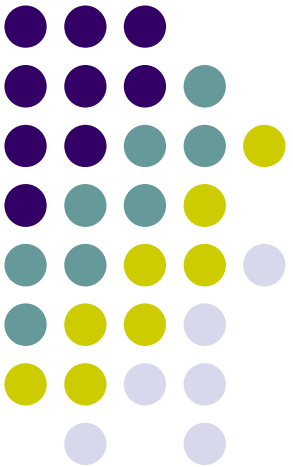
ME451
Kinematics and Dynamics of Machine Systems

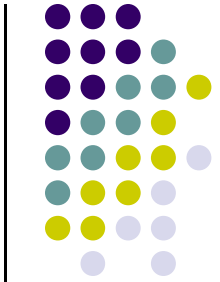


Dynamics of Planar Systems

December 9, 2010

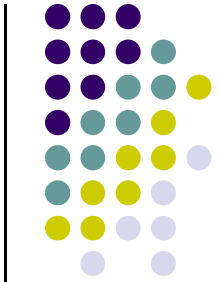
Solving Index 3 DAEs using Newmark Method

Before we get started…



- Last Time:
  - Started discussion on the numerical solution of DAEs of Multibody Dynamics using Newmark's method
- Today
  - Finish the discussion started last time
    - Marks the end of the Dynamics Analysis of Mechanical Systems chapter
  - Discuss the final exam
  - Class evaluation forms need to be filled out
  - Finish a bit early today, have to introduce Hammad in the graduate student seminar at 12 noon.
- Next time: last class
  - Equilibrium Analysis and Inverse Dynamics

Solution Strategy: Important Slide



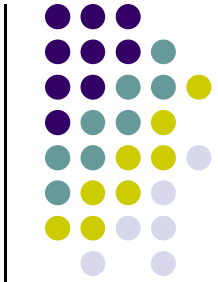
- This slide explains the strategy through which the numerical solution; i.e., an approximation of the actual solution of the dynamics problem, is produced

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

- Step 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)

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

Stage 3: The Discretization of the Constrained Equations of Motion



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

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

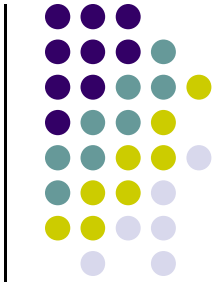
- Above you see $\mathbf{q}_{n+1}$ and $\dot{\mathbf{q}}_{n+1}$, but remember that they are functions of the accelerations:

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

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

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

Solving the Nonlinear System Ψ=0
~ Newton Method ~



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

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

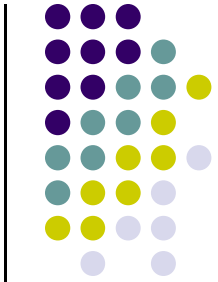
- Corrections Computed as :

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

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

Note: subscripts “n+1” dropped to keep presentation simple

Simple Pendulum Example: Looking at Stage 3



First.

Find initial conditions for general coordinate positions and velocities that satisfy position and velocity constraint equations

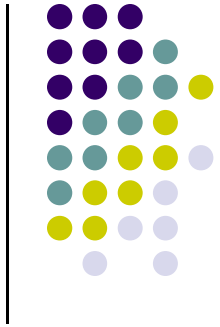
Use Newton's equations of motion along with the acceleration constraint equations to find general coordinate accelerations and lambdas at time = 0.

$$\begin{bmatrix} \mathbf{M} & \mathbf{\Phi}_{\mathbf{q}}^T \\ \mathbf{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q}^A \\ \gamma \end{bmatrix}$$

At each time step (solving for the state of the system at t_{n+1}…)

Increment $t_{n+1} = t_n + h$
Define initial guess for accelerations q_dotdot and lambdas (take values from previous time step t_n)

Iterate to solve nonlinear system obtained after Newmark discretization of DAEs.



Update t_{n+1} position and velocity using Newmark's formulas and the most recent values for acceleration and lambdas.

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

Get Jacobian of discretized equation. Use most recent q, q_dot, q_dotdot, and lambda at n+1

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

Use most recent values at n+1 to compute what each equation equals. Each equation will probably not equal zero. These values are called the residuals.

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

Yes. Refine guess.

Compute the correction vector.

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

residual

Correct the most recent acceleration and lambda to get better approximations for accelerations and lambdas.

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

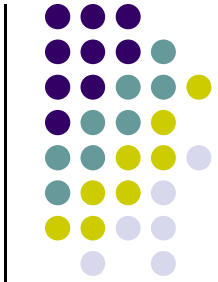
Compute the largest value present in the correction factor matrix and compare to a specified limit. This is the convergence check (largest correction is easily computed as infinity norm in MATLAB)

Is value greater than limit?

No.

Store acceleration and lambda for time t_{n+1}. Use this accelerations to compute the position and velocity at t_{n+1} and store these quantities as well.

Newton-Type Methods:
[Geometric Interpretation]

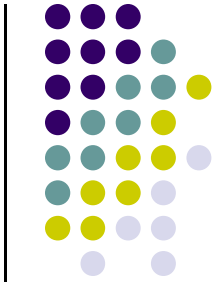


- Straight Newton-Raphson: when moving towards the root of the function, at each iteration you search in the direction given by the current tangent

- Modified Newton: when moving towards the root of the function, at each iteration you search in the direction given by the tangent evaluated at the point where you started the iterative process

- Quasi-Newton: when moving towards the root of the function, at each iteration you search in some direction that is not given by any tangent to the function. Yet, it is a direction that you computed cheaply and hopefully is not very different than the direction of the actual tangent

Solving the Dynamics Problem using Quasi-Newton Method



- What are we after? We want to solve for $\ddot{\mathbf{q}}_{n+1}$ and $\lambda_{n+1} = \mathbf{0}$ the following nonlinear system:

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

- Expression of $\mathbf{J}$, the Jacobian of the discretization nonlinear system:

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

- Here's what you are missing in simEngine2D:
  - Sensitivity of the reaction forces with respect to the generalized positions $\mathbf{q}$:

$$\frac{\partial (\Phi_q^T \lambda)}{\partial \mathbf{q}}$$

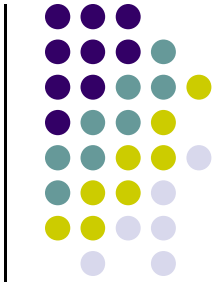
  - Sensitivity of the generalized applied forces with respect to the generalized positions $\mathbf{q}$:

$$\frac{\partial \mathbf{Q}^A}{\partial \mathbf{q}}$$

  - Sensitivity of the generalized applied forces with respect to the generalized velocities $\dot{\mathbf{q}}$:

$$\frac{\partial \mathbf{Q}^A}{\partial \dot{\mathbf{q}}}$$

Solving the Dynamics Problem using Quasi-Newton Method



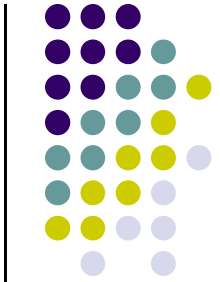
- Conclusion from previous slide: when solving the discretization nonlinear system $\Psi(\ddot{\mathbf{q}}_{n+1}, \lambda_{n+1}) = \mathbf{0}$, we are not going to work with the exact Jacobian $\mathbf{J}$, but with an approximation of it called $\tilde{\mathbf{J}}$.
- We will thus rely on a Quasi-Newton method since we dont have all the sensitivities available in simEngine2D
- Expression of $\mathbf{J}$, the actual Jacobian:

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

- Expression of $\tilde{\mathbf{J}}$, the substitute Jacobian:

$$\tilde{\mathbf{J}} \equiv \begin{bmatrix} \mathbf{M} & \Phi_{\mathbf{q}}^T \\ \Phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix}$$

Solving the Dynamics Problem using Quasi-Newton Method



- Quantities dropped from the expression of the actual Jacobian:

$$\beta h^2 \frac{\partial(\Phi_q^T \lambda)}{\partial \mathbf{q}} \qquad \beta h^2 \frac{\partial \mathbf{Q}^A}{\partial \mathbf{q}} \qquad \gamma h \frac{\partial \mathbf{Q}^A}{\partial \dot{\mathbf{q}}}$$

- Important question: When can we get away with neglecting these terms?
  - This should be ok if the step-size h is very small: $h \approx 0.001$. Note: this is a rule of thumb, not always small enough...
- All quantities that we decided to neglect are fairly straightforward to compute, but we are not going to pursue them in ME451
- All these quantities are evaluated fully in ADAMS, which works with an [approximately] accurate Jacobian

First.

Find initial conditions for general coordinate positions and velocities that satisfy position and velocity constraint equations

Use Newton's equations of motion along with the acceleration constraint equations to find general coordinate accelerations and lambdas at time = 0.

$$\begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}^T \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{Q}^A \\ \gamma \end{bmatrix}$$

At each time step (solving for the state of the system at t_{n+1}…)

Increment $t_{n+1} = t_n + h$
Define initial guess for accelerations q_dotdot and lambdas (take values from previous time step t_n)

Iterate to solve nonlinear system obtained after Newmark discretization of DAEs.

Update t_{n+1} position and velocity using Newmark's formulas and the most recent values for acceleration and lambdas.

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

Get Jacobian of discretized equation. Use most recent q, q_dot, q_dotdot, and lambda at n+1

$$\tilde{\mathbf{J}} \equiv \begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}^T \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix}$$

Use most recent values at n+1 to compute what each equation equals. Each equation will probably not equal zero. These values are called the residuals.

$$\begin{cases} \mathbf{M}\ddot{\mathbf{q}} + \boldsymbol{\Phi}_{\mathbf{q}}^T(\mathbf{q})\lambda - \mathbf{Q}^A(\dot{\mathbf{q}}, \mathbf{q}, t) = \mathbf{0} \\ \frac{1}{\beta h^2}\boldsymbol{\Phi}\,(\mathbf{q}, t) = \mathbf{0} \end{cases}$$

Yes. Refine guess.

Compute the correction vector.

$$\begin{bmatrix} \delta\ddot{\mathbf{q}} \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} \mathbf{M} & \boldsymbol{\Phi}_{\mathbf{q}}^T \\ \boldsymbol{\Phi}_{\mathbf{q}} & \mathbf{0} \end{bmatrix}^{-1} \cdot \boldsymbol{\Psi}(\ddot{\mathbf{q}}^{(old)}, \lambda^{old})$$

residual

Correct the most recent acceleration and lambda to get better approximations for accelerations and lambdas.

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

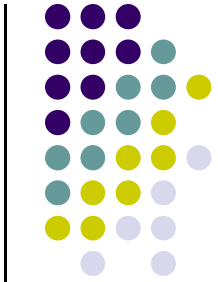
Compute the largest value present in the correction factor matrix and compare to a specified limit. This is the convergence check (largest correction is easily computed as infinity norm in MATLAB)

Is value greater than limit?

No.

Store acceleration and lambda for time t_{n+1}. Use this accelerations to compute the position and velocity at t_{n+1} and store these quantities as well.

Final Exam Info



- Tuesday, December 21, 5:05 PM. Room: **ME1245**

- For simEngine2D, please make sure you support all the modeling elements that you were assigned to handle in the MATLAB assignment 3:

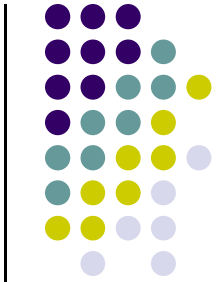
http://sbel.wisc.edu/Courses/ME451/2010/Documents/MATLAB/matlabAssignment03.pdf

- In terms of forces/torques, please make sure you support the ones you were assigned to handle in the MATLAB assignment 5:

http://sbel.wisc.edu/Courses/ME451/2010/Documents/MATLAB/matlabAssignment05.pdf

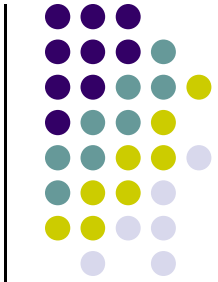
- Note: only the last 15 points (out of 100) of the final exam are tied to simEngine2D. In other words, if you don't even have a simEngine2D and answer all the other questions right you'll end up with a score of 85.

Final Exam, Rules of Engagement



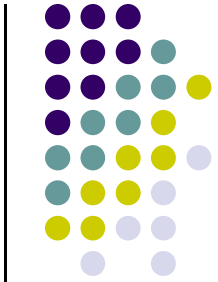
- Define an acf and adm pair of files associated with the type of analysis that you are supposed to carry out and the model that you were given.
- Run simulations (Kinematics, Dynamics, and Equilibrium) using the simulation engine.
- Generate a set of plots that show the time evolution of an attribute of the model (the motion of a point, the value of a reaction force as a function of time, etc.)
- Answer questions that are more theoretical in nature. For instance, "How has been the Lagrange Multiplier Theorem used in deriving the equations of motion?", "Why are initial conditions important in the context of Dynamics analysis?", etc.
- Email the TA and class instructor a zipped directory that contains your code, adm/acf files, png plots of your results, and the answer to the theoretical questions. The naming convention for this directory should be "LastNameME451.zip". For instance, "NegrutME451.zip". The answers to the theoretical questions should be typed in MS-Word in a file called FinalExam.doc[x].

Final Exam, Comments



- If your simEngine2D does not use MATLAB, it will be your responsibility to have the compiler support allowing you to run the software during the Final Exam.
- Running your simEngine2D code should also report the amount of time it took for completing a simulation. This information should also be included in the email to the TA & instructor.
- I will not insist on having simEngine2D that you use during the exam be implemented exclusively by you. However, in good faith, you will have to indicate in the email that you will be sending to the TA & the instructor the percentage of your contribution to the simEngine2D code that you are using for the exam. I will then understand that the remaining percent came from code written by other ME451 colleague[s]. This is fine, but should be acknowledged.
- If you contributed more than 66% to your simEngine2D, you qualify for entering the race for the fastest solver. Winning that race translates into an automatic A-grade in the course.
- One other automatic A grade *might* be assigned for the most general, flexible, and neatly organized simEngine2D code.

Course Evaluation



- Please provide comments on the back of the form
  - Your feedback is important
- If there were things that you didn't like please mention them
- If there were things that you like please mention them too
- What worked / What didn't work