Numerical Techniques in Interior Point Methods for Linear Programming

Voicu Chis, Yang Li, Zhenghue Nie, Doron Pearl,

**Instructor:** Prof. Tamas Terlaky
School of Computational Engineering and School
McMaster University

April 11, 2007
Outline

1. Introduction of LP

2. Interior Point Method for LP
   - Barrier Method
   - Newton Direction

3. Solving the Normal Equation System
   - The (Modified) Cholesky Factorization
   - The sparse Cholesky Factorization: Orderings
   - Dense Columns in $A$
   - Symbolic Factorization

4. Solving Augmented System

5. Refinement techniques
What are Linear Programming?

**Primal Problem:**

\[
\begin{align*}
\min \quad & c^T x \\
\text{(P)} \quad & \text{s.t.} \quad Ax = b \\
\quad & x \geq 0
\end{align*}
\]

where \( x, c \in \mathbb{R}^n \), \( b \in \mathbb{R}^m \), \( A \in \mathbb{R}^{m \times n} \) with \( \operatorname{rank}(A) = m \).

**Dual Problem:**

\[
\begin{align*}
\max \quad & b^T x \\
\text{(D)} \quad & \text{s.t.} \quad A^T y + s = c \\
\quad & s \geq 0
\end{align*}
\]

where \( y \in \mathbb{R}^m \), \( s \in \mathbb{R}^n \).
What are Linear Programming?

**Primal Problem:**

\[
\begin{align*}
& \text{min} & & c^T x \\
& \text{s.t.} & & Ax = b \\
& & & x \geq 0
\end{align*}
\]

where \( x, c \in \mathbb{R}^n \), \( b \in \mathbb{R}^m \), \( A \in \mathbb{R}^{m \times n} \) with \( \text{rank}(A) = m \).

**Dual Problem:**

\[
\begin{align*}
& \text{max} & & b^T x \\
& \text{s.t.} & & A^T y + s = c \\
& & & s \geq 0
\end{align*}
\]

where \( y \in \mathbb{R}^m \), \( s \in \mathbb{R}^n \).
Optimality Conditions

Lagrangian

\[ L(x, y) = c^T x - y^T (Ax - b). \]

Optimality Conditions in LP

\[ Ax = b, \ x \geq 0, \]
\[ A^T y + s = c, \ s \geq 0, \]
\[ Xs = 0, \] (1)

where \( X = \text{diag}\{x_1, x_2, \ldots, x_n\} \).
Outline

1. Introduction of LP

2. Interior Point Method for LP
   - Barrier Method
   - Newton Direction

3. Solving the Normal Equation System
   - The (Modified) Cholesky Factorization
   - The sparse Cholesky Factorization: Orderings
   - Dense Columns in $A$
   - Symbolic Factorization

4. Solving Augmented System

5. Refinement techniques
Barrier Method

Replace the Primal LP with the Primal Barrier Program under the assumption that \((P)\) has a strictly interior feasible solution.

\[
\begin{align*}
\min & \quad c^T x - \mu \sum_{j=1}^{n} \log x_j \\
(P_{\mu}) & \quad s.t \quad Ax = b \\
& \quad x > 0
\end{align*}
\]

where \(\mu \geq 0\) is a barrier parameter.
Write out the Lagrangian

\[ L(x, y, \mu) = c^T x - y^T (Ax - b) - \mu \sum_{j=1}^{n} \ln x_j, \]

and the condition for a stationary point

\[ \nabla_x L(x, y, u) = c - A^T y - \mu X^{-1} e = 0 \]
\[ \nabla_y L(x, y, u) = Ax - b = 0, \]

where \( X^{-1} = \text{diag}\{x_1^{-1}, x_2^{-1}, \ldots, x_n^{-1}\} \).
Barrier Method (Ctd.)

The Optimality Conditions (KKT Conditions) are:

\[ Ax = b, \quad x > 0 \]
\[ A^T y + s = c, \quad s > 0 \]
\[ Xs = \mu e. \]

Theorem

For a sequence \( \mu > 0, k \in N \) with \( \mu \to 0 \), the corresponding solutions \( (x_{\mu_k}, s_{\mu_k}) \) of (2) converge to a pair of solutions \( (x^*, s^*) \).
The Optimality Conditions (KKT Conditions) are:

\[ \begin{align*}
Ax & = b, \quad x > 0 \\
A^T y + s & = c, \quad s > 0 \\
Xs & = \mu e.
\end{align*} \tag{2} \]

**Theorem**

For a sequence \( \mu > 0, k \in \mathbb{N} \) with \( \mu \to 0 \), the corresponding solutions \((x_{\mu k}, s_{\mu k})\) of (2) converge to a pair of solutions \((x^*, s^*)\).
Barrier Method (Ctd.)

The Optimality Conditions (KKT Conditions) are:

\[ Ax = b, \ x > 0 \]
\[ A^T y + s = c, \ s > 0 \]
\[ Xs = \mu e. \]  \hspace{1cm} (2)

Theorem

For a sequence \( \mu > 0, k \in N \) with \( \mu \rightarrow 0 \), the corresponding solutions \( (x_{\mu k}, s_{\mu k}) \) of (2) converge to a pair of solutions \( (x^*, s^*) \).

Nonlinear!!!
Outline

1. Introduction of LP

2. Interior Point Method for LP
   - Barrier Method
   - Newton Direction

3. Solving the Normal Equation System
   - The (Modified) Cholesky Factorization
   - The sparse Cholesky Factorization: Orderings
   - Dense Columns in $A$
   - Symbolic Factorization

4. Solving Augmented System

5. Refinement techniques
Newton Method for Nonlinear System

Rewrite **Optimality Conditions (KKT Conditions)** in the form as follows:

\[
F(x, y, s) = \begin{bmatrix}
Ax & - & b \\
A^T y + s & - & c \\
Xs & - & \mu e
\end{bmatrix} = 0.
\]

**Newton Method**

\[
F(x, y, s) + \nabla F(x, y, s) \begin{bmatrix}
\Delta x \\
\Delta y \\
\Delta s
\end{bmatrix} = 0.
\]
Newton Direction

Note that

\[ \nabla F(x, y, s) = \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix}, \]

Solve this linear system for search direction

\[ \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^T y - s \\ \mu e - Xs \end{bmatrix} \] (3)
Augmented System

Eliminate $\triangle s$ from (3) and use the notation $D = S^{-1/2}X^{1/2}$, we obtain so-called Augmented System:

$$
\begin{bmatrix}
  0 & A \\
  A^T & -D^{-2}
\end{bmatrix}
\begin{bmatrix}
  \triangle y \\
  \triangle x
\end{bmatrix}
= 
\begin{bmatrix}
  -r_b \\
  -r_c + X^{-1}r_{xs}
\end{bmatrix},
$$

(4)

$$
\triangle s = -X^{-1}(r_{xs} + S \triangle x),
$$

(5)

where

$$
\begin{bmatrix}
  r_b \\
  r_c \\
  r_{xs}
\end{bmatrix}
= 
\begin{bmatrix}
  Ax - b \\
  r_c = A^T y + s - c \\
  r_{xs} = Xs - \mu e
\end{bmatrix}
$$
Normal Equations System

Eliminate $\triangle x$ from (9) to obtain the most compact of the three forms:

$$AD^2 A^T \triangle y = -r_b + A(-S^{-1}Xr_c + S^{-1}r_{xs}), \quad (6)$$

$$\triangle s = -r_c - A^T \triangle y \quad (7)$$

$$\triangle x = -S^{-1}(r_{xs} + X \triangle s) \quad (8)$$
Generic Algorithm

Algorithm

**Input parameters:**

\( \mu = \mu_0 \): barrier parameter;
\( \theta \): reduction parameter, \( 0 < \theta < 1 \)
\( \epsilon > 0 \): accuracy parameter;
\( x_0 \): an initial point;

**Step 0.** \( x := x_0, \mu := \mu_0 \);

**Step 1.** If \( \mu < \epsilon \) STOP, \( (x, y, s)^T \) is returned as a solution.

**Step 2.** Calculate (approximately) search direction, and update iterate,
\[
(x, y, s)^T = (x, y, s)^T + \alpha(\Delta x, \Delta y, \Delta s);
\]

**Step 3.** \( \mu := (1 - \theta)\mu \);

**Step 4.** Go to Step 1.
Outline

1. Introduction of LP

2. Interior Point Method for LP
   - Barrier Method
   - Newton Direction

3. Solving the Normal Equation System
   - The (Modified) Cholesky Factorization
   - The sparse Cholesky Factorization: Orderings
   - Dense Columns in A
   - Symbolic Factorization

4. Solving Augmented System

5. Refinement techniques
The classical Cholesky Factorization

Observations:

- $M := AD^2A^T$ is symmetric and positive definite.
- A Cholesky decomposition ($L$) exists

$$M = LL^T$$

$$M = \begin{bmatrix} m_{11} & m_{21}^T \\
 m_{21} & M_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\
 l_{21} & L_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21}^T \\
 0 & L_{22}^T \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}l_{21}^T \\
 l_{11}l_{21} & l_{21}^2 + L_{22}L_{22}^T \end{bmatrix}$$

From this we derive

$$l_{11} = \sqrt{m_{11}}$$
$$l_{21} = \frac{m_{21}}{l_{11}}$$
$$M_{22} = L_{22}L_{22}^T = M_{22} - l_{21}l_{21}^T$$
The classical Cholesky Factorization (Ctd.)

The Cholesky algorithm:

1. \( \text{for } j = 1, \ldots, m \)
2. \( l_{jj} := \sqrt{l_{jj}} \)
3. \( l_{(j+1:m)j} := l_{(j+1:m)j} / l_{jj} \)
4. \( \text{for } k = j + 1, \ldots, m \)
5. \( l_{(k+1:m)k} := l_{(k+1:m)k} - l_{kj} l_{(k+1:m)j} \)

Notes:

- Works if \( M \) is positive definite.
- \( \frac{1}{6}m^3 \) flops.
- Cholesky Factorization = Gauss Elimination using diagonal pivots.
- Numerical stable.
- Problem: \( M \) is only P.S.D.
The Modified Cholesky Factorization

The modified Cholesky algorithm:

1. \( \text{for } j = 1, \ldots, m \)
2. \( \text{if } l_{jj} \leq \varepsilon \)
3. \( l_{jj} := \delta \)
4. \( l_{jj} := \sqrt{l_{jj}} \)
5. \( l_{(j+1:m)j} := l_{(j+1:m)j}/l_{jj} \)
6. \( \text{for } k = j + 1, \ldots, m \)
7. \( l_{(k+1:m)k} := l_{(k+1:m)k} - l_{kj}l_{(k+1:m)j} \)

Notes:

- Choice: \( \varepsilon = 1.0e-12, \delta = 1.0e128 \).
- Analyzed by [Zhang, 1996] and [Wright, 1996]
Outline

1. Introduction of LP

2. Interior Point Method for LP
   - Barrier Method
   - Newton Direction

3. Solving the Normal Equation System
   - The (Modified) Cholesky Factorization
   - The sparse Cholesky Factorization: Orderings
   - Dense Columns in $A$
   - Symbolic Factorization

4. Solving Augmented System

5. Refinement techniques
Fill-in

Cholesky factorization of a sparse matrix usually produces *fill-in*; that is, some lower triangular locations in the factor $L$ contain nonzero elements even though the same locations in the original matrix are zero.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph](image)
Fill-in

Cholesky factorization of a sparse matrix usually produces *fill-in*; that is, some lower triangular locations in the factor L contain nonzero elements even though the same locations in the original matrix are zero.

\[
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
1 & \times & \times & \times & \times & \times \\
2 & \times & \times & + & + & + \\
3 & \times & + & \times & + & + \\
4 & \times & + & + & \times & + \\
5 & \times & + & + & + & \times \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 & 5 \\
1 & \times & \times & \times & \times & \times \\
2 & \times & \times & + & + & + \\
3 & \times & + & \times & + & + \\
4 & \times & + & + & \times & + \\
5 & \times & + & + & + & \times \\
\end{array}
\]
An Example

1 2 3 4 5
1 × × × × ×
2 × × + + + +
3 × + × + + +
4 × + + × +
5 × + + + ×
Troubles by fill-in

Disadvantages:

- Fill-in requires additional storage.
- Fill-in increase the cost of updating the remaining matrix and the triangular substitutions.

Our Goal: *minimizes* fill-in

Bad News: NP-complete

Good News: Inexpensive ordering heuristics available
Minimum-Degree Ordering

Degree: the number of nonzero elements in a column of the matrix, excluding the diagonal.

Motivation: From $L_{22} - l_{21}l_{21}^T$, we see if $l_{11}$ has degree $d$, the update matrix contains $d^2$ nonzeros.

Process: At each step, examine the remaining matrix and select the diagonal element with minimum degree as the pivot at this step.
An Example

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

![Graph](image_url)
## An Example

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>2</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>×</td>
<td>×</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>×</td>
<td></td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>×</td>
<td></td>
<td></td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

![Graph illustration](image)
An Example

\[
\begin{array}{ccccc}
2 & 1 & 3 & 4 & 5 \\
2 & \times & \times & & \\
1 & \times & \times & \times & \times \\
3 & \times & \times & & \\
4 & \times & \times & & \\
5 & \times & & & \\
\end{array}
\]
An Example

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>1</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>✗</td>
<td>✗</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>✗</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Example

\[
\begin{array}{ccccc}
2 & 3 & 4 & 1 & 5 \\
2 & \times & \times & \times & \\
3 & \times & \times & \times & \\
4 & \times & \times & \times & \\
1 & \times & \times & \times & \times & \times & \times \\
5 & \times & \times & \times & \\
\end{array}
\]
An Example

\[
\begin{array}{ccccc}
2 & 3 & 4 & 5 & 1 \\
2 & \times & \times & \times & \\
3 & \times & \times & \times & \\
4 & \times & \times & \times & \\
5 & \times & \times & \times & \\
1 & \times & \times & \times & \times & \times \\
\end{array}
\]
Another Example
Another Example
Minimum Local fill-in

Process: At the $k$-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduces the least amount of fill-in at this stage as pivot.
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.
Minimum Local fill-in

Process: At the \( k \)-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.

1 → 2 → 3
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.

1 $\rightarrow$ 2 $\rightarrow$ 3 $\rightarrow$ 4
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.

1 → 2 → 3 → 4 → 5 → 6
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduce the least amount of fill-in at this stage as pivot.

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8
Minimum Local fill-in

**Process:** At the $k$-th stage, select the diagonal element which introduces the least amount of fill-in at this stage as pivot.

1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9
Comparisons:

Minimum degree ordering faster than minimum local fill-in ordering, but generate denser $L$.

<table>
<thead>
<tr>
<th>Name</th>
<th>Analysis time</th>
<th>Nonzeros in $L$</th>
<th>Flops in thousand</th>
<th>Factorization time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MDO</td>
<td>MFO</td>
<td>MDO</td>
<td>MFO</td>
</tr>
<tr>
<td>25fv47</td>
<td>0.50</td>
<td>1.38</td>
<td>32984</td>
<td>27219</td>
</tr>
<tr>
<td>80bau3b</td>
<td>1.22</td>
<td>2.12</td>
<td>37730</td>
<td>34006</td>
</tr>
<tr>
<td>bnl2</td>
<td>0.91</td>
<td>2.82</td>
<td>59437</td>
<td>56705</td>
</tr>
<tr>
<td>cycle</td>
<td>0.93</td>
<td>1.80</td>
<td>54682</td>
<td>39073</td>
</tr>
<tr>
<td>d2q06c</td>
<td>1.89</td>
<td>5.74</td>
<td>135960</td>
<td>91614</td>
</tr>
<tr>
<td>degen3</td>
<td>20.77</td>
<td>13.33</td>
<td>119403</td>
<td>115681</td>
</tr>
<tr>
<td>df001</td>
<td>37.40</td>
<td>552.44</td>
<td>1632101</td>
<td>1445468</td>
</tr>
<tr>
<td>greenbea</td>
<td>2.21</td>
<td>2.11</td>
<td>47014</td>
<td>45507</td>
</tr>
<tr>
<td>grow22</td>
<td>0.21</td>
<td>0.51</td>
<td>8618</td>
<td>8590</td>
</tr>
<tr>
<td>maros-r7</td>
<td>6.70</td>
<td>47.49</td>
<td>510148</td>
<td>511930</td>
</tr>
<tr>
<td>pilot</td>
<td>5.67</td>
<td>25.18</td>
<td>191704</td>
<td>172264</td>
</tr>
<tr>
<td>pilot87</td>
<td>19.27</td>
<td>110.71</td>
<td>423656</td>
<td>389787</td>
</tr>
<tr>
<td>pilot-wo</td>
<td>0.29</td>
<td>0.58</td>
<td>14904</td>
<td>13887</td>
</tr>
</tbody>
</table>

MDO: Minimum degree ordering
MFO: Minimum local fill-in ordering
Outline

1. Introduction of LP

2. Interior Point Method for LP
   - Barrier Method
   - Newton Direction

3. Solving the Normal Equation System
   - The (Modified) Cholesky Factorization
   - The sparse Cholesky Factorization: Orderings
   - Dense Columns in $A$
   - Symbolic Factorization

4. Solving Augmented System

5. Refinement techniques
\[ M = AD^2A^T = \sum_{j=1}^{n} d_{jj}^2 a_j a_j^T \]

Partition \( A \) and \( D^2 \) as

\[ A = (A_s, A_d), \quad D^2 = \begin{pmatrix} D_s^2 & 0 \\ 0 & D_d^2 \end{pmatrix}, \]

Then

\[ AD^2A^T = (A_s, A_d) \begin{pmatrix} D_s^2 & 0 \\ 0 & D_d^2 \end{pmatrix} \begin{pmatrix} A_s^T \\ A_d^T \end{pmatrix} = A_s D_s^2 A_s^T + A_d D_d^2 A_d^T, \]
Let $P = A_s D_s^2 A_s^T$ and $U = A_d D_d$, then Normal Equation System can be considered as

$$AD^2 A^T \triangle y = (A_s D_s^2 A_s^T + A_d D_d^2 A_d^T) \triangle y = (P + UU^T) \triangle y = b,$$

**Sherman-Morrison formula:**

$$(P + RS^T)^{-1} = P^{-1} - P^{-1}R(I + S^TP^{-1}R)^{-1}S^TP^{-1}$$

Then

$$\triangle y = P^{-1}b - P^{-1}U(I + U^TP^{-1}U)^{-1}U^TP^{-1}b.$$
\[ \triangle y = P^{-1}b - P^{-1}U(I + U^T P^{-1}U)^{-1} U^T P^{-1}b. \]

**Step 1:** \( Px_0 = b. \) (Sparse symmetric positive definite: \( \rho < O(n^3) \) flops)

**Step 2:** \( PX_0 = U. \) (Sparse symmetric positive definite: \( k\rho \) flops)

**Step 3:** \( (I + U^T P^{-1}U)y_0 = U^T x_0 \) (Dense, but very small: \( O(k^3) \) flops)

**Step 4:** \( \triangle y = x_0 - X_0 y_0. \) ((\( k + 1 \))\( \rho + O(k^3) \))

In many practical application, \( \rho = O(m) \) and \( k \leq O(\sqrt{m}) \), then the total computational complexity becomes \( O(m\sqrt{m}) \). Otherwise, solving the dense matrix will cost \( O(m^3) \) flops.
\[ \triangle y = P^{-1} b - P^{-1} U (I + U^T P^{-1} U)^{-1} U^T P^{-1} b. \]

**Step 1:** \( Px_0 = b. \) (Sparse symmetric positive definite: \( \rho < O(n^3) \) flops)

**Step 2:** \( PX_0 = U. \) (Sparse symmetric positive definite: \( k\rho \) flops)

**Step 3:** \( (I + U^T P^{-1} U)y_0 = U^T x_0 \) (Dense, but very small: \( O(k^3) \) flops)

**Step 4:** \( \triangle y = x_0 - X_0 y_0. \) \( ((k + 1)\rho + O(k^3)) \)

In many practical application, \( \rho = O(m) \) and \( k \leq O(\sqrt{m}) \), then the total computational complexity becomes \( O(m\sqrt{m}) \). Otherwise, solving the dense matrix will cost \( O(m^3) \) flops.

\( A_s \) may be singular!
Strategy for Singular $A_s$

Andersen [Andersen, 1996] provides a tricky way to deal with this problem.

**Idea:** Construct a subsidiary matrix $H$ such that $P + HH^T$ is nonsingular any more. Then solve

$$[(P + HH^T) + (-HH^T + UU^T)] \triangle y = b$$
How to construct $H$

Apply Gaussian Elimination on matrix $A_s$ to get the upper triangular matrix $\bar{A}_s$. If $A_s$ is a singular matrix, then there is at least one zero row in $\bar{A}_s$. Assume $i_1, i_2, \ldots, i_{m_1}, m_1 < m$, are the indices of zero rows in matrix $\bar{A}_s$. Let

$$H = (h_{i_1}, h_{i_2}, \ldots, h_{i_{m_1}}),$$

where $h_k$ is the $k$–th unit vector.
Outline

1. Introduction of LP

2. Interior Point Method for LP
   - Barrier Method
   - Newton Direction

3. Solving the Normal Equation System
   - The (Modified) Cholesky Factorization
   - The sparse Cholesky Factorization: Orderings
   - Dense Columns in $A$
   - Symbolic Factorization

4. Solving Augmented System

5. Refinement techniques
What is Symbolic Sparse Cholesky Decomposition?

- Cholesky decomposition for positive definite matrices is unconditionally stable\(^1\).
- It means that we can determine the structure of Cholesky factor without actually computing it.
- Symbolic decomposition is to compute the sparsity structure of the Cholesky decomposition, that is, the locations of the nonzero entries.

Sparse Matrix Decomposition

Sparse symmetric positive definite matrix can be factorized in two phases:

symbolic decomposition To get the structure of the matrix factor $L$
numerical decomposition To efficiently compute the numeric values of $L$ and $D$ using the structure determined.

In many sparse matrix programs, these two decompositions are implemented as two separate tasks$^1$.

Advantages of Symbolic Decomposition

- To build an efficient data structure to store the nonzero entries of $L$ before computing them.
- To reduce the time to handle data structure operators during numerical computing, so most operations during numerical decomposition are float-point operations.
- To design efficient numerical decompositions.
Application on the Implementation of IPM

- **Iteration 0:**
  - Find sparsity pattern of $AA^T$
  - Choose a sparsity preserving ordering
  - Find sparsity pattern of $L$

- **At iteration $k$:**
  - Form $M = ADA^T$
  - Do numerical factorization $M$
  - Do solves.
Methods of Symbolic Decomposition

Column-Cholesky (left-looking) The columns of L are computed one by one. It is the most commonly used form; some packages such as YSMP\textsuperscript{1} and SPARSPAK\textsuperscript{2} provide efficient implementations.

Row-Cholesky (bordering method) The rows of L are computed one by one\textsuperscript{3,4}.

Submatrix-Cholesky (right looking) Once a column $L_j$ has been computed, it immediately generates all contributions to subsequent columns\textsuperscript{5,6}.

---


A Graph-theoretic Approach

A symmetric positive definite $n \times n$ matrix $A$ can be described by the graph $G = (V, E)$, where $V = 1, 2, \ldots, n$ and $E = \{(i, j) | 1 \leq j < i \leq n \& a_{ij} \neq 0\}$.

- $V$ is a set of $n$ vertices, every vertex $i$ is associated with row and column $i$ of $A$.
- $E$ is a set of edges; each edge joins a pair of distinct vertices; the diagonal entries are not described.
- There is an edge $(i, j) \in E$ if and only if $a_{ij}$ is nonzero.

$$A = \begin{pmatrix} \times & \times & \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times & \times \end{pmatrix} \Rightarrow \begin{array}{c} 2 \\ 3 \\ 4 \\ 1 \\ 5 \\ 6 \\ 7 \end{array}$$
Relation of Cholesky Decomposition and Graph

Let $G = (V, E)$ be the graph associated with an $n \times n$ symmetric positive definite matrix $A$. Consider pivoting with $a_{ii} \neq 0$ to eliminate entries in column $i$. Show that the resulting graph $G_1$ is contained in $\hat{G}$, which obtained from $G$ by

- removing vertex $v_i$
- creating a clique on the vertices adjacent to $v_i$

From Problem 8 of Assignment 1 of CES 702 Advanced Computation Methods and Models (2007 Winter)
A sequence of elimination graphs is generated, each represents the sparsity structure of the Schur complement.
Chordal Graph and Elimination Tree

If we put the elimination graphs together, we get a Chordal graph (triangulated graph), which describes the structure of $L$.

$$G^+(A)$$

Elimination tree $T = (V, E)$, associated with the Cholesky factor $L$, is a rooted tree with $V = 1, 2, \cdots, n$ and $E = \{(j, i) | j = \min\{k | (k > i) \& l_{ki} \neq 0\}\}$. 

$\text{parent}(i) = \min\{k > i : (k, i) \in G^+(A)\}$

$\text{parent}(\text{col } i) = \text{first nonzero row below diagonal in } L$
Elimination Tree

- If $a_{ij}$ is nonzero and $i > j$, then $i$ is an ancestor of $j$ in $T(A)$.
- If $l_{ij}$ is nonzero, then $i$ is an ancestor of $j$ in $T(A)$.
- If $(i, k)$ is an edge of $G$ with $i > k$, then the edges of $G^+$ include: $(i, k); (i, p(k)); (i, p(p(k))); (i, p(p(p(k)))) \cdots$
- Let $i > j$. Then $(i, j)$ is an edge of $G^+$ iff $j$ is an ancestor in $T$ of some $k$ such that $(i, k)$ is an edge of $G$. (Construct $G^+$)
- Row structure of $L$ is given by a row subtree of the elimination tree.

Ref:

Computing the Elimination Tree

1. Given the graph $G = G(A)$ of $n$-by-$n$ matrix $A$
2. start with an empty forest (no vertices)
3. for $i = 1 : n$
   add vertex $i$ to the forest
   for each edge $(i, j)$ of $G$ with $i > j$
     make $i$ the parent of the root of the tree containing $j$

\[
A = \begin{pmatrix}
  \times & \times \\
  \times & \times & \times \\
  \times & \times & \times & \times \\
  \times & \times & \times & \times & \times \\
  \times & \times & \times & \times & \times & \times
\end{pmatrix}
\]

Cost: $O(\text{nnz}(A) \times \text{inverse Ackermann function})$
Column Structure

Construct $G^+$: For $i > j$, $(i,j)$ is an edge of $G^+$ iff either $(i,j)$ is an edge of $G$ or $(i,k)$ is an edge of $G^+$ for some child $k$ of $j$ in $T$.

The algorithm for finding structures of columns is called **symbolic factorization**.

$$\text{struct}(L(:,j)) = \text{struct}(A(j : n,j)) \cup (\cup (\text{struct}(L(:,k)) | j = \text{parent}(k)\text{in}T))$$

here, $\text{struct}(j) = \{i : j \leq i < n \land l_{ij} \neq 0\}$

Best Cost: $O(\text{nnz}(L))$
Row Structure

Theorem: Let $i > j$. Then $(i, j)$ is an edge of $G^+$ iff $j$ is an ancestor in $T$ of some $k$ such that $(i, k)$ is an edge of $G$. 
One issue of Implementation

\[ A = LL^T \]

- Traditional
- Straightforward

\[ A = LDL^T \]

- Avoid square roots \(\Rightarrow\) better numerical stability
- Factorize symmetric matrices
- Save memory
Software

- [http://www.cise.ufl.edu/research/sparse/ldl/](http://www.cise.ufl.edu/research/sparse/ldl/)
- Language: C and mex for Matlab
- Test matrices:
  - [http://www.cise.ufl.edu/research/sparse/matrices/](http://www.cise.ufl.edu/research/sparse/matrices/)
  - [http://math.nist.gov/MatrixMarket/](http://math.nist.gov/MatrixMarket/)
More Reference of Symbolic Sparse Cholesky Decomposition

Augmented System

\[
\begin{bmatrix}
0 & A \\
A^T & -D^{-2}
\end{bmatrix}
\begin{bmatrix}
\triangle y \\
\triangle x
\end{bmatrix}
= 
\begin{bmatrix}
-r_b \\
-r_c + X^{-1} r_{xs}
\end{bmatrix},
\]
\[
\triangle s = -X^{-1}(r_{xs} + S \triangle x),
\]

- Symmetric Indefinite, Bunch-Parlett Factorization
- Advantage:
  - No trouble by dense column.
  - More stable, ill conditioning easier to trace the effects on factorization and the quality of solution.
  - More flexible, easily extended to IPMs in NLP.
- Disadvantage:
  - Algorithms and software not as highly developed as sparse Cholesky codes.
  - More computing time (typically, 50%-100% more) than Cholesky.
Reference

Erling D. Andersen, Jacek Gondzio, Csaba Meszaros, Xiaojie Xu
*Implementation of Interior Point Methods for Large Scale Linear Programming*
in Interior point methods of mathematical programming Edited by Tamaas Terlaky, Kluwer, 1996

E.D. Andersen, C. Roos, T. Terlaky, T. Trafalis, and J.P. Warners.
*The use of low-rank updates in interior-point methods.*

E.D. Andersen and K.D. Anderson
*The Mosek Interior point Optimizer for Linear Programming: An Implementation of the Homogeneous Algorithm*

K.D. Andersen
*A Modified Schur Complement Method for Handling Dense Columns in Interior-Point Methods for Linear Programming.*

I.S. Duff A.M. Erisman J.K. Reid
*Direct Methods for Sparse Matrices*
Clarendon Press, Xoford, 1986

Alan George, Joseph W-H Liu
*Computer Solution of Large Sparse Positive Definite Systems*
Reference

W.W. Hager

*Updating the inverse of a matrix*

B. Jansen, C. Roos, T. Terlaky

*A short survey on ten years interior point methods*

S.J. Wright

*Modified Cholesky factorizations in interior-point algorithms for linear programming,*
Preprint ANL/MCS-P600-0596, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, ILL, May 1996.

Yin Zhang

*Solving Large-Scale Linear Programs by Interior-Point Methods Under the MATLAB Environment*
Department of Mathematics and Statistics, University of Maryland Baltimore County, Technical Report TR96-01
Thanks!

Questions?