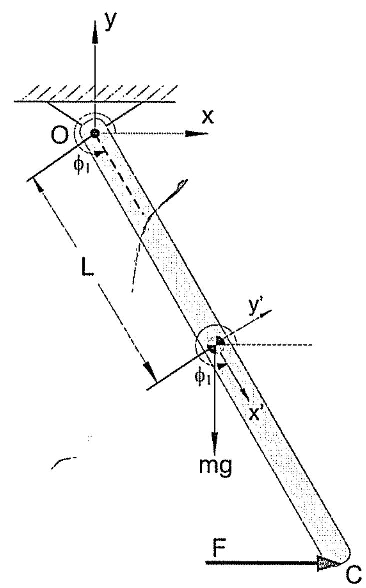
Example Numerical Solution

- **Simple Pendulum:**
  - Mass 20 kg
  - Length L=2m
  - 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] & $\phi_0 = 3\pi/2$
    - $c = 10$ [N/s]
  - ICs: hanging down, starting from rest



$$\begin{cases} M\ddot{q} + \Phi_q^T \lambda = Q^A \\ \Phi(q) = 0 \\ \Phi_q \cdot \dot{q} = \nu \\ \Phi_q \cdot \ddot{q} = \gamma \end{cases}$$

$$q = \begin{bmatrix} x \\ y \\ \phi \end{bmatrix} \qquad M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J' \end{bmatrix} = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & \frac{80}{3} \end{bmatrix}$$

Constraints: Revolute joint between pendulum and ground.

$$r^O = r + A s'^O = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c\phi & -s\phi \\ s\phi & c\phi \end{bmatrix} \cdot \begin{bmatrix} -L \\ 0 \end{bmatrix} = \begin{bmatrix} x - Lc\phi \\ y - Ls\phi \end{bmatrix} = \begin{bmatrix} x - 2c\phi \\ y - 2s\phi \end{bmatrix}$$

$$\Phi = \begin{bmatrix} x - 2c\phi \\ y - 2s\phi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \nu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \gamma = \begin{bmatrix} -2\dot{\phi}^2 c\phi \\ -2\dot{\phi}^2 s\phi \end{bmatrix}$$

Generalized applied forces:

- produced by gravity, Q_1^A
- produced by F (at tip C), Q_2^A
- produced by RSDA (in revolute joint), Q_3^A

$$F = \begin{bmatrix} 30 \sin(2\pi t) \\ 0 \end{bmatrix}$$ (the force always stay horizontal)

$$s'^{C} = \begin{bmatrix} L \\ 0 \end{bmatrix}$$

Apply Eq. 6.2.3 : $Q_2^A = \begin{bmatrix} F \\ s'^{C^T} B^T F \end{bmatrix}$

$$B s'^{C} = \begin{bmatrix} -s\phi & -c\phi \\ c\phi & -s\phi \end{bmatrix} \begin{bmatrix} L \\ 0 \end{bmatrix} = \begin{bmatrix} -Ls\phi \\ Lc\phi \end{bmatrix}$$

$$(B s'^{C})^T \cdot F = \begin{bmatrix} -Ls\phi & Lc\phi \end{bmatrix} \begin{bmatrix} 30 \sin(2\pi t) \\ 0 \end{bmatrix} = -30 L s\phi \cdot \sin(2\pi t)$$

Then $$Q_2^A = \begin{bmatrix} 30 \sin(2\pi t) \\ 0 \\ -30 L s\phi \cdot \sin(2\pi t) \end{bmatrix}$$

For Q_3^A apply Eq. 6.2.22 :

$$Q_3^A = \begin{bmatrix} 0 \\ 0 \\ -k(\phi - \theta_0) - c\dot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -45(\phi - \frac{3\pi}{2}) - 10\dot{\phi} \end{bmatrix}$$

Then,

$$Q^A = Q_1^A + Q_2^A + Q_3^A = \begin{bmatrix} 30 \sin(2\pi t) \\ -mg \\ -30 L s\phi \cdot \sin(2\pi t) - 45(\phi - \frac{3\pi}{2}) - 10\dot{\phi} \end{bmatrix}$$

I will use Newmark's formula to "discretize" the differential equation, and thus transform the differential problem into an algebraic problem:

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

I use the EOM ⊕ position constraints (the latter are scaled):

$$\begin{cases} M\ddot{q}_{n+1} + \Phi_q^T\lambda_{n+1} = Q^A(q_{n+1}, \dot{q}_{n+1}) & \text{(3 equations)} \\ \frac{1}{\beta h^2}\Phi(q_{n+1}) = 0 & \text{(2 equations)} \end{cases}$$

This represents a set of 5 equations in 5 unknowns, that is, the values of $\ddot{x}, \ddot{y}, \ddot{\phi}, \lambda_1, \lambda_2$ at time $n+1$: $\ddot{x}_{n+1}$, $\ddot{y}_{n+1}$, $\ddot{\phi}_{n+1}$, $\lambda_{1,n+1}$, $\lambda_{2,n+1}$. Or, equivalently, $\ddot{q}_{n+1}$ and λ_{n+1}.

Note that if $\ddot{q}_{n+1}$ is known, q_{n+1} and $\dot{q}_{n+1}$ are computed like in Eq.(*) above. In other words, q_{n+1} and $\dot{q}_{n+1}$ are functions of $\ddot{q}_{n+1}$:

$$\begin{cases} q_{n+1} = q_{n+1}(\ddot{q}_{n+1}) \\ \dot{q}_{n+1} = \dot{q}_{n+1}(\ddot{q}_{n+1}) \end{cases} \quad \text{(see Eq.(*))}.$$

Note that

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

the identity matrix. (dimension 3 X 3)

The nonlinear system that I have to solve for $\ddot{q}_{n+1}$ and λ_{n+1} is

$$\Psi(\ddot{q}_{n+1}, \lambda_{n+1}) = \begin{bmatrix} M\ddot{q}_{n+1} + \Phi_q^T \lambda_{n+1} - Q^A(q_{n+1}, \dot{q}_{n+1}) \\ \frac{1}{\beta h^2} \Phi(q_{n+1}) \end{bmatrix} = \begin{bmatrix} 0_{3\times1} \\ 0_{2\times1} \end{bmatrix}$$

We then need to solve the nonlinear system:

$$\Psi(\ddot{q}_{n+1}, \lambda_{n+1}) = 0$$

to find $\ddot{q}_{n+1}$ and λ_{n+1}. For this, we'll use Newton's method

$$\begin{bmatrix} \ddot{q}_{n+1} \\ \lambda_{n+1} \end{bmatrix}^{new} = \begin{bmatrix} \ddot{q}_{n+1} \\ \lambda_{n+1} \end{bmatrix}^{old} - J^{-1} \cdot \Psi(\ddot{q}_{n+1}^{old}, \lambda_{n+1}^{old})$$

Here, J is the Jacobian required be Newton's method:

$$J = \left[\frac{\partial \Psi}{\partial \ddot{q}_{n+1}} \;\Big|\; \frac{\partial \Psi}{\partial \lambda_{n+1}} \right],$$ and therefore,

$$J = \left[\begin{array}{c|c} M + \frac{\partial(\Phi_q^T \lambda_{n+1})}{\partial q_{n+1}} \cdot \frac{\partial q_{n+1}}{\partial \ddot{q}_{n+1}} - \frac{\partial Q^A}{\partial q_{n+1}} \cdot \frac{\partial q_{n+1}}{\partial \ddot{q}_{n+1}} - \frac{\partial Q^A}{\partial \dot{q}_{n+1}} \cdot \frac{\partial \dot{q}_{n+1}}{\partial \ddot{q}_{n+1}} & \Phi_q^T \\ \frac{1}{\beta h^2} \frac{\partial \Phi}{\partial q_{n+1}} \cdot \frac{\partial q_{n+1}}{\partial \ddot{q}_{n+1}} & 0 \end{array}\right]$$

Note: I will drop the superscript "n+1" to keep the notation simpler. keep in mind though that all these quantities are evaluated at t_{n+1}, and a superscript is implied.

Therefore,

$$J = \left[\begin{array}{c|c} M + \beta h^2\left(\dfrac{\partial(\phi_q^T \lambda)}{\partial q} - \dfrac{\partial Q^A}{\partial q}\right) - \gamma h \dfrac{\partial Q^A}{\partial \dot{q}} & \phi_q^T \\ \Phi_q & 0 \end{array}\right]$$

Next, we need to compute the partial derivatives of interest.
For our problem,

$$\phi_p = \begin{bmatrix} 1 & 0 & 2s\phi \\ 0 & 1 & -2c\phi \end{bmatrix}$$

Then,

$$\phi_p^T \lambda = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2s\phi & -2c\phi \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ 2\lambda_1 s\phi - 2\lambda_2 c\phi \end{bmatrix}$$

$\Rightarrow (\phi_p^T \lambda)_q =$ (partial derivative of the reaction forces wrt q)

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 2\lambda_1 c\phi + 2\lambda_2 s\phi \end{bmatrix}$$

$$\frac{\partial Q^A}{\partial p} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -30L \cdot c\phi \cdot \sin(2\pi t) - 45 \end{bmatrix}$$

$$\frac{\partial Q^A}{\partial \dot{q}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -10 \end{bmatrix}$$

I will introduce the following notation:

$$w = \beta h^2\left[(2\lambda_1 c\phi + 2\lambda_2 s\phi) + 30 L \cdot c\phi \cdot \sin(2\pi t) + 45\right] + \gamma h \cdot 10$$

Then, the Jacobian J will assume the expression:

$$J = \left[\begin{array}{ccc|cc} 20 & 0 & 0 & 1 & 0 \\ 0 & 20 & 0 & 0 & 1 \\ 0 & 0 & \frac{80}{3} + w & 2s\phi & -2c\phi \\ \hline 1 & 0 & 2s\phi & 0 & 0 \\ 0 & 1 & -2c\phi & 0 & 0 \end{array}\right]$$

With this, I have all the information needed to implement the MATLAB code that determines the time evolution of my system.

Note that at time $t=0$, the acceleration $\ddot{q}_0$ and Lagrange multiplier λ_0 are obtained as the solution of the linear system

$$\begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix} \begin{bmatrix} \ddot{q}_0 \\ \lambda_0 \end{bmatrix} = \begin{bmatrix} Q^A_0 \\ \gamma_0 \end{bmatrix}$$

That is,

$$\left[\begin{array}{ccc|cc} 20 & 0 & 0 & 1 & 0 \\ 0 & 20 & 0 & 0 & 1 \\ 0 & 0 & \frac{80}{3} & 2s\phi & -2c\phi \\ \hline 1 & 0 & 2s\phi & 0 & 0 \\ 0 & 1 & -2c\phi & 0 & 0 \end{array}\right] \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\phi} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 30 \sin(2\pi t) \\ -mg \\ -30 \cdot L s\phi \cdot \sin(2\pi t) - 45(\phi - \frac{3\pi}{2}) - 10\dot{\phi} \\ -2\dot{\phi}^2 c\phi \\ -2\dot{\phi}^2 s\phi \end{bmatrix}$$

The pseudo code for computing the time evolution of the pendulum is provided on the next page.

```
get initial conditions q0 & q̇0
compute acceleration and Lagrange multipliers based on
eq. on previous page
for t=0 to Tend

        q̈1 = q̈0     (initial guess for new acceleration)
        λ1 = λ0     (initial guess for new Lagrange multipliers)

        do while normCorrection > correctionEpsilon
              Use Newmark's formulas to compute q1 and q̇1
              Evaluate Jacobian J
              Evaluate Ψ (also called "residual")
              compute correction = J^-1 · Ψ
              Update q̈ & λ : [q̈1; λ1] = [q̈1; λ1] - correction
              normCorrection = norm(correction)
        end "do while"
end "for".
```