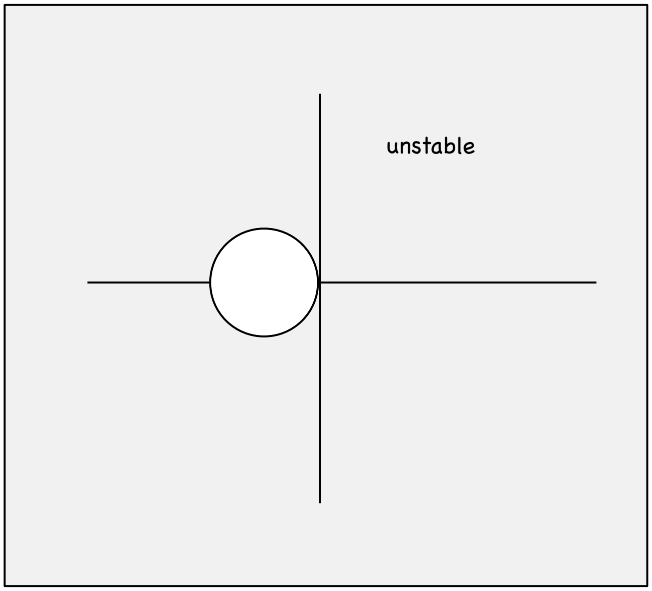
AM111 Lectures **Sections** **Office Hours**

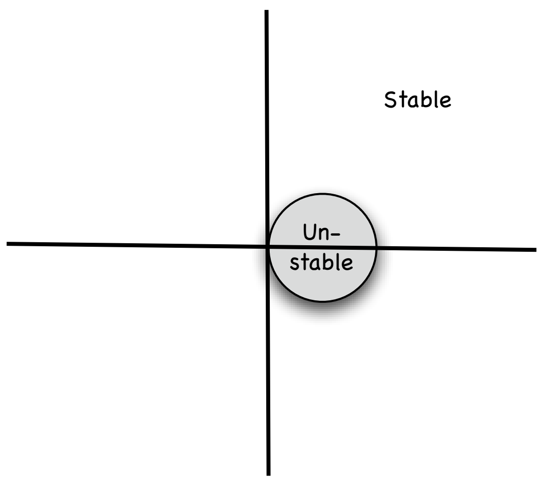
- ☐ Week 5
- ☐ Week 6
- ☐ Week 7
- ☐ Week 8
- ☐ Week 9 Spring Break!
- ☐ Week 10
  - ☐ *Apr. 3rd*
  - ☐ *Apr. 4th*
    - ☐ Numerical Solutions of Ordinary Differential Equations (ODEs)
      - ☐ Initial Value Problems
      - ☐ $\frac{d\vec{y}(t)}{dt} = \vec{f}(t, \vec{y}(t))$
      - ☐ Example 1: Compound Interest
        - ☐ $\frac{dy}{dt} = ry$
        - ☐ $y(t=0) = y_0$
        - ☐ $y(t) = y_0 e^{rt}$
      - ☐ Example 2: single pendulum
        - ☐ $\frac{d^2\theta}{dt} = -\frac{g}{L}\sin\theta$
        - ☐ $\theta(t=0) = a, \frac{d\theta(t=0)}{dt} = b$
        - ☐ $y_1 = \theta$
        - ☐ $y_2 = \frac{d\theta}{dt} = \dot{\theta}$
        - ☐ so now
        - ☐ $\frac{dy_1}{dt} = y_2, \text{and } \frac{dy_2}{dt} = -\frac{g}{L}\sin y_1$
      - ☐ Example 3: # of rabbits
        - ☐ r = # of rabbits
        - ☐ f = # of foxes
        - ☐ $\frac{dr}{dt} = 2r - 2rf$
        - ☐ $\frac{df}{dt} = -f + 2rf$
        - ☐ $r(0) = r_0, \text{and } f(0) = f_0$
        - ☐ This is the Lotta-Volterra predator-prey model
        - ☐ This is a nonlinear equation!
      - ☐ Example 4: Chemical Reactions
        - ☐ Part of Ozone Chemistry is:
        - ☐ $O_3 + O_2 \leftarrow^{k_2}\rightarrow_{k_1} O + 2O_2$
        - ☐ $O_3 + O \rightarrow_{k_3} 2O_2$
        - ☐ Defining $y_1 = [O_3], y_2 = [O], y_3 = [O_2]$
        - ☐ $\dot{y}_1 = -k_1 y_1 y_3 + k_2 y_2 y_3^2 - k_3 y_1 y_2$
        - ☐ $\dot{y}_2 = k_1 y_1 y_3 - k_2 y_2 y_3^2 - k_3 y_1 y_2$
        - ☐ $\dot{y}_3 = k_1 y_1 y_3 - k_2 y_2 y_3^2 + 2k_3 y_1 y_2$
      - ☐ Numerical solution:

- ☐ Generate a sequence of values for t, $\{t_0, t_1, \ldots, t_n, \ldots\}$, and a corresponding sequence for the dependent variable $\{y_0, y_1, \ldots, y_n, \ldots\}$ s.t. $y_n \simeq y(t_n)$
- ☐ Let's take a constant step size:
- ☐ $t_{n+1} - t_n = h \rightarrow t_n = nh + t_0$
- ☐ $\frac{dy}{dt} = f(t, y)$
- ☐ recall that:
  - ☐ $\left.\frac{dy}{dt}\right|_{t_n} = \lim_{t_{n+1} \rightarrow t_n} \frac{y_{n+1} - y_n}{t_{n+1} - t_n}$
- ☐ So that
  - ☐ $f(t_n, y_n) = \frac{y_{n+1} - y_n}{\Delta t} \rightarrow y_{n+1} = y_n + f(t_n, y_n)\Delta t$
  - ☐ This is Euler's Method.
- ☐ Check for the case of $\frac{dy}{dt} = y$.
  - ☐ Notice that the result is only approximate. We have errors (discretization errors) at each step.
- ☐ Another way to solve this involves another way to take a derivative:
  - ☐ $\left.\frac{dy}{dt}\right|_{t_{n+1}} = \lim_{t_n \rightarrow t_{n+1}} \frac{y_n - y_{n+1}}{t_n - t_{n+1}}$
  - ☐ so that
  - ☐ $y_{n+1} - f(t_{n+1}, y_{n+1})\Delta t = y_n$
  - ☐ This is the Backward Euler Method, and is an *implicit* method.

- ☐ A method is called *explicit* if y_{n+1} can be computed <u>directly</u> in terms of the previous values of $y_k, k \leq n$.
- ☐ A method is called *implicit* if y_{n+1} depends implicitly upon itself through $f(t, y)$.
- ☐ The backward Euler method can be found through the familiar process of zero finding! We simply want to find the zero of $y_{n+1} - y_n - f(t_{n+1}, y_{n+1})\Delta t$
- ☐ We now check out the case of $\lambda = i$. Looking at the real part, we find that the exact solution is a nice sinusoid, but that our Euler Method solution is a growing sinusoid, and that the Backward Euler Method produces a damped sinusoid. What's up with that?
  - ☐ Using the forward Euler method, we find that $y_n = (1 + \Delta t\lambda)^n y_0$
  - ☐ $\frac{|y_n|}{|y_0|} = |1 + \Delta t\lambda|^n$
  - ☐ For $\lambda = i, |1 + \Delta t i| = \sqrt{1 + (\Delta t)^2} > 1$
  - ☐ Defining $z = \Delta t\lambda$, we find that the above quantity is only less than 1 in the unit circle centered at z=-1.
  - ☐ When |1+z|>1, then $|y_n| \rightarrow \infty$ as $n \rightarrow \infty$

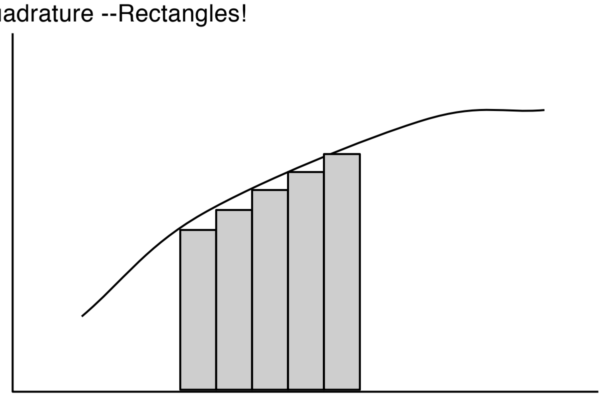
- 

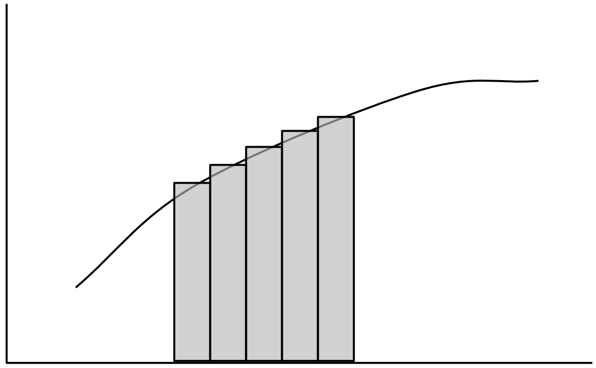
- For the backward Euler Method, $y_n = y_0(1-\Delta t\lambda)^{-n}$, which is going to blow up for all λ inside the unit circle centered on +1.

- 

- What can we figure out about the solution from the local behavior of f(t,y) near t_c, y_c?
  - $f(t,y) = f(t_c,y_c) + \alpha(t-t_c) + J(y-y_c) + \ldots$
    - $\alpha = \frac{\partial f}{\partial t}(t_c,y_c)$
    - $J = \frac{\partial f}{\partial y}(t_c,y_c)$
    - J is the jacobian matrix.
    - The local behavior of $\frac{dy}{dt} = f(t,y)$ near t_c, y_c can be approximated by $\frac{d\vec{y}}{dt} = J\vec{y}$ if we ignore the α term.
    - $J = V\Lambda V^{-1}$
    - $\Lambda =$ diagonal eigenvalue matrix, and $V\vec{x} = \vec{y}$, $\frac{d\vec{x}}{dt} = \Lambda\vec{x}$ (V is an eigenvector matrix)
  - So let's go back to our pendulum problem. The jacobian matrix is now

- ☐ $J = \begin{pmatrix} 0 & 1 \\ -\frac{g}{L}\cos y_1 & 0 \end{pmatrix}$
- ☐ whose eigenvalues are: $\pm\sqrt{-\frac{g}{L}\cos y_1}$

- ☐ What about accuracy?
  - ☐ Definition: Local Discretization Error
    - ☐ $\varepsilon_{k+1} = y(t_{k+1}) - [y(t_k) + \Delta t \phi]$
    - ☐ where $y(t_{k+1}), y(t_k)$ are the exact solutions at t_{k+1} and t_k, respectively, and \phi is the one-step approximation over the time interval $t \in [t_k, t_{k+1}]$.
    - ☐ i.e. $\phi = f(t_n, y(t_n))$
  - ☐ Definition: Global Discretization Error
    - ☐ $E_k = y(t_k) - y_k$, where y_k is the approximate solution.
  - ☐ Example: Forward Euler
    - ☐ $y(t_k + \Delta t) = y(t_k) + \left.\frac{dy(t)}{dt}\right|_{t_k} \Delta t + \frac{1}{2}\frac{d^2 y(c_k)}{dt^2}\Delta t^2$
      - ☐ $c_k \in [t_k, t_k + \Delta t]$
    - ☐ $y(t_{k+1}) = y(t_k) + f(t_k, y_k(t))\Delta t + \frac{1}{2}\frac{d^2 y(c_k)}{dt^2}(\Delta t)^2$
    - ☐ Local Error: $\varepsilon_k = \frac{(\Delta t)^2}{2}\frac{d^2 y(c_k)}{dt^2} \sim O(\Delta t^2)$
    - ☐ Global Error: Consider integration over a specified time interval $t \in [a, b]$. We have K steps, so that $\Delta t K = b - a$.
      - ☐ $E_K = \sum_{j=1}^{K} \frac{\Delta t^2}{2}\frac{d^2 y(c_j)}{dt^2} \simeq \frac{\Delta t^2}{2} C K = \frac{\Delta t}{2} C (b-a) \sim O(\Delta t)$
  - ☐ Next time, we'll look at quadrature methods.
  - ☐ Notice that the forward Euler method is essentially just the simplest form of quadrature --Rectangles!
    - ☐ 
  - ☐ Backwards Euler is just about the same.

- ☐ 

- ☐ *Apr. 6th*
  - ☐ Last time...
    - ☐ We went over initial value problems for ODEs.
      - ☐ $\frac{d\vec{y}(t)}{dt} = f(t, \vec{y}(t))$
      - ☐ $\vec{y}(t) = \vec{y}_0$
    - ☐ Euler's Method
      - ☐ Forward (explicit)
      - ☐ Backward (implicit)
    - ☐ Stability
      - ☐ Forward Euler
      - ☐ Backward Euler
    - ☐ Accuracy
  - ☐ How can we get better accuracy?
    - ☐ Given $y(t_i)$, how do we get $y(t_i + \Delta t)$ accurately?
    - ☐ Taylor Methods
      - ☐ Suppose y has n continuous derivatives on the interval [a,b], and y^(n+1) exists on [a,b], and that $t_i, t_i + \Delta t \in [a,b]$.
      - ☐ Then there exists a number $c \in (t_i, t_i + \Delta t)$ s.t.
        - ☐ $y(t_i + \Delta t) = P_n + R_n$
        - ☐ Where $P_n = \sum_{k=0}^{n} \frac{y^{(k)}(t_i)}{k!} (\Delta t)^k$ is the nth Taylor Polynomial.
        - ☐ and $R_n = \frac{y^{(n+1)}(c)}{(n+1)!} (\Delta t)^{n+1}$ is the residual.
      - ☐ We will use P_n to approximate $y(t_i + \Delta t)$. This is the Taylor method of order n.
      - ☐ For n=1
        - ☐ $y(t_i + \Delta t) = y(t_i) + \frac{dy(t_i)}{dt} \Delta t = y(t_i) + f(t_i, y(t_i))$
        - ☐ This is the Euler method. The error scales as order 1 in Δt.
      - ☐ For n=2
        - ☐ $y(t_i + \Delta t) = y(t_i) + \frac{dy(t_i)}{dt} \Delta t + \frac{1}{2} \frac{d^2 y(t_i)}{dt^2} (\Delta t)^2$
        - ☐ $= y(t_i) + f(t_i, y(t_i)) + \frac{1}{2} \frac{d f(t_i, y(t_i))}{dt} (\Delta t)^2$
        - ☐ This is a tad messy. We want higher order methods but we don't want to have to take higher order derivatives.
    - ☐ Different approach -- Numerical Quadrature

- ☐ $\frac{dy}{dt} = f(t,y)$
- ☐ $y(t_i + \Delta t) = y(t_i) + \int_{t_i}^{t_i+\Delta t} f(s, y(s))ds$
- ☐ If the function f(t,y) does not depend upon y, then
- ☐ $y(t_i + \Delta t) = y(t_i) + \int_{t_i}^{t_i+\Delta t} f(s)ds$
- ☐ Recall that the Euler method is equivalent to evaluating this integral as sum of rectangles.
- ☐ One way to improve it would be to use the midpoint method to estimate the height of these rectangles.
  - ☐ $f(t + \frac{\Delta t}{2})\Delta t$
  - ☐ This yields an error of order $O(\Delta t^2)$
  - ☐
  - ☐ The midpoint rule would yield for y_{n+1}
  - ☐ $y_{n+1} = y_n + f(t_n + (\Delta t/2), y(t_n + (\Delta t/2)))\Delta t$
  - ☐ We approximate $y(t_n + \frac{\Delta t}{2})$ by the Euler method:
  - ☐ $s_1 = f(t_n, y_n)$
  - ☐ $s_2 = f(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2}s_1)$
  - ☐ $y_{n+1} = y_n = \Delta t s_2$
  - ☐ This is 2nd-order Runge-Kutta.
- ☐ We could also use the trapezoidal quadrature.
  - ☐ $\frac{\Delta t}{2}[f(t) + f(t + \Delta t)]$
  - ☐ This also yields an error of order $O(\Delta t^2)$
  - ☐ Trapezoidal Rule:
  - ☐ $y_{n+1} = y_n + \frac{\Delta t}{2}[f(t_n, y_n) + f(t + \Delta t, y(t_n + \Delta t)]$
    - ☐ We can approximate $y(t_n + \Delta t)$ by y_{n+1} or estimate using Euler's Method.
    - ☐ Choice #1, The Trapezoidal (or Crank-Nicolson) method
      - ☐ $y_{n+1} = y_n + \Delta t \frac{f(t_n, y_n) + f(t_n + \Delta t, y_{n+1})}{2}$
    - ☐ Choice #2, Heun's Method.
      - ☐ $s_1 = f(t_n, y_n)$
      - ☐ $s_2 = f(t_n + \Delta t, y_n + \Delta t s_1)$
      - ☐ $y_{n+1} = y_n + \Delta t \frac{s_1 + s_2}{2}$
    - ☐ Choice #1 is an implicit method, while choice #2 is an explicit method.
- ☐ We could also consider Simpson's rule:
  - ☐ $\frac{\Delta t}{6}\left[f(t) + 4f\left(t + \frac{\Delta t}{2}\right) + f(t + \Delta t)\right]$
  - ☐ which is a fourth order method $O(\Delta t^4)$

- ☐ Before we move on, is it worth the effort? We have to do extra work to use these methods instead of Euler's method.
  - ☐ Decreasing the step size by a factor of 10:
    - ☐ For 1st order methods, we use 9 times more evaluations, and the error drops by a factor of 10.
    - ☐ For 2nd order methods, we use 18 times more evaluations, and the error drops by a factor of 100.
- ☐ 4th order Runge-Kutta method:
  - ☐ $h = \Delta t$
  - ☐ $s_1 = f(t_n, y_n)$
  - ☐ $s_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2} s_1)$
  - ☐ $s_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2} s_2)$
  - ☐ $s_4 = f(t_n + h, y_n + h s_3)$
  - ☐ $y_{n+1} = y_n + \frac{h}{6}(s_1 + 2s_2 + 2s_3 + s_4)$
  - ☐ This is $O(\Delta t^4)$
  - ☐ If the function f(t,y) does not depend upon y, then $s_2 = s_3$, and
  - ☐ $y_{n+1} = y_n + \frac{h}{6}(s_1 + 4s_2 + s_4)$
  - ☐ Simpson's rule!
- ☐ How do people come up with these things?
  - ☐ A general single-step method is characterized by a number of parameters: $\alpha_i, \beta_{i,j}$, and γ_i.
  - ☐ There are k stages. Each stage computes a slope s_i by evaluating f(t,y) for a particular set of values of t and y, obtained by a linear combination of the previous slopes:
  - ☐ $s_i = f\left(t_n + \alpha_i h, y_n + h\sum_{j=1}^{i-1}(\beta_{i,j} s_j)\right)$
  - ☐ These slopes are then combined to produce our estimate for y_{n+1}
  - ☐ $y_{n+1} = y_n + h\sum_{i=1}^{k}\gamma_i s_i$
  - ☐ For k=2, we have already seen two-stage single-step methods:
  - ☐ $s_1 = f(t_n, y_n) \rightarrow \alpha_1 = 0$
  - ☐ $s_2 = f(t_n + \alpha_2 h, y_n + h\beta_{2,1} s_1))$
    - ☐ Note that if β has diagonal elements, then we have an implicit method (there exists an i for which s_i is equal to a linear combination which includes itself..)
  - ☐ $y_{n+1} = y_n + h(\gamma_1 s_1 + \gamma_2 s_2)$
  - ☐ Upon Taylor expanding s_2 around t_n, y_n, we have
  - ☐ $s_2 = f(t_n, y_n) + \frac{\partial f}{\partial t}\alpha h + \frac{\partial f}{\partial y}\beta s_1 h + O(h^2)$
  - ☐ so
    $$y_{n+1} = y_n + h(\gamma_1 + \gamma_2) f(t_n, y_n) + \gamma_2\left[\alpha\frac{\partial f}{\partial t} + \beta\frac{\partial f}{\partial y} f(t_n, y_n)\right] h^2 + O(h^3)$$

- ☐ The Taylor expansion of $y(t_n+h)$ at t_n is:
- ☐ $y(t_n+h)=y(t_n)+hf(t_n,y(t_n))+\frac{h^2}{2}\frac{\partial f}{\partial t}+\frac{h^2}{2}\frac{\partial f}{\partial y}f(t_n,y(t_n))+O(h^3)$
- ☐ Comparing these, letting $y_n=y(t_n)$, we find that we must have
- ☐ $\gamma_1+\gamma_2=1$
- ☐ $\gamma_2\alpha=1/2$
- ☐ $\gamma_2\beta=1/2$
- ☐ Thus the method has almost been determined. We are free to choose one of the parameters however we like.
- ☐ If we choose $\gamma_1=0$, then $\gamma_2=1,\alpha=\beta=1/2$
  - ☐ $s_1=f(t_n,y_n)$
  - ☐ $s_2=f(t_n+h/2,y_n+hs_1/2)$
  - ☐ $y_{n+1}=y_n+s_2h$
  - ☐ This is the 2nd Order Runge-Kutta method.
- ☐ If we choose $\gamma_1=1/2$, then $\gamma_2=1/2,\alpha=\beta=1.$
  - ☐ $s_1=f(t_n,y_n)$
  - ☐ $s_2=f(t_n+h,y_n+hs_1)$
  - ☐ $y_{n+1}=y_n+\frac{s_1+s_2}{2}h$
  - ☐ This is Heun's method.
- ☐
- ☐ For arbitrary Multi-stage single-step methods:
  - ☐ # of calculations per step: 2, 3, 4, $5\le n\le 7$, $8\le n\le 9$, $n>10$
  - ☐ Best possible error:

    $O(h^2)$, $O(h^3)$, $O(h^4)$, $O(h^{n-1})$, $O(h^{n-2})$ $O(h^{n-3})$
- ☐ Error control (adaptive stepsize)
  - ☐ Basic idea: If we know the error is $\propto h^n$, and we know how big this error is for a particular h, then we can find a new h^* for which the error is smaller than a specified tolerance.
  - ☐ How do we estimate the error? If we have access to the exact solution, it's pretty easy, but generally this option isn't available. Another way to go is to use a higher-order method in place of the exact solution to estimate the error of the lower-order method.
  - ☐ If we have 2 methods:
  - ☐ $\tilde{e}=y(t_{i+1})-\tilde{y}_{i+1}$ $\quad O(h^{n+2})$
  - ☐ $e=y(t_{i+1})-y_{i+1}$ $\quad O(h^{n+1})$
  - ☐ $e=\tilde{e}+\tilde{y}_{i+1}-y_{i+1}\simeq\tilde{y}_{i+1}-y_{i+1}$ since $\tilde{e}\ll e$
  - ☐ $e\sim O(h^{n+1})$.
  - ☐ Given an error tolerance ε, we want to change the step size from $h\to qh$ so that the new error $|q^ne|<\varepsilon$. Thus making $q<\left|\frac{1}{\tilde{y}_{i+1}-y_{i+1}}\right|^{1/n}$.
- ☐ *Apr. 7th*
- ☐ Week 11
- ☐ Week 12

AM111 Lectures **Sections** **Office Hours**

- ☐ Week 13
- ☐ Week 14