ME751
Advanced Computational Multibody Dynamics

October 12, 2016

Dan Negrut
University of Wisconsin-Madison

Before we get started…

- Last Time:
  - Simple example: deriving EOM for one body connected to ground via spherical joint
    - That concluded Dynamics Analysis of systems of rigid bodies. Flex body: coming up, in 10 days
  - Inverse Dynamics Analysis
  - Equilibrium Analysis

- Today:
  - Elements of the numerical solution of Initial Value Problems

- Reading:
  - Chapter 3 ("Basic Methods, Basic Concepts") of Asher and Petzold: *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, 1998

simEngine3D vs. Chrono
+
Final Project Issues

- Current assignment has MATLAB component
  - Everybody thus gets exposed to implementing support for two GCons in MATLAB

- In future assignments you can take one of two paths
  - `simEngine3D` path, implementing in MATLAB your own dynamics engine
  - `Chrono` path – assignments call for modeling four mechanisms (see slides of Sept. 28)
    - No more MATLAB stuff, you'll move to C++ and Chrono ⇨ deal w/ four mechanisms in Chrono

- Final project
  - If in future assignments you go the `simEngine3D` way:
    - I encourage you to make it be your Final Project
      - `simEngine3D` needs to be capable of simulating two benchmark problems (see discussion of 9/28)
    - However, it's ok if you choose a different Final Project (discuss topic with me first)

  - If you go the Chrono way, you'll have to come up with a topic for your Final Project
    - You'll need to discuss the topic w/ me first

ODE vs. IVP

- What's the difference between ODE and IVP?

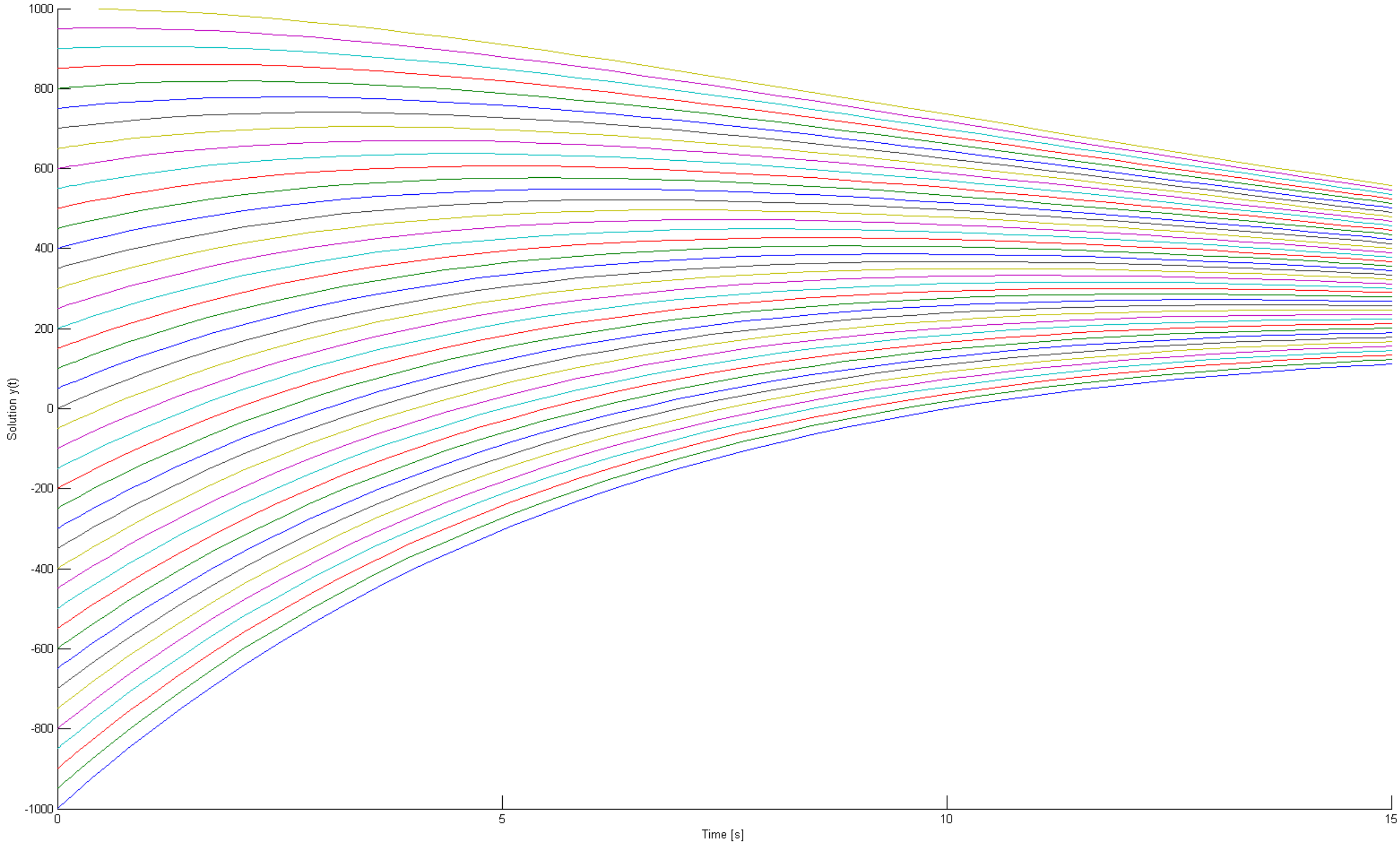
- Ordinary Differential Equation (ODE)
  - Typically, has an infinite number of solutions

- Initial Value Problem (IVP)
  - Is an ODE **plus** an initial condition (IC):
    - The IC: The unknown function assumes at time T=0 a certain prescribed value

  - The solution for the IVP's that we'll deal with is **UNIQUE**
    - We'll assume that $f(t, y)$ is well behaved (Lipschitz continuous)

ODE: Infinite Number of Solutions

- ODE Problem: $\dot{y}(t) = -0.1y(t) + 100e^{-0.1\,t}$
- A range of Initial Conditions (ICs) is specified: $y_0 = [-1000 : 50 : 1000]$



ODE vs. IVP
[Cntd.]

- Remember:

  1. **IVP = ODE + IC**
  2. **IVP has a UNIQUE solution (unlike on ODE)**

- Why is observation above important?
  - When we seek to find a numerical solution to a problem it's good to know that the problem has a solution and it is unique; i.e., problem is well posed

- In the Dynamics Analysis we are dealing with a well posed problem (an IVP)
  - Focus then on finding a way to approximate its solution by using the computer
    - The computer will produce numbers that at each node of the time grid will approximate the value of the generalized positions and generalized velocities

Initial Value Problems:
Basic Concepts

- Truncation Error
- Accuracy/Consistency
- Convergence
  - 0-stability
- Order of a method
- Local Error
- Stability
  - Absolute stability
  - A-stable Integration Methods
  - L-Stable Integration Methods (Methods with Stiff Decay)

Framework

- Interested in finding a function $y(t)$ over an interval $[0, b]$
- This function must satisfy the following IVP:

$$\begin{cases} \dot{y} &= f(t, y) \\ y(0) &= c \end{cases} \qquad t \in [0, b]$$

- We assume that f is bounded and smooth, so that y exists, is unique, and smooth

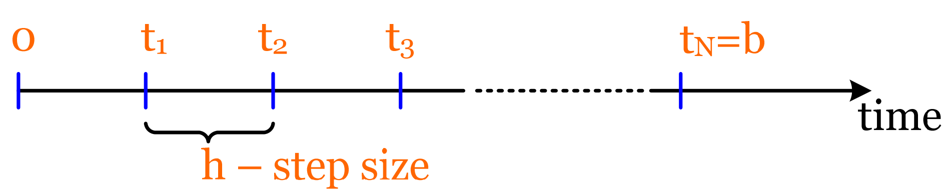
- Quantities given to you:
  - The constants c and b
  - The function $f(t, y)$.

Framework [Cntd.]

- Expression of given function $f(t, y)$
  - Case 1: $f(t, y)$ is simple, and in very rare cases you can find $y(t)$ analytically; i.e., find the exact solution of the IVP (the pen and paper case)
  - Case 2: $f(t, y)$ is complex but nonetheless you have direct access to it.
    - Producing an exact solution is not possible, resort to numerical integration, solution approximated using the computer
  - Case 3: $f(t, y)$ is so complex that you don't even have an expression for it. Instead you have to evaluate $f(t, y)$ based on value of t and y
    - Producing an exact solution is not possible, resort to numerical integration
    - This is ME751 - don't have a $f(t, y)$ explicit expression for most mechanisms

Framework [Cntd.]

- Implications (in relation to the complexity of $f(t, y)$):
  - Most likely we can't find the function $y(t)$ analytically (using pen and paper)
  - We'll have to use a numerical approximation method to come up with approximations of the unknown function $y(t)$
    - What we'll produce is an approximation of the value of $y(t)$ at 0, t_1, t_2, etc.
    - We are working on a grid of N time points, starting at 0 and ending at b
    - For simplicity, we assume that the grid points are equally spaced by a value h
    - This value is called the "integration step-size", usually denoted by h or Δt



Framework [Cntd.]

- Remember this: we'll approximate the value of $y(t_n)$ by a value y_n that we'll learn how to obtain

The unknown function y evaluated at t_n

$$y_n \approx y(t_n)$$

An approximation of the value of the unknown function at time t_n

- Notation: y^h used to denote the set of values y_0, y_1, …, y_N that I produce in my effort to approximate the unknown function $y(t)$ at the grid points t_0, t_1, …, t_N:

$$y^h = \{y_0, y_1, \ldots, y_N\}$$

- Quick Remark: a professional grade method for finding the approximate solution does not use an equally spaced grid of points 0, t_1, t_2, t_3…
  - We'll stick with assumption of equally spaced points though

Basic Concepts: Truncation Error
[Preliminaries]

- We have our standard IVP: $$\begin{cases} \dot{y} & = & f(t,y) \\ y(0) & = & c \end{cases}$$

- First, introduce simplest numerical integration scheme: Forward Euler
- To this end, invoke Taylor series expansion of $y(t)$ around t_{n-1}:

$$y(t_n) = y(t_{n-1}) + h\dot{y}(t_{n-1}) + \frac{1}{2}h^2\ddot{y}(t_{n-1}) + \dots$$

- Drop terms with powers of h of order 2 and higher
- Note that I won't use $y(t_n)$ anymore, but I will now use y_n
  - Make sure you understand this distinction between the exact value $y(t_n)$ and approximate value y_n …

[Basic Concepts]

Truncation Error

[Preliminaries]

- I had this:

$$y(t_n) = y(t_{n-1}) + h\dot{y}(t_{n-1}) + \frac{1}{2}h^2\ddot{y}(t_{n-1}) + \dots$$

Truncation error

- But compute my solution like this

$$y_n = y_{n-1} + h\dot{y}_{n-1}$$

- Or equivalently,

$$\boxed{y_n = y_{n-1} + h\, f(t_{n-1}, y_{n-1})}$$

- It's fair to expect that there is a slight difference between the exact value $y(t_n)$ and its numerical approximation y_n

Example, Forward Euler (FE)

- Solve the IVP

$$\dot{y} = -10y$$
$$y(0) = 1$$

using Forward Euler (FE) with a step-size $h = 0.01$

- Compare to the exact solution

FE integration with $h = 0.01,\ f(t, y) = -10y$

Exact solution: $y(t) = e^{-10t}$

$n = 0$	$y_0 = 1.0$				$= 1.0$	$y(t_0)$	$=$	**1.0**
$n = 1$	$y_1 = y_0 + h \cdot f(t_0, y_0)$	$= 1.0$	$+$	$0.01 \cdot (-10 * 1.0)$	$= 0.9$	$y(t_1)$	$=$	**0.9048**
$n = 2$	$y_2 = y_1 + h \cdot f(t_1, y_1)$	$= 0.9$	$+$	$0.01 \cdot (-10 * 0.9)$	$= 0.81$	$y(t_2)$	$=$	**0.8187**
$n = 3$	$y_3 = y_2 + h \cdot f(t_2, y_2)$	$= 0.81$	$+$	$0.01 \cdot (-10 * 0.81)$	$= 0.729$	$y(t_3)$	$=$	**0.7408**
$n = 4$	$y_4 = y_3 + h \cdot f(t_3, y_3)$	$= 0.729$	$+$	$0.01 \cdot (-10 * 0.729)$	$= 0.6561$	$y(t_4)$	$=$	**0.6703**
$n = 5$	$y_5 = y_4 + h \cdot f(t_4, y_4)$	$= 0.6561$	$+$	$0.01 \cdot (-10 * 0.6561)$	$= 0.5905$	$y(t_5)$	$=$	**0.6065**

[Basic Concepts]

Truncation Error

[Definition]

- Consider how the solution is obtained:

$$\frac{y_n - y_{n-1}}{h} - f(t_{n-1}, y_{n-1}) = 0$$

- Note that in general, if you plug the actual solution in the equation above it is not going to be satisfied. That is,

$$\frac{y(t_n) - y(t_{n-1})}{h} - f(t_{n-1}, y(t_{n-1})) \neq 0$$

- By definition, the quantity above is called the truncation error and is denoted by

$$\mathcal{N}(y, t_n, h) = \frac{y(t_n) - y(t_{n-1})}{h} - f(t_{n-1}, y(t_{n-1}))$$

- Note that this depends on the function (y), the point where you care to evaluate the truncation error (t_n), and the step size used (h)

Consistency/Accuracy Order

- Important remark:
  - Note that the truncation error depends on the integration scheme used to compute y^h:

$$\mathcal{N}(y, t_n, h)_{F.Euler} \neq \mathcal{N}(y, t_n, h)_{Runge-Kutta}$$

- Definition: an integration scheme is said to be consistent/accurate to order $p > 0$ if

$$\mathcal{N}(y, t_n, h) = \mathcal{O}(h^p)$$

- **Definition: A quantity d is said to be order p and denoted as $d = \mathcal{O}(h^p)$, if there is a constant C such that when $h \to 0$ we have that**

$$|d| \leq Ch^p$$

[Basic Concepts]

Consistency/Accuracy Order

- Exercise:
  - Prove that Forward Euler is order 1 accurate, that is,

$$\mathcal{N}(y, t_n, h)_{F.\ Euler} = \mathcal{O}(h)$$

[Straightforward, simply go back to Taylor series expansion]

[Basic Concept]
Local Error

- Imagine it as being the error that is registered in *one* integration step

- Specifically, you start at time step t_{n-1}, from the point y_{n-1}
  - The numerical solution yields at t_n the approximate value y_n (no surprise here)
  - The analytical solution that passes through the initial value y_{n-1} produces at t_n the value $\bar{y}_n$:

$$\begin{cases} \dot{\bar{y}} & = & f(t, \bar{y}) \\ \bar{y}(t_{n-1}) & = & y_{n-1} \end{cases}$$

- The local error is the difference

$$l_n = \bar{y}(t_n) - y_n$$

- Note the relationship that exists between the local error and local truncation error:

$$h \cdot |\mathcal{N}(\bar{y}, t_n, h)| = |l_n| \cdot (1 + \mathcal{O}(h))$$

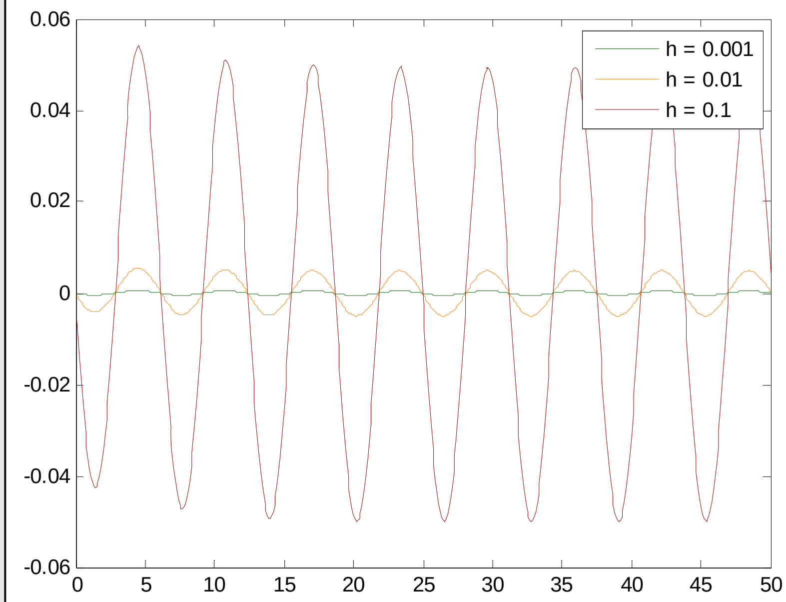
Example, Forward Euler: Effect of Step-Size

IVP: $\dot{y} = -0.1y + \sin(t)$, $y(0) = 0$

$$y_{exact}(t) = \frac{1}{1.01}\left(e^{-0.1t} + 0.1\sin(t) - \cos(t)\right)$$

```
% IVP (RHS + IC)
f = @(t,y) -0.1*y + sin(t);
y0 = 0;
tend = 50;

% Analytical solution
y_an = @(t) (0.1*sin(t) - cos(t) + exp(-0.1*t)) / (1+0.1^2);

% Loop over the various step-size values and plot errors
colors = [[0, 0.4, 0]; [1, 0.5, 0]; [0.6, 0, 0]];
Figure, hold on, box on
h = [0.001 0.01 0.1];
for ih = 1:length(h)
    tspan = 0:h(ih):tend;
    y = zeros(size(tspan));  err = zeros(size(tspan));
    y(1) = y0;               err(1) = 0;
    for i = 2:length(tspan)
        y(i) = y(i-1) + h(ih) * f(tspan(i-1), y(i-1));
        err(i) = y(i) - y_an(tspan(i));
    end
    plot(tspan, err, 'color', colors(ih,:));
end
legend('h = 0.001', 'h = 0.01', 'h = 0.1');
```

FE errors for different values of the step-size $h = 0.001, 0.01, 0.1$



[Basic Concepts]

Convergence

- Important question:
  - Does the numerical solution y^h actually converge to the unique solution of the IVP?
  - "Converge" means as h gets smaller and smaller, do you see the numerical solution at each t_n approaching the exact solution?

- More formal way to frame the convergence concept
  - Define

$$e_n = |y_n - y(t_n)|, \qquad n = 1, 2, ..., N$$

  - Note that $N \times h = b$ (as h goes to zero, N keeps growing…)

- The numerical integration is said to be convergent of order p if

$$e_n = \mathcal{O}(h^p), \qquad n = 1, 2, ..., N$$

[side trip]

Zero-stability (0-stability)

- There is a relationship between Consistency/Accuracy and Convergence
- This relationship requires the concept of *zero-stability*
- Definition: the numerical discretization scheme is 0-stable if there are positive constants h_0 and K such that for any solutions x^h and z^h, obtained for $h < h_0$, one has that

$$|x_n - z_n| \leq K \left(|x_0 - z_0| + \max_{1 \leq j \leq N} |\mathcal{N}(x, t_j, h) - \mathcal{N}(z, t_j, h)| \right), \qquad 1 \leq n \leq N$$

- Tells you something about how close two solutions x^h and z^h stay if they start a distance $|x_0 - z_0|$ apart.

Order "p" Convergence

- Theorem:

$$\text{Consistency} \quad + \quad \text{0-stability} \quad \Rightarrow \quad \text{Convergence}$$

- Some more specifics:
  - If the method is accurate of order p and 0-stable, then it is convergent of order p:

$$e_n = \mathcal{O}(h^p), \qquad n = 1, 2, ..., N$$

Challenge Homework

- Prove that Forward Euler is a 0-stable discretization scheme

[Basic Concept]

Stability of a IVP Solution Method

- Convergence is good, but it tells me what happens if I work with smaller and smaller step-sizes h

- In reality, I want to operate with values of h that are large
  - Recall that if h is small you have to take a very large number of integration steps to cover the interval $[0, b]$

- The relevant question: How large can I consider h yet know for a fact that I don't get garbage approximations of the solution?

  - The answer to this question is posed to each numerical discretization scheme
  - Moreover, it is posed in conjunction with a test problem:

$$\begin{cases} \dot{y} &= \lambda y \\ y(0) &= 1 \end{cases}$$

[Basic Concept, Short Detour]

Stability of a IVP Solution Method

- Example: mass-spring-damper oscillation

$$\ddot{x} + 2\zeta\omega_n\dot{x} + \omega_n^2 x = 0$$
$$x(0) = x_0$$
$$\dot{x}(0) = \dot{x}_0$$

- Case 1: underdamped system

$$\lambda_1 = -\omega_n\zeta - j\omega_d \qquad \lambda_2 = -\omega_n\zeta + j\omega_d$$

- Case 2: overdamped system

$$\lambda_1 = -\omega_n\zeta - \omega_n\sqrt{\zeta^2 - 1} \qquad \lambda_2 = -\omega_n\zeta + \omega_n\sqrt{\zeta^2 - 1}$$

- Case 3: critically damped system

$$\lambda_1 = \lambda_2 = -\omega_n$$

[Basic Concept]

Stability of a IVP Solution Method

- Example, the concluding remark:
  - In mechanical engineering, λ assumes values for which typically

$$\mathbb{R}\mathrm{e}\{\lambda\} \leq 0$$

- It better be that our numerical discretization scheme leads to numerical solutions that can handle the test IVP for negative values of λ (or its real part, when dealing with complex values…)

- It turns out that this is not trivial
  - Forward Euler provides quick example of a method that is not smart enough

Example:

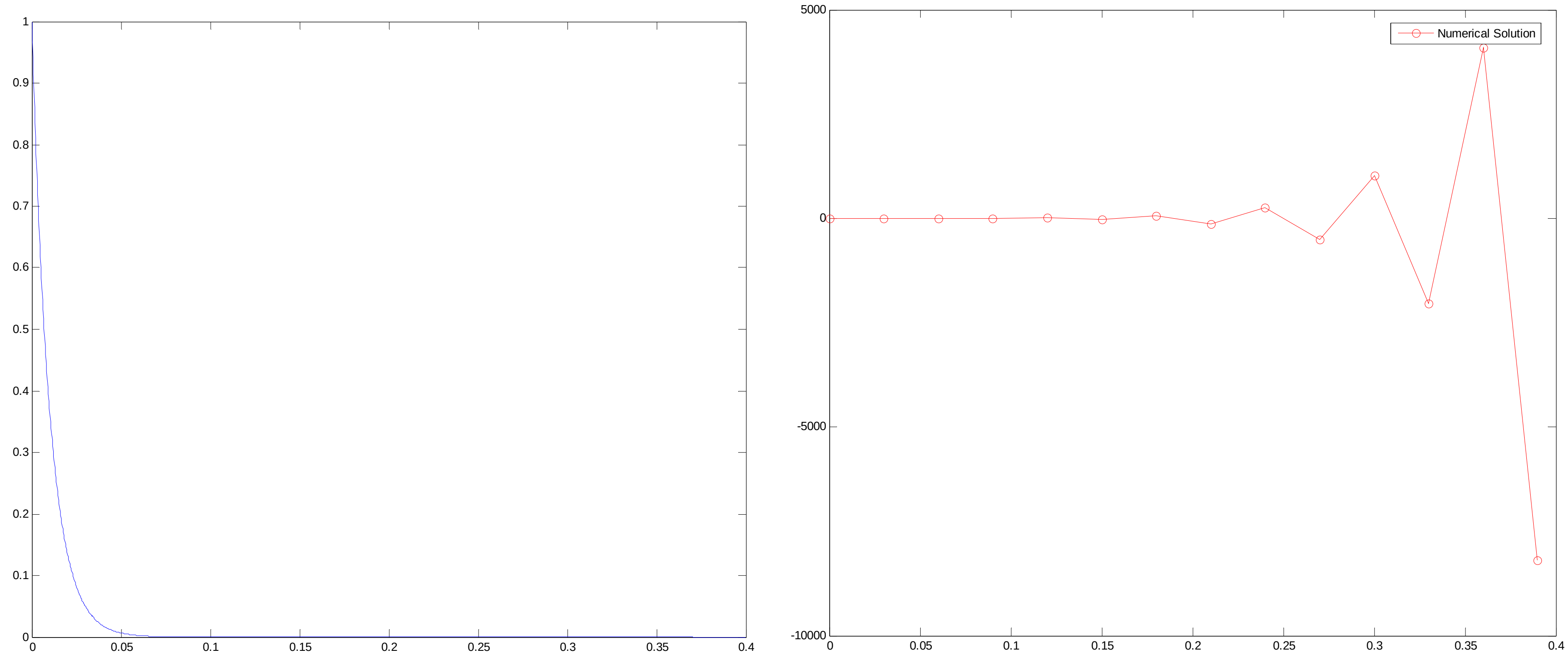
$$\left.\begin{aligned}\dot{y} &= -100y \\ y(0) &= 1\end{aligned}\right\} \Rightarrow y(t) = e^{-100\,t}$$

- Integrate 5 steps using forward Euler formula: h=0.002, h=0.01, h=0.03
- Compare errors between numerical and analytical solutions at t_0, t_1, t_2, t_3, t_4, t_5

h=0.002: Error @ t_0,..., t_5	h=0.01: Error @ t_0,..., t_5	h=0.03: Error @ t_0,..., t_5
0	0	0
0.01873075307798	0.36787944117144	2.04978706836786
0.03032004603564	0.13533528323661	-3.99752124782333
0.03681163609403	0.04978706836786	8.00012340980409
0.03972896411722	0.01831563888873	-15.99999385578765
0.04019944117144	0.00673794699909	32.00000030590232

Example:
(λ =-100)

$$\left.\begin{array}{r}\dot{y} = -100y \\ y(0) = 1\end{array}\right\} \Rightarrow y(t) = e^{-100\,t}$$



Analytical Solution

Forward Euler

(Δt=0.03)

Basic Concept: Stability

- One quick way to see that things went south. First note that:

$$\text{If } \mathbb{R}\text{e}\{\lambda\} < 0, \text{ then } 0 < y(t_n) < y(t_{n-1}) < \ldots < y(t_1) < y(t_0) = 1$$

- You'd expect that the numerical solution will mirror this behavior:

$$0 < y_n < y_{n-1} < \ldots < y_1 < y_0 = 1$$

- Use Forward Euler to express y_n as a function of y_{n-1} and require that the condition $y_n < y_{n-1}$to obtain that

$$|1 + h\lambda| < 1$$

Basic Concept: Stability

- Recall what happened
  - We started with the test problem
  - We required that for the test problem, the numerical approximation should behave like the solution. That is, we required that

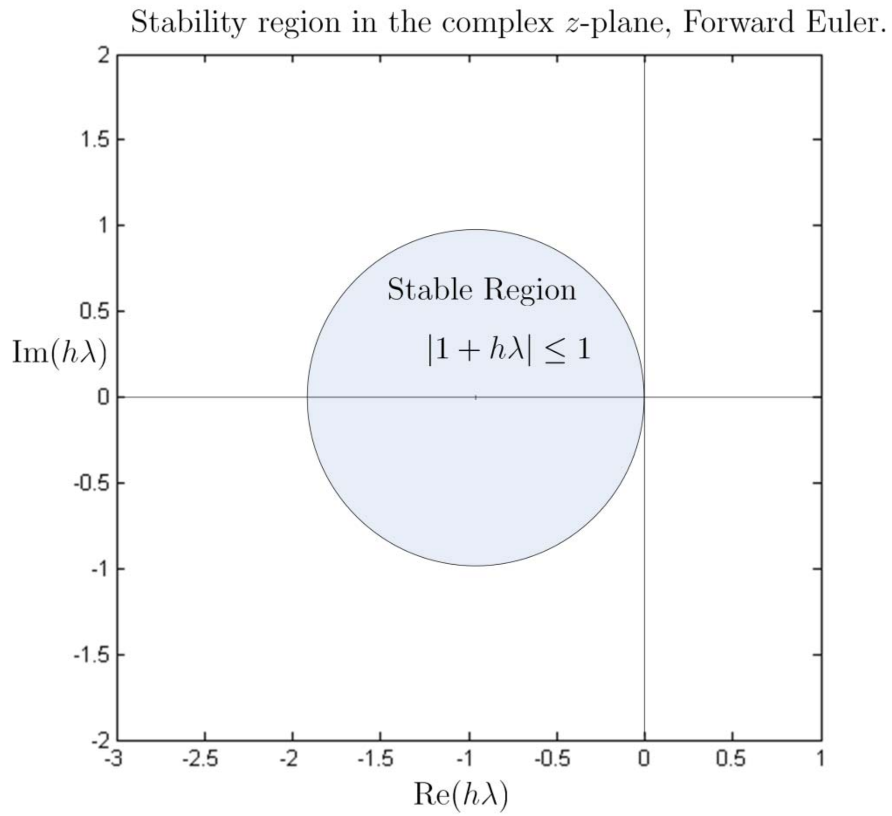
$$y_n \leq y_{n-1}$$

  - We used the discretization scheme (Forward Euler, in our case) to express how y_n is related to y_{n-1}
  - This led to a condition that the step size should satisfy, specifically, for Forward Euler, we obtained that

$$|1 + h\lambda| < 1$$

  - What we just did was to determine the region of absolute stability of Forward Euler
  - The KEY point: to get the absolute stability region for other integration methods, you have to use that integration method to express how y_n is related to y_{n-1}

Basic Concept: Stability



- The step size h should be such that $h\lambda$ lands into the shaded circle
- Note that a very negative value for λ will require a very small value of h so that the product $h\lambda$ is inside the circle

Accuracy vs. Stability. Any contradiction here?

- Recall that Forward Euler is accurate to order 1
  - That is, locally,

$$\mathcal{N}(y, t_n, h)_{F.\ Euler} = \mathcal{O}(h)$$

- This is an asymptotic and *local* result, which holds better as h gets smaller
- For the test IVP, the local error compounds due to the particular form of the problem that we work with (the test IVP)
- This compounding and the fact that h does not assume small values (you try to work w/ large h) leads to the phenomenon of loss of stability

- To conclude, there is no contradiction here (the numerical scheme being order 1 accurate yet losing stability for large values of h)

Accuracy vs. Stability: The Concept of Stiffness

- Recall that the size/shape of the stability region (SR) is specific to each discretization scheme

- For some discretization schemes the SR is ridiculously small
  - Forward Euler is one of them

- A small SR to start with, combined with a problem for which λ is very negative leads to unreasonably small values of h
  - Such a problem is called a "stiff" IVP

- In this case it is *not* the accuracy concerns that restrict the value of the step-size h, but rather the stability issue prevails

- Note that it can be the case that if you change the integration method, the stiff problem is just fine (if the stability region is generous)

Example:

- Use Forward Euler to find an approximation of the solution of the following IVP:

$$\begin{cases} \dot{y} = -100y + \sin(t) \\ y(0) = 0 \end{cases} \qquad t \in [0, 8]$$

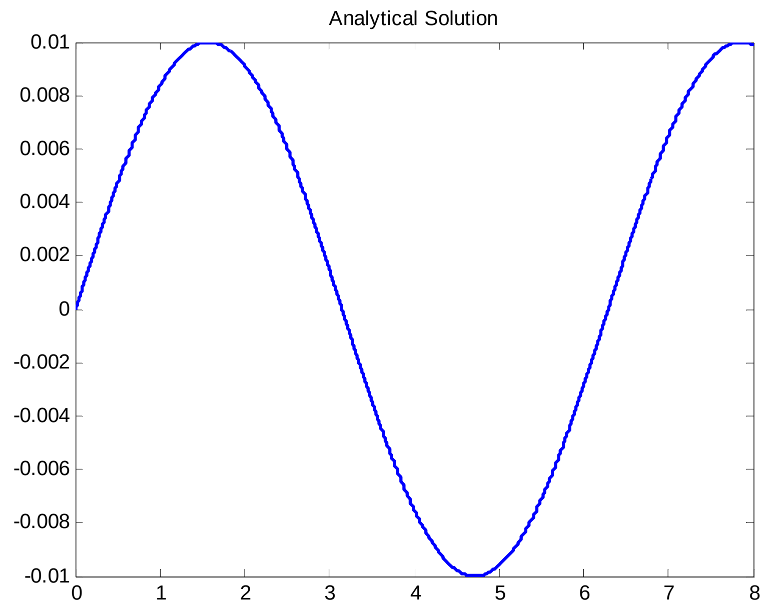
Example [Cntd.]

- Easy to figure out that the exact solution is

$$y(t) = \frac{1}{10001}\left(100\sin(t) - \cos(t) + e^{-100t}\right)$$

- For all purposes, the solution can be rewritten as

$$y(t) \approx 0.01\sin(t - \phi_0) \qquad \text{where} \qquad \tan\phi_0 = \frac{1}{100}$$

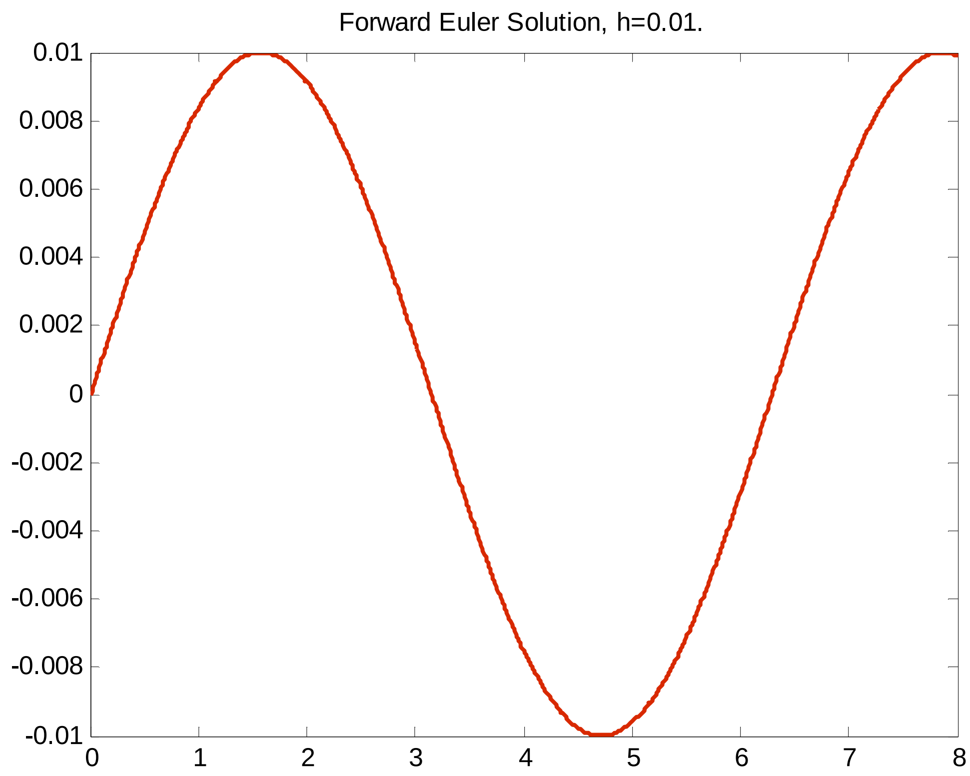


Example [Cntd.]

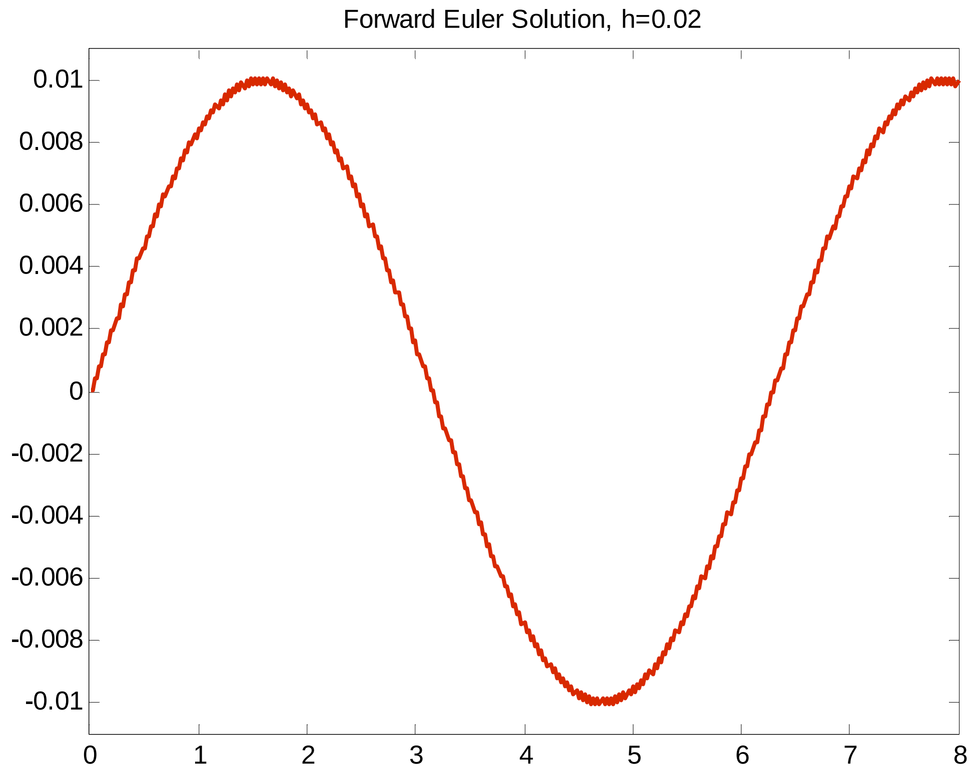
- Note that for the given IVP, $\lambda = -100$, which suggests we'll have to work with small step-sizes…
- Note in the plot that the contribution of the exponential is not even seen in the MATLAB plot
- You'd expect that since the exponential component of the solution goes away so quickly one could use Forward Euler and have no difficulties, which is not true… (see next slide)

```
% Solves the IVP y_dot = -100*y + sin(t) and y(0)=0
% yExact is the analytical solution
% yFE is the solution obtained with Forward Euler
% yBE is the solution obtained with Backward Euler
%
% Input: the integration step-size

h = input('Input step size h:');
tend = 8;
tm=0:h:tend;

yExact = 1/10001*(100*sin(tm)-cos(tm)+exp(-100*tm));

yFE = zeros(size(tm'));
yBE = zeros(size(tm'));

for i=2:1:length(yFE)
   yFE(i) = yFE(i-1)*(1-100*h) + h*sin(tm(i-1));
end

dummyINV = 1/(1+100*h);
for i=2:1:length(yBE)
   yBE(i) = yBE(i-1)*dummyINV + h*dummyINV*sin(tm(i));
end
```

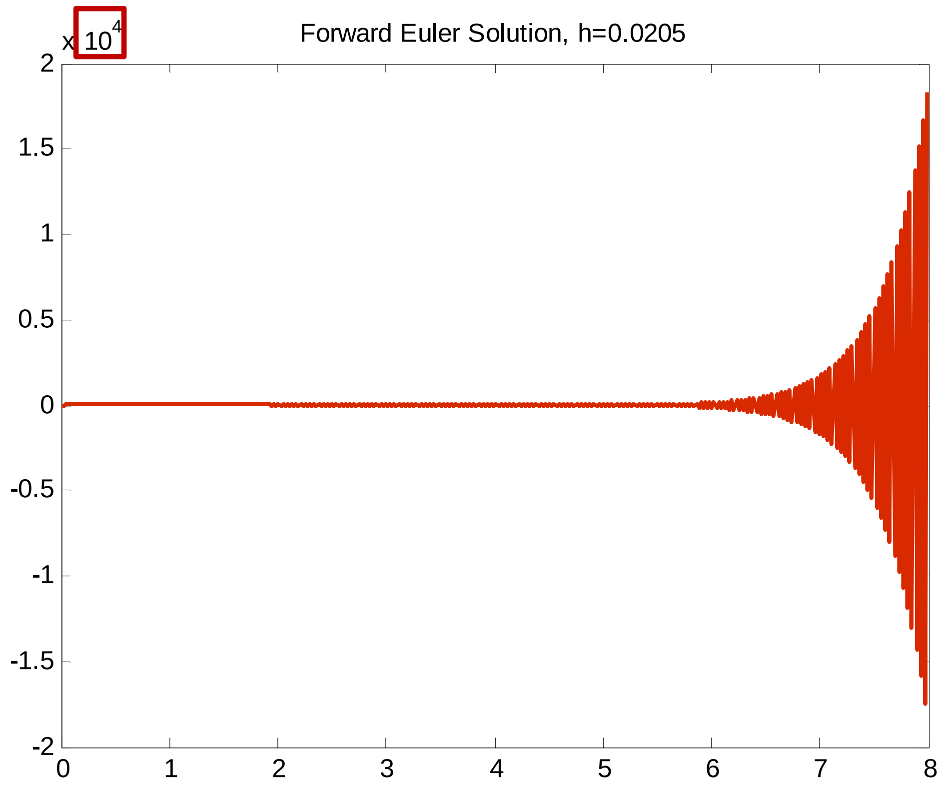
Example, Approached with Forward Euler: h=0.01 s



Example, Approached with Forward Euler : h=0.02 s



Example, Approached with Forward Euler: h=0.0205 s



- As soon as you go beyond the limit value h=0.02 (that goes hand in hand for Forward Euler with $\lambda = -100$), you run into trouble
- Note that this happens even though the contribution of the exponential goes away very fast…

Example, Approached with Forward Euler

- Conclusion
  - For this type of problem with very negative λ, Forward Euler is bad
    - The step size is significantly limited on stability grounds
- Qualitative definition:
  - An IVP where Forward Euler behaves bad is called STIFF IVP

[New Topic]
Implicit Methods

- Implicit methods were derived to answer the limitation on the step size noticed for Forward Euler, which is an explicit method

- Simplest implicit method: Backward Euler
  - Given the IVP

$$\begin{cases} \dot{y} & = & f(t,y) \\ y(0) & = & c \end{cases}$$

  - Backward Euler finds at each time step t_n the solution by solving the following equation for y_n:

$$\boxed{y_n = y_{n-1} + hf(t_n, y_n)}$$

Explicit vs. Implicit Methods

- A method is called explicit if the approximation of the solution at the next time step is computed straight out of values computed at previous time steps
  - In other words, in the right side of the formula that gives y_n, you only have dependency on *past* values; i.e., y_{n-1}, y_{n-2}, etc. – it's like a recursive formula
  - Example: Forward Euler

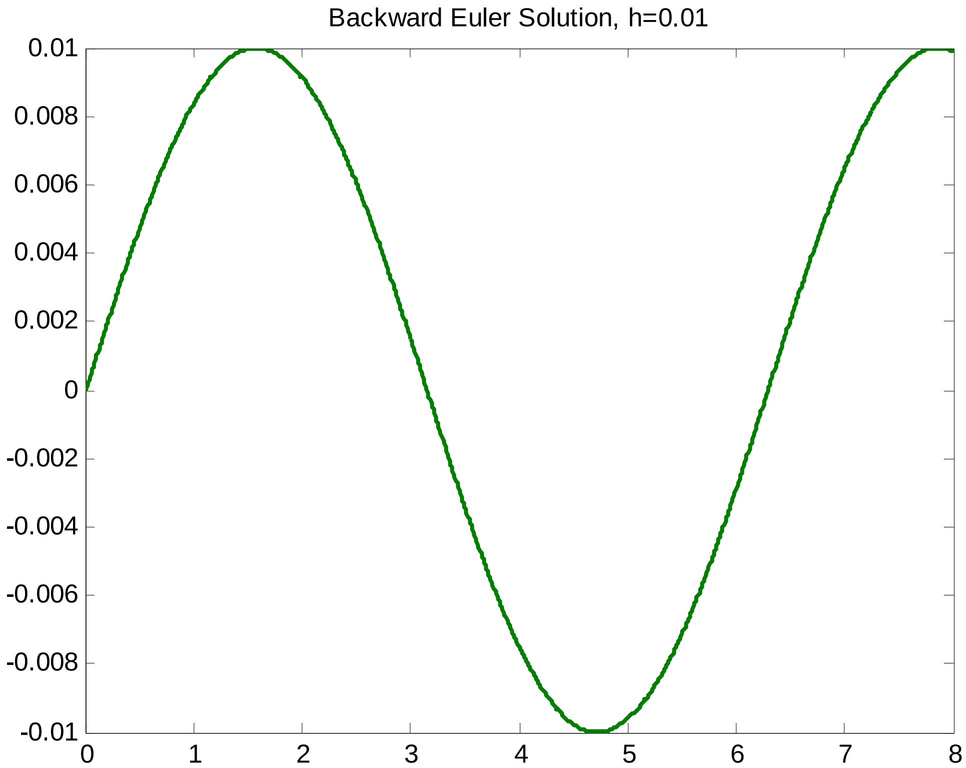
$$y_n = y_{n-1} + hf(t_{n-1}, y_{n-1})$$

- A method is called implicit if the solution at the new time step is found by solving an equation:
  - Specifically, in the right side of the formula that gives y_n, you have dependency on y_n
  - Example: Backward Euler

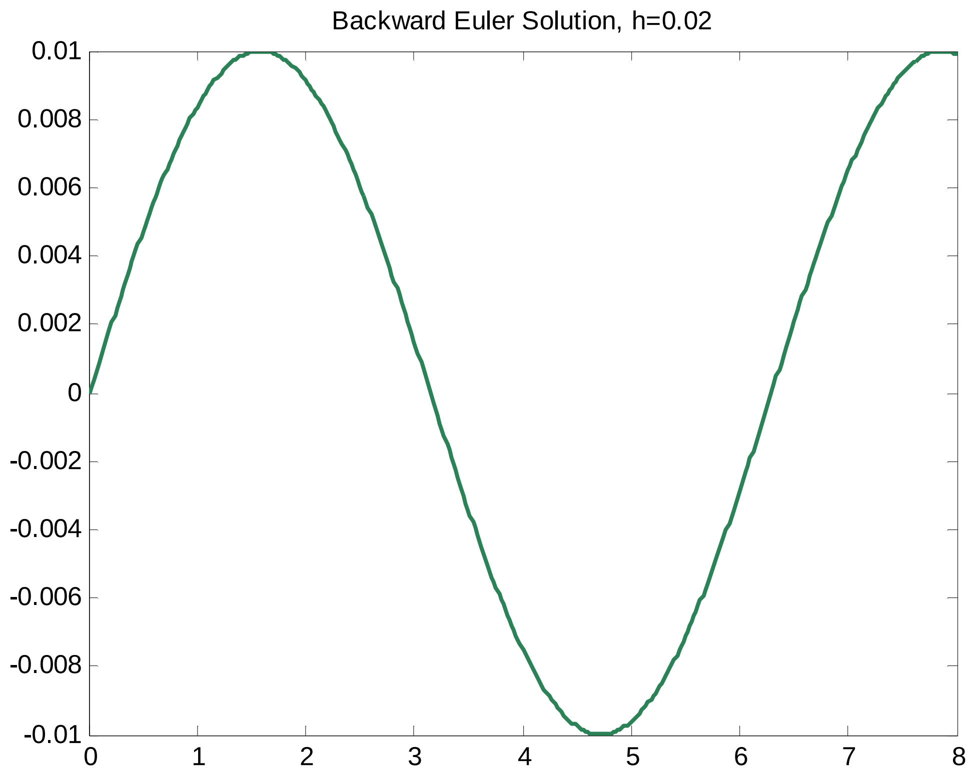
$$y_n = y_{n-1} + hf(t_n, y_n)$$

Example, Approached with Backward Euler: $h = 0.01$

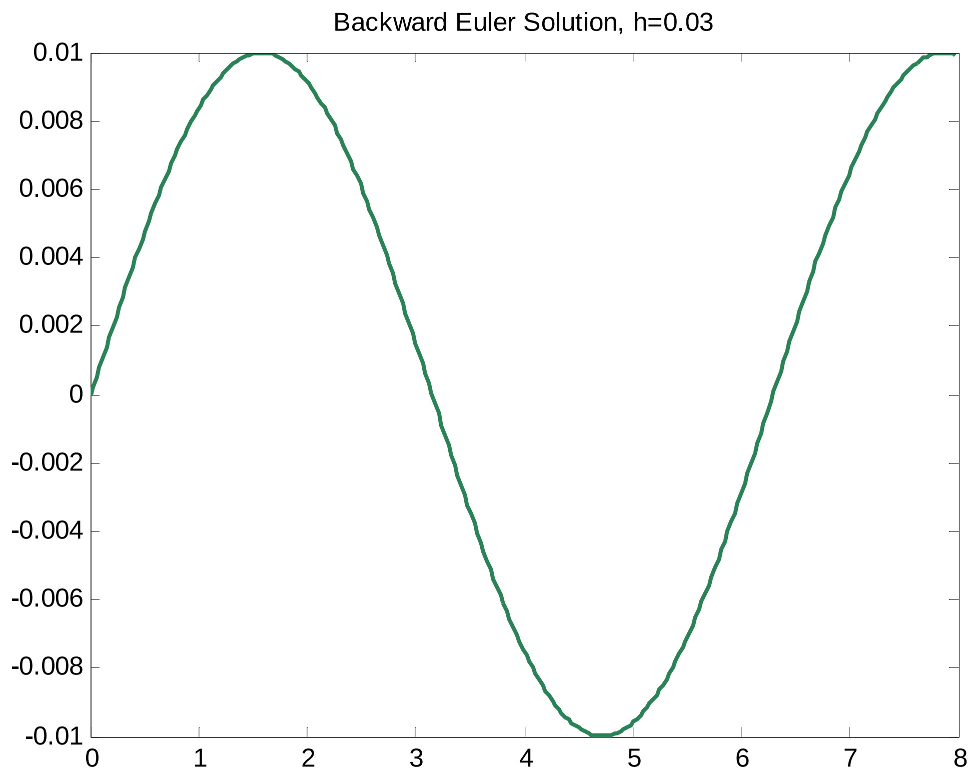
$$\begin{cases} \dot{y} = -100y + \sin(t) \\ y(0) = 0 \end{cases} \qquad t \in [0, 8]$$



Example, Approached with Backward Euler: $h = 0.02$



Example, Approached with Backward Euler: $h = 0.03$



- Note that things are good at large values of the integration step size

Exercise, Backward Euler

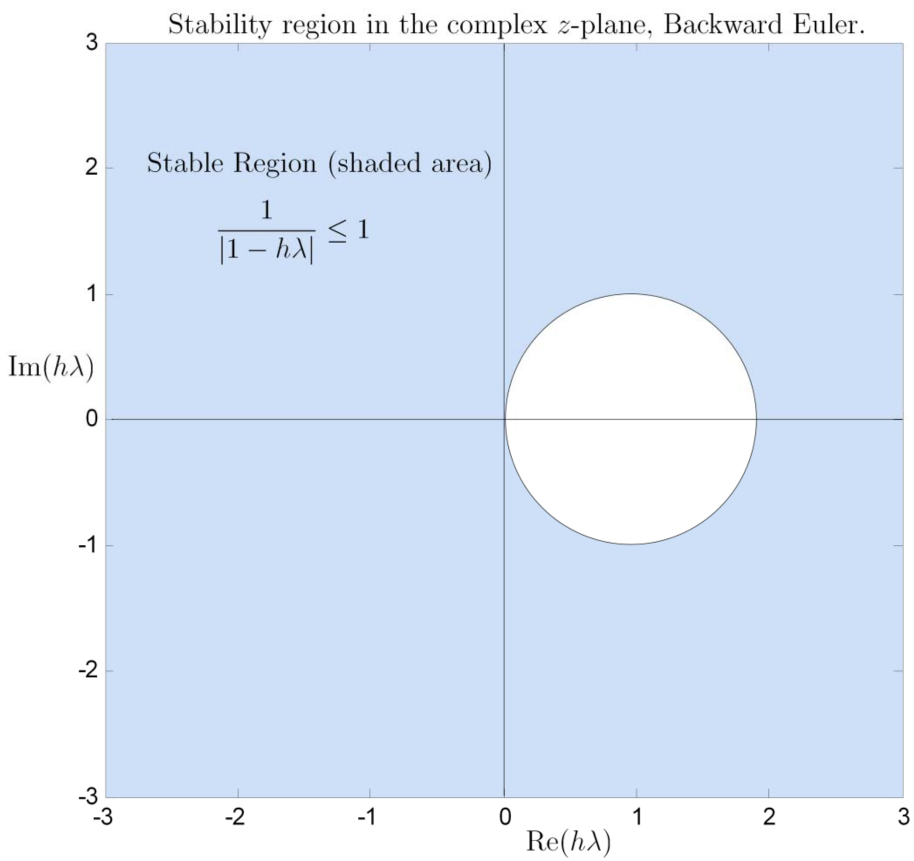
- Prove that
  - Backward Euler is accurate of order 1
  - It satisfies the 0-order stability condition
  - It's convergent with convergence order 1

- Generate
  - The stability region of the method and compare to Forward Euler
  - A convergence plot for the IVP

$$\begin{cases} \dot{y} = -100y + \sin(t) \\ y(0) = 0 \end{cases} \qquad t \in [0, 8]$$

s1

Stability Region, Backward Euler



Slide 47

s1 Need to go pen and paper
sbel, 10/11/2016

Generating Convergence Plot

- Procedure to generate Convergence Plot:
  - First, get the exact solution, or some highly accurate numerical solution that can serve as the reference solution

  - Run a sequence of 6 to 8 simulations with decreasing values of step size h
    - Each simulation halves the step-size of the previous simulation

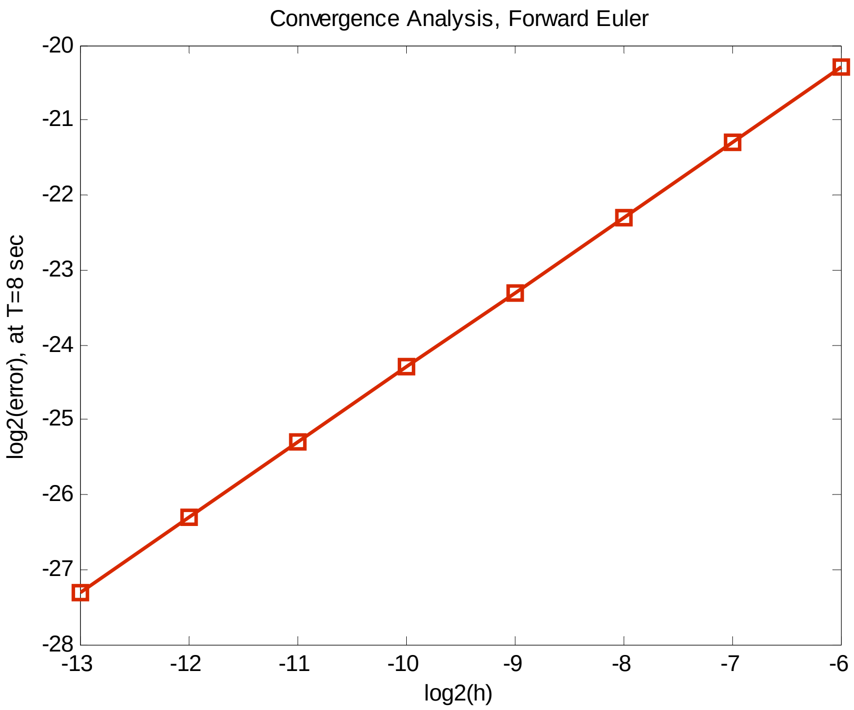
  - For each simulation of the sequence, compare the value of the approximate solution at Tend to the value of the reference solution at Tend

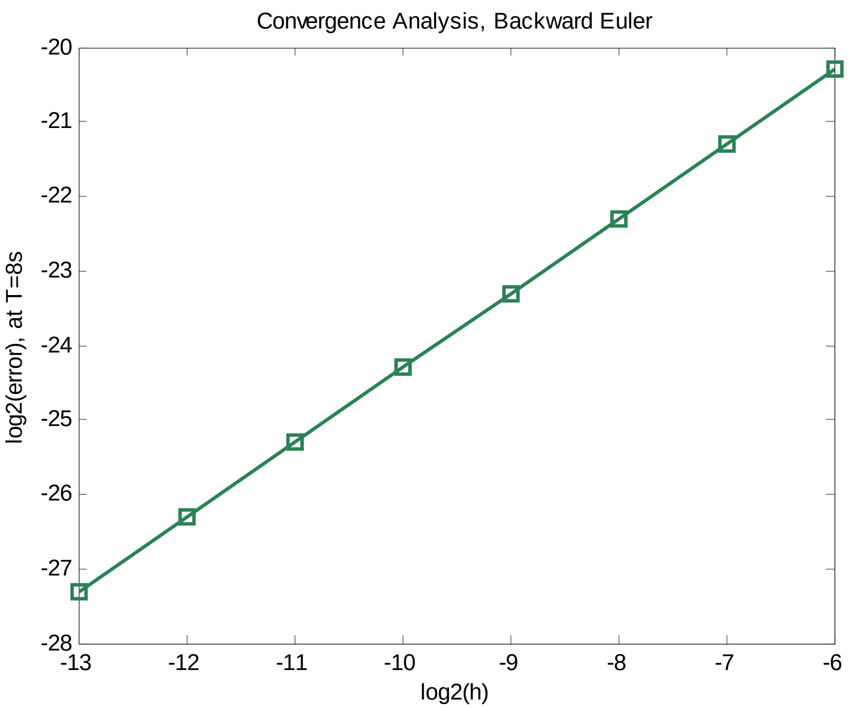
$$error = |y_{end} - y(T_{end})|$$

    - You don't necessarily have to use Tend, some other representative time is ok
    - Generate an array of pairs (h, error), and plot log2(h) vs. log2(error)
      - You should see a line of constant slope. The slope represents the convergence order

Convergence Plots

$$\begin{cases} \dot{y} = -100y + \sin(t) \\ y(0) = 0 \end{cases} \qquad t \in [0, 8]$$





Code to Generate Convergence Plot

```
nPoints = 8;        % number of points used to generate the convergence plot
hLargest = 2^(-6);  % largest step-size h considered in the convergence analysis
tEnd = 8;           % Tend

hSize = zeros(8,1);
hSize(1) = hLargest;
for i=1:nPoints-1
    hSize(i+1)=hSize(i)/2;
end

% First column of "results" : the step size used for integration
% Second column of "results": the error in the Forward Euler at Tend
% Third column of "results" : the error in the Backward Euler at Tend
results = zeros(nPoints, 3);

% Run a batch of analyses, the step size is gradually smaller
for i=1:nPoints
    yE = zeros(size(0:hSize(i):tEnd))';
    yFE = zeros(size(yE));
    yBE = zeros(size(yE));
    [yE, yFE, yBE] = fEulerVSbEuler(hSize(i), tEnd);
    results(i,1) = hSize(i);
    results(i,2) = abs(yE(end)-yFE(end));
    results(i,3) = abs(yE(end)-yBE(end));
end
```