Sparse Linear Algebra: LU Factorization

Kristin Davies
Peter He
Feng Xie
Hamid Ghaffari

April 4, 2007
Outline

- Introduction to LU Factorization (Kristin)
- LU Transformation Algorithms (Kristin)
- LU and Sparsity (Peter)
- Simplex Method (Feng)
- LU Update (Hamid)
Introduction – What is LU Factorization?

- Matrix decomposition into the product of a lower and upper triangular matrix:

$$A = LU$$

- Example for a 3x3 matrix:

$$\begin{bmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{bmatrix} = \begin{bmatrix}
l_{11} & 0 & 0 \\
l_{21} & l_{22} & 0 \\
l_{31} & l_{32} & l_{33}
\end{bmatrix}\begin{bmatrix}
u_{11} & u_{12} & u_{13} \\
u_{22} & u_{22} & u_{23} \\
u_{33}
\end{bmatrix}$$
Introduction – LU Existence

- LU factorization can be completed on an invertible matrix if and only if all its principle minors are non-zero.

- Recall: invertible matrix

\[
A = \begin{bmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{bmatrix}
\]

\[A \text{ is invertible if } B \text{ exists s.t. } AB = BA = I\]

- Recall: principle minors

\[
\begin{align*}
    & \det(a_{11}) \quad \det(a_{22}) \quad \det(a_{33}) \\
    & \det\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \det\begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix} \quad \det\begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix} \\
    & \det\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}
\end{align*}
\]
Imposing the requirement that the diagonal of either $L$ or $U$ must consist of ones results in **unique** LU factorization.
Introduction – Why LU Factorization?

- LU factorization is useful in numerical analysis for:
  - Solving systems of linear equations (AX=B)
  - Computing the inverse of a matrix

- LU factorization is advantageous when there is a need to solve a set of equations for many different values of B
Transformation Algorithms

- Modified form of Gaussian elimination
- Doolittle factorization – $L$ has 1’s on its diagonal
- Crout factorization – $U$ has 1’s on its diagonal
- Cholesky factorization – $U = L^T$ or $L = U^T$

Solution to $AX = B$ is found as follows:
- Construct the matrices $L$ and $U$ (if possible)
- Solve $LY = B$ for $Y$ using forward substitution
- Solve $UX = Y$ for $X$ using back substitution
Transformations – Doolittle

- Doolittle factorization – $L$ has 1’s on its diagonal
- General algorithm – determine rows of $U$ from top to bottom; determine columns of $L$ from left to right

```plaintext
for i=1:n
    for j=i:n
        $L_{ik} U_{kj} = A_{ij}$ gives row $i$ of $U$
    end
    for j=i+1:n
        $L_{jk} U_{ki} = A_{ji}$ gives column $i$ of $L$
    end
end
```
Transformations – Doolittle example

\[
\begin{pmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{pmatrix}
\begin{pmatrix}
u_{11} \\
u_{22} \\
u_{33}
\end{pmatrix} =
\begin{pmatrix}
2 & -1 & -2 \\
-4 & 6 & 3 \\
-4 & -2 & 8
\end{pmatrix}
\]

for \( j = i : n \)

\[ L_{ik} U_{kj} = A_{ij} \]
gives row \( i \) of \( U \)

end

\[
(1 \ 0 \ 0)
\begin{pmatrix}
u_{11} \\
u_{22} \\
u_{33}
\end{pmatrix} = 2 \quad \Rightarrow \quad u_{11} = 2
\]

Similarly

\[
\begin{pmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{pmatrix}
\begin{pmatrix}
u_{11} \\
u_{12} \\
u_{13}
\end{pmatrix} =
\begin{pmatrix}
2 & -1 & -2 \\
-4 & 6 & 3 \\
-4 & -2 & 8
\end{pmatrix}
\]

\Rightarrow \quad u_{12} = -1 \ , \ u_{13} = -2
Transformations – Doolittle example

\[
\begin{bmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{bmatrix}
\begin{bmatrix}
2 & -1 & -2 \\
0 & u_{22} & u_{23} \\
0 & 0 & u_{33}
\end{bmatrix}
= 
\begin{bmatrix}
2 & -1 & -2 \\
-4 & 6 & 3 \\
-4 & -2 & 8
\end{bmatrix}
\]

for \(j = i + 1 : n\)

\(L_{jk}U_{ki} = A_{ji}\) gives column \(i\) of \(L\)

\[\begin{bmatrix}
2 \\
0 \\
0
\end{bmatrix} = -4 \Rightarrow 2l_{21} = -4 \Rightarrow l_{21} = -2
\]

Similarly

\[
\begin{bmatrix}
1 & 0 & 0 \\
l_{21} & 1 & 0 \\
l_{31} & l_{32} & 1
\end{bmatrix}
\begin{bmatrix}
2 & -1 & -2 \\
0 & u_{22} & u_{23} \\
0 & 0 & u_{33}
\end{bmatrix}
= 
\begin{bmatrix}
2 & -1 & -2 \\
-4 & 6 & 3 \\
-4 & -2 & 8
\end{bmatrix}
\]

\[\Rightarrow l_{31} = -2\]
Transformations – Doolittle example

\[
\begin{bmatrix}
1 & 0 & 0 & u_{11} & u_{12} & u_{13} \\
l_{21} & 1 & 0 & 0 & u_{22} & u_{23} \\
l_{31} & l_{32} & 1 & 0 & 0 & u_{33}
\end{bmatrix}
\begin{bmatrix}
2 & -1 & -2 \\
-4 & 6 & 3 \\
-4 & -2 & 8
\end{bmatrix}
= 
\begin{bmatrix}
1 & 0 & 0 & 2 & -1 & -2 \\
-2 & 1 & 0 & 0 & 4 & -1 \\
-2 & -1 & 1 & 0 & 0 & 3
\end{bmatrix}
\begin{bmatrix}
2 & -1 & -2 \\
-4 & 6 & 3 \\
-4 & -2 & 8
\end{bmatrix}
\]
Transformations – Doolittle example

- Execute algorithm for our example (n=3)

```plaintext
for i=1:n
    for j=i:n
        L_{ik}U_{kj}=A_{ij} \text{ gives row i of U}
    end
    for j=i+1:n
        L_{jk}U_{ki}=A_{ji} \text{ gives column i of L}
    end
end
```

1. \( i=1 \)
   - \( j=1 \Rightarrow L_{1k}U_{k1}=A_{11} \text{ gives row 1 of U} \)
   - \( j=2 \Rightarrow L_{1k}U_{k2}=A_{12} \text{ gives row 1 of U} \)
   - \( j=3 \Rightarrow L_{1k}U_{k3}=A_{13} \text{ gives row 1 of U} \)
   - \( j=1+1=2 \Rightarrow L_{2k}U_{k1}=A_{21} \text{ gives column 1 of L} \)
   - \( j=3 \Rightarrow L_{3k}U_{k1}=A_{31} \text{ gives column 1 of L} \)

2. \( i=2 \)
   - \( j=2 \Rightarrow L_{2k}U_{k2}=A_{22} \text{ gives row 2 of U} \)
   - \( j=3 \Rightarrow L_{2k}U_{k3}=A_{23} \text{ gives row 2 of U} \)
   - \( j=2+1=3 \Rightarrow L_{3k}U_{k2}=A_{32} \text{ gives column 2 of L} \)

3. \( i=3 \)
   - \( j=3 \Rightarrow L_{3k}U_{k3}=A_{33} \text{ gives row 3 of U} \)
Transformations – Crout

- Crout factorization – U has 1’s on its diagonal
- General algorithm – determine columns of L from left to right; determine rows of U from top to bottom (same!?)

```plaintext
for i=1:n
    for j=i:n
        L_{jk}U_{ki} = A_{ji} gives column i of L
    end
    for j=i+1:n
        L_{ik}U_{kj} = A_{ij} gives row i of U
    end
end
```
Transformations – Crout example

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ -2 \end{bmatrix} \Rightarrow l_{11} = 2$$

Similarly

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ -2 \end{bmatrix} \Rightarrow l_{21} = -4 \ , \ l_{31} = -4$$
Transformations – Solution

Once the L and U matrices have been found, we can easily solve our system

\[ \text{LY} = \text{B} \]

\[ \text{UX} = \text{Y} \]

Forward Substitution

\[ y_1 = \frac{b_1}{l_{11}} \]

\[ b_i - \sum_{j=1}^{i-1} l_{ij} x_j \]

\[ y_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij} x_j}{a_{ii}} \text{ for } i = 2, \ldots, n \]

Backward Substitution

\[ x_n = \frac{y_n}{u_{nn}} \]

\[ x_i = \frac{y_i - \sum_{j=i+1}^{n} u_{ij} y_j}{u_{ii}} \text{ for } i = (n-1), \ldots, 1 \]
References - Intro & Transformations

- Doolittle Decomposition of a Matrix. www.engr.colostate.edu/~thompson/hPage/CourseMat/Tutorials/CompMethods/doolittle.pdf
Definition and Storage of Sparse Matrix

- sparse ... many elements are zero
  for example: \( \text{diag}(d_1, \ldots, d_n) \), as \( n \gg 0 \).

- dense ... few elements are zero

- In order to reduce memory burden, introduce some kinds of matrix storage format
Definition and Storage of Sparse Matrix

- Regular storage structure.

<table>
<thead>
<tr>
<th>row</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>4</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>list = column</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>value</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>
Definition and Storage of Sparse Matrix

- A standard sparse matrix format

Subscripts: 1 2 3 4 5 6 7 8 9 10 11
Colptr: 1 4 6 8 10 12
Rowind: 1 3 5 1 4 2 5 1 4 2 5
Value: 1 2 5 -3 4 -2 -5 1 -4 3 6
Structure Decomposition of Sparse Matrix (square)

- $A \leftrightarrow G(A) \leftrightarrow Adj(G)$
- $\forall i, j$ such that $Adj(G)_{ij} \in \{0,1\}$
- Matrix $P$ is of permutation and $G(A) = G(P^T AP)$

$$\text{opt}\{P^T AP | P : G(P^T AP) = G(A)\} \iff \text{opt}\{P^T adj(G)P | P : G(P^T adj(G)P) = G(A)\}$$
Structure Decomposition of Sparse Matrix (square)

1. There is no loop in $G(A)$
   Step 1. No loop and $t = 1$
   Step 2. $\exists$ output vertices
   \[ S_t = \{i_1^t, i_2^t, \ldots, i_{n_t}^t\} \subset V(G) \]
   Step 3.

\[
\text{adj}(G) = \text{adj}(G) \setminus \begin{pmatrix}
  i_1^t & i_2^t & \ldots & i_{n_t}^t \\
  i_1^t & i_2^t & \ldots & i_{n_t}^t \\
\end{pmatrix}
\]
Structure Decomposition of Sparse Matrix (square)

$t = t + 1$, returning to step 2.

- After \( p \) times, \( \{1, 2, \ldots, n\} = S_1 \cup S_2 \cup \ldots \cup S_p \)

(separation of set).

\[
\exists P = \left( P_{S_1}^T, P_{S_2}^T, \ldots, P_{S_p}^T \right)^T
\]

such that \( P^T \text{adj}(G) P \) is a lower triangular block matrix. Therefore \( P^T A P \) is a lower
triangular block matrix.

2. There is a loop in $G(A)$.

If the graph is not strongly connected, in the reachable matrix of $\text{adj}(A)$, there are naught entries.

Step 1. Choose the $j$-th column, $t = 1$, and

$$S_t(j) = \left\{ i \left( R \cap R^T \right) \begin{pmatrix} i \\ j \end{pmatrix} = 1, 1 \leq i \leq n \right\} =$$
Structure Decomposition of Sparse Matrix (square)

\[
= \left\{ i_1^t, i_2^t, \ldots, i_{n_t}^t \right\}
\]

(j ∈ S_t ⊂ \{1, 2, \ldots, n\},

S_t is closed on strong connection.)

Step 2: Choose the \( j_1 \)-th column (\( j_1 \neq j \)),

\( j = j_1, \ t = t + 1, \) returning step 1.

- After \( p \) times,

\[
\{1, 2, \ldots, n\} = S_1 \cup S_2 \cup \ldots \cup S_p
\]
Structure Decomposition of Sparse Matrix (square)

\[ \exists P = \left(P_{s_1}^T \ P_{s_2}^T \ \ldots \ P_{s_p}^T \right)^T, \ \exists P^T \text{adj}(G)P \]

is a lower triangular block matrix. Therefore

\[ P^TAP \]

is a lower triangular block matrix.

- Note of case 2 mentioned above.

\[ \hat{A} = P^T \text{adj}(G)P = \begin{bmatrix}
\hat{A}_{11} & \hat{A}_{12} & \ldots & \hat{A}_{1p} \\
\hat{A}_{21} & \hat{A}_{22} & \ldots & \hat{A}_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
\hat{A}_{p1} & \ldots & \ldots & \hat{A}_{pp}
\end{bmatrix}. \]
Structure Decomposition of Sparse Matrix (square)

\[ B = (b_{ij}) \in \mathbb{R}^{p \times p}, \]

\[ b_{ij} = \begin{cases} 
= 0 & \text{if } i = j \\
= 1 & \text{if } \hat{A}_{ij} \neq 0 \Rightarrow \\
= 0 & \text{if } \hat{A}_{ij} = 0
\end{cases} \]
There is no loop of $G(B)$. According to similar analysis to case 1,

$$P(j_1, j_2, \ldots, j_p)^TBP(j_1, j_2, \ldots, j_p) = \begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix} \in \mathbb{R}^{p \times p}$$

Therefore, $P^TAP$ is a lower triangular block matrix.
Structure Decomposition of Sparse Matrix (square)

- Reducing computation burden on solution to large scale linear system
- Generally speaking, for large scale system, system stability can easily be judged.
- The method is used in hierarchical optimization of Large Scale Dynamic System
Structure Decomposition of Sparse Matrix (General)

Dulmage-Mendelsohn Decomposition

\[ \exists P, Q \quad \text{such that} \]

\[ P^T A Q = \begin{pmatrix}
A_h & X & X \\
A_s & X \\
A_v
\end{pmatrix} \]
Structure Decomposition of Sparse Matrix (General)

- Further, fine decomposition is needed.
  \[ A_h \rightarrow \text{the block diagonal form} \]
  \[ A_v \rightarrow \text{the block diagonal form} \]
  \[ A_s \rightarrow \text{the block upper triangular form} \]
- D-M decomposition can be seen in reference [3].
- Computation and storage: Minimum of filling in
  The detail can be see in reference [2].
LU Factorization Method: Gilbert/Peierls

- left-looking. kth stage computes kth column of L and U

1. \( L = I \)
2. \( U = I \)
3. for \( k = 1:n \)
4. \( s = L \backslash A(:,k) \) \((\text{partial pivoting on } s)\)
5. \( U(1:k,k) = s(1:k) \)
6. \( L(k:n,k) = s(k:n) / U(k,k) \)
7. end
The $k$-th column of $L$ and $U$ computed.

Columns 1 to $k-1$ accessed.

Not modified or computed.
**LU Factorization Method:**

Gilbert/Peierls

- **THEOREM (Gilbert/Peierls).** The entire algorithm for LU factorization of $A$ with partial pivoting can be implemented to run in $O(\text{flops (LU)} + m)$ time on a RAM where $m$ is number of the nonzero entries of $A$.

- **Note:** The theorem expresses that the LU factorization will run in the time within a constant factor of the best possible, but it does not say what the
Sparse lower triangular solve, $x = L \backslash b$

```matlab
x = b;
for j = 1:n
    if x(j) ~= 0, x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j);
end;
--- Total time: $O(n+flops)$
```
Sparse lower triangular solve, $x = L\backslash b$

- $x_j \neq 0 \land l_{ij} \neq 0 \Rightarrow x_i \neq 0$
- $b_i \neq 0 \Rightarrow x_i \neq 0$
- Let $G(L)$ have an edge $j \rightarrow i \land l_{ij} \neq 0$
- If $\beta = \{i | b_i \neq 0\}$ then $\chi = \{i | x_i \neq 0\}$
- Let $\chi = \text{Reach}_{G(L)}(\beta)$ and
- Then

--- Total time: $O(\text{flops})$
Sparse lower triangular solve, $x = L \backslash b$
Sparse lower triangular solve, $x = L \backslash b$
Sparse lower triangular solve, $x = L\backslash b$
Sparse lower triangular solve, $x = L \backslash b$
Sparse lower triangular solve, $x=L\backslash b$

function $x = lsolve(L, b)$
   $\mathcal{X} = \text{Reach}(L, B)$
   $x = b$
   for each $j$ in $\mathcal{X}$
      $x(j+1:n) = x(j+1:n) - L(j+1:n,j) \times x(j)$

function $\mathcal{X} = \text{Reach}(L, B)$
   for each $i$ in $B$ do
      if (node $i$ is unmarked) dfs($i$)

function dfs($j$)
   mark node $j$
   for each $i$ in $\mathcal{L}_j$ do
      if (node $i$ is unmarked) dfs($i$)
push $j$ onto stack for $\mathcal{X}$

Total time: $O(\text{flops})$
References

- Mihalis Yannakakis’ website: http://www1.cs.columbia.edu/~mihalis
Simplex Method
– Problem size

\[ \min \left\{ c^T x \mid Ax = b, \ x \geq 0, \ A \in \mathbb{R}^{m \times n} \right\} \]

- Problem size determined by \( A \)
- On the average, 5~10 nonzeros per column

greenbea.mps of NETLIB
Simplex Method
– Computational Form, Basis

\[ \min \{ c^T x \mid Ax = b, \ x \geq 0, \ A \in \mathbb{R}^{m \times n} \} \]

Note: \( A \) is of full row rank and \( m < n \)

**Basis** (of \( R^m \)): \( m \) linearly independent columns of \( A \)

Basic variables

\[
\begin{bmatrix}
  x_1 & x_2 & x_3 & x_4 \\
  1 & 0 & 1 & 2 \\
  0 & 1 & 2 & 1
\end{bmatrix}
\]
Simplex Method
– Notations

\( \beta \) \quad \text{index set of basic variables}

\( \gamma \) \quad \text{index set of non-basic variables}

\( B := A_\beta \) \quad \text{basis}

\( R := A_\gamma \) \quad \text{non-basis (columns)}

\[
A = [B \mid R], \quad x = \begin{bmatrix} x_\beta \\ x_\gamma \end{bmatrix}, \quad c = \begin{bmatrix} c_\beta \\ c_\gamma \end{bmatrix}
\]
**Simplex Method**  

**– Basic Feasible Solution**

\[ Ax = b \longrightarrow Bx_\beta + Rx_\gamma = b \]

\[ x_\beta = B^{-1}(b - Rx_\gamma) \]

**Basic feasible solution**

\[ x_\gamma = 0, \quad x \geq 0, \quad Ax = b \]

If a problem has an optimal solution, then there is a **basic solution** which is also optimal.
Simplex Method
— Checking Optimality

Objective value:

\[ c^T x = c_\beta^T x_\beta + c_\gamma^T x_\gamma \]

\[ = c_\beta^T B^{-1} b + \left( c_\gamma^T - c_\beta^T B^{-1} R \right) x_\gamma \]

\[ = c_0 + d^T x_\gamma \]

Optimality condition:

\[ d_j \geq 0 \text{ for all } j \in \gamma \]
Simplex Method
– Improving Basic Feasible Solution

- Choose $x_q$ (incoming variable) s.t. $d_q < 0$
- Increase $x_q$ as much as possible

Objective value
$c_0 + d^T x_γ$

- Non-basic variables
  \[
  x_{m+1} = 0 \\
  \vdots \\
  x_q \uparrow \\
  \vdots \\
  x_n = 0
  \]

- Basic variables
  \[
  x_1 \uparrow \\
  \vdots \\
  x_p \rightarrow \\
  \vdots \\
  x_n \uparrow
  \]

Basic variables remain feasible ($\geq 0$) even if $x_q \to \infty$

Objective value is unbounded
Simplex Method
– Improving Basic Feasible Solution

- Choose \( x_q \) (incoming variable) s.t. \( d_q < 0 \)
- increase \( x_q \) as much as possible

Objective value
\[
c_0 + d^T x_\gamma
\]

non-basic variables
\[
\begin{align*}
x_{m+1} &= 0 \\
\vdots \\
x_q &\uparrow \\
\vdots \\
x_n &= 0
\end{align*}
\]

basic variables
\[
\begin{align*}
x_1 &\uparrow \\
\vdots \\
x_p &\rightarrow \\
\vdots \\
x_n &\uparrow
\end{align*}
\]

Unbounded solution

basic variables
\[
\begin{align*}
x_1 &\uparrow \\
\vdots \\
x_p &\downarrow \text{goes to 0 first} \\
\vdots \\
x_n &\uparrow
\end{align*}
\]

neighboring improving basis
Simplex Method
– Basis Updating

Neighboring bases:

\[ B = \begin{bmatrix} b_1, \ldots, b_p, \ldots, b_m \end{bmatrix} \]

\[ \bar{B} = \begin{bmatrix} b_1, \ldots, a, \ldots, b_m \end{bmatrix} \]
Simplex Method
-- Basis Updating

Write $a = \sum_{i=1}^{m} v^i b_i = Bv$ \hspace{1cm} (v = B^{-1}a)

(as the linear combination of the bases)
Simplex Method – Basis Updating

Write $a = \sum_{i=1}^{m} v^i b_i = Bv \quad (v = B^{-1}a)$

$\downarrow$

$b_p = \frac{1}{v^p}a - \sum_{i \neq p} \frac{v^i}{v^p} b_i \quad (v^p \neq 0)$

$B = \begin{bmatrix} b_1, \ldots, b_p, \ldots, b_m \end{bmatrix}$

$\bar{B} = \begin{bmatrix} b_1, \ldots, a, \ldots, b_m \end{bmatrix}$
Simplex Method – Basis Updating

Write \( a = \sum_{i=1}^{m} v^i b_i = Bv \) \((v = B^{-1} a)\)

\[
\begin{align*}
 b_p &= \frac{1}{v^p} a - \sum_{i \neq p} \frac{v^i}{v^p} b_i \\
 b_p &= \bar{B} \eta, \text{ where } \eta = \left[ -\frac{v^1}{v^p}, \ldots, -\frac{v^{p-1}}{v^p}, \frac{1}{v^p}, -\frac{v^{p+1}}{v^p}, \ldots, -\frac{v^m}{v^p} \right]^T
\end{align*}
\]

\( v^p \text{ : pivot element} \)
Simplex Method

\[ a = \sum_{i=1}^{m} v_i b_i = Bv \quad (v = B^{-1}a) \]

\[ b_p = \frac{1}{v_p} a - \sum_{i \neq p} \frac{v_i}{v_p} b_i \]

\[ b_p = \bar{B}\eta, \quad \text{where} \quad \eta = \left[ -\frac{v^1}{v^p}, \ldots, -\frac{v^{p-1}}{v^p}, \frac{1}{v^p}, -\frac{v^{p+1}}{v^p}, \ldots, -\frac{v^m}{v^p} \right]^T \]

\[ B = \bar{B}E, \quad \text{where} \quad E = \left[ e_1, \ldots, e_{p-1}, \eta, e_{p+1}, \ldots, e_m \right] \]

(ascending transformation matrix)
Simplex Method

\[ a = \sum_{i=1}^{m} v^i b_i = B v \quad (v = B^{-1} a) \]

\[ b_p = \frac{1}{v^p} a - \sum_{i \neq p} \frac{v^i}{v^p} b_i \]

\[ b_p = \bar{B} \eta, \quad \text{where} \quad \eta = \begin{bmatrix} -\frac{v^1}{v^p}, \ldots, -\frac{v^{p-1}}{v^p}, \frac{1}{v^p}, -\frac{v^{p+1}}{v^p}, \ldots, -\frac{v^m}{v^p} \end{bmatrix}^T \]

\[ B = \bar{B} E, \quad \text{where} \quad E = \begin{bmatrix} e_1, \ldots, e_{p-1}, \eta, e_{p+1}, \ldots, e_m \end{bmatrix} \]

\[ \bar{B}^{-1} = EB^{-1} \]
Simplex Method
– Basis Updating

\[ E = \begin{bmatrix} e_1, \ldots, e_{p-1}, \eta, e_{p+1}, \ldots, e_m \end{bmatrix} \]

\[ = \begin{bmatrix} 1 & \eta^1 \\ \vdots & \vdots \\ \eta^p & \eta^p \\ \vdots & \vdots \\ \eta^m & 1 \\ 1 & 0 \\ \vdots & \vdots \\ \eta^p & \eta^p \\ 0 & 1 \end{bmatrix} \]

ETM
( Elementary Transformation Matrix )
Simplex Method
– Basis Updating

Basis tends to get denser after each update \((\overline{B}^{-1} = EB^{-1})\)

\[
Ew = \begin{bmatrix}
1 & \eta^1 \\
\vdots & \vdots \\
\eta^p & \vdots \\
\eta^m & 1
\end{bmatrix}
\begin{bmatrix}
w^1 \\
\vdots \\
w^p \\
w^m
\end{bmatrix} = 
\begin{bmatrix}
w^1 + w^p \eta^1 \\
\vdots \\
w^p \eta^p \\
w^m + w^p \eta^m
\end{bmatrix}
\]

\(Ew = w \) if \(w^p = 0\)
Simplex Method
– Algorithm

Steps

1. Find an initial feasible basis $B$
2. Initialization
3. Check optimality
4. Choose incoming variable $x_q$
5. Choose outgoing variable $x_p$
6. Update basis

Major ops

- $B^{-1}b$
- $c^T_B B^{-1}$
- $B^{-1}a_q$
- $E B^{-1}$
## Simplex Method

### Algorithm

<table>
<thead>
<tr>
<th>Steps</th>
<th>Major ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Find an initial feasible basis B</td>
<td>$B^{-1}b$</td>
</tr>
<tr>
<td>2. Initialization</td>
<td></td>
</tr>
<tr>
<td>3. Check optimality</td>
<td>$c_{\beta}^T B^{-1}$</td>
</tr>
<tr>
<td>4. Choose incoming variable $x_q$</td>
<td>$B^{-1}a_q$</td>
</tr>
<tr>
<td>5. Choose outgoing variable $x_p$ (pivot step)</td>
<td>$EB^{-1}$</td>
</tr>
</tbody>
</table>
Choice of pivot (numerical considerations)

- resulting less fill-ins
- large pivot element

Conflicting goals sometimes

In practice, compromise.
Typical operations: \( B^{-1}w, w^T B^{-1} \)

Challenge: sparsity of \( B^{-1} \) could be destroyed by basis update

Need a proper way to represent \( B^{-1} \)

Two ways:
- Product form of the inverse \( (B^{-1} = E_k E_{k-1} \cdots E_1) \) (obsolete)
- LU factorization
Simplex Method
– LU Factorization

- Reduce complexity using LU update
  \( B = \overline{BE}, \overline{B}^{-1} = EB^{-1} \)

  Side effect: more LU factors

- Refactorization
  (reinstate efficiency and numerical accuracy)
Sparse LU Updates in Simplex Method

Hamid R. Ghaffari

April 10, 2007
Outline

LU Update Methods
Preliminaries
Bartels-Golub LU UPDATE
Sparse Bartels-Golub Method
Reid’s Method
The Forrest-Tomlin Method
Suhl-Suhl Method
More Details on the Topic
### Revised Simplex Algorithm

<table>
<thead>
<tr>
<th>Simplex Method</th>
<th>Revised Simplex Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the current basis, $d$</td>
<td>$d = B^{-1}b$</td>
</tr>
<tr>
<td>Choose $x_q$ to enter the basis based on the greatest cost contribution</td>
<td>$\bar{c} = c'_N - c'_B B^{-1} N,$ { $q</td>
</tr>
<tr>
<td>If $x_q$ cannot decrease the cost, $d$ is optimal solution</td>
<td>$\bar{c}_q \geq 0$, $d$ is optimal solution</td>
</tr>
<tr>
<td>Determine $x_p$ that leaves the basis (become zero) as $x_q$ increases.</td>
<td>$w = B^{-1}A_q,$ { $p</td>
</tr>
<tr>
<td>If $x_q$ can increase without causing another variable to leave the basis, the solution is unbounded</td>
<td>If $w_p \leq 0$ for all $i$, the solution is unbounded.</td>
</tr>
<tr>
<td>Update dictionary.</td>
<td>Update $B^{-1}$</td>
</tr>
</tbody>
</table>

**Note:** In general we do not compute the inverse.
Revised Simplex Algorithm

### Simplex Method

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Determine the current basis, ( d )</td>
</tr>
<tr>
<td>2</td>
<td>Choose ( x_q ) to enter the basis based on the greatest cost contribution</td>
</tr>
<tr>
<td>3</td>
<td>If ( x_q ) cannot decrease the cost, ( d ) is optimal solution</td>
</tr>
<tr>
<td>4</td>
<td>Determine ( x_p ) that leaves the basis (become zero) as ( x_q ) increases.</td>
</tr>
<tr>
<td>5</td>
<td>If ( x_q ) can increase without causing another variable to leave the basis, the solution is unbounded</td>
</tr>
<tr>
<td>6</td>
<td>Update dictionary.</td>
</tr>
</tbody>
</table>

### Revised Simplex Method

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( d = B^{-1}b )</td>
</tr>
<tr>
<td>2</td>
<td>( \bar{c} = c'_N - c'_B B^{-1} N, { q</td>
</tr>
<tr>
<td>3</td>
<td>If ( \bar{c}_q \geq 0 ), ( d ) is optimal solution</td>
</tr>
<tr>
<td>4</td>
<td>( w = B^{-1}A_q, { p</td>
</tr>
<tr>
<td>5</td>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded</td>
</tr>
<tr>
<td>6</td>
<td>Update ( B^{-1} )</td>
</tr>
</tbody>
</table>

**Note:** In general we do not compute the inverse.
## Revised Simplex Algorithm

<table>
<thead>
<tr>
<th>Simplex Method</th>
<th>Revised Simplex Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine the current basis, $d$</td>
<td>$d = B^{-1}b$</td>
</tr>
<tr>
<td>Choose $x_q$ to enter the basis based on the greatest cost contribution</td>
<td>$\bar{c} = c'_N - c'_B B^{-1} N$, { $q</td>
</tr>
<tr>
<td>If $x_q$ cannot decrease the cost, $d$ is optimal solution</td>
<td>$\bar{c}_q \geq 0$, $d$ is optimal solution</td>
</tr>
<tr>
<td>Determine $x_p$ that leaves the basis (become zero) as $x_q$ increases.</td>
<td>$w = B^{-1} A_q$, { $p</td>
</tr>
<tr>
<td>If $x_q$ can increase without causing another variable to leave the basis, the solution is unbounded</td>
<td>If $w_p \leq 0$ for all $i$, the solution is unbounded.</td>
</tr>
<tr>
<td>Update dictionary.</td>
<td>Update $B$</td>
</tr>
</tbody>
</table>

**Note:** In general we do not compute the inverse.
Problems with Revised Simplex Algorithm

- The physical limitations of a computer can become a factor.

- Round-off error and significant digit loss are common problems in matrix manipulations (ill-conditioned matrices).

- It also becomes a task in numerical stability.

- It takes $m^2(m - 1)$ multiplications and $m(m - 1)$ additions, a total of $m^3 - m$ floating-point (real number) calculations.

Many variants of the Revised Simplex Method have been designed to reduce this $O(m^3)$-time algorithm as well as improve its accuracy.
Introducing Spike

- If $A_q$ is the entering column, $B$ the original basis and $\bar{B}$ the new basis, then we have

$$\bar{B} = B + (A_q - Be_p)e_q^T,$$
Introducing Spike

- If $A_q$ is the entering column, $B$ the original basis and $\bar{B}$ the new basis, then we have

$$\bar{B} = B + (A_q - Be_p)e_q^T,$$

- Having LU decomposition $B = LU$ we have

$$L^{-1}\bar{B} = U + (L^{-1}A_q - Ue_p)e_q^T,$$
Introducing Spike

- If $A_q$ is the entering column, $B$ the original basis and $\tilde{B}$ the new basis, then we have

$$\tilde{B} = B + (A_q - Be_p)e_q^T,$$

- Having LU decomposition $B = LU$ we have

$$L^{-1}\tilde{B} = U + (L^{-1}A_q - Ue_p)e_q^T,$$
Introducing Spike

- If $A_q$ is the entering column, $B$ the original basis and $\tilde{B}$ the new basis, then we have

$$\tilde{B} = B + (A_q - B e_p) e_q^T,$$

- Having LU decomposition $B = LU$ we have

$$L^{-1} \tilde{B} = U + (L^{-1} A_q - U e_p) e_q^T,$$

- How to deal with this?

The various implementations and variations of the Bartels-Golub generally diverge with the next step: reduction of the spiked upper triangular matrix back to an upper-triangular matrix. (Chvátal, p150)
Introducing Spike

- If $A_q$ is the entering column, $B$ the original basis and $\bar{B}$ the new basis, then we have

$$\bar{B} = B + (A_q - Be_p)e_q^T,$$

- Having LU decomposition $B = LU$ we have

$$L^{-1}\bar{B} = U + (L^{-1}A_q - U e_p)e_q^T,$$

- How to deal with this?

The various implementations and variations of the Bartels-Golub generally diverge with the next step: reduction of the spiked upper triangular matrix back to an upper-triangular matrix. (Chvátal, p150)
The first variant of the Revised Simplex Method was the Bartels-Golub Method.
## Bartels-Golub Method

**Algorithm**

### Revised Simplex Method

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d = B^{-1} b )</td>
<td></td>
</tr>
<tr>
<td>( \bar{c} = c_N' - c_B' B^{-1} N, )</td>
<td>( \bar{c} = c_N' - c_B' U^{-1} L^{-1} N, )</td>
</tr>
<tr>
<td>{ ( q</td>
<td>\bar{c}_q = \min_t (\bar{c}_t) ) }</td>
</tr>
<tr>
<td>( \bar{c}_q \geq 0, d ) is optimal solution</td>
<td>( \bar{c}_q \geq 0, d ) is optimal solution</td>
</tr>
<tr>
<td>( w = B^{-1} A_q, ) ( { p \left</td>
<td>\frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), w_t &gt; 0 \right. } )</td>
</tr>
<tr>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
</tr>
<tr>
<td>Update ( B^{-1} )</td>
<td>Update ( U^{-1} ) and ( L^{-1} )</td>
</tr>
</tbody>
</table>

### Bartels-Golub

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d = U^{-1} L^{-1} b )</td>
<td></td>
</tr>
<tr>
<td>( \bar{c} = c_N' - c_B' U^{-1} L^{-1} N, )</td>
<td></td>
</tr>
<tr>
<td>{ ( q</td>
<td>\bar{c}_q = \min_t (\bar{c}_t) ) }</td>
</tr>
<tr>
<td>( \bar{c}_q \geq 0, d ) is optimal solution</td>
<td></td>
</tr>
<tr>
<td>( w = U^{-1} L^{-1} A_q, ) ( { p \left</td>
<td>\frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), w_t &gt; 0 \right. } )</td>
</tr>
<tr>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
<td></td>
</tr>
<tr>
<td>Update ( U^{-1} ) and ( L^{-1} )</td>
<td></td>
</tr>
</tbody>
</table>
Bartels-Golub Method

Characteristics

- It significantly improved numerical accuracy.

- Can we do better?
Bartels-Golub Method
Characteristics

- It significantly improved numerical accuracy.

- Can we do better? In sparse case, yes.
First take a look at the following facts:

**Column-Eta factorization of triangular matrices:**

\[
\begin{bmatrix}
1 & l_{21} & 1 \\
l_{31} & l_{32} & 1 \\
l_{41} & l_{42} & l_{43} & 1
\end{bmatrix}
= \begin{bmatrix} 1 \\
1 \\
l_{43} & 1 
\end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\
1 & l_{43} \\
l_{42} & 1 
\end{bmatrix} \cdot \begin{bmatrix} 1 & l_{21} & 1 \\
l_{31} & l_{32} & l_{43} \\
l_{41} & l_{42} & 1 
\end{bmatrix}
\]

So \(L\) can be expressed as the multiplication of single-entry eta matrices, and hence, \(L^{-1}\) is also the product of the same matrices with off-diagonal entries negated.
Sparse Bartels-Golub Method

First take a look at the following facts:

Column-Eta factorization of triangular matrices:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
l_{21} & 1 & 1 & 1 \\
l_{31} & l_{32} & 1 & 1 \\
l_{41} & l_{42} & l_{43} & 1
\end{bmatrix}
= \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & l_{43} & 1 \\
1 & l_{32} & 1 & 1 \\
1 & l_{31} & 1 & 1
\end{bmatrix} \cdot \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
l_{42} & 1 & 1 & 1 \\
l_{41} & 1 & 1 & 1
\end{bmatrix}
\]

Single-Entry-Eta Decomposition:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
l_{21} & 1 & 1 & 1 \\
l_{31} & 1 & 1 & 1 \\
l_{41} & 1 & 1 & 1
\end{bmatrix}
= \begin{bmatrix}
1 & 1 & 1 & 1 \\
l_{21} & 1 & 1 & 1 \\
l_{31} & 1 & 1 & 1 \\
l_{41} & 1 & 1 & 1
\end{bmatrix} \cdot \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]
Sparse Bartels-Golub Method

eta matrices

First take a look at the following facts:

Column-Eta factorization of triangular matrices:

\[
\begin{bmatrix}
1 & l_21 & 1 & l_31 & l_32 & 1 & l_41 & l_42 & 1
\end{bmatrix}
= \begin{bmatrix}
1 & \ 1 & \ & \ & \ & \ & \ & \ & 1
\end{bmatrix}
\cdot \begin{bmatrix}
1 & 1 & l_32 & 1 & l_42 & 1 & l_41 & 1 & 1
\end{bmatrix}
\cdot \begin{bmatrix}
1 & 1 & 1
\end{bmatrix}
\]

Single-Entry-Eta Decomposition:

\[
\begin{bmatrix}
l_21 & 1
\end{bmatrix}
= \begin{bmatrix}
l_21 & 1 & 1 & 1
\end{bmatrix}
\cdot \begin{bmatrix}
l_31 & 1 & 1 & 1
\end{bmatrix}
\cdot \begin{bmatrix}
l_41 & 1 & 1 & 1
\end{bmatrix}
\]

So \( L \) can be expressed as the multiplication of single-entry eta matrices, and hence, \( L^{-1} \) is also is the product of the same matrices with off-diagonal entries negated.
## Sparse Bartels-Golub Method

### Algorithm

<table>
<thead>
<tr>
<th>Bartels-Golub Method</th>
<th>Sparse Bartels-Golub Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d = U^{-1}L^{-1}b )</td>
<td>( d = U^{-1} \prod_t \eta_t b )</td>
</tr>
<tr>
<td>( \bar{c} = c'_N - c'_B U^{-1}L^{-1}N, ) ( { q</td>
<td>\bar{c}_q = \min_t (\bar{c}_t) } )</td>
</tr>
<tr>
<td>( \bar{c}_q \geq 0, ) ( d ) is optimal solution</td>
<td>( \bar{c}_q \geq 0, ) ( d ) is optimal solution</td>
</tr>
<tr>
<td>( w = U^{-1}L^{-1}A_q, ) ( { p</td>
<td>\frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), \ w_t &gt; 0 } )</td>
</tr>
<tr>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
</tr>
<tr>
<td>Update ( U^{-1} ) and ( L^{-1} )</td>
<td>Update ( U^{-1} ) and create any necessary eta matrices. If there are too many eta matrices, completely refactor the basis.</td>
</tr>
</tbody>
</table>
## Sparse Bartels-Golub Method

### Algorithm

<table>
<thead>
<tr>
<th>Bartels-Golub Method</th>
<th>Sparse Bartels-Golub Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d = U^{-1}L^{-1}b )</td>
<td>( d = U^{-1} \prod_t \eta_t b )</td>
</tr>
<tr>
<td>( \bar{c} = c_N' - c_B' U^{-1} L^{-1} N, ) ( { q</td>
<td>\bar{c}_q = \min_t (\bar{c}_t) } )</td>
</tr>
<tr>
<td>( \bar{c}_q \geq 0, \ d ) is optimal solution</td>
<td>( \bar{c}_q \geq 0, \ d ) is optimal solution</td>
</tr>
<tr>
<td>( w = U^{-1} L^{-1} A_q, ) ( { p</td>
<td>\frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), \ w_t &gt; 0 } )</td>
</tr>
<tr>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
</tr>
<tr>
<td>Update ( U^{-1} ) and ( L^{-1} )</td>
<td>Update ( U^{-1} ) and create any necessary eta matrices. If there are too many eta matrices, completely refactor the basis.</td>
</tr>
</tbody>
</table>
### Sparse Bartels-Golub Method

#### Algorithm

<table>
<thead>
<tr>
<th>Bartels-Golub Method</th>
<th>Sparse Bartels-Golub Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d = U^{-1} L^{-1} b )</td>
<td>( d = U^{-1} \prod_t \eta_t b )</td>
</tr>
<tr>
<td>( \bar{c} = c'_N - c'_B U^{-1} L^{-1} N ), ( { q</td>
<td>\bar{c}_q = \min_t(\bar{c}_t) } )</td>
</tr>
<tr>
<td>( \bar{c}_q \geq 0, d ) is optimal solution</td>
<td>( \bar{c}_q \geq 0, d ) is optimal solution</td>
</tr>
<tr>
<td>( w = U^{-1} L^{-1} A_q, { p</td>
<td>\frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), w_t &gt; 0 } )</td>
</tr>
<tr>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
</tr>
<tr>
<td>Update ( U^{-1} ) and ( L^{-1} )</td>
<td>Update ( U^{-1} ) and create any necessary eta matrices. If there are too many eta matrices, completely refactor the basis.</td>
</tr>
</tbody>
</table>
### Sparse Bartels-Golub Method

#### Algorithm

<table>
<thead>
<tr>
<th>Bartels-Golub Method</th>
<th>Sparse Bartels-Golub Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d = U^{-1}L^{-1}b )</td>
<td>( d = U^{-1} \prod_t \eta_t b )</td>
</tr>
</tbody>
</table>
| \( \tilde{c} = c_N' - c_B' U^{-1}L^{-1}N, \)  
\( \{ q | \tilde{c}_q = \min_t (\tilde{c}_t) \} \) | \( \tilde{c} = c_N' - c_B' U^{-1} \prod_t \eta_t N, \)  
\( \{ q | \tilde{c}_q = \min_t (\tilde{c}_t) \} \) |
| \( \tilde{c}_q \geq 0, \ d \) is optimal solution | \( \tilde{c}_q \geq 0, \ d \) is optimal solution |
| \( w = U^{-1}L^{-1}A_q, \)  
\( \{ p | \frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), \ w_t > 0 \} \) | \( w = U^{-1} \prod_t \eta_t A_q, \)  
\( \{ p | \frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), \ w_t > 0 \} \) |
| If \( w_p \leq 0 \) for all \( i \), the solution is unbounded. | If \( w_p \leq 0 \) for all \( i \), the solution is unbounded. |
| Update \( U^{-1} \) and \( L^{-1} \) | Update \( U^{-1} \) and create any necessary eta matrices.  
If there are too many eta matrices, completely refactor the basis. |
Sparse Bartels-Golub Method Advantages

- It is no more complex than the Bartels-Golub Method.
Sparse Bartels-Golub Method Advantages

- It is no more complex than the Bartels-Golub Method.

- Instead of just $L$ and $U$, the factors become the lower-triangular eta matrices and $U$. 
Sparse Bartels-Golub Method Advantages

- It is no more complex than the Bartels-Golub Method.

- Instead of just $L$ and $U$, the factors become the lower-triangular eta matrices and $U$.

- The eta matrices were reduced to single-entry eta matrices.
Sparse Bartels-Golub Method Advantages

- It is no more complex than the Bartels-Golub Method.
- Instead of just $L$ and $U$, the factors become the lower-triangular eta matrices and $U$.
- The eta matrices were reduced to single-entry eta matrices.
- Instead of having to store the entire matrix, it is only necessary to store the location and value of the off-diagonal element for each matrix.
Sparse Bartels-Golub Method Advantages

- It is no more complex than the Bartels-Golub Method.

- Instead of just $L$ and $U$, the factors become the lower-triangular eta matrices and $U$.

- The eta matrices were reduced to single-entry eta matrices.

- Instead of having to store the entire matrix, it is only necessary to store the location and value of the off-diagonal element for each matrix.

- Refactorizations occur less than once every $m$ times, so the complexity improves significantly to $O(m^2)$. 
Sparse Bartels-Golub Method Disadvantages

- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.

- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.

- In practice, in solving large sparse problems, the basis is refactorized quite frequently, often after every twenty iterations or so. (Chvátal, p. 111)

- If the spike always occurs in the first column and extends to the bottom row, the Sparse Bartels-Golub Method becomes worse than the Bartels-Golub Method.

- The upper-triangular matrix will always be fully-decomposed resulting in huge amounts of fill-in;

- Large numbers of eta matrices;

- $O(n^3)$-cost decomposition;
Sparse Bartels-Golub Method Disadvantages

- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.

- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.
Sparse Bartels-Golub Method Disadvantages

- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.

- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.

- In practice, in solving large sparse problem, the basis is refactorized quite frequently, often after every twenty iterations of so. (Chvátal, p. 111)
Sparse Bartels-Golub Method Disadvantages

- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.

- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.

- In practice, in solving large sparse problem, the basis is refactorized quite frequently, often after every twenty iterations of so. (Chvátal, p. 111)

- If the spike always occurs in the first column and extends to the bottom row, the Sparse Bartels-Golub Method becomes worse than the Bartels-Golub Method.
Sparse Bartels-Golub Method Disadvantages

- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.

- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.

- In practice, in solving large sparse problem, the basis is refactorized quite frequently, often after every twenty iterations of so. (Chvátal, p. 111)

- If the spike always occurs in the first column and extends to the bottom row, the Sparse Bartels-Golub Method becomes worse than the Bartels-Golub Method.
  - The upper-triangular matrix will always be fully-decomposed resulting in huge amounts of fill-in;
Sparse Bartels-Golub Method Disadvantages

- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.

- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.

- In practice, in solving large sparse problem, the basis is refactorized quite frequently, often after every twenty iterations of so. (Chvátal, p. 111)

- If the spike always occurs in the first column and extends to the bottom row, the Sparse Bartels-Golub Method becomes worse than the Bartels-Golub Method.
  - The upper-triangular matrix will always be fully-decomposed resulting in huge amounts of fill-in;
  - Large numbers of eta matrices;
Sparse Bartels-Golub Method Disadvantages

- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.
- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.
- In practice, in solving large sparse problem, the basis is refactorized quite frequently, often after every twenty iterations of so. (Chvátal, p. 111)
- If the spike always occurs in the first column and extends to the bottom row, the Sparse Bartels-Golub Method becomes worse than the Bartels-Golub Method.
  - The upper-triangular matrix will always be fully-decomposed resulting in huge amounts of fill-in;
  - Large numbers of eta matrices;
  - $O(n^3)$-cost decomposition;
Ried Suggestion on Sparse Bartels-Golub Method

Rather than completely refactoring the basis, applying LU-decomposition only to the part of that remained upper-Hessenberg.
Reid’s Method

**Task:** The task is to find a way to reduce that bump before attempting to decompose it;

- **Row singleton:** any row of the bump that only has one non-zero entry.
- **Column singleton:** any column of the bump that only has one non-zero entry.

**Method:**

- When a column singleton is found, in a bump, it is moved to the top left corner of the bump.
- When a row singleton is found, in a bump, it is moved to the bottom right corner of the bump.
Reid’s Method

Column Rotation
Reid’s Method
Row Rotation
Reid’s Method

Characteristics

**Advantages:**
- It significantly reduces the growth of the number of eta matrices in the Sparse Bartels-Golub Method
- So, the basis should not need to be decomposed nearly as often.
- The use of LU-decomposition on any remaining bump still allows some attempt to maintain stability.

**Disadvantages:**
- The rotations make absolutely no allowance for stability whatsoever,
- So, Reid’s Method remains numerically less stable than the Sparse Bartels-Golub Method.
The Forrest-Tomlin Method
The Forrest-Tomlin Method

<table>
<thead>
<tr>
<th>Bartels-Golub Method</th>
<th>Forrest-Tomlin Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ d = U^{-1}L^{-1}b ]</td>
<td>[ d = U^{-1} \prod_t R_t L^{-1}b ]</td>
</tr>
<tr>
<td>[ \tilde{c} = c'_N - c'_B U^{-1}L^{-1}N, ] [ { q</td>
<td>\tilde{c}_q = \min_t (\tilde{c}_t) } ]</td>
</tr>
<tr>
<td>[ \tilde{c}_q \geq 0, \ d \text{ is optimal solution} ]</td>
<td>[ \tilde{c}_q \geq 0, \ d \text{ is optimal solution} ]</td>
</tr>
<tr>
<td>[ w = U^{-1}L^{-1}A_q, ] [ \left{ p \mid \frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), \ w_t &gt; 0 \right} ]</td>
<td>[ w = U^{-1} \prod_t R_t L^{-1}A_q, ] [ \left{ p \mid \frac{d_i}{w_i} = \min_t \left( \frac{d_t}{w_t} \right), \ w_t &gt; 0 \right} ]</td>
</tr>
<tr>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
<td>If ( w_p \leq 0 ) for all ( i ), the solution is unbounded.</td>
</tr>
<tr>
<td>Update ( U^{-1} ) and ( L^{-1} )</td>
<td>Update ( U^{-1} ) creating a row factor as necessary. If there are too many factors, completely refactor the basis.</td>
</tr>
</tbody>
</table>
The Forrest-Tomlin Method Characteristics

Advantages:

▸ At most one row-eta matrix factor will occur for each iteration where an unpredictable number occurred before.
▸ The code can take advantage of such knowledge for predicting necessary storage space and calculations.
▸ Fill-in should also be relatively slow, since fill-in can only occur within the spiked column.

Disadvantages:

▸ Sparse Bartels-Golub Method allowed LU-decomposition to pivot for numerical stability, but Forrest-Tomlin Method makes no such allowances.
▸ Therefore, severe calculation errors due to near-singular matrices are more likely to occur.
Suhl-Suhl Method

This method is a modification of Forrest-Tomlin Method.
Leena M. Suhl, Uwe H. Suhl
*A fast LU update for linear programming.*

Stiven S. Morgan
*A Comparison of Simplex Method Algorithms*
University of Florida, 1997

Vasek Chvátal
*Linear Programming*
W.H. Freeman & Company (September 1983)
Thanks