Computational Methods
CMSC/AMSC/MAPL 460

Vectors, Matrices, Linear Systems, LU Decomposition,

Ramani Duraiswami,
Dept. of Computer Science
Easy systems to solve

- Diagonal system
- Triangular system
- On board and then matlab

- Cost of diagonal solve is $O(n)$

```matlab
x=zeros(n,1)
for k=1:n
    x(k)=b(k)/A(k,k)
end
```
Solving a triangular system

\[ x = \text{zeros}(n,1); \]
\[ \text{for } k = n:-1:1 \]
\[ x(k) = b(k)/U(k,k); \]
\[ i = (1:k-1)'; \]
\[ b(i) = b(i) - x(k)*U(i,k); \]
\[ \text{end} \]

\[ x = \text{zeros}(n,1); \]
\[ \text{for } k = n:-1:1 \]
\[ j = k+1:n; \]
\[ x(k) = (b(k) - U(k,j)*x(j))/U(k,k); \]
\[ \text{end} \]

Cost of solving a triangular system
Loop of size \( n \). Each loop has a cost of \( k \) (or \( n-k \))
So total cost is
\[ n*1 + n*2 + \ldots + n*n = n^2 \]
Gaussian Elimination

- Zero elements of first column below 1st row

  \[
  \begin{pmatrix}
  10 & -7 & 0 \\
  -3 & 2 & 6 \\
  5 & -1 & 5 \\
  \end{pmatrix}
  \begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \end{pmatrix}
  =
  \begin{pmatrix}
  7 \\
  4 \\
  6 \\
  \end{pmatrix}
  \]

  - multiplying 1st row by 0.3 and add to 2nd row

  \[
  \begin{pmatrix}
  10 & -7 & 0 \\
  0 & -0.1 & 6 \\
  0 & 2.5 & 5 \\
  \end{pmatrix}
  \begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \end{pmatrix}
  =
  \begin{pmatrix}
  7 \\
  6.1 \\
  2.5 \\
  \end{pmatrix}
  \]

  - Results in

  \[
  \begin{pmatrix}
  10 & -7 & 0 \\
  0 & 2.5 & 5 \\
  0 & -0.1 & 6 \\
  \end{pmatrix}
  \begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \end{pmatrix}
  =
  \begin{pmatrix}
  7 \\
  2.5 \\
  6.1 \\
  \end{pmatrix}
  \]

- Zero elements of first column below 2nd row

  - Swap rows

  \[
  \begin{pmatrix}
  10 & -7 & 0 \\
  0 & 2.5 & 5 \\
  0 & 0 & 6.2 \\
  \end{pmatrix}
  \begin{pmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  \end{pmatrix}
  =
  \begin{pmatrix}
  7 \\
  2.5 \\
  6.2 \\
  \end{pmatrix}
  \]
Representing linear systems as matrix-vector equations

\[10x_1 - 7x_2 = 7\]
\[-3x_1 + 2x_2 + 6x_3 = 4\]
\[5x_1 - x_2 + 5x_3 = 6\]

- Represent it as a matrix-vector equation (linear system)
- We will apply the familiar elimination technique, and then see its matrix equivalent

\[
\begin{pmatrix}
10 & -7 & 0 \\
-3 & 2 & 6 \\
5 & -1 & 5
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
=
\begin{pmatrix}
7 \\
4 \\
6
\end{pmatrix}
\]
Solution

- Start from last equation which can be solved by division
- Next substitute in the previous line and continue
- This describes the way to do the algorithm by hand
- How to represent it using matrices?
- Also, how do we solve another system that has the same matrix?
  - Upper triangular matrix we end up with will be the same, but the sequence of operations on the r.h.s needs to be repeated

\[
\begin{align*}
6.2x_3 &= 6.2 \\
2.5x_2 + (5)(1) &= 2.5. \\
10x_1 + (-7)(-1) &= 7 \\
x &= \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}
\]
Gaussian Elimination: LU Matrix decomposition

• It turns out that Gaussian elimination corresponds to a particular matrix decomposition …
  – Product of permutation, lower triangular and upper triangular matrices

• What is a permutation matrix?
  – It rearranges a system of equations and changes the order.
  – Multiplying by it swaps the order of rows in a matrix
  – Essentially a rearrangement of the identity
  – Nice property: transpose is its inverse: \( PP^T = I \)

\[
P = \begin{pmatrix}
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0
\end{pmatrix}
\]

\[
Px = b
\]

\[
x = P^Tb
\]
LU Decomposition

• What is an upper triangular matrix?
  – Elements below diagonal are zero
  \[ U = \begin{pmatrix}
  1 & 2 & 3 & 4 \\
  0 & 5 & 6 & 7 \\
  0 & 0 & 8 & 9 \\
  0 & 0 & 0 & 10
\end{pmatrix} \]
• Lower triangular matrix
• Elements above diagonal are zero
• Unit lower triangular matrix
• Elements along diagonal are one
• Upper triangular part of Gauss Elimination is clear …
  – final matrix we end up with
• What about lower triangular and permutation?
\[ LU = PA \]

\[ L = \begin{pmatrix}
1 & 0 & 0 \\
0.5 & 1 & 0 \\
-0.3 & -0.04 & 1
\end{pmatrix} \quad U = \begin{pmatrix}
10 & -7 & 0 \\
0 & 2.5 & 5 \\
0 & 0 & 6.2
\end{pmatrix} \quad P = \begin{pmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0
\end{pmatrix} \]

- Identify the elements of \( L \) and \( P \)?
- \( L \) has the multipliers we used in the elimination steps.
- \( P \) has a record of the row swaps we did to avoid dividing by small numbers.
- In fact we can write each step of Gaussian elimination in matrix form:
  \[ U = M_{n-1} P_{n-1} \cdots M_2 P_2 M_1 P_1 A \]
  \[ L_1 L_2 \cdots L_{n-1} U = P_{n-1} \cdots P_2 P_1 A \]
\[
A = \begin{pmatrix}
10 & -7 & 0 \\
-3 & 2 & 6 \\
5 & -1 & 5 \\
\end{pmatrix}
\quad L = L_1L_2\cdots L_{n-1} \\
\quad P = P_{n-1}\cdots P_2P_1
\]

the matrices defined during the elimination are

\[
P_1 = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{pmatrix},
\quad M_1 = \begin{pmatrix}
1 & 0 & 0 \\
0.3 & 1 & 0 \\
-0.5 & 0 & 1 \\
\end{pmatrix},
\]

\[
P_2 = \begin{pmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
\end{pmatrix},
\quad M_2 = \begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0.04 & 1 \\
\end{pmatrix},
\]
\[ A = \begin{pmatrix} 10 & -7 & 0 \\ -3 & 2 & 6 \\ 5 & -1 & 5 \end{pmatrix} \]

\[ P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0.3 & 1 & 0 \\ -0.5 & 0 & 1 \end{pmatrix}, \]

\[ P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.04 & 1 \end{pmatrix}, \]

\[ L_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 1 & 0 \\ -0.3 & 0 & 1 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -0.04 & 1 \end{pmatrix}, \]
Solving a system with the LU decomposition

\[ Ax = b \]
\[ LU = PA \]
\[ P^T LUx = b \]
\[ L[Ux] = Pb \]
Solve \[ Ly = Pb \]
Then \[ Ux = y \]
Solving a system with the LU decomposition

\[ Ax=b \]

\[ LU = PA \]

\[ P^T LUx = b \]

\[ L[Ux] = Pb \]

Solve \( Ly = Pb \)

Then \( Ux = y \)
Look at LU code

```matlab
%LU Triangular factorization
%    [L,U,p] = lutx(A) produces a unit lower triangular
%    matrix L, an upper triangular matrix U, and a
%    permutation vector p, so that L*U = A(p,:).

[n,n] = size(A);
p = (1:n);

for k = 1:n-1
    % Find largest element below diagonal in k-th column
    [r,m] = max(abs(A(k:n,k)));
m = m+k-1;

    % Skip elimination if column is zero
    if (A(m,k) == 0)
        % Swap pivot row
        if (m ~= k)
            A([k m], :) = A([m k], :);
p([k m]) = p([m k]);
        end
    end
```

- Initialize
  - Matrix size
  - Permutation vector
- Second output argument to `max` is index of max element
- If max element is zero then we need not eliminate
- Exchange rows
- Update permutation vector