The Conjugate Gradient Method

Christian Feichtinger

Friedrich-Alexander Universität Erlangen-Nürnberg

8.12.06

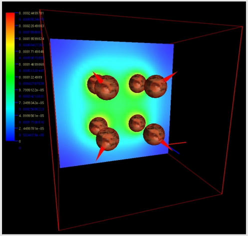
Outline

Motivation

Colloids

- Colloids consist from a continuous phase and a dispersed material
- Aim: Control of the agglomeration or the stabilization
- Possible solution: Electrostatic stabilization
- One aspect: Calculation of the potential

Examples

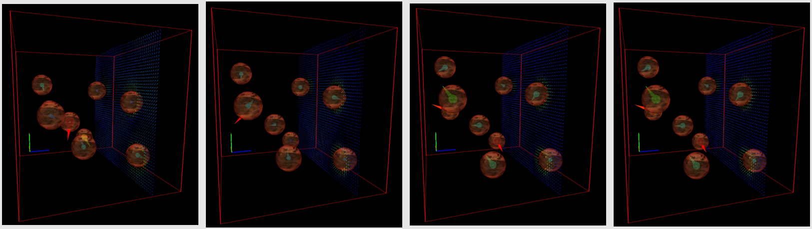


Motivation

Colloids

- Colloids consist from a continuous phase and a dispersed material
- Aim: Control of the agglomeration or the stabilization
- Possible solution: Electrostatic stabilization
- One aspect: Calculation of the potential

Stabilization

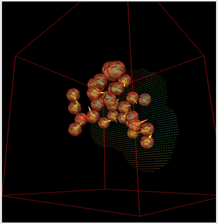


Motivation

Colloids

- Colloids consist from a continuous phase and a dispersed material
- Aim: Control of the agglomeration or the stabilization
- Possible solution: Electrostatic stabilization
- One aspect: Calculation of the potential

Agglomeration



- Electrostatic interaction influences the structure of the agglomerate
- In drying processes:
- Higher electrostatic repulsion results in dense agglomerates
- Lower repulsion in porous structures

Motivation

Problem

Solve linear system of equations: $\mathbf{A}\vec{x} = \vec{b}$

Motivation

Problem

Solve linear system of equations: $\mathbf{A}\vec{x} = \vec{b}$

Definitions

- $\mathbf{A}$ is a square, symmetric, positive-definite matrix
  - symmetric: $\mathbf{A} = \mathbf{A}^T$
  - positive-definite: $\vec{x}^T\mathbf{A}\vec{x} > 0 \quad \forall \vec{x} \neq 0$
- $\vec{x}$ unknown vector
- $\vec{b}$ right hand side
- restriction for simpification only
- other matricies need extensions to the algorithm

Motivation

Problem

Solve linear system of equations: $\mathbf{A}\vec{x} = \vec{b}$

Solutions

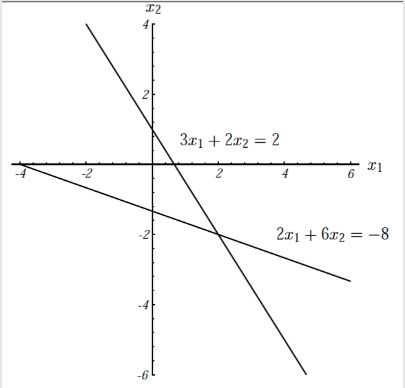
- Direct Methods
  - Gaussian Elimination
  - LU Factorization
- Iterative Methods
  - Gauss-Seidel
  - SOR
  - Multigrid
  - Conjugate Gradient

Linear System Example

Example

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

$$\begin{aligned} 3x_1 + 2x_2 &= 2 \\ 2x_1 + 6x_2 &= -8 \end{aligned}$$



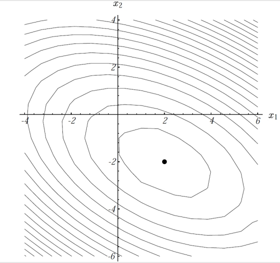
From Linear Equation to a Minimizing Problem

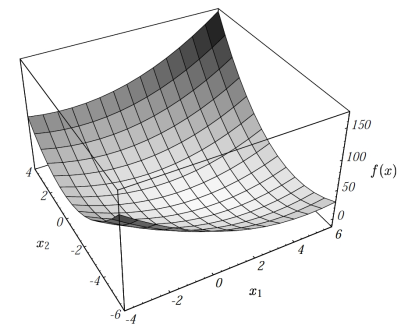
Quadratic Form

$$f(\vec{x}) = 0.5\vec{x}^T \mathbf{A}\vec{x} - b^T\vec{x} + c$$

Example

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





From Linear Equation to a Minimizing Problem

Quadratic Form

$$f(\vec{x}) = 0.5\vec{x}^T\mathbf{A}\vec{x} - b^T\vec{x} + c$$

Minimum Search

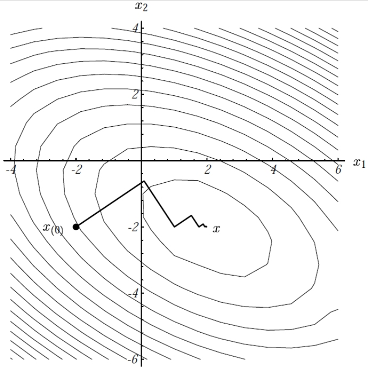
$$\begin{aligned}\vec{f}'(\vec{x}) &= 0.5\,\mathbf{A}^T\vec{x} + 0.5\,\mathbf{A}\vec{x} - \vec{b} \\ \vec{f}'(\vec{x}) &= \mathbf{A}\vec{x} - \vec{b}\ \ (by\ symmetry\ of\ \mathbf{A}) \\ 0 &= \mathbf{A}\vec{x} - \vec{b}\ \ (\vec{f}'(\vec{x}) = 0)\end{aligned}$$

Idea of CG

Find the minimum of $f(\vec{x})$ instead of solving the linear system $\mathbf{A}\vec{x} = \vec{b}$

Steepest Descent

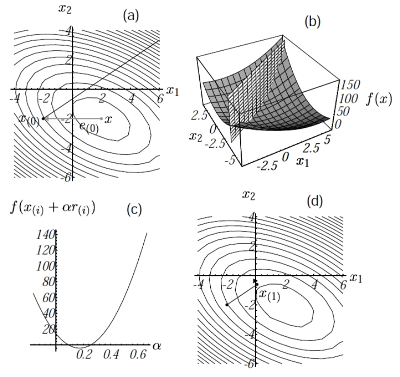
Algorithm



- Take a series of steps: $\vec{x}_{(0)}$, $\vec{x}_{(1)}$,..., $\vec{x}_{(m)}$
- Go in direction of steepest descent of $f(\vec{x}_{(i)}) \rightarrow -\vec{f}'(\vec{x}_{(i)})$
- $\vec{x}_{(i+1)} = \vec{x}_{(i)} + \alpha(-1)\vec{f}'(\vec{x}_{(i)})$
- $\vec{r}_{(i)} = \vec{b} - \mathbf{A}\vec{x}_{(i)} = -\vec{f}'(\vec{x}_i)$
- Done when $\vec{r}_{(i)}$ is small enough

Determination of α

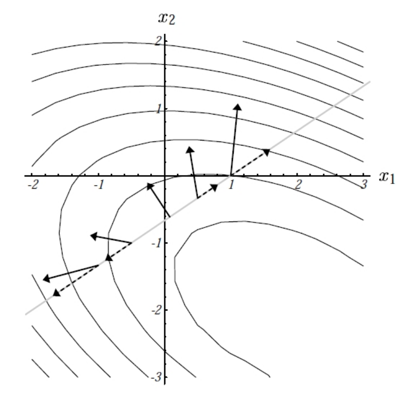
Principle



- Aim: Minimize f
- Go in direction $\vec{r}_{(i)}$ until the minimum of f along the search direction is reached
- $\frac{\partial}{\partial\alpha} f(\vec{x}_{(i+1)}) = 0$
- $\alpha = \frac{\vec{r}_{(0)}^T \vec{r}_{(0)}}{\vec{r}_{(0)}^T \mathbf{A} \vec{r}_{(0)}}$

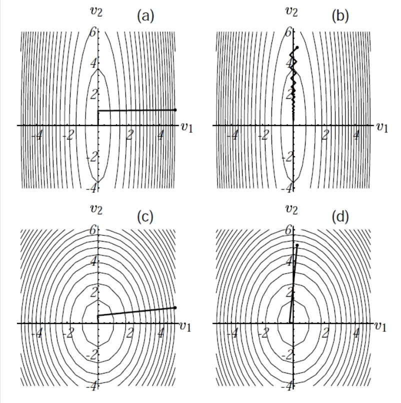
Determination of α

Principle



- At $\vec{x}_{(i+1)}$: $\vec{r}^{\,T}_{(i+1)}\vec{d}_{(i)} = 0$
- Also valid for CD and CG
- For SD: $\vec{x}_{(i+1)}$: $\vec{r}^{\,T}_{(i+1)}\vec{r}_{(i)} = 0$

Convergence



- Energy Norm: $\|\vec{e}_{(i)}\|_{\mathbf{A}}^2 = \vec{e}_{(i)}^T \mathbf{A} \vec{e}_{(i)}$
- $\|\vec{e}_{(i+1)}\|_{\mathbf{A}}^2 < \|\vec{e}_{(i)}\|_{\mathbf{A}}^2$
- Bad convergence due to steps in same search direction
- Although $\vec{r}_{(i+1)}^T \vec{r}_{(i)} = 0$
- $\vec{r}_{(i+1)}$ must not be orthogonal to $\vec{r}_{(i-1)}$

Conjugate Directions

Concept

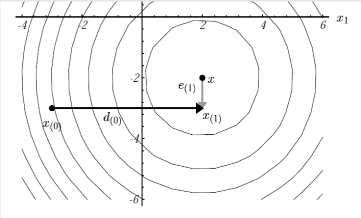
- Choose search direction $\vec{d}_{(i)}$ in a way, that it is linear independant to all previous $\vec{d}_{(j)}$

Conjugate Directions

Side Remark

- Decomposition of $\vec{e}_{(0)}$ in n orthogonal, linear independent vectors
- Use the orthogonal search directions $\vec{d}_{(i)}$
- $\vec{e}_{(0)} = \sum_{i=0}^{n-1} \alpha_i \vec{d}_i \quad \vec{e}_{(0)} \in R^n$

Example



$$\vec{e}_{(0)} = -5\vec{d}_0 - 1\vec{d}_1 \quad \vec{e}_{(1)} = -1\vec{d}_1$$

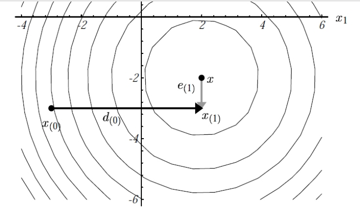
$$\vec{d}_{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \vec{d}_{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Conjugate Directions

Concept

- Choose search direction $\vec{d}_{(i)}$ in a way, that it is linear independant to all previous $\vec{d}_{(j)}$
- Step in each direction only once
- Eliminate the error component $\alpha\vec{d}_{(i)}$ of $\vec{e}_{(0)}$ in iteration i

Example



$$\vec{e}_{(0)} = -5\vec{d}_0 - 1\vec{d}_1 \quad \vec{e}_{(1)} = -1\vec{d}_1$$

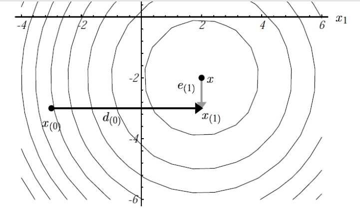
$$\vec{d}_{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \vec{d}_{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Conjugate Directions

Concept

- Bad convergence of SD avoided
- Done after n-steps for $\vec{x} \in R^n$ (Exact Solution)

Example



$$\vec{e}_{(0)} = -5\vec{d}_0 - 1\vec{d}_1 \quad \vec{e}_{(1)} = -1\vec{d}_1$$

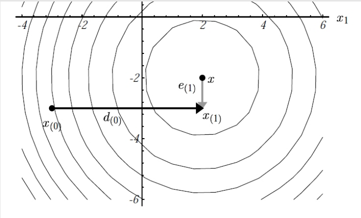
$$\vec{d}_{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \vec{d}_{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Conjugate Directions

First Idea

- Use as search vectors the coordinate axes $\vec{u}_i$
- $\alpha_{(i)} = -\frac{\vec{d}_{(i)}^T \vec{e}_{(i)}}{\vec{d}_{(i)}^T \vec{d}_{(i)}} \longrightarrow$ not possible!

Example



$$\vec{e}_{(0)} = -5\vec{d}_0 - 1\vec{d}_1 \quad \vec{e}_{(1)} = -1\vec{d}_1$$

$$\vec{d}_{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \vec{d}_{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Conjugate Directions

First Idea

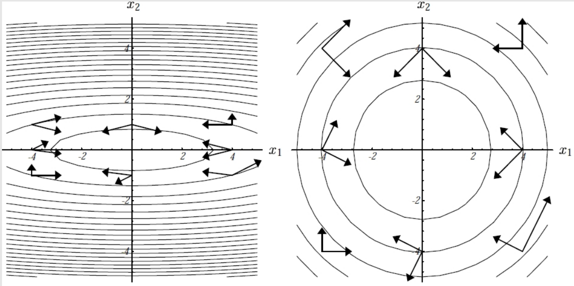
- Use as search vectors the coordinate axes $\vec{u}_i$
- $\alpha_{(i)} = -\frac{\vec{d}_{(i)}^T \vec{e}_{(i)}}{\vec{d}_{(i)}^T \vec{d}_{(i)}} \rightarrow$ not possible!

A-Orthogonal Approach

- Expand Equation for α with $\mathbf{A}$
- $\vec{r}_{(i)} = -\mathbf{A}\vec{e}_{(i)}$
- $\alpha_{(i)} = \frac{\vec{d}_{(i)}^T \vec{r}_{(i)}}{\vec{d}_{(i)}^T \mathbf{A} \vec{d}_{(i)}}$
- Search directions have to be A-orthogonal
- $\vec{d}_{(i)} \mathbf{A} \vec{d}_{(j)} = 0 \ \forall \ j \neq i$
- Search directions still linearly independent
- In iteration i the error component $\alpha \vec{d}_{(i)}$ is removed from $\vec{e}_{(i)}$

A-Orthogonality

Example



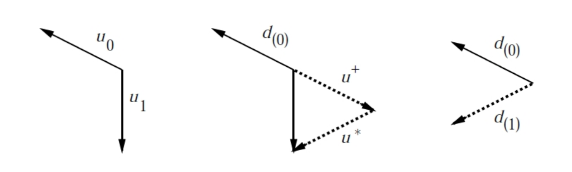
Determination of $\vec{d}_{(i)}$

Task

Search directions have to be A-orthogonal to all previous search directions

Solution: Gram-Schmidt Conjugation

- Use coordinate axes $\vec{u}_i$ to construct $\vec{d}_{(i)}$
- Remove the parts of $\vec{u}_i$, that are parallel to any previous search direction
- $\vec{d}_{(i)} = \vec{u}_i + \sum_{k=0}^{i-1} \beta_{ik} \vec{d}_{(k)}$
- β specifies the length of the parallel part



Computational Cost and Stability

Calculation of $\vec{d}_{(i)}$

$$\vec{d}_{(i)} = \vec{u}_i + \sum_{k=0}^{i-1} \beta_{ik} \vec{d}_{(k)}$$

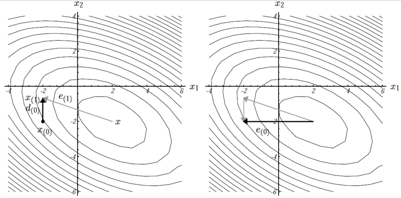
Gram-Schmidt Conjugation

- Calculation of β : $O(n^2)$ operations (Matrix-Vector multiplication)
- $O(n^3)$ operations needed for full set
- β depends on the current search direction and the previous
- Previous search directions have to be stored
- Roundoff errors may cause the search vectors to loose A-orthogonality

Conjugate Directions

Algorithm

- Use $\vec{u}_0$ as initial search direction
- For $i = 0$ *to* $n-1$
  - Compute $\vec{r}_{(i)} = \vec{b} - \mathbf{A}\vec{x}_{(i)}$
  - Compute $\alpha_{(i)} = \frac{\vec{d}_{(i)}^T \vec{r}_{(i)}}{\vec{d}_{(i)}^T \mathbf{A}\vec{d}_{(i)}}$
  - $\vec{x}_{(i+1)} = \vec{x}_{(i)} + \alpha_{(i)}\vec{d}_{(i)}$
  - Compute new direction $\vec{d}_{(i+1)}$



Conjugate Gradient

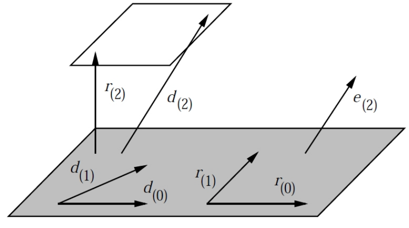
Problem

Improve the computation of the search directions

Solution

Construct $\vec{d}_{(i)}$ from $\vec{r}_{(i)}$ instead of $\vec{u}_i$

Example



Conjugate Gradient

Problem

Improve the computation of the search directions

Solution

Construct $\vec{d}_{(i+1)}$ from $\vec{r}_{(i+1)}$ instead of $\vec{u}_{i+1}$

Implications

- Previous search directions $\vec{d}_{(j)}$ are not needed to make $\vec{d}_{(i+1)}$ A-orthogonal $\quad i > j$
- $\vec{r}_{(i+1)}$ already A-orthogonal to $\vec{d}_{(j)}$ $i > j$
- $\vec{r}_{(i+1)}^T \vec{d}_{(j)} = 0 \quad i \geq j$
- $\vec{r}_{(i)}^T \vec{r}_{(j)} = 0 \quad i \neq j$ (needed for Gram-Schmidt Conjugation)

Equations

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

Conjugate Gradient

Algorithm

- $\vec{d}_{(0)} = \vec{r}_{(0)} = \vec{b} - \mathbf{A}\vec{x}_{(0)}$
- for (($i = 0$ *to* $n - 1$)
  - Compute α
  - Determine the new position $\vec{x}_{(i+1)}$
  - Calculate the next residual $\vec{r}_{(i+1)}$
  - Determine $\beta_{(i+1)}$
  - Compute the new search direction $\vec{d}_{(i+1)}$

Optimality of the Error Term

Statement

CG finds the best solution within the bonds of where it has been allowed to explore

Where it has been allowed to explore?

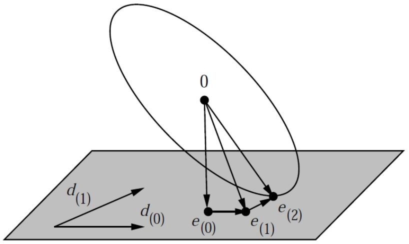
- CG is allowed to choose the error $\vec{e}_{(i)}$ from $\vec{e}_{(0)} + D_i$
- D_i is the subspace, that is spanned by the search vectors $\vec{d}_{(0)}, \ldots, \vec{d}_{(i-1)}$
- $\vec{e}_{(i)} = \vec{e_{(0)}} + \textit{linear combination of the search vectors}$

Best solution?

CG chooses the error $\vec{e}_{(i)}$, that it minimizes $\|\vec{e}_{(i)}\|_A$

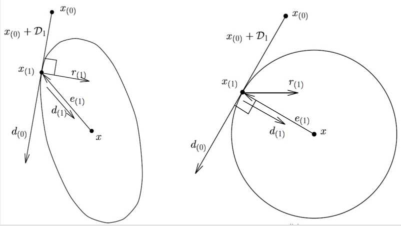
Optimality of the Error Term

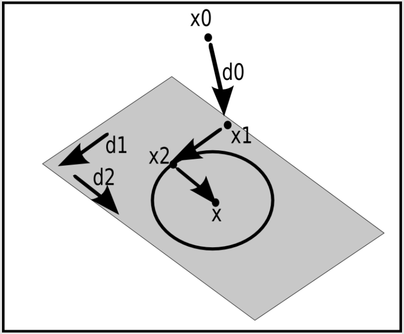
Example



Best Solution

Example





Complexity of CG

Max Iterations

- Reduce $\|\vec{e}_{(i)}\| \leq \epsilon \|\vec{e}_{(0)}\|$
- Max number of iterations: $i \leq \left\lceil 0.5\sqrt{\kappa} \ln \frac{2}{\epsilon} \right\rceil$

Dominating operations

- Matrix / Vector multiplications: O(m)
- Sparse matrices: O(n)
- m = number of non-zero entries in the matrix
- n = length of vector

Complexity

For sparse matrices: $O(\sqrt{\kappa}n)$

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}}$$

Preconditioning

Idea

- Complexity depends on $\kappa = \frac{\lambda_{max}}{\lambda_{min}} \geq 1$
- Improve condition number κ
- The lower the condition number, the more spherical the paraboloid is

Optimization

- Solve $\mathbf{M}^{-1}\mathbf{A}\vec{x} = \mathbf{M}^{-1}\vec{b}$
- $\kappa(\mathbf{M}^{-1}\mathbf{A}) \ll \kappa(\mathbf{A})$
- Best $\kappa(\mathbf{M}^{-1}\mathbf{A})$ for $\mathbf{M} = \mathbf{A}$
- $\mathbf{M}\vec{x} = \vec{b}$ not useful
- There exist numerous preconditioners
- CG should be used with preconditioning

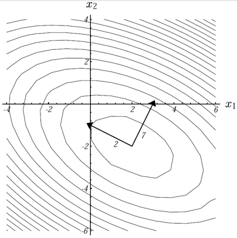
Preconditioning

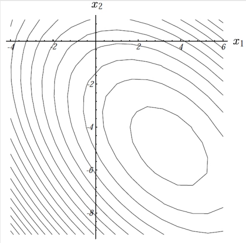
Diagonal Preconditioning

- Choose for $\mathbf{M}$ only the diagonal part of $\mathbf{A}$
- Effect: Scaling along the coord axes
- Perfect conditioner scales along the eigenvectors of $\mathbf{A}$

Example

$$\mathbf{A} = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} 3 & 0 \\ 0 & 6 \end{bmatrix}, \quad \kappa(\mathbf{A}) = 3.5, \quad \kappa(\mathbf{M}^{-1}\mathbf{A}) = 2.8$$





Starting and Stopping

Starting

- Use a priori knowledge of the solution, e.g. solution of the last time step
- Otherwise use zero vector as starting position $\vec{x}_{(0)}$

Stopping

- Exact solution after n-steps (infinite machine precision)
- Iterative method: Stop when $\|\vec{r}_{(i)}\| < \epsilon \|\vec{r}_{(0)}\|$
- $\vec{r}_{(i)} = \vec{r}_{(i-1)} - \alpha_{(i-1)} \mathbf{A} \vec{d}_{(i-1)}$
- Compute new $\vec{r}_{(i)}$ after some iterations, due to roundoff errors
- $\vec{r}_{(i)} = \vec{b} - \mathbf{A} \vec{x}_{(i)}$

Conclusion

Overview

- Algorithms solve minimizing problem instead of a linear system
- CG used as an iterative solver for sparse large systems

Comparison of the three Algorithms

Attribute/Algorithm	SD	CD	CG
Search direction	residual	A-orthogonal directions	A-orthogonal directions
Construction $d_{(i)}$	-	from coord-axes	from residuals
Convergence	finite number of steps	at most n steps	at most n steps
Complexity	O(κ n)		O($\sqrt{\kappa}n$)
Else		$\vec{d}_{(i)}$ have to be stored	
		compute intensive	
	bad convergence		

References

- Jonathan Richard Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, August 1994
- Richard L. Burden, J.Douglas Faires, Numerical Analysis, 2001