Scientific Computing II

Conjugate Gradient Methods

Michael Bader

Summer term 2012

Part I: Quadratic Forms and Steepest Descent

Part II: Conjugate Gradients

4. Conjugate Directions
5. *A*-Orthogonality
6. Conjugate Gradients
7. A Miracle Occurs . . .
8. CG Algorithm

Part III: Preconditioning

9 CG Convergence

10 Preconditioning

11 CG with "Change-of-Basis" Preconditioning

12 CG with Matrix Preconditioner

13 Preconditioners – Examples

Families of Iterative Solvers

- **relaxation methods**:
  - Jacobi-, Gauss-Seidel-Relaxation, ...
  - Over-Relaxation-Methods
- **Krylov methods**:
  - Steepest Descent, Conjugate Gradient, ...
  - GMRES, ...
- **Multilevel/Multigrid methods**,
  Domain Decomposition, ...

Remember: The Residual Equation

- for $Ax = b$, we defined the **residual** as:

$$r^{(i)} = b - Ax^{(i)}$$

- and the error: $e^{(i)} := x - x^{(i)}$
- leads to the **residual equation:**

$$Ae^{(i)} = r^{(i)}$$

- relaxation methods: solve a modified (easier) SLE:

$$B\,\hat{e}^{(i)} = r^{(i)} \qquad \text{where} \quad B \sim A$$

- multigrid methods: coarse-grid correction on residual equation

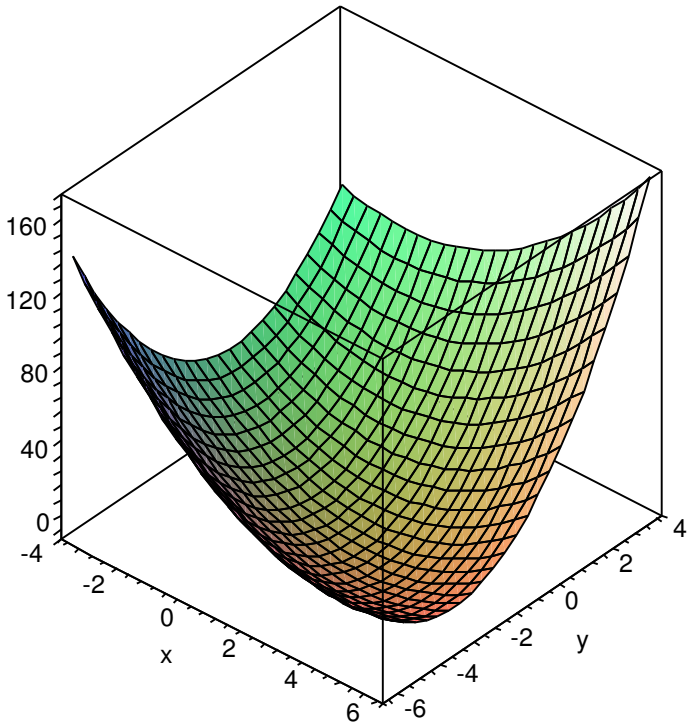
Part I

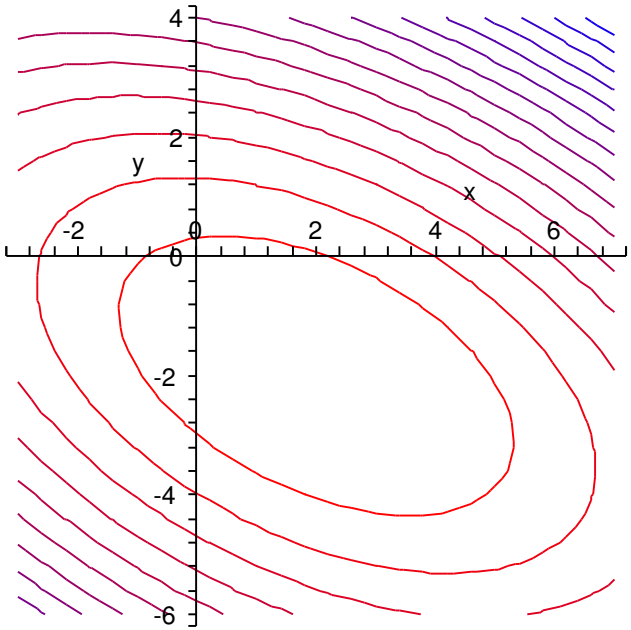
Quadratic Forms and Steepest Descent

Quadratic Forms

A *quadratic form* is a scalar, quadratic function of a vector of the form:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c, \qquad \text{where } A = A^T$$





Quadratic Forms (2)

The *gradient* of a quadratic form is defined as

$$f'(x) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(x) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x) \end{pmatrix}$$

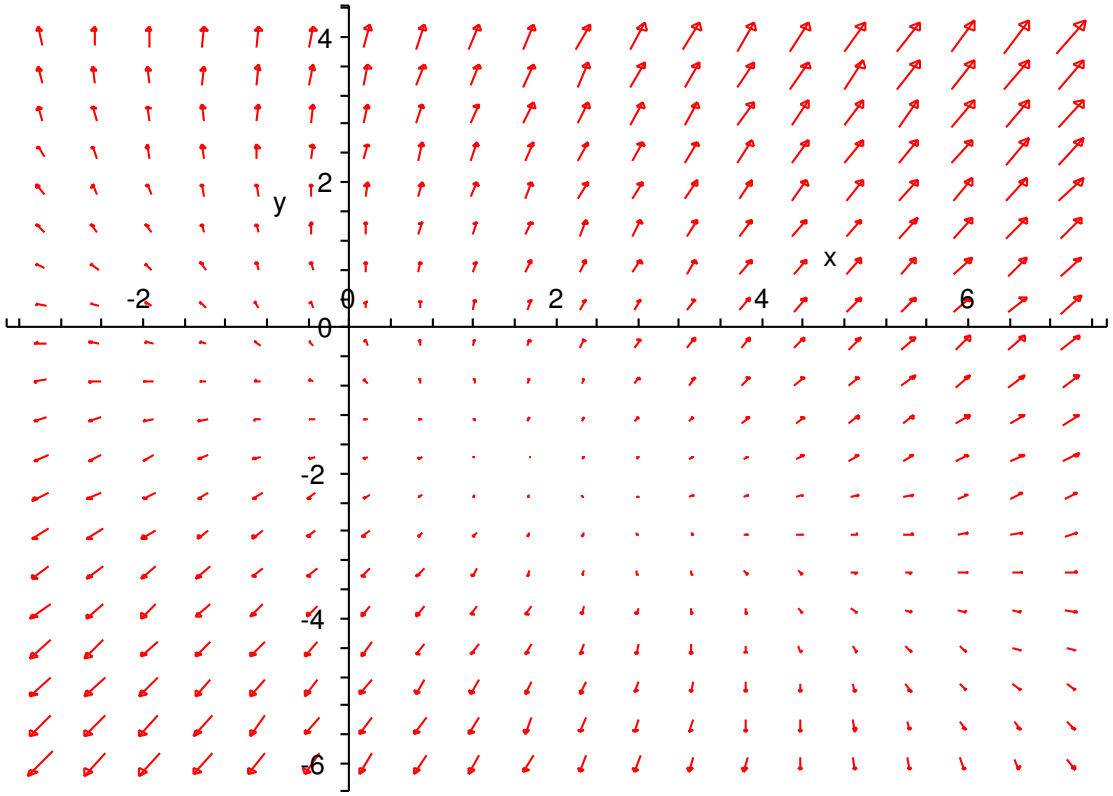
- $f'(x) = Ax - b$
- $f'(x) = 0 \quad \Leftrightarrow \quad Ax - b = 0 \quad \Leftrightarrow \quad Ax = b$

$\Rightarrow Ax = b$ equivalent to a **minimisation problem**

$\Rightarrow$ proper minimum **only if** A **positive definite**

Direction of Steepest Descent

- gradient $f'(x)$: direction of "steepest ascent"
- $f'(x) = Ax - b = -r$ (with residual $r = b - Ax$)
- residual r: direction of "steepest descent"



Solving SLE via Minimum Search

- basic idea to find minimum:
  move into direction of steepest descent
- most simple scheme:

$$x^{(i+1)} = x^{(i)} + \alpha r^{(i)}$$

- α constant $\Rightarrow$ **Richardson** iteration
  (often considered as a relaxation method)
- better choice of α:
  move to lowest point in that direction
  $\Rightarrow$ **Steepest Descent**

Steepest Descent – find an optimal α

- task: *line search* along the line $x^{(1)} = x^{(0)} + \alpha r^{(0)}$
- choose α such that $f(x^{(1)})$ is minimal:

$$\frac{\partial}{\partial \alpha} f(x^{(1)}) = 0$$

- use chain rule:

$$\frac{\partial}{\partial \alpha} f(x^{(1)}) = f'(x^{(1)})^T \frac{\partial}{\partial \alpha} x^{(1)} = f'(x^{(1)})^T r^{(0)}$$

- remember $f'(x^{(1)}) = -r^{(1)}$, thus:

$$-\left(r^{(1)}\right)^T r^{(0)} \stackrel{!}{=} 0$$

Steepest Descent – find α (2)

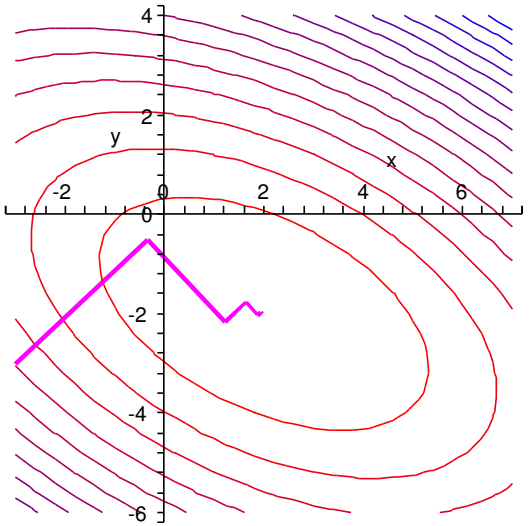
$$\begin{aligned}
\left(r^{(1)}\right)^T r^{(0)} = \left(b - Ax^{(1)}\right)^T r^{(0)} &= 0 \\
\left(b - A(x^{(0)} + \alpha r^{(0)})\right)^T r^{(0)} &= 0 \\
\left(b - Ax^{(0)}\right)^T r^{(0)} - \alpha \left(Ar^{(0)}\right)^T r^{(0)} &= 0 \\
\left(r^{(0)}\right)^T r^{(0)} - \alpha \left(r^{(0)}\right)^T Ar^{(0)} &= 0
\end{aligned}$$

Solve for α:

$$\alpha = \frac{\left(r^{(0)}\right)^T r^{(0)}}{\left(r^{(0)}\right)^T Ar^{(0)}}$$

Steepest Descent – Algorithm

1. $r^{(i)} = b - Ax^{(i)}$
2. $\alpha_i = \dfrac{\left(r^{(i)}\right)^T r^{(i)}}{\left(r^{(i)}\right)^T Ar^{(i)}}$
3. $x^{(i+1)} = x^{(i)} + \alpha_i r^{(i)}$



Observations:

- slow convergence (sim. to Jacobi relaxation)
- $\left\|e^{(i)}\right\|_A \leq \left(\frac{\kappa-1}{\kappa+1}\right)^i \left\|e^{(0)}\right\|_A$
- for positive definite A: $\kappa = \lambda_{\text{max}}/\lambda_{\text{min}}$ (largest/smallest eigenvalues of A)
- many steps in the same direction

Part II

Conjugate Gradients

Conjugate Directions

- observation: Steepest Descent takes repeated steps in the same direction
- obvious idea:
  try to do only one step in each direction
- possible approach:
  choose orthogonal search directions
  $d^{(0)} \perp d^{(1)} \perp d^{(2)} \perp \ldots$
- notice: errors orthogonal to previous directions:
  $e^{(1)} \perp d^{(0)}, e^{(2)} \perp d^{(1)} \perp d^{(0)}, \ldots$

Conjugate Directions (2)

- compute α from

$$\left(d^{(0)}\right)^T e^{(1)} = \left(d^{(0)}\right)^T \left(e^{(0)} - \alpha d^{(0)}\right) = 0$$

requires propagation of the error $e^{(1)} = x - x^{(1)}$

$$\begin{aligned} x^{(1)} &= x^{(0)} + \alpha_i d^{(0)} \\ x - x^{(1)} &= x - x^{(0)} - \alpha_i d^{(0)} \\ e^{(1)} &= e^{(0)} - \alpha_i d^{(0)} \end{aligned}$$

- formula for α:

$$\alpha = \frac{\left(d^{(0)}\right)^T e^{(0)}}{\left(d^{(0)}\right)^T d^{(0)}}$$

- **but:** we don't know $e^{(0)}$

A-Orthogonality

- make the search directions *A-orthogonal*:

$$\left(d^{(i)}\right)^T A d^{(j)} = 0$$

- again: errors A-orthogonal to previous directions:

$$\left(e^{(i+1)}\right)^T A d^{(i)} \stackrel{!}{=} 0$$

- equiv. to minimisation in search direction $d^{(i)}$:

$$\begin{aligned} \frac{\partial}{\partial \alpha} f\left(x^{(i+1)}\right) = \left(f'\left(x^{(i+1)}\right)\right)^T \frac{\partial}{\partial \alpha} x^{(i+1)} &= 0 \\ \Leftrightarrow \quad -\left(r^{(i+1)}\right)^T d^{(i)} &= 0 \\ \Leftrightarrow \quad -\left(d^{(i)}\right)^T A e^{(i+1)} &= 0 \end{aligned}$$

A-Conjugate Directions

- remember the formula for conjugate directions:

$$\alpha = \frac{\left(d^{(0)}\right)^T e^{(0)}}{\left(d^{(0)}\right)^T d^{(0)}}$$

- same computation, but with A-orthogonality:

$$\alpha_i = \frac{\left(d^{(i)}\right)^T A e^{(i)}}{\left(d^{(i)}\right)^T A d^{(i)}} = \frac{\left(d^{(i)}\right)^T r^{(i)}}{\left(d^{(i)}\right)^T A d^{(i)}}$$

  (for the i-th iteration)
- **these α_i can be computed!**
- still to do: find A-orthogonal search directions

A-Conjugate Directions (2)

classical approach to find orthogonal directions $\rightarrow$ **conjugate Gram-Schmidt process**:

- from linearly independent vectors $u^{(0)}, u^{(1)}, \ldots, u^{(i-1)}$
- construct orthogonal directions $d^{(0)}, d^{(1)}, \ldots, d^{(i-1)}$

$$
\begin{aligned}
d^{(i)} &= u^{(i)} + \sum_{k=0}^{i-1} \beta_{ik} d^{(k)} \\
\beta_{ik} &= -\frac{(u^{(i)})^T A d^{(k)}}{(d^{(k)})^T A d^{(k)}}
\end{aligned}
$$

- needs to keep all old search vectors in memory
- $\mathcal{O}(n^3)$ computational complexity $\Rightarrow$ infeasible

Conjugate Gradients

- use residuals (i.e., $u^{(i)} := r^{(0)}$) to construct conjugate directions:

$$d^{(i)} = r^{(i)} + \sum_{k=0}^{i-1} \beta_{ik} d^{(k)}$$

- new direction $d^{(i)}$ should be A-orthogonal to all $d^{(j)}$:

$$0 \stackrel{!}{=} \left(d^{(i)}\right)^T A d^{(j)} = \left(r^{(i)}\right)^T A d^{(j)} + \sum_{k=0}^{i-1} \beta_{ik} \left(d^{(k)}\right)^T A d^{(j)}$$

- all directions $d^{(k)}$ (for $k = 0, \ldots, i-1$) are already A-orthogonal (and $j < i$), hence:

$$0 = \left(r^{(i)}\right)^T A d^{(j)} + \beta_{ij} \left(d^{(j)}\right)^T A d^{(j)} \Rightarrow \beta_{ij} = -\frac{\left(r^{(i)}\right)^T A d^{(j)}}{\left(d^{(j)}\right)^T A d^{(j)}}$$

Conjugate Gradients – Status

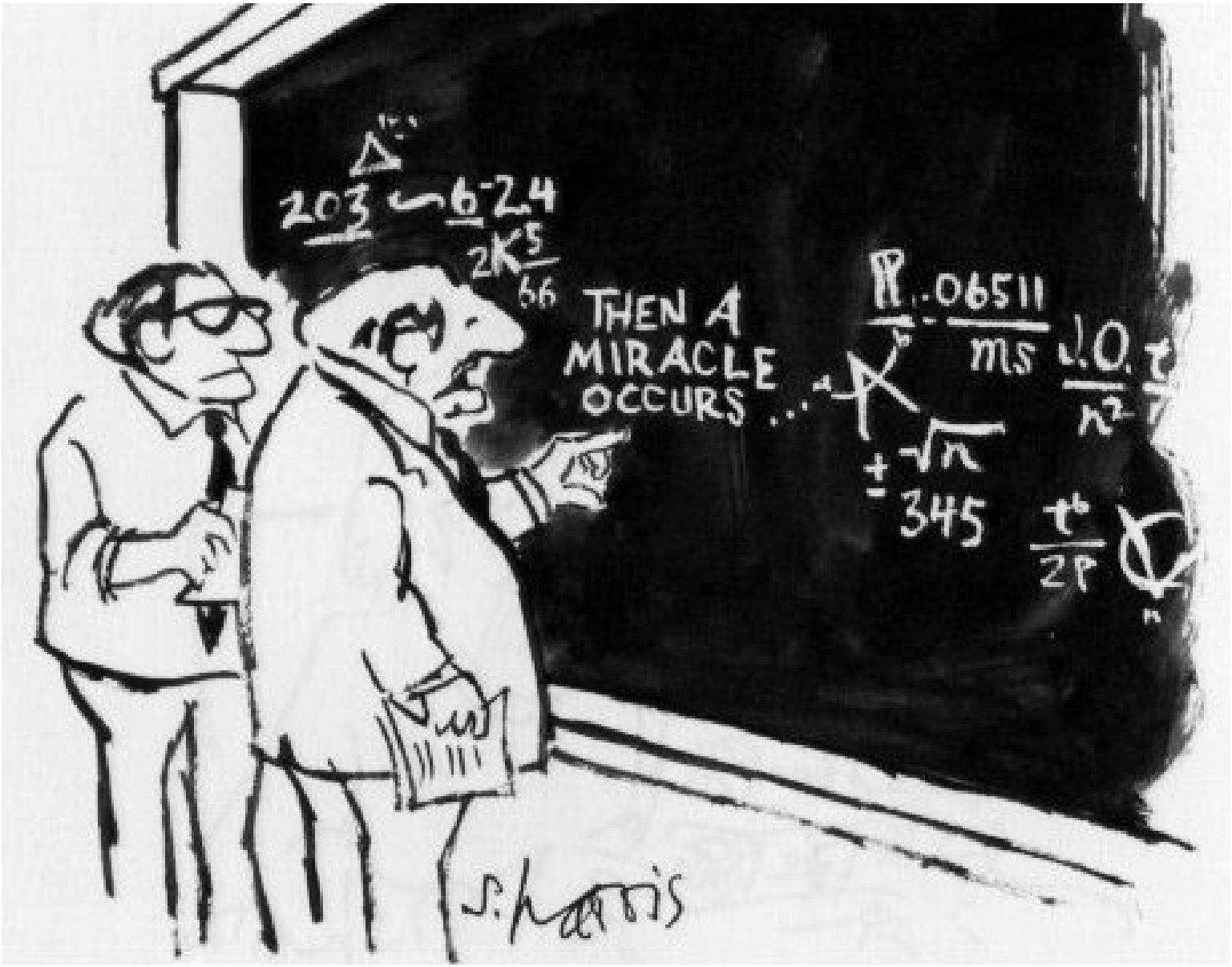
1. conjugate directions:

$$\begin{aligned} \alpha_i &= \frac{\left(d^{(i)}\right)^T r^{(i)}}{\left(d^{(i)}\right)^T A d^{(i)}} \\ x^{(i+1)} &= x^{(i)} + \alpha_i d^{(i)} \end{aligned}$$

2. use residuals to compute search directions:

$$\begin{aligned} d^{(i)} &= r^{(i)} + \sum_{k=0}^{i-1} \beta_{ik} d^{(k)} \\ \beta_{ik} &= -\frac{\left(r^{(i)}\right)^T A d^{(k)}}{\left(d^{(k)}\right)^T A d^{(k)}} \end{aligned}$$

$\rightarrow$ **still infeasible**, as we need to store all vectors $d^{(k)}$



"I think you should be more explicit here in step two."

from *What's so Funny about Science?* by Sidney Harris (1977)

A Miracle Occurs – Part 1

Two small contributions:

1. propagation of the error $e^{(i)} = x - x^{(i)}$

$$\begin{aligned} x^{(i+1)} &= x^{(i)} + \alpha_i d^{(i)} \\ x - x^{(i+1)} &= x - x^{(i)} - \alpha_i d^{(i)} \\ e^{(i+1)} &= e^{(i)} - \alpha_i d^{(i)} \end{aligned}$$

   (we have used this once, already)

2. propagation of residuals

$$\begin{aligned} r^{(i+1)} &= Ae^{(i+1)} = A\left(e^{(i)} - \alpha_i d^{(i)}\right) \\ \Rightarrow r^{(i+1)} &= r^{(i)} - \alpha_i A d^{(i)} \end{aligned}$$

A Miracle Occurs – Part 2

Orthogonality of the residuals:

- search directions are A-orthogonal
- only one step in each directions
- hence: error is A-orthogonal to previous search directions: $\left(d^{(i)}\right)^T A e^{(j)} = 0$, for $i < j$
- residuals are orthogonal to previous search directions: $\left(d^{(i)}\right)^T r^{(j)} = 0$, for $i < j$
- search directions are built from residuals: $\text{span}\left\{d^{(0)}, \ldots, d^{(i-1)}\right\} = \text{span}\left\{r^{(0)}, \ldots, r^{(i-1)}\right\}$
- hence: **residuals are orthogonal**

$$\left(r^{(i)}\right)^T r^{(j)} = 0, \qquad i < j$$

A Miracle Occurs – Part 3

- combine orthogonality and recurrence for residuals:

$$\begin{aligned} \left(r^{(i)}\right)^T r^{(j+1)} &= \left(r^{(i)}\right)^T r^{(j)} - \alpha_j \left(r^{(i)}\right)^T A d^{(j)} \\ \Rightarrow \alpha_j \left(r^{(i)}\right)^T A d^{(j)} &= \left(r^{(i)}\right)^T r^{(j)} - \left(r^{(i)}\right)^T r^{(j+1)} \end{aligned}$$

- $\left(r^{(i)}\right)^T r^{(j)} = 0$, if $i \neq j$:

$$\left(r^{(i)}\right)^T A d^{(j)} = \begin{cases} \frac{1}{\alpha_i}\left(r^{(i)}\right)^T r^{(i)}, & i = j \\ -\frac{1}{\alpha_{i-1}}\left(r^{(i)}\right)^T r^{(i)}, & i = j+1 \\ 0 & \text{otherwise.} \end{cases}$$

A Miracle Occurs – Part 4

- computation of β_{ik} (for $k = 0, \ldots, i-1$):

$$\begin{aligned}
\beta_{ik} &= -\frac{\left(r^{(i)}\right)^T A d^{(k)}}{\left(d^{(k)}\right)^T A d^{(k)}} \\
&= \begin{cases} \dfrac{\left(r^{(i)}\right)^T r^{(i)}}{\alpha_{i-1}\left(d^{(i-1)}\right)^T A d^{(i-1)}}, & i = k+1 \\ 0 & i > k+1 \end{cases}
\end{aligned}$$

- thus: search directions

$$\begin{aligned}
d^{(i)} &= r^{(i)} + \sum_{k=0}^{i-1} \beta_{ik} d^{(k)} = r^{(i)} + \beta_{i,i-1} d^{(i-1)} \\
\beta_i &:= \beta_{i,i-1} = \frac{\left(r^{(i)}\right)^T r^{(i)}}{\alpha_{i-1}\left(d^{(i-1)}\right)^T A d^{(i-1)}}
\end{aligned}$$

A Miracle Occurs – Part 5

- build search directions

$$\begin{aligned} d^{(i+1)} &= r^{(i+1)} + \beta_i d^{(i)} \\ \beta_{i+1} &= \frac{\left(r^{(i+1)}\right)^T r^{(i+1)}}{\alpha_i \left(d^{(i)}\right)^T A d^{(i)}} \end{aligned}$$

- remember: $\alpha_i = \dfrac{(d^{(i)})^T r^{(i)}}{(d^{(i)})^T A d^{(i)}}$
- thus: $\alpha_i (d^{(i)})^T A d^{(i)} = (d^{(i)})^T r^{(i)}$

$$\Rightarrow \beta_{i+1} = \frac{\left(r^{(i+1)}\right)^T r^{(i+1)}}{\left(d^{(i)}\right)^T r^{(i)}} = \frac{\left(r^{(i+1)}\right)^T r^{(i+1)}}{\left(r^{(i)}\right)^T r^{(i)}}$$

- last step: $\left(d^{(i)}\right)^T r^{(i)} = \left(r^{(i)} + \beta_{i-1} d^{(i-1)}\right)^T r^{(i)} = \left(r^{(i)}\right)^T r^{(i)} + \beta_{i-1} \left(d^{(i-1)}\right)^T r^{(i)} = \left(r^{(i)}\right)^T r^{(i)}$
  (residual $r^{(i)}$ orthogonal to previous search direction $d^{(i-1)}$)

Conjugate Gradients – Algorithm

Start: $d^{(0)} = r^{(0)} = b - Ax^{(0)}$

1. $\alpha_i = \dfrac{(r^{(i)})^T r^{(i)}}{(d^{(i)})^T A d^{(i)}}$
2. $x^{(i+1)} = x^{(i)} + \alpha_i d^{(i)}$
3. $r^{(i+1)} = r^{(i)} - \alpha_i A d^{(i)}$
4. $\beta_{i+1} = \dfrac{\left(r^{(i+1)}\right)^T r^{(i+1)}}{\left(r^{(i)}\right)^T r^{(i)}}$
5. $d^{(i+1)} = r^{(i+1)} + \beta_{i+1} d^{(i)}$

Part III

Preconditioning

Conjugate Gradients – Convergence

Convergence Analysis:

- uses *Krylow subspace*:

$$\text{span}\left\{r^{(0)}, Ar^{(0)}, A^2r^{(0)}, \ldots, A^{i-1}r^{(0)}\right\}$$

- "Krylow subspace method"

Convergence Results:

- in principle: direct method (n steps)
- in practice: iterative scheme

$$\left\|e^{(i)}\right\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^i \left\|e^{(0)}\right\|_A, \quad \kappa = \lambda_{\text{max}}/\lambda_{\text{min}}$$

Preconditioning

- convergence depends on matrix A
- idea: modify linear system

$$Ax = b \quad \rightsquigarrow \quad M^{-1}Ax = M^{-1}b,$$

  then: convergence depends on matrix $M^{-1}A$
- optimal preconditioner: $M^{-1} = A^{-1}$:

$$A^{-1}Ax = A^{-1}b \Leftrightarrow x = A^{-1}b.$$

- in practice:
  - avoid explicit computation of $M^{-1}A$
  - find an M similar to A, compute effect of M^{-1} (i.e., approximate solution of SLE)
  - or: find an M^{-1} similar to A^{-1}

CG and Preconditioning

- just replace A by $M^{-1}A$ in the algorithm??
- problem: $M^{-1}A$ not necessarily symmetric (even if M and A both are)
- we will try an alternative first: **symmetric preconditioning**

$$Ax = b \quad \rightsquigarrow \quad LAL^T\hat{x} = Lb, \quad x = L^T\hat{x}$$

- Note: for Finite Element discretization, this corresponds to a change of basis functions!
- requires some re-computations in the CG algorithm (see following slides)

"Change-of-Basis" Preconditioning

- preconditioned system of equations:

$$Ax = b \quad \rightsquigarrow \quad \underbrace{(L^T A L)}_{=:\widehat{A}} \hat{x} = \underbrace{L^T b}_{=:\hat{b}}, \quad x = L\hat{x}$$

- computation of residual:

$$\hat{r} = \hat{b} - \widehat{A}\,\hat{x} = L^T b - L^T A L \hat{x} = L^T (b - Ax) = L^T r$$

- computation of α for new system:

$$\alpha_i := \frac{\left(\hat{r}^{(i)}\right)^T \hat{r}^{(i)}}{\left(\hat{d}^{(i)}\right)^T \widehat{A}\,\hat{d}^{(i)}} = \frac{\left(\hat{r}^{(i)}\right)^T \hat{r}^{(i)}}{\left(\hat{d}^{(i)}\right)^T L^T A L\,\hat{d}^{(i)}} = \frac{\left(\hat{r}^{(i)}\right)^T \hat{r}^{(i)}}{\left(\tilde{d}^{(i)}\right)^T A\,\tilde{d}^{(i)}}$$

  where we defined $L\hat{d}^{(i)} =: \tilde{d}^{(i)}$

- update of solution:

$$\begin{aligned} \hat{x}^{(i+1)} &= \hat{x}^{(i)} + \alpha_i \hat{d}^{(i)} \\ \Rightarrow x^{(i+1)} = L\hat{x}^{(i+1)} &= L\hat{x}^{(i)} + L\alpha_i \hat{d}^{(i)} \\ &= x^{(i)} + \alpha_i \tilde{d}^{(i)} \end{aligned}$$

"Change-of-Basis" Preconditioning (2)

- update residuals $\hat{r}$:

$$\begin{aligned}\hat{r}^{(i+1)} &= \hat{r}^{(i)} - \alpha_i \widehat{A}\hat{d}^{(i)} = \hat{r}^{(i)} - \alpha_i L^T A L\, \hat{d}^{(i)} \\ &= \hat{r}^{(i)} - \alpha_i L^T A\, \tilde{d}^{(i)}\end{aligned}$$

- computation of β_i:

$$\beta_{i+1} = \frac{\left(\hat{r}^{(i+1)}\right)^T \hat{r}^{(i+1)}}{\left(\hat{r}^{(i)}\right)^T \hat{r}^{(i)}}$$

- update of search directions:

$$\begin{aligned}\hat{d}^{(i+1)} &= \hat{r}^{(i+1)} + \beta_i \hat{d}^{(i)} \\ \Rightarrow \tilde{d}^{(i+1)} = L\hat{d}^{(i+1)} &= L\hat{r}^{(i+1)} + L\beta_i \hat{d}^{(i)} \\ &= L\hat{r}^{(i+1)} + \beta_i \tilde{d}^{(i)}\end{aligned}$$

CG with "Change-of-Basis" Preconditioning

Start: $\hat{r}^{(0)} = L^T(b - Ax^{(0)})$; $\tilde{d}^{(0)} = L\,\hat{r}^{(0)}$

1. $\alpha_i = \dfrac{\left(\hat{r}^{(i)}\right)^T \hat{r}^{(i)}}{\left(\tilde{d}^{(i)}\right)^T A\,\tilde{d}^{(i)}}$
2. $x^{(i+1)} = x^{(i)} + \alpha_i \tilde{d}^{(i)}$
3. $\hat{r}^{(i+1)} = \hat{r}^{(i)} - \alpha_i L^T A\,\tilde{d}^{(i)}$
4. $\beta_{i+1} = \dfrac{\left(\hat{r}^{(i+1)}\right)^T \hat{r}^{(i+1)}}{\left(\hat{r}^{(i)}\right)^T \hat{r}^{(i)}}$
5. $\tilde{d}^{(i+1)} = L\,\hat{r}^{(i+1)} + \beta_i \tilde{d}^{(i)}$

Hierarchical Basis Preconditioning

Some specifics for the CG implementation:

- L transforms vector from nodal basis to hierarchical basis, for example $\hat{r} = L\,r$
- L^T transforms the vector of basis functions from nodal basis to hierarchical basis (cmp. FEM)
- effect of L and L^T can be computed in $\mathcal{O}(N)$ operations

HB-CG for the Poisson problem:

- in 1D: convergence after a $\log N$ iterations! (in this case: $L^T A L$ diagonal matrix with $\log N$ different eigenvalues)
- in 2D and 3D very fast convergence!
- further improved by additional diagonal preconditioning
- so-called *hierarchical generating systems* (change to a multigrid basis) achieve multigrid-like performance

CG and Preconditioning (revisited)

- preconditioning: replace A by $M^{-1}A$
- problem: $M^{-1}A$ not necessarily symmetric
- compare symmetric preconditioning

$$Ax = b \quad \rightsquigarrow \quad L^T A L \hat{x} = Lb, \quad x = L\hat{x}$$

- workaround: find $E^T E = M$ (Cholesky fact.), then

$$Ax = b \quad \rightsquigarrow \quad E^{-T} A E^{-1} \hat{x} = E^{-T} b, \quad \hat{x} = Ex$$

- what if E cannot be computed (efficiently)? (neither M nor M^{-1} might be known explicitly!)
- E, E^{-T}, E^{-1} can be eliminated from algorithm (requires some re-computations)

CG with Preconditioner

Start: $r^{(0)} = b - Ax^{(0)}$; $d^{(0)} = M^{-1} r^{(0)}$

1. $\alpha_i = \dfrac{(r^{(i)})^T M^{-1} r^{(i)}}{(d^{(i)})^T A d^{(i)}}$
2. $x^{(i+1)} = x^{(i)} + \alpha_i d^{(i)}$
3. $r^{(i+1)} = r^{(i)} - \alpha_i A d^{(i)}$
4. $\beta_{i+1} = \dfrac{\left(r^{(i+1)}\right)^T M^{-1} r^{(i+1)}}{\left(r^{(i)}\right)^T M^{-1} r^{(i)}}$
5. $d^{(i+1)} = M^{-1} r^{(i+1)} + \beta_{i+1} d^{(i)}$

Implementation

Preconditioning steps: $M^{-1}r^{(i)}$, $M^{-1}r^{(i+1)}$

- M^{-1} known then multiply $M^{-1}r^{(i)}$
- M known, then solve $My = r^{(i)}$ to obtain $y = M^{-1}r^{(i)}$
- neither M, nor M^{-1} are known explicitly:
  - algorithm to solve $My = r^{(i)}$ is sufficient!
    $\rightarrow$ any approximate solver for $Ae = r^{(i)}$
  - algorithm to compute M^{-1} is sufficient!
    $\rightarrow$ compute (sparse) approximate inverse (SPAI)
- Examples: Multigrid, Jacobi, ILU, SPAI, ...

Preconditioners for CG – Examples

- find $M \approx A$ and compute effect of M^{-1}:
  - Jacobi preconditioning: $M := D_A$
  - Gauss-Seidel preconditioning: $M := L_A$, etc.
- just compute effect of M^{-1}:
  - any approximate solver might do
    $\rightarrow$ incl. multigrid methods
  - incomplete LU-decomposition (ILU)
  - use a multigrid method as preconditioner(?)
    $\rightarrow$ worthwhile (only) in situations where multigrid does not work (well) as stand-alone solver
- find an M^{-1} similar to A^{-1}
  - "sparse approximate inverse" (SPAI)
  - tries to minimise $\|I - MA\|_F$, where M is a matrix with (given) sparse non-zero pattern

Literature

General:

- Gander, Hrebicek: *Solving Problems in Scientific Computing Using Maple and MATLAB*.
- Golub, Ortega: *Scientific Computing and Differential Equations*.
- Dongarra, et. al.: *Numerical linear algebra for high-performance computers*.

Literature (2)

Multigrid:

- Briggs, Henson, McCormick: *A Multigrid Tutorial* (2nd ed.).

Conjugate Gradients:

- Shewchuk: *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*.