Homework 1

Ting-Hao Liu

I. Implicit Method Simulation Results

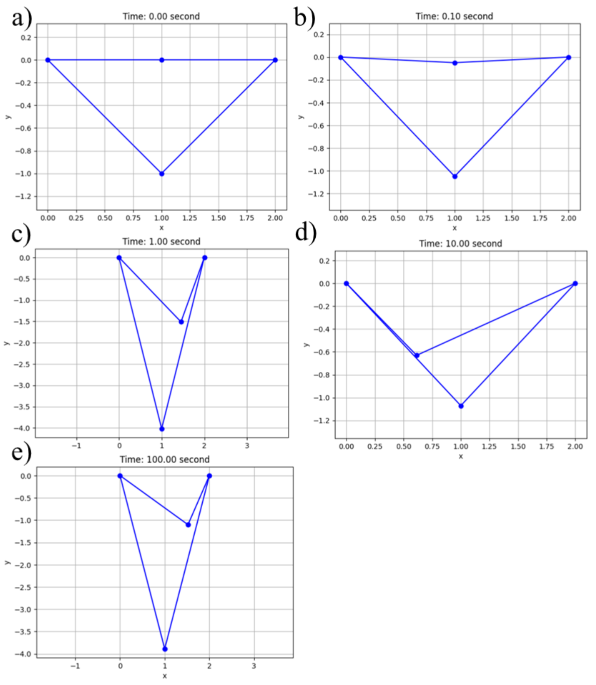

Figure 1: Shown is the spring network at $t = \{0, 0.01, 1, 10, 100\}\ s$ by implicit method with $\Delta t = 0.001$.

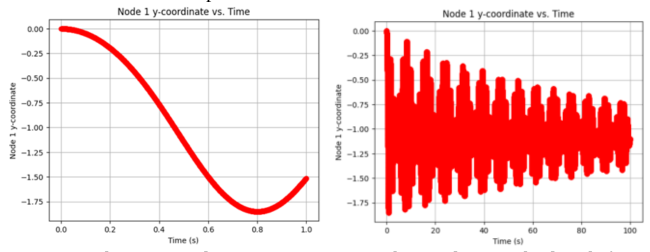

Figure 2: Shown is node 1 position in y-axis by implicit method with $\Delta t = 0.001$.

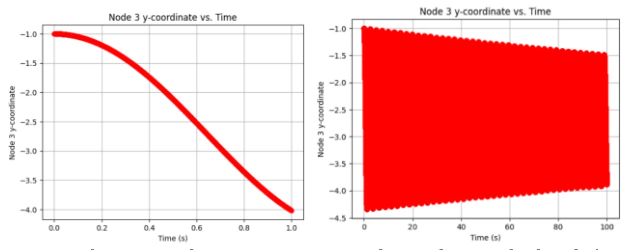

Figure 3: Shown is node 3 position in y-axis by implicit method with $\Delta t = 0.001$.

II. Method

The initial position, connection of all nodes and stiffness of all springs are given. In addition, for each time step, current position vector (x_{old}) and velocity vector (u_{old}) are knowns. Our goal is to find next position (x_{new}) and velocity (u_{new}).

The "nodes.txt" gives the node position in each row. If there are N nodes, the text file can be written as

$$\begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_N & y_N \end{bmatrix} \tag{1}$$

We can further fatten those position into a position vector (x).

$$x = [x_1 \quad y_1 \quad \dots \quad x_N \quad y_N]^T \tag{2}$$

And, the velocity vector becomes

$$u = [\dot{x}_1 \quad \dot{y}_1 \quad \dots \quad \dot{x}_N \quad \dot{y}_N]^T \tag{3}$$

The "spring.txt" gives the sprint connect of node a and node b with stiffness k. If there are M springs, the text file can be written as

$$\begin{bmatrix} a_1 & b_1 & k_1 \\ \vdots & \vdots & \vdots \\ a_M & b_M & k_M \end{bmatrix} \tag{4}$$

We can further write those connection into the index of elements in position vector (x) by an index matrix (M_{ind})

$$M_{ind} = \begin{bmatrix} 2a_1 & 2a_1 + 1 & 2b_1 & 2b_1 + 1 \\ \vdots & \vdots & \vdots & \vdots \\ 2a_M & 2a_M + 1 & 2b_M & 2b_M + 1 \end{bmatrix} \tag{5}$$

And, the stiffness matrix (M_s) can represent the stiffness of each spring.

$$M_s = [k_1 \quad k_2 \quad \dots \quad k_{M-1} \quad k_M]^T \tag{6}$$

The mass of each node is also given and be written into a mass vector (m)

$$m = [m_1 \quad m_2 \quad \dots \quad m_{N-1} \quad m_N]^T \tag{7}$$

Finally, we can calculate the initial spring length of spring k by initial position vector (x_0) as

$$l_k = \sqrt{\left(x_0[M_{ind}[k,1]] - x_0[M_{ind}[k,3]]\right)^2 + \left(x_0[M_{ind}[k,2]] - x_0[M_{ind}[k,4]]\right)^2} \tag{8}$$

And, the initial length vector (l) is

$$l = [l_1 \quad l_2 \quad \dots \quad l_{M-1} \quad l_M]^T \tag{9}$$

A. *"Get Spring Force" and "Get Spring Jacobian" Functions*

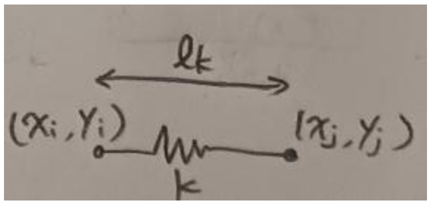

Figure 4: Shown is the spring connecting 2 nodes with stiffness k N/m and initial length l_k.

The spring energy of the spring in Fig. 4 can be written as

$$E_s = \frac{1}{2} k_k l_k \varepsilon^2 \tag{10}$$

$$\varepsilon = \frac{\sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}}{l_k} \tag{11}$$

Spring force:

$$f_s = k_k l_k \varepsilon (\nabla \varepsilon) \tag{12}$$

The "GetSpringForce" and "GetSpringJacobian" function can be expressed as

$$f_s = function\ \nabla E_s\left(x_i, y_i, x_j, y_j, l_k, k_k\right) \tag{13}$$

$$J_s = function\ \nabla^2 E_s\left(x_i, y_i, x_j, y_j, l_k, k_k\right) \tag{14}$$

B. *"Get Total Force and Jacobian" function*

Since we have to solve ODE in Eq. 15, the total forces and Jacobian should be calculated.

$$diag(m)\ddot{x} + \frac{\partial E_s}{\partial x} - f_{ext} = 0 \tag{15}$$

Eq. 15 can also be written as

$$f_{inertia} + f_{sping} - f_{ext} = 0 \tag{16}$$

Where f_{ext} is external force, which causes by gravity in this assignment. The pseudocode can be expressed as

```
$[f, J] = GetForceJacobian(x_{new}, x_{old}, u_{old}, M_s, M_{ind}, m, l, \Delta t)$:
  $f_{inertia} = \frac{m}{\Delta t}\left(\frac{x_{new} - x_{old}}{\Delta t} - u_{old}\right)$
  $J_{inertia} = \frac{diag(m)}{(\Delta t)^2}$
  $f_{sping} = zeros(len(x))$
  $J_{sping} = zeros(len(x), len(x))$
  $for\ k = 1{:}M$
    $ind = M_{ind}[k, :]$
    $x_i = x_{new}[ind[1]], x_j = x_{new}[ind[3]]$
    $y_i = x_{new}[ind[2]], y_j = x_{new}[ind[4]]$
    $f_{sping}[ind] += \nabla E_s(x_i, y_i, x_j, y_j, l_k, k_k)$
    $J_{sping}[ind, ind] += \nabla^2 E_s(x_i, y_i, x_j, y_j, l_k, k_k)$
  $f_{ext} = [0 \quad -9.81 \quad \ldots \quad 0 \quad -9.81]^T$
  $J_{ext} = zeros(len(x), len(x))$
  $f = f_{inertia} + f_{sping} - f_{ext}$
  $J = J_{inertia} + J_{sping} - J_{ext}$
```

C. *"Integrator" Function*

For each free nodes, we want to find the new position and velocity such that Eq. 16 is satisfied.

We can design a "Integrator" function by know previous position, velocity and the index of free degree of freedom (I_{free}) matching position and velocity vectors:

```
$x_{new}, u_{new} = Integrator(x_{old}, u_{old}, I_{free}, M_s, M_{ind}, m, \Delta t)$:
  $x_{new} = x_{old}$
  $eps = 10^{-6}$
  $err = 1$
  $while\ err > eps$:
    $[f, J] = GetForceJacobian(x_{new}, x_{old}, u_{old}, M_s, M_{ind}, m, l, \Delta t)$
    $f_{free} = f[I_{free}]$
    $J_{free} = J[I_{free}, I_{free}]$
    $\Delta x_{free} = J_{free} \backslash f_{free}$
    $x_{new}[I_{free}] -= \Delta x_{free}$
    $err = \|f_{free}\|_2$
  $u_{new} = \frac{x_{new} - x_{old}}{\Delta t}$
```

In this assignment, the index of free degree of freedom (I_{free}) is

$$I_{free} = [3 \quad 4 \quad 7 \quad 8]^T \tag{17}$$

D. *"Main" Function*

For each iteration, we'll keep sending current position, velocity vectors into "Integrator" to update new state. The pseudocode can be expressed as

```
$t = 0{:}\Delta t{:}t_f$
$x_{all}, u_{all} = zeros(len(x), len(t))$
$x_{all}[:, 1] = x_0$
$For\ k = 2{:}len(t)$
  $x_{all}[:, k], u_{all}[:, k] = Integrator(x_{all}[:, k-1], u_{all}[:, k-1], I_{free}, M_s, M_{ind}, m, \Delta t)$
```

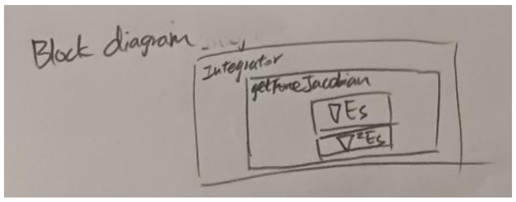

Figure 5: Block diagram

Fig. 5 indicates the main function keeps calling "Integrator" to update new position and velocity. And, to solve force equation Eq. 16, the "Integrator" keeps calling "GetForceJacobian," which finds the total force and Jacobian of the system. Finally, the "GetForceJacobian" will call "GetSpringForce" and "GetSpringJacobian."

III. DISCUSSION

A. *Sampling Time*

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

B. *Explicit Method*

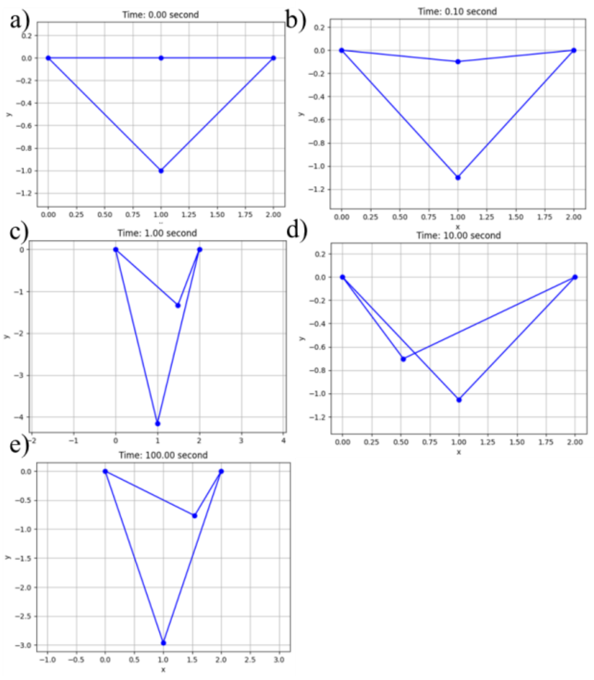

Figure 6: Shown is the spring network at $t = \{0,0.01,1,10,100\}\ s$ by explicit method with $\Delta t = 0.1$.

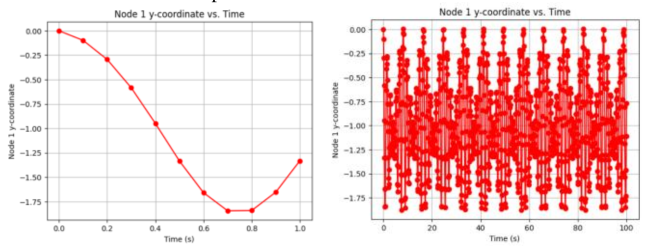

Figure 7: Shown is node 1 position in y-axis by explicit method with $\Delta t = 0.1$.

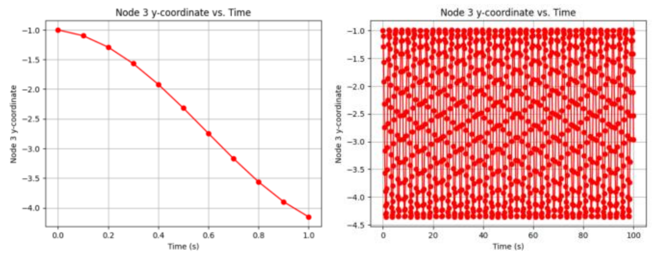

Figure 8: Shown is node 3 position in y-axis by explicit method with $\Delta t = 0.1$.

Theoretically, implicit method has a better accuracy. However, for this spring network, explicit method is preferable since there not only is no damping effect but don't diverge at a long sampling time $\Delta t = 0.1$.

C. *Newmark-β Method*

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.

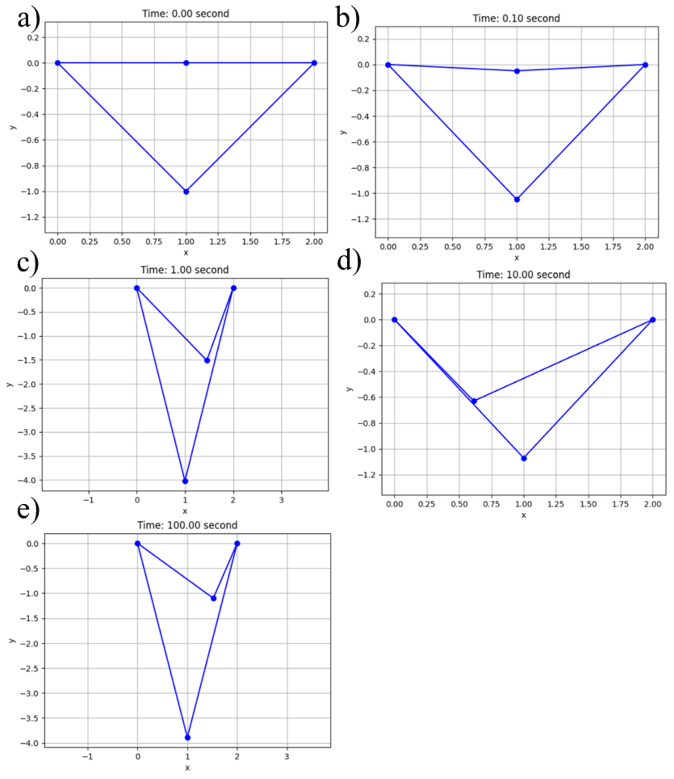

Figure 9: Shown is the spring network at $t = \{0,0.01,1,10,100\}\ s$ by Newmark-β($\beta = 0.25$, $\gamma = 0.5$) with $\Delta t = 0.1$

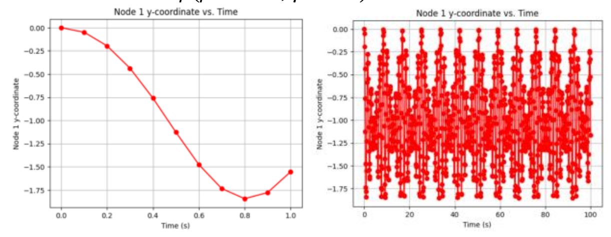

Figure 10: Shown is node 1 position in y-axis by Newmark-β($\beta = 0.25$, $\gamma = 0.5$) with $\Delta t = 0.1$.

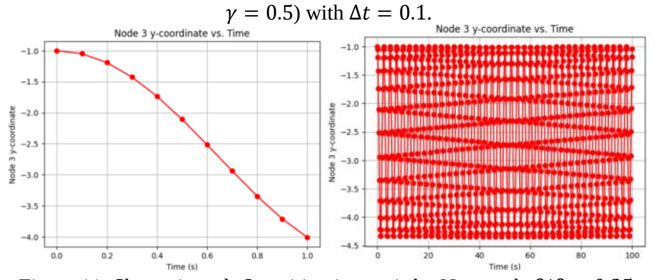

Figure 11: Shown is node 3 position in y-axis by Newmark-β($\beta = 0.25$, $\gamma = 0.5$) with $\Delta t = 0.1$.