Conjugate Gradient Method

A gradient-based method

Outline

- The conjugate-relation between vectors
- Basic ideas of conjugate gradient method
- The intuitive algorithm
- Improvements
- The final algorithm
- Discussion
  - Time complexity
  - Effects of round-off errors

Mutual Conjugate Vectors

- ***A*-conjugate**: matrix ***A*** is SPD, *u* and *v* are vectors in the R^n space and mutually **conjugate** (with respect to ***A***)

  if $u^TAv = v^TAu = 0.$
  - Denoted as $< u, v >_A = 0$.
    - ***A*** is SPD, $\vec{u}^T A\vec{u} > 0$
- Mutual-conjugate is a symmetric relation.
- Mutual-conjugate ≈ orthogonality.
  - 類似相互垂直，但其中任意一個向量經過***A***矩陣轉換後，會與另一個向量垂直。
- Questions:
  - Can two parallel vectors be conjugate (with respect to ***A***)?
    - No, because ***A*** is SPD. $\vec{u}^T A(k\vec{u}) = k\vec{u}^T A\vec{u} > 0$, if k>0.

Extra Materials

- $\vec{u}^T A\vec{v} = \vec{v}^T A\vec{u}$ if and only if ***A*** is symmetric.
  - If ***A*** is symmetric, this relation holds.
  - If ***A*** is non-symmetric, this relation doesn't hold.
    - Let ***A*** be a rotational matrix (by 30 degrees, for example). (rotational matrices are NOT symmetric.)
  - Thus, mutual-conjugate relation is possible, only if ***A*** is symmetric.
- Given matrix ***A***, the vector *v*, being conjugate with vector *u*, is not unique.
  - If $\langle \vec{u}, \vec{v} \rangle_A$ =0, $\langle \vec{u}, k\vec{v} \rangle_A$ =0 for k=1, 2, 3, …

Basic Properties (1/3)

- P1, if u and v are mutual conjugate, they are linearly independent.
  - Proof by contradiction:
  - Assume they are linear dependent, then $u = kv$, k is a scalar.
  - $0 = u^T Av = kv^T Av = k(v^T Av) \neq 0$, because A is SPD.
  - A contradiction.

- P2, Given matrix $\boldsymbol{A}$, we can create exactly n mutual conjugate vectors in the R^n space.
  - Proof: see the algorithm of conjugate gradient method.
    - There are at most n linear independent vectors in R^n space.

- P3, These n mutual conjugate vectors form a basis in R^n space. (Since they are linearly independent.)

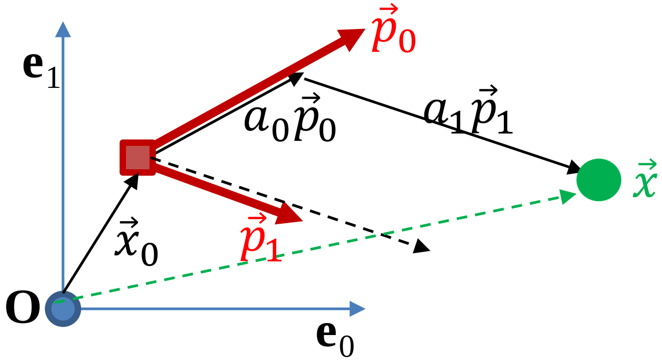
Basic Properties (2/3)

- P4, Vectors $\{p_1 \quad p_2 \cdots \quad p_n\}$ are mutual conjugate, then we can express any vector x by

$$\vec{x} = \vec{x}_0 + a_1\vec{p}_1 + \cdots + a_n\vec{p}_n.$$

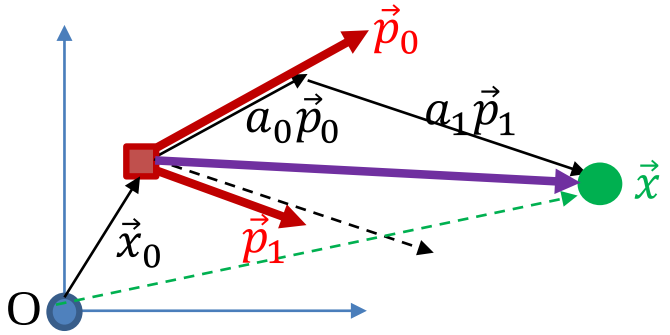
//$\vec{x}_0$is a vector.

$$\vec{x} = \vec{x}_0 + \sum_{i=1}^{n} a_i\vec{p}_i \, , a_i \in R.$$



Basic Properties (3/3)

$$\vec{x} = \vec{x}_0 + a_1\vec{p}_1 + \cdots + a_n\vec{p}_n$$
$$= \vec{x}_0 + \sum_{i=1}^{n} a_i\vec{p}_i\,, a_i \in R.$$



- P5, The coefficients a_i is computed by

$$a_i = \frac{(\vec{x}-\vec{x}_0)^T A\vec{p}_i}{\vec{p}_i{}^T A\vec{p}_i} \text{ or } a_i = \frac{<\vec{x}-\vec{x}_0,\vec{p}_i>_A}{<\vec{p}_i,\vec{p}_i>_A}.$$

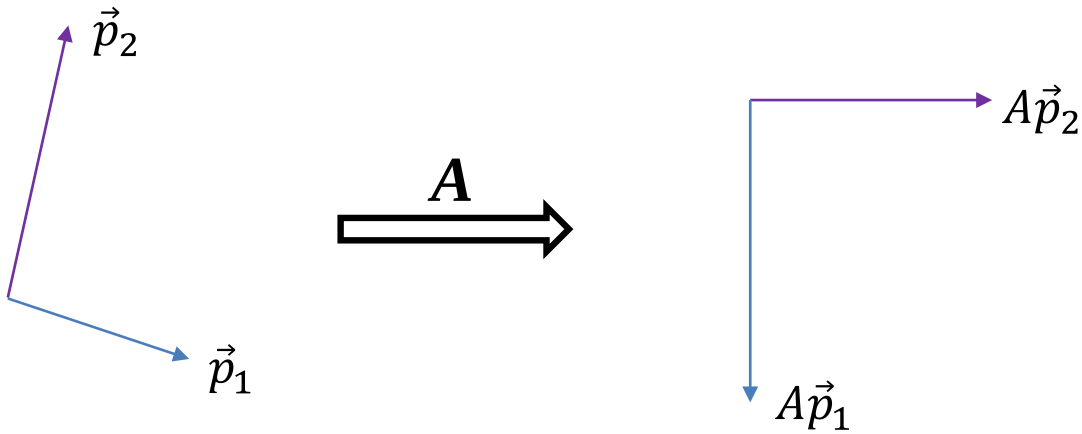
Operator: A-inner product:
$< \vec{x}, \vec{y} >_A = \vec{x}^T A\vec{y}$

  - Move $\vec{x}_0$ to the left side.
  - Take $\boldsymbol{A}$-inner product $<\text{-}, \vec{p}_i >_A$ on both sides.

$$< \vec{x} - \vec{x}_0, \vec{p}_i >_A = \sum_{j=1}^{n} a_i < \vec{p}_i, \vec{p}_j >_A,$$
$$< \vec{p}_i, \vec{p}_j >_A = 0, if\ \ i \neq j,$$
$$< \vec{x} - \vec{x}_0, \vec{p}_i >_A = a_i < \vec{p}_i, \vec{p}_i >_A,$$
$$a_i = \frac{<\vec{x}-\vec{x}_0,\vec{p}_i>_A}{<\vec{p}_i,\vec{p}_i>_A}.$$ //$\boldsymbol{A}$ is SPD, the denominator>0.

The Basis

- Vectors $\{\vec{p}_1, \vec{p}_2, \dots, \vec{p}_n\}$ are not mutually orthogonal, but $\{A\vec{p}_1, A\vec{p}_2, \dots, A\vec{p}_n\}$ are!



Ideas of Conjugate Gradient Method (1/2)

- Select the initial guess $\vec{x}_0$ for solving $\boldsymbol{A}x = b$;
  - Compute $\vec{d}_0 = \vec{b} - A\vec{x}_0$; //The 1st searching direction.
- Construct a mutual conjugate basis $\{\vec{d}_0 \quad \vec{d}_1 \quad \dots \quad \vec{d}_{n-1}\}$ iteration by iteration.
- The solution $\vec{x}^*$ can be obtained as follows

$\vec{x}^* = \vec{x}_0 + \sum_{i=0}^{n-1} a_i \vec{d}_i$, where $a_i = \frac{<\vec{x}^* - \vec{x}_0, \vec{d}_i>_A}{<\vec{d}_i, \vec{d}_i>_A}$.

//Based on P4 and P5

Ideas of Conjugate Gradient Method (2/2)

In the previous slide, we have

$\vec{x}^* = \vec{x}_0 + \sum_{i=0}^{n-1} a_i \vec{d}_i$, where $a_i = \frac{<\vec{x}^* - \vec{x}_0, \vec{d}_i>_A}{<\vec{d}_i, \vec{d}_i>_A}$.

- But $\vec{x}^*$ is the unknown, we can't use it to compute a_i. Instead, a_i is computed by

$a_i = -\frac{<\vec{g}_0, \vec{d}_i>}{<\vec{d}_i, \vec{d}_i>_A}$, where $\vec{g}_0 = \boldsymbol{A}\vec{x}_0 - \vec{b}$, the gradient of the **quadratic form,** $f(x) = \frac{1}{2}x^T A x - x^T b$.

(Call this property P6).

Proof:

$\vec{g}_0 = \boldsymbol{A}\vec{x}_0 - \vec{b} = \boldsymbol{A}\vec{x}_0 - A\vec{x}^* = A(\vec{x}_0 - \vec{x}^*)$, thus

$-< \vec{g}_0, \vec{d}_i > = -(\vec{x}_0 - \vec{x}^*)^T A\vec{d}_i =< \vec{x}_0 - \vec{x}^*, \vec{d}_i >_A$. //$\boldsymbol{A}$ is symmetric

Basic Algorithm

Select x_0;

Compute $d_0 = -g_0 = b - Ax_0$; //Negation of the initial gradient.

for $k = 0,1,2,\ldots,n\text{-}1$ *do*{

Compute $a_k = -\frac{<g_0,d_k>}{<d_k,d_k>_A}$; //Using P6

Compute $x_{k+1} = x_0 + \sum_{i=0}^{k} a_k d_k = x_k + a_k d_k$; //Improving the solution

Compute d_{k+1} *such that* $< d_{k+1}, d_j >_A = 0, \forall j = 0, \ldots, k$;

}

- Key problems:
  - How to compute d_{k+1} ?
  - Time complexity =?

Searching Directions

- We regard d_k as searching directions for improving x_k.

  $\vec{x}_{k+1} = \vec{x}_k + a_k \vec{d}_k$, where

  $a_k = -\frac{<\vec{g}_0, \vec{d}_k>}{<\vec{d}_k, \vec{d}_k>_A}$. //The searching distance, based on P6

- Given matrix $\boldsymbol{A}$, the method for creating n mutual conjugate vectors are not unique.

- Popular methodology
  - Select $\vec{d}_0 = -\vec{g}_0$, where $\vec{g}_0 = \boldsymbol{A}\vec{x}_0 - \vec{b}$ is the *gradient* of $f(\vec{x}) = -\frac{1}{2}\vec{x}^T A \vec{x} b$.
  - Then use $\vec{d}_k$ and gradient $\vec{g}_{k+1}$ to compute $\vec{d}_{k+1}$. Where

    $\vec{g}_{k+1} = \boldsymbol{A}\vec{x}_{k+1} - \vec{b}$. //The new gradient

New Searching Distance

- Lemma: $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{d}_k$, $\alpha_k = -\frac{<\vec{g}_0, \vec{d}_k>}{<\vec{d}_k, \vec{d}_k>_A}$. Where α_k can be replaced by

  $\alpha_k = -\frac{<\vec{g}_k, \vec{d}_k>}{<\vec{d}_k, \vec{d}_k>_A}$, //call it P7

  $\vec{g}_k = \boldsymbol{A}\vec{x}_k - \vec{b}$.

  $\boxed{\text{//P6, } \alpha_k = -\frac{<\vec{g}_0, \vec{d}_k>}{<\vec{d}_k, \vec{d}_k>_A}}$

Proof:

$\vec{g}_k = \boldsymbol{A}\vec{x}_k - \vec{b} = A(\vec{x}_0 + \sum_{i=0}^{k-1} \alpha_i \vec{d}_i) - A(\vec{x}_0 + \sum_{i=0}^{n-1} \alpha_i \vec{d}_i)$

$= -A(\sum_{i=k}^{n-1} \alpha_i \vec{d}_i)$,

- Taking inner-product with $\vec{d}_k$ on both sides and $\vec{d}_k$ is conjugate to $\vec{d}_i$, $k+1 \leq i \leq n-1$, we have

$\langle \vec{g}_k, \vec{d}_k \rangle = -\alpha_k \langle \vec{d}_k, \vec{d}_k \rangle_A$, //be aware of the rhs $<-,->_A$.

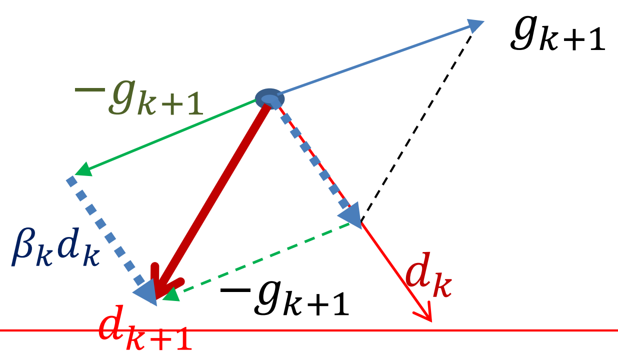
Thus, $\alpha_k = -\frac{<\vec{g}_k, \vec{d}_k>}{<\vec{d}_k, \vec{d}_k>_A}$.

New Searching Direction

- **Theorem**: If the previous searching directions $\{\vec{d}_0 \quad \vec{d}_1 \quad \dots \quad \vec{d}_k\}$ have been computed, then the new searching direction can be calculated by

$$\vec{d}_{k+1} = -\vec{g}_{k+1} + \beta_k \vec{d}_k,$$

where $\vec{g}_{k+1} = A\vec{x}_{k+1} - \vec{b}$ and $\beta_k = \frac{<\vec{g}_{k+1}, \vec{d}_k>_A}{<\vec{d}_k, \vec{d}_k>_A}$.



$$\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{d}_k,$$

$$\alpha_k = -\frac{<\vec{g}_k, \vec{d}_k>}{<\vec{d}_k, \vec{d}_k>_A}. \text{ //P7}$$

The New Direction

- Theorem: $\vec{d}_{k+1} = -\vec{g}_{k+1} + \beta_k \vec{d}_k$ is conjugate to $\vec{d}_i$, $0 \leq i \leq k$, where

  $\beta_k = \frac{<\vec{g}_{k+1}, \vec{d}_k>_A}{<\vec{d}_k, \vec{d}_k>_A}$. // P8

  - It is easy to prove that $\vec{d}_{k+1}$ and $\vec{d}_k$ are mutual conjugate.
    - Compute and verify $\langle \vec{d}_{k+1}, \vec{d}_k \rangle_A = 0$
  - But to prove that $\vec{d}_{k+1}$ is conjugate to other $\vec{d}_i$ is much difficult. It can be proved by induction. Omit the proof here.

The Naïve Algorithm

Select x_0;

$d_0 = b - Ax_0$; //Initial searching direction, negative gradient direction

$g_0 = -d_0$; //the initial gradient

For k=0,1,2,3,… do{

$\alpha_k = -\frac{<g_k, d_k>}{<d_k, d_k>_A} = -\frac{{g_k}^T d_k}{{d_k}^T A d_k}$; //Based on P7

$x_{k+1} = x_k + \alpha_k * d_k$; //Update the solution

$g_{k+1} = Ax_{k+1} - b$; //The new gradient

$\beta_k = \frac{<g_{k+1}, d_k>_A}{<d_k, d_k>_A} = \frac{{g_{k+1}}^T A d_k}{{d_k}^T A d_k}$; //P8

$d_{k+1} = -g_{k+1} + \beta_k d_k$; //New searching direction

}

Improvements

- The operation $<-,->_A$ is expensive. It takes O(n^2) steps.
  - For computing $\alpha_k = -\frac{<\vec{g}_k,\vec{d}_k>}{<\vec{d}_k,\vec{d}_k>_A}$ and $\beta_k = \frac{<\vec{g}_{k+1},\vec{d}_k>_A}{<\vec{d}_k,\vec{d}_k>_A}$.
- The gradient $\vec{g}_{k+1}$ needs O(n^2) steps too.
  - Because $\vec{g}_{k+1} = A\vec{x}_{k+1} - \vec{b}$.
- We need O($3n^2$) steps in each iteration.
- Can we simplify the algorithm?
  - For computing $\alpha_k = -\frac{g_k{}^T\vec{d}_k}{\vec{d}_k{}^T A\vec{d}_k}$ and $\beta_k = \frac{\vec{g}_{k+1}{}^T A\vec{d}_k}{\vec{d}_k{}^T A\vec{d}_k}$;

Improvements 1 & 2

1. Define $\vec{h}_k = A\vec{d}_k$ then $\vec{g}_{k+1} = \vec{g}_k + \alpha_k \vec{h}_k$. //O($n^2$) steps

Proof:

$$\vec{g}_{k+1} = A\vec{x}_{k+1} - \vec{b} = A(\vec{x}_k + \alpha_k \vec{d}_k) - \vec{b}.$$

$$\vec{g}_{k+1} = A\vec{x}_k - \vec{b} + \alpha_k A\vec{d}_k = \vec{g}_k + \alpha_k \vec{h}_k.$$

2. $\alpha_k = \frac{\|\vec{g}_k\|^2}{\vec{d}_k{}^T \vec{h}_k}$. //call it P9, O(n) steps.

Proof :

$$\alpha_k = -\frac{<\vec{g}_k, \vec{d}_k>}{<\vec{d}_k, \vec{d}_k>_A} = -\frac{g_k{}^T \vec{d}_k}{\vec{d}_k{}^T A\vec{d}_k}.$$ //based on P7

$$< \vec{g}_k, \vec{d}_k > = < \vec{g}_k, -\vec{g}_k + \beta_{k-1}\vec{d}_{k-1} >.$$

//$\vec{g}_k \perp \vec{d}_{k-1}$, current gradient is perpendicular to previous searching direction

$$< \vec{g}_k, \vec{d}_{k-1} > = 0,$$

Thus $< \vec{g}_k, \vec{d}_k > = -< \vec{g}_k, -\vec{g}_k > = \|\vec{g}_k\|^2$.

– The denominator part is easy to prove.

Improvements

3. $\beta_k = \frac{\|\vec{g}_{k+1}\|^2}{\|\vec{g}_k\|^2}$. // P10, O($n^2$) steps

Proof:

$\beta_k = \frac{<\vec{g}_{k+1}, \vec{d}_k>_A}{<\vec{d}_k, \vec{d}_k>_A}$ // P8

$= \frac{<\vec{g}_{k+1}, \vec{d}_k>_A}{\vec{d}_k^{\,T} A \vec{d}_k} = \frac{<\vec{g}_{k+1}, \vec{d}_k>_A}{\vec{d}_k^{\,T} \vec{h}_k}$.

$\vec{g}_{k+1} = \vec{g}_k + \alpha_k A \vec{d}_k$, //by P9

$A\vec{d}_k = \frac{\vec{g}_{k+1} - \vec{g}_k}{\alpha_k}$,

$< \vec{g}_{k+1}, \vec{d}_k >_A = \vec{g}_{k+1}^{\,T} A \vec{d}_k$

$= \frac{\vec{g}_{k+1}^{\,T} \vec{g}_{k+1} - \vec{g}_{k+1}^{\,T} \vec{g}_k}{\alpha_k}$,

Since $\vec{g}_{k+1}^{\,T} \vec{g}_k = 0$, we have

$< \vec{g}_{k+1}, \vec{d}_k >_A = \frac{\|\vec{g}_{k+1}\|^2}{\alpha_k}$.

$\alpha_k = \frac{\|\vec{g}_k\|^2}{\vec{d}_k^{\,T} \vec{h}_k} = \frac{\|\vec{g}_k\|^2}{<\vec{d}_k, \vec{d}_k>_A}$, //by P9

Thus, $< \vec{d}_k, \vec{d}_k >_A = \frac{\|\vec{g}_k\|^2}{\alpha_k}$ and

$$\beta_k = \frac{\frac{\|\vec{g}_{k+1}\|^2}{\alpha_k}}{\frac{\|\vec{g}_k\|^2}{\alpha_k}} = \frac{\|\vec{g}_{k+1}\|^2}{\|\vec{g}_k\|^2}.$$

The Revised Algorithm

$\vec{x}^* = \vec{x}_0 + \sum_{k=0}^{n-1} \alpha_k \vec{d}_k.$

- Searching direction

  $\vec{d}_{k+1} = -\vec{g}_{k+1} + \beta_k \vec{d}_k.$
- Revising solution

  $\vec{x}_{k+1} = \vec{x}_k + \alpha_k \vec{d}_k.$
- Improvement 1:

  $\vec{h}_k = A\vec{d}_k,$

  $\vec{g}_{k+1} = \vec{g}_k + \alpha_k \vec{h}_k.$
- Improvement 2:

  $\alpha_k = \frac{\|\vec{g}_k\|^2}{\vec{d}_k^{\,T} \vec{h}_k}.$
- Improvement 3:

  $\beta_k = \frac{\|\vec{g}_{k+1}\|^2}{\|\vec{g}_k\|^2}.$

//Revised method

Select x_0; //initial solution

$d = b - Ax_0$; //initial searching direction

$g = -d$; //initial gradient

For k=0,1,2,3,… do{

$h = A * d;$

$oldG2 = < g, g >;$ // $\|\vec{g}_k\|^2$

$\alpha = \frac{oldG2}{<d,h>};$ // $\alpha_k = \frac{\|\vec{g}_k\|^2}{\vec{d}_k^{\,T} \vec{h}_k}$

$x = x + \alpha * d;$ //new x

$g = g + \alpha * h;$ //new gradient

$newG2 = < g, g >;$ // $\|\vec{g}_{k+1}\|^2$

$\beta = \frac{newG2}{oldG2};$ // $\beta_k = \frac{\|\vec{g}_{k+1}\|^2}{\|\vec{g}_k\|^2}.$

$d = -g + \beta * d;$ //new searching direction

}

Comparison

//Revised method

Select x_0;

$d = b - Ax_0$;

$g = -d$;

For k=0,1,2,3,… do{

$h = A * d;$

$oldG2 =< g, g >;$

$\alpha = \frac{oldG2}{<d,h>};$

$x = x + \alpha * d;$

$g = g + \alpha * h;$

$newG2 =< g, g >;$

$\beta = \frac{newG2}{oldG2};$

$d = -g + \beta * d;$

}

//O(n^2) steps in each iteration.

//Naïve method

Select x_0;

$d_0 = b - Ax_0$;

$g_0 = -d_0$;

For k=0,1,2,3,… do{

$\alpha_k = -\frac{<g_k,d_k>}{<d_k,d_k>_A} = -\frac{{g_k}^T d_k}{{d_k}^T A d_k};$

$x_{k+1} = x_k + \alpha_k * d_k;$

$g_{k+1} = Ax_{k+1} - b;$

$\beta_k = \frac{<g_{k+1},d_k>_A}{<d_k,d_k>_A} = \frac{{g_{k+1}}^T A d_k}{{d_k}^T A d_k};$

$d_{k+1} = -g_{k+1} + \beta_k d_k;$

}

//O($3n^2$) steps in each iteration.

Time Complexity

- Theoretically, Conjugate Gradient Method will stop within n iterations.
  - There are exactly n mutual conjugate d_i in R^n space.

- Each iteration needs $O(n^2)$ steps.
- Thus the total time complexity is $O(n^3)$.

- However, conjugate gradient method is numerical unstable.
  - The searching directions d_i are not mutual conjugate.

Round-Off Errors

- Consider $h = Ax_k$. It may produce an $O(n^2\varepsilon)$ relative error.
  - Thus $\|g_{k+1}\|^2$ may produce an $O(n^2\varepsilon)$ relative error.
- The relative error of $\alpha_k = \frac{\|g_k\|^2}{{d_k}^T h_k}$ is at least $O(n^2\varepsilon)$.
  - However, if $|\alpha_k|>1$ then the absolute error grows exponentially with the iterations.
- $\beta_k = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}$ also encounters similar problems.

Conclusion

- Conjugate Gradient (CG) method can be used to solve SPD systems.
- CG method modifies the solution in each iteration by creating a new A-conjugate direction d.

$$d_0 = -g_0 = b - Ax_0;$$
$$x_{k+1} = x_k + \alpha_k d_k,\ \alpha_k = -\frac{<g_0, d_k>}{<d_k, d_k>_A};$$
$$g_{k+1} = Ax_{k+1} - b;$$
$$\beta_k = \frac{<g_{k+1}, d_k>_A}{<d_k, d_k>_A} = \frac{{g_{k+1}}^T A d_k}{{d_k}^T A d_k};$$
$$d_{k+1} = -g_{k+1} + \beta_k d_k;$$

  - New search direction = -*gradient* + β**previous search direction*
- Theoretically, CG method converges at *n* iterations.
- But, numerical errors hinder the converge rate.

Interesting Topics

- CG methods is good for SPD, does it works for
  - Symmetric systems
  - Diagonal dominant systems
  - Positive definite systems
- Conjugate relation between vectors is similar to orthogonality relation, consider special matrices $\boldsymbol{A}$:
  - Diagonal dominant matrices
  - Ill-conditioned matrices
- Study the acute angles between vectors for these matrices.
  - Given a SPD matrix $\boldsymbol{A}$, compute $\{\vec{d}_i\}$, i=0,1,…,n-1.
  - Normalize $\{\vec{d}_i\}$;
  - Compute $\left|\langle \vec{d}_i, \vec{d}_j \rangle\right|$ to form a 2D table;
  - Visualize the table;
  - Try another matrix;
- Since $\boldsymbol{A}$ is symmetric, all its eigenvectors are mutually orthogonal::
  - Let $\{\vec{d}_i\}$ be its eigenvectors;
  - Given the 1^{st} eigenvector, can we deduce an algorithm to compute other eigenvectors?