# *L-Implicit-U* Factorization and the Simplex Update

Sparse Matrices for Scientific Computation:
In Honour of John Reid's 70th Birthday

Roger Fletcher

July 15th, 2009

# Overview

- Solving $A\mathbf{x} = \mathbf{b}$ and $A^T\mathbf{x} = \mathbf{b}$ for nonsingular $A$ by means of *L-Implicit-U* (LIU) factors
- The Simplex Update is the replacement of one column of $A$ by a new column. It is important that LIU factors can be updated quickly.
- Issues concerned with LIU factors for large sparse systems

# What are LIU factors?

*L-Implicit-U* (LIU) factors of nonsingular $A$ comprise a lower triangular matrix $\mathbb{L}$ and a diagonal matrix $D$ such that

$$\mathbb{L}PA = U$$

where $D = \mathrm{diag}(U)$ and $P$ is a permutation matrix allowing row pivoting. When using the factors, access is required to $A$, but not to the off-diagonals of $U$, which therefore are not stored.

LIU factors are related to regular LU factors

$$PA = LU$$

by $\mathbb{L} = L^{-1}$, and $U$ is the same upper triangular matrix.

# Are LIU factors practicable?

For dense, non-symmetric $A$, with row pivoting, operation counts and error analysis results are the same as for regular LU, both for forming the factors, and for using them to solve systems.

When the Fletcher-Matthews method is used to update the factors after a Simplex update, LIU factors are significantly more efficient, since there is no $U$ to update.

Given that $A$ must be stored (as in LP), only the diagonal of $U$ is needed, resulting in a significant saving in the storage requirement.

(see R. Fletcher, "Dense Factors of Sparse Matrices", in Approximation Theory and Optimization: Tributes to M.J.D.Powell, Eds. M.D.Buhmann and A.Iserles, C.U.P., 1997.)

# Solving Systems using LIU factors

The methodology here is quite different from the regular LU case.

Assume row permutations have been subsumed into $A$.

To solve $A\mathbf{x} = \mathbf{b}$ use

$$
\begin{aligned}
&\mathbf{b}^{(n)} = \mathbf{b} \\
&\texttt{for} \quad i = n, n-1, \ldots, 1 \\
&\qquad x_i = \mathbf{l}_i^T \mathbf{b}^{(i)} / d_i \\
&\qquad \mathbf{b}^{(i-1)} = \mathbf{b}^{(i)} - \mathbf{a}_i x_i \\
&\texttt{end}
\end{aligned}
$$

and to solve $A^T\mathbf{x} = \mathbf{b}$ use

$$
\begin{aligned}
&\mathbf{x}^{(0)} = \mathbf{0} \\
&\texttt{for} \quad i = 1, 2, \ldots, n \\
&\qquad y_i = (b_i - \mathbf{a}_i^T \mathbf{x}^{(i-1)}) / d_i \\
&\qquad \mathbf{x}^{(i)} = \mathbf{x}^{(i-1)} + \mathbf{l}_i y_i \\
&\texttt{end} \\
&\mathbf{x} = \mathbf{x}^{(n)}.
\end{aligned}
$$

where $\mathbf{l}_i^T$ denote the rows of $\mathbb{L}$ and $\mathbf{a}_i$ the columns of $A$.

Note just saxpy and scalar product operations are performed on these vectors.

# So why are LIU factors now almost unknown in NA?

many early papers discussed what are essentially LIU factors (or (Implicit-L)-U factors), including some very famous names, but its all lost in the mists of time

...

| | |
|---|---|
| Fox, Huskey and Wilkinson | 1948 |
| Hestenes and Stiefel | 1952 |
| Purcell | 1953 |
| Householder | 1955 |
| Pietrzykowski | 1960 |
| Faddeev and Faddeeva | 1963 |
| Stewart | 1973 |
| Enderson and Wassyng | 1978 |
| Sloboda | 1978 |
| Wassyng | 1982 |
| Abaffy, Broyden and Spedicato | 1984 |
| Hegedüs | 1986 |
| Tuma | 1993 |
| Benzi and Meyer | 1994,1995 |

# LIU factors for a sparse matrix

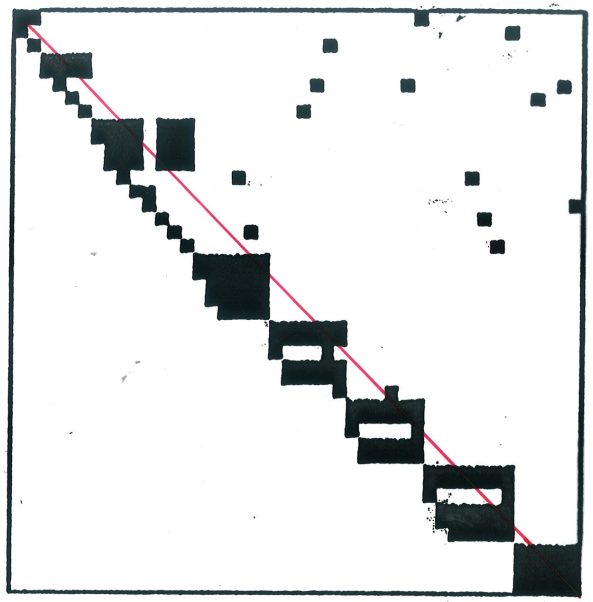
The main idea is to make row and column permutations, $PAQ$ say, to concentrate nonzeros into $U$, as far as possible.

1. Permute unit columns to the left, that is
$$A = \begin{bmatrix} I & A_1 \\ 0 & A_2 \end{bmatrix}.$$
2. Reduce $A_2$ to irreducible block upper triangular (traversal/Tarjan's algorithm)
3. Transform each irreducible block to upper triangular + a few row spikes in the lower triangle

# Processing an irreducible block (1)

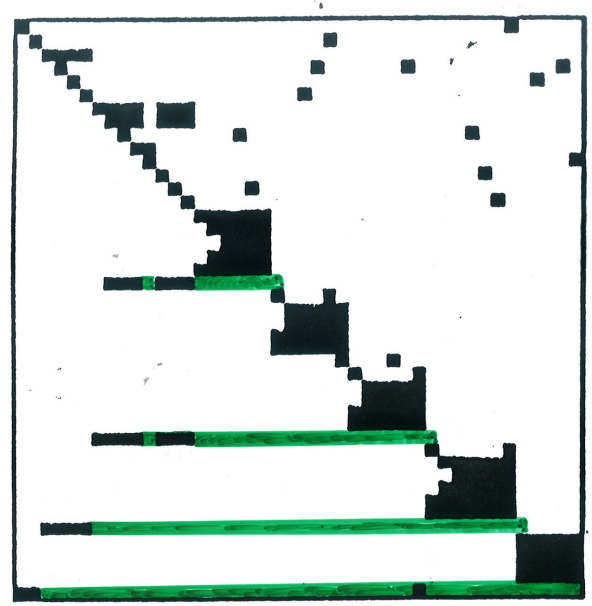
Ideally, using further row and column permutations, we would proceed in two stages

Upper block rectangular



(SPK1 algorithm)

Spiked upper triangular



(spike stack algorithm)

Now, when factorizing $PAQ$, $\mathbb{L}$ only fills in where there are row spikes.

# Processing an irreducible block (2)

Unfortunately the ideal case is only realised numerically if stable pivots can be found from the rectangular blocks in the SPK1 ordering. Hence we need to employ threshold pivoting when forming the spikes, and this can increase the number and/or total length of spikes, albeit usually not substantially

The representation of $\mathbb{L}$ as a collection of dense row spikes is very efficient for implementing the algorithms for $A\mathbf{x} = \mathbf{b}$ and $A^T\mathbf{x} = \mathbf{b}$, involving saxpy and scalar product operations between the relatively few row spikes and the sparse columns of $A$.

Also don't need to remember the Tarjan block structure or the decomposition into $A_1$ and $A_2$.

# The Simplex Update for sparse matrices

Huge variety of methods in the literature.

My QP code uses the Fletcher-Matthews method as it applies to LIU factors, represented in spiked form. However, growth in the total spike length can be quite rapid, and it becomes necessary to reinvert, typically every 30 iterations.

In this talk I describe how ideas inherent in Schur complement updates (SCU) (Bisschop and Meeraus) can be used to advantage in the context of LIU factors.

The motivation is to make use of factors of an initial matrix, $A^{\circ}$ say, over a number of Simplex updates

# The Augmented Matrix in SCU (1)

A key observation of SCU is that if column $\mathbf{a}_p$ of $A^\circ$ is replaced by a new column $\mathbf{a}_q$, then solves with the updated matrix are readily effected from factors of the augmented matrix

$$\begin{bmatrix} A^\circ & \mathbf{a}_q \\ \mathbf{e}_p^T & 0 \end{bmatrix}.$$

But it follows from

$$\begin{bmatrix} I\!L & \\ \boldsymbol{\lambda}^T & 1 \end{bmatrix} \begin{bmatrix} A^\circ & \mathbf{a}_q \\ \mathbf{e}_p^T & 0 \end{bmatrix} = \begin{bmatrix} U & \mathbf{u} \\ & \mu \end{bmatrix}$$

that LIU factors of the augmented matrix are simply obtained by adding the single spike $\boldsymbol{\lambda}$ to the spike set of $A^\circ$, where $\boldsymbol{\lambda}$ is obtained by solving $A^{\circ T}\boldsymbol{\lambda} = -\mathbf{e}_p$, and setting $\mu = \boldsymbol{\lambda}^T\mathbf{a}_q$.

# The Augmented Matrix in SCU (2)

If $\mathbf{a}_p$ is a unit vector, it is first moved to the right of the set of unit columns, so as to keep the length of $\boldsymbol{\lambda}$ as short as possible.

This process can be repeated as further Simplex updates are made ($k$ say), leading to an augmented matrix of the form

$$\begin{bmatrix} A \\ E \end{bmatrix}$$

where $A$ is a rectangular matrix containing $A^\circ$ and all the columns $\mathbf{a}_q$ that have been added, and $E$ is a matrix of $k$ unit rows, whose unit elements mark the columns of $A$ that have been deleted.

One new spike in $\mathbb{L}$ is added for each Simplex update, by means of a solve with the factors of the previous augmented matrix.

Eventually reinversion may be needed if the total spike length is too large.

# Adding a previously deleted column (1)

It is convenient if the extended matrix $A$ in the above contains at most one copy of each column $\mathbf{a}_j$ in the LP problem.

Thus if a previously deleted column is added, we update the augmented matrix by deleting the corresponding row of $E$. In this case, updating the LIU factors of the augmented matrix is less straightforward.

Here is a sketch of how to proceed ...

# Adding a previously deleted column (2)

- Let the last few spikes of $\mathbb{L}$ be as shown, with the top row corresponding to the row of $E$ being deleted

$$\begin{bmatrix} \times & \times & \times & \times & \times & 1 & & & \\ \times & \times & \times & \times & \times & \times & 1 & & \\ \times & \times & \times & \times & \times & \times & \times & 1 & \\ \times & \times & \times & \times & \times & \times & \times & \times & 1 \end{bmatrix}$$

# Adding a previously deleted column (2)

- Let the last few spikes of $\mathbb{L}$ be as shown, with the top row corresponding to the row of $E$ being deleted

$$\begin{bmatrix} \times & \times & \times & \times & \times & 1 & & & \\ \times & \times & \times & \times & \times & \times & 1 & & \\ \times & \times & \times & \times & \times & \times & \times & 1 & \\ \times & \times & \times & \times & \times & \times & \times & \times & 1 \end{bmatrix}$$

- Working upwards, carry out row operations to introduce zeros in the starred positions, and then delete the top row

$$\begin{bmatrix} -m_3 & 1 & & \\ & -m_2 & 1 & \\ & & -m_1 & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times & \times & 1 & & & \\ \times & \times & \times & \times & \times & * & 1 & & \\ \times & \times & \times & \times & \times & * & \times & 1 & \\ \times & \times & \times & \times & \times & * & \times & \times & 1 \end{bmatrix}$$

# Adding a previously deleted column (2)

- Let the last few spikes of $\mathbb{L}$ be as shown, with the top row corresponding to the row of $E$ being deleted

$$\begin{bmatrix} \times & \times & \times & \times & \times & 1 & & \\ \times & \times & \times & \times & \times & \times & 1 & \\ \times & \times & \times & \times & \times & \times & \times & 1 \\ \times & \times & \times & \times & \times & \times & \times & \times & 1 \end{bmatrix}$$

- Working upwards, carry out row operations to introduce zeros in the starred positions, and then delete the top row

$$\begin{bmatrix} -m_3 & 1 & & \\ & -m_2 & 1 & \\ & & -m_1 & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times & \times & 1 & & \\ \times & \times & \times & \times & \times & * & 1 & \\ \times & \times & \times & \times & \times & * & \times & 1 \\ \times & \times & \times & \times & \times & * & \times & \times & 1 \end{bmatrix}$$

- Finally close up the columns of $\mathbb{L}$ and remove a row of $E$

$$\begin{bmatrix} \times & \times & \times & \times & \times & 0 & 1 & \\ \times & \times & \times & \times & \times & 0 & \times & 1 \\ \times & \times & \times & \times & \times & 0 & \times & \times & 1 \end{bmatrix} \begin{bmatrix} A \\ E \end{bmatrix}$$

# Adding a previously deleted column (3)

However a near-zero pivot may cause numerical instability

To avoid this we use the Fletcher-Matthews idea, based on allowing a row interchange in $E$, as illustrated in the following example (only the trailing submatrix of $\mathbb{L}$ is shown).

For the first step we use the basic row elimination operation

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & {}^{1}/_{6} & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ {}^{-1}/_{8} & 1 & & \\ {}^{3}/_{4} & {}^{-10}/_{3} & 1 & \\ {}^{-1}/_{8} & {}^{-1}/_{9} & {}^{-1}/_{2} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ {}^{-1}/_{8} & 1 & & \\ {}^{3}/_{4} & {}^{-10}/_{3} & 1 & \\ 0 & {}^{-2}/_{3} & {}^{-1}/_{3} & 1 \end{bmatrix}$$

For the next step we shall use a Fletcher-Matthews type step ...

# A Fletcher-Matthews type step

First we create a unit $2 \times 2$ block in $\mathbb{L}$

$$\begin{bmatrix} 1 & & \\ & 1 & \\ & {}^{10}/_{3} & 1 \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ {}^{-1}/_{8} & 1 & & \\ {}^{3}/_{4} & {}^{-10}/_{3} & 1 & \\ 0 & {}^{-2}/_{3} & {}^{-1}/_{3} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ {}^{-1}/_{8} & 1 & 0 & \\ {}^{1}/_{3} & 0 & 1 & \\ 0 & {}^{-2}/_{3} & {}^{-1}/_{3} & 1 \end{bmatrix}$$

Next, interchange rows and eliminate: this can be expressed as

$$\begin{bmatrix} 1 & & & \\ & 0 & 1 & \\ & 1 & {}^{3}/_{8} & \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ {}^{-1}/_{8} & 1 & 0 & \\ {}^{1}/_{3} & 0 & 1 & \\ 0 & {}^{-2}/_{3} & {}^{-1}/_{3} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ {}^{1}/_{3} & 0 & 1 & \\ 0 & 1 & {}^{3}/_{8} & \\ 0 & {}^{-2}/_{3} & {}^{-1}/_{3} & 1 \end{bmatrix}$$

Finally, interchanging columns in $\mathbb{L}$ and rows in $E$ returns $\mathbb{L}$ to lower triangular

# Adding a previously deleted column (4)

A basic elimination step now completes the elimination process

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

Either the basic step or the Fletcher-Matthews step can be made in a numerically stable manner at each stage.

The choice is usually made on the basis of minimizing a bound on growth in the factors.

# Conclusions

The spike based factorization scheme has been working well for many years.

The augmented matrix approach for Simplex updates is currently being coded.

Hopefully some numerical results will be available soon

Finally: Happy 70th Birthday John!!