Math 151B HW 1 Project Report

Justin JR Wang

4/11/14

User Guide

odesolver.m : This function implements the other function in this program, Euler's method, midpoint method, and Modified Euler method. It atakes in all the arguments that make up an initial value problem, then passes them to the actual solver functions. Doing things this way simplifies calling.

The calling syntax is: function yt = odesolver (a, b, y0, N, f, argf, option)

Inputs:

- a is the initial time
- b is the final time
- y0 is the initial condition
- N is the number of steps that we want to compute (b-a=N*h)
- f is the function that defines the IVP
- argf is a vector stacking all parameters required by f. You can set argf=[] if no parameters are involved in f
- option is a string specifying the method being used, which can be 'euler', 'midpoint', or 'meuler'
- yt is a vector containing all approximations of the solution at given times ti.

Outputs:

- yt is the solution to the initial value problem. It is a function f with respect to t, y(t).

euler.m : This function implements the Euler's method for solving initial value problems. The algorithm is given on page 267 in Burden and Faires, Algorithm 5.1.

Euler's method is the most elementary approximation technique for solving initial-value problems.

The calling syntax is : function wc = euler(ti, wi, h, f, argf)

Inputs:

- ti is the preceding time when we have computed the approximation of the solution
- wi is the approximation of the solution at time
- h is the step size.
- f is the function that defines the IVP.
- argf is a vector containing all parameters in f
- wc is the current
- w_i+1 updated by Euler's method

Output:

- wc is the approximation w to y at the (N+1) values of t.

midpoint.m : This functions implements the Midpoint method, the procedure is explained on page 286 of the book.

$f(ti + h2; wi + h2f(ti; wi))$ is the Phi function for midpoint method

Inputs:

- ti is the preceding time when we have computed the approximation of the solution
- wi is the approximation of the solution at time
- h is the step size.
- f is the function that defines the IVP.
- argf is a vector containing all parameters in f
- wc is the current
- w_i+1 updated by Euler's method

Output:

- wc is the approximation w to y at the (N+1) values of t.

meuler.m : This function implements modified Euler's method.
$1/2[f(ti; wi) + f(ti + h; wi + hf(ti; wi))]$ is the Phi function for Modified Euler's Method.

Inputs:

- ti is the preceding time when we have computed the approximation of the solution
- wi is the approximation of the solution at time
- h is the step size.
- f is the function that defines the IVP.
- argf is a vector containing all parameters in f
- wc is the current
- w_i+1 updated by Euler's method

Output:

- wc is the approximation w to y at the (N+1) values of t.

In all cases, f is the function handle which defines the f(t, y), in the IVP of the following format.

Implementation:

This program solves initial value problems using, Euler's method, Midpoint method, and Modified Euler's method. When calling these 3 functions, we use the odesolver.m function to choose a method and pass the arguments. The step size, h, is solved for in the odesolver.m, so we modify the algorithms given in the book with this in mind.

The tests we used to validate our code are done in the two matlab functions, f1a.m, and f2d.m for f(t,y) at each IVP of the mentioned format. The IVPs are solved in main. We solve the IVPs given in the book, they are Exercise 5.4.1a, and 5.4.2d. The actual solutions are given so we can calculate the error to our approximations and graph them.

In the first of these two examples, the error encountered using Modified Euler's Method was greater than that given using Euler's method.
Modified Euler's method is a Runge-Kutta Method of Order Two, so the order of error for this new method is the same as that of the Taylor method of order two. O(h^2).

I learned that it could be the case that either Midpoint method or modified Euler's method could be a better approximation, it depends on the specific IVP. They are both Runge-Kutta methods of order 2.
Usually, Euler's method is way too inaccurate to use in practice. In exercise 5.4.1, Euler's was better, perhaps because the problem and solutions contains an e^mt term.

We wanted good graphs, since 2 step sizes was too little, so we set N=100 to get a nice smooth curve for the graph.

Solutions:

The two lines fit pretty close, they're good approximations.
The solutions to the IVPs in 5.4.1a an d5.4.2d are in the f1a.m and f2d.m functions.