

CS341 SYSTEMS BIOLOGY I

LECTURE 10

NUMERICAL METHODS FOR SOLVING DIFFERENTIAL EQUATIONS

Goal in general: find $\hat{y}(t_1), \hat{y}(t_2), \dots, \hat{y}(t_{NT})$

for the ODE system $\frac{d\vec{y}(t)}{dt} = f(t, \vec{y})$ and $\vec{y}(0) = y_0$

where $\vec{y}$ is the vector of state variables.

It is common practice in the world of numerical analysis to develop a method for computing $\hat{y}$ with what is called the “test” equation in mind. The “test” equation is simply the ODE model for degradation, so this is something we are already familiar with.

So, let’s keep this in mind.

First, we will consider just one variable

$$\frac{dy(t)}{dt} = -\lambda \cdot y(t)$$

If we start from $y(0) = y_0$, how do we estimate values of y at later time steps? There are many methods to do this, and we are going to spend a couple of weeks on a family of methods called the Runge-Kutta methods. They use information about one timestep along with estimates of the slope of y at that timestep to determine the value of y at the next time step. We will begin with the simplest. It is named after an 18th century Swiss mathematician named Euler (pronounced something like “oiler” - not “ewler” - he isn’t Greek).

First, let’s develop an intuition for how it works. Our goal is to use our knowledge of the derivatives (f) to estimate y over time. But to do that, we will begin by thinking about how we would do the opposite: we will begin by writing down a (forward) finite difference approximation for the derivative. We estimate the slope of a curve $y(t)$ by looking at the difference between two nearby points $y(t)$ and $y(t + \Delta t)$ and dividing by the difference in time Δt :

$$\frac{dy(t)}{dt} \approx \frac{y(t + \Delta t) - y(t)}{\Delta t}$$

1

Then, rearrange and plug in function that defines the actual derivatives.

$$\begin{aligned} y(t + \Delta t) &\approx y(t) + \Delta t \cdot \frac{dy(t)}{dt} \\ &\approx y(t) + \Delta t \cdot f(t, y(t)) \end{aligned}$$

And now we have an approximation of y at a later time step using our function f at the current time step. We can think of it like this: If we know how much protein we have at time t , then we can estimate how much we will have at time $t + \Delta t$ by taking the rate of change of that protein and multiplying it by the time step Δt . Or, if you want to step out of biology for a moment: if we know where we are at time t and how fast we are going, then we can determine where we will be at time $t + \Delta t$.

The equations above are written as approximate equality for the variable y . The forward Euler method rephrases it using equality, but with an approximate of y , which we will call $\hat{y}$. So, the forward Euler method is

$$\hat{y}(t + \Delta t) = \hat{y}(t) + \Delta t \cdot f(t, \hat{y}(t))$$

We apply the forward Euler method to the degradation model in Figure 1, showing where each step is taken. We begin at the given initial condition, then estimate the next position using $f(t, y(0))$. Each subsequent step is taken from an estimate. So, although we have the function defining the derivative, we are not necessarily evaluating it a point on the solution. This is OK, as long as it stays close to the solution (and gets closer over time).

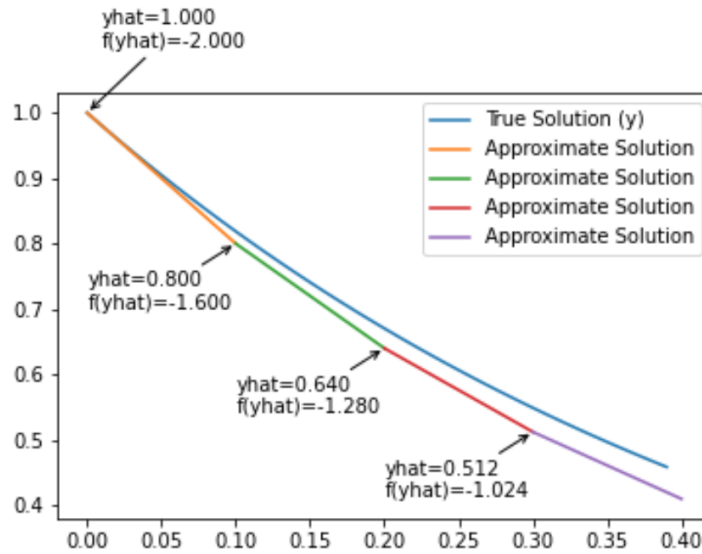


FIGURE 1. Forward Euler method used to estimate the test equation (degradation model) with degradation rate constant 2, initial condition $y(0) = 1$, and a time step of 0.1. We show the true solution (y) and the approximate (which should be $\hat{y}$, but I couldn't figure out how to get that notation in matplotlib, so I wrote it as "yhat") solution as estimated by the forward Euler method.

EFFECT OF STEP SIZE

The choice of step size determines how accurate the solution is. In the Jupyter notebook, we explored several step sizes that provide the correct qualitative answer, and show this trend. We will return to this topic of accuracy several times over the next few weeks.

But what happens when the step is is way too big? The solution goes nuts and the method is considered *unstable*.

We want to know then the forward Euler method will be able to make a qualitatively decent (stable) approximation). We want to know when the approximation won't blow up. I.e. we want to be sure the solution is not increasing over time (actually, we will be even stricter and make sure it is decreasing over time). How can we tell how the time step affects that?

Answer: We examine how the method behaves on the test equation and answer these questions for the test equation. We want to determine under what conditions we will be guaranteed that the solution decreases over time. For the method's solution to decrease over time, we need

$$|\hat{y}(t + \Delta t)| < |\hat{y}(t)|$$

When does that happen with the test equation?

$$\begin{aligned} |\hat{y}(t + \Delta t)| &= |\hat{y}(t) + \Delta t \cdot (-\lambda) \cdot \hat{y}(t)| \\ &= |\hat{y}(t)(1 + \Delta t \cdot (-\lambda))| \end{aligned}$$

Since the ratio of $|\hat{y}(t + \Delta t)|$ to $|\hat{y}(t)|$ must be less than or equal to 1 (the solution is not increasing in magnitude!), we find that

$$|1 + \Delta t \cdot (-\lambda)| < 1$$

which means that $1 + \Delta t \cdot (-\lambda) < 1$ and $1 + \Delta t \cdot (-\lambda) > -1$.

Let's see what that means for Δt in terms of λ .

First, $1 - \lambda\Delta t < 1$ will be true as long as Δt and λ are positive, so that doesn't really constrain us.

$$\begin{aligned}1 - \lambda \Delta t &> -1 \\ -\lambda \Delta t &> -2 \\ \Delta t &> \frac{-2}{\lambda} \\ \Delta t &< \frac{2}{\lambda}\end{aligned}$$

For the example we have been using in the Jupyter notebook, $\lambda = 2$, which means our steps size must be less than $\frac{2}{2} = 1$.