

Mini–Project on Solving Large Linear Systems

Richard S. Laugesen

Goal of the project

A system of two linear equations in two unknowns can be easily solved by hand. But for linear equations with more unknowns, the calculations are better done by computer. This mini–project explains how to solve such systems of equations numerically, using Matlab or Octave commands.

The connection to Differential Equations is as follows. Suppose you know the general solution of an n th order linear differential equation. Then you apply the initial conditions, yielding a system of equations for the unknown constants c_1, \dots, c_n . To evaluate these constants, you need to solve the system of equations.

Instructions

Turn in your written solution to the problem, together with a printout of the Matlab or Octave calculations that you used.

Technical Notes.

If you are using Matlab, you can print out the calculations as follows. First select (or “highlight”) all your calculations using the mouse, then choose the **File->Print Selection** menu item in the main Matlab window.

If you are using Octave, you will need to copy and paste the calculations from the Octave window into a text file, say using Notepad or Word (on a PC) or using Pico or Emacs (on a Unix system). On a Unix system, you can create a suitable text file called `calc` by typing `pico calc` at a Unix prompt (not at an Octave prompt). Then just copy and paste the calculations from the Octave window into the Pico window. Lastly, save the file from Pico, and print it as usual with `lpr calc`. Ask the lab assistant if you have any trouble with the copying and pasting (Unix is different from Windows).

1 Linear Systems of Equations in Matlab and Octave

When we solve an n^{th} order linear differential equation, we typically end up with a general solution that contains n arbitrary constants c_1, \dots, c_n . If we apply the initial conditions, then we obtain n linear equations involving c_1, \dots, c_n . It is not hard to solve these equations by hand when $n = 2$ or $n = 3$. But for larger values of n , it makes sense to use a computer. Here is an example of how to do it in Matlab or Octave.

Example. Suppose we have a 4-th order equation that (after we apply the initial conditions) gives us the system of equations

$$1c_1 + 2c_2 + 0c_3 + 0c_4 = 5, \tag{1}$$

$$0c_1 + 3c_2 + 4c_3 + 0c_4 = 6, \tag{2}$$

$$0c_1 + 0c_2 - 2c_3 + 1c_4 = 7, \tag{3}$$

$$1c_1 + 2c_2 + 3c_3 + 4c_4 = 8. \tag{4}$$

(For consistency we have written in all the coefficients, even those that are 0 or 1.) If you have studied matrix algebra, then you know what to do next: let

$$M = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & 3 & 4 & 0 \\ 0 & 0 & -2 & 1 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

be the matrix of coefficients from the lefthand sides of the equations (can you see where all the numbers in M came from?), and let

$$c = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix}$$

be the vector of c -values, and let

$$v = \begin{pmatrix} 5 \\ 6 \\ 7 \\ 8 \end{pmatrix}$$

be the vector of numbers on the righthand sides of the equations (can you see where all the numbers in v came from?). Then the above system of equations can be written $Mc = v$.

[*Aside.* When you multiply the matrix M by the vector c , you can imagine taking each row of M and first turning it into a column vector instead of a row vector, and then taking the dot product of this vector with c , giving you a single number. After you do this for each row of M , you will end up with as many numbers as M has rows. You regard these numbers as making up a column vector called Mc . By doing this above, you see that our four equations really are equivalent to writing $Mc = v$.]

Conclusion. Since $Mc = v$, the desired solution is $c = M^{-1}v$.

Thus the whole problem boils down to finding the inverse matrix M^{-1} . If you have studied matrix algebra, then you know how this is done. But there is no need to do it by hand, because Matlab and Octave can compute the answer numerically as follows:

```
>> M=[1,2,0,0; 0,3,4,0; 0,0,-2,1; 1,2,3,4]
M =
    1  2  0  0
    0  3  4  0
    0  0 -2  1
    1  2  3  4

>> v=[5;6;7;8]
v =
     5
     6
     7
     8

>> c=(M^(-1))*v
c =
   -5.0606
    5.0303
   -2.2727
    2.4545
```

That is, the solution is (to 4 decimal places)

$$\begin{aligned}c_1 &= -5.0606, \\c_2 &= 5.0303, \\c_3 &= -2.2727, \\c_4 &= 2.4545.\end{aligned}$$

Notes.

1. When we enter matrices and vectors in Matlab or Octave, we must use commas to separate the entries within each row, and semicolons to separate the different rows.
2. Matlab and Octave find the solution numerically, not exactly. For example, the number 5.0303 should really be $5\frac{1}{33}$.
3. Even if you have not studied matrix algebra and don't understand the above process, you can certainly still check that the values for c_1, \dots, c_4 are correct: just substitute them into the original equations (1)–(4) and verify that they work.

2 Problem on Solving Large Linear Systems

Before starting this problem you need to work through the previous section.

Solve

$$(D^4 - 6D^3 + 25D^2 - 28D - 60)y = 0$$

with the initial conditions

$$y(0) = -1, \quad y'(0) = 3, \quad y''(0) = 2, \quad y'''(0) = -5.$$

Note. $D = \frac{d}{dx}$.

Remark. Your first task is to find the roots of the characteristic polynomial. One of them is $r_1 = 3$. You can guess another root r_2 (for example by plotting the characteristic polynomial), then you can check your guess directly by substituting it in to the characteristic polynomial. Then factor the characteristic polynomial as $(r - r_1)(r - r_2)(r^2 + br + c)$, after which you will be able to find the other two roots.

Reminder. For this problem, you will turn in your written solution to the problem together with a printout of the Matlab or Octave calculations that you used. See the Instructions on the first page.

3 Aside — Other software packages

Maple and Mathematica are all-purpose mathematical software packages that can perform calculations both numerically and symbolically. (Octave calculates only numerically. Matlab can do both.) For example, Mathematica can solve the differential equation $(D^4 + 5D^3 - 3D^2 - 43D - 60)y = 0$ with the initial conditions $y(0) = -1, y'(0) = 3, y''(0) = 2, y'''(0) = -5$ in a single step, using the command:

```
DSolve[{ y''''[x] + 5 y'''[x] - 3 y''[x] - 43 y'[x] - 60 y[x] == 0, y[0] == -1,
y'[0] == 3, y''[0] == 2, y'''[0] == -5 }, y, x]
```

But it can take a while to learn the commands for solving problems in Mathematica. The Help Browser feature can often provide similar examples.

Conclusion. Once you have taken an ODE course and understand the technique for solving a certain type of differential equation, then you are generally better off letting a software package handle the actual calculations.