ME451
Kinematics and Dynamics of Machine Systems

Dynamics of Planar Systems

Tuesday, November 30, 2010

Numerical Solution of IVPs

[not in the book]

Before we get started…

- Last Time
  - How the reaction forces are obtained
    - Recall that there are two important selections that you have to make:
      - The point at which you want the effect of the kinematic constraint; i.e., reaction force/torque, to be "felt"
      - The reference frame in which you represent the reaction force/torque
- Today:
  - How to solve ordinary differential equations (Initial Value Problems)
  - MATLAB Assignment – will be posted online.
    - Due date: December 7.
- Miscellaneous
  - Dan's trip cancelled
  - Th, Dec. 2 @ 11AM in 1152ME: Hammad explains how you can add a post-processing component to simEngine2D in order to animate the motion of the mechanism (attend only if you are interested in the topic)
  - Th, Dec. 2 @ 5PM in 1152ME: Dan to hold review for midterm exam
  - Th, Dec. 2 @ 7:15-9:15 PM in 1152ME: Second midterm exam
  - The take-home component of the exam available on the class website

Numerical Method

(also called Numerical Algorithm, or simply Algorithm)

- Represents a recipe, a succession of steps that one takes to find an approximation the solution of a problem that otherwise does not admit an analytical solution
  - Analytical solution: sometimes called “closed form” or “exact” solution
  - The approximate solution obtained with the numerical method is also called “numerical solution”

- Examples:
  - Evaluate the integral $$I = \int_0^3 e^{-x^2}\, dx$$
  - Solve the equation $$e^x + \sin(\frac{2}{x}) + \sqrt{x} = 2$$
  - Solve the differential equation that governs time evolution of simple pendulum
  - Many, many others (actually very seldom can you find the exact solution of a problem…)

Where/How are Numerical Methods Used?

- Powerful and inexpensive computers have revolutionized the use of numerical methods and their impact
  - Simulation of a car crash in minute detail
  - Formation of galaxies
  - Motion of atoms
  - Finding the electron distribution around nuclei in a nanostructure

- Numerical methods enable the concept of “simulation-based engineering”
  - You use computer simulation to understand how your mechanism (mechanical system) behaves, how it can be modified and controlled

Numerical Methods in ME451

- In regards to ME451, one would use numerical method to solve the dynamics problem (the resulting set of differential equations that capture Newton's second law)
  - The particular class of numerical methods used to solve differential equations is typically called "**numerical integrators**", or "integration formulas"
  - A numerical integrator generates a numerical solution at **discrete time points** (also called grid points, station points, nodes)
    - This is in fact just like in Kinematics, where the solution is computed on a time grid
  - Different numerical integrators generate **different solutions**, but the solutions are typically very close together, and [hopefully] closed to the actual solution of our problem
  - Putting things in perspective: In 99% of the cases, the use of numerical integrators is **the only alternative** for solving complicated systems described by non-linear differential equations

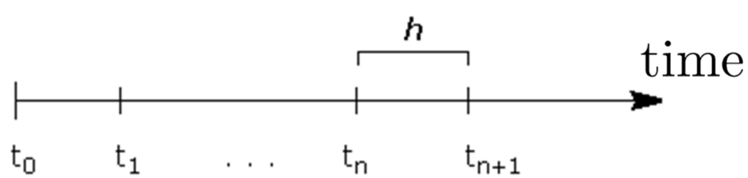
Numerical Integration
~Basic Concepts~

- Initial Value Problem: (IVP)

$$\begin{cases} \dot{y} = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

- So what's the problem?
  - You are looking for a function y(t) that depends on time (changes in time), whose time derivative is equal to a function f(t,y) that is given to you (see IVP above)
  - In other words, I give you the derivative of a function, can you tell me what the function is?
  - Remember that both y_0 and the function f are given to you. You want to find y(t).

- In ME451, the best you can hope for is to find an approximation of the unknown function y(t) at a sequence of discrete points (as many of them as you wish)
  - The numerical algorithm produces an <u>approximation</u> of the value of the unknown function y(t) at the each grid point. That is, the numerical algorithm produces $y(t_1)$, $y(t_2)$, $y(t_3)$, etc.



Relation to ME451

- When carrying out Dynamics Analysis, what you can compute is the acceleration of each part in the model

- Acceleration represents the second time derivative of your coordinates

- Somewhat oversimplifying the problem to make the point across, in ME451 you get the second time derivate

$$\ddot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}, t)$$

- This represents a second order differential equation since it has two time derivatives taken on the position **q**

Numerical Integration: Euler's Method

Find solution of this Initial Value Problem:

$$\begin{cases} \dot{y}(t) = f(y,t) \\ y(t_0) = y_0 \end{cases}$$

- The idea: at each grid point t_k, **turn the differential problem into an algebraic problem** by approximating the value of the time derivative:

$$\dot{y}(t_k) \approx \frac{y(t_{k+1}) - y(t_k)}{t_{k+1} - t_k} = \frac{y_{k+1} - y_k}{\Delta t} \quad \Rightarrow \quad y_{k+1} = y_k + \Delta t\, \dot{y}(t_k)$$

This step is called "discretization". It transforms the problem from a continuous **ODE** problem into a discrete **algebraic** problem

Euler's Method (Δt is the step size):

$$\boxed{y_{k+1} = y_k + \Delta t\, f(t_k, y_k)}$$

Example:

$$\dot{y} = -10y$$

$$y(0) = 1$$

- Integrate 5 steps using Euler's Method
- Use an integration step $\Delta t=0.01$
- Compare to exact solution

f(t,y) = -10y (no explicit dependency on time t for f in this example)

k=0	y0 = 1.0		
k=1	y1 = y0+f(t0,y0)Δt = 1.0	+ (-10*1.0)*0.01	= 0.9
k=2	y2 = y1+f(t1,y1)Δt = 0.9	+ (-10*.9)*0.01	= 0.81
k=3	y3 = y2+f(t2,y2)Δt = 0.81	+ (-10*.81)*0.01	= 0.729
k=4	y4 = y3+f(t3,y3)Δt = 0.729	+ (-10*.729)*0.01	= 0.6561
k=5	y5 = y4+f(t4,y4)Δt = 0.6561	+ (-10*.6561)*0.01	= 0.5905

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

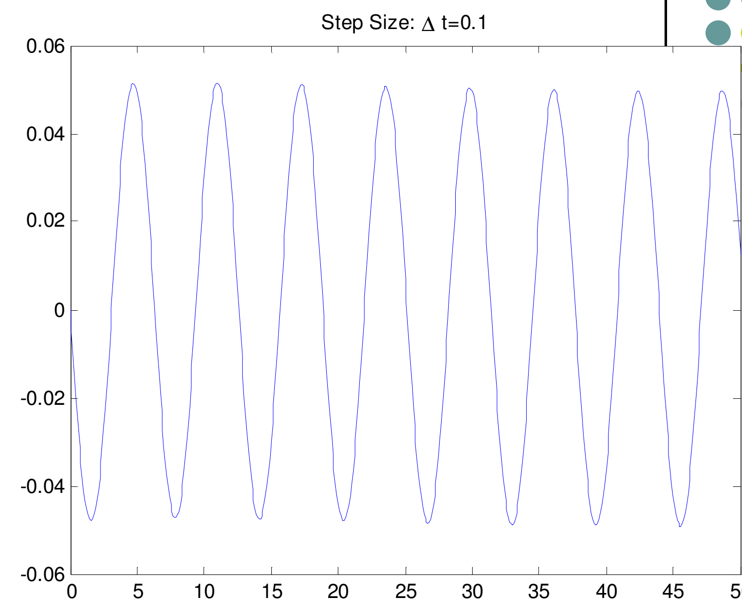
Solution:
y(0) =1.0000
y(0.01)=0.9048
y(0.02)=0.8187
y(0.03)=0.7408
y(0.04)=0.6703
y(0.05)=0.6065

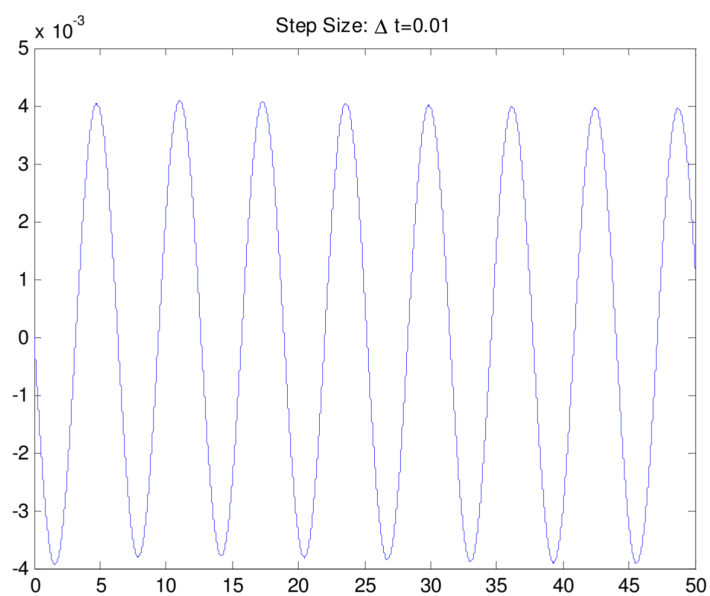
Euler Method:
~ Effect of Step Size ~

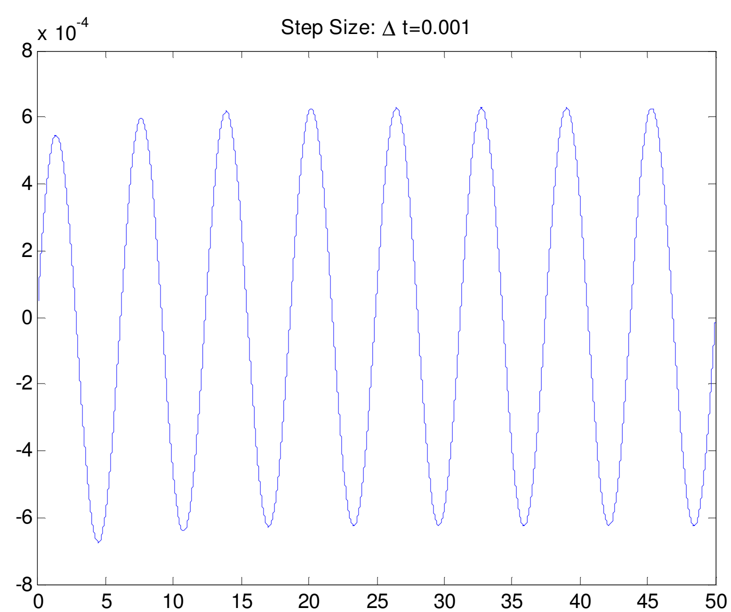
$$\begin{cases} \dot{y} &= -0.1y + \sin t \\ y(0) &= 0 \end{cases}$$

MATLAB code

```
clear
figure
dt = 0.001;
t=0:dt:50;
yExact=.99*exp(-t/10)+.995*sin(t-1.47);
% Numerical solution
th=0:dt:50;
yh=0;
for i=2:length(th)
   f=-yh(i-1)/10+sin(th(i));
   yh(i)=yh(i-1)+f*dt;
end
plot(t,yExact-yh) % this is printing the error...
```



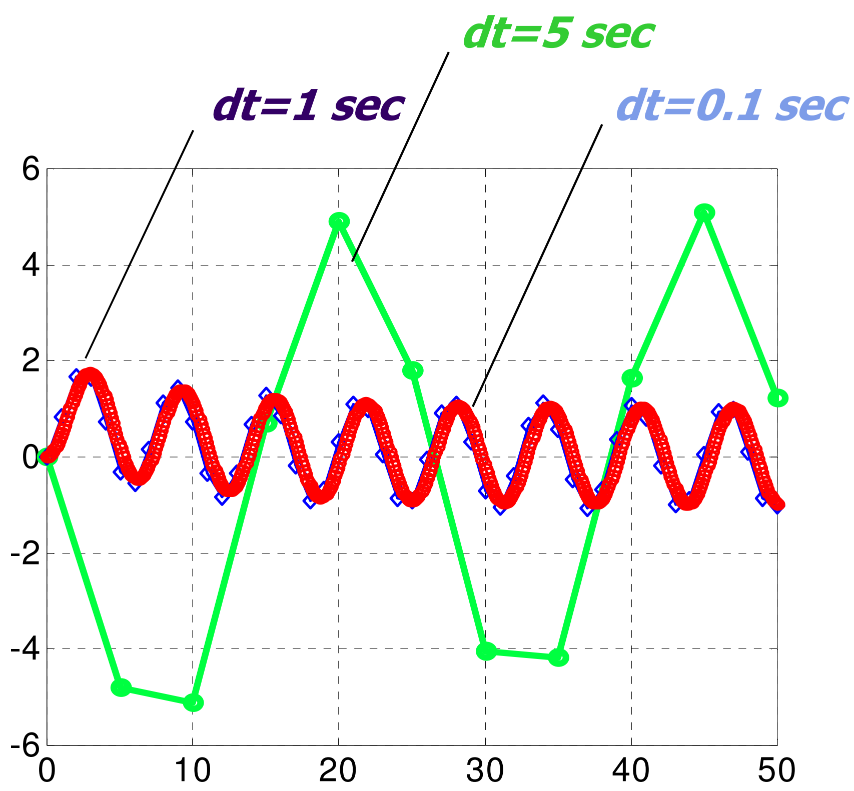




Euler Method:
~ Effect of Step Size ~

$$\begin{cases} \dot{y} & = & -0.1y + \sin t \\ y(0) & = & 0 \end{cases}$$

- Solve using step sizes *Δt=0.1, 1 and 5 sec*



Conclusion: If you use large step-sizes Δt, the **ACCURACY** of the solution is very poor (you can't be too aggressive with size of Δt)

The concept of stiff differential equations, and how to solve the corresponding IVP

Example: IVP

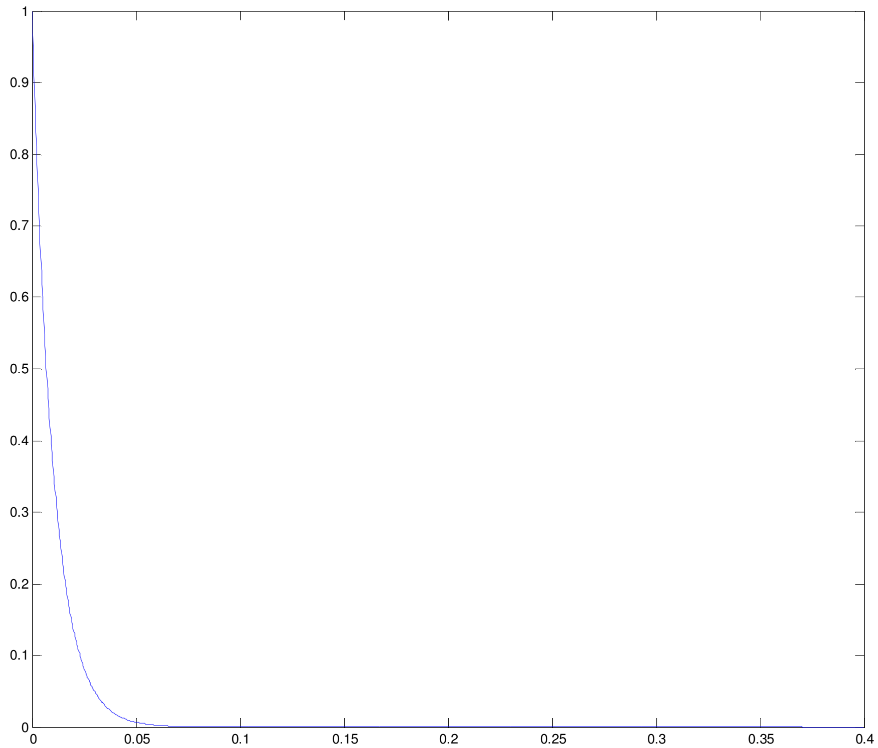
$$\left.\begin{aligned}\dot{y} &= -100y \\ y(0) &= 1\end{aligned}\right\} \Rightarrow \quad y(t) = e^{-100t}$$

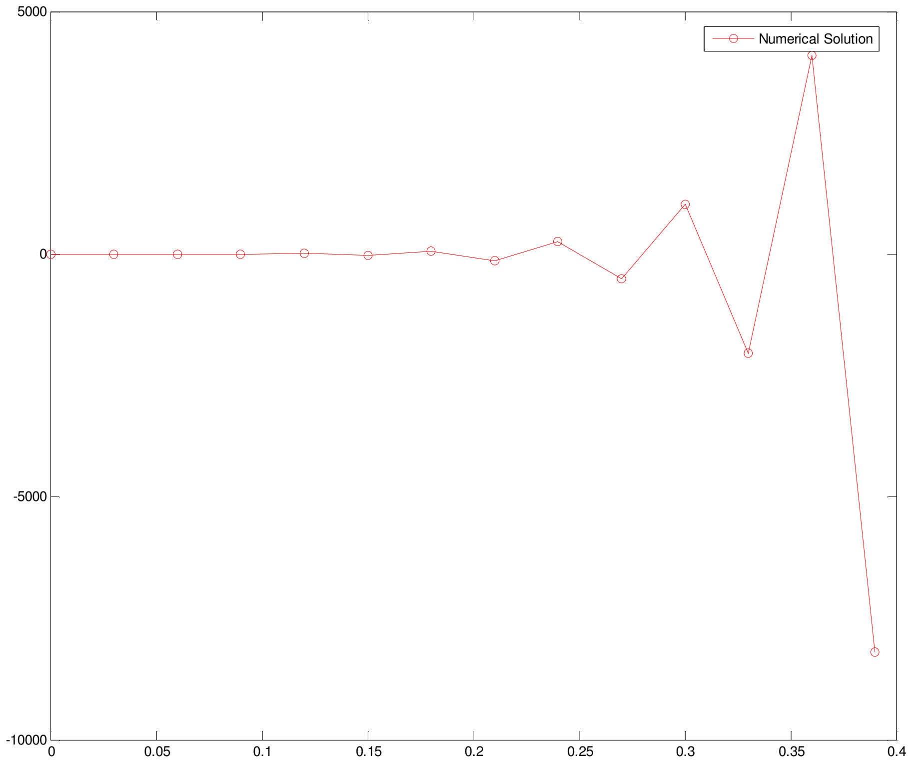
- Integrate 5 steps using forward Euler formula: Δt=0.002, Δt=0.01, Δt=0.03
- Compare the errors between numerical and analytical solutions (Algorithm Error)

Algorithm Error when Δt=0.002:	Algorithm Error when Δt=0.01:	Algorithm Error when Δt=0.03:
0	0	0
0.01873075307798	0.367879441171442	2.04978706836786
0.03032004603564	0.135335283236613	-3.99752124782333
0.03681163609403	0.04978706836786	8.00012340980409
0.03972896411722	0.01831563888873	-15.9999385578765
0.04019944117144	0.00673794699909	32.0000030590232

Example:

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





Forward Euler

Analytical Solution

(Δt=0.03)

Concept of stiff IVP's

- IVP's for which forward Euler doesn't work well (see example)
  - In general, the entire class of so called **explicit** formulas doesn't work
    - Forward Euler, Runge-Kutta (RK23, RK45), DOPRI5, Adams-Bashforth, etc.

- Stiff IVP's require a different class of integration formulas
  - **Implicit** formulas
    - Example: Backward Euler

$$\boxed{y_{k+1} = y_k + f(t_{k+1}, y_{k+1})\Delta t}$$

Explicit vs. Implicit Formulas (look at Euler family)

- Initial Value Problem

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}$$

- Forward Euler

$$y_{k+1} = y_k + \Delta t \, y'_k$$
$$y'_k = f(y_k, t_k)$$

$$\boxed{y_{k+1} = y_k + f(t_k, y_k)\Delta t}$$

- Backward Euler

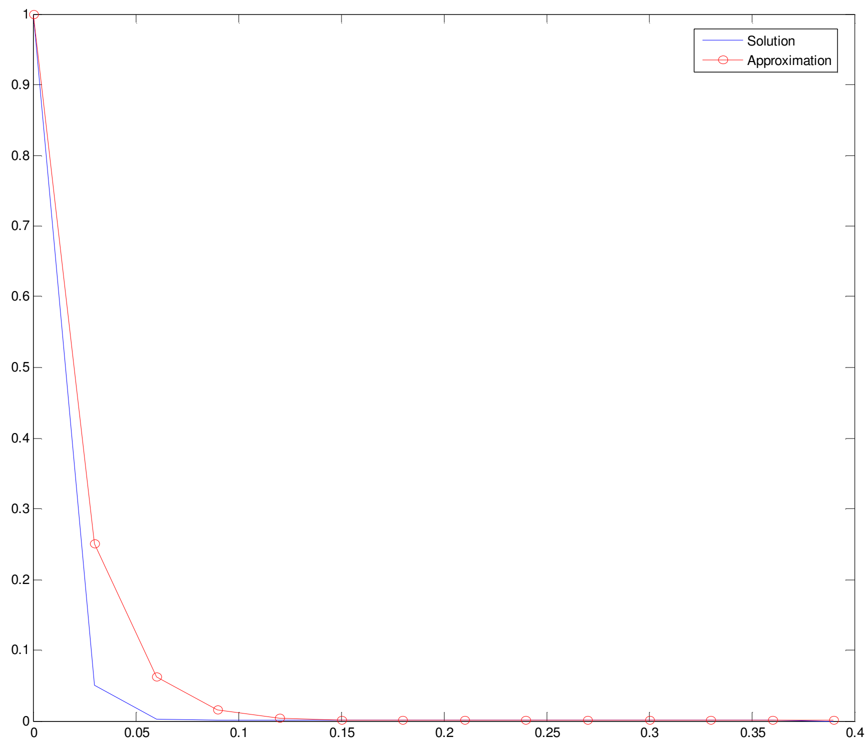
$$y_{k+1} = y_k + \Delta t \, y'_{k+1}$$
$$y'_{k+1} = f(y_{k+1}, t_{k+1})$$

$$\boxed{y_{k+1} = y_k + f(t_{k+1}, y_{k+1})\Delta t}$$

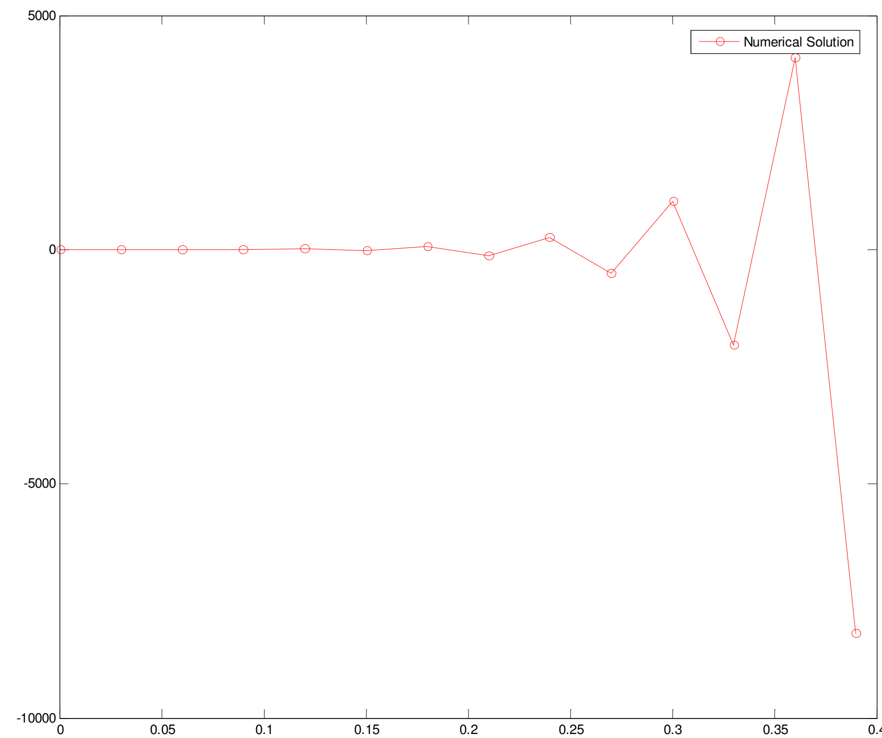
Example:

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

Exact Solution



Backward Euler and
Analytical Solution



Forward Euler

(Δt=0.03)

Other Popular Algorithms for Stiff IVPs

The family of BDF methods (Backward-Difference Formulas):

- BDF of 1st order: $y_{n+1} = y_n + hy'_{n+1}$
- BDF of 2nd order: $y_{n+1} = \frac{4}{3} y_n - \frac{1}{3} y_{n-1} + \frac{2}{3} hy'_{n+1}$
- BDF of 3rd order: $y_{n+1} = \frac{18}{11} y_n - \frac{9}{11} y_{n-1} + \frac{2}{11} y_{n-2} + \frac{6}{11} hy'_{n+1}$
- BDF of 4th order: $y_{n+1} = \frac{48}{25} y_n - \frac{36}{25} y_{n-1} + \frac{16}{25} y_{n-2} - \frac{3}{25} y_{n-3} + \frac{12}{25} hy'_{n+1}$
- BDF of 5th order: $y_{n+1} = \frac{300}{137} y_n - \frac{300}{137} y_{n-1} + \frac{200}{137} y_{n-2} - \frac{75}{137} y_{n-3} + \frac{12}{137} y_{n-4} + \frac{60}{137} hy'_{n+1}$

Bill Gear
Inventor of BDF, 1970

The Two Key Attributes of a Numerical Integrator

- Two attributes are relevant when considering a numerical integrator for finding an approximation of the solution of an IVP

  - The STABILITY of the numerical integrator

  - The ACCURACY of the numerical integrator

Numerical Integration Formula: The STABILITY Attribute

- The stability question:
  - How big can I choose the integration step-size Δt and be safe?
    - Tough question, answered in a Numerical Analysis class
- Different integration formulas, have different stability regions
- You'd like to use an integration formula with large stability region:
  - Example: Backward Euler, BDF methods, Newmark, etc.
- Why not always use these methods with large stability region?
  - There is **no free lunch**: these methods are implicit methods that require the solution of an algebra problem at each step (we'll see this on Th)

Numerical Integration Formula : The ACCURACY Attribute

- The accuracy question:
  - How accurate is the formula that I'm using?
  - If I start decreasing Δt , how will the accuracy of the numerical solution improve?
    - Tough question answered in a Numerical Analysis class
- Examples:
  - Forward and Backward Euler: accuracy $O(\Delta t)$
  - RK45: accuracy $O(\Delta t^4)$
- Why not always use methods with high accuracy order?
  - There is **no free lunch**: these methods usually have very small stability regions
  - Therefore, you are limited to very small values of Δt

MATLAB Support for solving IVP
[3 slide detour]

Ordinary Differential Equations (Initial Value Problem)

- An ODE + initial value: $\begin{cases} \dot{y} = f(t, y) \\ y(t_0) = y_0 \end{cases}$

- Use ***ode45*** for non-stiff IVPs and ***ode23t*** for stiff IVPs (concept of "stiffness" discussed shortly)

```
[t,y] = ode45(odefun,tspan,y0,options)
```

```
function dydt = odefun(t,y)
```

```
Initial value
```

```
[initialtime  finaltime]
```

- Use ***odeset*** to define `options` parameter above

IVP Example (MATLAB at work):

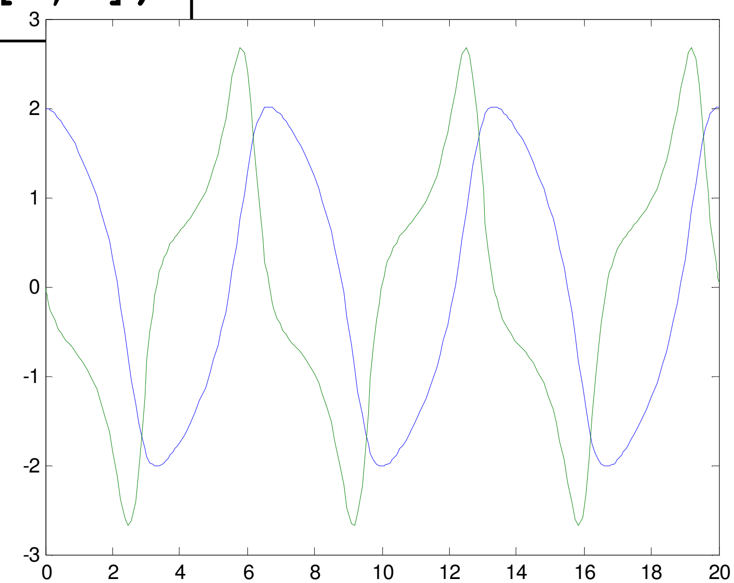
$$\ddot{y}_1 - (1 - y_1^2)\dot{y}_1 + y_1 = 0$$

$$\dot{y}_1 = y_2$$

$$\dot{y}_2 = (1 - y_1^2)y_2 - y_1$$

```
function dydt = myfunc(t,y)
dydt=zeros(2,1);
dydt(1)=y(2);
dydt(2)=(1-y(1)^2)*y(2)-y(1);
```

```
» [t,y]=ode45('myfunc',[0 20],[2;0])
```



Note:
Help on ***odeset*** to set options for more **accuracy** and other useful utilities like drawing results during solving.

ODE solvers in MATLAB

Solver	Problem Type	Order of Accuracy	When to use
ode45	Nonstiff	Medium	Most of the time. This should be the first solver tried
ode23	Nonstiff	Low	For problems with crude error tolerances or for solving moderately stiff problems.
ode113	Nonstiff	Low to high	For problems with stringent error tolerances or for solving computationally intensive problems
ode15s	Stiff	Low to medium	If ods45 is slow because the problem is stiff
ode23s	Stiff	Low	If using crude error tolerances to solve stiff systems and the mass matrix is constant
ode23t	Moderately stiff	Low	For moderately stiff problems is you need a solution without numerical damping
ode23tb	Stiff	Low	If using crude error tolerances to solve stiff systems