MATH 353
Engineering mathematics III

Instructor: Francisco-Javier 'Pancho' Sayas

Spring 2014 – University of Delaware

NOTIONS ABOUT LINEAR SYSTEMS

Problem

A great deal of computational time is spent solving linear systems

$$\mathrm{A}\mathbf{x} = \mathbf{b},$$

where A is a **square invertible** matrix. For what we are going to do, we only need to consider real matrices and right–hand sides.

- The matrix A is invertible if and only if $\det \mathrm{A} \neq 0$.
- **Do never compute** the determinant to see if the matrix is invertible! There are better ways. Many methods find out if the matrix is not invertible (we say singular) as they proceed. Invertible = non–singular.
- We often write $\mathbf{x} = \mathrm{A}^{-1}\mathbf{b}$ to denote the solution of $\mathrm{A}\mathbf{x} = \mathbf{b}$. **Please, do never, never, never invert a matrix** to solve a linear system. It is a huge waste of computational effort.

Again...

If you ever see the expression

$$\mathbf{x} = A^{-1}\mathbf{b}$$

in the middle of an algorithm or an explanation, it means

$$\text{solve } A\mathbf{x} = \mathbf{b}$$

and not find A^{-1} and then multiply by b.

What can MATLAB do for me?

The magic of backslash \

The expression

```
A \ b
```

can be used to solve the system $A\mathbf{x} = \mathbf{b}$. MATLAB does actually look at your matrix and tries to use the best available method for it.

- There will be situations when you will have to choose your own method.
- You might start by knowing more about your matrix
- Careful with almost singular matrices (matrices that should be singular but are not because of round–off error) and with cases where there's more than one solution. MATLAB tries to give an answer always, sometimes with a warning.

SIMPLE SYSTEMS

The no–brainers

1. Diagonal systems (almost nothing to do!)
2. Permutation matrices (almost nothing to do!)
3. Lower triangular systems (substitution)
4. Upper triangular systems (back–substitution)
5. Orthogonal systems

Diagonal systems

Aim: Solve D**x** = **b** where D is diagonal

$$\mathrm{D} = \begin{bmatrix} d_{11} & & \\ & \ddots & \\ & & d_{NN} \end{bmatrix}$$

If your diagonal matrix D is fully stored (with all its zeros), you only need to do

$$x_i = b_i / d_{ii}.$$

If your diagonal matrix is stored as a vector (you only store the diagonal elements), it is even simpler.

```
x= b./d      % d = vector with diagonal elements of D
```

Permutation matrices

A permutation matrix P is the matrix obtained by reordering the rows of the identity matrix. Left–multiplication by a permutation matrix produces the same effect on the vector.

Example: consider the permutation $(2, 4, 1, 3)$ of the numbers from 1 to 4. We apply this permutation to the rows of the identity matrix:

$$\mathrm{P} = \left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 \end{array}\right]$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_4 \\ x_1 \\ x_3 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 1 \\ 3 \end{bmatrix}$$

Permutation matrices (2)

Permutation matrices satisfy the following property $P^{-1} = P^\top$.

However, solving systems with permutation matrices is even cheaper than multiplying by the transpose. You need the vector $\sigma(i)$ that gives you the permutation. The permutation matrix can be written as

$$p_{i,\sigma(i)} = 1, \qquad \text{and } p_{i,j} = 0 \text{ if } j \neq \sigma(i).$$

Solving $P\mathbf{x} = \mathbf{b}$ can be done in a simple loop

$$x_{\sigma(i)} = b_i, \qquad i = 1, \ldots, N$$

If the permutation is encoded in the vector `perm`, then we have yet another MATLAB one–liner: `x(perm)=b`.

Lower triangular systems (the idea)

Aim: solve a system $\mathrm{L}\mathbf{x} = \mathbf{b}$, where L is lower triangular and invertible (all its diagonal elements are non–zero)

$$\mathrm{L} = \begin{bmatrix} L_{11} & & & \\ L_{21} & L_{22} & & \\ \vdots & & \ddots & \\ L_{N1} & \dots & \dots & L_{NN} \end{bmatrix}$$

Lower triangular systems can be solved by forward substitution (solve one equation at a time, starting in the first one):

$$x_1 = b_1/L_{11}, \qquad x_i = \Big(b_i - \sum_{j=1}^{i-1} L_{ij}x_j\Big)/L_{ii}, \quad i = 2, \dots, N.$$

Lower triangular systems (the code)

We write

$$x_i = \Big(b_i - \sum_{j=1}^{i-1} L_{ij} x_j\Big) / L_{ii}, \quad i = 1, \ldots, N,$$

understading that $\sum_{j=1}^{0}$ is an empty loop or expression. Note how

$$\sum_{j=1}^{i-1} L_{ij} x_j = \begin{bmatrix} L_{i1} & \ldots & L_{i,i-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{i-1} \end{bmatrix}$$

```
n=length(b);
x=zeros(n,1);   % prepare room for the solution
                      % (column vector, like b)
for i=1:n
    x(i)=( b(i) - L(i,1:i-1)*x(1:i-1) )/L(i,i);
end
```

Upper triangular systems

Systems U$\mathbf{x} = \mathbf{b}$ with U upper triangular invertible

$$\mathrm{U} = \begin{bmatrix} U_{11} & U_{12} & \dots & U_{1N} \\ & U_{22} & & \vdots \\ & & \ddots & \vdots \\ & & & U_{NN} \end{bmatrix}$$

can be solved by back substitution

$$x_i = \Big(b_i - \sum_{j=i+1}^{N} U_{ij} x_j\Big) / U_{ii}, \qquad i = N, N-1, \dots, 1.$$

(The summation sign $\sum_{j=N+1}^{N}$ has to be understood as empty.)

Upper triangular systems (the code)

$$x_i = \Big(b_i - \sum_{j=i+1}^{N} U_{ij} x_j \Big) / U_{ii}, \qquad i = N, N-1, \ldots, 1.$$

```
n=length(b);
x=zeros(n,1);   % prepare room for the solution
                   % (column vector, like b)
for i=n:-1:1
    x(i)=( b(i) - U(i,i+1:n)*x(i+1:n) )/U(i,i);
end
```

Systems with orthogonal matrices

In some cases, we will want to solve

$$\mathsf{Q}\mathbf{x} = \mathbf{b}$$

where Q is an orthogonal matrix (that is, a matrix such that $\mathsf{Q}^\top \mathsf{Q} = \mathrm{I}$, where I is the identity matrix). In this case, we solve the system by premultiplying with the transpose of the matrix

$$\mathsf{Q}^\top \mathsf{Q}\mathbf{x} = \mathsf{Q}^\top \mathbf{b} \qquad \Longrightarrow \qquad \mathbf{x} = \mathsf{Q}^\top \mathbf{b}.$$