Carnegie Mellon

18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

Overview

- **Conjugate Gradient Method (Part 1)**
  - Quadratic programming
  - Gradient method
  - Orthogonal search direction

Linear Equation

- **Linear equation**

$$AX = B$$

  - A is symmetric and positive definite
  - Can be solved by Cholesky decomposition

- **However, Cholesky decomposition (or Gaussian elimination in general) is not efficient if A is large and sparse**

Linear Equation

- **A matrix is sparse if it contains a large number of zero elements**

$$\begin{bmatrix} \times & & & \times & & & \\ & \times & & & & & \\ & & \times & & \times & & \\ \times & & & \times & & & \times \\ & & \times & & \times & & \\ & & & & & \times & \\ & & & \times & & & \times \end{bmatrix}$$

- **Sparse matrix can be saved with small memory requirements**
  - We do not explicitly save zero elements
  - We only save non-zero values and their locations

Linear Equation

- Cholesky decomposition or Gaussian elimination can generate a large number of **fill-ins** (i.e., non-zeros)
  - Matrix becomes much less sparse and consumes much memory

$Ax = B$

$$A = \begin{bmatrix} \times & & \times & & \times & \\ \times & \times & \times & \times & \times & \times \\ & & \times & & \times & \\ & \times & & \times & & \\ & & \times & & & \times \end{bmatrix}$$

Fill-ins

- Iterative methods (e.g., conjugate gradient) are much more efficient in these cases

Quadratic Programming

- Reformulate linear equation as a quadratic programming problem

$$AX = B$$

$$\min_x f(x) = \frac{1}{2} x^T A x - B^T x + C \rightarrow \text{Convex}$$

$\nabla f(x) = 0$

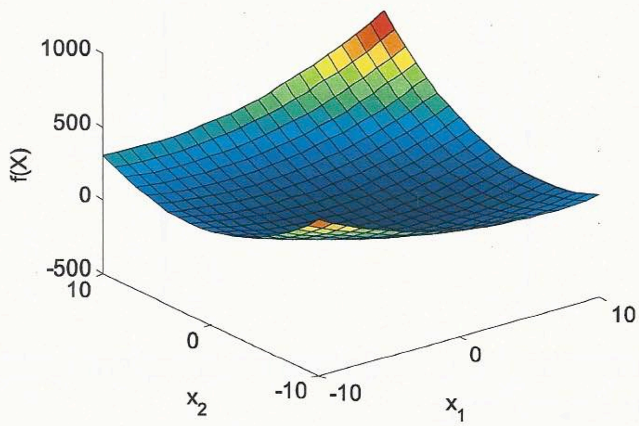
$Ax - B = 0$

$x = A^{-1} B$

A: pos. definite

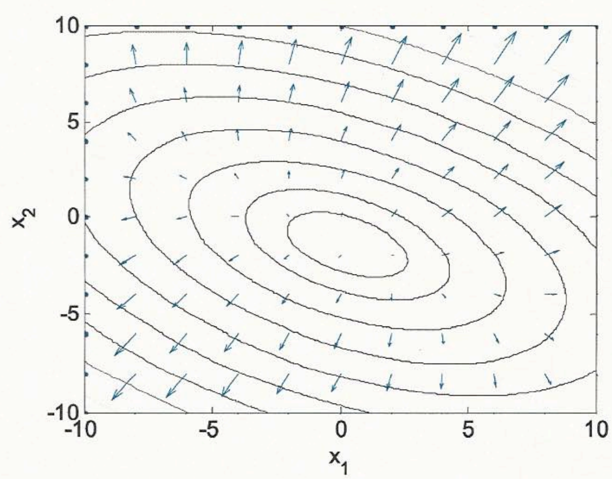
Quadratic Programming Example

$$f(X)=\frac{1}{2}X^T AX - B^T X + C \qquad A=\begin{bmatrix}3 & 2\\ 2 & 6\end{bmatrix} \quad B=\begin{bmatrix}-2\\ -8\end{bmatrix} \quad C=0$$



Quadratic Programming Example

- Contour and gradient



Gradient Method

- **Iteration scheme**

$$\min_{X} \quad f(X) = \frac{1}{2}X^T AX - B^T X + C$$

$\nabla f(x) = Ax - B$

$x^{(k+1)} = x^{(k)} - \mu^{(k)} \nabla f[x^{(k)}] = x^{(k)} - \mu^{(k)}[Ax^{(k)} - B]$

$= x^{(k)} + \mu^{(k)} \cdot \underbrace{[B - Ax^{(k)}]}_{R^{(k)}\text{: residual}}$

$x^{(k+1)} = x^{(k)} + \mu^{(k)} R^{(k)}$

Gradient Method

- **Optimal step size**

$$f(X) = \frac{1}{2}X^T AX - B^T X + C \qquad R^{(k)} = B - AX^{(k)} \qquad X^{(k+1)} = X^{(k)} + \mu^{(k)}R^{(k)}$$

$\min_{\mu^{(k)}} f[x^{(k+1)}] = \frac{1}{2} x^{(k+1)^T} A x^{(k+1)} - B^T x^{(k+1)} + C$

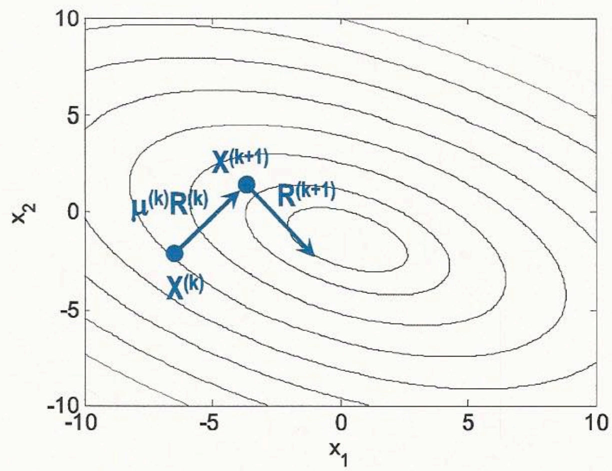
$\frac{df[x^{(k+1)}]}{d\mu^{(k)}} = \left[\frac{df}{dx^{(k+1)}}\right]^T \cdot \frac{dx^{(k+1)}}{d\mu^{(k)}} = [Ax^{(k+1)} - B]^T \cdot R^{(k)}$

$= -R^{(k+1)^T} R^{(k)} = 0$

$R^{(k+1)} \perp R^{(k)}$

Quadratic Programming Example

$$f(X)=\frac{1}{2}X^T AX - B^T X + C \qquad A=\begin{bmatrix}3 & 2\\ 2 & 6\end{bmatrix} \quad B=\begin{bmatrix}-2\\ -8\end{bmatrix} \quad C=0$$



Gradient Method

- **Optimal step size**

$$R^{(k)} = B - AX^{(k)} \qquad X^{(k+1)} = X^{(k)} + \mu^{(k)} R^{(k)} \qquad R^{(k+1)T} R^{(k)} = 0$$

$$[B - Ax^{(k+1)}]^T R^{(k)} = 0$$

$$\{B - A[x^{(k)} + \mu^{(k)} AR^{(k)}]\}^T R^{(k)} = 0$$

$$[R^{(k)} - \mu^{(k)} AR^{(k)}]^T R^{(k)} = 0$$

$$R^{(k)T} R^{(k)} - \mu^{(k)} R^{(k)T} AR^{(k)} = 0$$

$$\mu^{(k)} = \frac{R^{(k)T} R^{(k)}}{R^{(k)T} AR^{(k)}}$$

Gradient Method

- **Iteration scheme**

$$\min_{X} \quad f(X) = \frac{1}{2} X^T A X - B^T X + C$$

$$\Downarrow$$

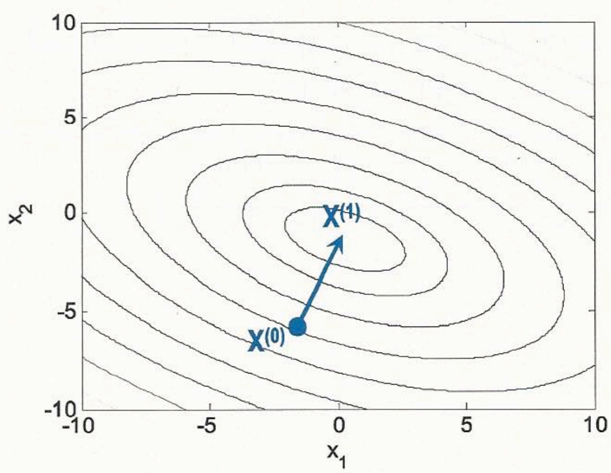
$$R^{(k)} = B - AX^{(k)}$$

$$\mu^{(k)} = \frac{R^{(k)T} R^{(k)}}{R^{(k)T} A R^{(k)}}$$

$$X^{(k+1)} = X^{(k)} + \mu^{(k)} R^{(k)}$$

Quadratic Programming Example

$$f(X) = \frac{1}{2} X^T A X - B^T X + C \qquad A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \quad B = \begin{bmatrix} -2 \\ -8 \end{bmatrix} \quad C = 0$$



Gradient method may converge by one iteration if a "good" initial guess is selected

Initial Guess

- **Gradient method converges by one iteration, if $R^{(0)}$ is an eigenvector of A**

$$R^{(k)} = B - AX^{(k)} \qquad \mu^{(k)} = \frac{R^{(k)T}R^{(k)}}{R^{(k)T}AR^{(k)}} \qquad X^{(k+1)} = X^{(k)} + \mu^{(k)}R^{(k)}$$

$$AR^{(0)} = \lambda R^{(0)}$$

$$\mu^{(0)} = \frac{R^{(0)T}R^{(0)}}{R^{(0)T}AR^{(0)}} = \frac{R^{(0)T}R^{(0)}}{R^{(0)T}\lambda R^{(0)}} = \frac{1}{\lambda} \qquad \frac{R^{(0)T}R^{(0)}}{R^{(0)T}\cancel{A}R^{(0)}} = \frac{1}{\lambda}$$

$$R^{(1)} = B - Ax^{(1)} = B - A[x^{(0)} + \tfrac{1}{\lambda}R^{(0)}]$$

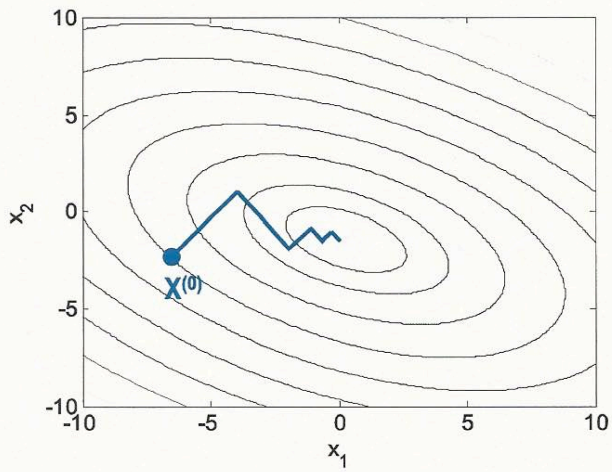
$$= B - Ax^{(0)} - \frac{1}{\lambda}A \cdot R^{(0)} = R^{(0)} - \frac{1}{\lambda}\lambda R^{(0)} = 0$$

Initial Guess

- **In practice, it is not possible to achieve such an ideal case**
  - We do not know the exact eigenvectors of A
- **Starting from a random initial guess, gradient method may take many iterations to converge**
  - Gradient method has slow convergence, even though optimal step size μ is used for each iteration
  - This is a big problem of gradient method

Quadratic Programming Example

$$f(X)=\frac{1}{2}X^T AX - B^T X + C \qquad A=\begin{bmatrix}3 & 2\\ 2 & 6\end{bmatrix} \quad B=\begin{bmatrix}-2\\ -8\end{bmatrix} \quad C=0$$



Newton Method

- **Newton method can converge by one iteration**
- **However, we have to solve the linear equation X = A⁻¹B**
  - It is exactly the problem that we try to solve at the beginning
  - Newton method does not tell us how to solve the large, sparse linear equation efficiently

$$\min_X \quad f(X)=\frac{1}{2}X^T AX - B^T X + C$$

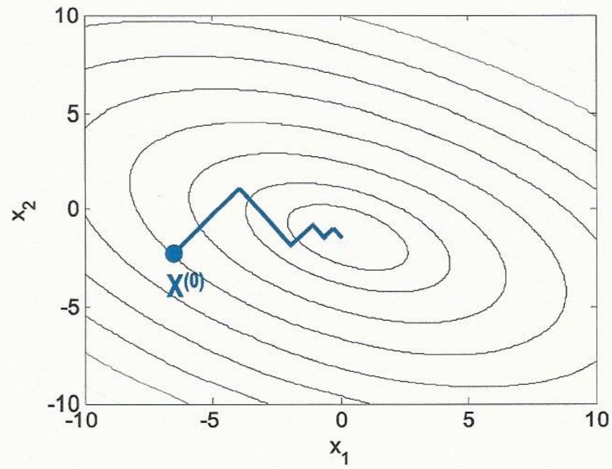
$$\Downarrow$$

$$\nabla f(X) = AX - B = 0$$

$$X = A^{-1}B$$

Orthogonal Search Direction

- **Gradient method often moves towards the same direction as earlier iteration steps – BAD idea**

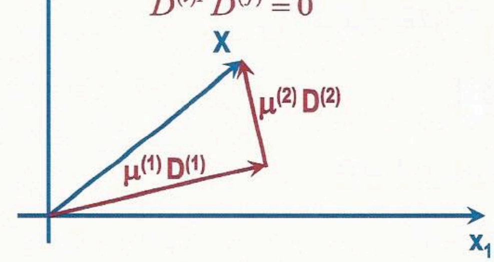


Orthogonal Search Direction

- **Ideally, we want to**
  - Select a set of orthogonal search directions $D^{(k)}$
  - Take exactly one iteration step for each direction
  - After at most N steps, we get the solution X
  - (N is the problem size, i.e., $A \in R^{N \times N}$)

$$X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)}$$

$$D^{(i)T} D^{(j)} = 0$$



How do we decide $\mu^{(k)}$ and $D^{(k)}$?

Orthogonal Search Direction

- **Determine step size $\mu^{(k)}$**

$D^{(i)T}D^{(j)}=0 \qquad X^{(k+1)}=X^{(k)}+\mu^{(k)}D^{(k)} \qquad X=X^{(0)}+\mu^{(0)}D^{(0)}+\cdots+\mu^{(N-1)}D^{(N-1)}$

$$x^{(k+1)}=x^{(0)}+\mu^{(0)}D^{(0)}+\ldots+\mu^{(k)}D^{(k)}$$

$$\Delta^{(k+1)}=x^{(k+1)}-x=-\mu^{(k+1)}D^{(k+1)}\ldots\mu^{(N-1)}D^{(N-1)}$$

$$D^{(k)T}\cdot\Delta^{(k+1)}=D^{(k)T}[-\mu^{(k+1)}D^{(k+1)}\ldots-\mu^{(N-1)}D^{(N-1)}]$$

$$=0$$

$$\Delta^{(k+1)}\perp D^{(k)}$$

Orthogonal Search Direction

- **Determine step size $\mu^{(k)}$**

$X^{(k+1)}=X^{(k)}+\mu^{(k)}D^{(k)} \qquad \Delta^{(k)}=X^{(k)}-X \qquad D^{(k)T}\Delta^{(k+1)}=0$

$$\Delta^{(k+1)}=x^{(k+1)}-x=x^{(k)}+\mu^{(k)}D^{(k)}-x$$

$$=\Delta^{(k)}+\mu^{(k)}\cdot D^{(k)}$$

$$D^{(k)T}\cdot[\Delta^{(k)}+\mu^{(k)}D^{(k)}]=0$$

$$D^{(k)T}\Delta^{(k)}+\mu^{(k)}D^{(k)T}D^{(k)}=0$$

$$\mu^{(k)}=\frac{-D^{(k)T}\Delta^{(k)}}{D^{(k)T}D^{(k)}}$$

must know $\Delta^{(k)}$!

$x=x^{(k)}-\Delta^{(k)}$

Orthogonal Search Direction

- Orthogonal search direction is difficult to apply to many practical optimization problems
- Instead of using orthogonal directions, we can make search directions **conjugate** (or equivalently **A-orthogonal**)
- More details in next lecture

Summary

- Conjugate gradient method (Part 1)
  - Quadratic programming
  - Gradient method
  - Orthogonal search direction

18-660: Numerical Methods for Engineering Design and Optimization

Xin Li
Department of ECE
Carnegie Mellon University
Pittsburgh, PA 15213

Electrical & Computer ENGINEERING

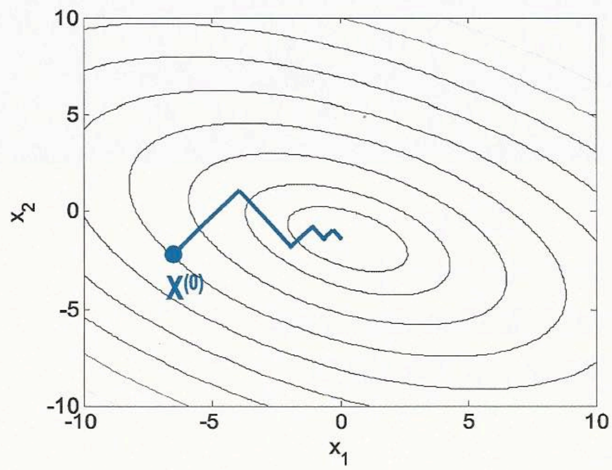
Overview

- **Conjugate Gradient Method (Part 2)**
  - Conjugate search direction
  - Gram-Schmidt conjugation
  - Conjugate gradient method

Quadratic Programming

- **Solve linear equation by quadratic programming**

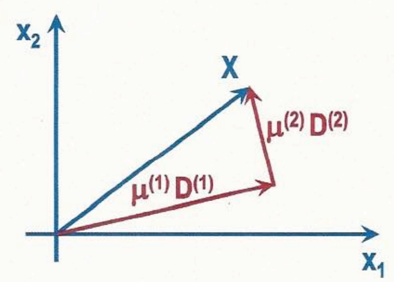
$$AX = B \quad \Rightarrow \quad \min_{X} \; f(X) = \frac{1}{2} X^T A X - B^T X + C$$



Gradient method has slow convergence

Orthogonal Search Direction

- **Ideally, we want to select a set of orthogonal search directions**



$$X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)}$$

$$D^{(i)T} D^{(j)} = 0$$

$$\Delta^{(k)} = X^{(k)} - X$$

$$\mu^{(k)} = -\frac{D^{(k)T} \Delta^{(k)}}{D^{(k)T} D^{(k)}}$$

However, we do not know $\Delta^{(k)}$ – otherwise, we know $X = X^{(k)} - \Delta^{(k)}$

Conjugate Search Direction

- We do not know $\Delta^{(k)}$, but we can easily calculate $A\Delta^{(k)}$

$$AX = B \qquad R^{(k)} = B - AX^{(k)} \qquad \Delta^{(k)} = X^{(k)} - X$$

$$A\Delta^{(k)} = A[x^{(k)} - x] = Ax^{(k)} - Ax = Ax^{(k)} - B$$
$$= -R^{(k)}$$

- Instead of using orthogonal directions $D^{(k)}$, we make search directions **conjugate** (or equivalently **A-orthogonal**)

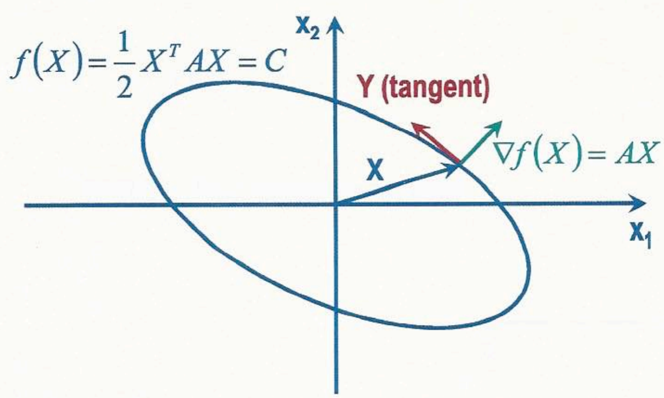
Conjugate Search Direction

- Two vectors $D^{(i)}$ and $D^{(j)}$ are conjugate (or **A-orthogonal**) if $D^{(i)T}AD^{(j)} = 0$ $(i \neq j)$

$D^{(i)T} D^{(j)} = 0$

$\Rightarrow$ orthogonal

- Geometrical interpretation



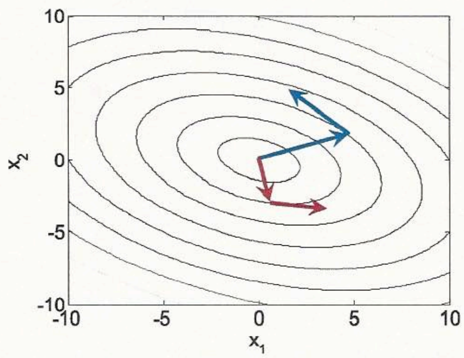
$y^T A x = 0$

$y^T \nabla f(x) = 0$

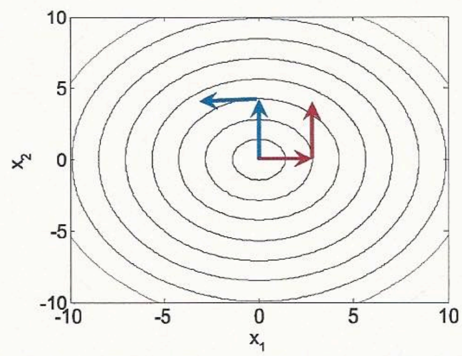
y and x are conjugate

Conjugate Example

- "Conjugate" = "orthogonal" if A = identity matrix



$$A = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 2 \end{bmatrix}$$



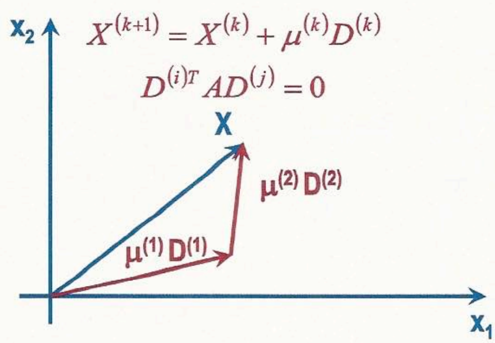
$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Conjugate Search Direction

- **Iteration scheme with conjugate search directions**
  - Select a set of conjugate search directions $D^{(k)}$
  - Take exactly one iteration step for each direction
  - After at most N steps, we get the solution X
  - (N is the problem size, i.e., $A \in R^{N \times N}$)

$$X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)}$$

$$D^{(i)T} A D^{(j)} = 0$$



How do we decide $\mu^{(k)}$ and $D^{(k)}$?

Conjugate Search Direction

- **Determine step size $\mu^{(k)}$**

$$D^{(i)T}AD^{(j)} = 0 \quad X^{(k+1)} = X^{(k)} + \mu^{(k)}D^{(k)} \quad X = X^{(0)} + \mu^{(0)}D^{(0)} + \cdots + \mu^{(N-1)}D^{(N-1)}$$

$x^{(k+1)} = x^{(0)} + \mu^{(0)} D^{(0)} + \ldots + \mu^{(k)} D^{(k)}$

$\Delta^{(k+1)} = x^{(k+1)} - x = -\mu^{(k+1)} D^{(k+1)} - \ldots - \mu^{(N-1)} D^{(N-1)}$

$D^{(k)T} A \Delta^{(k+1)} = D^{(k)T} A \cdot [-\mu^{(k+1)} D^{(k+1)} - \ldots - \mu^{(N-1)} D^{(N-1)}]$

$= 0$

$\Delta^{(k+1)}$ and $D^{(k)}$ are conjugate

Conjugate Search Direction

- **Determine step size $\mu^{(k)}$**

$$A\Delta^{(k+1)} = -R^{(k+1)} \qquad D^{(k)T}\underbrace{A\Delta^{(k+1)}}_{-R^{(k+1)}} = 0$$

$D^{(k)T} \cdot R^{(k+1)} = 0$

$R^{(k+1)} \perp D^{(k)}$

Conjugate Search Direction

- **Determine step size $\mu^{(k)}$**

$$X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)} \qquad R^{(k)} = B - AX^{(k)} \qquad D^{(k)T} R^{(k+1)} = 0$$

$$D^{(k)T}[B - AX^{(k+1)}] = D^{(k)T}\{B - A[X^{(k)} + \mu^{(k)} D^{(k)}]\}$$

$$= D^{(k)T} \cdot [B - AX^{(k)} - \mu^{(k)} A D^{(k)}]$$

$$= D^{(k)T} \cdot [R^{(k)} - \mu^{(k)} A D^{(k)}]$$

$$D^{(k)T} R^{(k)} - \mu^{(k)} D^{(k)T} A D^{(k)} = 0$$

$$\mu^{(k)} = \frac{D^{(k)T} R^{(k)}}{D^{(k)T} A D^{(k)}}$$

Conjugate Search Direction

- **$\mu^{(k)}$ minimizes $f[X^{(k+1)}]$ along the direction $D^{(k)}$**

$$f(X) = \frac{1}{2} X^T A X - B^T X + C \qquad R^{(k)} = B - AX^{(k)} \qquad X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)}$$

$$\min_{\mu^{(k)}} f[X^{(k+1)}] = \frac{1}{2} X^{(k+1)T} A X^{(k+1)} - B^T X^{(k+1)} + C$$

$$\frac{df[X^{(k+1)}]}{d\mu^{(k)}} = \left[\frac{df}{dX^{(k+1)}}\right]^T \left[\frac{dX^{(k+1)}}{d\mu^{(k)}}\right] = \underbrace{[AX^{(k+1)} - B]^T}_{-R^{(k+1)}} D^{(k)}$$

$$= -R^{(k+1)T} D^{(k)} = 0$$

$$R^{(k+1)} \perp D^{(k)}$$

Conjugate Search Direction

- **Important equations about conjugate search direction**

$AX = B$ → Linear equation

$\min_{X} \quad f(X) = \frac{1}{2} X^T AX - B^T X + C$ → Equivalent optimization

$R^{(k)} = B - AX^{(k)}$ → Residual definition

$\nabla f\left[X^{(k)}\right] = AX^{(k)} - B = -R^{(k)}$ → Residual vs. gradient

Conjugate Search Direction

- **Important equations about conjugate search direction**

$$\left.\begin{aligned} X^{(k+1)} &= X^{(k)} + \mu^{(k)} D^{(k)} \\ \mu^{(k)} &= \frac{D^{(k)T} R^{(k)}}{D^{(k)T} AD^{(k)}} \end{aligned}\right\} \longrightarrow \text{Iteration scheme}$$

$D^{(i)T} AD^{(j)} = 0$ → Conjugate search directions

$D^{(k)T} R^{(k+1)} = 0$ → Orthogonal residual

Gradient Method vs. Conjugate Search Direction

Gradient method

$$X^{(k+1)} = X^{(k)} + \mu^{(k)} R^{(k)}$$

$$\mu^{(k)} = \frac{R^{(k)T} R^{(k)}}{R^{(k)T} A R^{(k)}}$$

- **Use $R^{(k)}$ as search direction**
- **$\mu^{(k)}$ is optimized to minimize cost**

Conjugate gradient method

$$X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)}$$

$$\mu^{(k)} = \frac{D^{(k)T} R^{(k)}}{D^{(k)T} A D^{(k)}}$$

- **Use $D^{(k)}$ as search direction**
- **$\mu^{(k)}$ is optimized to minimize cost**

We need to further develop an algorithm to generate $D^{(k)}$'s that are conjugate

Conjugate Gradient Method

- **Construct search directions by conjugation of residuals**
  - Residual is directly related to gradient
  - Search directions are defined by conjugation of gradients

$$\min_{X} \quad f(X) = \frac{1}{2} X^T A X - B^T X + C$$

$$\Downarrow$$

$$\nabla f\left[X^{(k)}\right] = A X^{(k)} - B = -R^{(k)}$$

Conjugate Gradient Method

- **Define subspace**

$$S^{(k)} = span\{R^{(0)}, R^{(1)}, \cdots, R^{(K-1)}\}$$

K-dimensional space with K basis vectors

- **Gradient method directly uses $R^{(k)}$ as search direction**
- **Conjugate gradient method uses conjugation of $R^{(k)}$ so that each iteration step searches along a different direction**
  - Given a set of basis vectors, how do we calculate the conjugation of them?
  - Introduce the algorithm of Gram-Schmidt conjugation

Gram-Schmidt Conjugation

$$D^{(0)} = R^{(0)}$$

$$D^{(1)} = R^{(1)} + \beta_{10} D^{(0)} \qquad D^{(0)T} A D^{(1)} = 0$$

$$D^{(0)T} A D^{(1)} = D^{(0)T} A \cdot [R^{(1)} + B_{10} D^{(0)}] = 0$$

$$D^{(0)T} A R^{(1)} + B_{10} D^{(0)T} A D^{(0)} = 0$$

$$B_{10} = \frac{-D^{(0)T} A R^{(1)}}{D^{(0)T} A D^{(0)}}$$

Gram-Schmidt Conjugation

$$D^{(k)} = R^{(k)} + \sum_{i=0}^{k-1} \beta_{ki} D^{(i)} \qquad D^{(i)T} A D^{(j)} = 0$$

$$D^{(i)T} A D^{(k)} = D^{(i)T} A \cdot \left[R^{(k)} + \sum_{j=0}^{k-1} \beta_{kj} D^{(j)} \right] = 0$$

$$D^{(i)T} A R^{(k)} + \sum_{j=0}^{k-1} \beta_{kj} \cdot D^{(i)T} A D^{(j)} = 0$$

$$D^{(i)T} A R^{(k)} + \beta_{ki} D^{(i)T} A D^{(i)} = 0$$

$$\beta_{ki} = -\frac{D^{(i)T} A R^{(k)}}{D^{(i)T} A D^{(i)}}$$

Conjugate Gradient Method

- **Step 1: start from an initial guess $X^{(0)}$, and set k = 0**
- **Step 2: calculate**

$$D^{(0)} = R^{(0)} = B - AX^{(0)}$$

- **Step 3: update solution**

$$X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)} \quad \text{where} \quad \mu^{(k)} = \frac{D^{(k)T} R^{(k)}}{D^{(k)T} A D^{(k)}}$$

- **Step 4: calculate residual**

$$R^{(k+1)} = B - AX^{(k+1)}$$

- **Step 5: determine search direction**

$$D^{(k+1)} = R^{(k+1)} + \sum_{i=0}^{k} \beta_{k+1,i} D^{(i)} \quad \text{where} \quad \beta_{k+1,i} = -\frac{D^{(i)T} A R^{(k+1)}}{D^{(i)T} A D^{(i)}}$$

- **Step 6: set k = k + 1 and go to Step 3**

Conjugate Gradient Method

- **This simple implementation is not numerically efficient**
- **There are a number of numerical tricks that we can apply to reduce computational complexity**
- **Key idea**
  - $X^{(k)}$, $D^{(k)}$ and $R^{(k)}$ are strongly correlated
  - They can be computed in many different ways – we should use the most efficient algorithm in our implementation
  - More details in next lecture

Summary

- **Conjugate gradient method (Part 2)**
  - Conjugate search direction
  - Gram-Schmidt conjugation
  - Conjugate gradient method