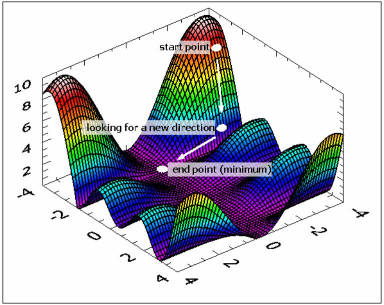
Università della Svizzera italiana
Institute of Computing CI

Introduction to the Conjugate-gradient method (without the agonizing pain)



D. Pasadakis, Lisa Gaedke-Merzhäuser
Prof. Olaf Schenk

November 16, 2022

Content

- Motivation $Ax = b$
- Gradient-based Methods
  - Solve equivalent minimization problem to $Ax = b$
  - Method of Steepest Descent
  - Conjugate-Gradient Method
- Outlook: Preconditioning

Motivation

Complexity of Gaussian Elimination (”direct methods”) for LARGE (sparse) linear systems of equations is too high

- *Direct methods* computes the exact solution in n steps
  - Cost for classical Gaussian Elimination: $\mathcal{O}(n^3)$
  - Cost for sparse matrices: $\mathcal{O}(n^{1.5})$ or $\mathcal{O}(n^2)$
  - High memory consumption due to additional fill-in elements

Motivation

Complexity of Gaussian Elimination (”direct methods”) for LARGE (sparse) linear systems of equations is too high

- *Direct methods* computes the exact solution in n steps
  - Cost for classical Gaussian Elimination: $\mathcal{O}(n^3)$
  - Cost for sparse matrices: $\mathcal{O}(n^{1.5})$ or $\mathcal{O}(n^2)$
  - High memory consumption due to additional fill-in elements
- *Iterative methods* uses an (arbitrary) initial guess (”starting point”) to compute an approximate (arbitrary) solution
  - Cost for conjugate-gradient algorithms: between $\mathcal{O}(n)$ and $\mathcal{O}(n^{3/2})$ (for an approximation using a fixed accuracy, e.g. 10^{-6})
  - Need no additional memory
  - Converges after a few iterations (this depends of course also on A).

But: Iterative methods are very often less robust and not as general as direct methods.

Sketch of an iterative method

Structure of the algorithm:

<u>Start:</u> $m = 0, \quad \boldsymbol{x}^{(m)} =$ Initial value

Iterate until error estimate $< \epsilon$:

  Find a new solution $\boldsymbol{x}^{(m+1)}$

  Compute new error estimate

- New, better solutions?
- Good error estimates?
- Reasonable error bounds?

Iterative methods

Basic idea: Use an (arbitrary) initial starting point to $\boldsymbol{x}^{(0)}$, and an iterative method to compute a better solution.

- Fixpoint-Iteration:
  - Sequence $\{\boldsymbol{x}^{(m)}\}$ with $\boldsymbol{x}^{(m+1)} = R\boldsymbol{x}^{(m)} + \boldsymbol{c}$
  - R has fix-point $\boldsymbol{x}^*$ at $\boldsymbol{x} = A^{-1}b$
  - Suitable, optimal R?

Iterative methods

Basic idea: Use an (arbitrary) initial starting point to $\boldsymbol{x}^{(0)}$, and an iterative method to compute a better solution.

- Fixpoint-Iteration:
  - Sequence $\{\boldsymbol{x}^{(m)}\}$ with $\boldsymbol{x}^{(m+1)} = R\boldsymbol{x}^{(m)} + \boldsymbol{c}$
  - R has fix-point $\boldsymbol{x}^*$ at $\boldsymbol{x} = A^{-1}b$
  - Suitable, optimal R?
- Gradient-methods (projections-based methods)
  - Computes an (sub-)space using search vectors from: $U_m := \operatorname{span}\{\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(m)}\}$
  - Minimizes an equivalent optimization problem $f(\boldsymbol{x}) := \frac{1}{2}\langle A\boldsymbol{x}, \boldsymbol{x}\rangle - \langle \boldsymbol{b}, \boldsymbol{x}\rangle$ along the search vectors $\boldsymbol{p}^{(0)}, \ldots, \boldsymbol{p}^{(m)}$
  - What are the optimal search vectors?
  - Why is the method called an iterative method?

Definition

Definition: The *error* in step m is the deviation of $\boldsymbol{x}^{(m)}$ from the exact solution $\boldsymbol{x}$:

$$\boldsymbol{e}^{(m)} = \boldsymbol{x} - \boldsymbol{x}^{(m)} = A^{-1}\boldsymbol{b} - \boldsymbol{x}^{(m)}$$

Unfortunately we do not know the error during the iterations (otherwise we would know the solution)

Definition

Definition: The *error* in step m is the deviation of $\boldsymbol{x}^{(m)}$ from the exact solution $\boldsymbol{x}$:

$$\boldsymbol{e}^{(m)} = \boldsymbol{x} - \boldsymbol{x}^{(m)} = A^{-1}\boldsymbol{b} - \boldsymbol{x}^{(m)}$$

Unfortunately we do not know the error during the iterations (otherwise we would know the solution)

Definition: The *residual* provides us a measure for the real error. It is relatively easy to compute:

$$\boldsymbol{r}^{(m)} = \boldsymbol{b} - A\boldsymbol{x}^{(m)} = -A\boldsymbol{e}^{(m)}$$

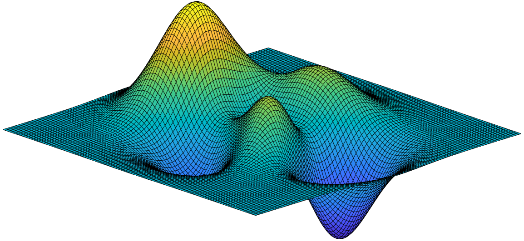
The property that the residual is equivalent the A-transformed error $(-A\boldsymbol{e}^{(m)})$ will be used in later analysis.

Notation

- $\langle \cdot, \cdot \rangle$ is the scalar product: $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^T \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i$
- $\boldsymbol{x} := A^{-1} \boldsymbol{b}$ denotes the exact solution
- $\boldsymbol{x}^{(m)}$ is the approximation in the m-th iteration

Gradient-based methods

www.myswissalps.com



A class of first order methods utilized for finding the nearest local minimum of a function, by following the negative of the direction of the negative gradient of this function at the current point (Cauchy 1847).

Minimization problem

Basic idea: Instead of $A\mathbf{x} = \mathbf{b}$ solve an equivalent minimization problem

$$f(\mathbf{x}) := \frac{1}{2}\langle A\mathbf{x}, \mathbf{x}\rangle - \langle \mathbf{b}, \mathbf{x}\rangle \tag{1}$$

Precondition: Let A a *symmetric positive definite (spd) matrix:*

$$\langle A\mathbf{x}, \mathbf{x}\rangle > 0, \quad \text{if } \mathbf{x} \neq \text{zero vector} \tag{2}$$

Minimization problem

Basic idea: Instead of $A\boldsymbol{x} = \boldsymbol{b}$ solve an equivalent minimization problem

$$f(\boldsymbol{x}) := \frac{1}{2}\langle A\boldsymbol{x}, \boldsymbol{x}\rangle - \langle \boldsymbol{b}, \boldsymbol{x}\rangle \tag{1}$$

Precondition: Let A a *symmetric positive definite (spd) matrix:*

$$\langle A\boldsymbol{x}, \boldsymbol{x}\rangle > 0, \quad \text{if } \boldsymbol{x} \neq \text{zero vector} \tag{2}$$

If we take the derivative of (1), it will lead to:

$$\begin{aligned} f'(\boldsymbol{x}) &= \frac{1}{2}A^T\boldsymbol{x} + \frac{1}{2}A\boldsymbol{x} - \boldsymbol{b} \\ &= A\boldsymbol{x} - \boldsymbol{b} \qquad \Big|\text{Condition: } A^T = A \end{aligned} \tag{3}$$

Setting the equation to zero will lead to the original problem $A\boldsymbol{x} = \boldsymbol{b}$.

Minimization problem, Example

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}$$

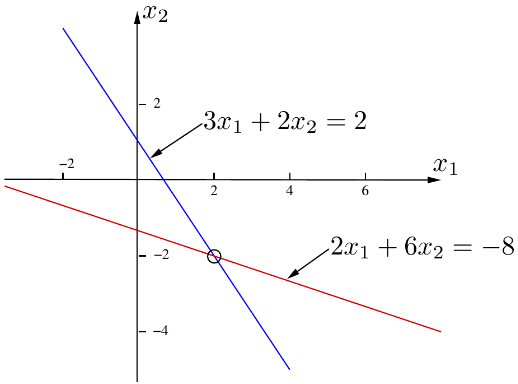
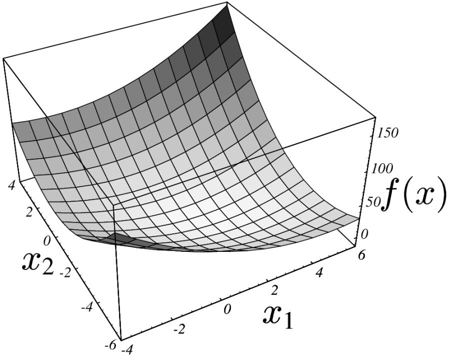


Figure: Linear equations with solution at the intersection of both lines.

Minimization problem, Example



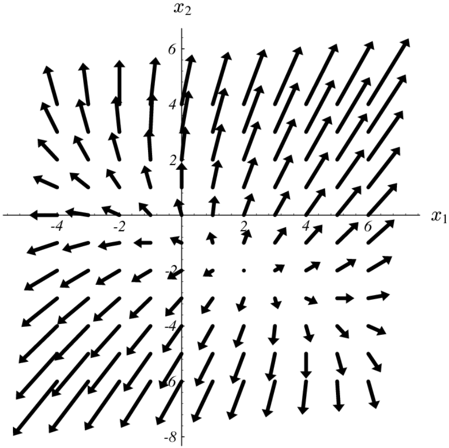


Figure: Quadratic Form $f(x)$ and gradients $f'(x)$

Method of steepest descent

Basic idea: From point x_m take one step in the direction of the negative gradients at the point $\boldsymbol{x}^{(m)}$ $(-f'(\boldsymbol{x}^{(m)})$.

- Use a step length, until f will be minimal along the search direction
- From equation (3) it follows, that $-f'(\boldsymbol{x}^{(m)}) = \boldsymbol{r}^{(m)}$

$$\underline{\text{Start:}}\ \boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$$

$$\textbf{until}\ \|\boldsymbol{r}^{(m)}\| > \epsilon:$$

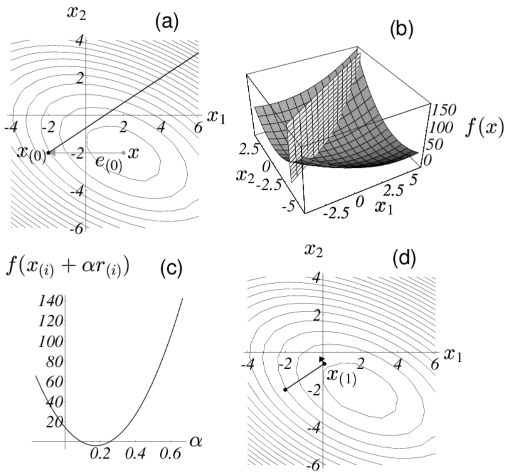
$$\text{Find } \alpha, \text{ such that } f(\boldsymbol{x}^{(m)} + \alpha \boldsymbol{r}^{(m)}) \text{ minimal}$$

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

$$\boldsymbol{r}^{(m+1)} = \boldsymbol{b} - A\boldsymbol{x}^{(m+1)}$$

- The residual is a measure for both the *error* and *the search direction!*

Method of steepest descent



Method of steepest descent

Minimize f along the search vector $\boldsymbol{p}^{(m)}$:

Approach:

$$
\begin{aligned}
f(\boldsymbol{x}^{(m+1)}) &= f(\boldsymbol{x}^{(m)} + \alpha \boldsymbol{p}^{(m)}) \stackrel{!}{=} \min \\
\Longleftrightarrow \quad \tfrac{d}{d\alpha} f(\boldsymbol{x}^{(m+1)}) &= 0
\end{aligned}
\tag{4}
$$

with the chain rule

$$
\begin{aligned}
\frac{d}{d\alpha} f(\boldsymbol{x}^{(m+1)}) &= \langle f'(\boldsymbol{x}^{(m+1)}), \frac{d}{d\alpha}(\boldsymbol{x}^{(m)} + \alpha \boldsymbol{p}^{(m)}) \rangle \\
&= \langle f'(\boldsymbol{x}^{(m+1)}), \boldsymbol{p}^{(m)} \rangle \\
&= \langle \boldsymbol{r}^{(m+1)}, \boldsymbol{p}^{(m)} \rangle
\end{aligned}
$$

for steepest descent we will use $\boldsymbol{p}^{(m)} = \boldsymbol{r}^{(m)}$. The more general (and interesting) case with $\boldsymbol{p}^{(m)} \neq \boldsymbol{r}^{(m)}$ will be discussed later.

Method of steepest descent

We search for an α, such that $\boldsymbol{r}^{(m+1)} \perp \boldsymbol{p}^{(m)}$.

Method of steepest descent

We search for an α, such that $\boldsymbol{r}^{(m+1)} \perp \boldsymbol{p}^{(m)}$.

Rearranging for α:

$$
\begin{aligned}
\langle \boldsymbol{r}^{(m+1)}, \boldsymbol{p}^{(m)} \rangle &= 0 \\
\langle b - A\boldsymbol{x}^{(m+1)}, \boldsymbol{p}^{(m)} \rangle &= 0 \\
\langle b - A(\boldsymbol{x}^{(m)} + \alpha \boldsymbol{p}^{(m)}), \boldsymbol{p}^{(m)} \rangle &= 0 \\
\langle b - A\boldsymbol{x}^{(m)}, \boldsymbol{p}^{(m)} \rangle - \alpha \langle A\boldsymbol{p}^{(m)}, \boldsymbol{p}^{(m)} \rangle &= 0 \\
\langle b - A\boldsymbol{x}^{(m)}, \boldsymbol{p}^{(m)} \rangle &= \alpha \langle A\boldsymbol{p}^{(m)}, \boldsymbol{p}^{(m)} \rangle \\
\langle \boldsymbol{r}^{(m)}, \boldsymbol{p}^{(m)} \rangle &= \alpha \langle A\boldsymbol{p}^{(m)}, \boldsymbol{p}^{(m)} \rangle \\
\alpha &= \frac{\langle \boldsymbol{r}^{(m)}, \boldsymbol{p}^{(m)} \rangle}{\langle A\boldsymbol{p}^{(m)}, \boldsymbol{p}^{(m)} \rangle}
\end{aligned}
$$

Method of steepest descent

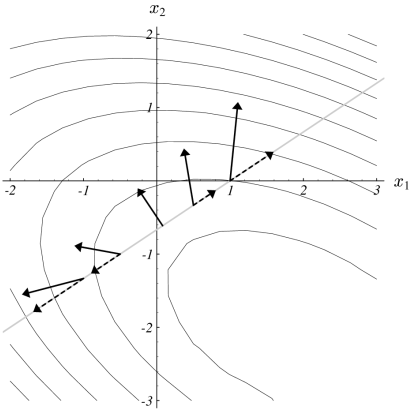


Figure: The gradients along the search vector

Method of steepest descent

- We can avoid the matrix-vector product $A\boldsymbol{x}^{(m)}$ for the computation of $\boldsymbol{r}$ since $\boldsymbol{r}^{(m+1)} = \boldsymbol{r}^{(m)} - \alpha^{(m)} A\boldsymbol{r}^{(m)}$
- The final algorithm will be:

$$\underline{\text{Start:}}\ \boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$$

$$\textbf{until}\ \|\boldsymbol{r}^{(m)}\| > \epsilon:$$

$$\alpha^{(m)} = \frac{\langle \boldsymbol{r}^{(m)}, \boldsymbol{r}^{(m)} \rangle}{\langle A\boldsymbol{r}^{(m)}, \boldsymbol{r}^{(m)} \rangle}$$

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

$$\boldsymbol{r}^{(m+1)} = \boldsymbol{r}^{(m)} - \alpha^{(m)} A\boldsymbol{r}^{(m)}$$

Method of steepest descent

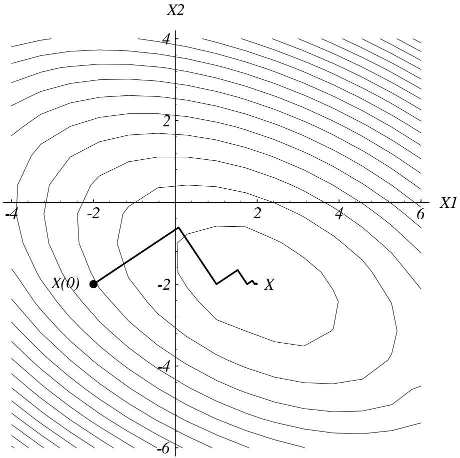


Figure: Method of steepest descent with $\boldsymbol{x}^{(0)} = (-2 \ \ -2)^T$

Method of steepest descent, Convergence

For the 2-dimensional case we can show that

$$\|e^{(m+1)}\|_A^2 = \omega^2 \|e^{(m)}\|_A^2$$

with

$$\omega^2 = 1 - \frac{(\kappa^2 + \mu^2)^2}{(\kappa + \mu^2)(\kappa^3 + \mu^2)}$$

with $\|e\|_A = (e^T A e)^{1/2}$ the so-called A-norm and μ the gradient of the error within the coordinate system spanned by the eigenvectors.

For well-conditioned systems is $\kappa \approx 1$ (eigenvalue are close to each other). In these cases, the algorithm will converge rapidly (see Shewchuk)

Observation: For ill-conditioned systems we will use search directions several times.

Convergence of steepest descent

Consider the functional $\boldsymbol{f}(x) = \frac{1}{2} < A\boldsymbol{x}, \boldsymbol{x} > - < \boldsymbol{b}, \boldsymbol{x} >$, which we would like to minimize to compute the solution of $A\boldsymbol{x} = \boldsymbol{b}$.

Example: $n = 2$

A has 2 orthonormal eigenvectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ with associated eigenvalues λ_1 and λ_2. We can write each vector $\boldsymbol{x}$ as a linear combination of eigenvectors $\boldsymbol{x} = a_1 \cdot \boldsymbol{v}_1 + a_2 \cdot \boldsymbol{v}_2, \quad a_1, a_2 \in \mathbb{R}$.

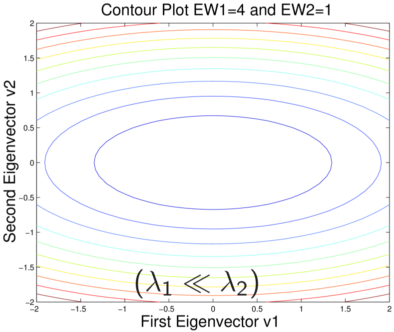
The following equation holds ($\boldsymbol{b} = 0$):

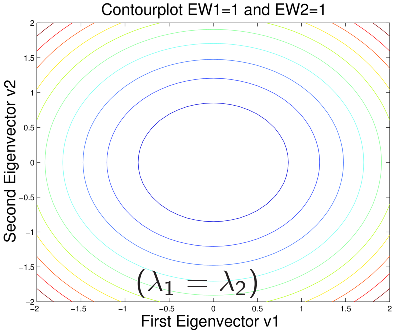
$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} = \frac{1}{2}(a_1\boldsymbol{v}_1 + a_2\boldsymbol{v}_2)(a_1\lambda_1\boldsymbol{v}_1 + a_2\lambda_2\boldsymbol{v}_2) \tag{5}$$

$$= \frac{1}{2}(a_1^2\lambda_1 + a_2^2\lambda_2) \tag{6}$$

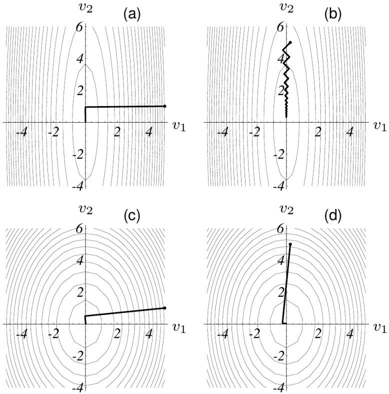
Convergence of steepest descent

Contour lines of the functional:



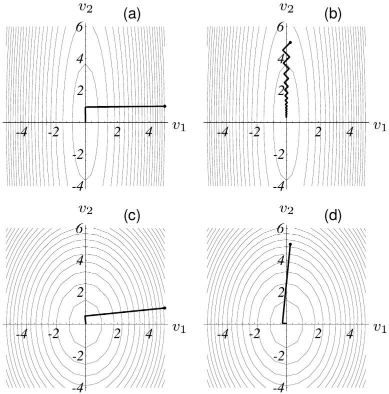


Impact on the convergence



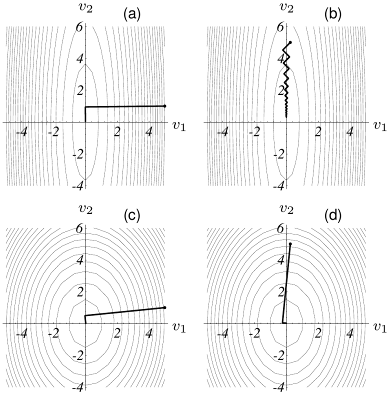
- (a) Large conditioning number, good starting point $\longrightarrow$ by accident fast convergence using steepest descent

Impact on the convergence



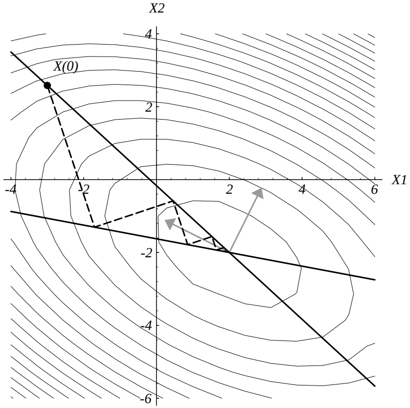
- (a) Large conditioning number, good starting point $\longrightarrow$ by accident fast convergence using steepest descent (b) Large conditioning number, bad starting point $\longrightarrow$ very slow convergence

Impact on the convergence



- (a) Large conditioning number, good starting point $\longrightarrow$ by accident fast convergence using steepest descent (b) Large conditioning number, bad starting point $\longrightarrow$ very slow convergence (c-d) Small conditioning number $\longrightarrow$ good convergence independent of the starting vector $\longrightarrow$ Preconditioning

Choice of the search direction



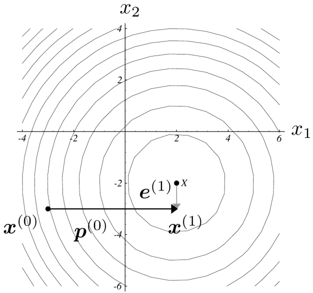
The continuous line are the set of points for the the steepest descent method has the worth convergence. The dotted line shows the search direction of the steepest descent method. Note, that if one choose such a direction then this direction will be reused in the following iterations.

«Conjugate Gradient Method»

Conjugate Gradient Search Directions

Basic idea: We only use *maximal n* orthogonal search directions. In each iteration, we will make one optimal search step.

- *Approach:* We define n orthogonal search directions $\boldsymbol{p}^{(0)}$ until $\boldsymbol{p}^{(n-1)}$ (e.g the coordinate axis)
- The minimization criteria is the orthogonality of $\boldsymbol{e}^{(m+1)}$, with: $\boldsymbol{p}^{(m)} \perp \boldsymbol{e}^{(m+1)}$



Idea

- Using this step we eliminate the error component of this particular search direction
- If $\boldsymbol{p}^{(m)} \perp \boldsymbol{e}^{(m+1)}$, then the scalar-product must be $= 0$:

$$
\begin{aligned}
\langle \boldsymbol{p}^{(m)}, \boldsymbol{e}^{(m+1)} \rangle &= 0 && (7)\\
\langle \boldsymbol{p}^{(m)}, \boldsymbol{e}^{(m)} + \alpha^{(m)} \boldsymbol{p}^{(m)} \rangle &= 0 && (8)\\
\alpha^{(m)} &= -\frac{\langle \boldsymbol{p}^{(m)}, \boldsymbol{e}^{(m)} \rangle}{\langle \boldsymbol{p}^{(m)}, \boldsymbol{p}^{(m)} \rangle} && (9)
\end{aligned}
$$

- This is not really helpful since we do not know $\boldsymbol{e}^{(m)}$!

Conjugate Search Direction

One solution is, instead of orthogonal direction, we will use *A-orthogonal* or *conjugate* vectors.

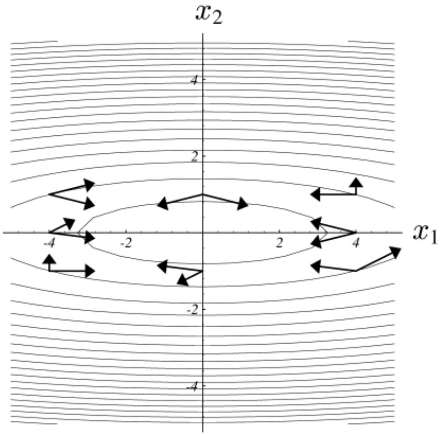
Definition: Two vectors $\boldsymbol{a}, \boldsymbol{b}$ are *A-orthogonal* or *conjugate* if and only if:

$$\boldsymbol{a}^T A \boldsymbol{b} = \langle \boldsymbol{a}, A\boldsymbol{b} \rangle = 0 \quad \Longleftrightarrow \quad \boldsymbol{p}^{(m)} \perp_A \boldsymbol{e}^{(m+1)}$$

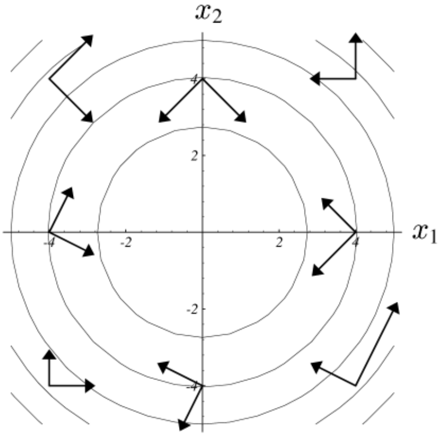
With $\boldsymbol{p}^{(m)} \perp_A \boldsymbol{e}^{(m+1)}$ (and $\boldsymbol{r}^{(m)} = -A\boldsymbol{e}^{(m)}$):

$$\begin{aligned}
\langle \boldsymbol{p}^{(m)}, A\boldsymbol{e}^{(m+1)} \rangle &= 0 && (10)\\
\langle \boldsymbol{p}^{(m)}, A\boldsymbol{e}^{(m)} + \alpha^{(m)} A\boldsymbol{p}^{(m)} \rangle &= 0 && (11)\\
\alpha^{(m)} &= -\frac{\langle \boldsymbol{p}^{(m)}, A\boldsymbol{e}^{(m)} \rangle}{\langle \boldsymbol{p}^{(m)}, A\boldsymbol{p}^{(m)} \rangle} && (12)\\
&= \frac{\langle \boldsymbol{p}^{(m)}, \boldsymbol{r}^{(m)} \rangle}{\langle \boldsymbol{p}^{(m)}, A\boldsymbol{p}^{(m)} \rangle} && (13)
\end{aligned}$$

Conjugate Search Direction

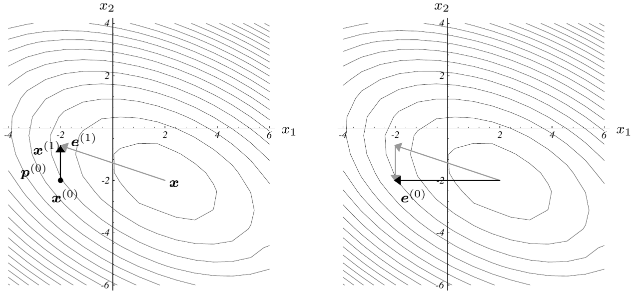


A-Conjugate vectors



rectified vectors

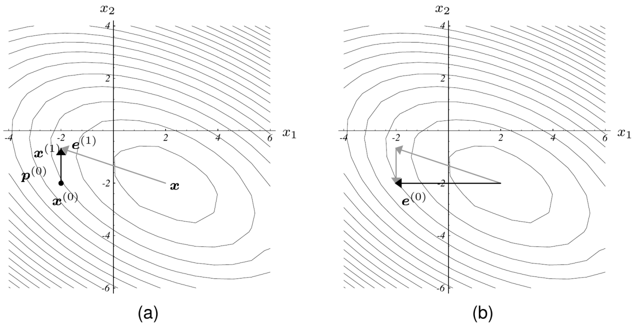
Elimination of A-orthogonal components



(a) (b)

(a) the first point $\boldsymbol{x}^{(1)}$ will be computed such that $\boldsymbol{e}^{(1)}$ is A-orthogonal to $\boldsymbol{p}^{(0)}$.

Elimination of A-orthogonal components



(a) the first point $\boldsymbol{x}^{(1)}$ will be computed such that $\boldsymbol{e}^{(1)}$ is A-orthogonal to $\boldsymbol{p}^{(0)}$. (b) the error $\boldsymbol{e}^{(0)}$ can be express as a sum of A-orthogonal components (gray arrows). In each iteration we will eliminated such a component.

Conjugate Directions

Condition.: We would like to minimize $f(x^{(m)} + \alpha p^{(m)}) \stackrel{!}{=} min$ (see (15))

The algorithm must satisfy the following iteration:

Start: $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$

For all $m = 1, \ldots, n$:

Select search conjugate vector $p^{(m)}$
to all previous computed $p^{(l)}, l < m$

$$\alpha^{(m)} = \frac{\langle \boldsymbol{r}^{(m)}, \boldsymbol{p}^{(m)} \rangle}{\langle A\boldsymbol{p}^{(m)}, \boldsymbol{p}^{(m)} \rangle}$$

$$\boldsymbol{x}^{(m+1)} = \boldsymbol{x}^{(m)} + \alpha^{(m)} \boldsymbol{p}^{(m)}$$

$$\boldsymbol{r}^{(m+1)} = \boldsymbol{r}^{(m)} - \alpha^{(m)} A\boldsymbol{p}^{(m)}$$

Question: How to compute n A-orthogonal search vectors?

Conjugate Gradients

Basic idea: Use the residual to construct the next conjugate search direction.

with

$$\boldsymbol{p}^{(m+1)} = \boldsymbol{r}^{(m+1)} + \underbrace{\beta^{(m+1)}}_{\text{search for}} \boldsymbol{p}^{(m)} \tag{14}$$

and $\underbrace{\langle \boldsymbol{p}^{(m+1)}, A\boldsymbol{p}^{(m)} \rangle = 0}_{\text{A-Orthogonal}}$ we obtain:

$$\begin{aligned}
0 &= \langle \boldsymbol{p}^{(m+1)}, A\boldsymbol{p}^{(m)} \rangle \\
&= \langle \underbrace{\boldsymbol{r}^{(m+1)} + \beta^{(m+1)} \boldsymbol{p}^{(m)}}_{(14) \text{ used}}, A\boldsymbol{p}^{(m)} \rangle \\
&= \langle \boldsymbol{r}^{(m+1)}, A\boldsymbol{p}^{(m)} \rangle + \beta^{(m+1)} \langle \boldsymbol{p}^{(m)}, A\boldsymbol{p}^{(m)} \rangle \\
\beta^{(m+1)} &= -\frac{\langle \boldsymbol{r}^{(m+1)}, A\boldsymbol{p}^{(m)} \rangle}{\langle \boldsymbol{p}^{(m)}, A\boldsymbol{p}^{(m)} \rangle}
\end{aligned}$$

Conjugate Gradients

- the *initial-search vector* is $\boldsymbol{r}^{(0)}$ (similar to the steepest descent)
- similar to the steepest descent we can eliminate the additional matrix-vector product (exercise):

$$\beta^{(m+1)} = -\frac{\langle \boldsymbol{r}^{(m+1)}, A\boldsymbol{p}^{(m)} \rangle}{\langle \boldsymbol{p}^{(m)}, A\boldsymbol{p}^{(m)} \rangle} = \frac{\langle \boldsymbol{r}^{(m+1)}, \boldsymbol{r}^{(m+1)} \rangle}{\langle \boldsymbol{r}^{(m)}, \boldsymbol{r}^{(m)} \rangle}$$

- The method how to construct the n conjugate vectors is the *Gram-Schmidt process*.
- The iterative construction of the search space is of the form

$$U_m := \mathrm{span}\{\boldsymbol{p}^{(0)}, A\boldsymbol{p}^{(0)}, A^2\boldsymbol{p}^{(0)}, \dots, A^n\boldsymbol{p}^{(0)}\}$$

This space is called *Krylov-Subspace*.

Conjugate Gradients

$$
\begin{aligned}
&\underline{\text{Start:}}\ \boldsymbol{r}^{(0)} := \boldsymbol{b} - A\boldsymbol{x}^{(0)} \text{ with } \boldsymbol{x}^{(0)} \text{ arbitrary} \\
&\qquad\ \boldsymbol{p}^{(0)} := \boldsymbol{r}^{(0)} \\
&\textbf{for all } m = 1, \ldots, n-1: \\
&\qquad \alpha^{(m)} = \frac{\langle \boldsymbol{r}^{(m)}, \boldsymbol{p}^{(m)} \rangle}{\langle A\boldsymbol{p}^{(m)}, \boldsymbol{p}^{(m)} \rangle} \\
&\qquad \boldsymbol{x}^{(m+1)} = \boldsymbol{x}^{(m)} + \alpha^{(m)} \boldsymbol{p}^{(m)} \\
&\qquad \boldsymbol{r}^{(m+1)} = \boldsymbol{r}^{(m)} - \alpha^{(m)} A\boldsymbol{p}^{(m)} \\
&\qquad \beta^{(m+1)} = \frac{\langle \boldsymbol{r}^{(m+1)}, \boldsymbol{r}^{(m+1)} \rangle}{\langle \boldsymbol{r}^{(m)}, \boldsymbol{r}^{(m)} \rangle} \\
&\qquad \boldsymbol{p}^{(m+1)} = \boldsymbol{r}^{(m+1)} + \beta^{(m+1)} \boldsymbol{p}^{(m)}
\end{aligned}
$$

Conjugate-Gradients algorithm (Hestenes & Stiefel 1952).

Conjugate Gradients (2)

Elements of the algorithm:

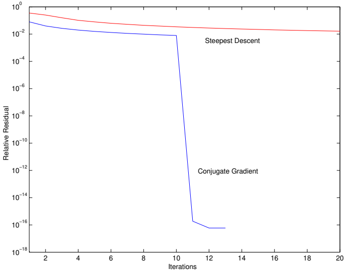
- $\boldsymbol{x}^{(m)}$: Actual solution
- $\boldsymbol{r}^{(m)}$: residual of the actual solution
- $\boldsymbol{p}^{(m)}$: Search direction
- $\alpha^{(m)}$: Optimal step length (factor) in the search direction $\boldsymbol{p}^{(m)}$
- $\beta^{(m+1)}$: Factor for $\boldsymbol{p}^{(m)}$ to compute from $\boldsymbol{p}^{(m)}$ and $\boldsymbol{r}^{(m+1)}$ a new search direction $\boldsymbol{p}^{(m+1)}$ which is A-orthogonal to $\boldsymbol{p}^{(m)}$.

CG, Theory and Praxis

- The exact solution will be computed after n iterations
- As a result CG is actually a direct method
- In praxis we need much less iterations since we only need to compute an approximate solution.

Test example

Convergence of steepest descent and conjugate gradients for an example $Ax = b$ with $A = tridiag(-1, 2, -1) \in \mathbb{R}^{20\times 20}$



(After exact computation the CG-methods with $n = dim(A)$ iterations will converge against the exact solution.)

Other popular Krylov Subspace Methods

Book: «Templates for the Solution of Linear System»
http://www.netlib.org/linalg/html_templates/Templates.html

Some popular iterative Krylov Subspace methods:

Name		Condition
MinRES	Minimal Residual	$A = A^T$, A indef.
CG	Conjugate Gradient	$A = A^T$, A s.p.d
QMR	Quasi-Minimal Residual	unsymmetric
BICGSTAB	Biconjugate Gradient Stabilized	unsymmetric
CGS	Conjugate Gradient Square	unsymmetric
GMRES	Generalized Minimal Residual	unsymmetric

« Preconditioning »

Preconditioning

Basic idea: Improve the conditioning number of A through a multiplication with a "preconditioner" M: $\kappa(M^{-1}A) \ll \kappa(A)$

- Solve equivalent problem: $M^{-1}A\boldsymbol{x} = M^{-1}\boldsymbol{b}$
- M should be easy to invert
- In order to use the CG method, the resulting matrix $M^{-1}A$ must be symmetric positive definite
- We can M in the following form $EE^T = M$, so that we can transform the problem $A\boldsymbol{x} = \boldsymbol{b}$ into

$$E^{-1}AE^{-T}\hat{\boldsymbol{x}} = E^{-1}\boldsymbol{b}, \quad \hat{\boldsymbol{x}} = E^T\boldsymbol{x} \tag{15}$$

  where the matrix $E^{-1}AE^{-T}$ is *spd*

Preconditioning

Here are a few method

- *Diagonal preconditioning*. Choose: $M = \mathrm{diag}(A)$
- *Incomplete LU-Decomposition*. Choose: $M = LU = A - R$.
    - Important that the Fill-In and the time for the factorization will be small.

Literature

- Jonathan R. Shewchuk, *An Introduction to the Conjugate Gradient method without the Agonizing Pain*, 1994. http://www-2.cs.cmu.edu/~jrs/jrspapers.html#cg
- Book: «Templates for the Solution of Linear System» http://www.netlib.org/linalg/html_templates/Templates.html
- James W. Demmel, *Applied Numerical Linear Algebra*, Siam 1997