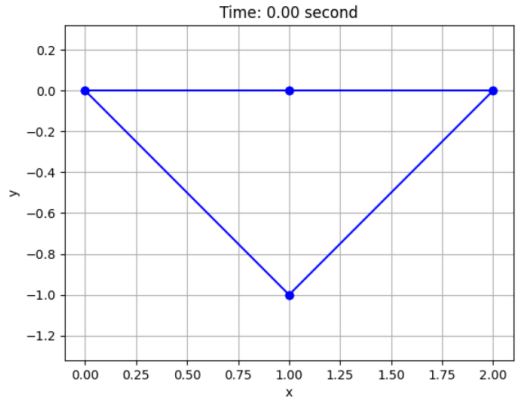
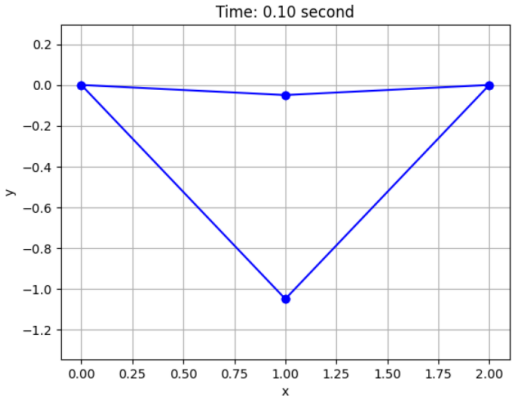
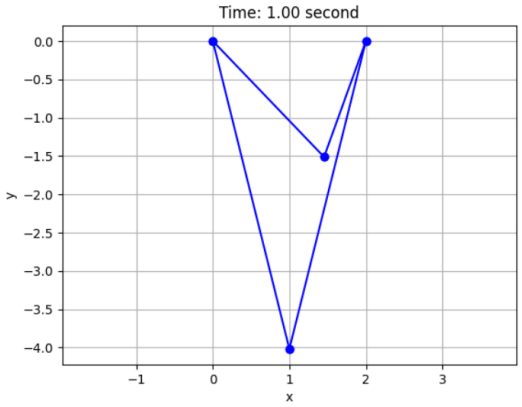
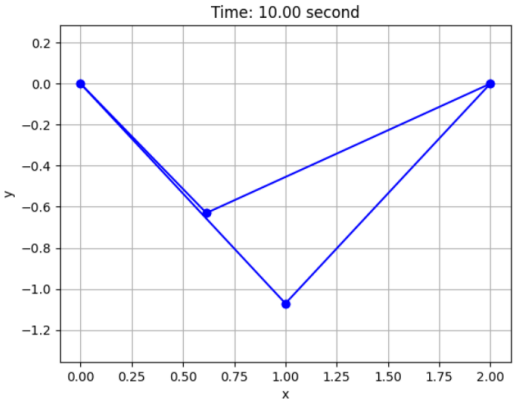
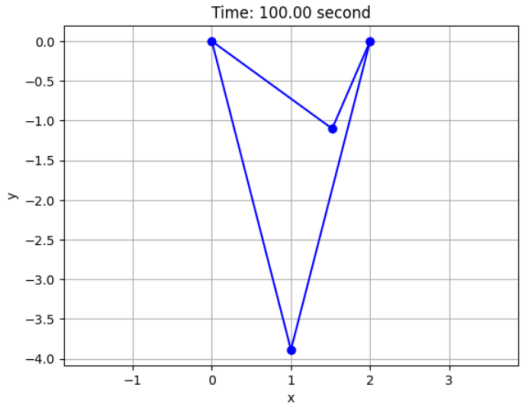
Simulation of (b) by implicit method with $\Delta t = 0.001$

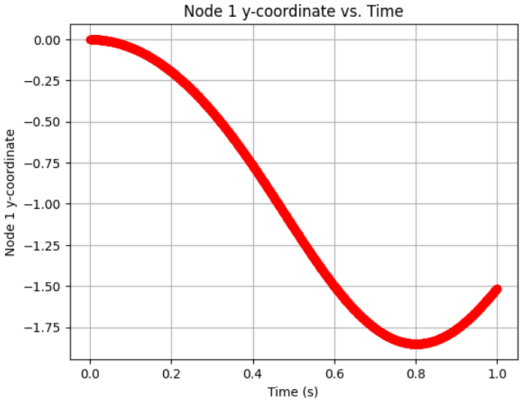


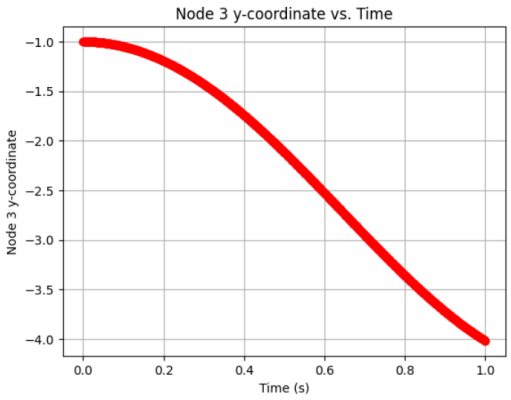


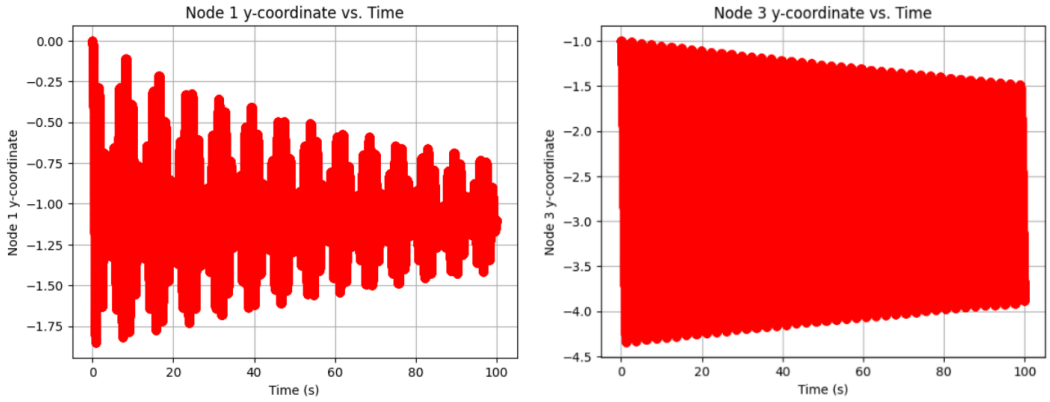












1.

Known : current position $\underline{x}_{old}$, current velocity $\underline{u}_{old}$, nodes.txt, spring.txt

unknown : next position $\underline{x}_{new}$, next velocity $\underline{u}_{new}$

- nodes.txt : position of each node

$$\begin{bmatrix} x_1, y_1 \\ x_2, y_2 \\ \vdots \\ x_n, y_n \end{bmatrix} \begin{matrix} \rightarrow \text{node 1} \\ \\ \\ \rightarrow \text{node n} \end{matrix}$$

spring.txt

$$\begin{bmatrix} a_1 & b_1 & k_1 \\ \vdots & \vdots & \vdots \\ a_m & b_m & k_m \end{bmatrix} \rightarrow \text{spring m}$$

node a_1 and node b_1 are connected with spring constant k_n

$$\Rightarrow \underline{x}_0 = \begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_n \\ y_n \end{bmatrix}$$

index matrix $Mind = \begin{bmatrix} 2a_1 & 2a_1+1 & 2b_1 & 2b_1+1 \\ \vdots & \vdots & \vdots & \vdots \\ 2a_m & 2a_m+1 & 2b_m & 2b_m+1 \end{bmatrix}$

stiffness matrix $M_s = [k_1 \quad k_2 \quad \cdots \quad k_m]^T$

mass $m = [m_1 \quad m_2 \quad \cdots \quad m_n]^T$

reference spring lengths $l = \begin{bmatrix} \sqrt{(x_0[Mind_{1,1}] - x_0[Mind_{1,3}])^2 + (x_0[Mind_{1,2}] - x_0[Mind_{1,4}])^2} \\ \vdots \\ \sqrt{(x_0[Mind_{m,1}] - x_0[Mind_{m,3}])^2 + (\cdots)^2} \end{bmatrix}$

$Mind_{i,j} = Mind[i,j]$ (row, col)

① Get spring force

$\underline{f}_s = \text{function } \nabla E_s(x_i, y_i, x_j, y_j, l_k, k)$

$E_s = \frac{1}{2} k l_k \varepsilon^2$, $\varepsilon = \frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{l_k} - 1$

$f_s = (k l_k \varepsilon)(\nabla \varepsilon)$

② Get spring Jacobian

$\underline{J}_s = \text{function } \nabla^2 E_s(x_i, y_i, x_j, y_j, l_k, k)$

2) Get total force and Jacobian

$$\underline{\underline{M}}\ddot{\underline{x}} + \frac{\partial E_s}{\partial \underline{x}} - \underline{f}_{ext} = 0$$

$$\Rightarrow \underline{f}_{inertia} + \underline{f}_{spring} - \underline{f}_{ext} = 0$$

$[\underline{f}, \underline{\underline{J}}] = \text{getForceJacobian}(\underline{x}_{new}, \underline{x}_{old}, \underline{v}_{old}, M_s, M_{ind}, m, \Delta t, l)$

$$\underline{f}_{inertia} = \frac{m}{\Delta t}\left[\frac{\underline{x}_{new} - \underline{x}_{old}}{\Delta t} - \underline{v}_{old}\right]$$

$$\underline{\underline{J}}_{inertia} = \frac{\text{diag}(m)}{\Delta t^2}$$

$\underline{f}_{spring} = \text{zeros}(\text{len}(\underline{x}))$

$\underline{\underline{J}}_{spring} = \text{zeros}(\text{len}(\underline{x}), \text{len}(\underline{x}))$

for i in range(len(M_s)):

ind = $M_{ind}[i, :]$

$x_i = \underline{x}_{new}[ind[0]]$, $x_j = \underline{x}_{new}[ind[2]]$

$y_i = \underline{x}_{new}[ind[1]]$, $y_j = \underline{x}_{new}[ind[3]]$

$\underline{f}_{spring}[ind] \mathrel{+}= \nabla E_s(x_i, y_i, x_j, y_j, l[i], M_s[i])$

$\underline{\underline{J}}_{spring}[ind, ind] \mathrel{+}= \nabla^2 E_s(x_i, y_i, x_j, y_j, l[i], M_s[i])$.

$\underline{f}_{ext} = [0 \ -9.81 \ 0 \ -9.81 \ 0 \ -9.81 \ \cdots \ 0 \ -9.81]^T$

$\underline{\underline{J}}_{ext} = \text{zeros}(\text{len}(x), \text{len}(x))$

$\underline{f} = \underline{f}_{inertia} + \underline{f}_{spring} - \underline{f}_{ext}$

$\underline{\underline{J}} = \underline{\underline{J}}_{inertia} + \underline{\underline{J}}_{spring} - \underline{\underline{J}}_{ext}$

④ For all free nodes, we want to find x_{new} such that

$f_{inertia} + f_{sping} - f_{ext} = 0$ is satisfied.

$x_{new}, u_{new} =$ Integrator (x_{old}, u_{old}, DOF_{free}, Ms, Mind, m, Δt)

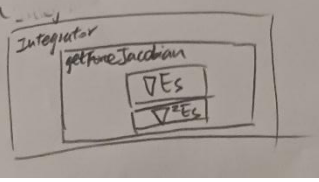
↑ index of free degree of freedom (matches x)

```
x_new = x_old
eps = 10^-6   # tolerance
err = 1
while err > eps :
    f, J = getForceJacobian(x_new, x_old, u_old, Ms, Mind, m, Δt, l)
    f_free = f[DOF_free]
    J_free = J[DOF_free]
    Δx_free = J_free \ f_free
    x_new[DOF_free] -= Δx_free
    err = ||f_free||_2
u_new = (x_new - x_old) / Δt
```

⑤ Main function to solve x at each time.

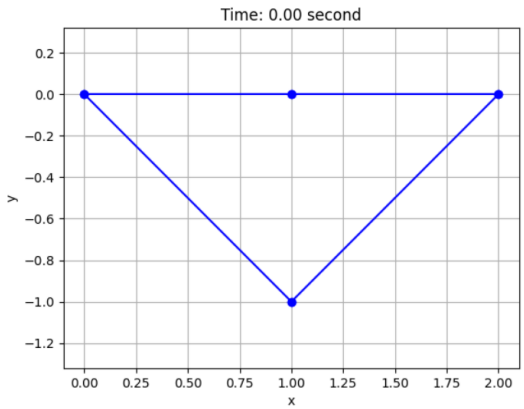
```
t = 0 : Δt : tf , x, u = zeros(len(x_0), len(t))
for i in range((len(t)-1)) :
    x[:,i+1], u[:,i+1] = Integrator(x[:,i], u[:,i], DOF_free, Ms, Mind, m, Δt)
```

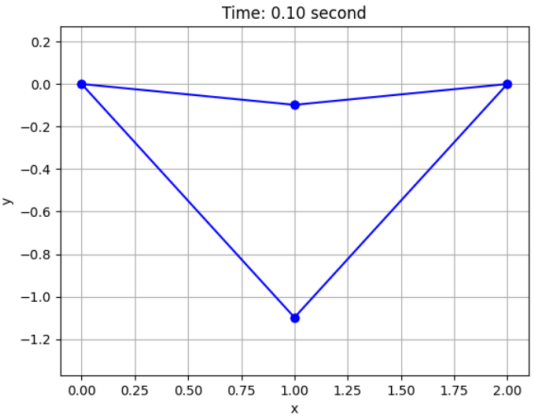
Block diagram

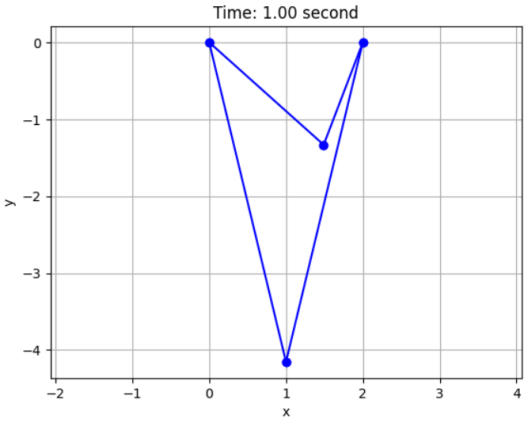


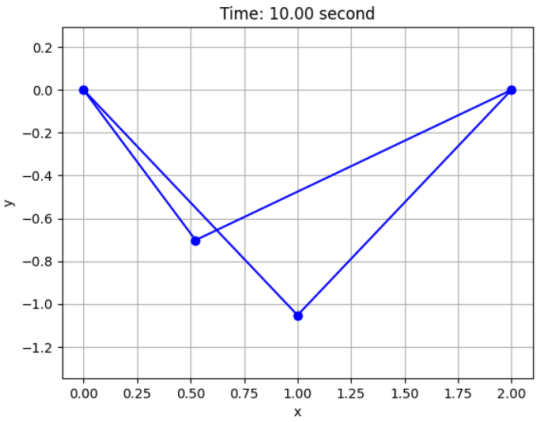
2. I choose $\Delta t = 0.001$ so that the damping effect of implicit method can be minimized but not taking too much time on calculation.

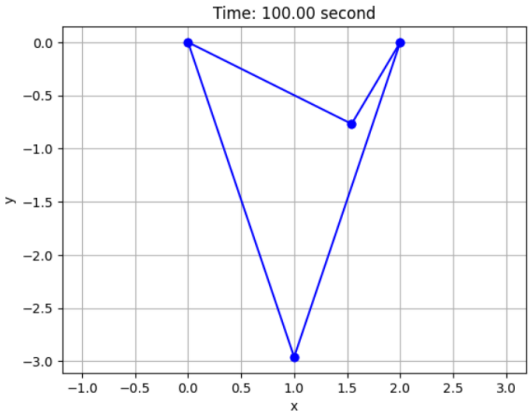
3. Explicit method is preferable for this spring network since there not only is no damping effect but don't diverge at a long sampling time $\Delta t = 0.1$

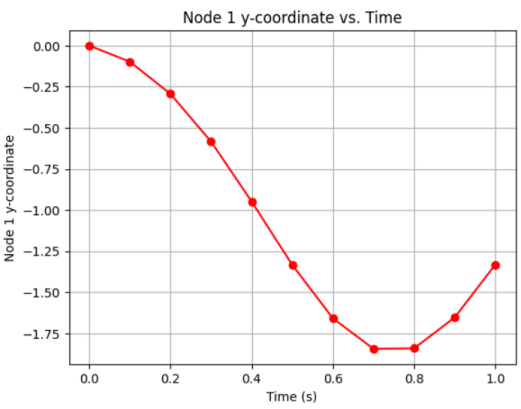

Time: 0.00 second
x
y

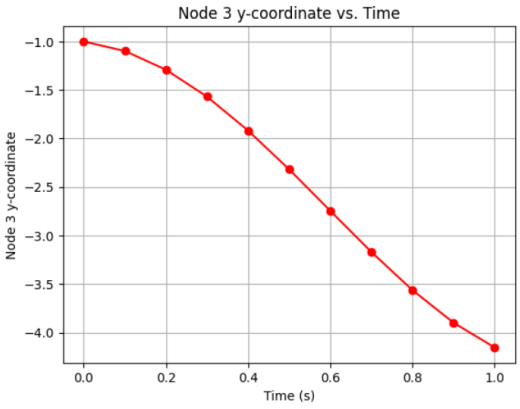

Time: 0.10 second
x
y

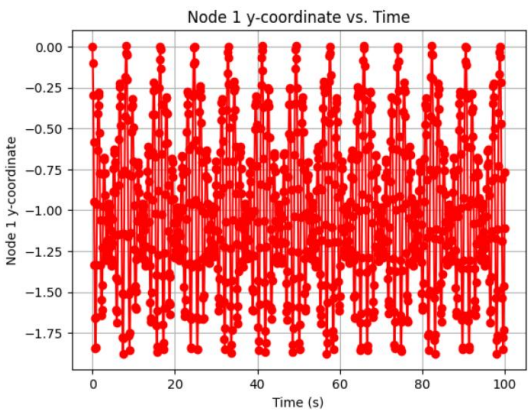

Time: 1.00 second
x
y

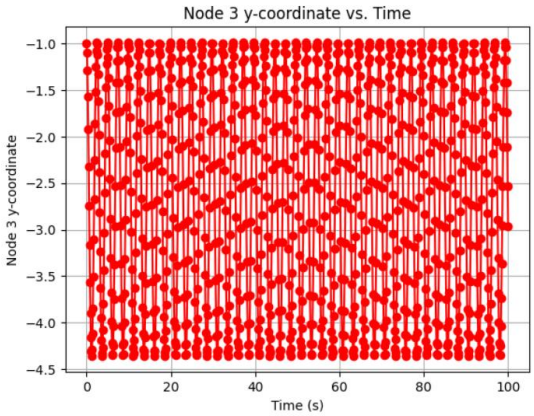

Time: 10.00 second
x
y


Time: 100.00 second
x
y


Node 1 y-coordinate vs. Time
Time (s)
Node 1 y-coordinate


Node 3 y-coordinate vs. Time
Time (s)
Node 3 y-coordinate

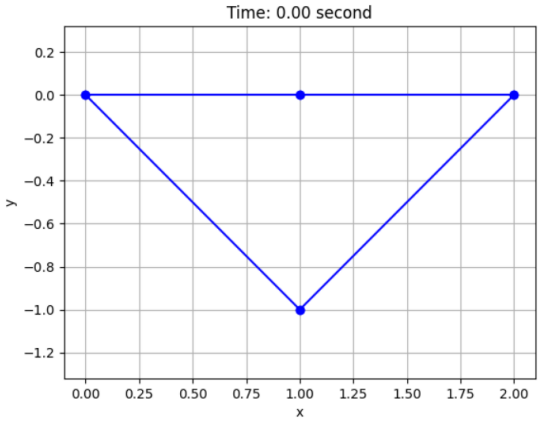


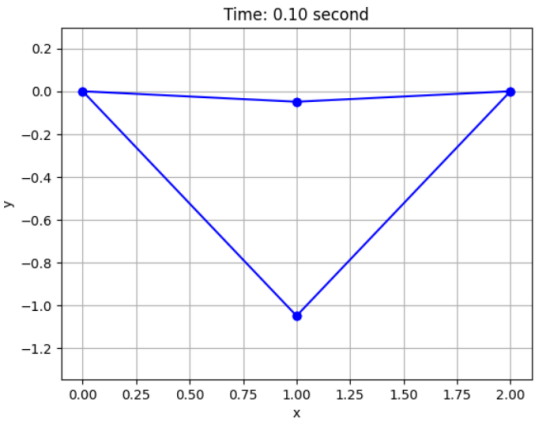


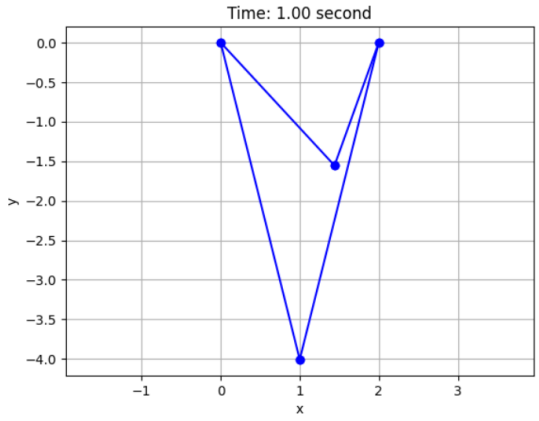
4. In implicit Euler method, the new position is only related to the old acceleration. If the acceleration of system is increasing, the simulated position won't reach the real position. And, if the acceleration of system is decreasing, the simulated position will go over the real position. This will work like a damper, which will prevent the system from approaching the state without any resistance.

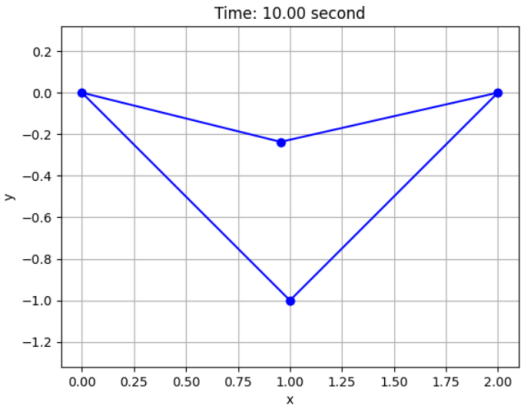
On the other hand, in Newmark-β, the new position is also related to the new acceleration. In addition, the most of time, β equals 0.25 such that the new position is related to the average of new and old acceleration. Therefore, as long as the sampling time is small enough, Newmark-β method can get a good guess of acceleration between current time and next time interval, and a proper next position can be calculated.

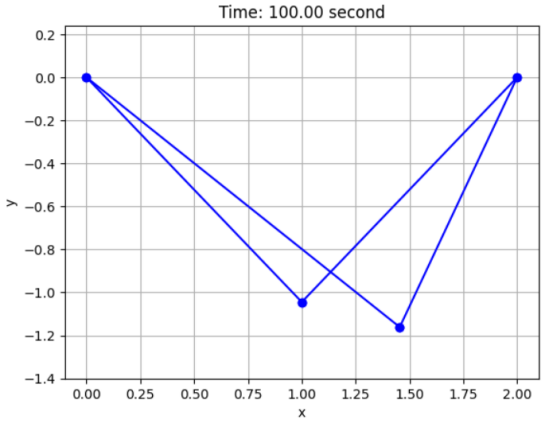
5. Simulation of (b) by Newmark-$\beta(\beta = 0.25)$ method with $\Delta t = 0.1$

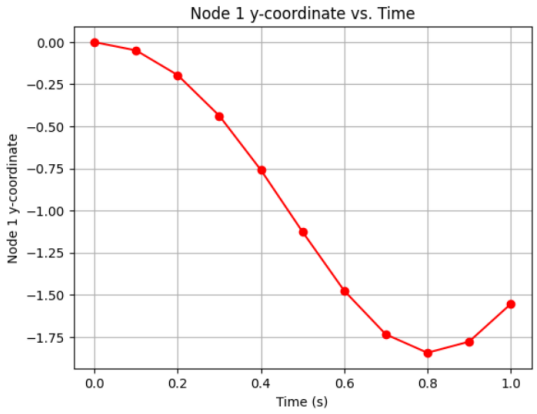


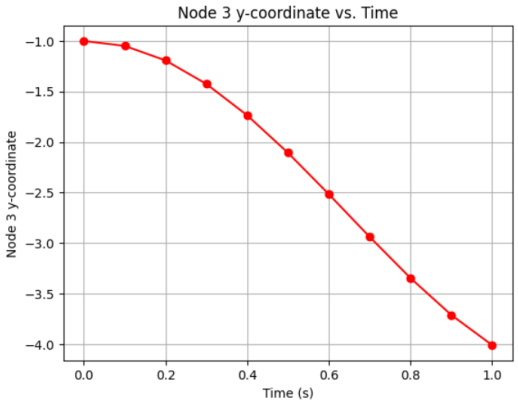


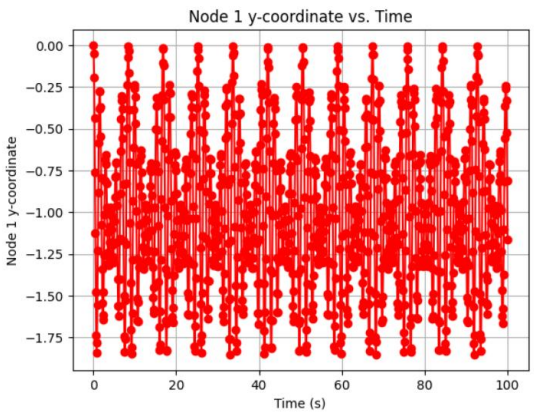

Time: 1.00 second
x
y

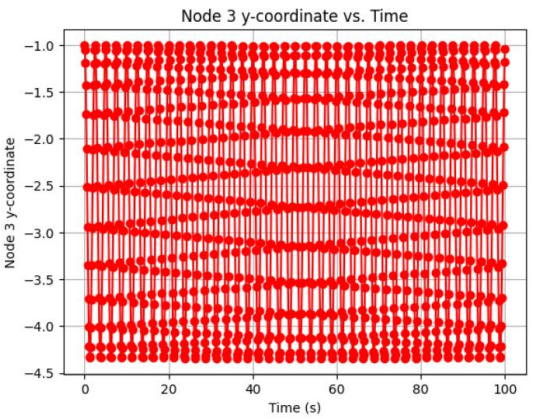

Time: 10.00 second
x
y


Time: 100.00 second
x
y


Node 1 y-coordinate vs. Time
Time (s)
Node 1 y-coordinate


Node 3 y-coordinate vs. Time
Time (s)
Node 3 y-coordinate


Node 1 y-coordinate vs. Time
Time (s)
Node 1 y-coordinate


Node 3 y-coordinate vs. Time
Time (s)
Node 3 y-coordinate