Go to full-screen mode now by hitting CTRL-L

One of the most efficient (and popular) sequential algorithms for solving:

$$Ax = b$$

relies on LU decomposition of A: $A = LU$.

If A is a nonsymmetrical, positively defined matrix of order n, A is decomposed into a product of two matrices L and U where L is lowe triangular matrix with 1 on the diagonal and U is upper triangular matrix.

LU

LU decomposition

- is a matrix decomposition which writes a matrix as the product of a lower triangular matrix and an upper triangular matrix
- is used to solve systems of linear equations or calculate the determinant
- exists if and only if all matrix's leading principal minors are non-zero

LU

LU decomposition

- is unique if we require that the diagonal of L (or U) consist of ones
- has few versions: Dollitle decomposition, Cholesky decomposition

LU

$$L = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 & 0 \\ l_{21} & 1 & \dots & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ l_{k,1} & l_{k,2} & \dots & 1 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ l_{n-1,1} & l_{n-1,2} & \dots & l_{n-1,k} & \dots & 1 & 0 \\ l_{n,1} & l_{n,2} & \dots & l_{n,k} & \dots & l_{n,n-1} & 1 \end{pmatrix}$$

LU

$$U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1,k} & \dots & u_{1,n-1} & u_{1,n} \\ 0 & u_{22} & \dots & u_{2,k} & \dots & u_{2,n-1} & u_{2,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & u_{k,k} & \dots & u_{k,n-1} & u_{k,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & \dots & u_{n-1,n-1} & u_{n-1,n} \\ 0 & 0 & \dots & 0 & \dots & 0 & u_{n,n} \end{pmatrix}$$

LU

The algorithm then proceeds to solve y from $Ly = b$ and then finds x by solving $Ux = y$.

$$Ly = b$$

$$Ux = y$$

LU

Then the elements of L and U can be calculated from the following dependencies:
for $s < t$:

$$u_{s,t} = a_{s,t} - \sum_{j=1}^{s} l_{sj} u_{jt}$$

for $s > t$:

$$l_{s,t} = \frac{(a_{s,t} - \sum_{j=1}^{t} l_{sj} u_{jt})}{u_{tt}}$$

We assume that $l_{tt} = 1$

LU

Order of action in the sequential calculation:

1. $l_{21}, l_{31}, ..., l_{n1}$ - $1-st$ row of L
2. $u_{11}, u_{12}, ..., u_{1n}$ - $1-st$ column of U
3. $u_{22}, u_{23}, ..., u_{2n}$ - $2-nd$ row of L
4. $u_{32}, u_{42}, ..., u_{n2}$ - $2-nd$ of column U
5. $l_{33}, l_{34}, ..., l_{3n}$ - $3-rd$ row of L

etc

LU by BLAS

The following BLAS procedures can be used to implement the LU decomposition:

- Reciprocal Scale x=x/alpha : RSCALE
- Scaled vector accumulation y=alpha*x+beta*y : AXPBY
- Rank one updates A=alpha*x*trans(y) +beta*A : SGER, DGER

Parallel LU

Processes are put on the two-dimensional grid.

$$\begin{matrix} P_{0,0} & P_{0,1} & P_{0,2} & & P_{0,n-1} & P_{0,n} \\ P_{1,0} & P_{1,1} & P_{1,2} & & P_{1,n-1} & P_{1,n} \end{matrix}$$

$$\begin{matrix} P_{n-2,0} & P_{n-2,1} & P_{n-2,2} & & P_{n-2,n-1} & P_{n-2,n} \\ P_{n-1,0} & P_{n-1,1} & P_{n-1,2} & & P_{n-1,n-1} & P_{n-1,n} \end{matrix}$$

Parallel LU

Each process P_{st} calculates one element of L or U . If $s \leq t$ then it is U, if it $s > t$ is L.

$$\begin{matrix} u_{0,0} & u_{0,1} & u_{0,2} & & u_{0,n-1} & u_{0,n} \\ l_{1,0} & u_{1,1} & u_{1,2} & & u_{1,n-1} & u_{1,n} \end{matrix}$$

$$\begin{matrix} l_{n-2,0} & l_{n-2,1} & l_{n-2,2-1} & & u_{n-2,n-2} & u_{n-2,n-1} \\ l_{n-1,0} & l_{n-1,1} & l_{n-1,2} & & l_{n-1,n-1} & u_{n-1,n-1} \end{matrix}$$

Parallel LU

Each process P_{st} waits for it's pre-processes. for P_{st} where $s < t$ (matrix U) there are: $P_{s1}, P_{s2}, ...P_{s,s-1}$ and $P_{1t}, P_{2t}, ...P_{s-1,t}$.

$$\begin{array}{cccccccccc}
u_{11} & * & \dots & * & * & \dots & u_{1t} & \dots & * & * \\
* & u_{22} & \dots & * & * & \dots & u_{2t} & \dots & * & * \\
* & * & \dots & * & * & \dots & & \dots & * & * \\
* & * & \dots & u_{s-1,s-1} & * & \dots & u_{s-1,t} & \dots & * & * \\
l_{s1} & l_{s2} & \dots & l_{s,s-1} & u_{ss} & \dots & u_{st} & \dots & * & * \\
* & * & \dots & * & * & \dots & & \dots & * & * \\
* & * & \dots & * & * & \dots & & \dots & * & *
\end{array}$$

Parallel LU

For P_{st} where $s > t$ (matrix L) there are: $P_{s1}, P_{s2}, \ldots P_{s,t-1}$ and $P_{1t}, P_{2t}, \ldots P_{t,t}$.

$$\begin{array}{ccccccccccc}
u_{11} & * & \ldots & * & u_{1t} & \ldots & \ldots & * & \ldots & * \\
* & u_{22} & \ldots & * & u_{2t} & \ldots & \ldots & * & \ldots & * \\
* & * & \ldots & * & * & \ldots & \ldots & * & \ldots & * \\
* & * & \ldots & * & u_{t,t} & \ldots & \ldots & * & \ldots & * \\
* & * & \ldots & * & * & \ldots & \ldots & * & \ldots & * \\
* & * & \ldots & * & l_{s-1,t} & \ldots & u_{s-1,s-1} & * & \ldots & * \\
l_{s1} & l_{s2} & \ldots & l_{s,t-1} & l_{st} & \ldots & l_{s,s-1} & u_{ss} & \ldots & * \\
* & * & \ldots & * & * & \ldots & & * & \ldots & *
\end{array}$$

Project is co-financed by European Union within European Social Fund

Parallel LU

When the message is comming, the next step in calculation is done - the next element of the sum or the division by the diagonal element is calculated.

Processes P_{st} calculating U, after termination of their calculation, send the results to all processes down:$P_{s+1,t}, P_{s+2,t}, \ldots P_{n,t}$.

Processes P_{st} calculating L , after termination of their calculation, send the results to all processes on the right $P_{s,t+1}, P_{s,t+2}, \ldots P_{s,n}$.

Parallel LU

REFERENCES: J.G.G. van de Vorst, 'The Formal Development of a Parallel Program Performing LU-Decomposition', Acta Informatica 26 ,1988, 1-17.

Let L be a lower matrix:
$l_{st} = 0$ for $s < t$ and $l_{st} = 1$ for $s = t$

and let U be an upper matrix:
$u_{st} = 0$ for $s > t$.

Parallel LU

We want to find L i U, such that $A = LU$.

Let:

$$X = (x_{st}, 0 =< s, t < n)$$

where:

$x_{st} = u_{st}$ for $s \leq t$,

$x_{st} = l_{st}$ for $s > t$.

Parallel LU

We have:

for $s =< t$

$$a_{st} = \sum_{j=1}^{n} l_{sj} u_{jt} = u_{st} + \sum_{j=1}^{s-1} l_{sj} u_{jt} = x_{st} + \sum_{j=1}^{s-1} x_{sj} x_{jt}$$

for $s > t$

$$a_{st} = \sum_{j=1}^{n} l_{sj} u_{jt} = l_{st} u_{tt} + \sum_{j=1}^{s-1} l_{sj} u_{jt} = x_{st} x_{tt} + \sum_{j=1}^{s-1} x_{sj} x_{jt}$$

Parallel LU

Elements of X we can calculate in the following way:

$$a_{st} = a_{st} + \sum_{j=1}^{s-1} x_{sj} x_{jt}$$

for $s \leq t$

$$x_{st} x_{tt} = a_{st} - \sum_{j=1}^{s-1} x_{sj} x_{jt}$$

for $s > t$

Sequential LU

For each i, j: $x_{ij} = a_{ij}$

Step 1

We calculate the first row of U:

$$x_{0t} = x_{0t}$$

Step 2

Calculate the first column of L:

$$x_{s0}x_{00} = x_{s0}$$

thus

$$x_{s0} = x_{s0}/x_{00}$$

Sequential LU

Step 3

We calculate the 2-nd row of U:

$$x_{1t} = x_{1t} - x_{10}x_{0t}$$

for $t >= 2$

Step 4

We calculate the 2-nd column of L:

$$x_{s1}x_{11} = x_{s1} - x_{s0}x_{01}$$

for $s > 1$

Sequential LU

Step 2*i+1

We calculate the i-th row of U:

$$x_{s,t} = x_{s,t} - \sum_{j=1}^{t} x_{sj} x_{jt}$$

for $s < t$

Step 2*(i+1)

We calculate the i-th column of L:

$$x_{s,t} = (a_{s,t} - \sum_{j=1}^{t} x_{sj} x_{jt}) / x_{tt}$$

for $s > t$

Parallel LU

The algorithm is written in the synchronized version.

Process $pr(s,t)$ calculates x_{st}.

Step 1

- calculate the first row of U:

$$x_{0t} = a_{0t}$$

- send x_{0t} to all $pr(s,t)$ for $s >= l$

Parallel LU

Step 2

- calculate the first column of L:

$$x_{s0} = x_{s0}/x_{00}$$

- send x_{s0} to all $pr(s,t)$ for $t >= l$

Parallel LU

Step 2'

is done by all processes $pr(s,t)$, where $s,t >= 1$

- receive x_{s0} from all pr(s,0)
- receive x_{0t} from all $pr(0,t)$
- $x_{st} = x_{st} - x_{s0}x_{0t}$

Parallel LU

Step 3

/* We should calculate the 2-nd row of U:

$$x_{1t} = x_{1t} - x_{10}x_{0t}$$

for $t >= 2$ but it has been done by Step 2' */

send x_{1t} to all $pr(s,t)$ where $s >= 2$

Parallel LU

Step 4

- calculate the 2-nd column of L:
  $x_{s1} = x_{s1}/x_{11}$ for $s > 1$
- send x_{s1} to all pr(s,t) where $t >= 2$

Step 4' is done by all processes $pr(s,t)$, where $s,t >= 2$

- receive x_{s1} from all $pr(s,l)$, $l < max(s,t)$
- receive x_{1t} from all $pr(l,t)$, $l < max(s,t)$
- $x_{st} = x_{st} - x_{s1}x_{1t}$

Parallel LU

Step 2*i+1

/* The calculation of the i-th row of U: $x_{it} = x_{it} - x_{ij}x_{jt}$ for $i =< t$ has been done by step 2i' */

send x_{it} to all $pr(s,t)$, where $s >= i$

Step 2*(i+1)

- calculate the i-th column of L:

$$x_{si} = x_{si} - x_{sj}x_{ji}$$

  for $s > i$

- send x_{si} to all $pr(s,t)$, where $s >= i$

Parallel LU

Step 2*(i+1)' is done by all processes $pr(s,t)$, where $s,t >= i$

- receive x_{si} from all $pr(s,i)$
- receive x_{it} from all $pr(i,t)$
- $x_{st} = x_{st} - x_{si}x_{it}$ for $s > i$

Each step can be done in parallel for each process $pr(s,t)$.

In fact the synchronization is not important.

Parallel LU

From above we can write the following asynchronized algorithm:

for all s,t: $0 =< s,t < n$ parallel do $process(s,t)$;

where $process(s,t)$ is working independently for each element of the matrix.
The 2D mesh of computers is the most convenient architecture for synchronized version of LU decomposition. The number of processes is equal the number of elements of the matrix. The schedule problem is solved by operating system.

Parallel LU

process(int s,int t)

{ float x_{st}; int k;

$x_{st} = a_{st}$;

k=0;

while ($k < n$)

{ if ($k < min(s,t)$)

{

receive x_{sk} from process (s,k);

receive x_{kt} from process (k,t);

$x_{st} = x_{st} - x_{sk} * x_{kt}$; };

else

Parallel LU

if ($k = t < s$)
{ receive x_{kk} from process (k,k);
$x_{st} = x_{st}/x_{kk}$;
send x_{st} to all processes (s,q) with $q > k$
};
else
if ($k = s < t$) send x_{kt} to all processes (q,t) with $q > k$;
else
if ($k > min(s,t)$) skip; k++;
}; /*while*/
}

LU in LAPACK

LAPACK is implemented for shared memory machines.

In the first stage the following vector-vector and matrix-matrix calculations are made:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1,n-1} & a_{1,n} \\ a_{21} & a_{22} & \dots & a_{2,n-1} & a_{2,n} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & a_{n,2} & \dots & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

$$= A^{(1)} + l^{(1)} {u^{(1)}}^T$$

LU in LAPACK

where $l_1^{(1)} = 1$ and $a_{ij}^{(1)} = 0$ for $i = 1$ or $j = 1$.
Assuming that $a_{11} \neq 0$ we have:

$$l^{(1)} = \begin{pmatrix} 1 \\ a_{21}/a_{11} \\ a_{31}/a_{11} \\ \dots \\ a_{n-1,1}/a_{11} \\ a_{n,1}/a_{11} \end{pmatrix} \qquad u^{(1)} = \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \\ \dots \\ a_{1,n-1} \\ a_{1,n} \end{pmatrix}$$

LU in LAPACK

$$A^{(1)} = A - l^{(1)} u^{(1)^T}$$

$$= \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & a_{22} - a_{21}a_{12}/a_{11} & \dots & a_{2,n} - a_{21}a_{1n}/a_{11} \\ \dots & \dots & \dots & \dots \\ 0 & a_{n-1,2} - a_{n-1,1}a_{12}/a_{11} & \dots & a_{n-1,n} - a_{n-1,1}a_{1n}/a_{11} \\ 0 & a_{n,2} - a_{21}a_{12}/a_{11} & \dots & a_{n,n} - a_{n1}a_{1n}/a_{11} \end{pmatrix}$$

LU in LAPACK

After $k-1$ stages of this elimination we have the partially eliminated matrix

$$A^{(k)} = A^{(k-1)} - l^{(k)} {u^{(k)}}^T$$

$$= \begin{pmatrix} 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & a_{kk}^{(k)} & \dots & a_{k,n}^{(k)} \\ \dots & \dots & \dots & \dots & & \\ 0 & \dots & 0 & a_{n-1,k}^{(k)} & \dots & a_{n-1,n}^{(k)} \\ 0 & \dots & 0 & a_{n,k}^{(k)} & \dots & a_{n,n}^{(k)} \end{pmatrix}$$

LU in LAPACK

Where

$$l_i^{(k)} = u_i^{(k)} = 0$$

for $i < k - 1$,

$$l_k^{(k)} = 1$$

and

$$a_{ij}^{(k)} = 0$$

for $i < k$ or $j < k$.

LU in LAPACK

Provided that $a_{kk}^{(k)} \neq 0$, we have:

$$l^{(k)} = \begin{pmatrix} 0 \\ \dots \\ 0 \\ 1 \\ a_{k+1,k}^{(k-1)} / a_{kk}^{(k-1)} \\ a_{31}^{(k-1)} / a_{kk}^{(k-1)} \\ \dots \\ a_{n-1,1}^{(k-1)} / a_{kk}^{(k-1)} \\ a_{n,1}^{(k-1)} / a_{kk}^{(k-1)} \end{pmatrix} \quad u^{(1)} = \begin{pmatrix} 0 \\ \dots \\ 0 \\ a_{kk}^{(k-1)} \\ a_{k,k+1}^{(k-1)} \\ a_{k,k+2}^{(k-1)} \\ \dots \\ a_{k,n-1}^{(k-1)} \\ a_{k,n}^{(k-1)} \end{pmatrix}$$

LU in LAPACK

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)} a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}$$

$$i, j = k+1, k+2, ..., n.$$

LU in LAPACK

After n stages of this elimination procedure we find that:

$$A = l^{(1)}u^{(1)^T} + A^{(1)}$$
$$= l^{(1)}u^{(1)^T} + l^{(12)}u^{(2)^T} + A^{(2)}$$
...
$$= l^{(1)}u^{(1)^T} + l^{(12)}u^{(2)^T} + ... + l^{(n)}u^{(n)^T} + A^{(n)}$$

where $A^{(n)}$ is the zero matrix.

LU in LAPACK

Hence

$$L = \begin{pmatrix} l^{(1)} & l^{(2)} & \dots & l^{(n)} \end{pmatrix}$$

$$U = \begin{pmatrix} u^{(1)^T} \\ u^{(2)^T} \\ \dots \\ u^{(n)^T} \end{pmatrix}$$

L is a unit diagonal, lower triangular, matrix , and U is upper triangular matrix.

LU in LAPACK

Example

$$\begin{pmatrix} 4 & 1 & 0 & 1 \\ -4 & 2 & 2 & -1 \\ 0 & 0 & 3 & 1 \\ -4 & -1 & -3 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 7 \\ 3 \\ 7 \\ -9 \end{pmatrix}$$

LU in LAPACK

Step 1

$$1^{(1)} = \begin{pmatrix} 1 \\ -1 \\ 0 \\ -1 \end{pmatrix} \qquad u^{(1)} = \begin{pmatrix} 4 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

.

LU in LAPACK

$$A^{(1)} = \begin{pmatrix} 4 & 1 & 0 & 1 \\ -4 & 2 & 2 & -1 \\ 0 & 0 & 3 & 1 \\ -4 & -1 & -3 & 3 \end{pmatrix} - \begin{pmatrix} 1 \\ -1 \\ 0 \\ -1 \end{pmatrix} \begin{pmatrix} 4 & 1 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & -3 & 4 \end{pmatrix}$$

LU in LAPACK

Step 2

$$l^{(2)} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \qquad u^{(2)} = \begin{pmatrix} 0 \\ 3 \\ 2 \\ 0 \end{pmatrix}$$

LU in LAPACK

$$A^{(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & -1 & -3 & 3 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \begin{pmatrix} 0 & 3 & 2 & 0 \end{pmatrix} =$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & -3 & 4 \end{pmatrix}$$

LU in LAPACK

Step3

$$l^{(3)} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix} \qquad u^{(3)} = \begin{pmatrix} 0 \\ 0 \\ 3 \\ 1 \end{pmatrix}$$

.

LU in LAPACK

$$A^{(3)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & -3 & 4 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 3 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

LU in LAPACK

We compute the vectors of multipiers:

$$l^{(4)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \qquad u^{(4)} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 5 \end{pmatrix}$$

.

LU in LAPACK

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & -1 & 1 \end{pmatrix} \qquad U = \begin{pmatrix} 4 & 1 & 0 & 1 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

LU in LAPACK

Finally:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} 4 & 1 & 0 & 1 \\ 0 & 3 & 2 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 7 \\ 3 \\ 7 \\ -9 \end{pmatrix}$$

LU in LAPACK

Parallelisation:

1. The same as for Gaussian elimination
2. The outer loop has inevitable synchronization restrictions

LAPACK Metodology

Subdivide the problem using blocked and partitioned algorithms which maximize the use of BLAS (mostly Level 3 Blas - matrix-matrix calculations).

Example
LU decomposition for 3×3 block matrix.

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

LAPACK Metodology

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix}$$

$$U = \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix}$$

LAPACK Metodology

$$A = LU =$$

$$\begin{pmatrix} L_{11}U_{11} & L_{11}U_{12} & L_{11}U_{13} \\ L_{21}U_{11} & L_{21}U_{12} + L_{22}U_{22} & L_{21}U_{13} + L_{22}U_{23} \\ L_{31}U_{12} & L_{32}U_{22} + L_{21}U_{11} & L_{31}U_{13} + L_{32}U_{23} + L_{33}U_{33} \end{pmatrix}$$

LAPACK Metodology

Algorithm

1. Find L_1 and U_{11} such that:

$$L_1 = \begin{pmatrix} L_{11} \\ L_{21} \\ L_{31} \end{pmatrix}$$

$$\begin{pmatrix} A_{11} \\ A_{21} \\ A_{31} \end{pmatrix} = \begin{pmatrix} L_{11} \\ L_{21} \\ L_{31} \end{pmatrix} (U_{11})$$

LAPACK Metodology

2. Find U_{12} from the equation:

$$A_{12} = L_{11}U_{12}$$

3. Update the blocks:

$$\begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix} \leftarrow \begin{pmatrix} L_{21} \\ L_{23} \end{pmatrix} U_{12} + \begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix}$$

LAPACK Metodology

4. Form the LU factors according to:

$$\begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix} \leftarrow \begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22}$$

5. We can find U_{13} and U_{23} by solving the system:

$$\begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{13} \\ U_{23} \end{pmatrix} = \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix}$$

LAPACK Metodology

6. Modify the diagonal block:

$$A_{33} \leftarrow \begin{pmatrix} L_{31} & L_{32} \end{pmatrix} \begin{pmatrix} U_{13} \\ U_{23} \end{pmatrix} + A_{33}$$

7. Calculate the LU factors of A_{33}:

$$A_{33} = L_{33} U_{33}$$

LAPACK Metodology

This algorithm deals with the blocks of A by column. Other computation order is possible.

Remarks

- LAPACK codes themselves employ standard Fortran 77 only.
- Parallelism is achieved due to BLAS (Level 3 BLAS mostly).
- It works on shared memory machines

ScaLAPACK

The **ScaLAPACK (or Scalable LAPACK)** library includes a subset of LAPACK routines redesigned for distributed memory MIMD parallel computers. It is currently written in a Single-Program-Multiple-Data style using explicit message passing for interprocessor communication. It assumes matrices are laid out in a two-dimensional block cyclic decomposition

Block LU

We divide our matrix A into m blocks, where each A_{ij} is the matrix on $n_i \times n_j$ dimensions.

$$\begin{pmatrix} A_{11} & \dots & A_{1,k} & \dots & A_{1,m} & \\ \dots & \dots & \dots & \dots & \dots & \\ A_{k,1} & \dots & A_{k,k} & \dots & A_{k,m} & \\ \dots & \dots & \dots & \dots & \dots & \dots \\ A_{m,1} & \dots & A_{m,k} & \dots & A_{m,m} & \end{pmatrix} \begin{pmatrix} X_1 \\ \dots \\ X_k \\ \dots \\ X_m \end{pmatrix} = \begin{pmatrix} B_1 \\ \dots \\ B_k \\ \dots \\ B_m \end{pmatrix}$$

Block LU

X and B are divided onto subvectors

$$X = (X_1 \quad X_2 \quad \ldots \quad X_k \quad \ldots \quad X_{m-1} \quad X_m)^T$$

$$B = (B_1 \quad B_2 \quad \ldots \quad B_k \quad \ldots \quad B_{m-1} \quad B_m \quad)^T$$

B_i and X_i , for i=1,...,m, are vectors.

Block LU

A is decomposed into a product of two matrices L and U where

$$L = \begin{pmatrix} I & 0 & \dots & 0 & \dots & 0 & 0 \\ L_{21} & I & \dots & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ L_{k,1} & L_{k,2} & \dots & I & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ L_{m-1,1} & L_{m-1,2} & \dots & L_{m-1,k} & \dots & I & 0 \\ L_{m,1} & L_{m,2} & \dots & L_{m,k} & \dots & L_{m,m-1} & I \end{pmatrix}$$

Block LU

$$U = \begin{pmatrix} U_{11} & U_{12} & \dots & U_{1,k} & \dots & U_{1,m-1} & U_{1,m} \\ 0 & U_{22} & \dots & U_{2,k} & \dots & U_{2,m-1} & U_{2,m} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & U_{k,k} & \dots & U_{k,n-1} & U_{k,n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & \dots & U_{m-1,m-1} & U_{m-1,m} \\ 0 & 0 & \dots & 0 & \dots & 0 & U_{m,m} \end{pmatrix}$$

Block LU

If A_{11} is non-singular we can define the $L^{(1)}$ and $U^{(1)}$ matrixes:

$$L^{(1)} = \begin{pmatrix} I \\ A_{21}A_{11}^{-1} \\ A_{31}A_{11}^{-1} \\ \dots \\ A_{m-1,1}A_{11}^{-1} \\ A_{m,1}A_{11}^{-1} \end{pmatrix} \qquad U^{(1)} = \begin{pmatrix} A_{11} \\ A_{12} \\ A_{13} \\ \dots \\ A_{1,m-1} \\ A_{1,m} \end{pmatrix}$$

Block LU

$$A^{(1)} = A - L^{(1)}U^{(1)^T} =$$

$$\begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & A_{22} - A_{21}A_{12}A_{11}^{-1} & \dots & A_{2,m} - A_{21}A_{1m}A_{11}^{-1} \\ \dots & \dots & \dots & \dots \\ 0 & A_{m-1,2} - A_{m-1,1}A_{12}A_{11}^{-1} & \dots & A_{m-1,n} - A_{mn-1,1}A_{1m}A_{11}^{-1} \\ 0 & A_{m,2} - A_{21}A_{12}A_{11}^{-1} & \dots & A_{m,m} - A_{m1}A_{1m}A_{11}^{-1} \end{pmatrix}$$

Block LU

After $k-1$ stages of this elimination we have the partially eliminated matrix:

$$A^{(k)} = A^{(k-1)} - L^{(k)} {U^{(k)}}^T =$$

$$\begin{pmatrix} 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & A^{(k)}_{kk} & \dots & A^{(k)}_{k,m} \\ \dots & \dots & \dots & \dots & & \\ 0 & \dots & 0 & A^{(k)}_{m-1,k} & \dots & A^{(k)}_{n-1,m} \\ 0 & \dots & 0 & A^{(k)}_{m,k} & \dots & A^{(k)}_{m,m} \end{pmatrix}$$

Block LU

Where $L_i^{(k)} = U_i^{(k)} = \mathbf{0}$ dla $ik - 1$, $L_k^{(k)} = I$ and $A_{ij}^{(k)} = \mathbf{0}$ dla ik or jk.

Block LU

Provided that $A_{kk}^{(k)}$ is non-singular we have:

$$L^{(k)} = \begin{pmatrix} 0 \\ \dots \\ 0 \\ 1 \\ A_{k+1,k}^{(k-1)} A_{kk}^{(k-1)^{-1}} \\ A_{31}^{(k-1)} A_{kk}^{(k-1)^{-1}} \\ \dots \\ A_{m-1,1}^{(k-1)} A_{kk}^{(k-1)^{-1}} \\ A_{m,1}^{(k-1)} A_{kk}^{(k-1)^{-1}} \end{pmatrix} \qquad U^{(k)} = \begin{pmatrix} 0 \\ \dots \\ 0 \\ A_{kk}^{(k-1)} \\ A_{k,k+1}^{(k-1)} \\ A_{k,k+2}^{(k-1)} \\ \dots \\ A_{k,m-1}^{(k-1)} \\ A_{k,m}^{(k-1)} \end{pmatrix}$$

Block LU

$$A_{ij}^{(k)} = A_{ij}^{(k-1)} - A_{ik}^{(k-1)} A_{kj}^{(k-1)} {A_{kk}^{(k-1)}}^{-1}$$

$i, j = k+1, k+2, ..., m$.

After m stages of this elimination procedure we find that:

$$A = L^{(1)} {U^{(1)}}^T + A^{(1)}$$
$$= L^{(1)} {U^{(1)}}^T + L^{(12)} {U^{(2)}}^T + A^{(2)}$$
$$... = L^{(1)} {U^{(1)}}^T + L^{(12)} {U^{(2)}}^T + ... + L^{(m)} {U^{(m)}}^T + A^{(m)}$$

where $A^{(m)}$ is the zero matrix.

Block LU

Hence

$$L = \begin{pmatrix} L^{(1)} & L^{(2)} & \dots & L^{(m)} \end{pmatrix}$$

$$U = \begin{pmatrix} U^{(1)^T} \\ U^{(2)^T} \\ \dots \\ U^{(m)^T} \end{pmatrix}$$

L is a unit diagonal, lower triangular, block matrix, and U is upper triangular block matrix.

Block LU

Each process P_{st} waits for it's pre-processes. for P_{st} where $s < t$ (matrix U) there are: $P_{s1}, P_{s2}, ...P_{s,s-1}$ and $P_{1t}, P_{2t}, ...P_{s-1,t}$.

$$\begin{array}{cccccccccc}
U_{11} & * & \dots & * & * & \dots & U_{1t} & \dots & * & * \\
* & U_{22} & \dots & * & * & \dots & U_{2t} & \dots & * & * \\
* & * & \dots & * & * & \dots & & \dots & * & * \\
* & * & \dots & U_{s-1,s-1} & * & \dots & U_{s-1,t} & \dots & * & * \\
L_{s1} & L_{s2} & \dots & L_{s,s-1} & U_{ss} & \dots & U_{st} & \dots & * & * \\
* & * & \dots & * & * & \dots & & \dots & * & * \\
* & * & \dots & * & * & \dots & & \dots & * & *
\end{array}$$

Block LU

For P_{st} where $s > t$ (matrix L) there are: $P_{s1}, P_{s2}, \ldots P_{s,t-1}$ and $P_{1t}, P_{2t}, \ldots P_{t,t}$.

$$
\begin{array}{cccccccccc}
U_{11} & * & \ldots & * & U_{1t} & \ldots & \ldots & * & \ldots \\
* & U_{22} & \ldots & * & U_{2t} & \ldots & \ldots & * & \ldots \\
* & * & \ldots & * & * & \ldots & \ldots & * & \ldots \\
* & * & \ldots & * & U_{t,t} & \ldots & \ldots & * & \ldots \\
* & * & \ldots & * & * & \ldots & \ldots & * & \ldots \\
* & * & \ldots & * & L_{s-1,t} & \ldots & U_{s-1,s-1} & * & \ldots \\
L_{s1} & L_{s2} & \ldots & L_{s,t-1} & L_{st} & \ldots & L_{s,s-1} & U_{ss} & \ldots \\
* & * & \ldots & * & * & \ldots & & * & \ldots
\end{array}
$$

Sparse LU

- Row-Cholesky: Taking i in the outer loop, successive rows of L are computed one by one, with the inner loops solving a triangular system for each new row in terms of the previously computed rows.
- Column-Cholesky: Taking j in the outer loop, successive columns of L are computed one by one, with the inner loops computing a matrix-vector product that gives the effect of previously computed columns on the column currently being computed.

Sparse LU

- Submatrix-Cholesky: Taking k in the outer loop, successive columns of L are computed one by one, with the inner loops applying the current column as a rank-1 update to the remaining unreduced submatrix.

LU - loops order

6 different methods are used, depending on the data spliting:

- ijk loop : A by columns (ddot)
- ikj loop - A by rows (daxpy)
- jik loop : A by columns (ddot)
- jki loop - A by rows (daxpy)
- kij loop : A by rows (daxpy)
- kji loop : A by columns (daxpy)

ijk - LU

ijk loop : A- by column (ddot)

```
for i=2,n
    for j=2,i
        l(i,j-1)=a(i,j-1)/a(j-1, j-1)
        for k=1,j-1
            a(i,j)=a(i,j)-l(i,k) * a(k,j)
        endfor
    endfor
    for j=i+1,n
        for k=1,i-1
            a(i,j)=a(i,j)-l(i,k) * a(k,j)
        endfor
    endfor
```

ikj - LU

ikj loop - A by row (daxpy)

```
for i=2,n
    for k=1,i-1
        l(i,k)=a(i,k)/a(k, k)
        for j=k+1,n
            a(i,j)=a(i,j)-l(i,k) * a(k,j)
        endfor
        endfor
endfor
```

jik - LU

jik loop : A- by column (ddot)

```
for j=2,n
    for p=j,n
        l(p,j-1)=a(p,j-1)/a(j-1, j-1)
    endfor
    for i=2,j
        for k=1,i-1
            a(i,j)=a(i,j)-l(i,k) * a(k,j)
        endfor
    endfor
```

jik - LU - cont.

```
for i=j+1,n
    for k=1,j-1
        a(i,j)=a(i,j)-l(i,k) * a(k,j)
    endfor
endfor
```

jki - LU

jki loop - A by row (daxpy)

```
for j=2,n
    for p=j,n
        l(p,j-1)=a(p,j-1)/a(j-1, j-1)
    endfor
    for k=1,j-1
        for i=k+1,n
            a(i,j)=a(i,j)-l(i,k) * a(k,j)
        endfor
    endfor
endfor
```

kij - LU

kij loop : A- by row (daxpy)

```
for k=1,n-1
    for i=k+1,n
        l(i,k)=a(i,k)/a(k, k)
        for j=k+1,n
            a(i,j)=a(i,j)-l(i,k) * a(k,j)
        endfor
    endfor
endfor
```

kji - LU

kji loop : A- by column (daxpy)

```
for k=1,n-1
    for p=k+1,n
        l(p,k)=a(p,k)/a(k, k)
    endfor
    for j=k+1,n1
        for i=k+1,n
            a(i,j)=a(i,j)-l(i,k) * a(k,j)
        endfor
    endfor
endfor
```

Thank you for your attention!

Any questions are welcome.