quantum chemistry

Further info → www5.in.tum.de
Feel free to come around and ask for thesis topics!
5.2. Nonstationary Methods

5.2.1. Gradient Method

- Consider $A = A^T > 0$ (A SPD)

  Function $\Phi(x) = \frac{1}{2} x^T A x - b^T x$

- n-dim. paraboloid $\mathbb{R}^n \rightarrow \mathbb{R}$

- Gradient $\nabla \Phi(x) = A x - b$

- Position with $\nabla \Phi(x) = 0$ is exactly minimum of paraboloid
5.3. Nonstationary Methods

5.3.1. Gradient Method

- Consider \( A = A^T > 0 \) (A SPD)

\[
\Phi(x) = \frac{1}{2} x^T A x - b^T x
\]

- n-dim. paraboloid \( \mathbb{R}^n \rightarrow \mathbb{R} \)
- Gradient \( \nabla \Phi(x) = Ax - b \)
- Position with \( \nabla \Phi(x) = 0 \) is exactly minimum of paraboloid
- Instead of solving \( Ax = b \) consider \( \min_x \Phi(x) \)
- Local descent direction in \( y \):
  \( \nabla \Phi(x) \cdot y \) is minimum for

\[
y = -\nabla \Phi(x)
\]
Gradient Method (cont.)

- Optimization: start with $x^{(0)}$
  
  $$x^{(k+1)} := x^{(k)} + \alpha_k d^{(k)}$$

  with search direction $d^{(k)}$ and step size $\alpha_k$.

- In view of previous results the optimal (local) search direction is
  
  $$-\nabla \Phi(x^{(k)}) =: d^{(k)}$$
Gradient Method (cont.)

- Optimization: start with $x^{(0)}$

$$x^{(k+1)} := x^{(k)} + \alpha_k d^{(k)}$$

with search direction $d^{(k)}$ and step size $\alpha_k$.

- In view of previous results the optimal (local) search direction is

$$-\nabla \Phi(x^{(k)}) =: d^{(k)}$$

- To define $\alpha_k$:

$$\min_{\alpha} g(\alpha) := \min_{\alpha} \Phi(x^{(k)} + \alpha(b - Ax^{(k)}))$$

$$= \min_{\alpha} \left( \frac{1}{2} (x^{(k)} + \alpha d^{(k)})^T A(x^{(k)} + \alpha d^{(k)}) - b^T (x^{(k)} + \alpha d^{(k)}) \right)$$

$$= \min_{\alpha} \left( \frac{1}{2} \alpha^2 d^{(k)^T} A d^{(k)} - \alpha d^{(k)^T} d^{(k)} + \frac{1}{2} x^{(k)^T} A x^{(k)} - x^{(k)^T} b \right)$$

$$\alpha_k = \frac{d^{(k)^T} d^{(k)}}{d^{(k)^T} A d^{(k)}}$$

$$d^{(k)} = -\nabla \Phi(x^{(k)}) = b - A x^{(k)}$$
Gradient Method (cont. 2)

\[ x^{(k+1)} = x^{(k)} + \frac{\| b - Ax^{(k)} \|^2_2}{(b - Ax^{(k)})^T A (b - Ax^{(k)})} (b - Ax^{(k)}) \]

- Method of steepest descent.
Gradient Method (cont. 2)

\[ x^{(k+1)} = x^{(k)} + \frac{\| b - Ax^{(k)} \|^2_2}{(b - Ax^{(k)})^T A(b - Ax^{(k)})} (b - Ax^{(k)}) \]

- Method of steepest descent.
- Disadvantage: Distorted contour lines.
- Slow convergence (zig zag path)
- Local descent direction is not globally optimal
Analysis of the Gradient Method

• Definition $A$-norm:

$$\|x\|_A := \sqrt{x^T Ax}$$

• Consider error:

$$\|x - \bar{x}\|^2_A = \|x - A^{-1}b\|^2_A = (x^T - b^T A^{-1})A(x - A^{-1}b)$$

$$= x^T Ax - 2b^T x + b^T A^{-1}b$$

$$= 2\Phi(x) + b^T A^{-1}b$$
Analysis of the Gradient Method

- Definition $A$-norm:
  \[ \|x\|_A := \sqrt{x^T A x} \]

- Consider error:
  \[
  \|x - \bar{x}\|^2_A = \|x - A^{-1} b\|^2_A = (x^T - b^T A^{-1}) A (x - A^{-1} b) \\
  = x^T A x - 2 b^T x + b^T A^{-1} b \\
  = 2 \Phi(x) + b^T A^{-1} b
  \]

- Therefore, minimizing $\Phi$ is equivalent to minimizing the error in the $A$-norm! More detailed analysis reveals:
  \[
  \|x^{(k+1)} - \bar{x}\|^2_A \leq \left(1 - \frac{1}{\kappa(A)}\right) \cdot \|x^{(k)} - \bar{x}\|^2_A
  \]

- Therefore, for $\kappa(A) \gg 1$ very slow convergence!
5.3.2. The Conjugate Gradient Method

- Improving descent direction being globally optimal.

- $x^{(k+1)} := x^{(k)} + \alpha_k p^{(k)}$ with search direction not being negative gradient, but projection of gradient that is $A$-conjugate to all previous search directions:

  $$ p^{(k)} \perp A p^{(j)} \quad \text{for all } j < k \quad \text{or} $$
  $$ p^{(k)} \perp_A p^{(j)} \quad \text{or} $$
  $$ p^{(k)^T} A p^{(j)} = 0 \quad \text{for } j < k $$
5.3.3. The Conjugate Gradient Method

- Improving descent direction being globally optimal.

- $x^{(k+1)} := x^{(k)} + \alpha_k p^{(k)}$ with search direction not being negative gradient, but projection of gradient that is $A$-conjugate to all previous search directions:

$$p^{(k)} \perp Ap^{(j)} \quad \text{for all } j < k \quad \text{or}$$

$$p^{(k)} \perp_A p^{(j)} \quad \text{or}$$

$$p^{(k)^T} Ap^{(j)} = 0 \quad \text{for } j < k$$

- We choose new search direction as component of last residual that is $A$-conjugate to all previous search directions.

- $\alpha_k$ again by 1-dim. minimization as before (for chosen $p^{(k)}$)
The Conjugate Gradient Algorithm

\[ p^{(0)} = r^{(0)} = b - Ax^{(0)} \]

for \( k = 1, 2, \ldots \) do

\[ \alpha^{(k)} = -\frac{\langle r^{(k)}, r^{(k)} \rangle}{\langle p^{(k)}, Ap^{(k)} \rangle} \]

\[ x^{(k+1)} = x^{(k)} - \alpha^{(k)} p^{(k)} \]

\[ r^{(k+1)} = r^{(k)} + \alpha^{(k)} Ap^{(k)} \]

if \( \| r^{(k+1)} \|^2 \leq \epsilon \) then break

\[ \beta^{(k)} = \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k))} \rangle} \]

\[ p^{(k+1)} = r^{(k+1)} + \beta^{(k)} p^{(k)} \]
Properties of Conjugate Gradients

- It holds
  \[ p^{(j)^T} A p^{(k)} = 0 = r^{(j)^T} r^{(k)} \text{ for } j \neq k \]

- and
  \[ \text{span}(p^{(1)}, \ldots, p^{(j)}) = \text{span}(r^{(0)}, \ldots, r^{(j-1)}) = \text{span}(r^{(0)}, Ar^{(0)}, \ldots, A^{j-1} r^{(0)}) = K_j(A, r^{(0)}) \]

- Especially for \( x^{(0)} = 0 \) it holds
  \[ K_j(A, r^{(0)}) = \text{span}(b, Ab, \ldots, A^{j-1} b) \]
Properties of Conjugate Gradients

- It holds
  \[ p^{(j)^T} A p^{(k)} = 0 = r^{(j)^T} r^{(k)} \text{ for } j \neq k \]

- and
  \[ \text{span}(p^{(1)}, \ldots, p^{(j)}) = \text{span}(r^{(0)}, \ldots, r^{(j-1)}) = \text{span}(r^{(0)}, Ar^{(0)}, \ldots, A^{j-1}r^{(0)}) = K_j(A, r^{(0)}) \]

- Especially for \( x^{(0)} = 0 \) it holds
  \[ K_j(A, r^{(0)}) = \text{span}(b, Ab, \ldots, A^{j-1}b) \]

- \( x^{(k)} \) is best approximate solution to \( Ax = b \) in subspace \( K_k(A, r^{(0)}) \). For \( x^{(0)} = 0 \) : \( x^{(k)} \in K_k(A, b) \)

- Error:
  \[ \| x^{(k)} - \bar{x} \|_A = \min_{x \in K_k(A, b)} \| x - \bar{x} \|_A \]

- Cheap 1-dim. minimization gives optimal \( k \)-dim. solution for free!
Properties of Conjugate Gradients (cont.)

- Consequence: After \( n \) steps \( K_n(A, b) = \mathbb{R}^n \) and therefore \( x^{(n)} = A^{-1}b \) is solution in exact arithmetic.

- Unfortunately, this is not true in floating point arithmetic.

- Furthermore, \( \mathcal{O}(n) \) iteration steps would be too costly:
  costs: \#iterations \(*\) matrix-vector-product
Properties of Conjugate Gradients (cont.)

• Consequence: After \( n \) steps \( K_n(A, b) = \mathbb{R}^n \) and therefore \( x^{(n)} = A^{-1}b \) is solution in exact arithmetic.

• Unfortunately, this is not true in floating point arithmetic.

• Furthermore, \( \mathcal{O}(n) \) iteration steps would be too costly:
  costs: \( \# \text{iterations} \times \text{matrix-vector-product} \)

• Matrix-vector-product easy in parallel.

• But, how to get fast convergence and reduce \#iterations?
Error Estimation \((x^{(0)} = 0)\)

\[
\|e^{(k)}\|_A = \|x^{(k)} - \bar{x}\|_A = \min_{x \in K_k(A,b)} \|x - \bar{x}\|_A = \\
= \min_{\alpha_0,\ldots,\alpha_{k-1}} \left\| \sum_{j=0}^{k-1} \alpha_j (A^j b) - \bar{x} \right\|_A = \\
= \min_{P^{(k-1)}(x)} \left\| P^{(k-1)}(A)b - \bar{x} \right\|_A = \\
= \min_{P^{(k-1)}(x)} \left\| P^{(k-1)}(A)A\bar{x} - \bar{x} \right\|_A = \\
= \min_{P^{(k-1)}(x)} \left\| (P^{(k-1)}(A)A - I)(\bar{x} - x^{(0)}) \right\|_A = \\
= \min_{Q^{(k)}(x), Q^{(k)}(0) = 1} \left\| Q^{(k)}(A)e^{(0)} \right\|_A
\]

for polynomial \(Q^{(k)}(x)\) of degree \(k\) with \(Q^{(k)}(0) = 1\).
Error Estimation

- Matrix $A$ has orthonormal basis of eigenvectors $u_j$, $j = 1, \ldots, n$, eigenvalues $\lambda_j$

- It holds
  \[
  Au_j = \lambda_j u_j, \quad j = 1, \ldots, n, \quad \text{and} \quad u_j^T u_k = 0 \quad \text{for} \quad j \neq k \quad \text{and} \quad 1 \quad \text{for} \quad j = k
  \]

- Start error in ONB: $e^{(0)} = \sum_{j=1}^{n} \rho_j u_j$
Error Estimation

- Matrix $A$ has orthonormal basis of eigenvectors $u_j$, $j = 1, \ldots, n$, eigenvalues $\lambda_j$
- It holds
  
  $$Au_j = \lambda_j u_j, \quad j = 1, \ldots, n, \text{ and } u_j^T u_k = 0 \text{ for } j \neq k \text{ and } 1 \text{ for } j = k$$
- Start error in ONB: $e^{(0)} = \sum_{j=1}^{n} \rho_j u_j$

\[
\|e^{(k)}\|_A = \min_{Q^{(k)}(0)=1} \left\| Q^{(k)}(A) \sum_{j=1}^{n} \rho_j u_j \right\|_A = \min_{Q^{(k)}(0)=1} \left\| \sum_{j=1}^{n} \rho_j Q^{(k)}(A) u_j \right\|_A = \\
= \min_{Q^{(k)}(0)=1} \left\| \sum_{j=1}^{n} \rho_j Q^{(k)}(\lambda_j) u_j \right\|_A \leq \min_{Q^{(k)}(0)=1} \left\{ \max_{j=1,\ldots,n} |Q^{(k)}(\lambda_j)| \right\} \left\| \sum_{j=1}^{n} \rho_j u_j \right\|_A = \\
= \min_{Q^{(k)}(0)=1} \left\{ \max_{j=1,\ldots,n} |Q^{(k)}(\lambda_j)| \right\} \|e^{(0)}\|_A
\]
Error Estimates

By choosing polynomials with \( Q^{(k)}(0) = 1 \), we can derive error estimates for the error after the \( k \)th step:

Choose, e.g. \( Q^{(k)}(x) := \left| 1 - \frac{2}{\lambda_{\text{max}} + \lambda_{\text{min}}} x \right|^k \)
Error Estimates

By choosing polynomials with $Q^{(k)}(0) = 1$, we can derive error estimates for the error after the $k$th step:

Choose, e.g. $Q^{(k)}(x) := \left| 1 - \frac{2}{\lambda_{\max} + \lambda_{\min}} x \right|^k$

$$\| e^{(k)} \|_A \leq \max_{j=1,...,n} |Q^{(k)}(\lambda_j)| \| e^{(0)} \|_A = \left| 1 - \frac{2\lambda_{\max}}{\lambda_{\max} + \lambda_{\min}} \right|^k \| e^{(0)} \|_A$$

$$= \left( \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \right)^k \| e^{(0)} \|_A = \left( \frac{\kappa(A) - 1}{\kappa(A) + 1} \right)^k \| e^{(0)} \|_A$$
Better Estimates

- Choose normalized Chebyshev polynomials
  \[ T_n(x) = \cos(n \arccos(x)) \]

\[
\| e^{(k)} \|_A \leq \frac{1}{T_k \left( \frac{\kappa(A) + 1}{\kappa(A) - 1} \right) \| e^{(0)} \|_A} \leq 2 \left( \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \| e^{(0)} \|_A
\]
Better Estimates

- Choose normalized Chebyshev polynomials
  \[ T_n(x) = \cos(n \arccos(x)) \]

  \[ \| e^{(k)} \|_A \leq \frac{1}{T_k \left( \frac{\kappa(A) + 1}{\kappa(A) - 1} \right)} \| e^{(0)} \|_A \leq 2 \left( \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \| e^{(0)} \|_A \]

- For clustered eigenvalues choose special polynomial, e.g. assume that \( A \) has only two eigenvalues \( \lambda_1 \) and \( \lambda_2 \):

  \[ Q^{(2)}(x) := \frac{(\lambda_1 - x)(\lambda_2 - x)}{\lambda_1 \lambda_2} \]

  \[ \| e^{(2)} \|_A \leq \max_{j=1,2} \left| Q^{(2)}(\lambda_j) \right| \| e^{(0)} \|_A = 0 \]

  Convergence of CG after two steps!
Outliers/Cluster

Assume the matrix has an eigenvalue $\lambda_1 > 1$ and all other eigenvalues are contained in an $\epsilon$-neighborhood of 1:

$$\forall \lambda \neq \lambda_1 : |\lambda - 1| < \epsilon$$

$$Q^{(2)}(x) := \frac{(\lambda_1 - x)(1 - x)}{\lambda_1}$$

$$\| e^{(2)} \|_A \leq \max_{|\lambda - 1| < \epsilon} \left| \frac{(\lambda_1 - \lambda)(1 - \lambda)}{\lambda_1} \right| \| e^{(0)} \|_A \leq \frac{(\lambda_1 - 1 + \epsilon)\epsilon}{\lambda_1} = O(\epsilon)$$

Very good approximation of CG after only two steps!

Important: small number of outliers combined with cluster.
Conjugate Gradients Summary

- To get fast convergence and reduce the number of iterations: find preconditioner $M$, such that $M^{-1}Ax = M^{-1}b$ with clustered eigenvalues.

- Conjugate gradients (CG) is always the method of choice for symmetric positive definite $A$ (in general).

- To improve convergence, include preconditioning (PCG).

- CG has two important properties: optimal and cheap.
while (sqrt(gamma) > epsilon * error_0) {
    if (iteration > 1)
        q = r + gamma / gamma_old * q;
    v = A * q;
    delta = dot(v, q);
    alpha = delta / gamma;
    x = x + alpha * q;
    r = r - alpha * v;
    gamma_old = gamma;
    gamma = dot(r, r);
    iteration = iteration + 1;
}

dot product: requires communication (MPI_ALLREDUCE)
Non-Blocking Collective Operations

- Use to hide communication!
- Allows overlap of numerical computations and communications.
- In MPI-1/MPI-2 only possible for point-to-point communication: MPI_Isend and MPI_Irecv.
  
  Additional libraries necessary for collective operations!
  Example: LibNBC (non-blocking collectives)

- Are included in new MPI-3 standard.
Example: Pseudocode for Nonblock. Reduct.

```c
MPI_Request req;
int sbuf1[SIZE], rbuf1[SIZE], buf2[SIZE];
// compute sbuf1
compute(sbuf1, SIZE);

// start non-blocking allreduce of subf1
// computation and communication overlap
MPI_Iallreduce(sbuf1, rbuf1, SIZE, MPI_INT, MPI_SUM, MPI_COMM, &req);

// compute buf2 (independent of buf1)
compute(buf2, SIZE);

// synchronization
MPI_WAIT(&req, &stat);

// use data in rbuf1; final computation
evaluate(rbuf1, buf2, SIZE);
```
Iter. Methods for general (nonsymmetric) $A$: BiCGSTAB

- Applicable if little memory at hand and $A^T$ not available.
- Computational costs per iteration similar to BiCG and CGS.
- Alternative for CGS that avoids irregular convergence patterns of CGS maintaining similar convergence speed.
- Less loss of accuracy in the updated residual.
Iter. Methods for general (nonsymmetric) $A$: GMRES

- Leads to smallest residual for fixed number of iteration steps, but these steps become increasingly expensive.
- To limit increasing storage requirements and work per iteration step, restarting is necessary. When to do so depends on $A$ and $b$; it requires skill and experience.
Iter. Methods for general (nonsymmetric) $A$: GMRES

- Leads to smallest residual for fixed number of iteration steps, but these steps become increasingly expensive.
- To limit increasing storage requirements and work per iteration step, restarting is necessary. When to do so depends on $A$ and $b$; it requires skill and experience.
- Requires only matrix-vector products with the coeff. matrix.
- Number of inner products grows linearly with iteration number (up to restart point).
- Implementation based on Gram-Schmidt $\rightarrow$ inner products independent $\rightarrow$ only one synchronization point. Using modified Gram-Schmidt $\rightarrow$ one synchronization point per inner product.

We consider GMRES in the following.
5.3.4. GMRES

- Iterative solution method for general $A$

- Consider small subspace $U_m$ and determine optimal approximate solution for $Ax = b$ in $U_m$. Hence $x$ is of the form $x := U_m y$

$$\min_{x \in U_m} \|Ax - b\|_2 = \min_y \|A(U_m y) - b\|_2$$

- Can be solved by normal equations: $(U_m^T A^T A U_m) y = U_m^T A^T b$
5.3.5. GMRES

- Iterative solution method for general $A$

- Consider small subspace $U_m$ and determine optimal approximate solution for $Ax = b$ in $U_m$. Hence $x$ is of the form $x := U_m y$

$$\min_{x \in U_m} \|Ax - b\|_2 = \min_y \|A(U_m y) - b\|_2$$

- Can be solved by normal equations: $(U_m^T A^T AU_m)y = U_m^T A^T b$

- Try to find sequence of "good" subspaces $U_1 \rightarrow U_2 \rightarrow U_3 \rightarrow \ldots$ such that iteratively we can update the optimal solutions

$$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow \ldots \rightarrow A^{-1} b$$

using mainly matrix-vector products.
GMRES: Subspace

What subspace for fast convergence and easy computations?
GMRES: Subspace

What subspace for fast convergence and easy computations?

\[ U_m := K_m(A, b) = \text{span}(b, Ab, \ldots, A^{m-1}b) \quad (\text{Krylov space}) \]

Problem: \( b, Ab, A^2b, \ldots \) is bad basis for this subspace!

So we need a first step to compute a good basis for \( U_m \):
GMRES: Subspace

What subspace for fast convergence and easy computations?

\[ U_m := K_m(A, b) = \text{span}(b, Ab, \ldots, A^{m-1}b) \]  \hspace{1cm} (Krylov space)

Problem: \( b, Ab, A^2b, \ldots \) is bad basis for this subspace!

So we need a first step to compute a good basis for \( U_m \):

Start with \( u_1 := \frac{b}{\|b\|_2} \) and do for \( j = 2 : m \):

\[
\tilde{u}_j := Au_{j-1} - \sum_{k=1}^{j-1} \left( u_k^T A u_{j-1} \right) u_k = Au_{j-1} - \sum_{k=1}^{j-1} h_{k,j-1} u_k
\]

\[
u_j := \frac{\tilde{u}_j}{\|\tilde{u}_j\|_2} = \frac{\tilde{u}_j}{h_{j,j-1}}
\]

which is the standard orthogonalization method (Arnoldi method)
Matrix Form of Arnoldi ONB

\[ U_m = \text{span}(b, Ab, \ldots, A^{m-1}b) = \text{span}(u_1, u_2, \ldots, u_m) \] (ONB)

Write this orthogonalization method in matrix form

\[ Au_{j-1} = \sum_{k=1}^{j-1} h_{k,j-1} u_k + \tilde{u}_j = \sum_{k=1}^{j} h_{k,j-1} u_k \]
Matrix Form of Arnoldi ONB

\[ U_m = \text{span}(b, Ab, \ldots, A^{m-1}b) = \text{span}(u_1, u_2, \ldots, u_m) \quad \text{(ONB)} \]

Write this orthogonalization method in matrix form

\[ Au_{j-1} = \sum_{k=1}^{j-1} h_{k,j-1} u_k + \tilde{u}_j = \sum_{k=1}^{j} h_{k,j-1} u_k \]

\[ AU_m = A(u_1 \ldots u_m) = (u_1 \ldots u_{m+1}) \tilde{H}_{m+1,m} = U_{m+1} \tilde{H}_{m+1,m} \]

\[ \tilde{H}_{m+1,m} = \begin{pmatrix} h_{11} & \cdots & \cdots & h_{1m} \\ h_{21} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & & \ddots & h_{mm} \\ 0 & 0 & & h_{m+1,m} \end{pmatrix} \]
**GMRES: Minimization**

This leads to minimization problem

\[
\min_{x \in U_m} \|Ax - b\| = \min_y \|AU_m y - b\|
\]

\[
= \min_y \left\| U_{m+1} \tilde{H}_{m+1,m} y - \|b\| u_1 \right\|
\]

\[
= \min_y \left\| U_{m+1} \left( \tilde{H}_{m+1,m} y - \|b\| e_1 \right) \right\|
\]

\[
= \min_y \left\| \tilde{H}_{m+1,m} y - \|b\| e_1 \right\|
\]

Because \(U_{m+1}\) is part of an orthogonal matrix.