18-660: Numerical Methods for Engineering Design and Optimization

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

- Conjugate Gradient Method (Part 4)
  - Pre-conditioning
  - Nonlinear conjugate gradient method
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
  \[
  R^{(k+1)} = R^{(k)} - \mu^{(k)} A D^{(k)}
  \]
- **Step 5:** determine search direction
  \[
  D^{(k+1)} = R^{(k+1)} + \beta_{k+1,k} D^{(k)} \quad \text{where} \quad \beta_{k+1,k} = \frac{R^{(k+1)T} R^{(k+1)}}{D^{(k)T} R^{(k)}}
  \]
- **Step 6:** set \( k = k + 1 \) and go to Step 3
Convergence Rate

\[ \| X^{(k+1)} - X \| \leq \left[ \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right]^k \cdot \| X^{(0)} - X \| \]

- Conjugate gradient method has slow convergence if \( \kappa(A) \) is large
  - i.e., \( AX = B \) is ill-conditioned

- In this case, we want to improve convergence rate by pre-conditioning
Pre-Conditioning

- **Key idea**
  - Convert $AX = B$ to another equivalent equation $\tilde{A}\tilde{X} = \tilde{B}$
  - Solve $\tilde{A}\tilde{X} = \tilde{B}$ by conjugate gradient method

- **Important constraints to construct $\tilde{A}\tilde{X} = \tilde{B}$**
  - $\tilde{A}$ is symmetric and positive definite – so that we can solve it by conjugate gradient method
  - $\tilde{A}$ has a small condition number – so that we can achieve fast convergence
Pre-Conditioning

\[ AX = B \]

\[ L^{-1}A \cdot X = L^{-1}B \]

\[ L^{-1}AL^{-T} \cdot L^TX = L^{-1}B \]

\[ \bar{A} \quad \bar{X} \quad \bar{B} \]

- \(L^{-1}AL^{-T}\) is symmetric and positive definite, if A is symmetric and positive definite

\[ (L^{-1}AL^{-T})^T = L^{-1}AL^{-T} \]

\[ X^T L^{-1}AL^{-T}X = (L^{-T}X)^T \cdot A \cdot (L^{-T}X) > 0 \]
Pre-Conditioning

\[
\begin{align*}
L^{-1} A L^{-T} \cdot L^T X &= L^{-1} B \\
\tilde{A} \quad \tilde{X} \quad \tilde{B}
\end{align*}
\]

- \(L^{-1} A L^{-T}\) has a small condition number, if \(L\) is properly selected.

- In theory, \(L\) can be optimally found by Cholesky decomposition.

\[
A = LL^T
\]

\[
L^{-1} A L^{-T} = L^{-1} \cdot LL^T \cdot L^{-T} = I \quad \text{(Identify matrix)}
\]

- However, Cholesky decomposition is not efficient for large, sparse problems.

- If we know Cholesky decomposition, we almost solve the equation – no need to use conjugate gradient method.
Pre-Conditioning

\[ L^{-1} A L^{-T} \cdot L^{-T} X = L^{-1} B \]

\( \tilde{\tilde{A}} \quad \tilde{\tilde{X}} \quad \tilde{\tilde{B}} \)

- In practice, \( L \) can be constructed in many possible ways

- **Diagonal pre-conditioning (or Jacobi pre-conditioning)**
  - Scale \( A \) along coordinate axes

\[ L = \begin{bmatrix} \sqrt{a_{11}} & \sqrt{a_{22}} & \ldots \\ \sqrt{a_{11}} & \sqrt{a_{22}} & \ldots \\ \vdots & \vdots & \ddots \end{bmatrix} \]
Pre-Conditioning

\[
\begin{align*}
L^{-1} A L^{-T} \cdot L^T X &= L^{-1} B \\
\tilde{A} &\quad \tilde{X} &\quad \tilde{B}
\end{align*}
\]

- **Incomplete Cholesky pre-conditioning**

\[
L = \begin{bmatrix}
\times & \times \\
\times & \times \\
\times & \times & \ddots
\end{bmatrix}
\]

- \(L\) is lower-triangular
- Few or no fill-ins are allowed
- \(A \approx LL^T\) (not exactly equal)
Pre-Conditioning

- **Step 1:** start from an initial guess $\tilde{X}^{(0)}$, and set $k = 0$
- **Step 2:** calculate
  
  \[
  \tilde{D}^{(0)} = \tilde{R}^{(0)} = L^{-1}B - L^{-1}AL^{-T}\tilde{X}^{(0)}
  \]

- **Step 3:** update solution
  
  \[
  \tilde{X}^{(k+1)} = \tilde{X}^{(k)} + \tilde{\mu}^{(k)}\tilde{D}^{(k)}
  \]
  where \( \tilde{\mu}^{(k)} = \frac{\tilde{D}^{(k)}T\tilde{R}^{(k)}}{\tilde{D}^{(k)}T L^{-1}AL^{-T}\tilde{D}^{(k)}} \)

- **Step 4:** calculate residual
  
  \[
  \tilde{R}^{(k+1)} = \tilde{R}^{(k)} - \tilde{\mu}^{(k)}L^{-1}AL^{-T}\tilde{D}^{(k)}
  \]

- **Step 5:** determine search direction
  
  \[
  \tilde{D}^{(k+1)} = \tilde{R}^{(k+1)} + \tilde{\beta}_{k+1,k}\tilde{D}^{(k)}
  \]
  where \( \tilde{\beta}_{k+1,k} = \frac{\tilde{R}^{(k+1)}T\tilde{R}^{(k+1)}}{\tilde{D}^{(k)}T \tilde{R}^{(k)}} \)

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

\[
L^{-1} AL^{-T} \cdot L^T X = L^{-1} B
\]

\[
\tilde{A} \quad \tilde{X} \quad \tilde{B}
\]

\[
\tilde{D}^{(0)} = \tilde{R}^{(0)} = L^{-1} B - L^{-1} AL^{-T} \tilde{X}^{(0)}
\]

\[
\tilde{X}^{(k+1)} = \tilde{X}^{(k)} + \tilde{\mu}^{(k)} \tilde{D}^{(k)} \quad \text{where} \quad \tilde{\mu}^{(k)} = \frac{\tilde{D}^{(k)T} \tilde{R}^{(k)}}{\tilde{D}^{(k)T} L^{-1} AL^{-T} \tilde{D}^{(k)}}
\]

\[
\tilde{R}^{(k+1)} = \tilde{R}^{(k)} - \tilde{\mu}^{(k)} L^{-1} AL^{-T} \tilde{D}^{(k)}
\]

\[
\tilde{D}^{(k+1)} = \tilde{R}^{(k+1)} + \tilde{\beta}_{k+1,k} \tilde{D}^{(k)} \quad \text{where} \quad \tilde{\beta}_{k+1,k} = \frac{\tilde{R}^{(k+1)T} \tilde{R}^{(k+1)}}{\tilde{D}^{(k)T} \tilde{R}^{(k)}}
\]

- \( L^{-1} \) should not be explicitly computed
  - Instead, \( Y = L^{-1}W \) or \( Y = L^{-T}W \) (where \( W \) is a vector) should be computed by solving linear equation \( LY = W \) or \( L^TY = W \)
Pre-Conditioning

- **Diagonal pre-conditioning**
  - \( L \) is a diagonal matrix
  - \( Y = L^{-1}W \) or \( Y = L^{-T}W \) can be found by simply scaling

\[
\begin{bmatrix}
\sqrt{a_{11}} & & \\
& \sqrt{a_{22}} & \\
& & \ddots
\end{bmatrix}
\begin{bmatrix}
Y
\end{bmatrix}
= 
\begin{bmatrix}
W
\end{bmatrix}
\]

\[
y_1 = w_1 / \sqrt{a_{11}}
\]
\[
y_2 = w_2 / \sqrt{a_{22}}
\]
\[
\vdots
\]
Pre-Conditioning

- Incomplete Cholesky pre-conditioning
  - $L$ is lower-triangular
  - $Y = L^{-1}W$ or $Y = L^{-T}W$ can be found by backward substitution

\[
\begin{bmatrix}
l_{11} & & \\
l_{21} & l_{22} & \\
l_{31} & l_{32} & \ddots
\end{bmatrix}
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots
\end{bmatrix}
=
\begin{bmatrix}
w_1 \\
w_2 \\
\vdots
\end{bmatrix}
\]

\[
y_1 = w_1 / l_{11}
y_2 = (w_2 - l_{21}y_1) / l_{22}
\vdots
\]
Pre-Conditioning

\[
L^{-1} A L^{-T} \cdot L^T X = L^{-1} B
\]
\[
\tilde{A} \quad \tilde{X} \quad \tilde{B}
\]

- Once \( \tilde{X} \) is known, \( X \) is calculated as \( X = L^{-T} \tilde{X} \)

\[
\begin{bmatrix}
\sqrt{a_{11}} & 0 & 0 \\
0 & \sqrt{a_{22}} & 0 \\
.. & .. & ..
\end{bmatrix}
\begin{bmatrix}
X
\end{bmatrix}
= 
\begin{bmatrix}
\tilde{X}
\end{bmatrix}
\]

\[
x_1 = \frac{\tilde{x}_1}{\sqrt{a_{11}}}
\]
\[
x_2 = \frac{\tilde{x}_2}{\sqrt{a_{22}}}
\]
\[
.. 
\]

Diagonal pre-conditioning

\[
\begin{bmatrix}
l_{11} & l_{21} & l_{31} \\
l_{22} & l_{32} & .. 
\end{bmatrix}
\begin{bmatrix}
X
\end{bmatrix}
= 
\begin{bmatrix}
\tilde{X}
\end{bmatrix}
\]

\[
x_N = \frac{\tilde{x}_N}{l_{NN}}
\]
\[
x_{N-1} = \left(\frac{\tilde{x}_{N-1} - l_{N,N-1} x_N}{l_{N-1,N-1}}\right)
\]
\[
.. 
\]

Incomplete Cholesky pre-conditioning
Conjugate gradient method can be extended to general (i.e., non-quadratic) unconstrained nonlinear optimization

\[
\min_x \frac{1}{2} X^TAX - B^TX + C
\]

Nonlinear programming

\[
\min_x f(X)
\]

Quadratic programming

A number of changes must be made to solve nonlinear optimization problems
Nonlinear 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
  \[ X^{(k+1)} = X^{(k)} + \mu^{(k)} D^{(k)} \quad \text{where} \quad \mu^{(k)} = \frac{D^{(k)T} R^{(k)}}{D^{(k)T} AD^{(k)}} \]
- **Step 4**: calculate residual
  \[ R^{(k+1)} = R^{(k)} - \mu^{(k)} AD^{(k)} \]
- **Step 5**: determine search direction
  \[ D^{(k+1)} = R^{(k+1)} + \beta_{k+1,k} D^{(k)} \quad \text{where} \quad \beta_{k+1,k} = \frac{R^{(k+1)T} R^{(k+1)}}{D^{(k)T} R^{(k)}} \]
- **Step 6**: set $k = k + 1$ and go to Step 3
Nonlinear Conjugate Gradient Method

- New definition of residual

\[ D^{(0)} = R^{(0)} = B - AX^{(0)} \]
\[ R^{(k+1)} = R^{(k)} - \mu^{(k)} AD^{(k)} \]

\[ R^{(k)} = -\nabla f[X^{(k)}] \]

Quadratic programming
Nonlinear programming

- “Residual” is defined by the gradient of f(X)

  - If X* is optimal, \( \nabla f(X^*) = 0 \)
  - \( -\nabla f(X^*) = B - AX \) for quadratic programming
Nonlinear Conjugate Gradient Method

- New formula for conjugate search directions
  \[ D^{(k+1)} = R^{(k+1)} + \beta_{k+1,k} D^{(k)} \]
  \[ \text{where } \beta_{k+1,k} = \frac{R^{(k+1)T} R^{(k+1)}}{D^{(k)T} R^{(k)}} \]

  Quadratic programming

- Ideally, search directions should be computed by Gram-Schmidt conjugation of residues
  - In practice, we often use approximate formulas

  \[ \beta_{k+1,k} = \frac{R^{(k+1)T} R^{(k+1)}}{R^{(k)T} R^{(k)}} \]

  Fletcher-Reeves formula

  \[ \beta_{k+1,k} = \frac{R^{(k+1)T} \left[ R^{(k+1)} - R^{(k)} \right]}{R^{(k)T} R^{(k)}} \]

  Polak-Ribiere formula
Nonlinear Conjugate Gradient Method

- Optimal step size calculated by one-dimensional search:

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

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

- Quadratic programming

- \( \mu^{(k)} \) cannot be calculated analytically
  - Optimize \( \mu^{(k)} \) by one-dimensional search:

\[ \min_{\mu^{(k)}} f\left[ X^{(k+1)} \right] = f\left[ X^{(k)} + \mu^{(k)} D^{(k)} \right] \]
Nonlinear Conjugate Gradient Method

- **Step 1**: start from an initial guess $X^{(0)}$, and set $k = 0$
- **Step 2**: calculate
  
  \[ D^{(0)} = R^{(0)} = -\nabla f[X^{(0)}] \]
- **Step 3**: update solution
  
  \[ \min_{\mu^{(k)}} f[X^{(k)} + \mu^{(k)}D^{(k)}] \quad X^{(k+1)} = X^{(k)} + \mu^{(k)}D^{(k)} \]
- **Step 4**: calculate residual
  
  \[ R^{(k+1)} = -\nabla f[X^{(k+1)}] \]
- **Step 5**: determine search direction (Fletcher-Reeves formula)
  
  \[ \beta_{k+1,k} = \frac{R^{(k+1)^T}R^{(k+1)}}{R^{(k)^T}R^{(k)}} \quad D^{(k+1)} = R^{(k+1)} + \beta_{k+1,k}D^{(k)} \]
- **Step 6**: set $k = k + 1$ and go to Step 3
Nonlinear Conjugate Gradient Method

- Gradient method, conjugate gradient method and Newton method
  - Conjugate gradient method is often preferred for many practical large-scale engineering problems

<table>
<thead>
<tr>
<th></th>
<th>Gradient</th>
<th>Conjugate Gradient</th>
<th>Newton</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st-Order Derivative</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2nd-Order Derivative</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Pre-conditioning</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cost per Iteration</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Convergence Rate</td>
<td>Slow</td>
<td>Fast</td>
<td>Fast</td>
</tr>
<tr>
<td>Preferred Problem Size</td>
<td>Large</td>
<td>Large</td>
<td>Small</td>
</tr>
</tbody>
</table>
Summary

- Conjugate gradient method (Part 4)
  - Pre-conditioning
  - Nonlinear conjugate gradient method