2.4. Pivoting

- Reading: Trefethen and Bau (1997), Lecture 21

- The Gaussian factorization and backward substitution fail when \( u_{ii} = 0, \ i = 1 : n \)
  - The system need not be singular, e.g.,
    \[
    \begin{bmatrix}
    0 & 1 \\
    1 & 1 \\
    \end{bmatrix}
    \]
  - The factorization can proceed upon a row interchange
    * i.e., upon exchanging equations
  - Small divisors with finite-precision arithmetic will also cause problems

- Example 1. Consider three-decimal floating-point arithmetic \((\beta = 10, \ t = 3)\)
  \[
  \begin{bmatrix}
  1.00 \times 10^{-4} & 1.00 \\
  1.00 & 1.00 \\
  \end{bmatrix}
  \begin{bmatrix}
  x_1 \\
  x_2 \\
  \end{bmatrix}
  =
  \begin{bmatrix}
  1.00 \\
  2.00 \\
  \end{bmatrix}
  \]
  - The exact solution is \( x_1 = 10000/9999 = 1.00010 \) \( x_2 = 9998/9999 = 0.99990 \)
  - Factorization:
    \[
    L = \begin{bmatrix}
    1.00 & 0.00 \\
    1.00 \times 10^4 & 1.00 \\
    \end{bmatrix}
    \]
    \[
    U = \begin{bmatrix}
    1.00 \times 10^{-4} & 1.00 \\
    0.00 & -1.00 \times 10^4 \\
    \end{bmatrix}
    \]
The need for Pivoting

- Forward substitution

\[
\begin{bmatrix}
1.00 & 0.00 \\
1.00 \times 10^4 & 1.00
\end{bmatrix}
\begin{bmatrix}
y_1 \\
y_2
\end{bmatrix}
= 
\begin{bmatrix}
1.00 \\
2.00
\end{bmatrix}
\]

or \(y_1 = 1.00, y_2 = -1.00 \times 10^4\)

- Backward substitution

\[
\begin{bmatrix}
1.00 \times 10^{-4} & 1.00 \\
0.00 & -1.00 \times 10^4
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 
\begin{bmatrix}
1.00 \\
-1.00 \times 10^4
\end{bmatrix}
\]

- Thus, \(x_2 = 1.00, x_1 = 0.00\)

- This is awful!

- Interchanging rows

\[
\begin{bmatrix}
1.00 & 1.00 \\
0.00 & 1.00
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 
\begin{bmatrix}
2.00 \\
1.00
\end{bmatrix}
\]

- The system is in upper triangular form \((l_{21} = 0.00)\), so \(x_2 = 1.00\) and \(x_1 = 1.00\)

  * This is correct to three digits
Pivoting Strategies

- **Row pivoting** (partial pivoting): at stage $i$ of the outer loop of the factorization (cf. Section 2.3, p. 5)
  1. Find $r$ such that $|a_{ri}| = \max_{i \leq k \leq n} |a_{ki}|$
  2. Interchange rows $i$ and $r$

- **Column pivoting**: Proceed as row pivoting but interchange columns
  - Column pivoting requires reordering the unknowns
  - Column pivoting does not work well with direct factorization

- **Complete pivoting**: Choose $r$ and $c$ such that
  1. Find $r$, $c$ such that $|a_{rc}| = \max_{i \leq k, l \leq n} |a_{kl}|$
  2. Interchange rows $i$ and $r$ and columns $i$ and $c$

- Complete pivoting is less common than partial pivoting
  - Have to search a larger space
  - Have to reorder unknowns

- **Row pivoting is usually adequate**

- **Row, column, and complete pivoting** have $l_{ij} \leq 1$, $i \neq j$
Scaled Partial Pivoting

• The equations and unknowns may be scaled differently

• Example 2. Multiply the first row of Example 1 by $10^5$

$$
\begin{bmatrix}
1.00 \times 10^1 & 1.00 \times 10^5 \\
1.00 & 1.00
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 
\begin{bmatrix}
1.00 \times 10^5 \\
2.00
\end{bmatrix}
$$

– Row pivoting would choose the first row as a pivot
  * This yields the result $x_2 = 1.00$ and $x_1 = 0.00$

• There is no general solution to this problem

  – One strategy is to “equilibrate” the matrix
    * Select all elements to have the same magnitude

• Scaled partial pivoting:

  – Select row pivots relative to the size of the row

    1. Before factorization select scale factors

       $$
       s_i = \max_{1 \leq j \leq n} |a_{ij}|, \quad i = 1 : n
       $$

    2. At stage $i$ of the factorization, select $r$ such that

       $$
       \left| \frac{a_{ri}}{s_r} \right| = \max_{i \leq k \leq n} \left| \frac{a_{ki}}{s_k} \right|
       $$

    3. Interchange rows $k$ and $i$
Factorization with Pivoting

- Gaussian elimination with partial pivoting always finds factors $L$ and $U$ of a nonsingular matrix
  - Neglecting roundoff errors

- **Theorem 1**: For any $n \times n$ matrix $A$ of rank $n$, there is a reordering of rows such that
  \[
  PA = LU
  \]  
  (1)
  where $P$ is a permutation matrix that reorders the rows of $A$
  - A permutation matrix is an identity matrix with its rows or columns interchanged
  - *Proof*: cf. Golub and Van Loan (1996), Section 3.4.4
Scaled Partial Pivoting

- It is not necessary to store the permutation matrix or rearrange the rows of $A$
  - Row interchanges can be recorded in a vector $p$

*Example 3.* Consider

$$A = \begin{bmatrix}
1 & -1 & 2 \\
1 & -1 & 1 \\
2 & 3 & -1
\end{bmatrix}, \quad b = \begin{bmatrix}
2 \\
1 \\
4
\end{bmatrix}$$

- Compute the scale factors and initialize the interchange vector

$$s = \begin{bmatrix}
2 \\
1 \\
3
\end{bmatrix}, \quad p = \begin{bmatrix}
1 \\
2 \\
3
\end{bmatrix}$$

* $p_i = i, i = 1 : n$, implies no rows have been interchanged

- $i = 1$:
  * Scale factor: $|a_{11}/s_1| = 1/2, |a_{21}/s_2| = 1/1, |a_{31}/s_3| = 2/3$
  * The second row is the pivot

$$s = \begin{bmatrix}
2 \\
1 \\
3
\end{bmatrix}, \quad A \Rightarrow \begin{bmatrix}
1 & 0 & 1 \\
1 & -1 & 1 \\
2 & 5 & -3
\end{bmatrix}, \quad p = \begin{bmatrix}
2 \\
1 \\
3
\end{bmatrix}$$

- $p$ records the implicit row interchange
Scaled Partial Pivoting

- \( i = 2 \):
  
  * Scale factors: \(|a_{12}/s_1| = 0/2, |a_{32}/s_3| = 5/3\)
  
  * The third row is the pivot

\[
\mathbf{s} = \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}, \quad \mathbf{A} \Rightarrow \begin{bmatrix} 1 & 0 & 1 \\ 1 & -1 & 1 \\ 2 & 5 & -3 \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}
\]

- Construct \( \mathbf{P}, \mathbf{L}, \) and \( \mathbf{U} \)

  * \( p_1 = 2 \), so Row 1 of \( \mathbf{L} \) and \( \mathbf{U} \) is Row 2 of \( \mathbf{A} \)
  
  * \( p_2 = 3 \), so Row 2 of \( \mathbf{L} \) and \( \mathbf{U} \) is Row 3 of \( \mathbf{A} \)
  
  * \( p_3 = 1 \), so Row 3 of \( \mathbf{L} \) and \( \mathbf{U} \) is Row 1 of \( \mathbf{A} \)

\[
\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & -1 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 1 \end{bmatrix}
\]

  * From \( \mathbf{p} \), Row 1 of \( \mathbf{P} \) is Row 2 of the identity matrix
  
  * Row 2 of \( \mathbf{P} \) is Row 3 of the identity matrix
  
  * Row 3 of \( \mathbf{P} \) is Row 1 of the identity matrix

\[
\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}
\]
Scaled Partial Pivoting

• Check that \( PA = LU \)

\[
PA = \begin{bmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
1 & -1 & 2 \\
1 & -1 & 1 \\
2 & 3 & -1
\end{bmatrix} = 
\]

\[
LU = \begin{bmatrix}
1 & 0 & 0 \\
2 & 1 & 0 \\
1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & -1 & 1 \\
0 & 5 & -3 \\
0 & 0 & 1
\end{bmatrix} = 
\]

• Forward and backward substitution.

\( PAx = Pb \)

- Use (1)

\( LUx = Pb \)

• Forward substitution: \( Ly = Pb \)

- \( p_1 = 2 \), so \( y_1 = b_2 = 1 \) or \( y_1 = 1 \)
- \( p_2 = 3 \), so \( 2y_1 + y_2 = b_3 = 4 \) or \( y_2 = 2 \)
- \( p_3 = 1 \), so \( y_1 + y_3 = b_1 = 2 \) or \( y_3 = 1 \)

• Backward substitution: \( Ux = y \)

- \( p_3 = 1 \), so \( x_3 = y_3 = 1 \) or \( x_3 = 1 \)
- \( p_2 = 3 \), so \( 5x_2 - 3x_3 = y_2 = 2 \) or \( x_2 = 1 \)
- \( p_1 = 2 \), so \( x_1 - x_2 + x_3 = y_1 = 1 \) or \( x_1 = 1 \)
LU Factorization

function [Ap] = plufactor(A)
% plufactor: Factor the n-by-n matrix A into LU. On return, L - I
% is stored in the lower triangular part of A and U
% is stored in the upper triangular part. The vector p
% stores the permuted row indices using scaled partial pivoting.

[n n] = size(A);
% Initialize p and compute the scale vector s
for i = 1; n
    s(i) = norm(A(i,1:n), inf);
    p(i) = i;
end
% Loop over the rows
for i = 1: n - 1
% Find the best pivot row
    colmax = 0;
    for k = i: n
        srow = abs(A(p(k),i))/s(p(k));
        if colmax < srow;
            colmax = srow;
            index = k;
        end
    end
    temp = p(i);
    p(i) = index;
    p(index) = temp;
% Calculate the i th column of L
    for j = i + 1: n
        for k = 1: i - 1
            A(p(j),i) = A(p(j),i) - A(p(j),k)*A(p(k),i);
        end
        A(p(j),i) = A(p(j),i)/A(p(i),i);
    end
% Calculate the (i + 1) th row of U
    for j = i+1: n
        for k = 1: i
            A(p(i+1),j) = A(p(i+1),j) - A(p(i+1),k)*A(p(k),j);
        end
    end
end
end
Forward and Backward Substitution

function y = pforward(L, b, p)
% pforward: Solution of a n-by-n lower triangular system
% Ly = Pb by forward substitution. Row permutations have been
% stored in the vector p.

    [n n] = size(L);
    y(1) = b(p(1));
    for i = 2:n
        y(i) = b(p(i)) - dot(L(p(i),1:i-1)', y(1:i-1));
    end

function x = backward(U, y, p)
% pbackward: Solution of a n-by-n upper triangular system
% Ux = y by backward substitution. Row permutations have been
% stored in the vector p.

    [n n] = size(U);
    x(n) = y(n)/U(p(n),n);
    for i = n - 1: -1: 1
        x(i) = (y(i) - dot(U(p(i),i+1:n)', x(i+1:n)))/U(p(i),i);
    end

• Note:

  i. No attempt has been made to check for failure
     - A is singular if colmax = 0 or s_i = 0 for some i
  ii. The MATLAB function norm computes vector and matrix norms.