LAB 4: DISCONTINUOUS FORCING, SIGNALS, AND NUMERICAL METHODS, PART A

(c)2019 UM Math Dept
licensed under a Creative Commons
By-NC-SA 4.0 International License.

1. Introduction

Recall that before working each part of the lab you should read through it. Both of Parts A and B have a first section that describes the *MATLAB* commands that we will be using in the lab. Read through those quickly, so that you know what they are, and remember to refer back to this section as you work the lab for help with *MATLAB*.

Next, we give an overview of the model that we introduced in the Prelab, The remainder of each part of the lab are the exercises that constitute the work that you will need to complete the reflection at the end of lab. The actual assignment you will complete then is given at the end of the Part B document.

Note that this lab is one of those for which you will submit a *reflection* at the end of the lab. You will complete all of the work for this lab **in pairs**, with a partner. At the end of the lab you will first think about the reflection questions independently, and then work with your partner to develop a response to the prompts given at the end of Part B of this lab.

2. Matlab

MATLAB commands we use in this lab include the following.

2.1. **disp.** Displays text to the command window. For example,

```
>> disp('This is text sent to the command window')
```

2.2. **eulermethod_s19.** This isn't a native *MATLAB* command; download it from the labs page. The command `eulermethod_s19` approximates the solution to a differential equation or system using Euler's method. It takes as arguments the same arguments as we use with ode45[1], plus a step size to use:

```
>> eulermethod_s19(f_handle, [tmin tmax], init_cond, h);
```

for example,

```
>> eulermethod_s19(@(t,x) [x(2); -x(1)], [0 10], [0;1], 0.1);
```

2.3. **ode45.** Finds a numerical approximation to a differential equation or system of equations:

```
>> [tsol,xsol] = ode45(f_handle, [tmin tmax], init_cond);
```

For example,

```
>> [tsol,xsol] = ode45(@(t,x) [x(2); -x(1)], [0 10], [0;1]);
```

It is possible to set options that determine how ode45 behaves; for example, we can set the maximum step size it is allowed to try by setting up an options

[1]Except that it doesn't support the addition of `options`.

object and passing that to `ode45`:

```
>> options = odeset('MaxStep', 1);
>> [tsol,xsol] = ode45(@(t,x) [x(2); -x(1)], [0 10],...
         [0;1], options);
```

(the elipses, `...`, are just to break the line here), Of course, in most instances this isn't necessary.

2.4. **ode15s.** This is another numerical solver for differential equations, and takes exactly the same arguments as `ode45`. It deals well with *stiff* systems, for which solutions have regions that change very much faster than they do in others.

2.5. **plot.** Plot one vector against another; e.g., to plot component plots from the output from `eulermethod_s19` and `ode45` in the examples above,

```
>> plot( tesol,xesol(:,1),'-k', t45sol,x45sol(:,1),'--k' );
```

2.6. **tic.** This starts *MATLAB*'s internal timer, so that you can see how long a command runs for; see `toc`

2.7. **toc.** This stops *MATLAB*'s internal timer, so that you can see how long a command runs for. For example, to see how long a call to `ode45` takes:

```
>> disp('timing for ode45');
>> tic
>> [t,x] = ode45(@(t,x) [1000*x(2); -1000*x(1)], [0 100],...
         [-2;5]);
>> toc
```

(obviously, it makes sense to use `tic` and `toc` in a script, where the only thing you are measuring is the time for the intervening commands, rather than how long it takes you to type in commands).

3. Background

In this lab we consider a circuit model, which is a second-order, linear, constant-coefficient differential equation. In the prelab we found this to be

$$y'' + 2\gamma y' + \omega_0^2 y = F(t). \tag{1}$$

The characteristic polynomial of the associated homogeneous equation is $\lambda^2 + 2\gamma\lambda + \omega_0^2$, with roots $\lambda = -\gamma \pm \sqrt{\gamma^2 - \omega_0^2}$. Thus, if γ is a small (relative to ω_0) positive number, then the system is underdamped and the solution can be written in the form $y_c(t) = Re^{-\gamma t}\cos((\sqrt{\omega_0^2 - \gamma^2})t - \phi_0)$ for some R and ϕ_0. Previously we considered forcing functions $F(t)$ of the form $F(t) = A\cos(\omega t)$. In this lab, we instead consider forcing functions, such as the step function $u_c(t)$, which are discontinuous.

To solve an equation such as (1) numerically (e.g., with `ode45`), we rewrite it as a system and the numerical method then uses known data (e.g., the initial conditions and system of equations) to predict the values of the variables y and y' at a later time. For Euler's method, which is very simple (and not very accurate), we approximate a solution to $x' = f(t,x)$ (or, for a system, $\mathbf{x}' = \mathbf{f}(t,\mathbf{x})$) by

$$\begin{aligned} t_{k+1} &= t_k + h \\ x(t_{k+1}) \approx x_{k+1} &= x_k + hf(t_k, x_k). \end{aligned} \tag{2}$$

Then x_{k+1} is an approximation of $x(t_{k+1})$ for all $k \geq 0$, and so $x(t_f) \approx x_n$. When $f(t,x)$ changes very rapidly (e.g., discontinuously), an approximation such as this may have difficulty resolving the solution accurately.

4. Reflection

For this lab you are not submitting a formal lab writeup. Instead, you will submit a shorter "reflection" at the end of Part B of the lab. In this you will be considering the four questions

a. Geometrically, how does Euler's method generate an approximate solution to a differential equation? How is this approximation related to the direction field for the differential equation?
b. Given that Euler's method, and other numerical solvers, generate an approximation for the solution to a differential equation by using previous approximate values and the differential equation, what issues do numerical methods have with short impulse forcing?
c. How can we deal numerically with problems involving delta function forces ($\delta(t)$)?
d. Why is it hard to cancel an existing signal in a real-world circuit?

As you work through the following you may wish to keep these questions in mind.

5. Part A Exercises

Unless otherwise stated, let $I(t) = \frac{1}{a}(u_c(t) - u_{c+a}(t))$. This function is defined by the file `Impulse.m`; you will need to download that from the course website, then define the impulse you want with

```
>> a = value;
>> c = value;
>> I = @(t) Impulse(t,c,a);
```

or, alternately, for $a = 1$ and $c = .5$,

```
>> I = @(t) Impulse(t, 0.5, 1);
```

Note that the `I` that this defines uses the values `a` and `c` have when `I` is defined (or, which are specified in the arguments of the call to Impulse); if you want to change those values, you will need to redefine $I(t)$, or create a new function handle `I2` that uses the different values.

For Euler's method, download the file `eulermethod_s19.m`. It is described in the *MATLAB* section, above, and takes the same arguments as `ode45` and `ode15s`, plus a stepsize h.

Note (or recall) that we define the unit step function $u_c(t)$ in [BB, §5.5] as the function $u_c(t) = \begin{cases} 0, & t < c \\ 1, & t \geq c \end{cases}$. Thus the impulse $I(t)$ is the function $I(t) = \begin{cases} 1/a, & c \leq t < c + a \\ 0, & \text{otherwise} \end{cases}$.

Exercise 1. If $a = c = 1$, what will the graph if $I(t)$ look like? Plot this impulse using `Impulse` to confirm your expectation. How will the graph be different if $a = 0.5$ and $c = 1$? If $a = 1$ and $c = 0.5$? Add both of these to your graph to see.

Exercise 2. Let $a = c = 1$. Find numerical solutions to the initial value problem $y'' + y' + 36y = I(t)$, $y(0) = y'(0) = 0$, on $0 \leq t \leq 3$, using `ode45`, `ode15s`, and `eulermethod_s19`. (For the last, use $h = 0.05$.) Plot the solutions (it may be easiest to do this on different graphs) so that you can see the steps that the different methods are taking. How are these different (for `ode45` and `ode15s`, look carefully at the step sizes)? Be sure you can explain why you might see the differences that you do.

Then plot the Euler's method solution with the solution from `ode15s`. What is the difference? Why?

Exercise 3. Next take $c = 1.01$ and $a = 0.5$. Find numerical solutions to the initial value problem $y'' + y' + 36y = I(t)$, $y(0) = y'(0) = 0$ on $0 \leq t \leq 3$ with `ode45` and `ode15s`. Then try $c = 1.01$ and $a = 0.1$. Explain what is going on in the latter case.

Exercise 4. Continue using $c = 1.01$ and $a = 0.1$, and solve the problem you considered in exercise 3. Add an `options` object (see the *MATLAB* section, above) to your function calls to `ode45` and `ode15s`, decreasing the maximum step size. How small does the maximum step size have to be for the numerical solutions to be valid?

After finding a maximum step size that works for both methods, add the *MATLAB* command `tic` and `toc` before and after the calls to `ode15s` and `ode45`. Which is faster?

References

[BB] Brannan, James R, and William E Boyce. *Differential Equations: an Introduction to Modern Methods And Applications.* Third edition. Hoboken, NJ: Wiley, 2015.