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:

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

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

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

$\text{A is invertible if B exists s.t.}$

$$AB = BA = I$$

Recall: principle minors

$$\det(a_{11}) \qquad \det(a_{22}) \qquad \det(a_{33})$$

$$\det\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \quad \det\begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix} \quad \det\begin{pmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{pmatrix}$$

$$\det\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Introduction – LU Unique Existence

- Imposing the requirement that the diagonal of either L or U must consist of ones results in **unique** LU factorization

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

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

```
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

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

```
for j=i:n
   LikUkj=Aij gives row i of U
end
```

$$\begin{pmatrix} 1 & 0 & 0 \end{pmatrix}\begin{pmatrix} u_{11} \\ 0 \\ 0 \end{pmatrix} = 2 \quad \Rightarrow \quad u_{11} = 2$$

Similarly

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

$$\Rightarrow \quad u_{12} = -1 \;,\; u_{13} = -2$$

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
   LjkUki=Aji gives column i of L
end
```

$$\begin{pmatrix} l_{21} & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 0 \end{pmatrix} = -4 \quad \Rightarrow \quad 2l_{21} = -4 \quad \Rightarrow \quad l_{21} = -2$$

Similarly

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

$$\Rightarrow \quad l_{31} = -2$$

Transformations – Doolittle example

$$\begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix}\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

1^{st} row of U

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

1^{st} coln of L

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

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -2 & l_{32} & 1 \end{bmatrix}\begin{bmatrix} 2 & -1 & -2 \\ 0 & 4 & -1 \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

2^{nd} row of U

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -2 & -1 & 1 \end{bmatrix}\begin{bmatrix} 2 & -1 & -2 \\ 0 & 4 & -1 \\ 0 & 0 & u_{33} \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

2^{nd} coln of L

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -2 & -1 & 1 \end{bmatrix}\begin{bmatrix} 2 & -1 & -2 \\ 0 & 4 & -1 \\ 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

3^{rd} row of U

Transformations – Doolittle example

- Execute algorithm for our example (n=3)

```
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
```

i=1

j=1 → $L_{1k}U_{k1}=A_{11}$ gives row 1 of U

j=2 → $L_{1k}U_{k2}=A_{12}$ gives row 1 of U

j=3 → $L_{1k}U_{k3}=A_{13}$ gives row 1 of U

j=1+1=2 → $L_{2k}U_{k1}=A_{21}$ gives column 1 of L

j=3 → $L_{3k}U_{k1}=A_{31}$ gives column 1 of L

i=2

j=2 → $L_{2k}U_{k2}=A_{22}$ gives row 2 of U

j=3 → $L_{2k}U_{k3}=A_{23}$ gives row 2 of U

j=2+1=3 → $L_{3k}U_{k2}=A_{32}$ gives column 2 of L

i=3

j=3 → $L_{3k}U_{k3}=A_{33}$ 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!?)

```
for i=1:n
    for j=i:n
          LjkUki=Aji gives column i of L
    end
    for j=i+1:n
          LikUkj=Aij gives row i of U
    end
end
```

(In the code: $L_{jk}U_{ki}=A_{ji}$ gives column i of L; $L_{ik}U_{kj}=A_{ij}$ gives row i of U.)

Transformations – Crout example

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}\begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

```
for j=i:n
 LjkUki=Aji gives column i of L
end
```

$$\begin{pmatrix} l_{11} & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = 2 \quad \Rightarrow \quad l_{11} = 2$$

Similarly

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix}\begin{bmatrix} 1 & u_{12} & u_{13} \\ 0 & 1 & u_{23} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & -1 & -2 \\ -4 & 6 & 3 \\ -4 & -2 & 8 \end{bmatrix}$$

$$\Rightarrow \quad l_{21} = -4 \ , \ \ l_{31} = -4$$

Transformations – Solution

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

AX=B → AUX=B

LY=B

$$\begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Forward Substitution

$$y_1 = \frac{b_1}{l_{11}}$$

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

UX=Y

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Backward Substitution

$$x_n = \frac{y_n}{u_{nn}}$$

$$x_i = \frac{y_i - \sum_{j=i+1}^{n} u_{ij} y_j}{u_{ii}} \quad for\, i = (n-1),\ldots 1$$

References - Intro & Transformations

- Module for Forward and Backward Substitution. http://math.fullerton.edu/mathews/n2003/BackSubstitutionMod.hml
- Forward and Backward Substitution. www.ac3.edu/papers/Khoury-thesis/node13.html
- LU Decomposition. http://en.wikipedia.org/wiki/LU_decomposition
- Crout Matrix Decomposition. http://en.wikipedia.org/wiki/Crout_matrix_decomposition
- 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: diag(d1,…,dn), as n>>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.

$$\text{list} = \begin{array}{l|cccccc} \text{row} & 1 & 1 & 2 & 2 & 4 & 4 \\ \text{column} & 3 & 5 & 3 & 4 & 2 & 3 \\ \text{value} & 3 & 4 & 5 & 7 & 2 & 6 \end{array}$$

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

$$\begin{matrix} 1 & -3 & 0 & -1 & 0 \\ 0 & 0 & -2 & 0 & 3 \\ 2 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & -4 & 0 \\ 5 & 0 & -5 & 0 & 6 \end{matrix}$$

Structure Decomposition of Sparse Matrix (square)

- $A \leftrightarrow G(A) \leftrightarrow Adj(G)$

  $Adj(G)\begin{pmatrix} i \\ j \end{pmatrix} \in \{0,1\}$

- Matrix P is of permutation and $G(A) = G(P^T AP)$

$$opt\{P^T AP \,|\, P : G(P^T AP) = G(A)\} \Leftrightarrow$$

$$opt\{P^T adj(G)P \,|\, P : G(P^T adj(G)P) = G(A)\}$$

Structure Decomposition of Sparse Matrix (square)

- 1. There is no loop in G(A)

Step 1. No loop and t = 1

Step 2. ∃ output vertices

$$S_t = \{ i_1^t \quad i_2^t \quad . \quad . \quad . \quad i_{n_t}^t \} \subset V(G)$$

Step 3.

$$adj(G) = adj(G) \setminus adj(G) \begin{pmatrix} i_1^t & i_2^t & . & . & . & i_{n_t}^t \\ i_1^t & i_2^t & . & . & . & i_{n_t}^t \end{pmatrix}$$

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 $P^T \mathrm{adj}(G) P$ is a lower triangular

block matrix . Therefore $P^T A P$ is a lower

Structure Decomposition of Sparse Matrix (square)

triangular block matrix .

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

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

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

$$S_t(j) = \left\{ i \,\middle|\, (R \cap R^T) \begin{pmatrix} i \\ j \end{pmatrix} = 1, 1 \le i \le n \right\} =$$

Structure Decomposition of Sparse Matrix (square)

$$=\left\{ i_1^t \quad i_2^t \quad . \quad . \quad . \quad i_{n_t}^t \right\}$$

$(j \in S_t \subset \{1 \quad 2 \quad . \quad . \quad . \quad n\},$

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)

$$\exists P = (P_{s_1}^T \quad P_{s_2}^T \quad . \quad . \quad . \quad P_{s_p}^T)^T, \ni P^T adj(G) P$$

is a lower triangular block matrix. Therefore

$P^T A P$ is a lower triangular block matrix.

- Note of case 2 mentioned above.

$$\hat{A} = P^T adj(G) P = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} & . & . & . & \hat{A}_{1p} \\ \hat{A}_{21} & \hat{A}_{22} & . & . & . & \hat{A}_{2p} \\ . & . & . & . & . & . \\ \hat{A}_{p1} & . & . & . & . & \hat{A}_{pp} \end{bmatrix}.$$

Structure Decomposition of Sparse Matrix (square)

$$B = (b_{ij}) \in R^{p \times p},$$

$$b_{ij} = \begin{cases} = 0 & i = j \\ = 1 & \hat{A}_{ij} \neq 0. \Rightarrow \\ = 0 & \hat{A}_{ij} = 0 \end{cases}$$

Structure Decomposition of Sparse Matrix (square)

There is no loop of G(B). According to similar analysis to case 1,

$$P(j_1 \quad j_2 \quad . \quad . \quad . \quad j_p)^T B P(j_1 \quad j_2 \quad . \quad . \quad . \quad j_p) = \begin{pmatrix} \times & & & \\ . & \times & & \\ \times & . & \times & \\ \times & \times & . & \times \end{pmatrix} \in R^{p \times p}$$

. Therefore, $P^T A P$ 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

- $\exists P,Q$ such that

$$P^T A Q = \begin{pmatrix} A_h & X & X \\ & A_s & X \\ & & A_v \end{pmatrix}$$

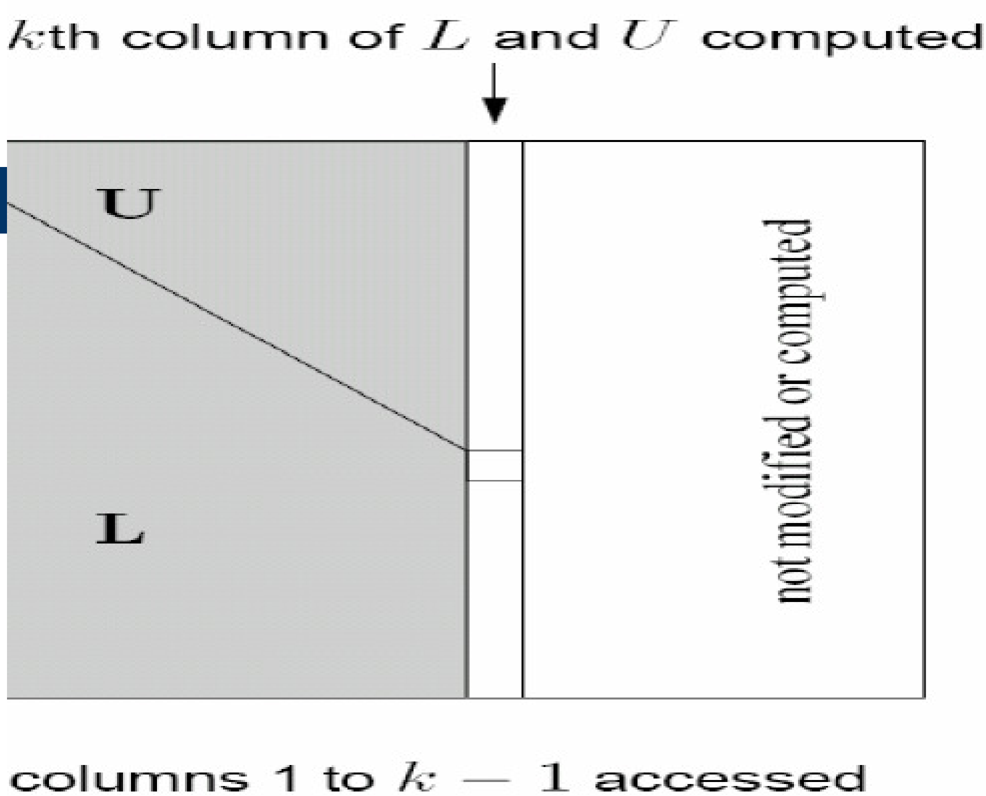
Structure Decomposition of Sparse Matrix (General)

- Further, fine decomposition is needed.
  $A_h \rightarrow$ the block diagonal form
  $A_v \rightarrow$ the block diagonal form
  $A_s \rightarrow$ 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 \ A(:,k)
5. (partial pivoting on s)
6. U(1:k,k) = s(1:k)
7. L(k:n,k) = s(k:n) / U(k,k)
8. end


kth column of L and U computed
U
L
not modified or computed
columns 1 to $k-1$ accessed

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(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\b

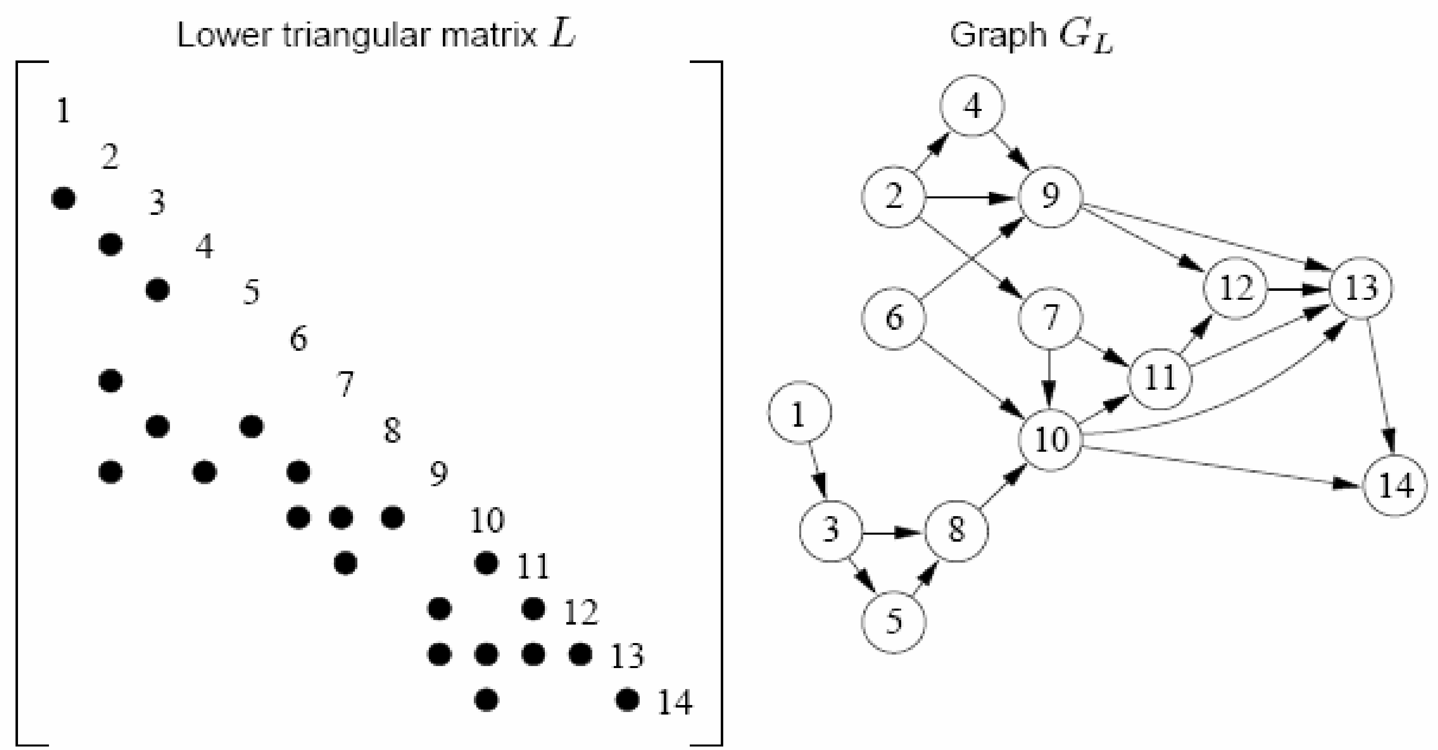
```
x = b;
for j = 1:n
    if (x(j) ≠ 0) , x(j+1:n) =
 x(j+1:n) - L(j+1:n,j) * x(j);
end;
```

--- Total time: O(n+flops)

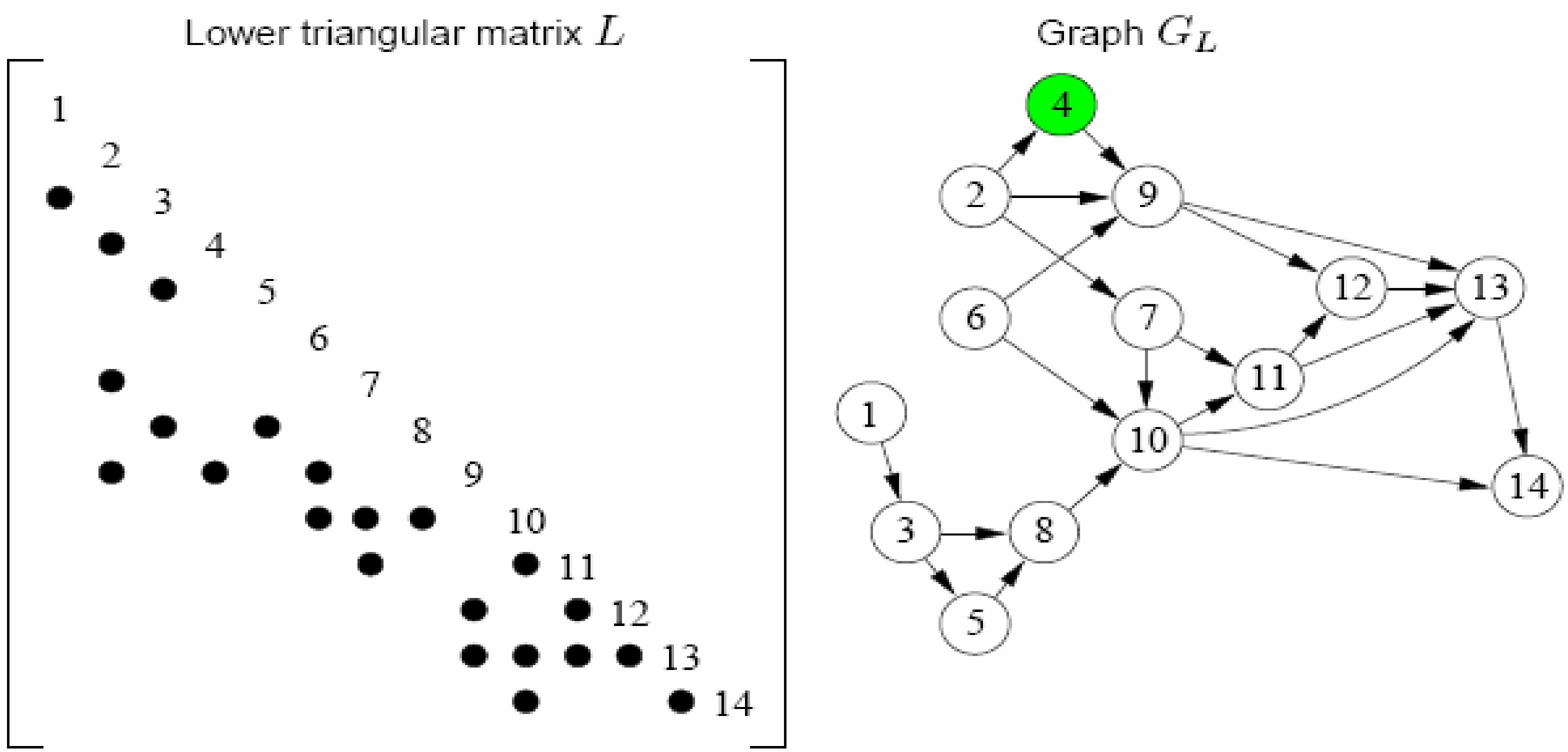
Sparse lower triangular solve, x=L\b

- $x_j \neq 0 \wedge l_{ij} \neq 0 \Rightarrow x_i \neq 0$
- $b_i \neq 0 \Rightarrow x_i \neq 0$
- Let $G(L)$ have an edge $j \rightarrow i$ $l_{ij} \neq 0$
  if $\beta = \{i \mid b_i \neq 0\}$ $\chi = \{i \mid x_i \neq 0\}$
- Let $\chi = \mathrm{Reach}_{G(L)}(\beta)$ and
- Then
  ---Total time: O(flops)

Sparse lower triangular solve, x=L\b

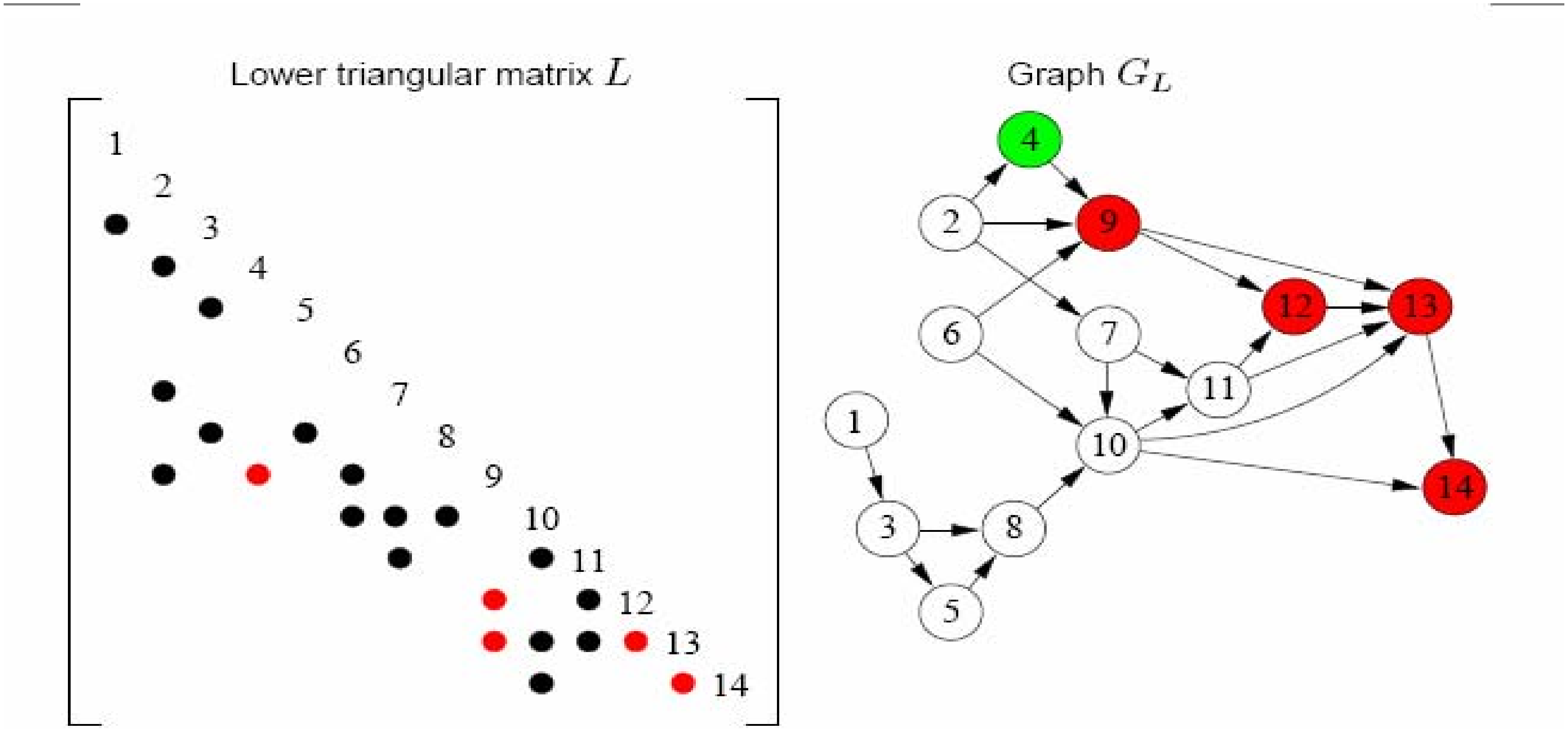


Sparse lower triangular solve, x=L\b



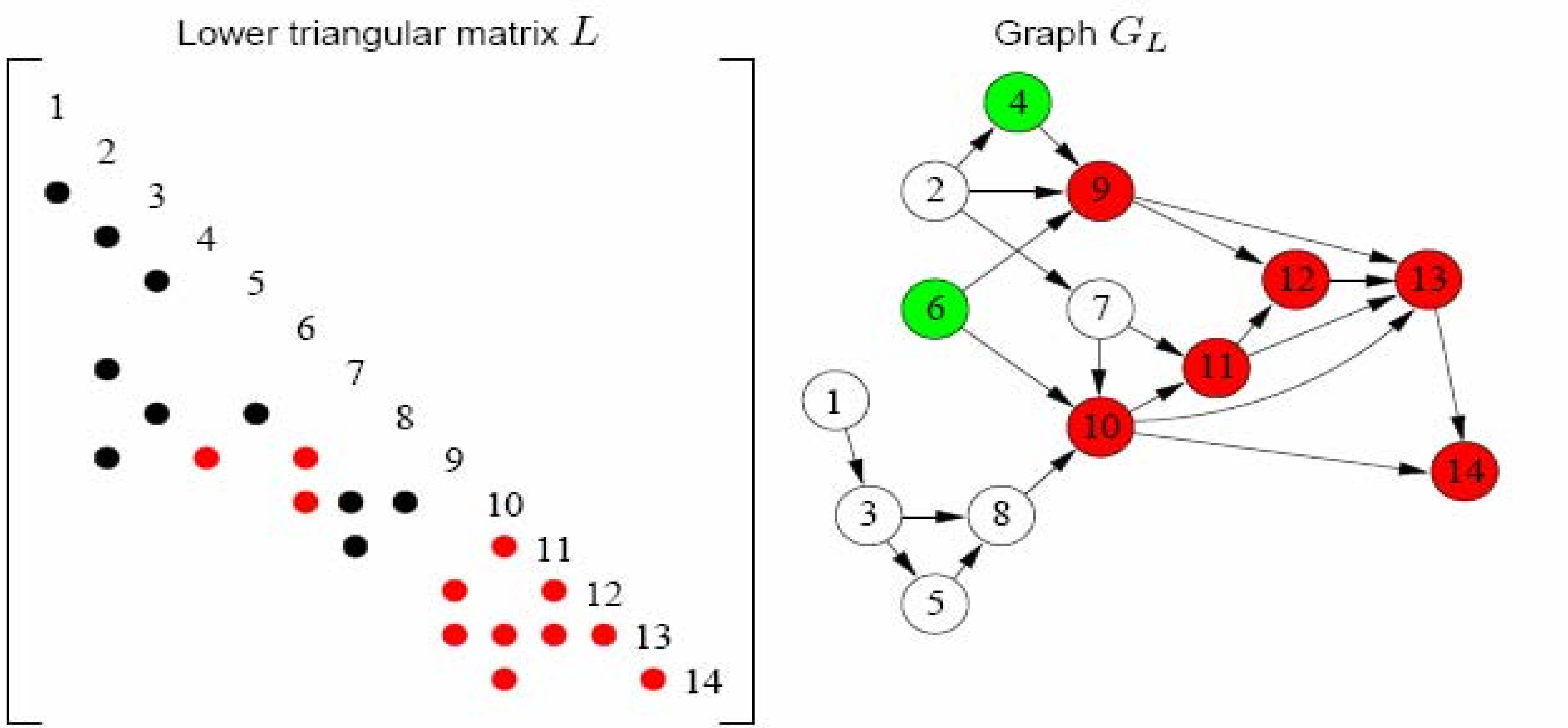
If $\mathcal{B} = \{4\}$

Sparse lower triangular solve, x=L\b



If $\mathcal{B} = \{4\}$
then $\mathcal{X} = \{4, 9, 12, 13, 14\}$

Sparse lower triangular solve, x=L\b



If $\mathcal{B} = \{4, 6\}$
then $\mathcal{X} = \{6, 10, 11, 4, 9, 12, 13, 14\}$

Sparse lower triangular solve, x=L\b

function `x = lsolve(L,b)`

    $\mathcal{X} = \text{Reach}(L, \mathcal{B})$

    `x = b`

    for each j in $\mathcal{X}$

        `x(j+1:n) = x(j+1:n) - L(j+1:n,j) * x(j)`

function $\mathcal{X} = \text{Reach}(\mathtt{L}, \mathcal{B})$

    for each i in $\mathcal{B}$ 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 $\mathcal{X}$

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

References

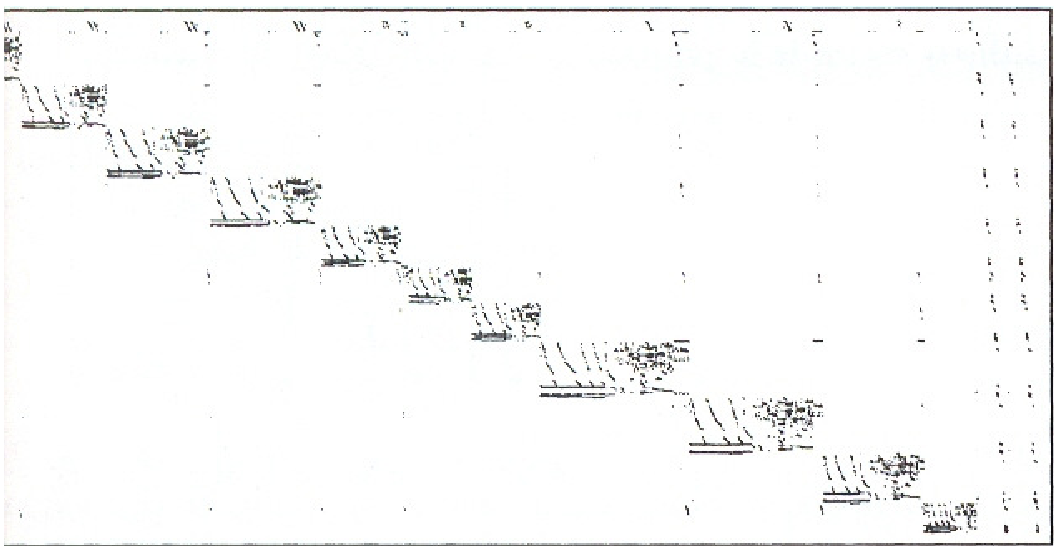
- J. R. Gilbert and T. Peierls, Sparse partial pivoting in time proportional to arithmetic operations, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 862-874
- Mihalis Yannakakis' website:

  http://www1.cs.columbia.edu/~mihalis
- A. Pothen and C. Fan, Computing the block triangular form of a sparse matrix, ACM Trans. On Math. Soft. Vol. 18, No.4, Dec. 1990, pp. 303-324

Simplex Method
– Problem size

$$\min\left\{c^T x \,\middle|\, Ax = b,\ x \ge 0,\ A \in R^{m\times n}\right\}$$

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



greenbea.mps
of NETLIB

Simplex Method
– Computational Form, Basis

$$\min\left\{c^T x \middle| Ax = b,\ x \geq 0,\ A \in R^{m\times n}\right\}$$

Note: ***A*** is of full row rank and ***m<n***

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

$\downarrow$

Basic variables

$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array}$$
$$\begin{bmatrix} 1 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 \end{bmatrix}$$

Simplex Method
– Notations

β	index set of basic variables
γ	index set of non-basic variables
$B:=A_\beta$	basis
$R:=A_\gamma$	non-basis (columns)

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

Simplex Method
– Basic Feasible Solution

$$Ax = b \longrightarrow Bx_\beta + Rx_\gamma = b$$

$$\downarrow$$

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

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 : $c^T x = c_\beta^T x_\beta + c_\gamma^T x_\gamma$

$$= \underbrace{c_\beta^T B^{-1} b}_{\text{constant}} + \underbrace{\left(c_\gamma^T - c_\beta^T B^{-1} R\right)}_{\text{Reduced Costs}} x_\gamma$$

$$= c_0 + d^T x_\gamma$$

Optimality condition : $d_j \geq 0$ 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
$c_0 + d^T x_\gamma$

non-basic variables	basic variables
$x_{m+1} = 0$	$x_1 \uparrow$
$\vdots$	$\vdots$
$x_q \uparrow$	$x_p \rightarrow$
$\vdots$	$\vdots$
$x_n = 0$	$x_n \uparrow$

basic variables remain feasible (≥ 0) even if $x_q \rightarrow \infty$

$\downarrow$

objective value is unbounded

Simplex Method
– Improving Basic Feasible Solution

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

Objective value
$c_0 + d^T x_\gamma$

non-basic variables

$x_{m+1} = 0$

$\vdots$

$x_q \uparrow$

$\vdots$

$x_n = 0$

basic variables

$x_1 \uparrow$

$\vdots$

$x_p \rightarrow$

$\vdots$

$x_n \uparrow$

Unbounded solution

basic variables

$x_1 \uparrow$

$\vdots$

$x_p \downarrow$ goes to $\mathbf{0}$ first (outgoing variable)

$\vdots$

$x_n \uparrow$

neighboring improving basis

Simplex Method
– Basis Updating

Neighboring bases :

$$B = \left[b_1, \ldots, b_p, \ldots, b_m \right]$$

$$\downarrow$$

$$\bar{B} = \left[b_1, \ldots, a, \ldots, b_m \right]$$

Simplex Method
– Basis Updating

$$B = \left[b_1, \ldots, b_p, \ldots, b_m \right]$$
$$\downarrow$$
$$\bar{B} = \left[b_1, \ldots, a, \ldots, b_m \right]$$

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

(as the linear combination of the bases)

Simplex Method
– Basis Updating

$$B = \left[b_1, \ldots, b_p, \ldots, b_m \right]$$
$$\downarrow$$
$$\bar{B} = \left[b_1, \ldots, a, \ldots, b_m \right]$$

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

$$\downarrow$$

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

Simplex Method
– Basis Updating

$$B = [b_1, \ldots, b_p, \ldots, b_m]$$
$$\downarrow$$
$$\bar{B} = [b_1, \ldots, a, \ldots, b_m]$$

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

$$\downarrow$$

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

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

v^p : pivot element

Simplex Method
– Basis Updating

$$B = [b_1, \ldots, b_p, \ldots, b_m]$$
$$\downarrow$$
$$\bar{B} = [b_1, \ldots, a, \ldots, b_m]$$

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

$$\downarrow$$

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

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

$$\downarrow$$

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

(elementary transformation matrix)

Simplex Method
– Basis Updating

$$B = [b_1, \ldots, b_p, \ldots, b_m]$$
$$\downarrow$$
$$\bar{B} = [b_1, \ldots, a, \ldots, b_m]$$

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

$$\downarrow$$

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

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

$$\downarrow$$

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

$$\downarrow$$

$$\bar{B}^{-1} = EB^{-1}$$

Simplex Method
– Basis Updating

$$E = \left[e_1, \ldots, e_{p-1}, \eta, e_{p+1}, \ldots, e_m \right]$$

$$= \begin{bmatrix} 1 & & \eta^1 & & \\ & \ddots & & & \\ & & \eta^p & & \\ & & & \ddots & \\ & & \eta^m & & 1 \end{bmatrix}$$

ETM

(Elementary Transformation Matrix)

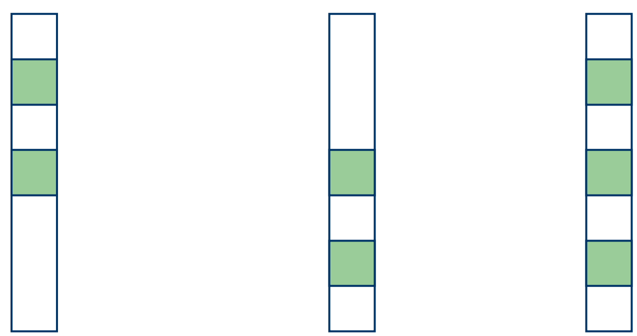
$$= \begin{bmatrix} 1 & & 0 & & \\ & \ddots & \vdots & & \\ & & \eta^p & & \\ & & \vdots & \ddots & \\ & & 0 & & 1 \end{bmatrix} \begin{bmatrix} 1 & & \eta^1 & & \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ & & 0 & & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & & 0 & & \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ & & \eta^m & & 1 \end{bmatrix}$$

Simplex Method
– Basis Updating

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

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

$Ew = w$ if $w^p = 0$



Simplex Method
– Algorithm

Steps	Major ops
1. Find an initial feasible basis B	
2. Initialization	$B^{-1}b$
3. Check optimality	$c_\beta^T B^{-1}$
4. Choose incoming variable x_q	$B^{-1}a_q$
5. Choose outgoing variable x_p	
6. Update basis	EB^{-1}

Simplex Method – Algorithm

Steps	Major ops
1. Find an initial feasible basis B	
2. Initialization	$B^{-1}b$
3. Check optimality	$c_\beta^T B^{-1}$
4. Choose incoming variable x_q	$B^{-1}a_q$
5. Choose outgoing variable x_p (pivot step)	
6. Update basis	EB^{-1}

Simplex Method
– Algorithm

Choice of pivot (numerical considerations)

- resulting less fill-ins
- large pivot element

Conflicting goals sometimes

In practice, compromise.

Simplex Method
– Typical Operations in Simplex Method

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
  $(B = \overline{B}E, \overline{B}^{-1} = EB^{-1})$

  Side effect : more LU factors

- Refactorization
  (reinstate efficiency and numerical accuracy)

Sparse LU Updates in Simplex Method

Hamid R. Ghaffari

April 10, 2007

Outline

Revised Simplex Algorithm

Simplex Method	Revised Simplex Method
Determine the current basis, d	$d = B^{-1}b$
Choose x_q to enter the basis based on the greatest cost contribution	$\bar{c} = c'_N - c'_B B^{-1} N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$
If x_q cannot decrease the cost, d is optimal solution	$\bar{c}_q \geq 0$, d is optimal solution
Determine x_p that leaves the basis (become zero) as x_q increases.	$w = B^{-1}A_q$, $\left\{p \,\middle|\, \frac{d_i}{w_i} = \min_t \left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$
If x_q can increase without causing another variable to leave the basis, the solution is unbounded	If $w_p \leq 0$ for all i, the solution is unbounded.
Update dictionary.	Update B^{-1}

Note: In general we do not compute the inverse.

Revised Simplex Algorithm

Simplex Method	Revised Simplex Method
Determine the current basis, d	$d = B^{-1}b$
Choose x_q to enter the basis based on the greatest cost contribution	$\bar{c} = c_N' - \underbrace{c_B' B^{-1}} N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$ BTRAN (backward transformation)
If x_q cannot decrease the cost, d is optimal solution	$\bar{c}_q \geq 0$, d is optim
Determine x_p that leaves the basis (become zero) as x_q increases.	$w = B^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$
If x_q can increase without causing another variable to leave the basis, the solution is unbounded	If $w_p \leq 0$ for all i, the solution is unbounded.
Update dictionary.	Update B^{-1}

Note: In general we do not compute the inverse.

Revised Simplex Algorithm

Simplex Method	Revised Simplex Method
Determine the current basis, d	$d = B^{-1}b$
Choose x_q to enter the basis based on the greatest cost contribution	$\bar{c} = c_N' - \underbrace{c_B' B^{-1}} N$, $\{q \mid \bar{c}_q = \min_t (\bar{c}_t)\}$
If x_q cannot decrease the cost, d is optimal solution	$\bar{c}_q \geq 0$, d is optim
Determine x_p that leaves the basis (become zero) as x_q increases.	$w = \underbrace{B^{-1}A_q}$, $\left\{p \,\middle|\, \frac{d_i}{w_i} = \min_t \left(\frac{d_t}{w_t}\right), \ w_t > 0\right\}$
If x_q can increase without causing another variable to leave the basis, the solution is unbounded	If $w_p \leq 0$ for all i, the solution is unbounded.
Update dictionary.	Update B^{-}

BTRAN (backward transformation)

FTRAN (forward transformation)

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 take $m^2(m-1)$ 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

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

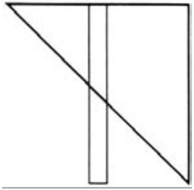
Introducing Spike

- 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

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



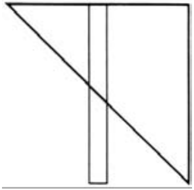
Introducing Spike

- 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

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



- How to deal with this?

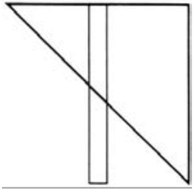
Introducing Spike

- 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

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



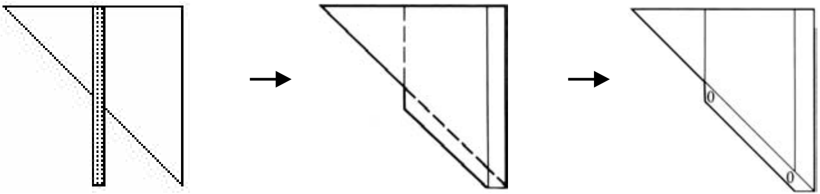
- How to deal with this?

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)

Bartels-Golub Method

Illustration

The first variant of the Revised Simplex Method was the Bartels-Golub Method.



Bartels-Golub Method

Algorithm

Revised Simplex Method	Bartels-Golub
$d = B^{-1}b$	$d = U^{-1}L^{-1}b$
$\bar{c} = c'_N - c'_B B^{-1}N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$	$\bar{c} = c'_N - c'_B U^{-1}L^{-1}N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$
$\bar{c}_q \geq 0$, d is optimal solution	$\bar{c}_q \geq 0$, d is optimal solution
$w = B^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t \left(\frac{d_t}{w_t}\right), \; w_t \; > 0\right\}$	$w = U^{-1}L^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t \left(\frac{d_t}{w_t}\right), \; w_t \; > 0\right\}$
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 B^{-1}	Update U^{-1} and L^{-1}

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.

Sparse Bartels-Golub Method

eta matrices

First take a look at the following facts:

Column-Eta factorization of triangular matrices:

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

Sparse Bartels-Golub Method

eta matrices

First take a look at the following facts:

Column-Eta factorization of triangular matrices:

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

Single-Entry-Eta Decomposition:

$$\begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & & 1 & \\ l_{41} & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & 1 & & \\ l_{31} & & 1 & \\ & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ l_{41} & & & 1 \end{bmatrix}$$

Sparse Bartels-Golub Method

eta matrices

First take a look at the following facts:

Column-Eta factorization of triangular matrices:

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

Single-Entry-Eta Decomposition:

$$\begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ l_{31} & & 1 & \\ l_{41} & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & 1 & & \\ l_{31} & & 1 & \\ & & & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ l_{41} & & & 1 \end{bmatrix}$$

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

Bartels-Golub Method	Sparse Bartels-Golub Method
$d = U^{-1}L^{-1}b$	$d = U^{-1}\prod_t \eta_t b$
$\bar{c} = c'_N - c'_B U^{-1}L^{-1}N$, $\{q \vert \bar{c}_q = \min_t(\bar{c}_t)\}$	$\bar{c} = c'_N - c'_B U^{-1}\prod_t \eta_t N$, $\{q \vert \bar{c}_q = \min_t(\bar{c}_t)\}$
$\bar{c}_q \geq 0$, d is optimal solution	$\bar{c}_q \geq 0$, d is optimal solution
$w = U^{-1}L^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$	$w = U^{-1}\prod_t \eta_t A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$
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

Algorithm

Bartels-Golub Method	Sparse Bartels-Golub Method
$d = U^{-1}L^{-1}b$	$d = U^{-1}\prod_t \eta_t b$
$\bar{c} = c_N' - c_B' U^{-1} L^{-1} N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$	$\bar{c} = c_N' - c_B' \underbrace{U^{-1} \prod_t \eta_t N}_{\text{Sparse Matrix-Vector Multiplication}}$, $\min_t(\bar{c}_t)\}$
$\bar{c}_q \geq 0$, d is optimal solution	d is optimal solution
$w = U^{-1}L^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$	$w = U^{-1}\prod_t \eta_t A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$
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

Algorithm

Bartels-Golub Method	Sparse Bartels-Golub Method
$d = U^{-1}L^{-1}b$	$d = U^{-1}\prod_t \eta_t b$
$\bar{c} = c'_N - c'_B U^{-1}L^{-1}N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$	$\bar{c} = c'_N - c'_B U^{-1}\prod_t \underbrace{\eta_t N}$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$
$\bar{c}_q \geq 0$, d is optimal solution	Sparse Matrix-Vector Multiplication imal solution
$w = U^{-1}L^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$	A_q, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$
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

Algorithm

Bartels-Golub Method	Sparse Bartels-Golub Method
$d = U^{-1}L^{-1}b$	$d = U^{-1}\prod_t \eta_t b$
$\bar{c} = c'_N - c'_B U^{-1}L^{-1}N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$	$\bar{c} = c'_N - c'_B U^{-1}\prod_t \eta_t N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$
$\bar{c}_q \geq 0$, d is optimal solution	$\bar{c}_q \geq 0$, d is optimal solution
$w = U^{-1}L^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$	$w = U^{-1}\underbrace{\prod_t \eta_t A_q}_{\text{Sparse Matrix-Vector Multiplication}}$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$
If $w_p \leq 0$ for all i, the solution unbounded.	0 for all i, the solution is ed.
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.

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

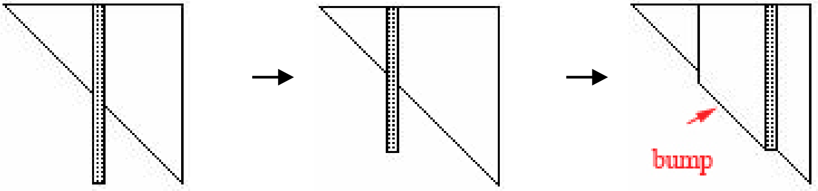
- Eventually, the number of eta matrices will become so large that it becomes cheaper to decompose the basis.
- Such a refactorization may occur prematurely in an attempt to promote stability if noticeable round-off errors begin to occur.
- In practice, in solving large sparse problem, the basis is refactorized 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

	1	2	3	4	5	6	7	8	9	10	11
1	X			X		X					X
2			X		X			X		X	
3		X	X	X					X		X
4				X	X		X				X
5		X			X					X	
6		X				X		X			
7		X					X				
8								X		X	X
9		X							X		X
10										X	
11											X

$\rightarrow$

	1	6	2	3	4	5	7	8	9	10	11
1	X	X			X						X
6		X	X					X			
2				X		X		X		X	
3			X	X	X				X		X
4					X	X	X				X
5			X			X				X	
7			X				X				
8								X		X	X
9			X						X		X
10										X	
11											X

Reid's Method

Row Rotation

	1	6	2	3	4	5	7	8	9	10	11
1	X	X			X						X
6		X	X					X			
2				X		X		X		X	
3			X	X	X				X		X
4					X	X	X				X
5			X			X				X	
7			X				X				
8								X		X	X
9			X						X		X
10										X	
11											X

→

	1	6	2	3	4	5	7	9	8	10	11
1	X	X			X						X
6		X	X						X		
2				X		X			X	X	
3			X	X	X			X			X
4					X	X	X				X
5			X			X					
7			X				X				
9			X					X			X
8									X	X	X
10										X	
11											X

Reid's Method

Characteristics

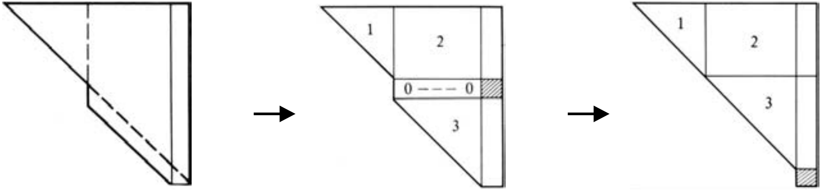
Advantages:

- 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

Bartels-Golub Method	Forrest-Tomlin Method
$d = U^{-1}L^{-1}b$	$d = U^{-1}\prod_t R_t L^{-1}b$
$\bar{c} = c'_N - c'_B U^{-1}L^{-1}N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$	$\bar{c} = c'_N - c'_B U^{-1}\prod_t R_t L^{-1}N$, $\{q \mid \bar{c}_q = \min_t(\bar{c}_t)\}$
$\bar{c}_q \geq 0$, d is optimal solution	$\bar{c}_q \geq 0$, d is optimal solution
$w = U^{-1}L^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$	$w = U^{-1}\prod_t R_t L^{-1}A_q$, $\left\{p \,\middle\vert\, \frac{d_i}{w_i} = \min_t\left(\frac{d_t}{w_t}\right),\ w_t > 0\right\}$
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} creating a row factor as necessary. If there are too many factors, completely refactor the basis.

The Forrest-Tomlin Method Characteristics

Advantages:

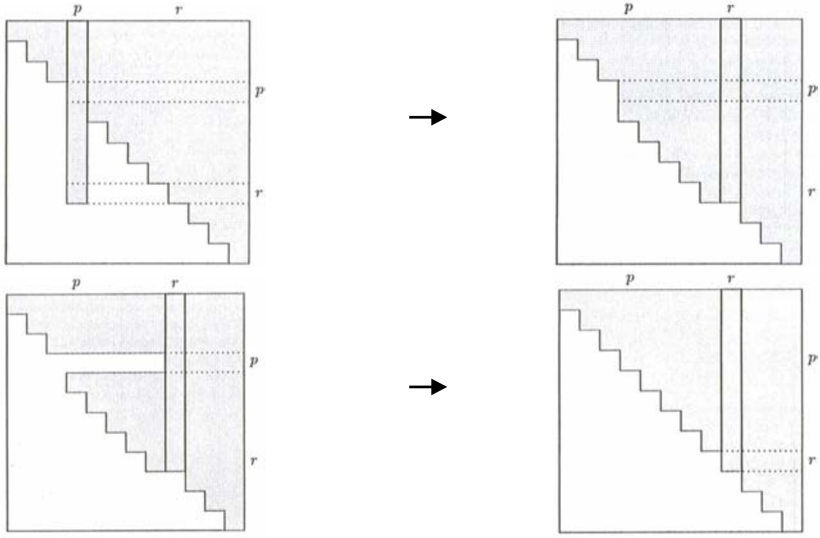
- At most one row-eta matrix factor will occur for each iteration where an unpredictable number occurred before.
- The code can take advantage of such knowledge for predicting necessary storage space and calculations.
- Fill-in should also be relatively slow, since fill-in can only occur within the spiked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



For More Detail

Leena M. Suhl, Uwe H. Suhl
A fast LU update for linear programming.
Annals of Operations Research 43(1993)33-47

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

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

Thanks