# CS 450: Numerical Anlaysis[1]

## Initial Value Problems for Ordinary Differential Equations

University of Illinois at Urbana-Champaign

[1] *These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book "Scientific Computing: An Introductory Survey" by Michael T. Heath (slides).*

# Ordinary Differential Equations

- An *ordinary differential equation (ODE)* usually describes time-varying system by a function $\boldsymbol{y}(t)$ that satisfies a set of equations in its derivatives.

- An ODE of any *order* $k$ can be transformed into a first-order ODE,

# Example: Newton's Second Law

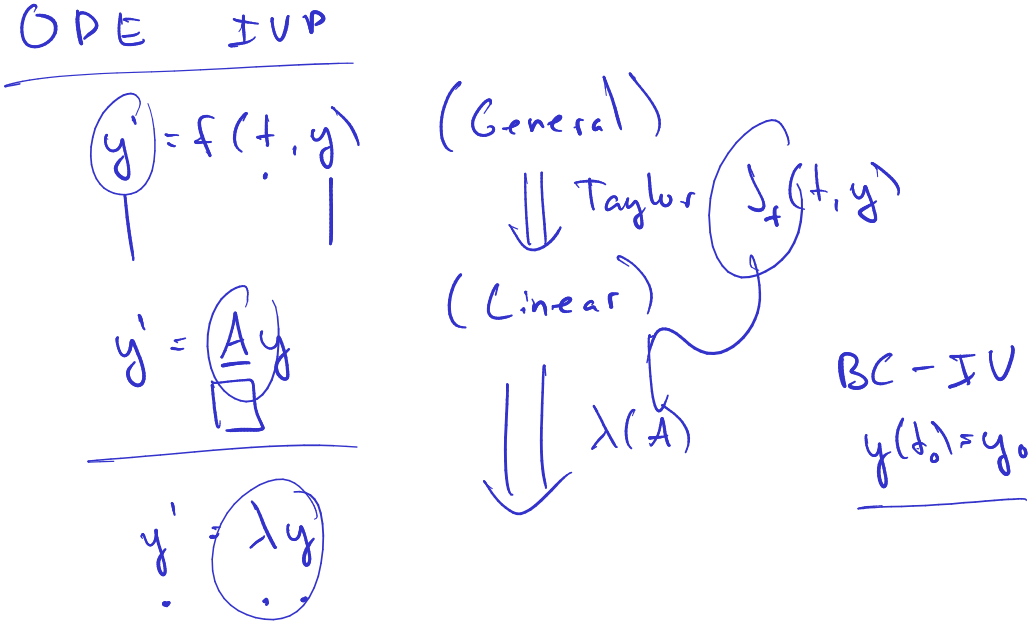
- $F = ma$ corresponds to a second order ODE,

- We can transform it into a first order ODE in two variables:

# Initial Value Problems

- Generally, a first order ODE specifies only the derivative, so the solutions are non-unique. An *initial condition* addresses this:

- Given an initial condition, an ODE must satisfy an integral equation for any given point $t$:


ODE IVP
y' = f(t, y)
y' = Ay
y' = λy
(General)
Taylor
(Linear)
λ(A)
J_f(t, y)
BC - IV
y(t_0) = y_0

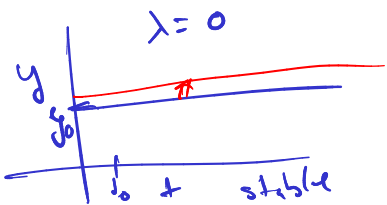
# Existence and Uniqueness of Solutions

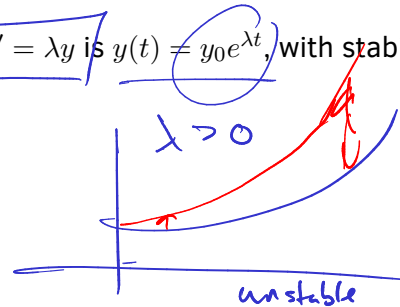
- For an ODE to have a unique solution, it must be defined on a closed domain $D$ and be *Lipschitz continuous*:

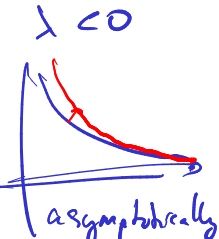
- The solutions of an ODE can be stable, unstable, or asymptotically stable:

# Stability of 1D ODEs

- The solution to the scalar ODE $y' = \lambda y$ is $y(t) = y_0 e^{\lambda t}$, with stability dependent on $\lambda$:



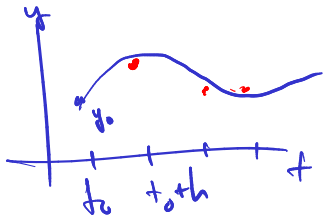

- A constant-coefficient linear ODE has the form $\boldsymbol{y' = Ay}$, with stability dependent on the real parts of the eigenvalues of $\boldsymbol{A}$:

# Numerical Solutions to ODEs

$y'(t) = f(t, y)$

**Demo:** *Forward Euler stability*

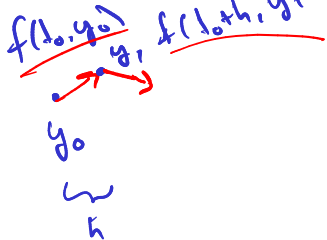
- Methods for numerical ODEs seek to approximate $\boldsymbol{y}(t)$ at $\{t_k\}_{k=1}^{m}$.



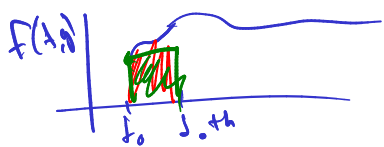

$$y(t_0+h) = y(t_0) + \int_{t_0}^{t_0+h} f(t, y)\, dt$$

$y_0$

- Euler's method provides the simplest method (attempt) for obtaining a numerical solution:



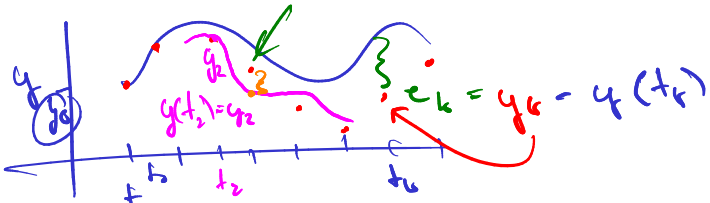

$$y_{k+1} = y_k + h_k f(t_k, y_k)$$



# Error in Numerical Methods for ODEs

- Truncation error is typically the main quantity of interest, which can be defined *globally* or *locally*:

local error?




- The *order of accuracy* of a given method is one less than than the order of the leading order term in the local error $l_k$:

- accuracy is of order $p$ if $\|l_k\| = O(h^{p+1})$

$$y(t_{k+1}) = y(t_k) + h f(t_k, y(t_k)) + O(h^2)$$

- forward Euler is a 1st order method

$\frac{h^2}{2} y''(t_k, y(t_k)) + \dots$

# Accuracy and Taylor Series Methods

- By taking a degree-$r$ Taylor expansion of the ODE in $t$, at each consecutive $(t_k, \boldsymbol{y}_k)$, we achieve $r$th order accuracy.

$$y(t_k + h) = y(t_k) + y'(t_k)h + \cdots + y^{(r)}(t_k)h^r/r! + O(h^{r+1})$$

- Taylor series methods require high-order derivatives at each step:

- finite differencing or other ....

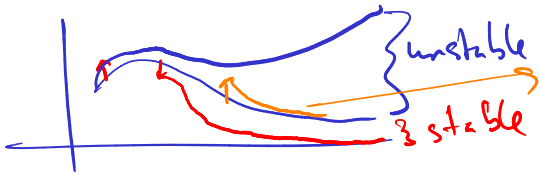
$(t_k, y_k) \Rightarrow$ multi-stage methods

$(t_{k-r}, y_{k-r}) \cdots (t_{k-1}, y_{k-1}) \Rightarrow$ multi-step methods

*Demo: Stability regions*

# Growth Factors and Stability Regions

- Stability of an ODE method discerns whether local errors are amplified, deamplified, or stay constant:




stability region:
values $h\lambda$ for which method is stable

- Basic stability properties follow from analysis of linear scalar ODE, which serves as a local approximation to more complex ODEs.

$y' = \lambda y$

$y_{k+1} = y_k + h\lambda y_k = \underbrace{(1 + h\lambda)}_{\text{growth factor}} y_k$

$e_k = \ell_k + (1 + h\lambda) e_{k-1}$

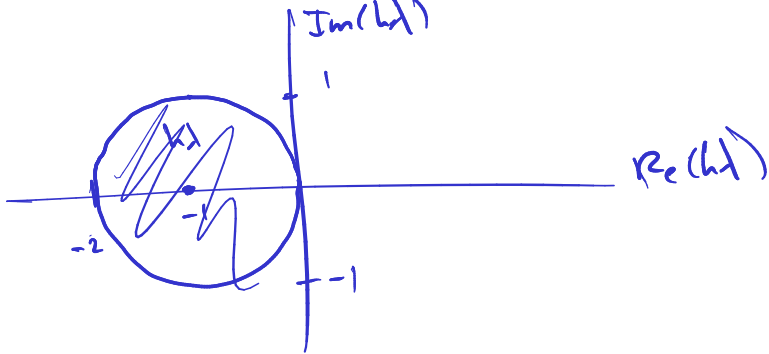
$|\text{growth factor}| \leq 1$

$1 + h\lambda$

$\Downarrow$

$-2 \leq h\lambda \leq 0$

$|1+h\lambda| \leq 1$ Stability region $h > 0$



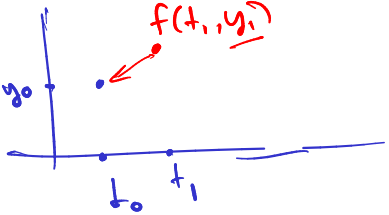
# Stability Region for Forward Euler

- The stability region of a general ODE constrains the eigenvalues of $h\boldsymbol{J_f}$

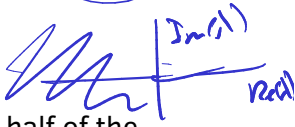
# Backward Euler Method

*Demo:* Backward Euler stability
*Activity:* Backward Euler Method

- Implicit methods for ODEs form a sequence of solutions that satisfy conditions on a local approximation to the solution:




$$y_{k+1} = y_k + h f(t_{k+1}, y_{k+1})$$




- The stability region of the backward Euler method is the left half of the complex plane:

$y' = \lambda y$

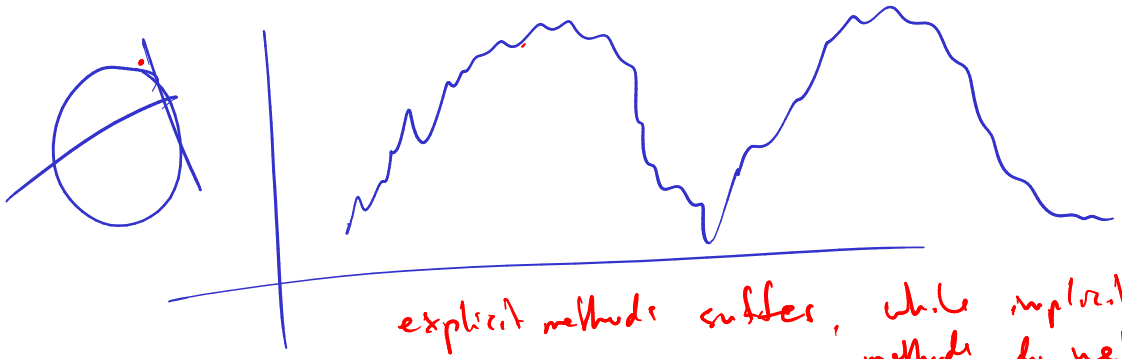
$|y_{k+1}| \le 1$, $h\lambda \le 0$

$y_{k+1} = y_k + h\lambda y_{k+1}$

$(1 - h\lambda) y_{k+1} = y_k \Rightarrow y_{k+1} = \frac{1}{(1-h\lambda)} y_k$

# Stiffness

- *Stiff* ODEs are ones that contain components that vary at disparate time-scales:



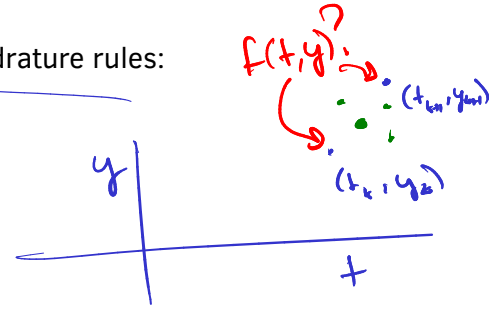
# Trapezoid Method

- A second-order accurate implicit method is the *trapezoid method*

$$y_{k+1} = y_k + \frac{h}{2}\left(f(t_k, y_k) + f(t_{k+1}, y_{k+1})\right)$$

$$\frac{|1 + h\lambda/2|}{|1 - h\lambda/2|} \leq 1 \quad \text{for} \quad h\lambda \leq 0$$

- Generally, methods can be derived from quadrature rules:

# Multi-Stage Methods

- *Multi-stage methods* construct $y_{k+1}$ by approximating $y$ between $t_k$ and $t_{k+1}$:

• Heun's method

$$y_{k+1} = y_k + h\left[f(t_k, y_k)/2 + f(t_k + h, y_k + hf(t_k, y_k))/2\right]$$

$v_1$


- The 4th order Runge-Kutta scheme is particularly popular:
  *This scheme uses Simpson's rule,*

$$\begin{aligned}
\boldsymbol{y}_{k+1} &= \boldsymbol{y}_k + (h/6)(\boldsymbol{v}_1 + 2\boldsymbol{v}_2 + 2\boldsymbol{v}_3 + \boldsymbol{v}_4) \\
\boldsymbol{v}_1 &= \boldsymbol{f}(t_k, \boldsymbol{y}_k), & \boldsymbol{v}_2 &= \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + (h/2)\boldsymbol{v}_1), \\
\boldsymbol{v}_3 &= \boldsymbol{f}(t_k + h/2, \boldsymbol{y}_k + (h/2)\boldsymbol{v}_2), & \boldsymbol{v}_4 &= \boldsymbol{f}(t_k + h, \boldsymbol{y}_k + h\boldsymbol{v}_3).
\end{aligned}$$

# Runge-Kutta Methods

- Runge-Kutta methods evaluate $\boldsymbol{f}$ at $t_k + c_i h$ for $c_0, \ldots, c_r \in [0, 1]$,

- A general family of Runge Kutta methods can be defined by

# Multistep Methods

- *Multistep methods* employ $\{\boldsymbol{y}_i\}_{i=0}^{k}$ to compute $\boldsymbol{y}_{k+1}$:

- Multistep methods are not self-starting, but have practical advantages: