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

Some sparse LU considerations

Michael Saunders
iCME, Stanford University

Cosener's House, Abingdon, Oxfordshire

Some sparse LU considerations

John Reid's LA05 software showed that the Bartels-Golub update could be implemented efficiently for sparse matrices. It strongly influenced the development of LUSOL, the engine for the large-scale optimization solvers MINOS, SNOPT, and lp_solve.

To gain rank-revealing properties, LUSOL implements threshold versions of rook pivoting and complete pivoting. In contrast to the folklore for complete pivoting, the cost of tracking the largest remaining nonzero is negligible (with the help of a heap data structure). We consider the value of tracking "Amax" for all of the threshold pivoting options.

We also review some of John's other invaluable contributions to scientific computing.

Supported by the Office of Naval Research and AHPCRC

Background

1970

- Gene, NPL, Harwell

1970

- Gene, NPL, Harwell
- Roger Fletcher, Mike Powell, John Reid

1970

- Gene, NPL, Harwell
- Roger Fletcher, Mike Powell, John Reid
- HSL (since 1963)

1970

- Gene, NPL, Harwell
- Roger Fletcher, Mike Powell, John Reid
- HSL (since 1963)
- Algol, Algol-W, Fortran 66

1975–77

- At SOL, finally using F66 for MINOS 1.0

1975–77

- At SOL, finally using F66 for MINOS 1.0
- Duff and Reid: MA28 (sparse LU)

1975–77

- At SOL, finally using F66 for MINOS 1.0
- Duff and Reid: MA28 (sparse LU)
- Reid: LA05 (sparse LU + updates)

1979–84

- At SOL, still using F66 for MINOS 5.0, QPSOL, NPSOL

1979–84

- At SOL, still using F66 for MINOS 5.0, QPSOL, NPSOL
- Still using upper case

```
X(J) = A(I,J)
```

1979–84

- At SOL, still using F66 for MINOS 5.0, QPSOL, NPSOL
- Still using upper case

```
X(J) = A(I,J)
```

- Great effort to follow ANSI standard

```
DATA D/8H GOODBYE/
```

required in F66 because ’ characters not available on all keyboards

1979–84

- At SOL, still using F66 for MINOS 5.0, QPSOL, NPSOL
- Still using upper case

```
X(J) = A(I,J)
```

- Great effort to follow ANSI standard

```
DATA D/8H GOODBYE/
```

  required in F66 because ’ characters not available on all keyboards
- Not allowed in F77!

1986–94

- At SOL, finally using F77

1986–94

- At SOL, finally using F77
- At St Girons, John describes advanced f90 features

1986–94

- At SOL, finally using F77
- At St Girons, John describes advanced f90 features
- Independent discovery of Matlab-type operations

```
x(1:n) = 0.0
x(:) = y(1:n)
x = y
```

1986–94

- At SOL, finally using F77
- At St Girons, John describes advanced f90 features
- Independent discovery of Matlab-type operations

```
x(1:n) = 0.0
x(:) = y(1:n)
x = y
```

- Some differences

```
options.maxit = 1000 (Matlab)
options%maxit = 1000 (f90)
```

An f90 feature

- f77:

```
      call subr( a, b, c,
     &           d, e, f, g )
```

An f90 feature

- f77:

```
      call subr( a, b, c,
     &           d, e, f, g )
```

- f90:

```
      call subr( a, b, c, &
                 d, e, f, g )
```

- f77:

```
      call subr( a, b, c,
     &           d, e, f, g,
     &           h, i, j, k, l, m,
     &           p, q )
```

- f77:

```
      call subr( a, b, c,
     &           d, e, f, g,
     &           h, i, j, k, l, m,
     &           p, q )
```

- f90:

```
      call subr( a, b, c, &
                 d, e, f, g, &
                 h, i, j, k, l, m, &
                 p, q )
```

- f77:

```
      call subr( a, b, c,
     &           d, e, f, g,
     &           h, i, j, k, l, m,
     &           p, q )
```

- f90:

```
      call subr( a, b, c, &
                 d, e, f, g, &
                 h, i, j, k, l, m, &
                 p, q )
```

- Eventually

```
      call subr( a, b, c,          &
                 d, e, f, g,       &
                 h, i, j, k, l, m, &
                 p, q )
```

- The LaTeX guide:

```
\begin{tabular}{lr}
   gnat & 3.24 \\
   gnu  & 24,985.47 \\
   f90  & 10.00 \\
   g95  & 0.00
\end{tabular}
```

- The LaTeX guide:

```
\begin{tabular}{lr}
   gnat & 3.24 \\
   gnu  & 24,985.47 \\
   f90  & 10.00 \\
   g95  & 0.00
\end{tabular}
```

- Preferably

```
\begin{tabular}{lr}
   gnat &      3.24
\\ gnu  & 24,985.47
\\ f90  &     10.00
\\ g95  &      0.00
\end{tabular}
```

Marvels of conciseness

- Leslie Lamport:
  The LaTeX guide

Marvels of conciseness

- Leslie Lamport:
  The LaTeX guide
- Metcalf and Reid:
  Fortran 8x, 90, 90/95 Explained

Marvels of conciseness

- Leslie Lamport:
  The LaTeX guide
- Metcalf and Reid:
  Fortran 8x, 90, 90/95 Explained
- Metcalf, Reid, and Cohen:
  Fortran 95/2003 Explained

Marvels of conciseness

- Leslie Lamport:
  The LaTeX guide
- Metcalf and Reid:
  Fortran 8x, 90, 90/95 Explained
- Metcalf, Reid, and Cohen:
  Fortran 95/2003 Explained
- JKR:
  Floating-point exception handling

Updating matrix factors

Updating matrix factors

The Simplex method

- Every iteration, solve $By = a$, $B^T w = c$

Updating matrix factors

The Simplex method

- Every iteration, solve $By = a$, $B^T w = c$
- Replace a column of B

Updating matrix factors

The Simplex method

- Every iteration, solve $By = a$, $B^T w = c$
- Replace a column of B

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}$$

Updating matrix factors

The Simplex method

- Every iteration, solve $By = a$, $B^T w = c$
- Replace a column of B

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}$$

The Product-Form update:

$$\boxed{B_k = B_0 T_1 T_2 \ldots T_k}$$

The Product-Form update

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}, \qquad \mu \equiv \frac{\max |y_i|}{|y_p|}$$

- T is a permuted triangle

The Product-Form update

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}, \qquad \mu \equiv \frac{\max |y_i|}{|y_p|}$$

- T is a permuted triangle
- $\text{cond}(T) = \mu$?

The Product-Form update

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}, \qquad \mu \equiv \frac{\max |y_i|}{|y_p|}$$

- T is a permuted triangle
- $\text{cond}(T) = \mu$?
- $\text{cond}(T) \approx \mu^2$!

The Product-Form update

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}, \qquad \mu \equiv \frac{\max |y_i|}{|y_p|}$$

- T is a permuted triangle
- $\text{cond}(T) = \mu$?
- $\text{cond}(T) \approx \mu^2$!
- Stability test: Don't update if μ is too big

The Product-Form update

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}, \qquad \mu \equiv \frac{\max |y_i|}{|y_p|}$$

- T is a permuted triangle
- $\text{cond}(T) = \mu$?
- $\text{cond}(T) \approx \mu^2$!
- Stability test: Don't update if μ is too big
- $B_k = B_0 T_1 T_2 \ldots T_k$ B_0 can be a *black box*

The Product-Form update

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}, \qquad \mu \equiv \frac{\max |y_i|}{|y_p|}$$

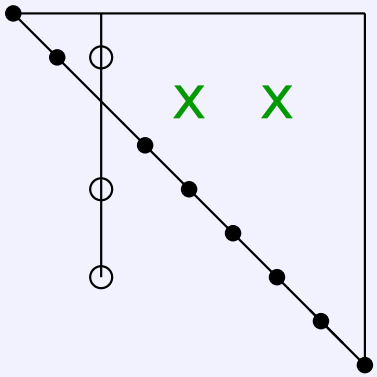
- T is a permuted triangle
- $\mathrm{cond}(T) = \mu$?
- $\mathrm{cond}(T) \approx \mu^2$!
- Stability test: Don't update if μ is too big
- $B_k = B_0 T_1 T_2 \ldots T_k$ B_0 can be a *black box*
- However: y can be rather dense

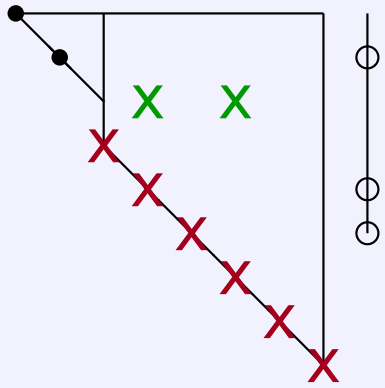
The Product-Form update

$$\bar{B} = BT, \qquad T = \begin{pmatrix} 1 & & y_1 & & \\ & 1 & y_2 & & \\ & & \vdots & & \\ & & y_p & & \\ & & \vdots & 1 & \\ & & y_m & & 1 \end{pmatrix}, \qquad \mu \equiv \frac{\max |y_i|}{|y_p|}$$

- T is a permuted triangle
- $\mathrm{cond}(T) = \mu$?
- $\mathrm{cond}(T) \approx \mu^2$!
- Stability test: Don't update if μ is too big
- $B_k = B_0 T_1 T_2 \ldots T_k$ B_0 can be a *black box*
- However: y can be rather dense
- Doesn't recover if B_k is suddenly well-conditioned

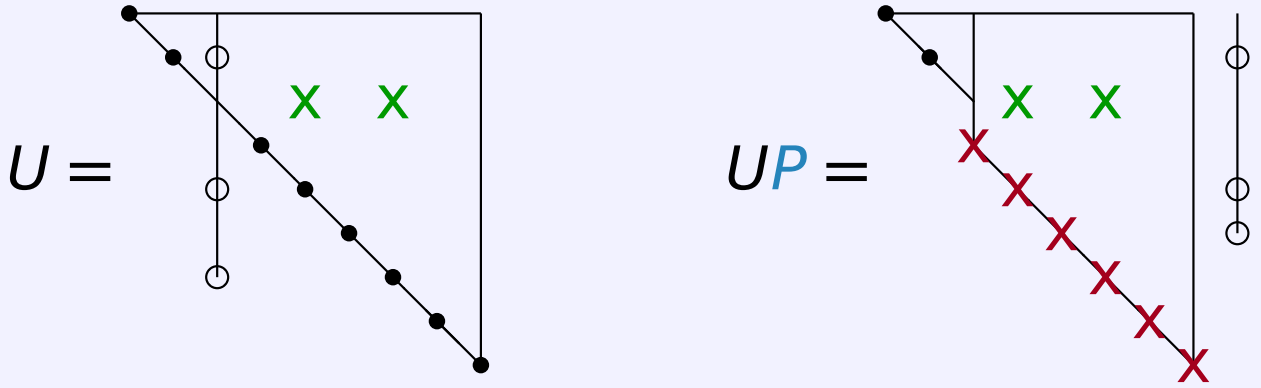
The Bartels-Golub update

$U =$ 

$UP =$ 

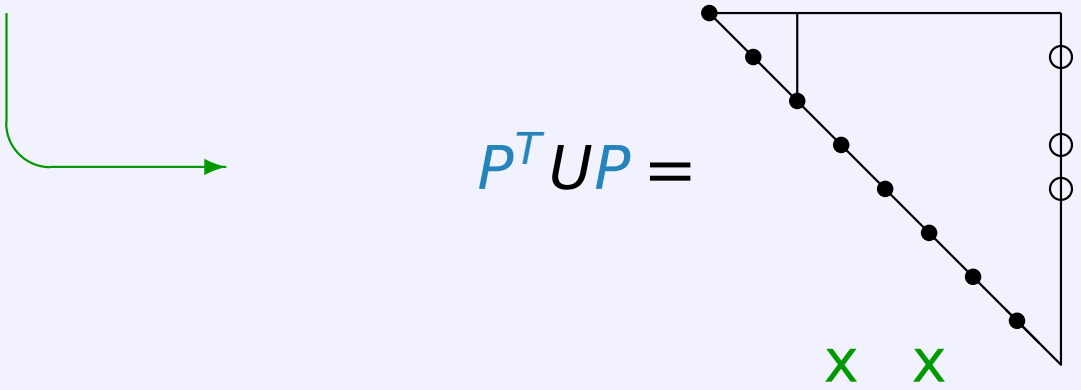
Apparently many x to eliminate

The Bartels-Golub update



Apparently many x to eliminate

The BG-Reid update 1976

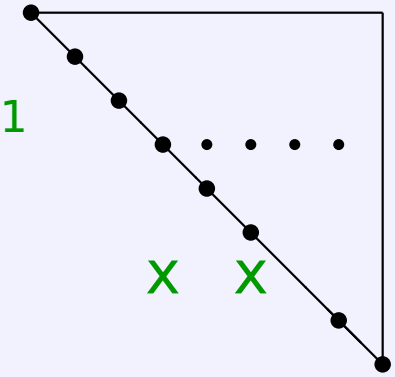


Only a few x to eliminate

LU updates

Reid: LA05, LA15

$U \leftarrow \ldots M_2 M_1$



$M_j = \begin{pmatrix} 1 & \\ \mu_j & 1 \end{pmatrix}$ or $\begin{pmatrix} 1 & \mu_j \\ & 1 \end{pmatrix}$ (several per update)

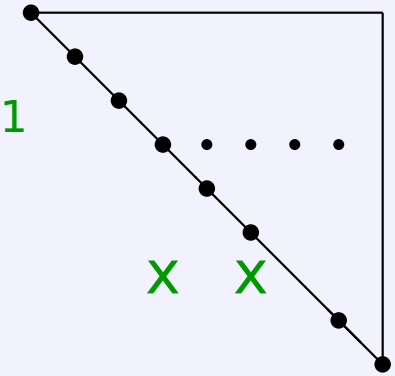
Store U by rows $\cdots$ to allow fill-in

Hundreds of updates are stable if $|\mu_j| \leq 10$ say

LU updates

Reid: LA05, LA15

$U \leftarrow \ldots M_2 M_1$



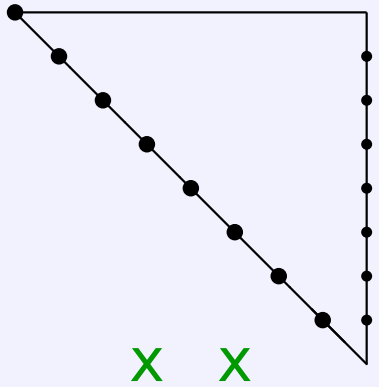
$M_j = \begin{pmatrix} 1 & \\ \mu_j & 1 \end{pmatrix}$ or $\begin{pmatrix} 1 & \mu_j \\ & 1 \end{pmatrix}$ (several per update)

Store U by rows $\cdots$ to allow fill-in

Hundreds of updates are stable if $|\mu_j| \leq 10$ say

Forrest-Tomlin:

$$U \leftarrow \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 1 & \\ & & \mu_1 & \mu_2 & 1 \end{pmatrix}$$



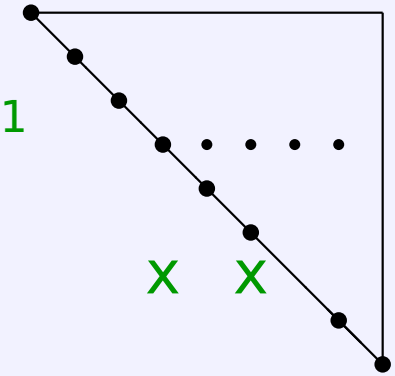
No permutations

No fill-in, store U by columns

LU updates

Reid: LA05, LA15

$U \leftarrow \ldots M_2 M_1$



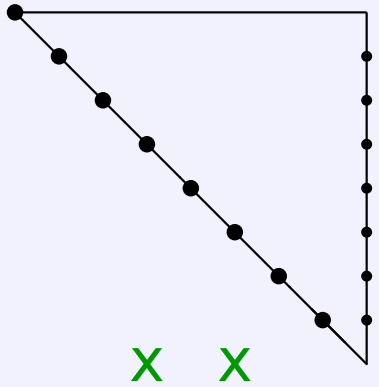
$M_j = \begin{pmatrix} 1 & \\ \mu_j & 1 \end{pmatrix}$ or $\begin{pmatrix} 1 & \mu_j \\ & 1 \end{pmatrix}$ (several per update)

Store U by rows $\cdots$ to allow fill-in

Hundreds of updates are stable if $|\mu_j| \leq 10$ say

Forrest-Tomlin:

$$U \leftarrow \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & 1 & \\ & & \mu_1 & \mu_2 & 1 \end{pmatrix}$$



No permutations

No update if $|\mu_j|$ big

No fill-in, store U by columns

Sparse LU

Sparse LU

What is the Chinese word for pivoting?

- Replace column in basis matrix?

Sparse LU

What is the Chinese word for pivoting?

- Replace column in basis matrix?
- Threshold pivoting: Permute for sparsity

Sparse LU

What is the Chinese word for pivoting?

- Replace column in basis matrix?
- Threshold pivoting: Permute for sparsity
- Threshold pivoting: Permute for stability

LUSOL $PAQ = LU$

- A sparse, rectangular

LUSOL $PAQ = LU$

- A sparse, rectangular
- L unit diagonals, well-conditioned

LUSOL $PAQ = LU$

- A sparse, rectangular
- L unit diagonals, well-conditioned
- Factor: like Duff & Reid MA28, LA05
  (Markowitz search + threshold pivoting)

LUSOL $PAQ = LU$

- A sparse, rectangular
- L unit diagonals, well-conditioned
- Factor: like Duff & Reid MA28, LA05
  (Markowitz search + threshold pivoting)
- Update: like Bartels-Golub-Reid

LUSOL $PAQ = LU$

- A sparse, rectangular
- L unit diagonals, well-conditioned
- Factor: like Duff & Reid MA28, LA05
  (Markowitz search + threshold pivoting)
- Update: like Bartels-Golub-Reid

Better to think of as $PAQ = LDU$

LUSOL $PAQ = LDU$

Factor tolerance τ (stability vs sparsity)

$|L_{ij}| \leq \tau$ always

$|U_{ij}| \leq \tau$ for TRP, TCP

$1 < \tau \leq 100$

LUSOL $PAQ = LDU$

Factor tolerance τ (stability vs sparsity)

$|L_{ij}| \leq \tau$ always
$|U_{ij}| \leq \tau$ for TRP, TCP
$1 < \tau \leq 100$

TPP Threshold Partial Pivoting
L is well-conditioned if $\tau \leq 10$

LUSOL $PAQ = LDU$

Factor tolerance τ (stability vs sparsity)

$|L_{ij}| \leq \tau$ always
$|U_{ij}| \leq \tau$ for TRP, TCP
$1 < \tau \leq 100$

TPP Threshold Partial Pivoting
L is well-conditioned if $\tau \leq 10$

TRP Threshold Rook Pivoting
TCP Threshold Complete Pivoting
L, U are well-conditioned if $\tau \leq 10$
D is rank-revealing if $\tau \leq 2.5, 2, 1.5, 1.2, \ldots$

MA48, HSL_MA78 $PAQ = LDU$

Pivot tolerance u (stability vs sparsity)

$|L_{ij}| \leq 1/u$ always
$|U_{ij}| \leq 1/u$ for TRP
$0 \leq u \leq 1$

MA48, HSL_MA78 $PAQ = LDU$

Pivot tolerance u (stability vs sparsity)

$$|L_{ij}| \leq 1/u \quad \text{always}$$
$$|U_{ij}| \leq 1/u \quad \text{for TRP}$$
$$0 \leq u \leq 1$$

TPP Threshold Partial Pivoting

L is well-conditioned if $u \geq 0.1$

MA48, HSL_MA78 $PAQ = LDU$

Pivot tolerance u (stability vs sparsity)

$$|L_{ij}| \le 1/u \quad \text{always}$$
$$|U_{ij}| \le 1/u \quad \text{for TRP}$$
$$0 \le u \le 1$$

TPP Threshold Partial Pivoting
L is well-conditioned if $u \ge 0.1$

TRP Threshold Rook Pivoting
L, U are well-conditioned if $u \ge 0.1$
D is rank-revealing if $u \ge \frac{1}{2.5}, \frac{1}{2}, \frac{1}{1.5}, \frac{1}{1.2}, \ldots$

MA27, MA47, MA57, . . . $PAP^T = LDL^T$

Pivot tolerance u (stability vs sparsity)

$$|L_{ij}| \leq 1/u$$
$$0 \leq u \leq 0.5$$

MA27, MA47, MA57, . . . $PAP^T = LDL^T$

Pivot tolerance u (stability vs sparsity)

$$|L_{ij}| \leq 1/u$$
$$0 \leq u \leq 0.5$$

TPP Threshold Partial Pivoting

Often ok with $u = 0.01, 0.001$

L is well-conditioned if $u \geq 0.1$

MA27, MA47, MA57, . . . $PAP^T = LDL^T$

Pivot tolerance u (stability vs sparsity)

$$|L_{ij}| \leq 1/u$$
$$0 \leq u \leq 0.5$$

TPP Threshold Partial Pivoting
Often ok with $u = 0.01, 0.001$
L is well-conditioned if $u \geq 0.1$

TRP Threshold Rook Pivoting (symmetric version)
D is rank-revealing if $u \geq \frac{1}{2.5}$ or $\frac{1}{2}, \frac{1}{1.5}, \frac{1}{1.2}, \ldots$?

The need for rank-revealing LU

$$A = \begin{pmatrix} \delta & 1 & 1 & 1 \\ & \delta & 1 & 1 \\ & & \delta & 1 \\ & & & \delta \end{pmatrix} = LDU \qquad \delta \text{ small}$$

The need for rank-revealing LU

$$A = \begin{pmatrix} \delta & 1 & 1 & 1 \\ & \delta & 1 & 1 \\ & & \delta & 1 \\ & & & \delta \end{pmatrix} = LDU \qquad \delta \text{ small}$$

TPP would give $L = I$, $D = \delta I$ $\qquad$ $\mathrm{rank}(A) = 4$ or 0(!)

The need for rank-revealing LU

$$A = \begin{pmatrix} \delta & 1 & 1 & 1 \\ & \delta & 1 & 1 \\ & & \delta & 1 \\ & & & \delta \end{pmatrix} = LDU \qquad \delta \text{ small}$$

TPP would give $L = I$, $D = \delta I$ $\qquad \text{rank}(A) = 4$ or 0(!)

TRP or TCP would give

$$\begin{pmatrix} 1 & 1 & 1 & \delta \\ \delta & 1 & 1 & \\ & \delta & 1 & \\ & & \delta & \end{pmatrix} \approx L \begin{pmatrix} 1 & 1 & 1 & \delta \\ & 1 & 1 & -\delta^2 \\ & & 1 & \delta^3 \\ & & & -\delta^4 \end{pmatrix} \qquad \text{rank}(A) \approx 3$$

Maintaining Amax

Maintaining Amax

$$A = \begin{pmatrix} \delta & 1 & 1 & 1 \\ & \delta & 1 & 1 \\ & & \delta & 1 \\ & & & \delta \end{pmatrix}$$

TPP needs α_j = max element in cols standard

Maintaining Amax

$$A = \begin{pmatrix} \delta & 1 & 1 & 1 \\ & \delta & 1 & 1 \\ & & \delta & 1 \\ & & & \delta \end{pmatrix}$$

TPP needs α_j = max element in cols — standard

TRP needs β_i = max element in rows — expensive

Maintaining Amax

$$A = \begin{pmatrix} \delta & 1 & 1 & 1 \\ & \delta & 1 & 1 \\ & & \delta & 1 \\ & & & \delta \end{pmatrix}$$

TPP needs α_j = max element in cols — standard

TRP needs β_i = max element in rows — expensive

TCP needs Amax $= \max |A_{ij}|$ — cheap!
(keep α_j in a heap)

Maintaining Amax

$$A = \begin{pmatrix} \delta & 1 & 1 & 1 \\ & \delta & 1 & 1 \\ & & \delta & 1 \\ & & & \delta \end{pmatrix}$$

TPP needs α_j = max element in cols — standard

TRP needs β_i = max element in rows — expensive

TCP needs Amax = $\max |A_{ij}|$ — cheap! (keep α_j in a heap)

Plan: Track Amax always (not just for TCP)

Using Amax in Markowitz search

Ignore any $|A_{ij}| <$ Atol

Atol $\equiv$ Amax/τ_{CP}

$\tau_{CP} = 1000.0$ say

Using Amax in Markowitz search

Ignore any $|A_{ij}| <$ Atol

Atol $\equiv$ Amax$/\tau_{\text{CP}}$

$\tau_{\text{CP}} = 1000.0$ say

TPP can skip whole columns if $\alpha_j <$ Atol

Avoids $\begin{pmatrix} \delta & 1 & 1 \\ & \delta & 1 \\ & & \delta \end{pmatrix}, \begin{pmatrix} 10^{-8} & x & x \\ 10^{-9} & x & x \\ & x & x \end{pmatrix}$

Using Amax in Markowitz search

Ignore any $|A_{ij}| <$ Atol

Atol $\equiv$ Amax$/\tau_{CP}$

$\tau_{CP} = 1000.0$ say

TPP can skip whole columns if $\alpha_j <$ Atol

$$\text{Avoids} \begin{pmatrix} \delta & 1 & 1 \\ & \delta & 1 \\ & & \delta \end{pmatrix}, \begin{pmatrix} 10^{-8} & x & x \\ 10^{-9} & x & x \\ & x & x \end{pmatrix}$$

TRP can skip whole rows if $\beta_i <$ Atol

$$\text{Avoids} \begin{pmatrix} 10^{-8} & 10^{-9} & 10^{-9} \\ 10^{-9} & x & x \\ & x & x \end{pmatrix}$$

Ignore any $|A_{ij}| <$ `Atol` $\quad$ `Atol` $\equiv$ `Amax`$/\tau_{\mathrm{CP}}$

Cons:

Ignore any $|A_{ij}| <$ `Atol` $\qquad$ `Atol` $\equiv$ `Amax`$/\tau_{\text{CP}}$

Cons:

- Always stricter (so LU factors usually less sparse)

Ignore any $|A_{ij}| <$ `Atol` `Atol` $\equiv$ `Amax`$/\tau_{\text{CP}}$

Cons:

- Always stricter (so LU factors usually less sparse)
- Sometimes `Amax` $= 10^{11}$!

Ignore any $|A_{ij}| <$ Atol Atol $\equiv$ Amax/τ_{CP}

Cons:

- Always stricter (so LU factors usually less sparse)
- Sometimes Amax $= 10^{11}$!
- Need big τ_{CP} if A not well scaled

Ignore any $|A_{ij}| <$ Atol Atol $\equiv$ Amax/τ_{CP}

Cons:

- Always stricter (so LU factors usually less sparse)
- Sometimes Amax $= 10^{11}$!
- Need big τ_{CP} if A not well scaled

$\tau_{CP} = 100.0$ 1000.0 10^6??

f95 test program

```
program lusolTestProgram4

  use   lusolScaleModule, only : gmgetscales, gmscale, gmunscale, triple2css, css2triple
  use   lusolTestModule3, only : dataAb, lusolTest3

  implicit none
  intrinsic             :: trim

  integer, parameter    :: maxm   =   200000, &   ! Storage for A, b, xexact.
                           maxn   =   200000, &
                           maxnz  =  1000000
  real(8)               :: Aij(maxnz)             ! A in triple format
  integer               :: iA(maxnz), jA(maxnz)   !
  real(8)               :: aa(maxnz)              ! A in CSS format
  integer               :: ha(maxnz), ka(maxn+1)  !
  real(8)               :: b(maxm), xexact(maxm)  ! b = A*xexact
  real(8)               :: rscale(maxm), cscale(maxn)

  integer, parameter    :: lena   = 25000000      ! Storage for LUSOL.
  integer               :: luparm(30)
  real(8)               :: parmlu(30) , a(lena)    , w(maxn)
  integer               :: indc(lena) , indr(lena) ,         &
                           ip(maxm)   , iq(maxn)   ,         &
                           lenc(maxn) , lenr(maxm) ,         &
                           iploc(maxn), iqloc(maxm),         &
                           ipinv(maxm), iqinv(maxn),         &
                           locc(maxn) , locr(maxm)
```

```
module lusolTestModule3

  implicit none
  public    :: lusolTest3, dataAb, dnormi
  intrinsic :: cpu_time, max, nint, real

contains

  subroutine lusolTest3( m    , n     , nelem , screen,              &
                         Aij  , iA    , jA    , b     , xexact,      &
                         lena , luparm, parmlu,                      &
                         a    , indc  , indr  , ip    , iq,          &
                         lenc , lenr  , locc  , locr  ,              &
                         iploc, iqloc , ipinv , iqinv , w , inform )

    integer, intent(in)    :: m, n, nelem, screen, lena
    real(8), intent(in)    :: Aij(nelem), b(m), xexact(n)
    integer, intent(in)    :: iA(nelem), jA(nelem)
    integer, intent(inout) :: luparm(30)
    real(8), intent(inout) :: parmlu(30)
    real(8), intent(out)   :: a(lena),    w(n)
    integer, intent(out)   :: indc(lena), indr(lena),      &
                              ip(m)   , iq(n)  ,           &
                              lenc(n) , lenr(m),           &
                              locc(n) , locr(m),           &
                              iploc(n),iqloc(m),           &
                              ipinv(m),iqinv(n)
    integer, intent(out)   :: inform
```

```
  !-------------------------------------------------------------------
  ! Load A into (a, indc, indr).
  !-------------------------------------------------------------------
  a   (1:nelem) = Aij(1:nelem)
  indc(1:nelem) = iA (1:nelem)
  indr(1:nelem) = jA (1:nelem)

  !-------------------------------------------------------------------
  ! Factor  A = L U.
  !-------------------------------------------------------------------
  call cpu_time(time1)
  call lu1fac( m    , n    , nelem, lena , luparm, parmlu,  &
               a    , indc , indr , ip   , iq    ,          &
               lenc , lenr , locc , locr ,                  &
               iploc, iqloc, ipinv, iqinv, w     , inform )
  call cpu_time(time2)
  write(screen, '(/ a,f10.2,a)') ' Time  =', time2-time1, ' seconds'

  ...

  if     (rnorm < bnorm  .and.  test <= 1d-9) then
     write(screen, *) 'rnorm is small.  The test seems successful.'
  elseif (rnorm < bnorm  .and.  test <= 1d-5) then
     write(screen, *) 'rnorm is not very small.  '
     write(screen, *) 'The matrix may be large or nearly singular,'
     write(screen, *) 'or the LU factors may not be good enough.'
     write(screen, *) 'Try a smaller factol and/or Rook Pivoting.'
  else
     ...
  end if

end subroutine lusolTest3
```

Numerical results

Test matrices

- Tim Davis: Univ of Florida Sparse Matrix Collection
  Over 2000 matrices
- Nick Gould, Yifan Hu, Jennifer Scott:
  Group GHS_indef
- 59 symmetric indefinite matrices (some KKTs)
- Sorted by nnz(A)
- Tried only first 13
- Even some of those are expensive!

GHS_indef matrices (first 13)

TPP $|L_{ij}| \leq 5.0$, $|A_{ij}| \geq \text{Amax}/100.0$ unscaled

matrix	m	n	nnz	Singularities	Original Time	tauCP=100.0 Time
laser	1239	3002	3002	9000	2	
qpband	1427	20000	20000	45000		
tuma2	1250	12992	12992	49365	3	5
bloweybq	1425	10001	10001	49999	1	
ncvxqp9	1243	16554	16554	54040	9	28
sit100	1245	10262	10262	61046	24	25
dtoc	1234	24993	24993	69972	4999	1	8
aug3d	1217	24300	24300	69984	12636	
ncvxqp1	1242	12111	12111	73963	17	560	796
aug2d	1215	29008	29008	76832	9800	
aug2dc	1216	30200	30200	80000	10200	
tuma1	1249	22967	22967	87760	18	31
linverse	1303	11999	11999	95977		

Hard to win on unscaled data with loose Factor tol

ncvxqp9

TPP 1.5

```
m        16554 =n        16554  Elems    54040  Amax   1.0E+00  Density   0.02  37.7 secs
Merit   6333.4  lenL    474706  L+U    1554126  Cmpressns   13  Increas   27.76
Utri       554  lenU   1079420  Ltol      1.50  Umax   2.3E+01  Ugrwth 2.3E+01
Ltri         0  dense1    1386  Lmax      1.50
bump     16000  dense2     947  DUmax  1.2E+01  DUmin  2.8E-05  condU  4.5E+05
```

TPP 1.5 $\tau_{CP} = 100.0$

```
m        16554 =n        16554  Elems    54040  Amax  1.00E+00  Density   0.02  32.6 secs
Merit   3707.6  lenL    676578  L+U    2029255  Cmpressns   20  Increas   36.55
Utri       554  lenU   1352677  Ltol     1.500  Umax  1.68E+01  Ugrwth 1.7E+01
Ltri         0  dense1    1708  Lmax     1.500  Akmax 1.68E+01  Agrwth 1.7E+01
bump     16000  dense2    1525  DUmax 1.05E+01  DUmin 5.29E-06  condU 1.98E+06
```

ncvxqp9

TPP 1.5

m	16554	=n	16554	Elems	54040	Amax	1.0E+00	Density	0.02	37.7 secs
Merit	6333.4	lenL	474706	L+U	1554126	Cmpressns	13	Incres	27.76	
Utri	554	lenU	1079420	Ltol	1.50	Umax	2.3E+01	Ugrwth	2.3E+01	
Ltri	0	dense1	1386	Lmax	1.50					
bump	16000	dense2	947	DUmax	1.2E+01	DUmin	2.8E-05	condU	4.5E+05	

TPP 1.5 $\tau_{CP} = 100.0$

m	16554	=n	16554	Elems	54040	Amax	1.00E+00	Density	0.02	32.6 secs
Merit	3707.6	lenL	676578	L+U	2029255	Cmpressns	20	Incres	36.55	
Utri	554	lenU	1352677	Ltol	1.500	Umax	1.68E+01	Ugrwth	1.7E+01	
Ltri	0	dense1	1708	Lmax	1.500	Akmax	1.68E+01	Agrwth	1.7E+01	
bump	16000	dense2	1525	DUmax	1.05E+01	DUmin	5.29E-06	condU	1.98E+06	

TRP 1.5

m	16554	=n	16554	Elems	54040	Amax	1.0E+00	Density	0.02	368.6 secs
MerRP	8050.4	lenL	1799272	L+U	3641392	Cmpressns	35	Incres	66.38	
Utri	554	lenU	1842120	Ltol	1.50	Umax	7.9E+00	Ugrwth	7.9E+00	
Ltri	0	dense1	1949	Lmax	1.50					
bump	16000	dense2	1694	DUmax	7.9E+00	DUmin	8.9E-08	condU	8.9E+07	

TRP 1.5 $\tau_{CP} = 100.0$

m	16554	=n	16554	Elems	54040	Amax	1.00E+00	Density	0.02	214.4 secs
MerRP	4000.2	lenL	2047418	L+U	4090319	Cmpressns	33	Incres	74.69	
Utri	554	lenU	2042901	Ltol	1.50	Umax	7.94E+00	Ugrwth	7.9E+00	
Ltri	0	dense1	1993	Lmax	1.50	Akmax	8.23E+00	Agrwth	8.2E+00	
bump	16000	dense2	1884	DUmax	7.94E+00	DUmin	8.87E-08	condU	8.95E+07	

Some gain on scaled data with tighter (rank-revealing) Factor tol

RR

RM

JR

RR

RM

JR

Thank you John
and Iain, Jennifer, Nick, ...