ME451 Kinematics and Dynamics of Machine Systems

Dynamics of Planar Systems

December 1, 2011

Solving Index 3 DAEs using Newmark Method

© Dan Negrut, 2011
ME451, UW-Madison

"They have computers, and they may have other weapons of mass destruction."
Janet Reno, while US Attorney General

Before we get started…

- Last Time:
  - Discussed concept of Accuracy and Stability of a integration formula used to solve an IVP
    - Stability region for implicit numerical methods much larger
    - No free lunch: implicit numerical methods require solution of an algebraic problem, which might in turn require Newton-Raphson
  - Covered some examples where we used implicit integration
  - Started discussion on the numerical solution of DAEs of Multibody Dynamics using Newmark's method

- Today
  - Finish the discussion on Newmark's method in conjunction with solving the Differential Algebraic Equations of multibody dyanamics
    - Marks the end of the Dynamics Analysis of Mechanical Systems chapter
  - Cover one example to show Newmark's method at work

- Next time: last class
  - Equilibrium Analysis and Inverse Dynamics

Regarding Today's Exam…

- Review starts at 5 pm in room 1152ME
- Exam starts at 7:15 pm in room 1152ME
- Topics that you should be very familiar with
  - How to formulate constraint equations $\Phi(\mathbf{q},t)=0$
  - How to formulate the equations of motion
  - How to specify initial conditions
  - How to compute applied generalized forces acting on a body
  - How to compute the reaction force associated with a kinematic/driving constraint
- Draws on material covered since 10/27, except the Newmark's method discussed on Tu and today
- Bring two sheets of paper with any notes you find useful. You can also bring the two sheets of notes you used for the first midterm exam
- Format of exam: five review questions followed by one problem that covers the dynamics of a 2D mechanism

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 **<u>differential problem is transformed into an algebraic problem</u>**
    - 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:

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

- 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 presentation simple

Do this 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.

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

Newton-Raphson Approach

Then, at each time step do this (iteratively solving for the state of the system at t_{n+1} using Newmark's integration formulas)

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

Update position and velocity at t_{n+1} 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

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

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.

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

Compute the correction vector.

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

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?

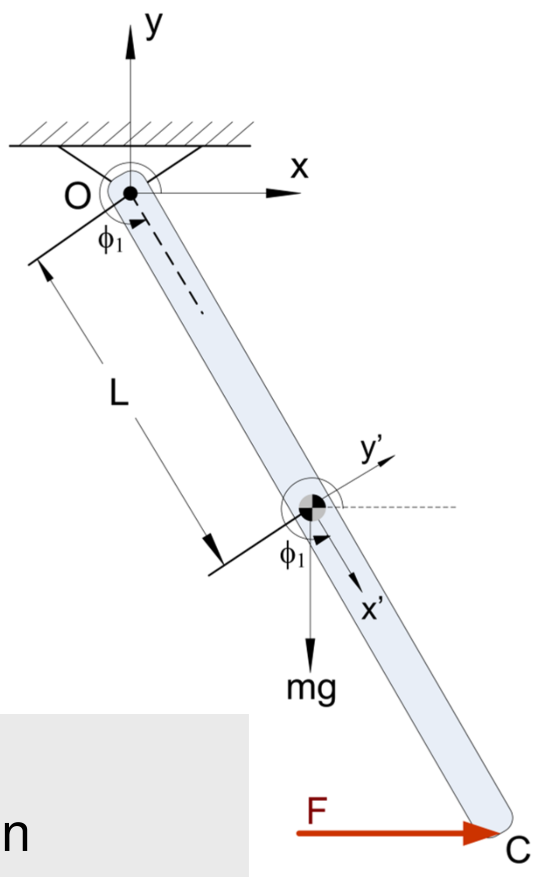
Yes. Need to further refine the value of lambda and acceleration

No.

Store acceleration and lambda at t_{n+1}. Use these accelerations to compute the position and velocity at t_{n+1}. Use lambda to compute the reaction forces. Save this info as you need it to plot results at end of simulation.

Example: Find the time evolution of the pendulum

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

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:

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

- 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

Do this 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.

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

Quasi-Newton Approach

Then, at each time step do this (iteratively solving for the state of the system at t_{n+1} using Newmark's integration formulas)

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

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

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

Approximate the Jacobian of discretized equations of motion with a suitable matrix J-tilde. Evaluate J-tilde only at the very first iteration, then recycle this matrix for subsequent iterations

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

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.

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

Compute the correction vector. Note that the matrix that multiplies the residual doesn't change during the iterative process.

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

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?

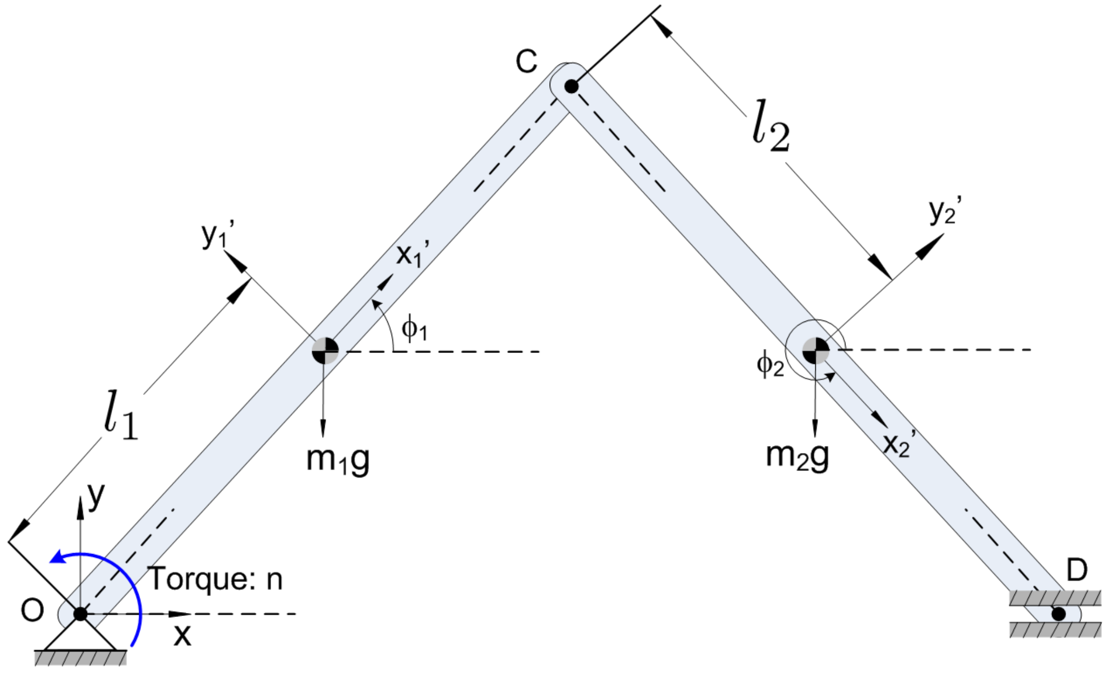
Yes. Need to further refine the value of lambda and acceleration

No.

Store acceleration and lambda at t_{n+1}. Use these accelerations to compute the position and velocity at t_{n+1}. Use lambda to compute the reaction forces. Save this info as you need it to plot results at end of simulation.

Example: Slider Crank Mechanism

- Formulate the EOM
- Evaluate reaction force/torque induced by Abs-Y constraint acting at D


C
l_2
y_2'
y_1'
x_1'
ϕ_1
ϕ_2
l_1
m_1g
m_2g
x_2'
y
Torque: n
O
x
D