Lecture 9

Introduction to Linear Systems

How linear systems occur

Linear systems of equations naturally occur in many places in engineering, such as structural analysis, dynamics and electric circuits. Computers have made it possible to quickly and accurately solve larger and larger systems of equations. Not only has this allowed engineers to handle more and more complex problems where linear systems naturally occur, but has also prompted engineers to use linear systems to solve problems where they do not naturally occur such as thermodynamics, internal stress-strain analysis, fluids and chemical processes. It has become standard practice in many areas to analyze a problem by transforming it into a linear systems of equations and then solving those equation by computer. In this way, computers have made linear systems of equations the most frequently used tool in modern engineering.

In Figure 9.1 we show a truss with equilateral triangles. In Statics you may use the "method of joints" to write equations for each node of the truss[1]. This set of equations is an example of a linear system. Making the approximation $\sqrt{3}/2 \approx .8660$, the equations for this truss are

$$\begin{aligned}
.5\,T_1 + T_2 &= R_1 = f_1 \\
.866\,T_1 &= -R_2 = -.433\,f_1 - .5\,f_2 \\
-.5\,T_1 + .5\,T_3 + T_4 &= -f_1 \\
.866\,T_1 + .866\,T_3 &= 0 \\
-T_2 - .5\,T_3 + .5\,T_5 + T_6 &= 0 \\
.866\,T_3 + .866\,T_5 &= f_2 \\
-T_4 - .5\,T_5 + .5\,T_7 &= 0,
\end{aligned} \tag{9.1}$$

where T_i represents the tension in the i-th member of the truss.

You could solve this system by hand with a little time and patience; systematically eliminating variables and substituting. Obviously, it would be a lot better to put the equations on a computer and let the computer solve it. In the next few lectures we will learn how to use a computer effectively to solve linear systems. The first key to dealing with linear systems is to realize that they are equivalent to matrices, which contain numbers, not variables.

As we discuss various aspects of matrices, we wish to keep in mind that the matrices that come up in engineering systems are *really large*. It is not unusual in real engineering to use matrices whose dimensions are in the thousands! It is frequently the case that a method that is fine for a 2×2 or 3×3 matrix is totally inappropriate for a 2000×2000 matrix. We thus want to emphasize methods that work for large matrices.

[1] See http://en.wikipedia.org/wiki/Truss or http://en.wikibooks.org/wiki/Statics for reference.

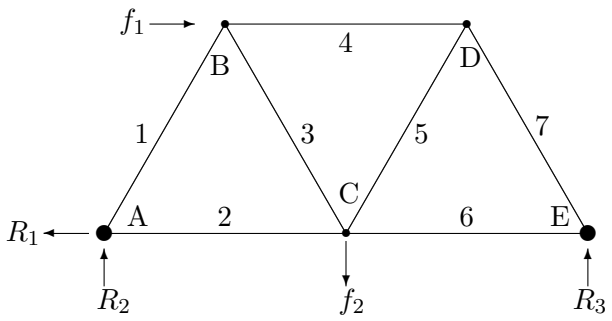

Figure 9.1: An equilateral truss. Joints or nodes are labeled alphabetically, A, B, ... and Members (edges) are labeled numerically: 1, 2, ... The forces f_1 and f_2 are applied loads and R_1, R_2 and R_3 are reaction forces applied by the supports.

Linear systems are equivalent to matrix equations

The system of linear equations

$$\begin{aligned} x_1 - 2x_2 + 3x_3 &= 4 \\ 2x_1 - 5x_2 + 12x_3 &= 15 \\ 2x_2 - 10x_3 &= -10 \end{aligned}$$

is equivalent to the matrix equation

$$\begin{pmatrix} 1 & -2 & 3 \\ 2 & -5 & 12 \\ 0 & 2 & -10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 15 \\ -10 \end{pmatrix},$$

which is equivalent to the *augmented matrix*

$$\left(\begin{array}{ccc|c} 1 & -2 & 3 & 4 \\ 2 & -5 & 12 & 15 \\ 0 & 2 & -10 & -10 \end{array}\right).$$

The advantage of the augmented matrix, is that it contains only numbers, not variables. The reason this is better is because computers are much better in dealing with numbers than variables. To solve this system, the main steps are called *Gaussian elimination* and *back substitution*.

The augmented matrix for the equilateral truss equations (9.1) is given by

$$\left(\begin{array}{ccccccc|c} .5 & 1 & 0 & 0 & 0 & 0 & 0 & f_1 \\ .866 & 0 & 0 & 0 & 0 & 0 & 0 & -.433f_1 - .5\,f_2 \\ -.5 & 0 & .5 & 1 & 0 & 0 & 0 & -f_1 \\ .866 & 0 & .866 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -.5 & 0 & .5 & 1 & 0 & 0 \\ 0 & 0 & .866 & 0 & .866 & 0 & 0 & f_2 \\ 0 & 0 & 0 & -1 & -.5 & 0 & .5 & 0 \end{array}\right). \tag{9.2}$$

Notice that a lot of the entries are 0. Matrices like this, called *sparse*, are common in applications and there are methods specifically designed to efficiently handle sparse matrices.

Triangular matrices and back substitution

Consider a linear system whose augmented matrix happens to be

$$\left(\begin{array}{ccc|c} 1 & -2 & 3 & 4 \\ 0 & -1 & 6 & 7 \\ 0 & 0 & 2 & 4 \end{array}\right). \tag{9.3}$$

Recall that each row represents an equation and each column a variable. The last row represents the equation $2x_3 = 4$. The equation is easily solved, i.e. $x_3 = 2$. The second row represents the equation $-x_2 + 6x_3 = 7$, but since we know $x_3 = 2$, this simplifies to: $-x_2 + 12 = 7$. This is easily solved, giving $x_2 = 5$. Finally, since we know x_2 and x_3, the first row simplifies to: $x_1 - 10 + 6 = 4$. Thus we have $x_1 = 8$ and so we know the whole solution vector: $\mathbf{x} = \langle 8, 5, 2\rangle$. The process we just did is called *back substitution*, which is both efficient and easily programmed. The property that made it possible to solve the system so easily is that A in this case is *upper triangular*. In the next section we show an efficient way to transform an augmented matrix into an upper triangular matrix.

Gaussian Elimination

Consider the matrix

$$A = \left(\begin{array}{ccc|c} 1 & -2 & 3 & 4 \\ 2 & -5 & 12 & 15 \\ 0 & 2 & -10 & -10 \end{array}\right).$$

The first step of Gaussian elimination is to get rid of the 2 in the (2,1) position by subtracting 2 times the first row from the second row, i.e. (new 2nd = old 2nd - (2) 1st). We can do this because it is essentially the same as adding equations, which is a valid algebraic operation. This leads to

$$\left(\begin{array}{ccc|c} 1 & -2 & 3 & 4 \\ 0 & -1 & 6 & 7 \\ 0 & 2 & -10 & -10 \end{array}\right).$$

There is already a zero in the lower left corner, so we don't need to eliminate anything there. To eliminate the third row, second column, we need to subtract -2 times the second row from the third row, (new 3rd = old 3rd - (-2) 2nd), to obtain

$$\left(\begin{array}{ccc|c} 1 & -2 & 3 & 4 \\ 0 & -1 & 6 & 7 \\ 0 & 0 & 2 & 4 \end{array}\right).$$

This is now just exactly the matrix in equation (9.3), which we can now solve by back substitution.

Matlab's matrix solve command

In MATLAB the standard way to solve a system $A\mathbf{x} = \mathbf{b}$ is by the command

```
>> x = A \ b
```

This syntax is meant to suggest dividing by A from the left as in

$$A\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad A\backslash A\mathbf{x} = A\backslash\mathbf{b} \quad \Leftrightarrow \quad \mathbf{x} = A\backslash\mathbf{b}\,.$$

Such division is not meaningful mathematically, but it helps for remembering the syntax.

This command carries out Gaussian elimination and back substitution. We can do the above computations as follows:

```
≫ A = [1 -2 3 ; 2 -5 12 ; 0 2 -10]
≫ b = [4 15 -10]'
≫ x = A \ b
```

Next, use the MATLAB commands above to solve $A\mathbf{x} = \mathbf{b}$ when the augmented matrix for the system is

$$\left(\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{array}\right),$$

by entering

```
≫ x1 = A \ b
```

Check the result by entering

```
≫ A*x1 - b
```

You will see that the resulting answer satisfies the equation exactly. Next try solving using the inverse of A:

```
≫ x2 = inv(A)*b
```

This answer can be seen to be inaccurate by checking

```
≫ A*x2 - b
```

Thus we see one of the reasons why the inverse is never used for actual computations, only for theory.

Exercises

9.1 Set $f_1 = 1000N$ and $f_2 = 5000N$ in the equations (9.1) for the equailateral truss. Input the coefficient matrix A and the right hand side vector b in (9.2) into MATLAB. Solve the system using the command `\` to find the tension in each member of the truss. Save the matrix A as `A_equil_truss` and keep it for later use. (Enter `save A_equil_truss A`.) Print out and turn in A, $\mathbf{b}$ and the solution $\mathbf{x}$.

9.2 Write each system of equations as an augmented matrix, then find the solutions using *Gaussian elimination and back substitution* (the algorithm in this chapter). Check your solutions using MATLAB.

(a)

$$\begin{aligned} x_1 + x_2 &= 2 \\ 4x_1 + 5x_2 &= 10 \end{aligned}$$

(b)

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &= -1 \\ 4x_1 + 7x_2 + 14x_3 &= 3 \\ x_1 + 4x_2 + 4x_3 &= 1 \end{aligned}$$