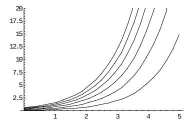
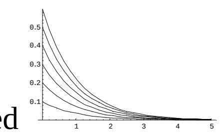
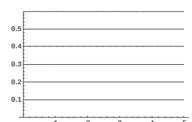
CS205 - Class 16

Readings: 9.3
Covered in Class: 1, 2, 3, 4, 5, 6

ODE's (Continued)

1. Model ODE Problems
   a. Scalar ODE $y' = f(t, y)$ and the
      i. linear model ODE is $y' = \lambda y$ which solution is $y = y_0 e^{\lambda(t-t_0)}$
      ii. Only three kinds of solutions
      iii. $\lambda > 0$ ill-posed unstable 

      iv. $\lambda < 0$ stable, well-posed 

      v. $\lambda = 0$ linearly stable 

   b. Vector ODE $\vec{y}' = \vec{f}(t, \vec{y})$ and the linear model ODE is $\vec{y} = J\vec{y}$. Here is where it gets more interesting as the characterization of the ODE is dependent on the eigenvalues of the Jacobian matrix.
2. Recall the model ODE from last time. $y' = f(t, y)$
   a. We stated that $\lambda > 0$ is ill-posed. But why?
      i. Errors accumulated and they increase exponentially.
3. (Forward) **Euler's Method** $\frac{y_{k+1} - y_k}{h} = f(t_k, y_k)$ or $y_{k+1} = y_k + hf(t_k, y_k)$
   a. **Accuracy**, truncation error usually dominates round-off error in ODE's.
   b. **Local truncation error** $y_{k+1} = y_k + hf(t_k, y_k) + O(h^2)$
      i. y_{k+1} is calculated by ignoring the $O(h^2)$ term.
      ii. If y_0 is exact, the error in y_1 is $O(h^2)$.
   c. **Global truncation error** integrating from $t = t_o$ to $t = t_{final}$ with $n = O(1/h)$ steps gives a total error of $O(nh^2) = O(h)$.
      i. Euler's method is *1[st] order accurate* with $\frac{y_{k+1} - y_k}{h} = f(t_k, y_k) + O(h)$
   d. For **stability** consider the model equation $y' = \lambda y$ where $\lambda < 0$
      i. For a general ode λ is df/dy or an eigenvalue of the Jacobian matrix
      ii. Euler's method applied to the model equation is
      $y_{k+1} = y_k + h\lambda y_k = (1 + h\lambda) y_k$

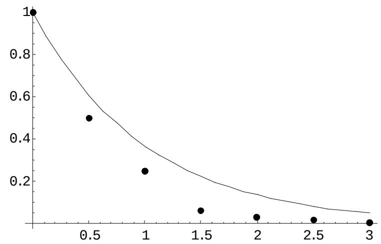
iii. So $y_k = (1+h\lambda)^k y_o$ and the error shrinks when $|1+h\lambda|<1$

1. Thus, $-2 < h\lambda < 0$ is needed for stability

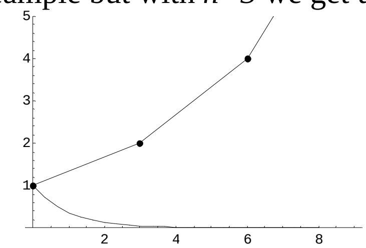
e. Example

i. Forward Euler on $y' = -y$ for $y_0 = 1$, $t_0 = 0$. Stability is h<2

ii. h=.5 is stable but truncation errors cause it to be smaller

1. 

iii. Same eample but with *h=3* we get unstable

1. 

4. As an aside, stability restriction related to your ability to get accuracy. Consider a large lambda and you also have another eigenvalue that is smaller. You need a small time step for the large eigenvalue. For example if you had y = c1 y1 + c2 y2. If they differ by a lot you get a stiff problem. For stiff problems you want a method with no stability requirement.