Solving Triangular Systems

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

Department of Mathematics

Overview

The goals of this week's lab are to implement both forward and backward substitution (i.e., solving triangular systems) in `MATLAB`.

Part I

In this part an implementation of the forward substitution method for solving a lower triangular system is developed. While this discussion addresses only 4×4 systems, you should be thinking about the changes required for a general $n \times n$ triangular system.

- Enter Problem
  Define the following lower triangular matrix $\mathbf{A}$ and vector $\mathbf{b}$:

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 4 & 5 & 2 & 0 \\ 3 & -2 & 1 & 1 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ 6 \\ 4 \end{bmatrix}.$$

- Explicit Solution by Forward Substitution

```
>> x(1)=b(1)/A(1,1)                          % solution to a_{1,1}x_1 = b_1
>> x(2)=(b(2)-A(2,1)*x(1))/A(2,2)            % solution to a_{2,1}x_1 + a_{2,2}x_2 = b_2
>> x(3)=(b(3)-A(3,1)*x(1)-A(3,2)*x(2))/A(3,3)
>> x(4)=(b(4)-A(4,1)*x(1)-A(4,2)*x(2)-A(4,3)*x(3))/A(4,4)
>> x = x'                                    % convert row vector to column vector
```

- Forward Substitution using Matrix Multiplication
  Note that the product `A(4,1)*x(1)+A(4,2)*x(2)+A(4,3)*x(3)` can be written as the matrix product `A(4,1:3)*x(1:3)'`. (See Lab 1 if you have any questions about the syntax used in this command.)

  The implementation of forward substitution can be rewritten as:

```
>> clear x                                   % remove x from MATLAB's memory
>> x(1) = b(1)/A(1,1)
>> x(2) = (b(2)-A(2,1:1)*x(1:1)')/A(2,2)
>> x(3) = (b(3)-A(3,1:2)*x(1:2)')/A(3,3)
>> x(4) = (b(4)-A(4,1:3)*x(1:3)')/A(4,4)
```

  NOTE: It is not necessary to type each command in its entirety. For example, after making the assignment to `x(2)`, press the up arrow key (once) and then edit the previous command (the assignment to `x(2)`) — a total of seven (7) changes are needed.

- Forward Substitution with a `for` Loop
  For a large system it would be very tedious to compute **x** with a separate command for each component. The `for` statement can be used to simplify the coding. (See `help for` for detailed information on the syntax of the `for` statement.)

  >> clear x **% remove x from MATLAB's memory**
  >> x(1) = b(1)/A(1,1)
  >> for i = 2:1:4, **% i starts at 2, end at 4, w/ steps of 1**
  >> x(i) = (b(i)-A(i,1:i-1)*x(1:i-1)′)/A(i,i);
  >> end
  >> x = x′ **convert and display result as column**

- Forward Substitution for 4×4 Lower Triangular Systems with an M-file
  Notice that the commands used in the loop can be used to solve any 4×4 lower triangular system (with non-zero components on the diagonal). Instead of typing the same commands for every 4×4 lower triangular system, the commands can be placed in an M-file that can be executed for specific choices of the matrix **A** and vector **b**.

  To create a new M-file either navigate the menus to **File** → **New** → **M-file** or click on the new document icon (at the far left end of tool bar). Once you have an empty M-file in the MATLAB Editor/Debugger, enter the following lines:

```
%FORWARD Forward substitution for 4x4 lower triangular systems
%        Written by <YOUR NAME> on <TODAY'S DATE>
function x = forward( A, b )
x(1) = b(1)/A(1,1);
for i = 2:1:4,        % i starts at 2, end at 4, w/ steps of 1
   x(i) = (b(i)-A(i,1:i-1)*x(1:i-1)')/A(i,i);
end
x = x';               % convert and display result as column
```

  Save this file as `forward.m`.

  Return to the MATLAB Command Window and enter the following commands:

  >> clear x
  >> x = forward(A, b)

- Forward Substitution for Lower Triangular Systems of Any Size
  The only part of the code in `forward.m` that restricts it to 4×4 systems is the upper limit of the index in the `for` statement. The built-in `size` command can be used to make the implementation work for any square lower triangular system (with non-zero components on the diagonal).

  >> [m,n] = size(A) **% # rows and columns of matrix A**
  >> [m,n] = size(b) **% # rows and columns of vector b**

  To increase the utility of this M-file, modify `forward.m` as follows:

```
%FORWARD Forward substitution for lower triangular systems
%        Written by <YOUR NAME> on <TODAY'S DATE>
function x = forward( A, b )
[m,n] = size(A);
if m ~= n,            % check if # rows ~= # columns in A
```

```
   error('The input matrix, A, is not a square matrix.')
end
x(1) = b(1)/A(1,1);
for i = 2:1:m,        % i starts at 2, end at 4, w/ steps of 1
   x(i) = (b(i)-A(i,1:i-1)*x(1:i-1)')/A(i,i);
end
x = x';               % convert and display result as column
```

Return focus to the MATLAB Command Window and test your implementation by finding the solution to $\mathbf{Ax} = \mathbf{b}$:

>> x = forward(A, b)

Next, let $\mathbf{B}$ be the 3×4 matrix formed by the first three rows of $\mathbf{A}$ and attempt to solve $\mathbf{Bx} = \mathbf{b}$. Execute the commands:

>> B = A(1:3,:);
>> x = forward(B,b) **% explain this result**

Use `help if` and `help error` to learn more about the `if` and `error` statements. What modifications to `forward.m` are needed to detect incompatibilities between the size of $\mathbf{A}$ and $\mathbf{b}$?

Lastly, execute the two commands:

>> type forward
>> help forward **% see also `type help.m`**

Explain what each of these commands returns. (Do not forget to consult the on-line help for additional details.)