ME751
Advanced Computational Multibody Dynamics

October 14, 2016

Dan Negrut
University of Wisconsin-Madison

Quote of the Day

[courtesy of Victor]

"I can control my destiny, but not my fate. Destiny means there are opportunities to turn right or left, but fate is a one-way street. I believe we all have the choice as to whether we fulfill our destiny, but our fate is sealed. "

-- Paulo Coelho

"The two worst strategic mistakes to make are acting prematurely and letting an opportunity slip; to avoid this, the warrior treats each situation as if it were unique and never resorts to formulae, recipes or other people's opinions."

-- Paulo Coelho

“When you want something, all the universe conspires in helping you to achieve it.”

-- Paulo Coelho

"A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools."

-- Douglas Adams

Before we get started…

- Last Time:
  - Elements of the numerical solution of Initial Value Problems

- Today:
  - More on implicit integration methods: The Backward Differentiation Formula (BDF)
  - Numerical integration method for second order IVPs
  - Numerical method for the solution of DAEs of multibody dynamics

- Homework:
  - Posted online, due in one week

- Reading:
  - Additional slides provided on the class website
    - Deal w/ Runge-Kutta and Adams-Moulton integration formulas
  - Handout regarding the coordinate partitioning approach to solving the constrained equations of motion [AO]

Implicit Methods, The Unpleasant Part

- Why not always use implicit integration methods?
- Implicit methods come with some baggage: you need to solve an equation (or system of equations) at *each* integration time step t_n
- Specifically, look at Backward Euler. At each t_n, you need to solve for y_n. This is a nonlinear equation whenever $f(t, y)$ is a nonlinear function (which is almost always the case)

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

- Solving nonlinear systems: not that much fun

Implicit Integration: Solving the Nonlinear System

- Note that if you are dealing with a system of ODEs, that is, if $\boldsymbol{y}$ is a vector quantity, you have to solve not a nonlinear equation, but a nonlinear system of equations:

$$\mathbf{g}(\mathbf{y}_n) \equiv \mathbf{y}_n - \mathbf{y}_{n-1} - h\mathbf{f}(t_n, \mathbf{y}_n) = \mathbf{0}$$

- We'll assume that the system above is a nonlinear one (almost always the case)
  - Points that can be made in this context:
    - Point 1: The "functional iteration" approach to finding $\boldsymbol{y}_n$
    - Point 2: Newton Iteration
    - Point 3: Approximating the Jacobian associated with the nonlinear system

Discussion Point 1: The Functional Iteration

- The basic idea is to solve the system through a functional iteration
  - The superscript (ν+1) indicates the iteration count
  - An initial guess $\mathbf{y}_n^{(0)}$ is needed to "seed" the iterative process

$$\mathbf{y}_n^{(\nu+1)} = \mathbf{y}_{n-1} + h\mathbf{f}(t_n, \mathbf{y}_n^{(\nu)})$$

- If this defines a contractive map in a Banach space, the functional iteration leads to a fixed point, which is the solution of interest

- However, for this to be a contractive mapping in some norm, the following needs to hold in a neighborhood of the solution y_n:

$$h \, \| \frac{\partial \mathbf{f}}{\partial \mathbf{y}} \| < 1$$

- For stiff systems, the above matrix norm is very large. This requires small h. And this defeats the purpose of using an implicit formula…

Part of Future Homework

- Analyze the restrictions on the step-size imposed by the requirement that the functional iteration convergence for the following IVP:

$$\begin{cases} \dot{y} = \lambda(ty^2 - 1/t) - 1/t^2 \\ y(1) = 1 \end{cases} \qquad t \in [1, 10]$$

- Here $\lambda < 0$ is a parameter that determines the stiffness of the IVP
- Note that for $\lambda = -1$, the solution is $y(t) = 1/t$

Discussion Point 2: The Newton Iteration

- This is simply applying Newton's method to solve the system

$$\mathbf{g}(\mathbf{y}_n) = \mathbf{0}$$

- Boils down to carrying out the iterative process:

Evaluating this term is where most of the computational effort is spent

$$\begin{cases} \left[\dfrac{\partial \mathbf{g}}{\partial \mathbf{y}}\right] \cdot \Delta \mathbf{y}_n^{(\nu)} = -\mathbf{g}(\mathbf{y}_n^{(\nu)}) \\ \mathbf{y}_n^{(\nu+1)} = \mathbf{y}_n^{(\nu)} + \Delta \mathbf{y}_n^{(\nu)} \end{cases}$$

  - The superscript (ν+1) indicates the iteration count
  - An initial guess $\mathbf{y}_n^{(0)}$ is needed to "seed" the iterative process (take it $\boldsymbol{y}_{n-1}$)
  - Iterative process stopped when correction is smaller than prescribed value
    - NTOL depends on the local error bound that the user aims to achieve
    - Stop when

$$\|\Delta \mathbf{y}_n^{(\nu)}\| \leq \text{NTOL}$$

Discussion Point 2: The Newton Iteration

- "Iteration matrix":

$$\left[\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right] = \mathbf{I} - h\,\frac{\partial \mathbf{f}}{\partial \mathbf{y}} \in \mathbb{R}^{m \times m}$$

- Note that the iteration matrix is guaranteed to be nonsingular for small enough values of the step-size h
- Typically, the approach does not place harsh limits on the value of the step size
- The iteration matrix is not updated at each iteration.
  - Updated only when convergence in Newton iteration gets poor
- Note that each update also requires LU factorization of iteration matrix
  - Adding insult to injury…

Exercise
[AO]

- For IVP below, find iteration matrix when solved with B. Euler
  - Find it analytically
  - Find it using finite differences
  - In both cases use $y[1] = 0 \quad \& \quad y[2] = 2$ for evaluating the matrix
  - Both α and β are assumed to be constants (some parameters)

$$\text{IVP:} \quad \begin{cases} \dot{y}[1] = \alpha - y[1] - \frac{4y[1]y[2]}{1+y^2[1]} \\ \dot{y}[2] = \beta y[1] \left(1 - \frac{y[2]}{1+y^2[1]}\right) \\ y[1](0) = 0 \quad y[2](0) = 2 \end{cases} \qquad t \in [0, 20]$$

Partial Discussion, Point 3: The Newton Iteration

- Iteration matrix, zoom in on entry (i, j):

$$\left[\frac{\partial \mathbf{g}}{\partial \mathbf{y}}\right]_{ij} = I_{ij} - h\,\frac{\partial f[i]}{\partial y[j]} \in \mathbb{R} \qquad \text{where} \qquad I_{ij} = \begin{cases} 1 & \text{if} \quad i = j \\ 0 & \text{if} \quad i \neq j \end{cases}$$

- The expensive part is computing the partial derivative $\dfrac{\partial f[i]}{\partial y[j]}$

- Ideally, you can compute this exactly

- Otherwise, compute using finite differences:

$$\frac{\partial f_i}{\partial y_j} = \lim_{\delta \to 0} \frac{f_i(y_1, \ldots, y_j + \delta, \ldots, y_m) - f_i(y_1, \ldots, y_j, \ldots, y_m)}{\delta} \qquad \Rightarrow \qquad \frac{\partial f_i}{\partial y_j} \approx \frac{f[i](y_1, \ldots, y[j] + \Delta, \ldots, y_m) - f_i(y_1, \ldots, y_j, \ldots, y_m)}{\Delta}$$

Be aware of notational inconsistency; employed to keep things simple

- Very amenable to parallel computing

Regarding Discussion of Point 3: Approximating the Jacobian

- Postpone full discussion for 20 slides or so
- Look into "Point 3" when integrating the differential equations associated with the time evolution of a mechanical system
  - Dealing with a second order IVP

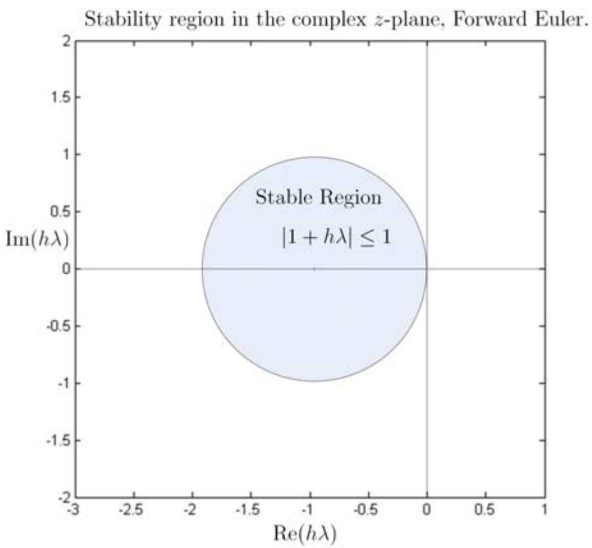
[Reason why we bother w/ Implicit Integration Formulas]

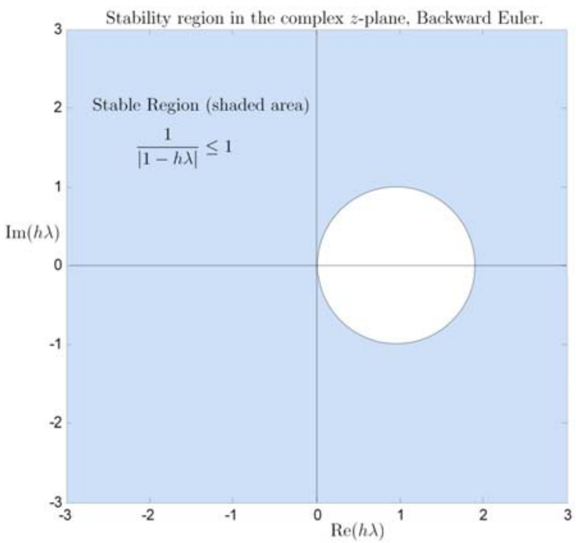
A-Stable Integration Methods

- Definition, A-Stability
  - First, recall the region of absolute stability: defined in conjunction with the test IVP, represents the region where $h\lambda$ should land so that

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

  - By definition, a numerical integration scheme is said to be A-stable if its region of absolute stability covers the entire left half-plane
    - Forward Euler is not A-stable
    - Backward Euler is A-stable





BDF Methods

BDF Methods

- BDF stands for Backward Differentiation Formula
- Proposed by Bill Gear in 1970
  - Super nice person
  - Back in ’70s he was a professor in Comp. Science at UIUC
  - Former director of NEC Research Institute
  - Professor Emeritus, Princeton

Bill Gear

- BDF methods are the most widely used implicit multistep methods
- BDF methods, together with HHT methods, are the two most used to integration formulas in ADAMS (the software package)

BDF Methods:
How to produce them

- Here's what Bill Gear came up with

  - Use solution values $y_n, \ldots, y_{n-k}$ to generate a polynomial that approximates $y(t)$
    - To this end, use the most recent $k+1$ values of the solution

  - Take the time derivative of this interpolation polynomial at time t_n

  - The value obtained should be an approximation of the time derivative of $y(t)$. By setting this time derivative to $f(t_n, y_n)$ one gets a BDF method

Exercise
[AO]

- Find the BDF that uses y_n, y_{n-1}, y_{n-2} in approximating the solution of t_n

BDF Methods

- The BDF methods are implicit methods

- With $-_3$=1, they assume the form

$$\sum_{i=0}^{k} \alpha_i y_{n-i} = h\beta_0 f(t_n, y_n)$$

- NOTE: for $k > 6$, the absolute stability region of the resulting BDF methods is so small that they are useless

- Example: BDF of order two

$$y_n - \tfrac{4}{3} y_{n-1} + \tfrac{1}{3} y_{n-2} = \tfrac{2}{3} h f(t_n, y_n)$$

Or equivalently,

$$y_n = \tfrac{4}{3} y_{n-1} - \tfrac{1}{3} y_{n-2} + \tfrac{2}{3} h f(t_n, y_n)$$

- Since BDF is a multistep method you'll need to 'prime' the method; i.e., providing the solution for a number of steps before the method is self sufficient

BDF Methods: $\sum_{i=0}^{k} \alpha_i y_{n-i} = h\beta_0 f(t_n, y_n)$

- Table below provides convergence order p, the number of steps k of the M method, the coefficients $\dagger_0$, and the values of the coefficients $-_3$, $-_4$,…

p	k	$\dagger_0$	$-_3$	$-_4$	$-_5$	$-_6$	$-_7$	$-_8$	$-_9$
1	1	1	1	-1					
2	2	2/3	1	-4/3	1/3				
3	3	6/11	1	-18/11	9/11	-2/11			
4	4	12/25	1	-48/25	36/25	-16/25	3/25		
5	5	60/137	1	-300/137	300/137	-200/137	75/137	-12/137	
6	6	60/147	1	-360/147	450/147	-400/147	225/147	-72/147	10/147

- Example: based on the table above, the second order BDF formula (k=2) is

$$y_n - \frac{4}{3} y_{n-1} + \frac{1}{3} y_{n-2} = \frac{2}{3} h f(t_n, y_n) \quad \Rightarrow \quad y_n = \frac{4}{3} y_{n-1} - \frac{1}{3} y_{n-2} + \frac{2}{3} h f(t_n, y_n)$$

BDF Method: Implementation Details (Newton Iteration)

- The approach adopted for stiff problems is to solve the discretization nonlinear system by using Newton-Raphson or some variant
- Recall the nonlinear algebraic problem that we have to solve at each time step t_n:

$$\sum_{i=0}^{k} \alpha_i \mathbf{y}_{n-i} = h\beta_0 \mathbf{f}(t_n, \mathbf{y}_n)$$

- It boils down to solving the following system of nonlinear equations:

$$\mathbf{g}(\mathbf{y}_n) \equiv \mathbf{y}_n - h\beta_0 \mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{c}_n^{\mathbf{y}}(l) = \mathbf{0}$$

- Note that $\mathbf{c}_n^{\mathbf{y}}(l)$ is a constant quantity that only depends on previous values of the unknown function y (l stands for the order of the BDF):

$$\mathbf{c}_n^{\mathbf{y}}(l) = \sum_{i=1}^{k} \alpha_i \mathbf{y}_{n-i}$$

BDF Method: Implementation Details (Newton Iteration)

- The Newton-Raphson iteration assumes the expression:

$$\begin{cases} \left(\mathbf{I} - h\beta_0 \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_n, \mathbf{y}_n^{(\nu)})\right) \Delta \mathbf{y}_n^{(\nu)} = -\mathbf{g}(t_n, \mathbf{y}_n^{(\nu)}) \\ \mathbf{y}_n^{(\nu+1)} = \mathbf{y}_n^{(\nu)} + \Delta \mathbf{y}_n^{(\nu)} \end{cases}$$

- The starting point is usually chosen like

$$\mathbf{y}_n^{(0)} = \mathbf{y}_{n-1}$$

- In practice, a modified Newton method is used since in the classical Newton-Raphson algorithm
  - Computing the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_n, \mathbf{y}_n^{(\nu)})$ at each iteration is expensive
  - Computing at each iteration the **LU** factorization of the iteration matrix $\mathbf{\Psi} \equiv \mathbf{I} - h\beta_0 \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_n, \mathbf{y}_n^{(\nu)})$ is expensive

BDF Method: Implementation Details

The Modified Newton step

- The modified-Newton assumes the form (note the (0) superscript):

$$\begin{cases} \left(\mathbf{I} - h\beta_0 \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t_n, \mathbf{y}_n^{(0)})\right) \Delta \mathbf{y}_n^{(\nu)} = -\mathbf{g}(t_n, \mathbf{y}_n^{(\nu)}) \\ \mathbf{y}_n^{(\nu+1)} = \mathbf{y}_n^{(\nu)} + \Delta \mathbf{y}_n^{(\nu)} \end{cases}$$

- In other words, the iteration matrix is evaluated once at the beginning of the step based on the predicted value $\mathbf{y}_n^{(0)}$
- The coefficient matrix is factored and subsequently used for all the iterations taken during that step
- This is the approach embraced by industrial strength integrators
- Unless we fall back on this “modified” Newton-method flavor, the numerical solution is going to be very slow

Supplemental Exercise

- Plot the absolute stability regions for the BDF formulas up to order 6
- Comment on the size of the region of absolute stability

Supplemental Exercise

- Prove that the BDF method with k=4 is convergent with order 4
- Approach:
  - Compute the roots of the characteristic equations to prove zero-stability
  - Verify that the order conditions are satisfied up to order 4
  - Use theorem that says that

    Zero-stability + Accuracy to order p , Convergence of order p

Supplemental Exercise

- Generate the convergence plot for the BDF method of order 6 for the following IVP:

$$\begin{cases} \dot{y} & = & -5ty + \frac{5}{t} - \frac{1}{t^2} \\ y(1) & = & 1 \end{cases}$$

- Use the analytical solution, that is, y(t)=1/t, t 5 [1,4] to generate the starting points (history) required by the integration formula
  - Note that in practice you can't count on this break for the starting points, so you will have to use RK methods or gradually increase the order of the method as past history becomes available

[New Topic]

Handling 2nd Order IVP

- Example:

$$\begin{cases} m\ddot{x} + c\dot{x}^3 + kx^3 = sin(2t) \\ x(0) = x_0 \qquad \rightarrow \quad \text{given to you} \\ \dot{x}(0) = v_0 \qquad \rightarrow \quad \text{given to you} \end{cases}$$

- Remarks, assumptions, notation used:
  - EOM for a mass-spring-damper system, see ME340 for derivation of EOM.
  - m, c, k - mass, damping coefficient, spring constant, respectively
  - Spring is nonlinear, so is damping (if they were linear there was no need to Newton method to solve the ensuing problem)
  - A time periodic force, $sin(2t)$, acts on the mass m
- We are in the business of finding approximations for x and $\dot{x}$, or x and v, given the model (through the m, c, k parameters) and the force acting on the mass
  - In other words, we need to find the position and velocity of the body as a function of time t
- We assume that c is large, which leads to a damped problem – you should use an implicit integrator to efficiently find the solution of this IVP

Outcome, Dynamics Analysis

[Nonlinear Mass-Spring-Damper]

Model Params.

$m = 2$

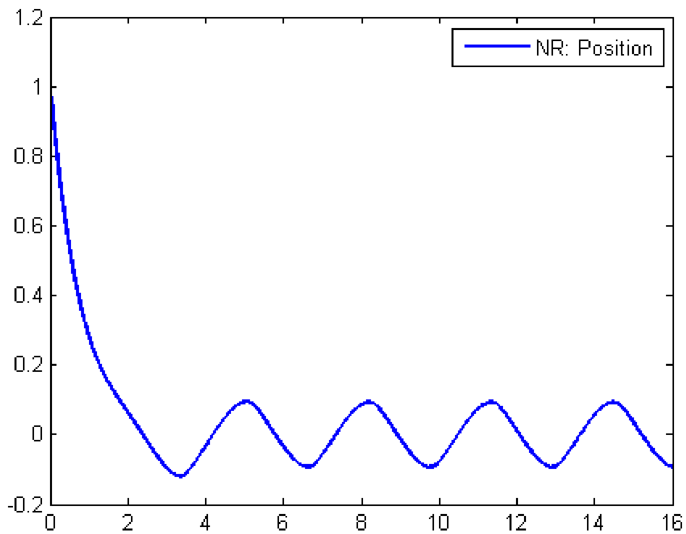
$c = 200$

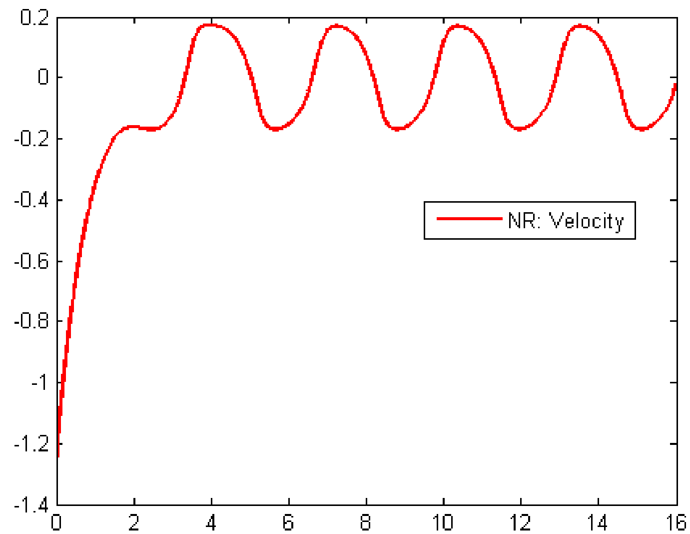
$k = 400.$

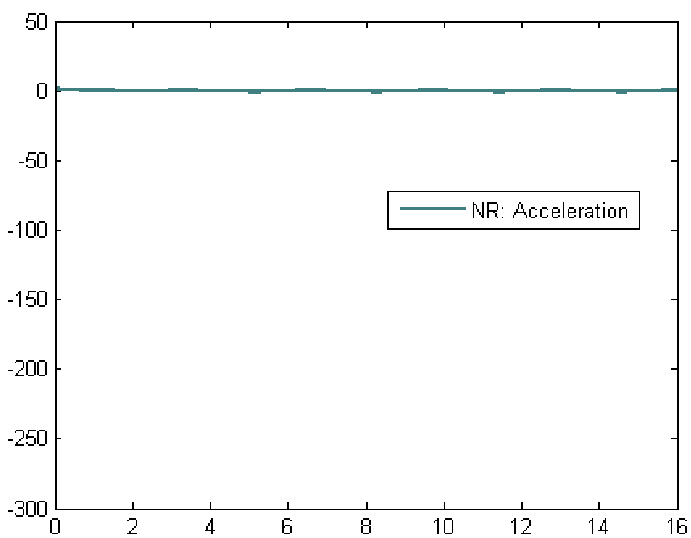
ICs:

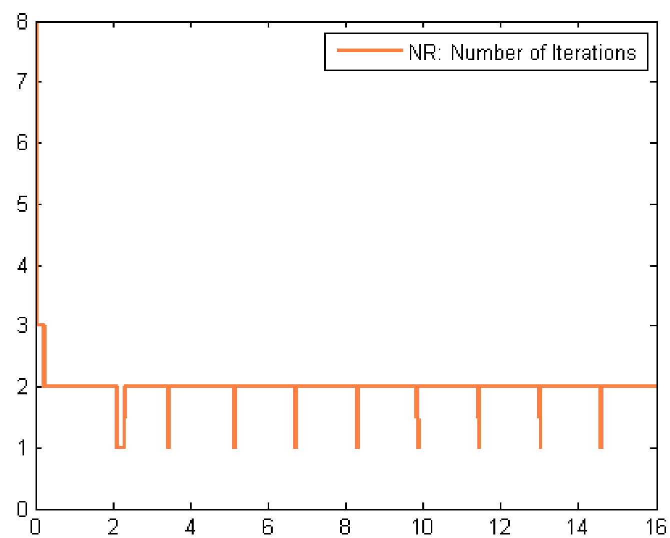
$x_0 = 1$ and $v_0 = -1.$

NOTE: x axis is time









Dealing w/ the 2nd Order IVP

- Two ways to solve 2^{nd} order IVP, they produce the same result but take different perspectives on solving the same problem:
  - (A) Reduce 2^{nd} order IVP to a first order IVP of dimension two and apply your favorite implicit integration formula (say BDF) – **we'll not work with this**
  - (B) Keep the IVP as is, and make a simple change to your favorite implicit integration formula (**our approach**)
- We'll work with the approach (B), and in the context of this approach, we'll use BDF
- In (B), you have to use the BDF formula **twice**: once to get from acceleration to velocity, and once again to get from velocity to position

[Dealing w/ the 2nd Order IVP, continued]

Keeping $\dot{\mathbf{y}}$ (instead of $\mathbf{f}(t, \mathbf{y})$) into the Picture

- Second order BDF (can consider any BDF formula, no restriction):

$$y_n = \frac{4}{3} y_{n-1} - \frac{1}{3} y_{n-2} + \frac{2}{3} h f(t_n, y_n) \tag{1}$$

- Equivalently (different way of looking at the same thing),

$$y_n = \frac{4}{3} y_{n-1} - \frac{1}{3} y_{n-2} + \frac{2}{3} h \dot{y}_n \tag{2}$$

- Small but important point to understand: In (1), the unknown is y_n; in (2), the unknown is $\dot{y}_n$.
  - When using (1), if you have y_n and need $\dot{y}_n$, you evaluate is as $\dot{y}_n = f(t_n, y_n)$
  - When using (2), if you have $\dot{y}_n$ and need y_n you simply plug $\dot{y}_n$ in (2) to get y_n
  - NOTE: As title of slide suggests, we'll work with (2)

Expressing the Position and Velocity as Functions of Acceleration

- For velocity:

$$v_n = \frac{4}{3}v_{n-1} - \frac{1}{3}v_{n-2} + \frac{2}{3}h\dot{v}_n$$

- That is,

$$v_n = \frac{4}{3}v_{n-1} - \frac{1}{3}v_{n-2} + \frac{2}{3}ha_n$$

- Handling the position x_n now:

$$x_n = \frac{4}{3}x_{n-1} - \frac{1}{3}x_{n-2} + \frac{2}{3}h\dot{x}_n$$

- That is,

$$x_n = \frac{4}{3}x_{n-1} - \frac{1}{3}x_{n-2} + \frac{2}{3}hv_n$$

- Based on the expression of v_n above, it follows that

$$x_n = \frac{4}{3}x_{n-1} - \frac{1}{3}x_{n-2} + \frac{2}{3}h(\frac{4}{3}v_{n-1} - \frac{1}{3}v_{n-2} + \frac{2}{3}ha_n) = \frac{4}{3}x_{n-1} - \frac{1}{3}x_{n-2} + \frac{8}{9}hv_{n-1} - \frac{2}{9}hv_{n-2} + \frac{4}{9}h^2a_n$$

Separating the Terms: Known vs. Unknown

- Important observation: take a look at the expression of the BDF integration formulas. No matter what BDF formula you use, the expression of x_n and v_n has two parts: one that depends on previous data (computed at t_{n-1}, t_{n-2}, t_{n-3}, t_{n-4}, etc. - in blue below), and one that depends on data that you don't know yet, but are about to compute at t_n (in red below)

$$x_n = \frac{4}{3}x_{n-1} - \frac{1}{3}x_{n-2} + \frac{8}{9}hv_{n-1} - \frac{2}{9}hv_{n-2} + \frac{4}{9}h^2 a_n$$
$$v_n = \frac{4}{3}v_{n-1} - \frac{1}{3}v_{n-2} + \frac{2}{3}ha_n$$

- For any BDF formula you use, say you use the one of order l, it is easy to see that x_n and v_n can be expressed as

$$\begin{aligned} x_n &= C_n^x(l) + \beta_0^2 h^2 a_n \\ v_n &= C_n^v(l) + \beta_0 h a_n \end{aligned}$$

Separating the Terms: The Known Terms

- Nomenclature:
  - The C in $C_n^x(l)$ is meant to suggest that $C_n^x(l)$ is quantity is a *constant*, which is evaluated based on values that were computed at previous time steps: t_{n-1}, t_{n-2}, t_{n-3}, t_{n-4}, etc.
  - The n in $C_n^x(l)$ is indicating that $C_n^x(l)$ is evaluated at time step t_n
  - The x in $C_n^x(l)$ is indicating that this constant $C_n^x(l)$ is the one associated with the position x. There is a constant term that is evaluated to enter the computation of v_n, like in $C_n^v(l)$
  - The l in $C_n^x(l)$ is indicating that $C_n^x(l)$ is as obtained for the BDF of order l. The higher the order, the more terms $C_n^x(l)$ and $C_n^v(l)$ will contain.
  - HOMEWORK: We just saw how to determine $C_n^x(2)$ and $C_n^v(2)$. Determine $C_n^x(1)$ and $C_n^v(1)$, as well as $C_n^x(3)$ and $C_n^v(3)$.

- **NOTE**: The relationships between position and acceleration, and between velocity and acceleration provided on the previous slide are **very important**. We will use it again when we solve the dynamics problem in the $\mathbf{r}-\mathbf{p}$ formulation, and here's how:

$$\mathbf{r}_n = \mathbf{C}_n^{\mathbf{r}}(l) + \beta_0^2 h^2 \ddot{\mathbf{r}}_n \qquad \dot{\mathbf{r}}_n = \mathbf{C}_n^{\dot{\mathbf{r}}}(l) + \beta_0 h \ddot{\mathbf{r}}_n$$

$$\mathbf{p}_n = \mathbf{C}_n^{\mathbf{p}}(l) + \beta_0^2 h^2 \ddot{\mathbf{p}}_n \qquad \dot{\mathbf{p}}_n = \mathbf{C}_n^{\dot{\mathbf{p}}}(l) + \beta_0 h \ddot{\mathbf{p}}_n$$

The Nonlinear System

- Recall the important expressions we derived on the previous slide:

$$x_n = C_n^x(l) + \beta_0^2 h^2 a_n$$

$$v_n = C_n^v(l) + \beta_0 h a_n$$

- Recall the expression of the EOM, discretized at time t_n (discretized here means that you take the continuum ODE problem and focus on the discrete form it assumes at time t_n):

$$m\ddot{x}_n + c\dot{x}_n^3 + kx_n^3 = sin(2t_n)$$

- Equivalently,

$$ma_n + cv_n^3 + kx_n^3 = sin(2t_n)$$

- Recall now that actually both v_n and x_n depend on a_n, according to our important expressions. With this in mind, define the following function g that only depends on a_n:

$$g(a_n) = ma_n + cv_n^3 + kx_n^3 - sin(2t_n)$$

- What we want to find is the root of $g(a_n)$; i.e., the solution of the equation

$$g(a_n) = 0$$