

5

Numerical Methods

Tools Used in Lab 5

Time Steps
Numerical Methods
Numerical Methods:
Stepsize Scaling

*How does a computer **solve** a differential equation?*

1. Time Steps

As we've mentioned in previous labs, some differential equations cannot be solved analytically. But all is not lost—we can try to draw a picture of the solution, using the slope field as in Lab 2, or, as shown in this lab, we can use a computer to produce an accurate numerical approximation to the solution, *using nothing more than arithmetic!* (There's no need for calculus or even algebra.) The **Time Steps** tool shows how this works.

To keep things simple, we pick a problem where we (who *do* know calculus!) can easily figure out the exact answer, and see if the computer method is clever enough to find a good approximation to it. For instance, suppose this initial value problem is

$$\frac{dx}{dt} = 2t, \text{ subject to the initial condition } x = 0 \text{ at } t = 0.$$

1.1 Find the exact solution to this problem.

The computer tries to solve this by following the slope field. The idea is to step forward in time by a very small amount Δt , and then ask how much x changes. Since the local slope is $\frac{\Delta x}{\Delta t} \approx \frac{dx}{dt}$, the change in x should be $\Delta x \approx \frac{dx}{dt} \Delta t$, which becomes $\Delta x \approx 2t\Delta t$ for our problem. So the rule is: given some point (t, x) , move along the slope field to the new point $(t + \Delta t, x + \Delta x)$, where $\Delta x \approx 2t\Delta t$. Then repeat this process, starting from the new point. Keep marching along the slope

field in this way to trace out the approximate solution. This is the computer's strategy in **Time Steps**. It is called the *Euler method*,

$$x_{n+1} = x_n + \Delta x_n \quad \text{or}$$

$$x_{next} = x_{now} + \Delta x_{now}$$

which is the simplest method for *numerically integrating* a differential equation.

- 1.2** Click on the $\Delta t = 1$ button in **Time Steps**. The computer takes a giant step from $t = 0$ to $t = 1$, following the slope field at $t = 0$. Then at $t = 1$, the computer again computes the local slope, and takes another giant step to $t = 2$.

What is the local slope at $t = 0$? At $t = 1$?

- 1.3** What is the computer's prediction of x when $t = 1$? How close is that to the true answer that you found above? What about $t = 2$?

- 1.4** Redo the numerical integration, but now click on the $\Delta t = \frac{1}{2}$ button in **Time Steps**. In what ways is this approximate solution different from the one produced when $\Delta t = 1$?

- 1.5** Now click on successive smaller step sizes Δt . Why does the approximation improve as Δt decreases?

- 1.6** If the step Δt is smaller than that shown here, can the numerical approximation ever lie *above* the exact solution, given by the parabolic curve? Explain.

2. Comparing Different Numerical Methods

As you've seen above, the Euler method does a pretty lousy job of approximating the solution unless Δt is extremely small. The **Numerical Methods** tool introduces you to some methods that do better. They are trickier to understand, but much more efficient and accurate.

Open up the **Numerical Methods** tool and click on *Forward Euler*. This uses the Euler method to approximate the solution to

$$\frac{dx}{dt} = 3t^2, \text{ subject to the initial condition } x = 0 \text{ at } t = 0.$$

The method is called *Forward Euler* because it uses steps whose slopes are given at the left hand endpoint of every step of length Δt . In contrast, try *Backward Euler*—this uses the slope at the right endpoint of each step.

- 2.1 Suppose we had used $\Delta t = 0.25$ instead of $\Delta t = 0.5$. What value would *Forward Euler* predict for x when $t = 0.5$? Remember that $\Delta x = \left(\frac{dx}{dt}\right)\Delta t$.

Symbolically rewrite Δx so that it applies to this example only:

Now use the following tables to answer the question about what *Forward Euler* will predict, and do the same for *Backward Euler*.

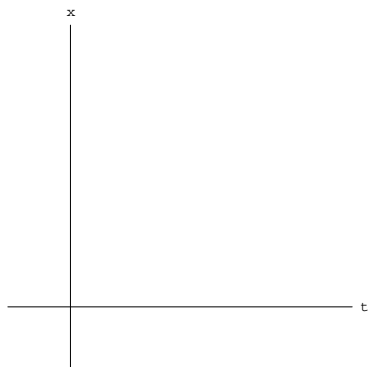
Forward Euler

t	x	Δx	new x

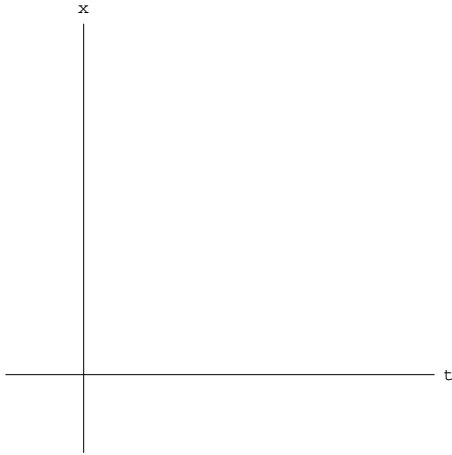
Backward Euler

t	x	Δx	new x

- 2.2 In this example, we see that *Forward Euler* gives an approximation that lies below the true curve, whereas *Backward Euler* lies above. Sketch the graph of a function $x = f(t)$ where the opposite is true.



- 2.3** The graphs suggest that if we could somehow average *Forward Euler* and *Backward Euler*, we'd get a better approximation. The *Trapezoidal (improved Euler)* method takes the slopes predicted by the *Forward Euler* and *Backward Euler* methods, and then averages those slopes. Click on the *Trapezoidal* button to see what that approximation looks like, and sketch the result.



- 2.4** The *Midpoint* method tries a different strategy: it uses the slope at the midpoint of a step, rather than at the left or right endpoint. For the example we are considering, which method produces a generally better approximation, *Midpoint* or *Trapezoidal*?
- 2.5** For the best method so far, we take an unequal mixture of *Midpoint* and *Trapezoidal*. The popular *Runge-Kutta* method (in the simplest case, sometimes called *Simpson's*) is defined as

$$\frac{2}{3} (\text{Midpoint}) + \frac{1}{3} (\text{Trapezoidal}).$$

Why is it plausible that this should be better than simply averaging *Midpoint* and *Trapezoidal* equally?

3. How the Error Depends on the Step Size

Our earlier work with the **Time Steps** tool showed that the Euler method gives an increasingly accurate approximation to the true solution as we decrease the step size Δt . The same trend holds for all the other methods introduced in the **Numerical Methods** tool. That makes sense, since smaller steps always mean a better approximation of the derivative, no matter what method we use. The interesting twist is that some of the methods provide much more “bang for the buck” as the step size is decreased. For instance, we’ll see below that a tenfold decrease in Δt yields about a tenfold increase in accuracy for the Euler method, but a *tenthousandfold* increase in accuracy for the Runge-Kutta method! Our goal now is to investigate this dramatic difference.

Open the **Numerical Methods: Step Size Scaling** tool. To change the step size, click on one of the data points in the graph on the right, and you’ll see that the curves are redrawn in the graphs on the left. Let’s now explain what these graphs are showing.

The top graph on the left shows three curves. The gray curve is the exact solution $x(t) = t^5$ of the equation $\frac{dx}{dt} = 5t^4$, subject to the initial condition $x(0) = 0$. The yellow curve is the approximate solution given by the (forward) Euler method. (When you first open the tool, the default step size is $\Delta t = 1$.) The blue curve is the approximate solution given by the Runge-Kutta method, also with the same step size.

Note that for $\Delta t = 1$ the Runge-Kutta points are almost indistinguishable from the exact solution, whereas the Euler method produces a visible discrepancy at each step.

To compare the methods quantitatively, let’s look at the error, which is defined as the absolute value of the difference between the true solution and the numerical solution at each value of t . The error is plotted as a function of t in the bottom graph on the left. The red and green numbers are the errors at $t = 3$ for the Euler and Runge-Kutta methods, respectively.

3.1 Estimate the value of the error for the Euler method at $t = 1$, $t = 2$, and $t = 3$, using a step size $\Delta t = 1$.

3.2 Explain intuitively why the error increases with time.

Although it is interesting to observe how the error grows with time, we are more concerned with the error as a function of the stepsize Δt . The graph on the right shows a plot of \log_{10} (error) measured at the right-hand endpoint $t = 3$, plotted as a function of $\log_{10}(\Delta t)$. The reason for using logarithms on both axes (known as a log-log plot) is that the data then fall on a straight line, as you can see. The five different data points correspond to different stepsizes, and the red and green colors correspond to the Euler and Runge-Kutta methods, respectively.

To decrease the stepsize, click on one of the data points farther to the left.

- 3.3** What are the five stepsizes used in this plot?
- 3.4** What is the approximate slope of the line corresponding to the Euler data?
- 3.5** What is the approximate slope of the line corresponding to the Runge-Kutta data?
- 3.6** Let E denote the error measured at $t = 3$. Assume that E is related to Δt by a relation of the form $E \approx C(\Delta t)^n$, where C is some positive constant and n is a positive integer. Show that if $\log_{10} E$ is plotted vs. $\log_{10}(\Delta t)$, the graph is a straight line with a slope equal to n .
- 3.7** The slope n is called the **order** of the numerical method. What are the orders of the Euler and Runge-Kutta methods? (You can assume that the data given here are representative of the methods' performance on other examples.)
- 3.8** Show that if we decrease Δt by a factor of 10, the resulting error E decreases by a factor of about 10^n .

Lab 5: Tool Instructions

Time Steps Tool

Buttons

Click on a [Δt] button below the graphing plane to set a value for the time step.

Numerical Methods Tool

Buttons

Click on one of the five buttons below the graphing plane to choose a numerical method.

Click the mouse on the [**Clear**] button to remove all numerical solutions from the graphs.

Numerical Methods: Stepsize Scaling Tool

Step Conditions

Click the mouse in the Stepsize window to choose a stepsize interval from 0.01 to 1.00.

