

CS341 SYSTEMS BIOLOGY I

LECTURE 14

NUMERICAL METHODS FOR SOLVING DIFFERENTIAL EQUATIONS (CONT)

In lectures 11 and 12, we implemented the forward Euler method and examined the effects of time step on the simulation output. We observed that if the step is smaller, the error is smaller. And if the method's output is relatively close to the true solution, that error depends linearly on the step size. In lecture 13, we implemented the explicit trapezoidal method, whose error depends quadratically on the step size.

Today, we make a method that can adapt its step size. The idea is that we use one method (the less precise method) to estimate the solution and another method (the more precise method) to estimate the error. If that error estimate is small (relative to a threshold set by the user), then we adjust the step size larger. If that error is large, then we adjust the step size smaller.

ADAPTIVE STEP SIZE METHOD

We will adjust the step size of forward Euler, based on the error estimate from the trapezoidal method.

Recall that the trapezoidal method is

$$\widehat{y}(t + \Delta t) = \widehat{y}(t) + \Delta t \cdot \frac{f(t, \widehat{y}(t)) + f(t + \Delta t, \widehat{y}_{FE})}{2}$$

where $y_{FE}(t + \Delta t)$ is the result from applying the forward Euler method:

$$\widehat{y}_{FE}(t + \Delta t) = \widehat{y}(t) + \Delta t \cdot f(t, \widehat{y}(t))$$

The step size will be determined based on an input parameter, ETOL. We want to ensure that the error (the Euclidean distance between the forward Euler estimate at a given time and the explicit trapezoidal estimate at that same time), is less than ETOL. Note that this is an “absolute” error, and that it makes sense also to control the “relative” error (error relative to the magnitudes of the state variable), but that we will stick with just the “absolute” error for the sake of simplicity.

The key idea is that we build up a list of solutions. We begin from the initial condition, and try one step size. If the error is small enough, then use that step size for the current step. Otherwise, adjust the step size and retry. It may take many adjustments to make a single step. (Ok, this the strategy, but not exactly the algorithm, because we compute the new step size regardless of whether or not the error estimate is small enough. If the error estimate is small enough, we use the new step size for the next step and if it isn't, we use it to retry this step.)

- Initialize h to a small number (e.g 1/100th of the time span (t_0 to t_F)) (Note 1/100th is chosen arbitrarily - you may want to start smaller)
- Initialize lists for y , t , and the step-sizes.
- While the final time in the list of time steps is $< t_F$
 - Compute YFE using step size h with the forward Euler method (the beginning of the step is the last element of the time list and the last element of the y list)
 - Compute YET using step size h with the explicit Trapezoidal method
 - Estimate the error (err) using the Euclidean distance between YET and YFE.
 - If $err \leq ETOL$, then append the time step to the list of time steps, append YFE to the list of solutions, and append h to the list of step sizes.
 - Determine the step size for the next step.

$$h_{new} = h \cdot \sqrt{0.9 \frac{ETOL}{err}}$$

unless $err = 0$, in which case $h_{new} = h$

- If $h_{new} > 2h$ then $h_{new} = 2 * h$ (make sure h doesn't more than double).
- Set h to h_{new} . This will be used either for the next step (if err was small enough) or to retry the current step (but with adjusted size) again.

Why is $h_{new} = h \cdot \sqrt{0.9 \frac{ETOL}{err}}$?

First observe that if $ETOL/err > 1$, then the error is smaller enough that we can allow the step size to grow. But we want to be conservative, so instead of letting it grow to exactly $ETOL$, we use a fraction to keep it a little smaller (0.9). And if $ETOL/err < 1$, then we can probably afford to shrink the step size. We will be conservative here, too and make sure the step size is always a little smaller than we think it could be.

Why the square root? It has to do with what is called the local truncation error of the forward Euler method (mathematics we did not delve into in CS341) and the global error. This is one you are just going to have to trust me on. But if you really want me to say more, I will say the the local truncation error of Forward Euler is $O(h^2)$, so we want to control our error proportionately to h^2 , i.e.

$$\left(\frac{h_{new}}{h}\right)^2 |YET - YFE| \approx 0.9ETOL$$

Should we return YFE or YET at each timestep? Great question! YET is the more accurate solution, so why not use it? For the many set of ODE's, the answer is "you should!". For a particular class of ODEs that are labeled as "stiff", it is not common practice to use the more accurate solution. So I have taken the more conservative approach (and simpler to follow, at least for me) - we are using YET just for error estimates to control the step-size of YFE.