

# Iterative Solvers for Large Linear Systems

## Part III: Conjugate Gradient Method

Andreas Meister

University of Kassel, Department of Analysis and Applied Mathematics

- Basics of Iterative Methods
- Splitting-schemes
  - Jacobi- u. Gauß-Seidel-scheme
  - Relaxation methods
- **Methods for symmetric, positive definite Matrices**
  - **Method of steepest descent**
  - **Method of conjugate directions**
  - **CG-scheme**

- Multigrid Method
  - Smoother, Prolongation, Restriction
  - Twogrid Method and Extension
- Methods for non-singular Matrices
  - GMRES
  - BiCG, CGS and BiCGSTAB
- Preconditioning
  - ILU, IC, GS, SGS, ...

# Projection method & Krylov subspace approach

We consider

$$Ax = b$$

with given data  $A \in \mathbb{R}^{n \times n}$ ,  $b \in \mathbb{R}^n$ .

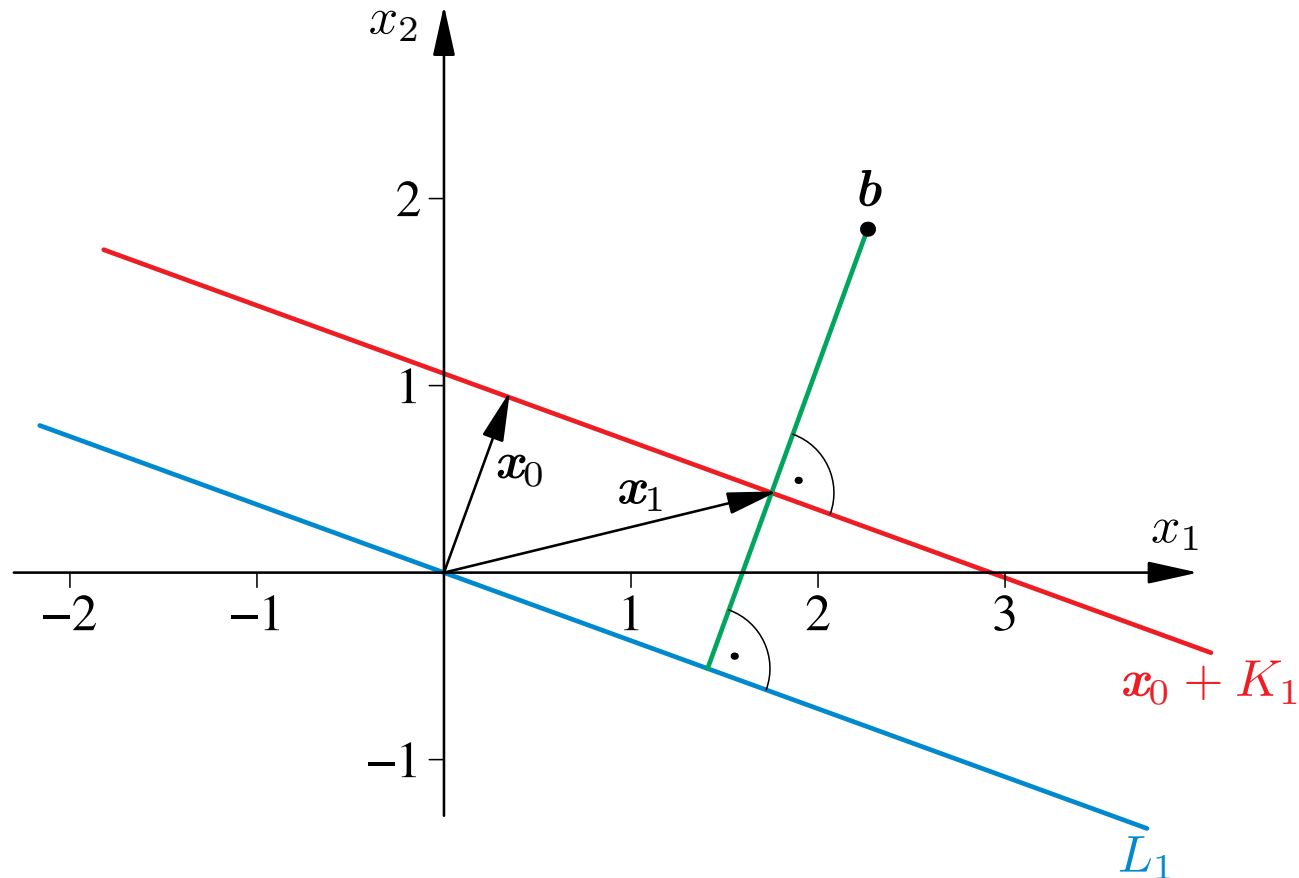
Splitting methods	Projection methods
Looking for approximations $x_m \in \mathbb{R}^n$	Looking for approximations $x_m \in x_0 + K_m \subset \mathbb{R}^n$ $\dim K_m = m \leq n$
Numerical algorithm $x_{m+1} = Mx_m + Nb$	Numerical algorithm (orthogonality constraint) $b - Ax_m \perp L_m \subset \mathbb{R}^n$ $\dim L_m = m \leq n$

# Projection method & Krylov subspace approach

## Example

$$A = I \in \mathbb{R}^{2 \times 2}$$

◦  $K_1 = L_1$  (Orthogonal projection method)

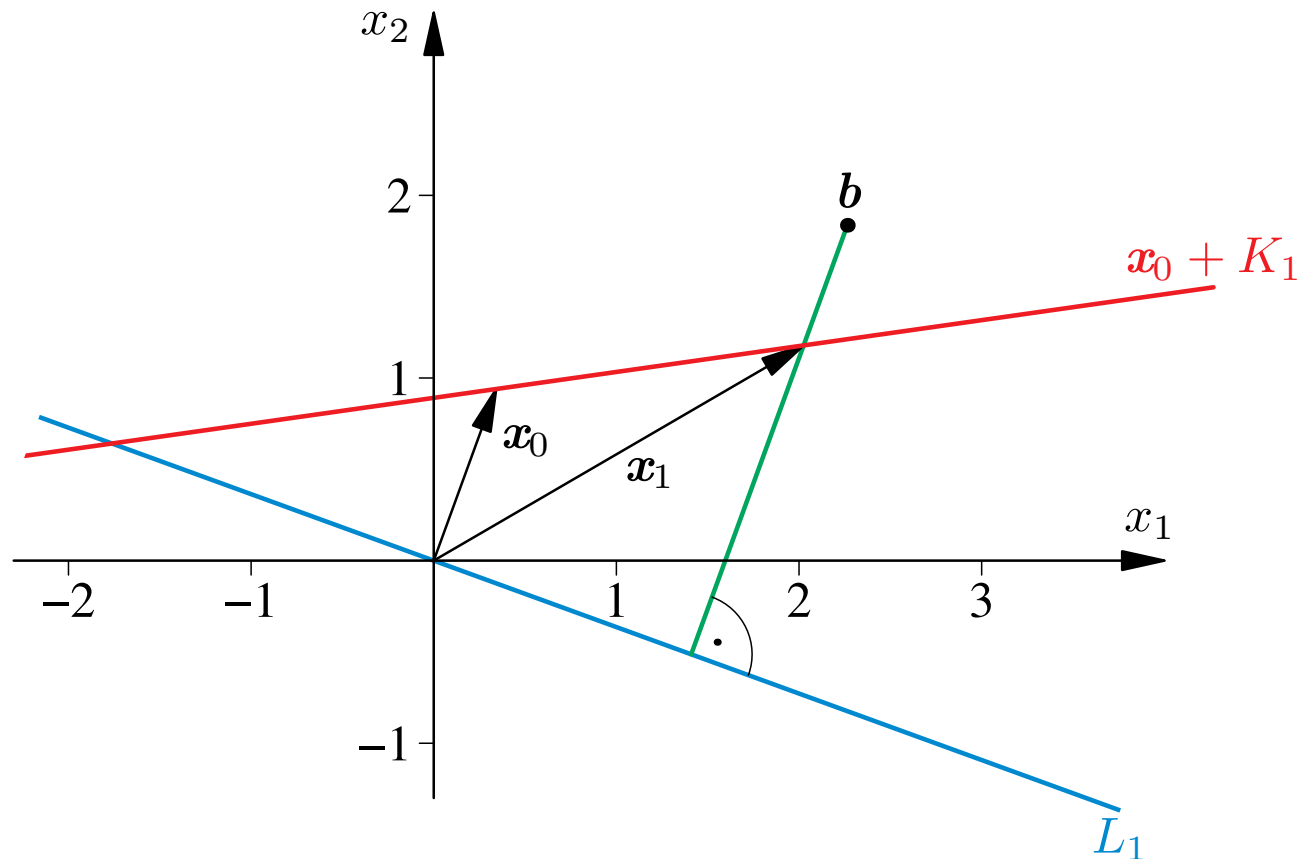


# Projection method & Krylov subspace approach

## Example

$$A = I \in \mathbb{R}^{2 \times 2}$$

- $K_1 = L_1$  (Orthogonal projection method)
- $K_1 \neq L_1$  (Skew projection method)



# Projection method & Krylov subspace approach

Krylov subspace approach:

Projection method based on

$$K_m = K_m(A, r_0) = \text{span}\{r_0, Ar_0, \dots, A^{m-1}r_0\},$$

with  $r_0 = b - Ax_0$  is called Krylov subspace method

# Methods for symmetric, positive definite matrices

## Basic idea:

Minimize the function

$$F(x) = \frac{1}{2}(Ax, x) - (b, x)$$

with respect to specific search directions

$$p_0, p_1, \dots \in \mathbb{R}^n \setminus \{0\}.$$

## Procedure:

- Choose  $x_0 \in \mathbb{R}^n$  and  $p_0, p_1, \dots \in \mathbb{R}^n \setminus \{0\}$ .
- For  $m = 0, 1, \dots$  we calculate  $x_{m+1}$  such that

$$F(x_{m+1}) \leq F(y) \quad \forall y \in x_m + \text{span}\{p_m\}$$

$$\begin{aligned} \implies x_{m+1} &= \arg \min_{\lambda \in \mathbb{R}} \underbrace{F(x_m + \lambda p_m)} \\ &= f_{x_m, p_m}(\lambda) \end{aligned}$$



# Methods for symmetric, positive definite matrices

## Questions:

- 1 Does  $x^* = A^{-1}b$  represent the global minimum of  $F$ ? **Yes**
- 2 How do we calculate  $\lambda \in \mathbb{R}$  ?

Concerning 1)

$$\begin{aligned} F(x) &= \frac{1}{2}(Ax, x) - (b, x) \\ \implies \nabla F(x) &= \frac{1}{2}(A + A^T)x - b \\ &\stackrel{\text{A symm.}}{=} Ax - b \end{aligned}$$

$$\implies \nabla^2 F(x) = A \stackrel{\text{A pos.def.}}{\implies} F \text{ is a convex mapping}$$

$$\nabla F(x) = 0 \iff x = A^{-1}b$$

# Methods for symmetric, positive definite matrices

## Questions:

- 1 Does  $x^* = A^{-1}b$  represent the global minimum of  $F$ ? Yes
- 2 How do we calculate  $\lambda \in \mathbb{R}$  ?

Concerning 1)

$$\begin{aligned} F(x) &= \frac{1}{2}(Ax, x) - (b, x) \\ \implies \nabla F(x) &= \frac{1}{2}(A + A^T)x - b \\ &\stackrel{\text{A symm.}}{=} Ax - b \end{aligned}$$

$$\implies \nabla^2 F(x) = A \stackrel{\text{A pos.def.}}{\implies} F \text{ is a convex mapping}$$

$$\nabla F(x) = 0 \iff x = A^{-1}b$$

# Methods for symmetric, positive definite matrices

## Questions:

- 1 Does  $x^* = A^{-1}b$  represent the global minimum of  $F$ ? **Yes**
- 2 How do we calculate  $\lambda \in \mathbb{R}$  ?

Concerning 1)

$$\begin{aligned} F(x) &= \frac{1}{2}(Ax, x) - (b, x) \\ \implies \nabla F(x) &= \frac{1}{2}(A + A^T)x - b \\ &\stackrel{\text{A symm.}}{=} Ax - b \end{aligned}$$

$$\implies \nabla^2 F(x) = A \stackrel{\text{A pos.def.}}{\implies} F \text{ is a convex mapping}$$

$$\nabla F(x) = 0 \iff x = A^{-1}b$$

# Methods for symmetric, positive definite matrices

## Questions:

① Does  $x^* = A^{-1}b$  represent the global minimum of  $F$ ? **Yes**

② How do we calculate  $\lambda \in \mathbb{R}$ ?  $\lambda = \frac{(b - Ax_m, p_m)}{(Ap_m, p_m)}$

Conc. 2)

$$f_{x,p}(\lambda) = \frac{1}{2}(A(x + \lambda p), x + \lambda p) - (b, x + \lambda p)$$

$$= F(x) + \lambda(Ax - b, p) + \frac{1}{2}\lambda^2(Ap, p)$$

$$f'_{x,p}(\lambda) = (Ax - b, p) + \lambda(Ap, p)$$

$$f''_{x,p}(\lambda) = (Ap, p) > 0 \quad \text{für } p \neq 0$$

• Thus,  $f_{x,p}$  is convex and the optimal  $\lambda$  is given in the form

$$f'_{x,p}(\lambda) = 0 \iff \lambda = \frac{(b - Ax, p)}{(Ap, p)}.$$

# Methods for symmetric, positive definite matrices

## Questions:

① Does  $x^* = A^{-1}b$  represent the global minimum of  $F$ ? **Yes**

② How do we calculate  $\lambda \in \mathbb{R}$ ?  $\lambda = \frac{(b - Ax_m, p_m)}{(Ap_m, p_m)}$

Conc. 2)

$$f_{x,p}(\lambda) = \frac{1}{2}(A(x + \lambda p), x + \lambda p) - (b, x + \lambda p)$$

$$= F(x) + \lambda(Ax - b, p) + \frac{1}{2}\lambda^2(Ap, p)$$

$$f'_{x,p}(\lambda) = (Ax - b, p) + \lambda(Ap, p)$$

$$f''_{x,p}(\lambda) = (Ap, p) > 0 \quad \text{für } p \neq 0$$

• Thus,  $f_{x,p}$  is convex and the optimal  $\lambda$  is given in the form

$$f'_{x,p}(\lambda) = 0 \iff \lambda = \frac{(b - Ax, p)}{(Ap, p)}.$$

# Methods for symmetric, positive definite matrices

## Questions:

① Does  $x^* = A^{-1}b$  represent the global minimum of  $F$ ? **Yes**

② How do we calculate  $\lambda \in \mathbb{R}$ ?  $\lambda = \frac{(b - Ax_m, p_m)}{(Ap_m, p_m)}$

Conc. 2)

$$f_{x,p}(\lambda) = \frac{1}{2}(A(x + \lambda p), x + \lambda p) - (b, x + \lambda p)$$

$$= F(x) + \lambda(Ax - b, p) + \frac{1}{2}\lambda^2(Ap, p)$$

$$f'_{x,p}(\lambda) = (Ax - b, p) + \lambda(Ap, p)$$

$$f''_{x,p}(\lambda) = (Ap, p) > 0 \quad \text{für } p \neq 0$$

• Thus,  $f_{x,p}$  is convex and the optimal  $\lambda$  is given in the form

$$f'_{x,p}(\lambda) = 0 \iff \lambda = \frac{(b - Ax, p)}{(Ap, p)}.$$

# Methods for symmetric, positive definite matrices

## Residual

The vector  $r = b - Ax$  is called **residual** (vector).

## Algorithm:

- Choose  $x_0 \in \mathbb{R}^n$  and  $p_0, p_1, \dots \in \mathbb{R}^n \setminus \{0\}$
- For  $m = 0, 1, \dots$

$$\begin{aligned}r_m &= b - Ax_m \\ \lambda_m &= \frac{(r_m, p_m)}{(Ap_m, p_m)} \\ x_{m+1} &= x_m + \lambda_m p_m\end{aligned}$$

## Problem:

Specification of the search direction  $p_0, p_1, \dots$ .

# Method of steepest descent

Basic idea:

Choose the optimal search direction in the local sense

$$\tilde{p}_m = -\nabla F(x_m) = -(Ax_m - b) = r_m$$

Normalizing the search direction:

$$\rho_m = \frac{\tilde{p}_m}{\|\tilde{p}_m\|_2} = \frac{r_m}{\|r_m\|_2}$$

Stopping criterion:  $r_m = 0$



# Method of steepest descent

## Algorithm:

- Choose  $x_0 \in \mathbb{R}^n$
- For  $m = 0, 1, \dots$

$$r_m = b - Ax_m$$

If  $r_m \neq 0$

$$\lambda_m = \frac{\|r_m\|_2^2}{(Ar_m, r_m)}$$

$$x_{m+1} = x_m + \lambda_m r_m$$

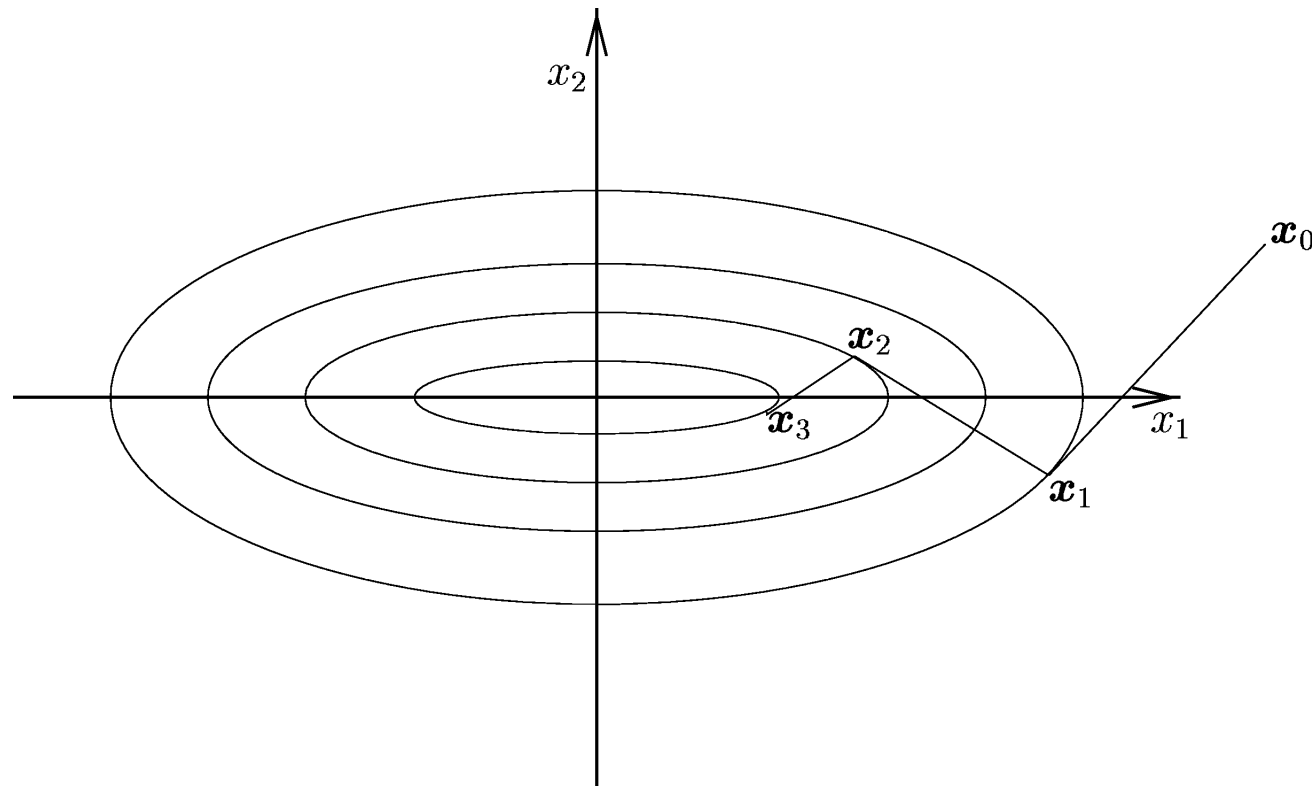
## Example

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 10 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad x_0 = \begin{pmatrix} 4 \\ \sqrt{1.8} \end{pmatrix}$$

# Method of steepest descent

$m$	$X_{m,1}$	$X_{m,2}$	$\varepsilon_m := \ X_m - X^*\ _A$	$\varepsilon_m / \varepsilon_{m-1}$
0	4.000000e+00	1.341641e+00	7.071068e+00	
1	2.987552e+00	-3.562863e-01	4.372680e+00	6.183904e-01
2	1.529627e+00	5.130523e-01	2.704023e+00	6.183904e-01
3	1.142460e+00	-1.362463e-01	1.672142e+00	6.183904e-01
4	5.849394e-01	1.961946e-01	1.034036e+00	6.183904e-01
5	4.368842e-01	-5.210148e-02	6.394382e-01	6.183904e-01
6	2.236847e-01	7.502613e-02	3.954224e-01	6.183904e-01
7	1.670674e-01	-1.992395e-02	2.445254e-01	6.183904e-01
8	8.553851e-02	2.869049e-02	1.512122e-01	6.183904e-01
9	6.388768e-02	-7.619051e-03	9.350814e-02	6.183904e-01
10	3.271049e-02	1.097143e-02	5.782453e-02	6.183904e-01
20	2.674941e-04	8.972025e-05	4.728673e-04	6.183904e-01
30	2.187466e-06	7.336985e-07	3.866930e-06	6.183904e-01
40	1.788827e-08	5.999910e-09	3.162230e-08	6.183904e-01
50	1.462836e-10	4.906500e-11	2.585953e-10	6.183904e-01
60	1.196252e-12	4.012351e-13	2.114695e-12	6.183904e-01
70	9.782499e-15	3.281150e-15	1.729318e-14	6.183904e-01
71	7.306431e-15	-8.713427e-16	1.069393e-14	6.183904e-01
72	3.740893e-15	1.254734e-15	6.613026e-15	6.183904e-01

# Method of steepest descent



Contour lines (level curves) of  $F(x) = \frac{1}{2}(Ax, x) - (b, x)$  w.r.t. the example

$$A = \begin{pmatrix} 2 & 0 \\ 0 & 10 \end{pmatrix}, \quad b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

are determined by  $F(x) = \frac{1}{2}(Ax, x) - (b, x) = x_1^2 + 5x_2^2$ .

# Method of steepest descent

## Problem:

### Forgetfulness

- During the calculation of the new search direction  $p_m$  we do not take into account any old search direction  $p_0, \dots, p_{m-1}$ .

## Aim:

- Choose linear independent  $p_0, p_1, \dots \in \mathbb{R}^n$
- Search into the direction  $p_m$  and find the optimal approximation  $x_{m+1} \in \mathbb{R}^n$  w.r.t.  $x_0 + \text{span}\{p_0, \dots, p_m\}$

## Effect:

- At least for  $m = n$  we obtain  $x_m = A^{-1}b$

# Method of steepest descent

## Problem:

### Forgetfulness

- During the calculation of the new search direction  $p_m$  we do not take into account any old search direction  $p_0, \dots, p_{m-1}$ .

## Aim:

- Choose linear independent  $p_0, p_1, \dots \in \mathbb{R}^n$
- Search into the direction  $p_m$  and find the optimal approximation  $x_{m+1} \in \mathbb{R}^n$  w.r.t.  $x_0 + \text{span}\{p_0, \dots, p_m\}$

## Effect:

- At least for  $m = n$  we obtain  $x_m = A^{-1}b$

# Method of steepest descent

## Problem:

### Forgetfulness

- During the calculation of the new search direction  $p_m$  we do not take into account any old search direction  $p_0, \dots, p_{m-1}$ .

## Aim:

- Choose linear independent  $p_0, p_1, \dots \in \mathbb{R}^n$
- Search into the direction  $p_m$  and find the optimal approximation  $x_{m+1} \in \mathbb{R}^n$  w.r.t.  $x_0 + \text{span}\{p_0, \dots, p_m\}$

## Effect:

- At least for  $m = n$  we obtain  $x_m = A^{-1}b$

# Method of steepest descent

## Optimality:

Let  $F : \mathbb{R}^n \rightarrow \mathbb{R}$ . A vector  $x \in \mathbb{R}^n$  is called

- 1 optimal w.r.t.  $p \in \mathbb{R}^n \setminus \{0\}$ , if

$$F(x) \leq F(x + \lambda p) \quad \forall \lambda \in \mathbb{R}.$$

- 2 optimal w.r.t.  $U \subset \mathbb{R}^n$ , if

$$F(x) \leq F(x + \xi) \quad \forall \xi \in U.$$

# Method of steepest descent

How to investigate the optimality of  $x \in \mathbb{R}^n$  w.r.t.  $U \subset \mathbb{R}^n$ ?

Consider

$$f_{x,\xi}(\lambda) = F(x + \lambda\xi)$$

$$f'_{x,\xi}(\lambda) = (Ax - b, \xi) + \lambda(A\xi, \xi)$$

$x$  is optimal w.r.t.  $U \ni \xi \neq 0$

$$\iff f'_{x,\xi}(0) = 0$$

$$\iff (Ax - b, \xi) = 0$$

$$\iff \boxed{r \perp U}$$



# Method of steepest descent

How to maintain optimality ?

Let  $x_m \in x_0 + \underbrace{\text{span}\{p_0, \dots, p_{m-1}\}}_{U_m :=}$ .

If  $x_m$  is optimal w.r.t.  $U_m$  and

$$x_{m+1} = x_m + \lambda_m p_m \quad , \quad \xi \in U_m$$

$$\implies (b - Ax_{m+1}, \xi) = \underbrace{(b - Ax_m, \xi)}_{=0} - \lambda_m (Ap_m, \xi)$$

Condition:

$$(Ap_m, p_i) = 0 \quad \text{für } i = 0, \dots, m-1$$

## Conjugate vectors

The vectors  $p_0, \dots, p_m \in \mathbb{R}^n \setminus \{0\}$  are called  
pairwise conjugated or A-orthogonal,  
if  
 $(Ap_j, p_i) = 0$  for all  $i \neq j$ .

# Method of steepest descent

How to obtain optimality w.r.t.  $U_{m+1}$ ?

Optimality w.r.t.  $U_m$ :

$$(Ap_m, p_i) = 0, \quad i = 0, \dots, m-1$$

Optimality w.r.t.  $p_m$ : ( $U_{m+1} = \{U_m, p_m\} := \text{span}\{p_0, \dots, p_{m-1}, p_m\}$ )

$$0 = (b - Ax_{m+1}, p_m) = (b - Ax_m, p_m) - \lambda_m (Ap_m, p_m)$$

$\implies$

$$\lambda_m = \frac{(b - Ax_m, p_m)}{(Ap_m, p_m)}$$

# Method of steepest descent

Eastern and Christmas simultaneously?

or in other words

Are pairwise conjugated vectors always linear independent?

Proof by contradiction: Assume:

$p_0, \dots, p_m \in \mathbb{R}^n \setminus \{0\}$  pairw. conjugated,  $p_m \in \text{span}\{p_0, \dots, p_{m-1}\}$

$$\implies p_m = \sum_{j=0}^{m-1} \alpha_j p_j$$

$$\implies 0 = (Ap_m, p_i) = \left( A \sum_{j=0}^{m-1} \alpha_j p_j, p_i \right) = \sum_{j=0}^{m-1} \alpha_j (Ap_j, p_i) = \alpha_i \underbrace{(Ap_i, p_i)}_{\neq 0}$$

holds for  $i=0, \dots, m-1 \implies p_m = 0$  **Contradiction!!!**

Answer: Yes, in the case that the matrix  $A$  is positive definite!

# Method of steepest descent

Eastern and Christmas simultaneously?

or in other words

Are pairwise conjugated vectors always linear independent?

Proof by contradiction: Assume:

$p_0, \dots, p_m \in \mathbb{R}^n \setminus \{0\}$  pairw. conjugated,  $p_m \in \text{span}\{p_0, \dots, p_{m-1}\}$

$$\implies p_m = \sum_{j=0}^{m-1} \alpha_j p_j$$

$$\implies 0 = (Ap_m, p_i) = \left( A \sum_{j=0}^{m-1} \alpha_j p_j, p_i \right) = \sum_{j=0}^{m-1} \alpha_j (Ap_j, p_i) = \alpha_i \underbrace{(Ap_i, p_i)}_{\neq 0}$$

holds for  $i=0, \dots, m-1 \implies p_m = 0$  **Contradiction!!!**

Answer: Yes, in the case that the matrix  $A$  is positive definite!

# Method of steepest descent

Eastern and Christmas simultaneously?

or in other words

Are pairwise conjugated vectors always linear independent?

Proof by contradiction: Assume:

$p_0, \dots, p_m \in \mathbb{R}^n \setminus \{0\}$  pairw. conjugated,  $p_m \in \text{span}\{p_0, \dots, p_{m-1}\}$

$$\implies p_m = \sum_{j=0}^{m-1} \alpha_j p_j$$

$$\implies 0 = (Ap_m, p_i) = \left( A \sum_{j=0}^{m-1} \alpha_j p_j, p_i \right) = \sum_{j=0}^{m-1} \alpha_j (Ap_j, p_i) = \alpha_i \underbrace{(Ap_i, p_i)}_{\neq 0}$$

holds for  $i=0, \dots, m-1 \implies p_m = 0$  **Contradiction!!!**

**Answer: Yes, in the case that the matrix  $A$  is positive definite!**

# Method of conjugate directions

## Summary:

- Choose **pairwise conjugated** search directions  $p_0, \dots, p_{n-1}$
- Calculate

$$\lambda_m = \frac{(b - Ax_m, p_m)}{(Ap_m, p_m)}, \quad m = 0, \dots, n-1$$

$\implies$  Hence, one obtains at least  $x_n = A^{-1}b$ .

# Method of conjugate directions

## Algorithm (Method of conjugate directions)

- Choose  $x_0 \in \mathbb{R}^n$  and  $p_0, \dots, p_{n-1}$  pairwise conjugated
- $r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

$$\lambda_m = \frac{(r_m, p_m)}{(Ap_m, p_m)}$$

$$x_{m+1} = x_m + \lambda_m p_m$$

$$r_{m+1} = r_m - \lambda_m Ap_m$$

## Problems

- Calculation of  $p_0, \dots, p_{n-1}$
- error reduction (convergence history)



# Method of conjugate directions

## Algorithm (Method of conjugate directions)

- Choose  $x_0 \in \mathbb{R}^n$  and  $p_0, \dots, p_{n-1}$  pairwise conjugated
- $r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

$$\lambda_m = \frac{(r_m, p_m)}{(Ap_m, p_m)}$$

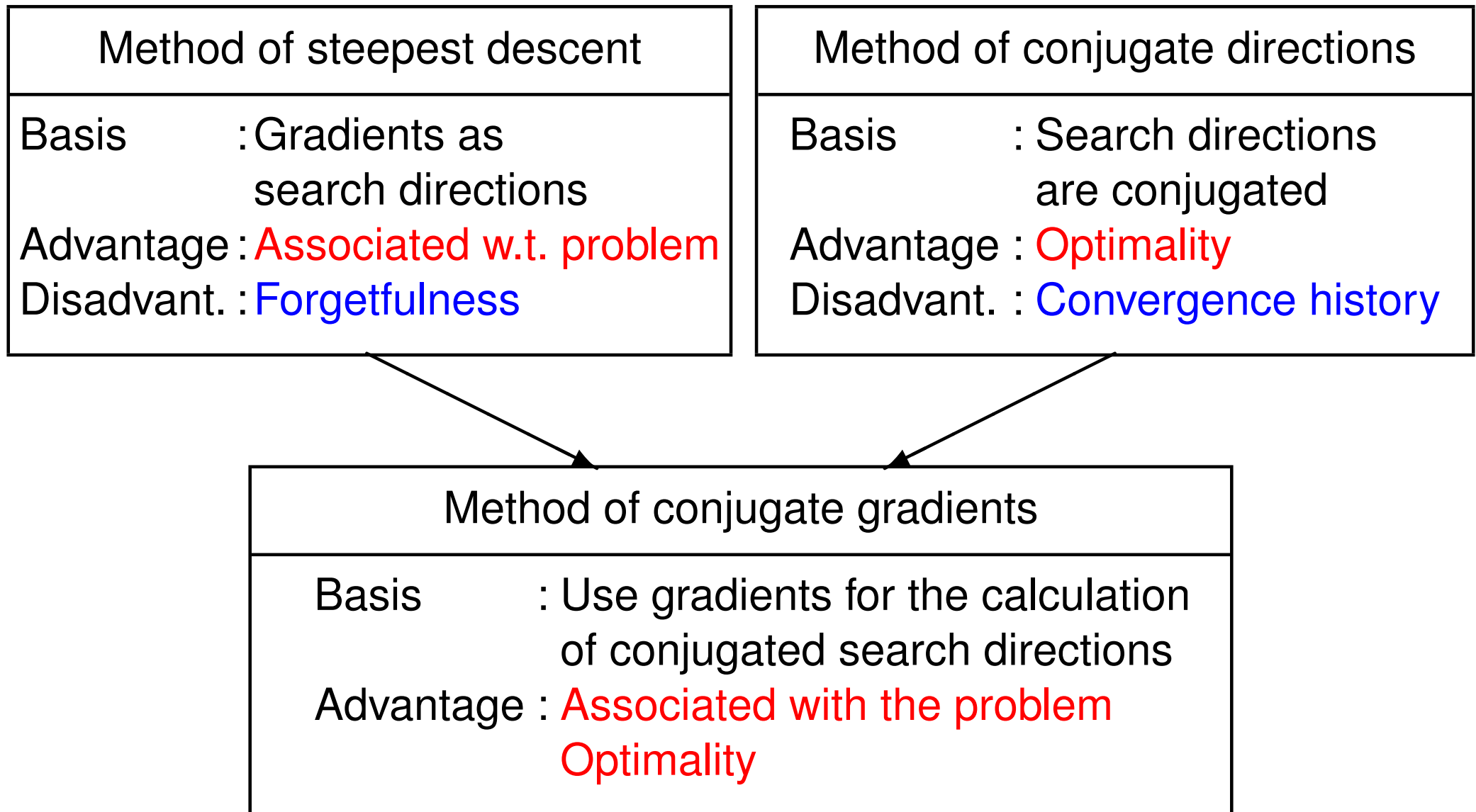
$$x_{m+1} = x_m + \lambda_m p_m$$

$$r_{m+1} = r_m - \lambda_m Ap_m$$

## Problems

- Calculation of  $p_0, \dots, p_{n-1}$
- error reduction (convergence history)

# Method of conjugate gradients (CG)



# Method of conjugate gradients (CG)

Ansatz:

$$p_0 = r_0$$

$$p_m = r_m + \sum_{j=0}^{m-1} \alpha_j p_j \quad , \quad m = 1, \dots, n-1$$

- m degrees of freedom
- Calculation of  $\alpha_0, \dots, \alpha_{m-1}$

$$0 = (Ap_m, p_i) = (Ar_m, p_i) + \sum_{j=0}^{m-1} \alpha_j \underbrace{(Ap_j, p_i)}_{=0 \quad i \neq j}$$

$$\alpha_j = -\frac{(Ar_m, p_i)}{(Ap_i, p_i)} \quad , \quad i = 0, \dots, m-1$$

# Method of conjugate gradients (CG)

## Algorithm

- Choose  $x_0 \in \mathbb{R}^n$  and define  $p_0 = r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

$$\begin{aligned}\lambda_m &= \frac{(r_m, p_m)}{(Ap_m, p_m)} \\ x_{m+1} &= x_m + \lambda_m p_m \\ r_{m+1} &= r_m - \lambda_m Ap_m \\ p_{m+1} &= r_{m+1} - \sum_{j=0}^m \frac{(Ar_{m+1}, p_j)}{(Ap_j, p_j)} p_j\end{aligned}$$

## Disadvantages

- **Break down** for  $p_m = 0$
- **Inapplicable** in the case of large, sparse matrices
- **Computational effort** increasing from iteration step to iteration step

# Method of conjugate gradients (CG)

## Algorithm

- Choose  $x_0 \in \mathbb{R}^n$  and define  $p_0 = r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

$$\begin{aligned}\lambda_m &= \frac{(r_m, p_m)}{(Ap_m, p_m)} \\ x_{m+1} &= x_m + \lambda_m p_m \\ r_{m+1} &= r_m - \lambda_m Ap_m \\ p_{m+1} &= r_{m+1} - \sum_{j=0}^m \frac{(Ar_{m+1}, p_j)}{(Ap_j, p_j)} p_j\end{aligned}$$

## Disadvantages

- Break down for  $p_m = 0$
- Inapplicable in the case of large, sparse matrices
- Computational effort increasing from iteration step to iteration step

# Method of conjugate gradients (CG)

## Algorithm

- Choose  $x_0 \in \mathbb{R}^n$  and define  $p_0 = r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

$$\begin{aligned}\lambda_m &= \frac{(r_m, p_m)}{(Ap_m, p_m)} \\ x_{m+1} &= x_m + \lambda_m p_m \\ r_{m+1} &= r_m - \lambda_m Ap_m \\ p_{m+1} &= r_{m+1} - \sum_{j=0}^m \frac{(Ar_{m+1}, p_j)}{(Ap_j, p_j)} p_j\end{aligned}$$

## Disadvantages

- **Break down** for  $p_m = 0$
- **Inapplicable** in the case of large, sparse matrices
- **Computational effort** increasing from iteration step to iteration step

# Method of conjugate gradients (CG)

## Algorithm

- Choose  $x_0 \in \mathbb{R}^n$  and define  $p_0 = r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

$$\begin{aligned}\lambda_m &= \frac{(r_m, p_m)}{(Ap_m, p_m)} \\ x_{m+1} &= x_m + \lambda_m p_m \\ r_{m+1} &= r_m - \lambda_m Ap_m \\ p_{m+1} &= r_{m+1} - \sum_{j=0}^m \frac{(Ar_{m+1}, p_j)}{(Ap_j, p_j)} p_j\end{aligned}$$

## Disadvantages

- **Break down** for  $p_m = 0$
- **Inapplicable** in the case of large, sparse matrices
- **Computational effort** increasing from iteration step to iteration step

# Method of conjugate gradients (CG)

## Algorithm

- Choose  $x_0 \in \mathbb{R}^n$  and define  $p_0 = r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

$$\begin{aligned}\lambda_m &= \frac{(r_m, p_m)}{(Ap_m, p_m)} \\ x_{m+1} &= x_m + \lambda_m p_m \\ r_{m+1} &= r_m - \lambda_m Ap_m \\ p_{m+1} &= r_{m+1} - \sum_{j=0}^m \frac{(Ar_{m+1}, p_j)}{(Ap_j, p_j)} p_j\end{aligned}$$

## Disadvantages

- **Break down** for  $p_m = 0$
- **Inapplicable** in the case of large, sparse matrices
- **Computational effort** increasing from iteration step to iteration step



# Method of conjugate gradients (CG)

## Properties and consequences

①  $U_m := \text{span}\{p_0, \dots, p_{m-1}\} = \text{span}\{r_0, Ar_0, \dots, A^{m-1}r_0\}$

•  $x_m = x_{m-1} + \lambda_{m-1}p_{m-1} = \dots = x_0 + \sum_{j=0}^{m-1} \lambda_j p_j$

$\implies x_m \in x_0 + U_m = x_0 + K_m$

②  $r_m \perp U_m$

•  $r_m = b - Ax_m \perp U_m = K_m$

$\implies$  Orthogonal Krylov subspace method

# Method of conjugate gradients (CG)

## Properties and consequences

$$\textcircled{1} \quad x_m = A^{-1}b \iff r_m = 0 \iff p_m = 0$$

- $p_m = 0 \equiv$  Stopping criterion

$$\textcircled{2} \quad (Ar_{m+1}, p_j) = 0 \quad , j = 0, \dots, m-1$$

$$\bullet \quad p_{m+1} = r_{m+1} - \sum_{j=0}^m \frac{(Ar_{m+1}, p_j)}{(Ap_j, p_j)} p_j$$

$$= r_{m+1} - \frac{(Ar_{m+1}, p_m)}{(Ap_m, p_m)} p_m$$

- Applicable for large sparse systems
- Low computational effort

# Method of conjugate gradients (CG)

## Algorithm

- Choose  $x_0 \in \mathbb{R}^n$  and define  $p_0 = r_0 = b - Ax_0$
- For  $m = 0, \dots, n - 1$

    If  $p_m \neq 0$  then

$$\lambda_m = \frac{(r_m, p_m)}{(Ap_m, p_m)}$$

$$x_{m+1} = x_m + \lambda_m p_m$$

$$r_{m+1} = r_m - \lambda_m Ap_m$$

$$p_{m+1} = r_{m+1} - \frac{(Ar_{m+1}, p_m)}{(Ap_m, p_m)} p_m$$

else STOP

# Method of conjugate gradients (CG)

Example: 1-D Poisson-Equation  $x'' = b$

$$D = [0, 1] \quad , \quad h = 1/8 \quad , \quad N = 7$$

$$\mathbb{R}^{7 \times 7} \ni A = \text{tridiag} \{-64, 128, -64\}$$

$$\mathbb{R}^7 \ni b = (128, -448, 704, -832, 512, 128, 320)^T, \quad x_0 = 0$$

Method of conjugate gradients (CG)								
$m$	$x_{m,1}$	$x_{m,2}$	$x_{m,3}$	$x_{m,4}$	$x_{m,5}$	$x_{m,6}$	$x_{m,7}$	$\ \vec{r}_m\ _2$
0	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1336.36
1	0.58	-2.04	3.21	-3.79	2.33	0.58	1.46	363.57
2	-0.39	-1.72	2.81	-4.57	3.00	4.99	4.26	252.76
3	-0.01	-2.38	2.06	-3.53	4.87	6.07	6.25	153.30
4	-0.14	-2.88	2.57	-2.13	6.50	7.48	5.93	117.64
5	-0.70	-2.18	3.53	-1.12	7.65	7.81	6.27	103.52
6	0.13	-1.14	5.40	0.54	8.23	8.54	6.98	89.70
7	1.00	0.00	6.00	1.00	9.00	9.00	7.00	0.00

# Convection-Diffusion Equation

## Governing Equation

$$\beta \cdot \nabla u(x, y) - \epsilon \Delta u(x, y) = 0 \quad \text{on } D = (0, 1) \times (0, 1)$$

with

$$\beta = \alpha \begin{pmatrix} \cos \frac{\pi}{4} \\ \sin \frac{\pi}{4} \end{pmatrix} \quad \alpha, \epsilon \in \mathbb{R}_0^+$$

## Boundary Conditions

$$u(x, y) = x^2 + y^2 \quad \text{for } (x, y) \in \partial D$$

## Mesh

$$x_i = i \cdot h \quad \text{and} \quad y_j = j \cdot h \quad \text{for } j = 0, \dots, N+1, \quad h = \frac{1}{N+1}$$

# Convection-Diffusion Equation

## Discretization of Laplacian (Central Difference)

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \approx \frac{1}{h^2}(u_{i+1,j} - 2u_{ij} + u_{i-1,j})$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) \approx \frac{1}{h^2}(u_{i,j+1} - 2u_{ij} + u_{i,j-1})$$

## Discretization of convective part (Backward Difference)

$$\frac{\partial u}{\partial x}(x_i, y_j) \approx \frac{1}{h}(u_{i,j} - u_{i-1,j})$$

$$\frac{\partial u}{\partial y}(x_i, y_j) \approx \frac{1}{h}(u_{i,j} - u_{i,j-1})$$

# Convection-Diffusion Equation

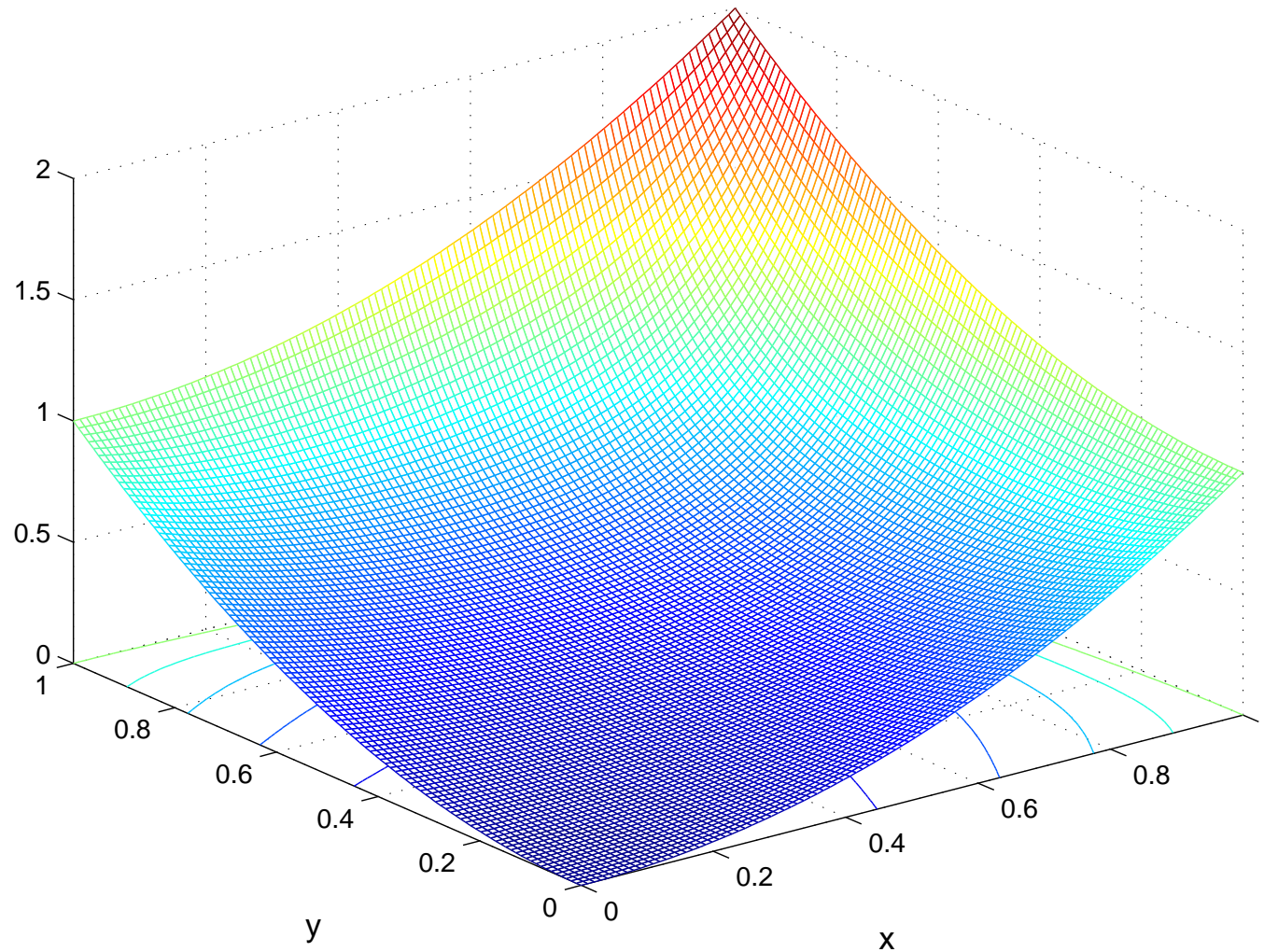
## Testcases

	$\alpha$	$\epsilon$	Matrix properties
Test 1	0	1	Symmetric, positive definite
Test 2	0.1	1	Non-symmetric, non-singular
Test 3	1	0.1	Non-symmetric, non-singular

- Number of unknowns:  $100 \times 100 = 10000$  ( $N = 100$ )
- Stopping criterion:  $\|r_j\|_2 < 10^{-12} \|b\|$

# Convection-Diffusion Equation

## Numerical Solution of Test 3

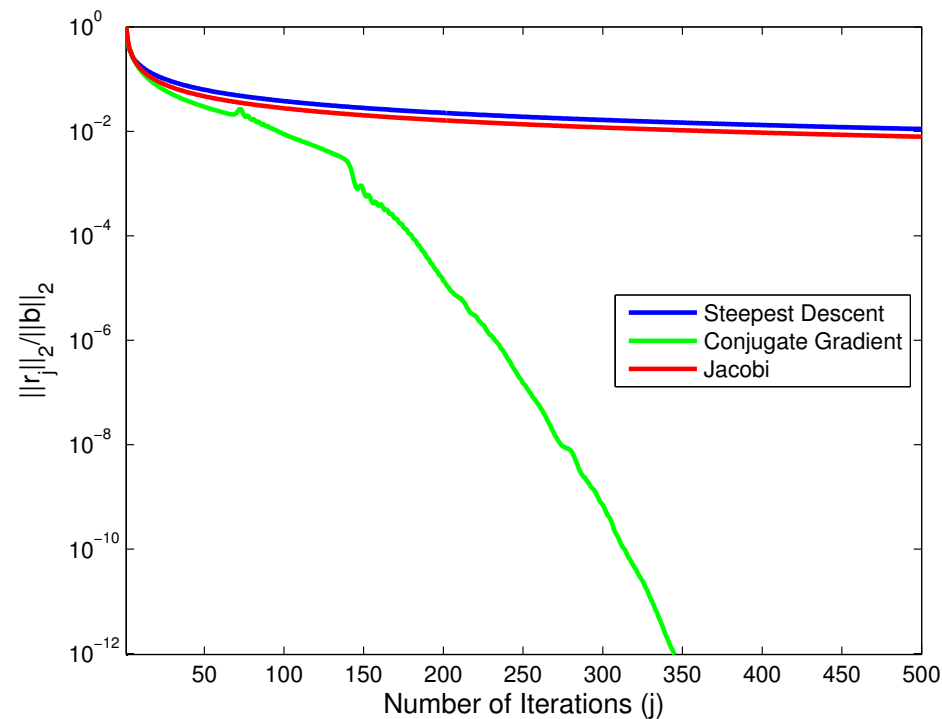




# Steepest Descent vs. Conjugate Gradient method

## Test 1: Pure Diffusion ( $\alpha = 0, \epsilon = 1$ )

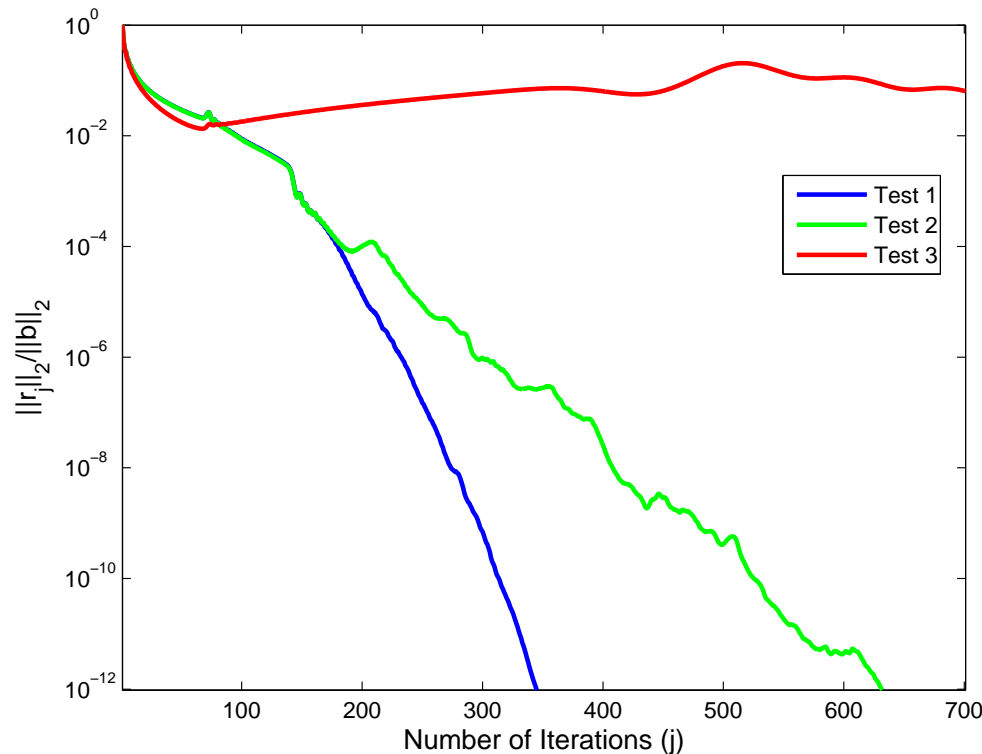
	Number of Iterations	CPU Time (%)
Conjugate Gradient	344	100
Steepest Descent	47258	12222
Jacobi Method	46582	14806



# Conjugate Gradients for Non-SPD Systems

## Comparison of CG method for all three test case

	$\alpha$	$\epsilon$	Number of Iterations
Test 1	0	1	344
Test 2	0.1	1	631
Test 3	1	0.1	Convergence failed



# Convergence properties of the CG-method

## Theorem

Let  $A \in \mathbb{R}^{n \times n}$  be symmetric and positive definite. Then the error estimate

$$\|e_m\|_A \leq 2 \left( \frac{\sqrt{\text{cond}_2(A)} - 1}{\sqrt{\text{cond}_2(A)} + 1} \right)^m \|e_0\|_A,$$

holds with  $e_m = x_m - A^{-1}b$ ,  $x_m =$  approximate solution.

## Properties: (A symm., positive definite )

- 1 Eigenvalues:  $\lambda_n \geq \dots \geq \lambda_1 > 0$ ,  
Eigenvectors:  $\{v_1, \dots, v_n\}$  ONB of  $\mathbb{R}^n$
- 2 Condition number:  $c := \text{cond}_2(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\lambda_n}{\lambda_1}$
- 3 Weighted vector norm (energy norm):

$$\|x\|_A = \sqrt{(Ax, x)}$$

## Relation between error- and residual vector:

Due to the relation

$$r_m = b - Ax_m = -A(x_m - A^{-1}b) = -Ae_m$$

one obtains

$$K_m = \text{span}\{r_0, Ar_0, \dots, A^{m-1}r_0\} = \text{span}\{Ae_0, \dots, A^m e_0\},$$

such that the approximate solution  $x_m \in x_0 + K_m$  reads

$$x_m = x_0 + \sum_{i=1}^m c_i A^i e_0.$$

# Convergence properties of the CG-method

## Formulation of the error vector

$$\mathbf{e}_m = \mathbf{x}_m - \mathbf{A}^{-1} \mathbf{b} = \underbrace{\mathbf{x}_0 - \mathbf{A}^{-1} \mathbf{b}}_{=\mathbf{e}_0} + \sum_{i=1}^m c_i \mathbf{A}^i \mathbf{e}_0 = \mathbf{p}_m(\mathbf{A}) \mathbf{e}_0$$

with  $\mathbf{p}_m \in \mathcal{P}_m^1 = \{p \in \mathcal{P}_m \mid p(0) = 1\}$

Equivalence to the minimization of the functional:

$$\begin{aligned} F(\mathbf{x}) &= \frac{1}{2} (\mathbf{A}\mathbf{x}, \mathbf{x}) - (\mathbf{b}, \mathbf{x}) = \frac{1}{2} (\underbrace{\mathbf{A}\mathbf{x} - \mathbf{b}}_{=\mathbf{A}\mathbf{e}}, \underbrace{\mathbf{x} - \mathbf{A}^{-1}\mathbf{b}}_{=\mathbf{e}}) - \frac{1}{2} \underbrace{(\mathbf{b}, \mathbf{A}^{-1}\mathbf{b})}_{const.} \\ &= \frac{1}{2} \|\mathbf{e}\|_{\mathbf{A}}^2 + const \end{aligned}$$

$$\mathbf{x}_m = \arg \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_m} F(\mathbf{x}) \iff \|\mathbf{e}_m\|_{\mathbf{A}} = \min_{p \in \mathcal{P}_m^1} \|p(\mathbf{A})\mathbf{e}_0\|_{\mathbf{A}}$$

# Convergence properties of the CG-method

Error estimate:  $\mathbf{e}_0 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$  with the ONB  $\mathbf{v}_1, \dots, \mathbf{v}_n$

$$1 \quad \|\mathbf{e}_0\|_A^2 = (\mathbf{A}\mathbf{e}_0, \mathbf{e}_0) = \left( \sum_{i=1}^n \lambda_i \alpha_i \mathbf{v}_i, \sum_{i=1}^n \alpha_i \mathbf{v}_i \right) = \sum_{i=1}^n \alpha_i^2 \lambda_i$$

$$2 \quad \|\mathbf{p}(\mathbf{A})\mathbf{e}_0\|_A^2 = \left( \sum_{i=1}^n \mathbf{p}(\lambda_i)^2 \alpha_i^2 \lambda_i \right)$$

$$3 \quad \|\mathbf{e}_m\|_A = \min_{\mathbf{p} \in \mathcal{P}_m^1} \left( \sum_{i=1}^n \mathbf{p}(\lambda_i)^2 \alpha_i^2 \lambda_i \right)^{\frac{1}{2}} \leq \min_{\mathbf{p} \in \mathcal{P}_m^1} \max_{\lambda \in \{\lambda_1, \dots, \lambda_n\}} |\mathbf{p}(\lambda)| \left( \sum_{i=1}^n \alpha_i^2 \lambda_i \right)^{\frac{1}{2}}$$

$$\leq \min_{\mathbf{p} \in \mathcal{P}_m^1} \max_{\lambda \in [\lambda_1, \lambda_n]} |\mathbf{p}(\lambda)| \|\mathbf{e}_0\|_A$$

# Convergence properties of the CG-method

Consideration of a suitable polynomial:

$$\|e_m\|_A \leq \min_{p \in P_m^1} \max_{\lambda \in [\lambda_1, \lambda_n]} |p(\lambda)| \|e_0\|_A$$

Case 1:  $\lambda_1 \neq \lambda_n$  Tschebyscheff-Polynomials

$$T_m(\lambda) = \cos(m \arccos \lambda), \quad m \in \mathbb{N}_0 \quad (\lambda \in [-1, 1])$$

- 1  $|T_m(\lambda)| \leq 1$
- 2  $T_m(\lambda) = 2 \lambda T_{m-1}(\lambda) + T_{m-2}(\lambda), \quad T_0(\lambda) = 1 \implies T_m \in P_m$
- 3  $T_m\left(\frac{1}{2}\left(\lambda + \frac{1}{\lambda}\right)\right) = \left(\frac{1}{2}\left(\lambda^m + \frac{1}{\lambda^m}\right)\right)$

$$p_m(\lambda) := \frac{T_m\left(\frac{2\lambda - (\lambda_n + \lambda_1)}{\lambda_1 - \lambda_n}\right)}{T_m\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right)} \stackrel{(2)}{\in} P_m^1$$

# Convergence properties of the CG-method

Utilizing

$$\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1} = \frac{\frac{\lambda_n}{\lambda_1} + 1}{\frac{\lambda_n}{\lambda_1} - 1} = \frac{c + 1}{c - 1} = \frac{1}{2} \left( \frac{\sqrt{c} + 1}{\sqrt{c} - 1} + \frac{\sqrt{c} - 1}{\sqrt{c} + 1} \right)$$

one gets

$$\begin{aligned} \frac{\|e_m\|_A}{\|e_0\|_A} &\leq \max_{\lambda \in [\lambda_1, \lambda_n]} |p_m(\lambda)| \stackrel{(1)}{\leq} \max_{\lambda \in [\lambda_1, \lambda_n]} \left| \frac{1}{T_m\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right)} \right| \\ &= \left| T_m\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right) \right|^{-1} \stackrel{(3)}{=} \left| 2 \frac{1}{\left(\frac{\sqrt{c} + 1}{\sqrt{c} - 1}\right)^m + \left(\frac{\sqrt{c} - 1}{\sqrt{c} + 1}\right)^m} \right| \\ &\leq 2 \left( \frac{\sqrt{c} - 1}{\sqrt{c} + 1} \right)^m \end{aligned}$$



# Convergence properties of the CG-method

Consideration of a suitable polynomial:

$$\|e_m\|_A \leq \min_{p \in P_m^1} \max_{\lambda \in [\lambda_1, \lambda_n]} |p(\lambda)| \|e_0\|_A$$

Case 2:  $\lambda_1 = \lambda_n$

Taking account of

$$c = \frac{\lambda_n}{\lambda_1} = 1.$$

we simply define

$$p_m(\lambda) = 1 - \frac{\lambda}{\lambda_n} \in P_m^1.$$

Thus,

$$\frac{\|e_m\|_A}{\|e_0\|_A} \leq \max_{\lambda \in [\lambda_1, \lambda_n]} |p_m(\lambda)| = 0 = 2 \left( \frac{\sqrt{c} - 1}{\sqrt{c} + 1} \right)^m$$