Project II — Numerical Methods*

Peter Brinkmann

January 23, 2007

The purpose of this project is to give you a deeper understanding of numerical methods. It assumes that you have worked through the first Iode project as well as the lab guide on numerical methods. In particular, you should

- be familiar with Iode's direction fields module,
- understand Euler's method and Iode's implementation of it.

You can refresh your memory by reading again the first project and the two lab guides, and don't hesitate to ask questions!

A complete solution for this project consists of

- your answers to the written problems,
- a printout of the `.m` file containing the code you will write, and
- printouts of your plots.

To help you stay organized, there are little check boxes on the margin whenever there is something to include in your homework submission.

1. The Improved Euler method for computing numerical approximations of solutions to $\frac{dx}{dt} = f(t,x)$ is described by the following "Improved Euler update rule":

$$\begin{aligned} h &= t_{i+1} - t_i \\ k_1 &= f(t_i, x_i) \\ k_2 &= f(t_i + h, x_i + hk_1) \\ x_{i+1} &= x_i + h \cdot \frac{k_1 + k_2}{2} \end{aligned}$$

Action: before going on, compare this rule line-by-line with the Euler update rule that was stated in Figure 3 of Lab 2.

(a) Explain the graphical meaning of the numbers k_1 and k_2 (in particular, explain the graphical meaning of the expression $x_i + hk_1$ in the formula for k_2). *Note.* I recommend drawing a diagram, to illustrate your answer.

Write up your answer. □

Then use your answer to contrast what the Improved Euler method is doing with what the basic Euler method is doing.

(b) Implement the Improved Euler method, i.e., add a module to Iode that computes numerical solutions using this method. To do this, first `Open` the file `euler.m` using the menu item in the Matlab main window. Then IMMEDIATELY save the file under a new name, such as `impeuler.m`, by using the `File->Save As` menu item.[1]

Print out the `.m` file containing your code. □

Now edit the first line of `impeuler.m` so as to reflect its new filename: for example the new first line could read

function xc=impeuler(fs,x0,tc);

(notice we do not use the `.m` extension in this line). Next, skip down to the end of the file, ignoring the comment lines that begin with "%". At the end you will find the lines expressing the Euler update formula. Carefully modify these lines so as to implement the Improved Euler update formula given above.

Then save your work.

[1] If you are using Octave, copy the file `euler.m` to the file `impeuler.m` and then open the new file in a text editor like pico, vi, or emacs (or Notepad under Windows).

2. Test your new numerical module by applying it to various initial value problems. To do this, start the direction fields module of Iode and choose the solution method `Other`. When prompted for the name of a numerical module, enter the name of your new module, without the `.m` extension. Now Iode will use your new module to compute numerical solutions.

   To test your new module, try plotting some solutions for equations where you already know the answer (such as the example from the second lab guide; and you can take other examples from a textbook). Do *not* turn in these plots.

   If the solutions computed by your module differ greatly from the correct solutions, or if you see error messages, then you need to debug your module.

   When you feel confident that your new module works correctly, proceed to the next problem.

Nothing to hand in for this problem. □

3. Quit out of the direction fields module, then restart it to restore it to its default settings. Consider the initial value problems (a)(b)(c) below, and for each one, do the following:

   (i) Plot the exact solution, by solving the problem by hand then plotting the solution using `Plot arbitrary function`. (*Alternative.* Enter the initial condition, then select the `Exact` solution method, and click on `Plot solution` to get the exact solution. This only works if the Symbolic Toolbox is available.)

   Then on the *same* plot, show the numerical solutions computed with Euler, with your Improved Euler module (`impeuler`), and with Runge–Kutta. Use different colors for these four graphs so that the results will be easier to interpret. On your printout, write in which method created each curve.

   (*Notes.* Use display parameters $-3 \le t \le 3, -3 \le x \le 3$. And use the same step size for all graphs in this problem: use a step size for which the Euler graph and the exact solution are visibly different, but for which the Runge–Kutta graph is *very close* to the exact solution. If you need to erase a plot, just right-click once in the graphics window. WARNING. If all four graphs look the same then your step size is too small.

(ii) Rank the three numerical solution methods according to how close they get to the exact solution.

The initial value problems are

(a)
$$\frac{dx}{dt} = x, \quad x(0) = -1.$$

(b)
$$\frac{dx}{dt} = x\sin(t) + e^{-\cos(t)}, \quad x(0) = e^{-1}.$$

(c)
$$\frac{dx}{dt} = e^{x}\sin(5t), \quad x(0) = 0.8.$$

For each of the equations, attach plots for 3i, and answers for 3ii. □

4. Consider the equation

$$\frac{dx}{dt} = t - x, \quad x(0) = 1.$$

(a) Use Iode to solve the equation numerically by Euler's method, for both $h = 0.1$ and $h = 0.05$. Create a plot showing these two numerical solutions along with the exact solution.

(Use display parameters $0 \le t \le 0.8, 0 \le x \le 1.2$. For the exact solution, either solve by hand then plot with `Plot arbitrary function`, or else plot with the `Exact` solution method if the Symbolic Toolbox is available.)

Estimate the error (*i.e.*, the difference) between the exact solution and the numerical solutions, at $t = 0.8$.

Then roughly calculate the ratio of the error for $h = 0.05$ over the error for $h = 0.1$, and use it to make a guess about the effect on the error of halving the step size, in Euler's method.

(b) Repeat Part 4a for the Improved Euler method, but using $h = 0.4$ and $h = 0.2$.

Hand in your labelled plots and error calculations. □

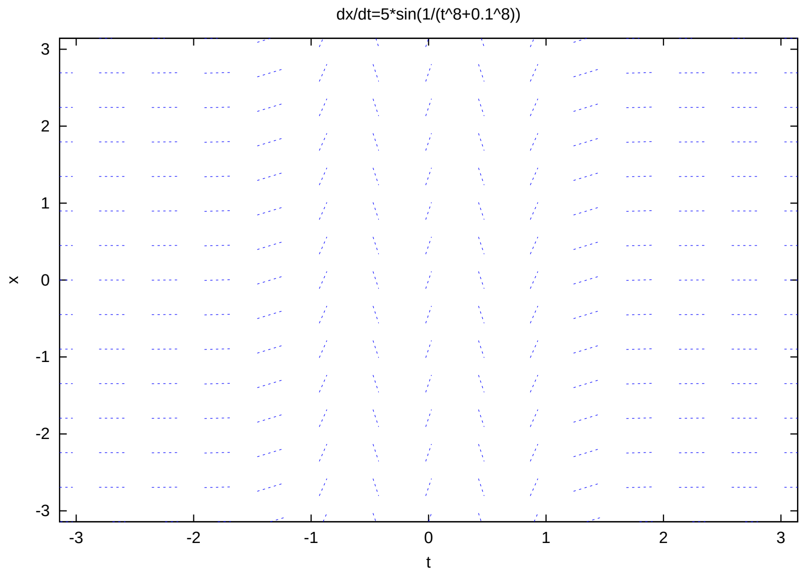

Figure 1: The plot for Problem 5.

Remark. If we used $h = 0.1$ and $h = 0.05$ then we would find the Improved Euler solutions are almost indistinguishable from the exact solution, so for the purposes of Part 4b we need bigger h-values than in part Part 4a.

5. In the examples we have seen so far, the Runge–Kutta method always agreed with the exact solution, at least up to a reasonably small margin of error. Let's investigate the accuracy of Runge–Kutta some more.

   Quit out of the direction fields module and then start it again to restore its default settings. Consider the following initial value problem:

$$\frac{dx}{dt} = 5\sin\left(\frac{1}{t^8 + 0.1^8}\right), \quad t_0 = -2, \quad x_0 = 0.$$

   When entering this differential equation, pay special attention to the proper placement of parentheses. If you entered it correctly, you'll see a plot that looks like Figure 1.

(a) Plot solutions with Runge–Kutta with step sizes 0.1, 0.05, and 0.01.

Attach your plot. □

(b) Answer the following questions.

(i) What do the solution plots look like (say, for $t < 0$, for t near 0, for $t > 0$)?

(ii) What is the value of $\frac{1}{t^8+0.1^8}$ for $t = 0$? for $t = 0.1$? for $t = 0.2$? for $t = 1$?

(iii) Graph $\sin\left(\frac{1}{t^8+0.1^8}\right)$ for $0 \leq t \leq 1$. Describe the most important behavior of this graph, and relate this behavior to your answer in (b).

Hint. Iode can plot $\sin(\frac{1}{t^8+0.1^8})$ for you as follows. Open another direction fields window in Iode, and enter the zero direction field $f(t, x) = 0$ (so as not to confuse matters), then enter sensible display parameters, and lastly use the menu item `Plot arbitrary function` to plot $\sin\left(\frac{1}{t^8+0.1^8}\right)$.

(iv) Is any of the three numerical solutions to $\frac{dx}{dt} = 5\sin\left(\frac{1}{t^8+0.1^8}\right)$ in part (a) likely to be close to an exact solution? Explain.

Hint. You can see what goes wrong by zooming in on the direction field near one of your solution curves. Does the curve follow the line segments of the direction field, as it ought to? If not, explain what is going wrong by referring to part (iii).

After you zoom in, decrease the step size suitably and then plot some more solutions. Do these new solutions follow the direction field lines?

Write up your answers. □

Remark. The point of the last exercise is that even extremely sophisticated numerical methods can fail miserably, when confronted with a badly behaved differential equation. We cannot just blindly trust numerical solutions!