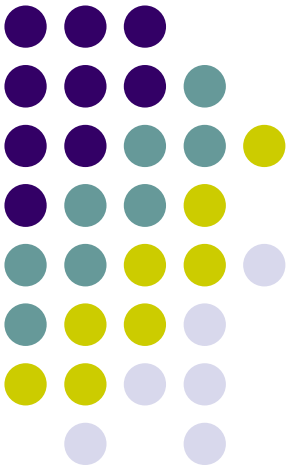
# ME451
# Kinematics and Dynamics of Machine Systems



## Dynamics of Planar Systems

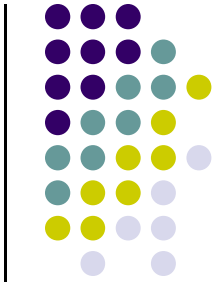
November 29, 2011

Elements of the Numerical Solution of the Dynamics Problem
Chapter 7



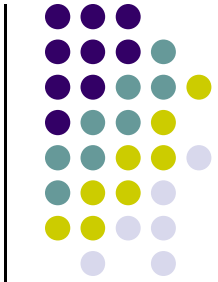
"How is education supposed to make me feel smarter? Besides, every time I learn something new, it pushes some old stuff out of my brain. Remember when I took that home winemaking course, and I forgot how to drive?"
Homer Simpson

# Before we get started…



- Last Time
  - Numerical solution of IVPs
  - Discretization schemes discussed:
    - Explicit: Forward Euler – expeditious but small stability region (can blow up)
    - Implicit: Backward Euler, BDF – requires solution of nonlinear system, typically much more stable than explicit methods
- Today
  - Look at a couple of examples for numerical solution of IVPs
  - Introduce the most basic approach for finding an approximation of the solution of the constrained equations of motion
    - We'll rely on the so called Newmark method
- Miscellaneous
  - Only MATLAB assignment this week
  - Emailed to you this morning, due on Tu, Dec. 6 at 11:59 pm

# Exam + Misc. Issues

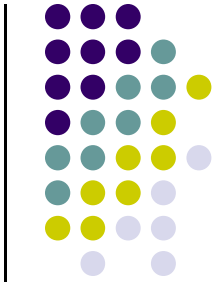


- Coming up on Thursday, Dec. 1st
  - Lecture at the usual time
  - Review at 5 pm (room TBA)
  - Midterm Exam at 7:15 pm – runs up to two hours long (room TBA)
  - Take home component of midterm exam due on Dec. 8 at 11:59 pm

- Midterm exam deals only with material covered 10/27 through 11/29
  - Dynamics of 2D systems
  - Solution of IVP
  - Numerical solution of the equations of motion
    - Uses material covered today (Newmark integration formula)

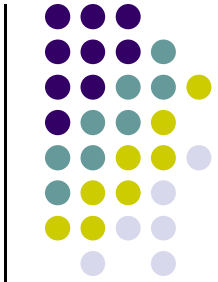
- Last day of lecture: Dec. 6
- Final exam is on Sat, Dec. 17 (2:45-4:45 PM)
  - 30-45 mins of questions followed by a problem that needs to be solved with simEngine2D
  - Final exam is comprehensive
- Last week of class dedicated to working on your final projects, which are due on Sat, Dec. 17 at 11:59 PM

# The Two Key Attributes of a Numerical Integrator



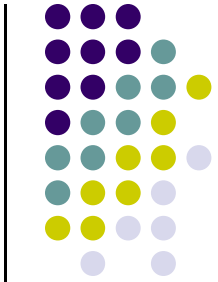
- Two attributes are relevant when considering a numerical integrator for finding an approximation of the solution of an IVP
  - The STABILITY of the numerical integrator
  - The ACCURACY of the numerical integrator

# Numerical Integration Formula: The STABILITY Attribute

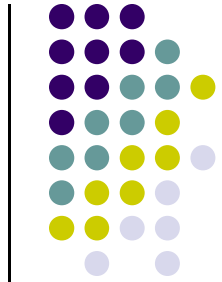


- The stability question:
  - How big can I choose the integration step-size $\Delta t$ before the numerical solution blows up?
    - Tough question, answered in a Numerical Analysis class
- Different integration formulas, have different stability regions
- You'd like to use an integration formula with 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 shortly)

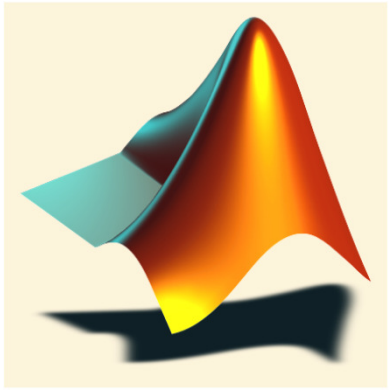
# Numerical Integration Formula : The ACCURACY Attribute



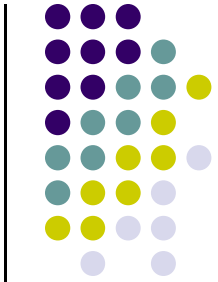
- The accuracy question:
  - How accurate is the formula that I'm using?
  - If I start decreasing $\Delta t$ , how will the accuracy of the numerical solution improve?
    - Tough question answered in a Numerical Analysis class
- Examples:
  - Forward and Backward Euler: accuracy $O(\Delta t)$
  - RK45: 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 $\Delta t$



MATLAB Support for solving IVP
[supplemental material]



# 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):

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

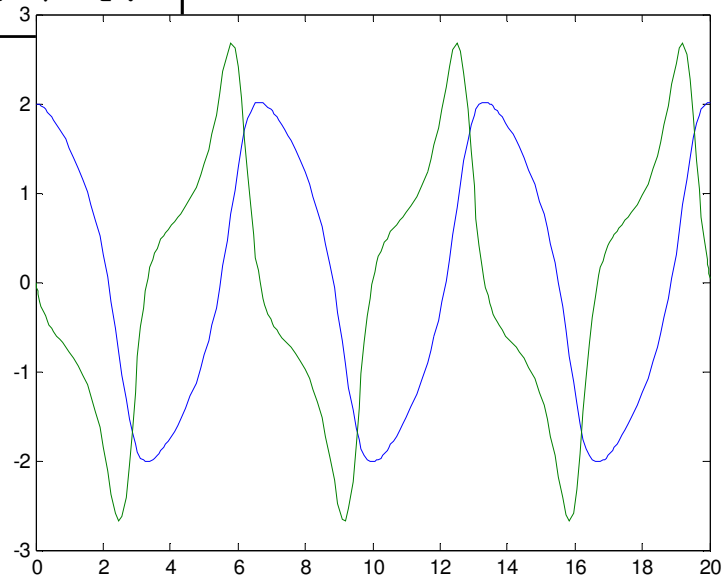
$$\dot{y}_1 = y_2$$

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

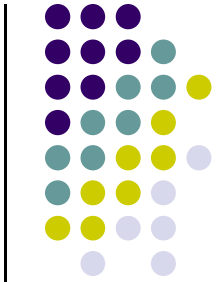
```
function dydt = myfunc(t,y)
dydt=zeros(2,1);
dydt(1)=y(2);
dydt(2)=(1-y(1)^2)*y(2)-y(1);
```

```
» [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.



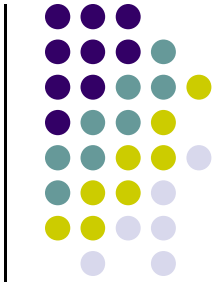
# ODE solvers in MATLAB



| 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 |

# Example

**[Solving IVP with Backward Euler]**



- Use Backward Euler with a step-size $\Delta$t=0.1 to solve the following IVP:

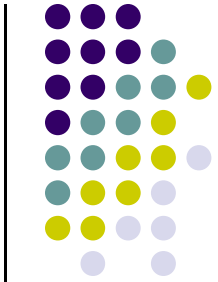
$$\begin{cases} \dot{y} &= -0.1y + \sin t \\ y(0) &= 0 \end{cases}$$

- Reason for looking at this example: understand that when using an implicit integration formula (such as Backward Euler) upon discretization you have to solve an algebraic problem

- Recall that “discretization” is the process of transforming the continuum problem into a discrete problem
  - Helps us get the values of the unknown function at the discrete grid points
  - It is the reason why you need an integration formula (also called discretization formula)

# Example

**[Solving IVP with Backward Euler]**

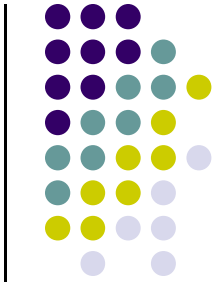


- This example shows how things can get tricky with implicit integrators
- Solve the following IVP using Backward Euler with a step-size $\Delta$t=0.1:

$$\begin{cases} \dot{y} &= -y^2 \\ y(0) &= 2.4 \end{cases}$$

# Example

**[Solving IVP with Backward Euler]**

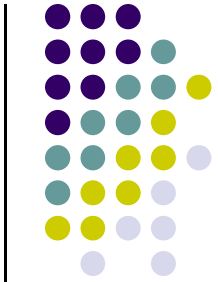


- This example shows why using an implicit integrator brings into the picture the Newton-Raphson method

- Solve the following IVP using Backward Euler with a step-size $\Delta$t=0.1:

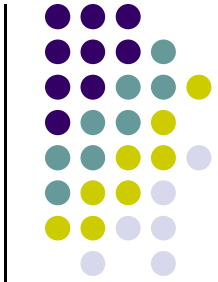
$$\begin{cases} \dot{y} &= \sin(y) \\ y(0) &= 0 \end{cases}$$

# Conclusions, Implicit Integration



- For stiff IVPs, Implicit Integration is the way to go
  - Because they have very good stability, implicit integration allows for step-sizes $\Delta t$ that might be orders of magnitude higher than if using explicit integration

- However, for most real-life IVP, an implicit integrator upon discretization leads to another nasty problem
  - You have to find the solution of a nonlinear algebraic problem
  - This brings into the picture the Newton-Raphson method (and its variants)
    - You have to deal with providing a starting point, computing the sensitivity matrix, etc.

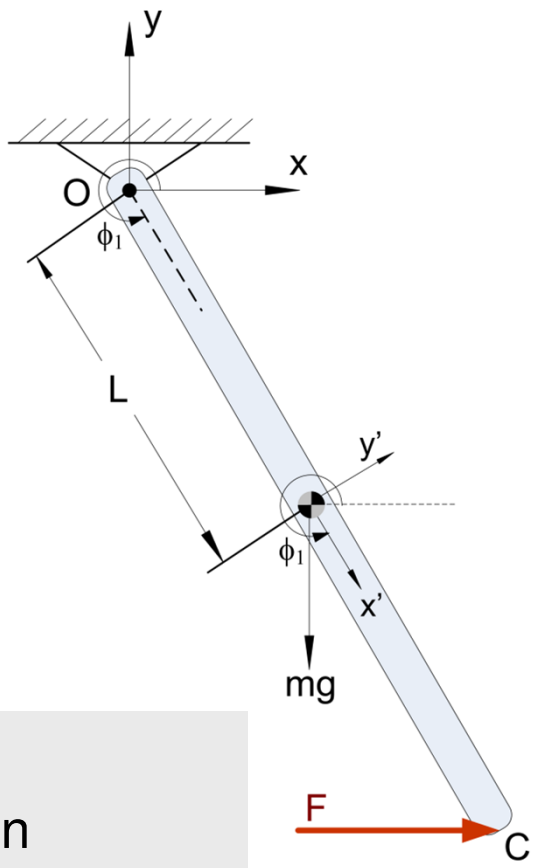


**End Numerical Methods for IVPs**

**Begin Numerical Methods for DAEs**

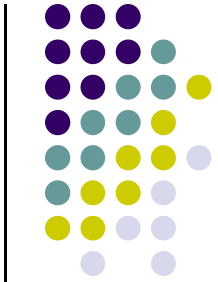
# Example: Find the time evolution of the pendulum

- Simple Pendulum:
  - Mass 20 kg
  - Length L=2 m
  - Force acting at tip of pendulum
    - F = 30 sin(2π t) [N]
  - Although not shown in the picture, assume RSDA element in revolute joint
    - k = 45 [Nm/rad] & $\phi_0$=3π/2
    - c = 10 [N/s]
  - ICs: hanging down, starting from rest




- Stages of the procedure (three):
  - Stage 1: Derive constrained equations of motion
  - Stage 2: Indicate initial conditions (ICs)
  - Stage 3: Apply numerical integration algorithm to discretize DAE problem and turn into algebraic problem

# Summary of the Lagrange form of the Constrained Equations of Motion



- Equations of Motion: $\mathbf{M}\ddot{\mathbf{q}} + \mathbf{\Phi}_{\mathbf{q}}^T \lambda = \mathbf{Q}^A$

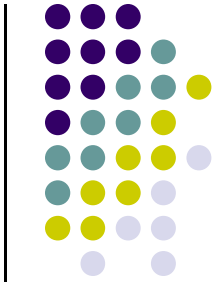
- Position Constraint Equations: $\mathbf{\Phi}(\mathbf{q}, t) = \mathbf{0}$

The Most Important Slide of ME451

- Velocity Constraint Equations: $\mathbf{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} = \nu$

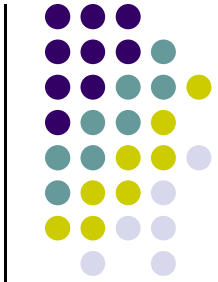
- Acceleration Constraint Equations: $\mathbf{\Phi}_{\mathbf{q}}\ddot{\mathbf{q}} = \gamma$

# What's special about ME451 problems?



- There are three things that make the ME451 dynamics problem challenging:
  - The problem is **not in standard form** $\dot{y} = f(t, y)$
  - Moreover, our problem is **not a first order** Ordinary Differential Equation (ODE) problem
    - Rather, it's a second order ODE problem, due to form of the equations of motion (contain the second time derivative of the positions)
  - In the end, it's **not even an ODE** problem
    - The unknown function **q**(t); that is, the position of the mechanism, is the solution of a second order ODE problem (first equation previous slide) but it must also satisfy a set of kinematic constraints at position, velocity, and acceleration levels, which are formulated as a bunch of algebraic equations
      - To conclude, you have to solve a set of differential-algebraic equations (DAEs)
      - DAEs are much tougher to solve than ODEs
      - This lecture is about using a numerical method to solve the DAEs of multibody dynamics

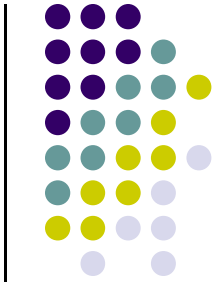
# Algorithm for Resolving Dynamics of Mechanisms



- If you have the EOM and ICs, how do you go about solving the problem?
- This is a research topic in itself
- We'll cover one of the simplest algorithm possible
  - Based on Newmark's integration formulas
    - That is, we are going to use Newmark's formulas to discretize our index 3 DAE of constrained multibody dynamics

# Solution Strategy

**[Important Slide]**



- This slide explains the strategy through which the numerical solution; i.e., an approximation of the actual solution of the dynamics problem, is produced

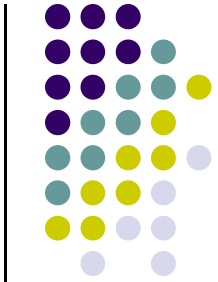
- **Stage 1**: two integration formulas (called Newmark in our case) are used to **express the positions and velocities as functions of accelerations**
  - These are also called “discretization formulas”

- **Stage 2**: everywhere in the constrained equations of motion, the positions and velocities are **replaced using the discretization formulas** and expressed in terms of the acceleration
  - This is the most important step, since through this “discretization” the **differential problem is transformed into an algebraic problem**
    - The algebraic problem, which effectively amounts to solving a nonlinear system, is approached using Newton’s method (so again, we need to produce a Jacobian)

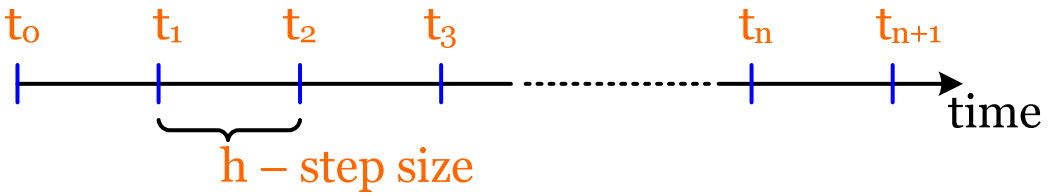
- **Stage 3**: **solve a nonlinear system** to find the acceleration and Lagrange multipliers

# Newmark Discretization/Integration Formulas

**[Two Slide Detour]**



- Newmark method (N.M. Newmark – 1957)
  - Goal: find the positions, velocities, accelerations and Lagrange multipliers on a grid of time points; i.e., at $t_0$, $t_1$, etc.




  - Newmark's method: integrate directly *second* order equations of motion:

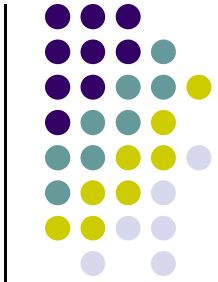
$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{F}(t) \quad \Leftrightarrow \quad \mathbf{M}\ddot{\mathbf{q}}_{n+1} + \mathbf{C}\dot{\mathbf{q}}_{n+1} + \mathbf{K}\mathbf{q}_{n+1} = \mathbf{F}(t_{n+1})$$

  - Newmark's Method: relies on a set of "discretization" or "integration" formulas that relate *position to acceleration* and *velocity to acceleration*:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\dot{\mathbf{q}}_n + \frac{h^2}{2}\left[(1-2\beta)\,\ddot{\mathbf{q}}_n + 2\beta\ddot{\mathbf{q}}_{n+1}\right]$$

$$\dot{\mathbf{q}}_{n+1} = \dot{\mathbf{q}}_n + h\left[(1-\gamma)\,\ddot{\mathbf{q}}_n + \gamma\ddot{\mathbf{q}}_{n+1}\right]$$

# Newmark (Cntd.)



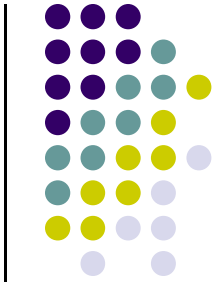
- Newmark Method
  - Accuracy: 1st Order
  - Initially introduced to deal with linear transient Finite Element Analysis
  - Stability: Very good stability properties
  - Choose $\beta$=0.3025, and $\gamma$=0.6 (these are two parameters that control the behavior of the method)

- If we write the equation of motion at each time $t_{n+1}$ one gets

$$\mathbf{M}\ddot{\mathbf{q}}_{n+1} + \mathbf{C}\dot{\mathbf{q}}_{n+1} + \mathbf{K}\mathbf{q}_{n+1} = \mathbf{F}(t_{n+1})$$

- Now is the time to replace $\mathbf{q}_{n+1}$ and $\dot{\mathbf{q}}_{n+1}$ with the discretization formulas (see previous slide)
- You end up with an algebraic problem in which the unknown is exclusively the acceleration $\ddot{\mathbf{q}}_{n+1}$

# The Problem at Hand



- Our rigid multibody dynamics problem is slightly more complicated than the Linear Finite Element problem used to introduce Newmark's discretization formulas
  - More complicated since we have some constraints that the solution must satisfy
  - We also have to deal with the Lagrange multipliers that come along with these constraints (from a physical perspective remember that they help enforce satisfaction of the constraints)

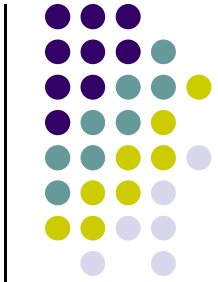
$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} = \mathbf{F}(t)$$

Linear Finite Element dynamics problem

$$\begin{cases} \mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^{T}(\mathbf{q})\lambda - \mathbf{Q}^{A}(\dot{\mathbf{q}}, \mathbf{q}, t) = \mathbf{0} \\ \boldsymbol{\Phi}\left(\mathbf{q}, t\right) = \mathbf{0} \end{cases}$$

Nonlinear multibody dynamics problem. Newmark algorithm still works as before, problem is slightly messier…

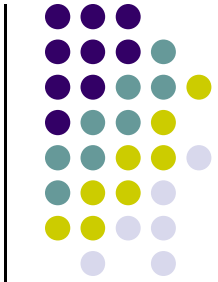
# Quantities of Interest…



- Generalized accelerations: $\ddot{\mathbf{q}}$
- Generalized velocities: $\dot{\mathbf{q}}$
- Generalized positions: $\mathbf{q}$
- Reaction Forces: $\lambda$

All these quantities are functions of time (they change in time)

# Stage 3: The Discretization of the Constrained Equations of Motion



- The Discretized Equations Solved at each Time-Step $t_{n+1}$:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{q}}_{n+1} + \mathbf{\Phi}_{\mathbf{q}}^{T}(\mathbf{q}_{n+1})\lambda_{n+1} - \mathbf{Q}^{A}(t_{n+1}, \mathbf{q}_{n+1}, \dot{\mathbf{q}}_{n+1}) = \mathbf{0} \\ \frac{1}{\beta h^2}\mathbf{\Phi}\left(\mathbf{q}_{n+1}, t_{n+1}\right) = \mathbf{0} \end{cases}$$

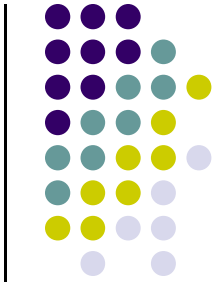
- Above you see $\mathbf{q}_{n+1}$ and $\dot{\mathbf{q}}_{n+1}$, but remember that they are functions of the accelerations:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + h\dot{\mathbf{q}}_n + \frac{h^2}{2}\left[(1-2\beta)\,\ddot{\mathbf{q}}_n + 2\beta\ddot{\mathbf{q}}_{n+1}\right]$$

$$\dot{\mathbf{q}}_{n+1} = \dot{\mathbf{q}}_n + h\left[(1-\gamma)\,\ddot{\mathbf{q}}_n + \gamma\ddot{\mathbf{q}}_{n+1}\right]$$

Again, these are Newmark's formulas that express the generalized position and velocity as functions of generalized accelerations

# The Discretization of the Constrained Equations of Motion (Cntd.)



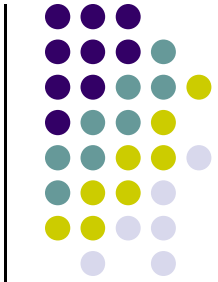
- Remarks on the discretization and its outcome:
  - Our unknowns are the accelerations and the Lagrange multipliers
  - The number of unknowns is equal to the number of equations
  - The equations that you have to solve now are algebraic and nonlinear
    - Differential problem has been transformed into an algebraic one
      - The new problem: find acceleration and Lagrange multipliers that satisfy

$$\Psi(\ddot{\mathbf{q}}_{n+1}, \lambda_{n+1}) \equiv \begin{bmatrix} \mathbf{M}\ddot{\mathbf{q}}_{n+1} + \mathbf{\Phi}_{\mathbf{q}}^{T}(\mathbf{q}_{n+1})\lambda_{n+1} - \mathbf{Q}^{A}(t_{n+1}, \mathbf{q}_{n+1}, \dot{\mathbf{q}}_{n+1}) \\ \\ \frac{1}{\beta h^2}\mathbf{\Phi}\left(\mathbf{q}_{n+1}, t_{n+1}\right) \end{bmatrix} = \mathbf{0}$$

    - You have to use Newton's method
      - This calls for the Jacobian of the nonlinear system of equations (chain rule will be used to simplify calculations)
      - This looks exactly like what we had to do when we dealt with the Kinematics analysis of a mechanism (there we solved $\mathbf{\Phi}(\mathbf{q},t)=0$ to get the positions $\mathbf{q}$)

[Three-Slide Detour]

# Chain Rule for Computing Partial Derivatives



- Define the function $\Pi$ as

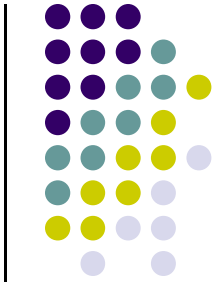
$$\mathbf{\Pi}(\ddot{\mathbf{q}}_{n+1}, \dot{\mathbf{q}}_{n+1}, \mathbf{q}_{n+1}) = \mathbf{M}\ddot{\mathbf{q}}_{n+1} + \mathbf{\Phi}_{\mathbf{q}}^{T}(\mathbf{q}_{n+1})\lambda_{n+1} - \mathbf{Q}^{A}(t_{n+1}, \mathbf{q}_{n+1}, \dot{\mathbf{q}}_{n+1})$$

- Note that in fact $\Pi$ is only a function of the acceleration
  - Based on Newmark's formulas, the velocity depends on acceleration and the position as well depends on acceleration.
- Therefore, I can define this new function $\Omega$ and clearly indicate that it only depends on acceleration:

$$\mathbf{\Omega}(\ddot{\mathbf{q}}_{n+1}) = \mathbf{\Pi}(\ddot{\mathbf{q}}_{n+1},\ \ \dot{\mathbf{q}}_{n+1}(\ddot{\mathbf{q}}_{n+1}),\ \ \mathbf{q}_{n+1}(\ddot{\mathbf{q}}_{n+1}))$$

[Three-Slide Detour]

# Chain Rule for Computing Partial Derivatives



- I'm interested in the sensitivity of Ω w.r.t. the acceleration. To this end, apply the chain rule:

$$\frac{\partial \boldsymbol{\Omega}}{\partial \ddot{\mathbf{q}}_{n+1}} = \frac{\partial \boldsymbol{\Pi}}{\partial \ddot{\mathbf{q}}_{n+1}} + \frac{\partial \boldsymbol{\Pi}}{\partial \mathbf{q}_{n+1}} \cdot \frac{\partial \mathbf{q}_{n+1}}{\partial \ddot{\mathbf{q}}_{n+1}} + \frac{\partial \boldsymbol{\Pi}}{\partial \dot{\mathbf{q}}_{n+1}} \cdot \frac{\partial \dot{\mathbf{q}}_{n+1}}{\partial \ddot{\mathbf{q}}_{n+1}}$$

- Note that:

$$\frac{\partial \mathbf{q}_{n+1}}{\partial \ddot{\mathbf{q}}_{n+1}} = \beta h^2 \mathbf{I}_{n\times n} \qquad \frac{\partial \dot{\mathbf{q}}_{n+1}}{\partial \ddot{\mathbf{q}}_{n+1}} = \gamma h \mathbf{I}_{n\times n}$$

- Therefore, I conclude that the sensitivity of Ω w.r.t. the generalized acceleration will be computed as

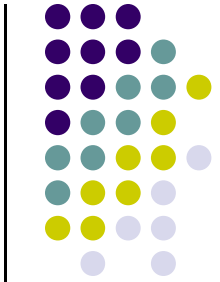
$$\frac{\partial \boldsymbol{\Omega}}{\partial \ddot{\mathbf{q}}_{n+1}} = \frac{\partial \boldsymbol{\Pi}}{\partial \ddot{\mathbf{q}}_{n+1}} + \frac{\partial \boldsymbol{\Pi}}{\partial \mathbf{q}_{n+1}} \beta h^2 \mathbf{I}_{n\times n} + \frac{\partial \boldsymbol{\Pi}}{\partial \dot{\mathbf{q}}_{n+1}} \gamma h \mathbf{I}_{n\times n}$$

- Equivalently,

$$\boxed{\frac{\partial \boldsymbol{\Omega}}{\partial \ddot{\mathbf{q}}_{n+1}} = \frac{\partial \boldsymbol{\Pi}}{\partial \ddot{\mathbf{q}}_{n+1}} + \beta h^2 \frac{\partial \boldsymbol{\Pi}}{\partial \mathbf{q}_{n+1}} + \gamma h \frac{\partial \boldsymbol{\Pi}}{\partial \dot{\mathbf{q}}_{n+1}}}$$

[Three-Slide Detour]

# Handling the Kinematic Constraints



- The focus here is on computing the sensitivity of the constraint equations with respect to the accelerations
- Note that I chose to scale the constraint equations by the factor $1/\beta h^2$. This is ok since both $\beta$ and h are constants

- The question is as follows:

$$\frac{\partial[\frac{1}{\beta h^2}\mathbf{\Phi}(\mathbf{q}_{n+1})]}{\partial \ddot{\mathbf{q}}_{n+1}} = \quad ?$$

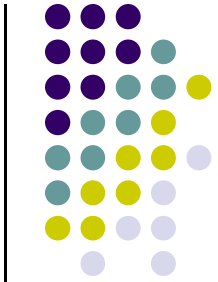
- Recall that

$$\frac{\partial \mathbf{q}_{n+1}}{\partial \ddot{\mathbf{q}}_{n+1}} = \beta h^2 \mathbf{I}_{n\times n} \qquad \frac{\partial \dot{\mathbf{q}}_{n+1}}{\partial \ddot{\mathbf{q}}_{n+1}} = \gamma h \mathbf{I}_{n\times n}$$

- Then, using the chain rule

$$\frac{\partial[\frac{1}{\beta h^2}\mathbf{\Phi}(\mathbf{q}_{n+1})]}{\partial \ddot{\mathbf{q}}_{n+1}} = \frac{\partial[\frac{1}{\beta h^2}\mathbf{\Phi}(\mathbf{q}_{n+1})]}{\partial \mathbf{q}_{n+1}} \cdot \frac{\partial \mathbf{q}_{n+1}}{\partial \ddot{\mathbf{q}}_{n+1}} = \frac{1}{\beta h^2}\mathbf{\Phi}_{\mathbf{q}}(\mathbf{q}_{n+1}) \cdot \beta h^2 \mathbf{I} = \mathbf{\Phi}_{\mathbf{q}}(\mathbf{q}_{n+1})$$

# Solving the Nonlinear System $\Psi$=0
# ~ Newton Method ~



- Based on Newmark Integration formulas, the Jacobian is calculated as:

$$\mathbf{J} \equiv \begin{bmatrix} \frac{\partial \mathbf{\Psi}}{\partial \ddot{\mathbf{q}}} & \frac{\partial \mathbf{\Psi}}{\partial \lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{M} + \beta h^2 \left( \frac{\partial (\Phi_q^T \lambda)}{\partial \mathbf{q}} - \frac{\partial \mathbf{Q}^A}{\partial \mathbf{q}} \right) - \gamma h \frac{\partial \mathbf{Q}^A}{\partial \dot{\mathbf{q}}} & \Phi_{\mathbf{q}}^T \\ \Phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix}$$

- Corrections Computed as :

$$\begin{bmatrix} \delta \ddot{\mathbf{q}} \\ \delta \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M} + \beta h^2 \left( \frac{\partial (\mathbf{\Phi}_q^T \lambda)}{\partial \mathbf{q}} - \frac{\partial \mathbf{Q}^A}{\partial \mathbf{q}} \right) - \gamma h \frac{\partial \mathbf{Q}^A}{\partial \dot{\mathbf{q}}} & \mathbf{\Phi}_{\mathbf{q}}^T \\ \mathbf{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix}^{-1} \cdot \mathbf{\Psi}(\ddot{\mathbf{q}}^{(old)}, \lambda^{old})$$

$$\begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix}^{(new)} = \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix}^{(old)} - \begin{bmatrix} \delta \ddot{\mathbf{q}} \\ \delta \lambda \end{bmatrix}$$

Note: subscripts “n+1” dropped to keep notation simpler