

(7) This is the slope field of $dy/dx = f(x, y)$. Where is the function $f(x, y)$ positive, negative, and zero? Is $f(x, y)$ a function of x alone, y alone, or both variables together? Find a function $f(x, y)$ whose slope field looks like this.

(8) This is the slope field of $dy/dx = g(x, y)$. Where is the function $g(x, y)$ positive, negative, and zero? Is $g(x, y)$ a function of x alone, y alone, or both variables together? Find a function $g(x, y)$ whose slope field looks like this.

§8 NUMERICAL METHODS OF APPROXIMATING SOLUTIONS

Discussion: So far we have learned some special techniques for solving first order equations. In some sense, a variety of special techniques is all that is available. Most first order equations can't be solved explicitly, even in integral form. That being the case, it is often useful to know how to find an approximation to the solution when you can't find the solution exactly. We will consider the two simplest numerical methods for approximating solutions, Euler's method and the improved Euler's method.

Chapter 1: First Order Differential Equations

We consider the initial value problem

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0.$$

We would like to approximate $y(x_1)$. Recall that dy/dx is the slope of the tangent line, so plugging the initial condition into the differential equation gives us the slope of the tangent line to the graph of the function $y(x)$ at $x = x_0$. We can then use the tangent line approximation from Calculus I,

$$\begin{aligned} y(x_1) &\approx y(x_0) + (x_1 - x_0)y'(x_0) \\ &= y(x_0) + (x_1 - x_0)f(x_0, y_0) \end{aligned}$$

If x_1 is near x_0 , this technique works okay. If x_1 is not close to x_0 however, then the tangent line approximation isn't accurate. To approximate $y(x_1)$ for x_1 far from x_0 , we build up the approximation in several steps. We pick a step size h . Then we use the tangent line approximation to find

$$\begin{aligned} y(x_0 + h) &\approx y_1 = y(x_0) + hy'(x_0) \\ &= y(x_0) + hf(x_0, y_0) \end{aligned}$$

We write $x_1 = x_0 + h$ so that our approximation becomes $y(x_1) \approx y_1$. Now repeat the process to approximate

$$y(x_0 + 2h) \approx y_2 = y_1 + hf(x_1, y_1).$$

Continue in this manner until you have reached $y(x_1)$.

Paradigm: Approximate $y(.3)$ using Euler's method with a step size of $h = .1$ where y is the solution to the initial value problem

$$\frac{dy}{dx} = 2x - y, \quad y(0) = 1.$$

Note that each step of the paradigm is the same. You just repeat the basic formula for Euler's method over and over again.

STEP 1: $y(x_0 + h) \approx y_1 = y_0 + hf(x_0, y_0)$.

Plugging into this formula we have

$$y(.1) \approx y_1 = 1 + .1(2 \times 0 - 1) = .9.$$

We also write $x_1 = 0 + .1 = .1$ to prepare for the next step.

STEP 2: $y(x_0 + 2h) \approx y_2 = y_1 + hf(x_1, y_1)$.

Plugging into this formula we have

$$\begin{aligned} y(.2) &\approx y_2 = y_1 + .1(2x_1 - y_1) \\ &= .9 + .1(2 \times .1 - .9) = .83. \end{aligned}$$

And we also write $x_2 = .1 + .1 = .2$.

STEP 3: $y(x_0 + 3h) \approx y_3 = y_2 + hf(x_2, y_2)$.

Plugging into this formula we have

$$\begin{aligned} y(.3) &\approx y_3 = y_2 + .1(2x_2 - y_2) \\ &= .83 + .1(2 \times .2 - .83) = .787. \end{aligned}$$

We write $x_3 = .2 + .1 = .3$ just for consistency, but we have found our answer $y(.3) \approx .787$.

While most equations don't have explicit solutions, it is possible to find an exact solution for $dy/dx = 2x - y$, $y(0) = 1$. The reason for choosing a paradigm where the exact solution is available is so we can check how accurate our approximation is. The exact solution to the initial value problem is $y(x) = 3e^{-x} + 2x - 2$ so $y(.3) = .8225\dots$. So the error in our approximation is $.8225 - .787 = .0355$ which is an error of a little more than 4% of the true answer. That isn't all that bad, but it isn't all that good either. We could get a more accurate answer by using a step size of $h = .05$ instead of $h = .1$. This would be twice as much work, since now it would take 6 steps to get to .3 rather than 3 steps as above. But the answer we would get is $y(.3) \approx .8053\dots$ which is only off by a little more than 2% of the true answer. So for twice as much work we get half the error. This is usually, but not always, the case. Some of you are probably thinking that if you have two approximations where the error in the first approximation is twice the error in the second approximation then you can do some algebra and find the true answer. This is Romberg extrapolation. You are welcome to stop by my office or take Math 655, Elementary Numerical Analysis for more details. Romberg extrapolation is sometimes used in numerical evaluation of integrals but rarely in numerical solution of differential equations, because there are better techniques available as we will discuss later.

The weakness in Euler's method is that the tangent line approximation is only accurate for quite small step sizes. Furthermore, since you reuse the results of the previous approximation in each step, small initial errors can build up to large errors quite rapidly. Some extra

work getting a more accurate approximation than the tangent line approximation will pay off. So how can we improve that approximation? Well, what we are looking for is not the tangent line to the curve, we are looking for the “secant” line between the points $(x_0, y(x_0))$ and $(x_1, y(x_1))$. Then the formula for the secant line is $y(x_1) = y(x_0) + (x_1 - x_0)S$ where S is the slope of the secant line. So how can we approximate the slope of the secant line? The tangent line approximation uses the slope of the tangent line at the left endpoint (x_0, y_0) as the approximation to the slope of the secant line. Geometrically, there is no reason why the left endpoint should give a better approximation than the right endpoint. In fact, the average of the slopes of the tangent lines at each endpoint is better still. Unfortunately, we only know the left endpoint (x_0, y_0) ; the right endpoint $(x_1, y(x_1))$ is what we are trying to find (which is why we used the left endpoint in the tangent line approximation). But once we have carried out Euler’s method, we have an approximation to the right endpoint, (x_1, y_1) . Now we can use this approximation to compute the approximate slope of the tangent line at the right endpoint and then the approximate slope of the secant line. All those “approximates” in the last sentence may look scary, but the process is actually fairly straightforward. For the initial value problem

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0,$$

we first compute

$$\tilde{y}_1 = y_0 + h(f(x_0, y_0)), \quad x_1 = x_0 + h$$

as in Euler’s method. We then improve this to

$$y_1 = y_0 + h \left(\frac{f(x_0, y_0) + f(x_1, \tilde{y}_1)}{2} \right)$$

We repeat this procedure as many times as needed to reach our desired point just as with the original Euler’s method.

Paradigm: Approximate $y(.3)$ using the improved Euler’s method with a step size of $h = .1$ where y is the solution to the initial value problem

$$\frac{dy}{dx} = 2x - y, \quad y(0) = 1.$$

Note that each step of the paradigm is the same. You just repeat the basic formula for the improved Euler’s method over and over again.

STEP 1: SubStep 1: Use the basic Euler's method $\tilde{y}_1 = y_0 + hf(x_0, y_0)$.

$$\tilde{y}_1 = 1 + .1(2 \times 0 - 1) = .9, \quad x_1 = 0 + .1 = .1.$$

SubStep 2: Improve the Euler's method approximation $y_1 = y_0 + h(f(x_0, y_0) + f(x_1, \tilde{y}_1))/2$.

$$y_1 = 1 + .1 \left(\frac{(2 \times 0 - 1) + (2 \times .1 - .9)}{2} \right) = .915.$$

STEP 2: SubStep 1: Use the basic Euler's method $\tilde{y}_2 = y_1 + hf(x_1, y_1)$.

$$\tilde{y}_2 = .915 + .1(2 \times .1 - .915) = .8435, \quad x_2 = .1 + .1 = .2.$$

SubStep 2: Improve the Euler's method approximation $y_2 = y_1 + h(f(x_1, y_1) + f(x_2, \tilde{y}_2))/2$.

$$y_2 = .915 + .1 \left(\frac{(2 \times .1 - .915) + (2 \times .2 - .8435)}{2} \right) = .8571.$$

STEP 3: SubStep 1: Use the basic Euler's method $\tilde{y}_3 = y_2 + hf(x_2, y_2)$.

$$\tilde{y}_3 = .8571 + .1(2 \times .2 - .8571) = .8114, \quad x_3 = .2 + .1 = .3.$$

SubStep 2: Improve the Euler's method approximation $y_3 = y_2 + h(f(x_2, y_2) + f(x_3, \tilde{y}_3))/2$.

$$y_3 = .8571 + .1 \left(\frac{(2 \times .2 - .8571) + (2 \times .3 - .8114)}{2} \right) = .8237.$$

(calculations here have only been reported to 4 decimal places.)

Note that the improved Euler's method is twice as much work as the original Euler's method, since you do an extra improvement substep in each step. On the other hand, the error is now just .0012..., which is only 0.15% of the true answer, much closer than the 2% error we got by doing twice as much work with a smaller step size with the original Euler's method. We can improve the accuracy of our approximation further by using a smaller step size of $h = .05$. The result of this calculation is .8227..., which is off by only .0002..., an error of just .03% of the true value. Notice that doing twice as many steps with the improved Euler's method does much better than just halving the error. This is another advantage of the improved Euler's method.

Many people prefer to use a hand calculator to do these sorts of computations (as long as we are only doing a couple of steps). If you don't have a calculator available, you can

Chapter 1: First Order Differential Equations

use Matlab to carry out the calculations for Euler's method and the improved Euler's method by just typing them into the main window. You can also store values in variables with commands like $\mathbf{x0=0}$, $\mathbf{h=.1}$, and $\mathbf{x1=x0+h}$. Matlab will print the result of each calculation. If you forget what value a variable has, you can just type the variable name, like $\mathbf{x1}$, and Matlab will print out its value. You can also use a hand calculator or even pencil and paper. You can program the Euler and improved Euler methods into Matlab pretty easily (as you will see in the next lab) but some better methods have already been programmed for you.

We could repeat the process of improving the estimate by using both \tilde{y}_1 and y_1 from the improved Euler's method to get a New And Improved Euler's method. And, of course, we could then improve the estimate another time and yet another time. This leads to a class of methods called Runge-Kutta methods. We will cover Runge-Kutta methods in the extra credit assignment this week.

Exercises:

Find $y(.2)$ for each of the initial value problems using Euler's method, first with a step size of $h = .1$ and then with a step size of $h = .05$.

- (1) $\frac{dy}{dx} = xy, \quad y(0) = 1$
- (2) $\frac{dy}{dx} = 3y, \quad y(0) = 2$
- (3) $\frac{dy}{dx} = \frac{x - y}{x + 2y}, \quad y(0) = 1$

Find $y(.2)$ for each of the initial value problems using the improved Euler's method, first with a step size of $h = .1$ and then with a step size of $h = .05$. (The problem numbers are funny to match the numbers in the printed notes).

- (9) $\frac{dy}{dx} = xy, \quad y(0) = 1$
- (10) $\frac{dy}{dx} = 3y, \quad y(0) = 2$
- (11) $\frac{dy}{dx} = \frac{x - y}{x + 2y}, \quad y(0) = 1$

Find $y(.2)$ for each of the initial value problems by solving the equation explicitly, then

compare the errors in the different methods used above.

$$(17) \quad \frac{dy}{dx} = xy, \quad y(0) = 1$$

$$(18) \quad \frac{dy}{dx} = 3y, \quad y(0) = 2$$

$$(19) \quad \frac{dy}{dx} = \frac{x - y}{x + 2y}, \quad y(0) = 1$$

§9 RUNGE-KUTTA METHODS

Discussion: Euler’s method and the improved Euler’s method are the simplest examples of a whole family of numerical methods to approximate the solutions of differential equations called Runge-Kutta methods. In this section we will give *third* and *fourth* order Runge-Kutta methods and discuss how Runge-Kutta methods are developed.

Euler’s method and the improved Euler’s method both try to approximate $y(x_0 + h)$ by approximating the slope m of the secant line from $(x_0, y(x_0))$ to $(x_0 + h, y(x_0 + h))$ and using the formula $y(x_0 + h) = y(x_0) + mh$ (see figure 1).

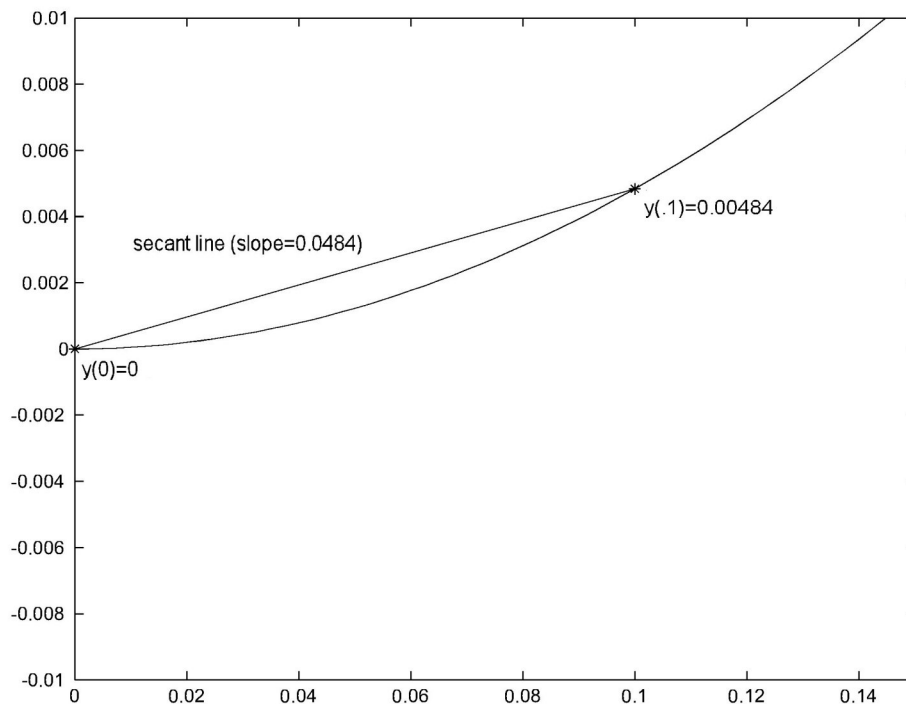


Figure 1

Euler's method approximates the slope of the secant line by the slope of the tangent line at the left endpoint $(x_0, y(x_0))$. The improved Euler's method uses the average of the slopes at the left endpoint and the approximate right endpoint (that is the right endpoint as computed by Euler's method) to approximate the slope of the secant line. We don't have to stop there either. We can keep finding slopes at different points and computing weighted averages to approximate the slope of the tangent line. Numerical methods to approximate the solution of differential equations in this fashion are called Runge-Kutta methods (after the mathematicians Runge and Kutta). A third order Runge-Kutta method is the following:

To approximate the solution of

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

compute

$$\begin{aligned}x_1 &= x_0 + h \\k_1 &= f(x_0, y_0) \\k_2 &= f(x_0 + h, y_0 + hk_1) \\k_3 &= f(x_0 + h/2, y_0 + (h/2)(k_1 + k_2)/2) \\y_1 &= y_0 + \frac{k_1 + k_2 + 4k_3}{6}h\end{aligned}$$

Then $y(x_1) \approx y_1$.

This method uses the improved Euler method to find an approximate midpoint of the secant line and then takes a weighted average of the slopes at the left and right endpoints and the midpoint. A fourth order Runge-Kutta method is the following

To approximate the solution of

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

compute

$$\begin{aligned}x_1 &= x_0 + h \\k_1 &= f(x_0, y_0) \\k_2 &= f(x_0 + h/2, y_0 + (h/2)k_1) \\k_3 &= f(x_0 + h/2, y_0 + (h/2)k_2) \\k_4 &= f(x_0 + h, y_0 + hk_3) \\y_1 &= y_0 + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6}h\end{aligned}$$

Then $y(x_1) \approx y_1$.

Just as with Euler's method and the improved Euler's method, we can repeat the process to find an $x_2 = x_1 + h = x_0 + 2h$ and $y_2 \approx y(x_2) = y(x_0 + 2h)$ and so on. The fourth order Runge-Kutta method listed above is powerful enough that it was a popular technique for practical computation (and not just for educational purposes). The steps are well suited to programming into a computer, just loop through the basic algorithm as many times as needed to get to the values you are interested in. The main difficulty is in choosing an appropriate size for h . The larger h is, the greater the error in each step of the computation. On the other hand, the smaller h is, the more steps you will have to take. It can be shown that the errors in each step shrink fast enough that even though you are making more steps, the total *truncation* error decreases as you take h smaller where truncation error is the error assuming all arithmetic is done perfectly. Unfortunately, on a computer or calculator you will have roundoff error that will build up as you take more steps and will eventually overwhelm your calculations. Picking an h small enough that the answer is sufficiently accurate but not so small that roundoff error builds up too great (or the program works too slowly because it is trying to take too many steps) requires more sophisticated methods, the Runge-Kutta-Fehlberg methods, which have largely replaced the Runge-Kutta methods in practical computation. We will go on to discuss these more sophisticated techniques in next week's extra credit assignment.

Exercises:

1. Use the third order Runge-Kutta method with step sizes $h = 0.1$ and $h = 0.05$ to approximate $y(0.1)$ for the initial value problem $dy/dx = x - y$, $y(0) = 0$.
2. Use the fourth order Runge-Kutta method with step sizes $h = 0.1$ and $h = 0.05$ to approximate $y(0.1)$ for the initial value problem $dy/dx = x - y$, $y(0) = 0$.
3. Use the third order Runge-Kutta method with step sizes $h = 0.1$ and $h = 0.05$ to approximate $y(1)$ for the initial value problem $dy/dx = x - y$, $y(0) = 0$. You can do this by hand but it is easier if you can program the technique into a calculator or computer (a spreadsheet actually works pretty well for this).
4. Use the fourth order Runge-Kutta method with step sizes $h = 0.1$ and $h = 0.05$ to approximate $y(1)$ for the initial value problem $dy/dx = x - y$, $y(0) = 0$. You can do this

by hand but it is easier if you can program the technique into a calculator or computer (a spreadsheet actually works pretty well for this).

5. Compute the exact solution to the initial value problem $dy/dx = x - y$, $y(0) = 0$ and compare your results in the first four problems to the exact answers.

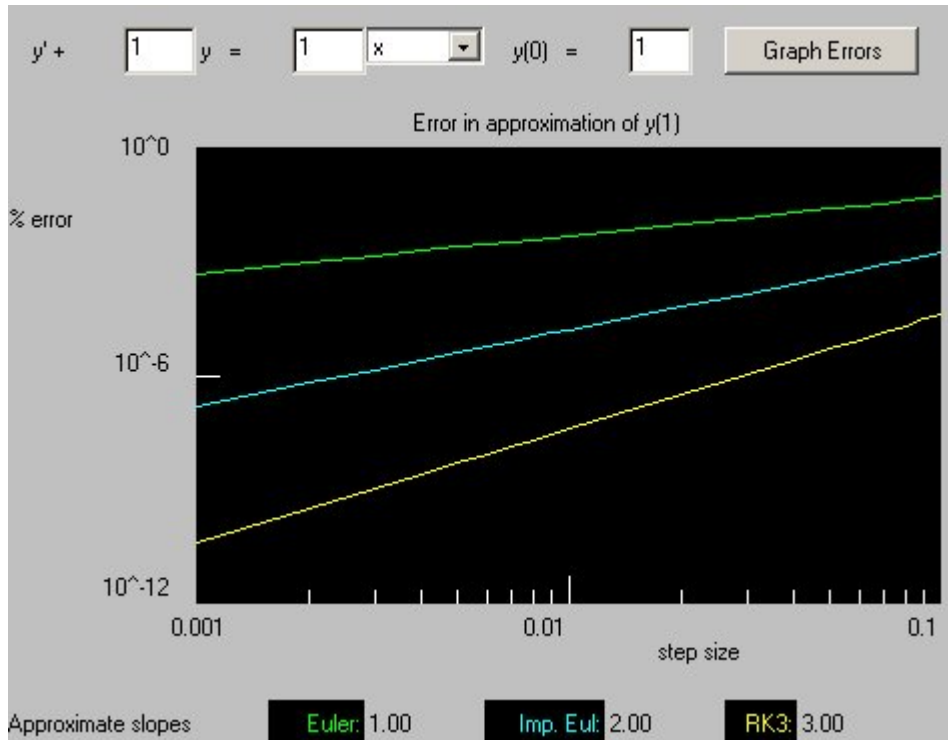
Section 8 of the notes goes over programming in matlab, which might be useful for solving problems 3 and 4 above. Unfortunately, since the network was switched over the summer you can't save files in the same manner as indicated in the text. I'll get revised instructions on programming in matlab up as soon as possible. In the meantime, matlab is available on the unix system on campus and the instructions as given in the notes still work fine for the public unix system.

§10 ERRORS IN NUMERICAL METHODS

Discussion: In general when using a numerical approximation, the smaller the step size the more accurate the approximation. Of course, the smaller the step size, the more steps you need to reach your desired value, and so the more time the process takes. Furthermore, the more steps you take the more roundoff error builds up and eventually shrinking the step size ceases to yield better approximations because the roundoff error becomes larger than the “discretization error” (the error introduced by the approximation procedure).

In understanding the tradeoffs involved in choosing the correct step size, it is useful to look at how much shrinking the step size improves the accuracy of the approximation. It turns out that there is a qualitative difference between Euler's method and the improved Euler's method and higher order Runge-Kutta methods. In this lab we will investigate the errors in these approximation methods applied to first order linear differential equations. It is a flaw that we will look solely at linear equations rather than more general types, but in order to compute the error we need to be able to compute the exact value of the answers and this motivates us to look at a set of equations where we can compute exact answers easily. Since you now have a little practice with the labs, we will move away from just writing down answers to questions.

Instructions: Get into Matlab, then enter **lab3** (no space) to start the lab. The lab



window will look like the illustration.

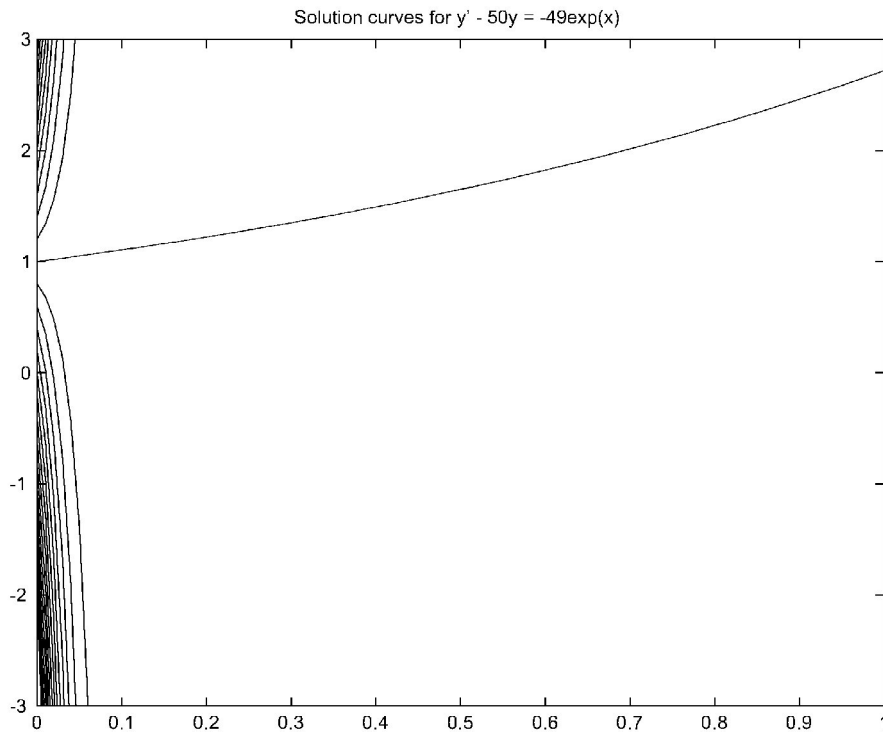
It will take the window a while to come up because it has to compute 57 different numerical approximations to the solution of the differential equation $y' + y = x$. This is a log plot of the data as you can tell by looking at the scale. The black line is for using Euler's method, the blue line is for the improved Euler's method and the red line is for a third order Runge-Kutta method (as explained in the extra credit, you aren't responsible for knowing exactly what that method is, just accept it as a more complicated and more efficient variation on the numerical methods we studied last week). As you can see, the plot of percentage error versus step size is almost a perfect line when plotted logarithmically. The slopes as measured at the left edge (small step sizes) of the three lines are given at the bottom of the window. You can change the initial value problem by entering new coefficients for the differential equation or a new initial value in the text fields at the top of the window. You can also change the function on the right hand side of the equation by clicking on the listbox and choosing the function you want from the menu. You must press the "Update Graph" button to update the graph and the slopes. Note that you will have the same wait you had for the window to first come up every time you update the graph.

Try at least 6 different initial value problems with at least two for each different function on the right hand side (x , $\cos(x)$, $\exp(x)$). Record the problems you try and the values of

Chapter 1: First Order Differential Equations

the slopes you get. Also make notes about the graphs. Most of the graphs will probably look linear like the graphs in the first illustration, but some may look a little different, especially near the right edge (large step sizes). You should quickly sketch any graphs you get that look different.

One example of a graph that looks different is what you get for the initial value problem $y' - 50y = -49\exp(x)$, $y(0) = 1$. This is a simple example of a stiff problem. To see what the difficulty is, look at the graph of the solution curves for the differential equation with 31 different initial values given below.



As you can see, the solution curve with $y(0) = 1$ is special. It is the fence between the curves that head to positive infinity and the curves that head to negative infinity. What's more, the slope field is so steep that the solution curves head toward infinity very quickly on either side of the fence. So once the approximate solution errs toward either side of the fence, the solution field will sweep it far away from the true solution. Stiff problems require special care and handling that we won't cover in this course. Can you find another example of a stiff problem?

Exercise: For this lab you are assigned to write an actual lab report. List the examples you tried and the results you got, both the slopes and comments about the graphs, including quick sketches of any unusual graphs (of course the stiff problem will give you at least one unusual graph). Then write a discussion about your results. Your discussion should include the answers to the following questions.

1. The percent error curves are (usually) lines when plotted in log scale. What does this mean about the relationship between the error, E , and the step size, h ? (Hint: the relationship can be written as $\log(E) = m \log(h) + c$ where m and c are the slope and intercept of the line in the log plot. Now exponentiate this to get a more direct relationship between E and h).
2. Euler's method is called a first order method. The improved Euler's method is a second order method. The Runge-Kutta method used in this lab (the first one listed in the extra credit) is a third order method. Why are those terms appropriate?
3. What is the general solution to the equation $y' - 50y = -49 \exp(x)$? What is special about the solution with initial condition $y(0) = 1$? This may give you some hints on finding another stiff problem.

§11 RUNGE-KUTTA-FEHLBERG METHODS

Discussion: The trickiest part of using Runge-Kutta methods to approximate the solution of a differential equation is choosing the right step-size. Too large a step-size and the error is too large and the approximation is inaccurate. Too small a step-size and the process will take too long and possibly have too much roundoff error to be accurate. Furthermore, the appropriate step-size may change during the course of a single problem. Many problems in celestial mechanics, chemical reaction kinematics, and other areas have long periods of time where nothing much is happening (and for which large step-sizes are appropriate) mixed in with periods of intense activity where a small step-size is vital. What we need is an algorithm which includes a method for choosing the appropriate step-size at each step. The Runge-Kutta-Fehlberg methods do just this.

Fehlberg's improvement to the Runge-Kutta method is based on two assumptions.

1. The error in a single step of the improved Euler's method is about $C'h^3$ and the error in a single step of the third order Runge-Kutta method is about $C''h^4$. (The error in a single step is the error in $y(x_0 + h)$ computed with a step-size of h . It is also called the local error.)
2. $C''h^4$ is much smaller than $C'h^3$.

Let us consider these assumptions in reverse order. Provided h is small, it seems reasonable to hope that $C''h^4 \ll C'h^3$ (the \ll means much less than). The danger is that C'' might be much larger than C' , but in practice that *usually* doesn't happen. The first assumption can be justified on theoretical grounds using Taylor series approximations, but we won't go into all that. We will look at some examples to see that these assumptions usually do hold in practice. Go to the labs page on the web site and launch lab 2. This lab works the same as lab 3, except that instead of plotting the error in the approximation for $y(1)$ for different step-sizes h , it plots the error for $y(h)$, that is the local error, for different step-sizes h . Try a variety of different problems. You should see that in most cases both assumptions are valid. The slope of the error curve in log scale is about 3 for the improved Euler's method and about 4 for the RK3 method (which you should check is equivalent to assumption 1). And for any given step-size, the error in the RK3 method is much smaller than the error in the improved Euler's method (remember things are plotted in log scale so a modest difference on the screen can correspond to a very large difference in magnitude). Of course, you can find exceptional cases where the assumptions fail if you play around for a little while – no numerical method works perfectly for all problems. But these assumptions hold for the vast majority of problems.

To see how Fehlberg used these assumptions to compute a good step-size, consider the following equations where y'_1 is the approximation from the improved Euler's method, y''_1 is the approximation from the third order Runge-Kutta method, and $y(h)$ is the true value.

$$\begin{aligned}y'_1 &\approx y(h) + C'h^3 \\y''_1 &\approx y(h) + C''h^4 \\|y'_1 - y''_1| &\approx |C'h^3 - C''h^4| \approx |C'|h^3\end{aligned}$$

where the last line uses the assumption that $C''h^4$ is very small compared to $C'h^3$. Now we can solve this last line for the constant C' to get

$$|C'| \approx \frac{|y'_1 - y''_1|}{h^3}$$

Once we have this approximation for C' , we can pick a step-size h_1 to get the local error of the size we want. If we want the local error to be about size T , we just take a step-size h_{new} where

$$h_{\text{new}} = h \left(\frac{T}{|y'_1 - y''_1|} \right)^{1/3}$$

and then the error should be about

$$\begin{aligned} |\text{error}| &\approx |Ch_{\text{new}}^3| \\ &\approx \frac{|y'_1 - y''_1|}{h^3} h^3 \frac{T}{|y'_1 - y''_1|} \\ &\approx T \end{aligned}$$

You might be a little worried about how all the errors in the different approximations mount up as we carry out all these computations to get our new step-size h_{new} . This is a serious consideration and is dealt with by introducing a *chicken factor*, usually taken to be 0.9. We actually use a step-size

$$h_1 = 0.9h \left(\frac{T}{|y'_1 - y''_1|} \right)^{1/3} .$$

This is a bit smaller than the approximation we computed above, and thus a bit safer too.

The Runge-Kutta-Fehlberg 2(3) method uses exactly this technique to pick the right step-size. Suppose the initial value problem we want to solve is

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = y_0$$

We have an initial step-size h (taken to be whatever value you fancy, we will update it automatically as needed). We compute the improved Euler's and RK3 estimates in the usual fashion.

$$\begin{aligned} k_1 &= f(x_0, y_0) \\ k_2 &= f(x_0 + h, y_0 + h * k_1) \\ k_3 &= f(x_0 + h/2, y_0 + h * (k_1 + k_2)/4) \\ y'_1 &= y_0 + h * (k_1 + k_2)/2 \\ y''_1 &= y_0 + h \frac{k_1 + k_2 + 4k_3}{6} \end{aligned}$$

Next we compute the estimated error

$$|\text{error}| \approx |y'_1 - y''_1|$$

Chapter 1: First Order Differential Equations

If this error is small enough, say within a tolerance of $T = .001 * \max(|y_0|, 1)$ (which is the tolerance used by the matlab program ode23), then we accept this step-size for the current step and let

$$\begin{aligned}x_1 &= x_0 + h \\y_1 &= y_1''\end{aligned}$$

If the error is greater than T , we reject this step-size for the current step and leave x_0 and y_0 as they are. In either case, we choose a new step-size

$$h_{\text{new}} = .9h \left(\frac{T}{|y_1' - y_1''|} \right)^{1/3}$$

We then either compute the next step with the new step-size (if our error was less than T) or we repeat the current step with the new step-size h_{new} (if the error was greater than T) and try again to find x_1 and y_1 .

Example: Approximate $y(1)$ if $dy/dx = x + y$, $y(0) = 0$ using the RK2(3) method with tolerance $T = 0.01$. (This tolerance is rather large, but I want to work things out by hand so I don't want to take too many steps.)

We need to pick an initial step-size to get things started. We have to go from $x_0 = 0$ to $x = 1$ so why not go for it all in one shot and guess $h = 1$ initially. Actually, the reason not to use that large a step-size for the beginning is that it is usually too large and often takes a couple of iterations before you get the step-size down to a reasonable size and actually get the process off and running. Choosing a step-size of about the length of the interval divided by 16 or 32 is more typical. But I wanted to be sure I had to reject the estimate sometime in the course of the example so I decided to start off wrong with too large a step size to be sure that happened. We carry out the following computations.

$$\begin{aligned}k_1 &= 0 \\k_2 &= 1 \\k_3 &= .75 \\y_1' &= .5 \\y_1'' &= .666666\dots \\|y_1' - y_1''| &= .166666\dots > .01\end{aligned}$$

The estimated error is greater than the tolerance .01 so we reject the initial step-size of $h = 1$. We compute a new step-size and try again.

$$\begin{aligned}h_{\text{new}} &= .9 * 1 * (.01/.166666\dots)^{1/3} = .3523380877 \\k_1 &= 0 \\k_2 &= .35233880877 \\k_3 &= .2072045759 \\y'_1 &= .062071064 \\y''_1 &= .069361064 \\|y'_1 - y''_1| &= .00729 < .01\end{aligned}$$

This time the estimated error is less than the tolerance so we accept the step-size and estimate and compute

$$\begin{aligned}x_1 &= x_0 + h = .3523380877 \\y_1 &= y'_1 = .069361064\end{aligned}$$

We now compute a new step-size h_{new} and go on to the next step (Twice in this problem, we compute a new step-size and it turns out to exactly equal the old step-size. This is a freak accident. I couldn't have planned that to happen if I tried.)

$$\begin{aligned}h_{\text{new}} &= .9 * .3523380877 * (.01/.00729\dots)^{1/3} = .3523380877 \\k_1 &= .4216991517 \\k_2 &= .9276179121 \\k_3 &= .7162817215 \\y'_2 &= .3061881158 \\y''_2 &= .3165523026 \\|y'_2 - y''_2| &= .0103641868 > .01\end{aligned}$$

This time the estimated error is greater than the tolerance so we reject the step-size and try again with the same x_1 and y_1 .

$$\begin{aligned}h_{\text{new}} &= .9 * .3523380877 * (.01/.0103641868)^{1/3} = .3133456655 \\k_1 &= .4216991515 \\k_2 &= .8671824185 \\k_3 &= .6793383478 \\y'_2 &= .2712937907 \\y''_2 &= .2785837907 \\|y'_2 - y''_2| &= .00729 < .01\end{aligned}$$

Chapter 1: First Order Differential Equations

The estimated error is less than the tolerance so we accept the step-size and compute

$$x_2 = x_1 + h = .6656837532$$

$$y_2 = y_2'' = .2785837907$$

We now compute the new step-size for the next step and repeat the process.

$$h_{\text{new}} = .9 * .3133456655 * (.01/.00729)^{1/3} = .3133456655$$

$$k1 = .9442675439$$

$$k2 = 1.553495351$$

$$k3 = 1.296606171$$

$$y_3' = .669915379$$

$$y_3'' = .679849358$$

$$|y_3' - y_3''| = .0099695568 < .01$$

The estimated error is less than the tolerance so we accept the estimate and compute

$$x_3 = x_2 + h = .9790294187$$

$$y_3 = y_3'' = .679849358$$

We now compute the step-size for the next step

$$h_{\text{new}} = .9 * .3133456655 * (.01/.0099695568)^{1/3} = .2822978588$$

But this step-size is too large since $x_3 + h = 1.261327277 > 1$ and so it would put us past our final value for x . Therefore we shrink h to hit $x = 1$ exactly.

$$h_{\text{new}} = 1 - x_3 = .0209705813$$

$$k1 = 1.658914354$$

$$k2 = 1.714673334$$

$$k3 = 1.687086169$$

$$y_4' = .7152579833$$

$$y_4'' = .7152620701$$

$$|y_4' - y_4''| = .00000408681 < .01$$

The estimated error is less than the tolerance so we accept this estimate and get the final computations

$$x_4 = x_3 + h = 1$$

$$y_4 = y_4'' = .7152620701$$

So our approximation for $y(1) = y(x_4) \approx y_4 = .7152620701$. The true value is $y(1) = .7182818285$ and the error is $.0030197584$ or a percent error of 0.42% . You may be wondering how we got a relative error of less than 1% when we had a tolerance of 1% error in each individual step. The trick is that we estimate the error using the less accurate improved Euler's method but use the estimates from the more accurate RK3 method. That way we have a safety margin for our estimate of the error.

There are many different Runge-Kutta-Fehlberg methods, all of which involve comparing two different Runge-Kutta approximations to get an estimated error and step-size. The RK2(3) method is a low order method which gives moderately accurate results with only 3 function evaluations at each step. If high accuracy is important, you should use a higher order method. On the other hand, if you just need an answer to within 1% for a problem that doesn't cover too large a range of x values, the RK2(3) method is quick, cheap and effective (with the recommended tolerance of $T = .001$).

Exercises:

1. Why does the slope of the local error vs. step-size line in log scale having about slope 3 for the improved Euler's method and about slope 4 for the R34 method imply assumption 1?
2. Why does the local error of the improved Euler's method being about $C'h^3$ and the local error of the RK3 method being about $C''h^4$ fit with the errors of their approximations for $y(1)$ being about $k'h^2$ and $k''h^3$ respectively as we found in lab 3? (Hint: How many steps of size h does it take to get from $x = 0$ to $x = 1$?)
3. Explain why you are guaranteed that if the estimated error is greater than the tolerance, so you reject the estimate, the revised step-size will be smaller than the previous step-size (i.e. $h_{\text{new}} < h$). If the estimated error is less than the tolerance, so you accept the estimate, will the revised step-size necessarily increase for the next step?
4. Approximate $y(1)$ using the RK2(3) method for $y' = x - y$, $y(0) = 0$ and a tolerance of $T = .01$. You can use whatever initial step-size you want. Then compute the exact answer for the initial value problem and find the percent error in the RK2(3) method.
5. Approximate $y(10)$ using the RK2(3) method for $y' = x - y$, $y(0) = 0$ with a tolerance of

$T = .001$. You may want to program the RK2(3) method into a calculator or computer. Unfortunately, unlike the first extra credit assignment, this method isn't well suited to a spreadsheet because you keep having to test the step-size and possibly change it (perhaps more than once) at each step, though some students have used a spreadsheet successfully anyway.

§12 LONG TERM BEHAVIOR OF SOLUTIONS

Discussion: Now we will examine the qualitative behavior of solutions to first order initial value problems of the form

$$\frac{dP}{dt} = f(P), \quad P(0) = P_0$$

with particular emphasis on their long term behavior. A first order equation of the above form, where the right hand side depends only on the dependent variable, P , and not on the independent variable, t , is called **autonomous**. Our goal is to be able to draw the solution curves just from looking at the function $f(P)$ in the equation

$$\frac{dP}{dt} = f(P).$$

Then in the next couple of sections we will develop and analyze models of population growth using the techniques we develop here.

We start by looking for equilibrium points for the differential equation. An **equilibrium point** for the differential equation $dP/dt = f(P)$ is a value P_0 which satisfies $f(P_0) = 0$.

Suppose P_0 is an equilibrium point for our differential equation. Then $P = P_0$ is a solution to the differential equation. We quickly check that $dP/dt = 0 = f(P)$ since P_0 is a constant and $f(P_0) = 0$. So if our initial value is $P(0) = P_0$, our solution is just a straight line (and P is unchanging, hence in equilibrium). Now since the solution curves don't cross each other (as long as $f(P)$ is continuous), we know that if $P(0) < P_0$, then $P(t) < P_0$ for all t . Similarly if $P(0) > P_0$, then $P(t) > P_0$ for all t . Next by examining the sign of dP/dt at $P(0)$ we can determine whether $P(t)$ is increasing or decreasing. For autonomous equations $P(t)$ will continue to increase or decrease until it encounters an equilibrium point, which will be a barrier since $P(t)$ can't cross an equilibrium point. If no equilibrium point is encountered, then $P(t)$ will increase or decrease without bound. This information is enough to let us determine the long term behavior of $P(t)$.