What should convergence criterion be?
Tricky, need to avoid being fooled by very ill-conditioned A that looks like it is converging.
See details in typed class notes.

Code available in LAPACK:
sgesvxx, dgesvxx

Ma221 Lecture 5 Segment 5

Return to minimizing communication.

Historically GEPP was written to perform most work using BLAS3,
used in LAPACK + ScALAPACK.

Idea similar to induction proof for GEPP, but instead of 1 column at a time,
will do b columns at a time, b is a tuning parameter: For simplicity, ignore pivoting.
\[
A = \begin{bmatrix}
\begin{array}{c|c}
A_{11} & A_{12} \\
\hline
A_{21} & A_{22}
\end{array}
\end{bmatrix}^{n-b}
\]

\[
= \begin{bmatrix}
L_{11}U_{11} & A_{12} \\
L_{21}U_{11} & A_{22}
\end{bmatrix}
\]

where we have performed GEPP using prior algorithm on
\[
\begin{bmatrix}
A_{11} \\
A_{21}
\end{bmatrix} = \begin{bmatrix}
L_{11} \\
L_{21}
\end{bmatrix}U_{11}
\]

\[
= \begin{bmatrix}
L_{11}U_{11} & L_{11}U_{12} \\
L_{21}U_{11} & A_{22}
\end{bmatrix}
\]

where have solved \(A_{12}=L_{11}U_{12}\)
using BLAS3 TRSM

\[
= \begin{bmatrix}
L_{11} & 0 \\
L_{21} & I
\end{bmatrix} \begin{bmatrix}
U_{11} & U_{12} \\
0 & A_{22}-L_{21}U_{12}
\end{bmatrix}
\]

Scher complement = S
computed using BLAS3 GEMM
proceed on S
Most work done in calls to
TRSM and GEMM, so should be fast
Often works well, but for some combinations
of n and cache size M, can reach \(\Omega(n^3/M)\)
Just as for matmul, there is a recursive, cache oblivious algorithm (Toledo, 1997)

**High Level Algorithm**

Do LU on left half of matrix
Update right half of matrix
(U at top, Schur compl. at bottom)

Do LU on Schur Complement

**function LL, U = RLUCA()** ... RLU = Recursive LU
... assume A nxm, nzm, m power of 2

if m=1 ... one column
pivot so $A_{11}$ largest, update rest of matrix
$L = A/A_{11}, \ U = A_{11}$

else
... write $A = [A_{11} \ A_{12}]$, $L_1 = [L_{11}]$

$A_{11}, A_{12}, L_{11}, U_1, U_2$ are $m/2 \times m/2$

$A_{21}, A_{22}, L_{12}$ are $n - \frac{m}{2} \times \frac{m}{2}$

$[L_1, U_1] = RLU ([A_{11}])$ ... LU of left half

Solve $A_{12} = L_{11} \cdot U_{12}$ for $U_{12}$ ... update $U$

$A_{22} = A_{22} - L_{21} \cdot U_{12}$ ... update Schur complement

$[L_2, U_2] = RLU (A_{22})$

$L = [L_1, [L_2]]$, $U = [U_1 \ U_2]^{n \times m}$
Correctness by induction
Recurrences for \( m=n \)
\[
A(n) = \text{#arith ops} = \frac{2}{3} n^3 + O(n^2)
\]
Similar recurrence
\[
W(n) = \text{# words moved} = O\left(\frac{n^3}{\log n}\right)
\]
RLU only hits lower bound for
\#
words moved, not \# messages
To minimize \# messages: either
1. Replace partial pivoting by
Tournament pivoting (discussed later, references in notes)
2. Keep GEPP, but more complicated
data structure; payoff unclear

How does Strassen etc extend?
Can modify RLU to run in \( O(n \log^2 n) \) flops
1. Multiply \( L_1, U_1 \) using Strassen
and
2. Solve \( A_{12} = L_{11} U_{12} \) as follows
\[
\begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}^{-1} = \begin{bmatrix} T_{11}^{-1} & -T_{11}^{-1} T_{12} T_{22}^{-1} \\ 0 & T_{22}^{-1} \end{bmatrix}
\]
perform all matmuls using Strassen
Slightly less numerically stable than GEMM

Where to find implementations
All blocked, unless marked recursive
Matlab: \( A \backslash b \), or \( [P,L,U]=lu(A) \)
          \( \text{cond, condest} \)
LAPACK: \( \text{xGETRF: GEPP where } x=S/D/C/Z \)
          \( \text{xGETRF2: GEPP recursively} \)
          \( \text{xGESV: solve } Ax=b \)
          \( \text{xGESVXX: condition estimation, iterative refinement with no extra precision} \)
          \( \text{xGESVXX: iterative refinement in extra precision} \)
          \( \text{xGECON: for condition estimation alone} \)
ScLAPACK: \( \text{pxGETRF etc} \)