Using the $\mathbf{PA} = \mathbf{LU}$ Factorization to Solve $\mathbf{Ax} = \mathbf{b}$

Susanne Brenner and Li-Yeng Sung
(modified by Douglas B. Meade)

Department of Mathematics

Overview

In this week's lab you will create an M-file to carry out the three steps in the solution of $\mathbf{Ax} = \mathbf{b}$ using Gaussian elimination with partial pivoting, i.e., the $\mathbf{PA} = \mathbf{LU}$ factorization.

New commands introduced in this lab include `disp`, `fprintf`, `flops`, `rand`, and `lu`.

Part I

The purpose of this part of the lab is to introduce `MATLAB` commands used to augment the matrix output with meaningful (and attractively formatted) textual output.

In Part II you will be asked to develop an M-file showing the major steps in the use of $\mathbf{PA} = \mathbf{LU}$ to solve $\mathbf{Ax} = \mathbf{b}$. Keep this in mind as you work through the separate steps in Part I.

Displaying Formatted Text

The `disp` command displays a string in the Maple window. Try the following:

```
≫ disp('This is Lab 4 in Math 526 (Fall 2005).')
```

Note that strings are surrounded by single quotes. The `MATLAB` command `fprintf` is used for the purpose of displaying text that includes values — numerical or string — from a variable.

```
≫ m = 4;
≫ s = 'Fall';
≫ y = 2005;
≫ fprintf('This is Lab %2g for Math 526 (%s %g).\n', m, s, y)
```

Use `help fprintf` to obtain more information about `fprintf`.
What does the `\n` *do? What happens if it is omitted?*

Counting Operations with `flops`

The `MATLAB` command `flops` counts the number of floating-point operations (additions, subtractions, multiplications, and divisions) performed by `MATLAB`. (See `help flops` for detailed instructions on the use of `flops`.

To illustrate the use of `flops` — and the importance of the order in which matrices are multiplied, let $\mathbf{A}$ be the 1×10 matrix of ones, $\mathbf{B}$ be the 10×1 matrix of ones, and $\mathbf{C}$ the 1×10 matrix of ones:

```
≫ A = ones(1,10);
≫ B = ones(10,1);
≫ C = ones(1,10);
```

Even though we know that $\mathbf{ABC} = \mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$ the amount of work needed to compute this number does depend on the order in which the terms are multiplied. (Note that $\mathbf{ABC}$ is a 1×1 matrix, i.e., a scalar).

≫ **flops(0)**	**% reset the count of floating-point operations to zero**
≫ **(A*B)*C**	**% compute product of three matrices**
≫ **flops**	**% number of flops since last reset**
≫ **flops(0)**	**% reset flops counter to zero**
≫ **A*(B*C)**	**% compute product of three matrices**
≫ **flops**	**% number of flops since last reset**

Which computation is more efficient for this problem?
How does this result compare with #3-6 on page 3-10 of the text?

Random Matrices
The `rand` command generates matrices (or vectors) with randomly selected components. Each component is selected from a uniform distribution on the interval $(0.0, 1.0)$. For example, a random 2×5 matrix can be obtained with

≫ **rand(2,5)**

Repeat the comparison in the computation of $\mathbf{ABC}$ when the entries in $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are random numbers between 0 and 1. Does the numerical value of the components of the matrices affect the computational cost?

Gaussian Elimination with Partial Pivoting
`MATLAB`'s `lu` command performs Gaussian elimination with partial pivoting on a matrix $\mathbf{A}$ and returns the matrices $\mathbf{L}$, $\mathbf{U}$, and $\mathbf{P}$ That is, $\mathbf{L}$ is a lower triangular matrix (with ones on the main diagonal), $\mathbf{U}$ is an upper triangular matrix, and $\mathbf{P}$ is a permutation matrix such that $\mathbf{PA} = \mathbf{LU}$.

Create the matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \\ 1 & 4 & 9 \end{bmatrix}$$

then execute the command:

≫ **[L,U,P] = lu(A)**

Use `MATLAB` to verify that $\mathbf{PA} = \mathbf{LU}$.

Clear all variables before you begin to work on Part II.