Systems of Differential Equations

Leigh J. Little SUNY Brockport November 4, 2018

Contents

1 Introduction

We have seen examples of a single first-order initial value problem. However, we can also have systems of differential equations. In this case, we still have a single independent variable, t, but we can have several dependent variables $y_1(t), y_2(t), \ldots$. In this lecture we will discuss some aspects of systems of differential equations and how to solve them using `ode45`.

2 A Familiar Problem Stated a New Way

Consider the problem below

$$\begin{aligned} \frac{dS(t)}{dt} &= V(t), \quad S(0) = S_0 \\ \frac{dV(t)}{dt} &= -g, \quad V(0) = V_0 \end{aligned}$$

for some interval $t \in [0, t_f]$ where g is a constant. This set of equations is a system of differential equations. This means that there are two equations to be solved in order to describe some physical phenomena. In this case, we have two first-order initial value problems in the variables $S(t)$ and $V(t)$. As with a single differential equation, the above system would frequently be written in a shorter form as

$$\begin{aligned} S' &= V, \quad S(0) = S_0 \\ V' &= -g, \quad V(0) = V_0. \end{aligned}$$

This set of equations can easily be solved because the required integrations are user-friendly.

Because g is a constant, the second of the equations can be solved using separation of variables.

$$\begin{aligned} \frac{dV}{dt} &= -g \\ dV &= -g\,dt \\ \int dV &= -\int g\,dt \\ V &= -gt + C \\ V_0 &= -g \cdot 0 + C \quad \rightarrow \quad C = V_0 \\ V &= -gt + V_0 \end{aligned}$$

where we have used the initial condition $V(0) = V_0$ to determine the value of the integration constant C.

Because the value of $V(t)$ is known, we can substitute this into the first equation to determine the value of $S(t)$, again using separation of variables

$$\begin{aligned}
\frac{dS}{dt} &= V_0 - gt \\
dS &= V_0 - gt\,dt \\
\int dS &= \int V_0 - gt\,dt \\
S &= -\frac{1}{2}gt^2 + V_0 t + C \\
S_0 &= -\frac{1}{2}g \cdot 0 + V_0 \cdot 0 + C \quad \rightarrow \quad C = S_0 \\
S &= -\frac{1}{2}gt^2 + V_0 t + S_0
\end{aligned}$$

Hopefully you recognize the solution to this set of equations. $S(t)$ and $V(t)$ respectively represent the position and velocity of a point mass that is thrown vertically up or down from some height S_0 with some initial velocity V_0 assuming that g is a constant (not a bad assumption if S_0 is less than a couple of miles) and ignoring wind resistance (not a bad assumption if S_0 is less than a couple of hundred feet).

2.1 Using `ode45` to Solve the System

Because the system is comprised of initial value problems, we can still use `ode45` to solve the system. The only change is in how we write the right-hand side function, how we call the solver and how the solution is returned.

Before we solve the system, the first thing we need to do is change the notation of the problem to a more computer-friendly version. Let $y_1(t) = S(t)$ and $y_2(t) = V(t)$. Then the system can be rewritten as

$$\begin{aligned}
y_1'(t) &= y_2(t), \quad y_1(0) = S_0 \\
y_2'(t) &= -g, \quad y_2(0) = V_0
\end{aligned}$$

We need to write a MATLAB function to evaluate the two right-hand sides. Enter the following into a MATLAB file called `sys1.m` and save it

```
function yp = sys1(t,y,g)
yp(1) = y(2);
yp(2) = -g;
yp = yp(:);
```

Note that `yp` is a vector of length two because we have two derivatives that need to be evaluated (one for y_1 and one for y_2). In addition, we can still send in the value of g as a parameter. Also note the last statement. MATLAB is expecting the output of this function to be a column vector.

The statement `yp = yp(:);` is a handy statement to know. If `yp` is a column vector, this statement does nothing. If `yp` is a row vector, then this statement will transpose the vector `yp` into a column vector.

We also need a driver for the problem. For this case, we will use the values $S_0 = 100, V_0 = 0$ and $g = -9.81$. The solution range will be the interval $t \in [0, 5]$. This represents a point mass dropped from rest at a height of 100 meters using the usual metric value of g. The driver would look like

```
clear
close all

T = [0 5];
g = 9.81;
Y = [100 0];
[t,y] = ode45(@(t,y)sys1(t,y,g),T,Y);
```

In this case, the initial condition is now a vector `Y = [100 0]` because we have an initial condition for each equation. The values in this vector need to be consistent with how we defined the equations in the right-hand side `sys1.m` file.

The solution to the problem is slightly different from before. We still obtain a vector of time values `t` however the value of `y` is now a matrix. The first column of this matrix contains values of y_1 (*i.e.,* the values of S) and the second column contains the values of y_2 (*i.e.,* the values of V). These columns can be plotted individually. For example, we can do

```
figure(1)
   hold on
   plot(t,y(:,1),'b-')
   plot(t,y(:,2),'r-')
   hold off
   legend('S','V')
   xlabel('t')
```

The resulting figure is shown in Figure 1 The solution is consistent with the physics of the problem. The

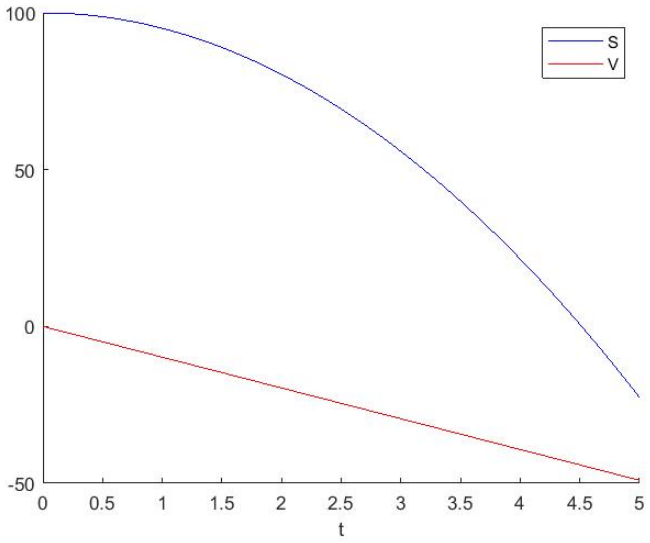

Figure 1: Solution to the falling mass problem.

position of the mass starts at a height of 100 meters, then approaches the ground quadratically, losing height more rapidly as time proceeds. The velocity starts at 0 and then increases more negatively as the mass accelerated towards the ground.

At what time does the mass reach the ground? This problem can be solved substituting the value $S = 0$ into the exact solution we found earlier and solving for t. If you do this, you find that $t_{\text{ground,exact}} = 4.51524$ seconds. However, we can also use the `revinterp` function we wrote earlier to reverse interpolate into the S array (which is `y(:,1)`) to obtain an approximate value for this

```
yval = 0;
tvals = revinter(t,y(:,1),yval);
```

If you compute the time to reach the ground in this manner, the value obtained is $t_{\text{ground,approx}} = 4.51511$ seconds. The relative error in these two values of time is

$$\begin{aligned} \text{Relative Error} &= \frac{4.51511 - 4.51524}{4.51524} \\ &= 2.87 \cdot 10^{-5} \end{aligned}$$

so the solution process seems quite accurate.

3 A Surprisingly More Complicated Problem

Instead of dropping the mass from rest, suppose we throw the mass straight up into the air with an initial velocity of $V_0 = 10m/s$? In this case, the three most common questions that would be asked are:

- When does the mass hit the ground?
- When does the mass achieve its maximum height?
- What is the maximum height of the mass?

The first of these two questions can easily be answered. We saw that we can answer the first question using the `revinterp` function. The second can also be answered using the `revinterp` function. The mass reaches its maximum height when the velocity changes sign from positive to negative. Thus, we can determine this time using the `revinterp` function, but this time using `y(:,2)` (*i.e.,* the velocity) to determine when `y(:,2)` is zero.

3.1 Finding the Maximum Height

The last question is harder than it looks at first. If you were doing this problem by hand, you would compute the time at which the velocity V is zero (call this time $t_{\max}$,) then substitute $t_{\max}$ into the equation for S to compute the maximum height.

In theory, we could do a standard (rather than a reverse) linear interpolation into the `y(:,1)` array. We could still use the `revinterp` function to do this, we would just need to reverse the input arrays; for example

```
vval = 0;
tmax = revinterp(t,y(:,2),vval);  % Get tmax time from V
smax = revinterp(y(:,1),t,tmax); % Get max height from S
```

However, this is likely to be highly inaccurate. Figure 2 illustrates the problem. The behavior of the position

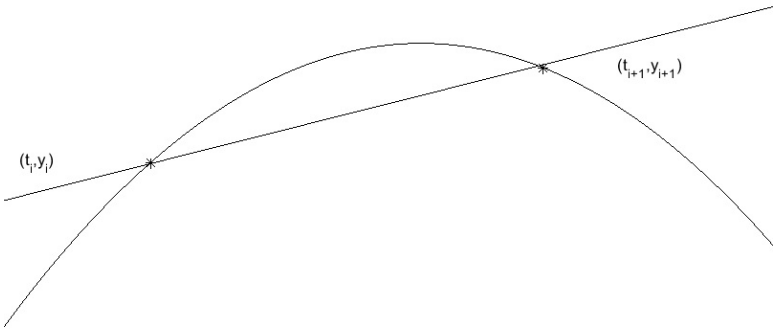

Figure 2: Estimating the maximum value of the position using linear interpolation.

goes from increasing to decreasing between two successive data values. The linear interpolation line will underestimate the true height.

3.2 A More Accurate Interpolation

To obtain an accurate value of the maximum height, we need to use quadratic interpolation instead of linear interpolation. This is not as difficult as it sounds because MATLAB has the functions we need to do this already built in.

In order to do quadratic interpolation, we need three data points. We can compute the quadratic function that goes through these three points. Once we know this, we can substitute the known value for $t_{\max}$ into the resulting quadratic expression to obtain the maximum height of the mass. Figure 3 illustrates this.

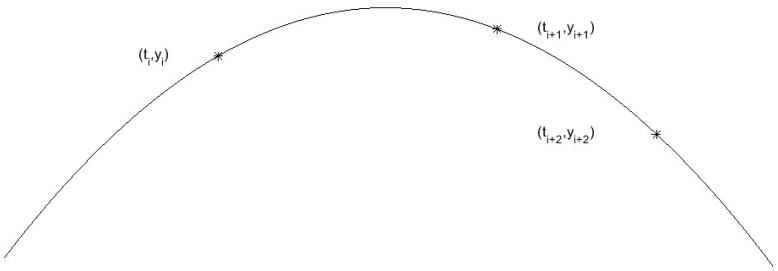

Figure 3: Estimating the maximum value of the position using quadratic interpolation. Three data points that bracket the maximum value are required.

3.3 A Modified `revinterp` Function

The first thing we are going to have to do is to make a change to the `revinterp` function. In addition to returning the independent variable values where the dependent variable attains some known input value, we are also going to need an indicator of where the sign change takes place. Essentially, we need another output vector. This contains a set of indicies of where the sign change we are looking for takes place. This modified version of `revinterp` is posted on the course website.

We would then do something like

```
vval = 0;
[tmax,tind] = revinterp(t,y(:,2),vval);  % Get tmax time from V
```

The variable `tind` contains the location in the `y(:,2)` array where `y(:,2)` changes sign. We need this information because this tells us which three points we are going to need in order to perform the quadratic interpolation.

3.4 Performing the Quadratic Fit Using `polyval`

The next step would be to determine the quadratic function that goes through the three data points closest to the relative maximum value of S. From the statements above, we know that one of these points is going to be located at position `tind`. This means that the next two data points in the arrays are going to be the data points closest to the maximum value. We need to fit a quadratic function though the points `(t(tind),y(tind,1))`, `(t(tind+1),y(tind+1,1))` and `(t(tind+2),y(tind+2,1))`.

Fortunately, MATLAB has a function we have seen before that will do exactly this. This is the `polyfit` function we saw back when were looking at the Pearson correlation coefficient. If we do

```
p = polyfit(t(tind:tind+2),y(tind:tind+2,1),2);
```

then `p` will contain the coefficients of the quadratic function that goes through the three inputs points.

3.5 Evaluating the Quadratic Fit Using `polyval`

MATLAB also has a function that will evaluate this quadratic function that we obtained using `polyval` for us rather than having to do it ourselves. This is called the `polyval` function. To get the maximum height, we send the vector of coefficients `p` obtained from the `polyfit` function along with the value of $t_{\max}$ obtained earlier into the `polyval` function:

```
smax = polyval(p,tmax);
```

3.6 The Final Driver

Putting it all together, the driver for this problem would be

```
clear
close all

T = [0 6];
g = 9.81;
Y = [100 10];
[t,y] = ode45(@(t,y)sys1(t,y,g),T,Y);
yval = 0;
tground = revinterp(t,y(:,1),yval)
vval = 0;
[tmax,tind] = revinterp(t,y(:,2),vval)
p = polyfit(t(tind:tind+2),y(tind:tind+2,1),2);
smaxquad = polyval(p,tmax)
smaxlinear = revinterp(y(:,1),t,tmax)
figure(1)
  hold on
  plot(t,y(:,1),'b-')
  plot(t,y(:,2),'r-')
  hold off
  legend('S','V')
  xlabel('t')
```

A more commented version of this is on the course web page. The results of the simulation are given in Table 1. It should be noted that the results of the simulation are atypically accurate because we are doing linear and quadratic interpolation on linear and quadratic functions.

	t_{ground}	t_{max}	S_{max}	$S_{\text{max,linear}}$
Exact	5.6479 s	1.0193 s	105.10 m	
Approximate	5.6482 s	1.0193 s	105.10 m	105.06 m

Table 1: Results of second example rounded to 5 digits. Note the difference in the maximum S values obtained using quadratic and linear interpolation.