MAE 263F HW1

Ursa Z., University of California Los Angeles

***Abstract*— This project investigates the 2D dynamics spring-mass networks using numerical time integration methods. The systems have 4 linear springs with different stiffness connecting 4 nodes, where node 0 and node 2 are fixed boundary conditions. Each node is assigned the mass and applied gravitational loading. By using the implicit Euler method, we derive the governing equations of motions from the total spring energy. The simulation tracks the nodal displacements from time = 0s to 100s and demonstrates the visualizations of structural deformation and y-coordinates evolution of the free nodes. Comparisons with the explicit Euler formulation confirm that explicit methods become unstable for large time steps, while implicit methods remain robust.**

I. Introduction

The objective is to implement and compare two time-integration schemes: the implicit Euler and the explicit Euler methods. The implicit Euler scheme is unconditionally stable but introduces numerical damping, while the explicit Euler method is conditionally stable and can diverge for stiff systems unless the time step is sufficiently small. The simulation is performed in Python, where the forces and Jacobians are derived from the gradient and Hessian of the stretching energy of each spring. The results include the temporal evolution of the nodal displacements, network deformation shapes, and stability analysis of both methods.

II. Pseudocode and Code Structure

A. *Pseudocode of the simulator*

BEGIN SPRING_NETWORK_SIMULATION

1. Read nodal coordinates and spring connections from input files

(nodes.txt, springs.txt). Initialize mass vector and fixed/free DOFs.

2. For each spring, compute its rest length l_k.

3. Define functions:

gradEs(): Compute gradient of stretching energy (spring forces)

hessEs(): Compute Hessian matrix (tangent stiffness)

getFexternal(): Compute gravitational force vector

getForceJacobian(): Assemble total internal/external forces and Jacobian

implicitEulerStep(): Newton – Raphson loop for implicit update

explicitEulerStep(): Direct explicit update of x and v

plot_network(): Plot current spring configuration at time t

4. Initialize displacements x_old and velocities u_old at t = 0.

5. Time integration loop:

For t = 0 → T_max with time step Δ t:

a. Compute forces on each node.

b. If implicit method:

Solve nonlinear residual f = 0 via Newton–Raphson.

Else if explicit method:

Update x, u directly using current acceleration.

c. Apply boundary conditions to fixed nodes.

d. Store y–coordinates of free nodes (1 and 3).

e. Plot network at specified times.

6. Plot y–coordinate histories of free nodes vs time.

END

B. *Code Interaction*

nodes.txt, springs.txt

↓

compute l_k

↓

gradEs(), hessEs()

↓

getForceJacobian() → builds f, J

↓

myInt() [implicit] OR explicitEulerStep()

↓

apply boundary conditions

↓

record results and plot_network()

III. MATH

Newton's Second Law

$$m_i\,\mathbf{a}_i = \mathbf{F}_{i,\text{spring}} + \mathbf{F}_{i,\text{external}}$$

Spring Energy

Each spring is modeled as a linear elastic element that resists deformation from its rest length.

$$\mathbf{F}_k = -\frac{\partial E_k^s}{\partial \mathbf{x}}$$

For the x– and y–components of the force on node i and j, the expressions are

$$F_{x_i} = -\frac{1}{2}\,k\,l_k\left(1-\frac{L}{l_k}\right)\frac{(x_j - x_i)}{L},$$
$$F_{y_i} = -\frac{1}{2}\,k\,l_k\left(1-\frac{L}{l_k}\right)\frac{(y_j - y_i)}{L},$$
$$F_{x_j} = -F_{x_i},$$
$$F_{y_j} = -F_{y_i}.$$

Spring Stiffness Matrix

The local tangent stiffness of each spring is obtained by taking the second derivative of the spring energy with respect to node coordinates. This results in a 4×4 matrix that relates infinitesimal displacements of the two connected nodes to the change in internal force.

$$\mathbf{K}_k = \frac{\partial^2 E_k^s}{\partial \mathbf{x}^2}$$

External Forces

Each node is subjected to a gravitational force acting in the negative y–direction.

The external force vector is defined as $\mathbf{F}_{\text{external}} = m_i\,\mathbf{g}$

Implicit Euler Time Integration

The implicit Euler scheme is used to integrate the equations of motion in time.

Velocity and position updates are defined as

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t\,\mathbf{a}_{n+1},$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t\,\mathbf{v}_{n+1}.$$

Since the acceleration an+1 depends on xn+1, the above equations form a nonlinear system that is solved iteratively using the Newton–Raphson method: $\mathbf{x}_{n+1}^{(k+1)} = \mathbf{x}_{n+1}^{(k)} - \mathbf{J}^{-1}\mathbf{f}$

The iteration terminates when the norm of the residual vector satisfies $\|\mathbf{f}\| < \varepsilon$ where ε is a small convergence tolerance.

Explicit Euler Time Integration

The explicit Euler method updates motion using only the current state of the system.

The acceleration, velocity, and position are updated as follows:

$$\mathbf{a}_n = \frac{\mathbf{F}(\mathbf{x}_n)}{m},$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + \Delta t\,\mathbf{a}_n,$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t\,\mathbf{v}_n.$$

This formulation does not require solving a system of equations and is computationally efficient; however, it is conditionally stable, meaning that the time step Δt must be smaller than a critical limit proportional to the system's smallest oscillation period.

IV. CHOOSING APPROPRIATE TIME STEP SIZE ΔT

The time step Δt determines both the accuracy and the stability of the integration scheme.

For a linear spring–mass system, the natural period T_n and frequency ω_n are

$T_n = 2\pi\sqrt{(m/k)}$, $\omega_n = \sqrt{(k/m)}$.

With m = 1 kg and the stiffest spring k_{max} = 20 N/m,

$\omega_n \approx 4.47$ rad/s, $T_n \approx 1.4$ s.

For explicit Euler, stability requires approximately $\Delta t < 2/\omega_n \approx 0.45$ s in an ideal linear single-spring system, but the coupled nonlinear geometry of the network demands a much smaller Δt to remain stable — typically 10^{-4} s or smaller.

In contrast, implicit Euler is unconditionally stable, allowing larger time steps (such as Δt = 0.1 s) while remaining numerically robust.

Therefore, for explicit Euler: Δt must be extremely small for stability. Implicit Euler: larger Δt values are acceptable, though excessive size can increase numerical damping.

V. EXPLICIT EULER SIMULATION AND METHOD COMPARISON

The explicit Euler method updates the velocity and position directly:

$v_{n+1} = v_n + \Delta t \cdot a(x_n)$

$x_{n+1} = x_n + \Delta t \cdot v_n$.

The acceleration a is computed from total forces:

a = (F_spring – F_gravity) / m.

Even with $\Delta t = 10^{-6}$ s, the system exhibits small oscillations that eventually grow or require impractically fine time steps to remain bounded.

The instability arises because the springs are relatively stiff while each node has small mass, giving a high natural frequency and therefore an extremely small critical time step. Thus, for this spring network, implicit Euler is clearly preferable.

Although it introduces numerical damping and requires iterative solutions, it remains stable for large time steps and efficiently converges to the correct static equilibrium.

The explicit Euler method becomes unstable even for very small Δt, making it unsuitable for stiff or nonlinear systems of this type.

VI. NUMERICAL DAMPING AND THE NEWMARK-B INTEGRATOR

In the implicit Euler scheme, acceleration is evaluated at the next time step t_{n+1}. This causes strong numerical damping,

which suppresses oscillations and drives the system to a static state faster than physically expected. Such damping is non-physical and purely numerical. The Newmark-β family of time-integration schemes provides a way to control stability and damping. Its general form is:
$x_{n+1} = x_n + \Delta t\, v_n + \Delta t^2 [(\frac{1}{2} - \beta)\, a_n + \beta\, a_{n+1}]$,
$v_{n+1} = v_n + \Delta t\, [(1 - \gamma)\, a_n + \gamma\, a_{n+1}]$. By choosing the parameters β and γ appropriately:
$\beta = 0$, $\gamma = \frac{1}{2} \rightarrow$ central difference (explicit, no damping, conditionally stable)
$\beta = \frac{1}{4}$, $\gamma = \frac{1}{2} \rightarrow$ average acceleration (implicit, unconditionally stable, minimal damping)
$\beta > \frac{1}{4} \rightarrow$ increased numerical damping (similar to implicit Euler)
A Newmark-β scheme with $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{2}$ maintains unconditional stability but significantly reduces artificial damping.
It provides a better balance between the stability of implicit Euler and the accuracy of explicit methods, producing a more realistic dynamic response.

VII. Results and Discussion

Figures 1–4 show the shape of the spring network at t= 0.1, 1, 10, and 100 seconds, respectively, simulated using the implicit Euler method.
Initially (t=0s), the structure is in its undeformed configuration.

As time progresses, the two free nodes (1 and 3) move downward under gravity while the fixed nodes (0 and 2) remain stationary.
The springs elongate, and the system oscillates slightly before gradually reaching a static equilibrium position due to the inherent numerical damping of the implicit scheme.

At t=0.1s, the network begins to deform; the upper free node (1) displaces slightly downward while the lower free node (3) experiences a larger deflection.
At t=1.0s, oscillations are visible but quickly attenuated.
By t=10.0s, the system is nearly at rest, and at t=100.0s, it has fully converged to a stable equilibrium configuration.
The final geometry shows both free nodes displaced downward and the springs stretched, forming a symmetric catenary-like shape.

The vertical displacements of nodes 1 and 3 are plotted as functions of time in Figure 5.
Both nodes exhibit an initial transient phase characterized by oscillatory motion caused by gravity-induced acceleration and spring elasticity.

However, for Figure 6-9, because the implicit Euler method introduces strong numerical damping, these oscillations decay rapidly.

Node 1 (the middle node) experiences smaller downward displacement due to its connection to the upper fixed node. Node 3 (the bottom free node) undergoes a larger displacement because it is more loosely constrained. The difference in their motion demonstrates how stiffness distribution and connectivity affect dynamic response in multi-spring systems.

After approximately 10s, both nodes reach equilibrium. At steady state, gravitational and elastic forces balance each other, leading to constant positions: $F_{spring} + F_{gravity} = 0$. This equilibrium confirms that the numerical scheme correctly enforces static equilibrium for nonlinear deformations.

To compare numerical stability, the explicit Euler method was implemented using the same network parameters.
For time steps $\Delta t = 10^{-3}$s or larger, the simulation diverged within the first few iterations, with displacements growing unboundedly. Even at $\Delta t = 10^{-4}$s, oscillations amplified exponentially, indicating instability. At extremely small time steps ($\Delta t = 10^{-6}$s), the method remained barely stable but required an impractically large number of iterations to reach t=1s.

This behavior is typical of stiff systems, where large stiffness values cause the highest natural frequency to dominate stability. The explicit Euler scheme is conditionally stable and must satisfy: $\Delta t < 2/\omega_{max}$

For this network, $\omega_{max} \approx 4.5$ rad/s, giving a critical time step around 0.45s.
However, geometric nonlinearities and multiple spring couplings make the effective stability limit much smaller, resulting in instability even for $\Delta t = 10^{-3}$s.

In contrast, the implicit Euler method is unconditionally stable, allowing larger time steps (e.g. $\Delta t = 0.1$s) while maintaining convergence.
Although it introduces damping, the method efficiently captures the correct equilibrium shape and energy balance without numerical blow-up.

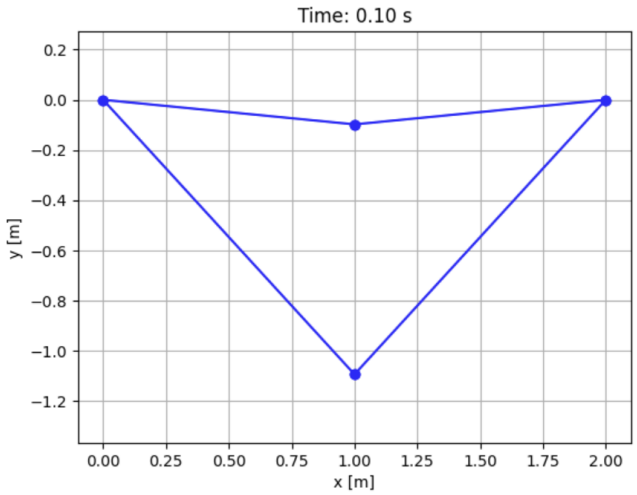

Figure1. Implicit method t=0.10s

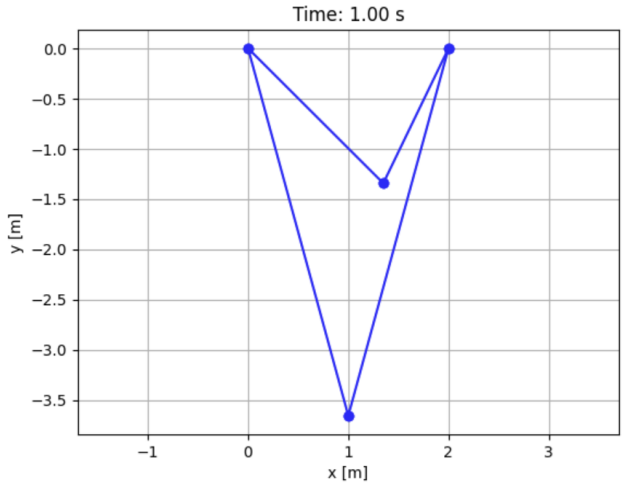

Figure 2. Implicit method t=1s

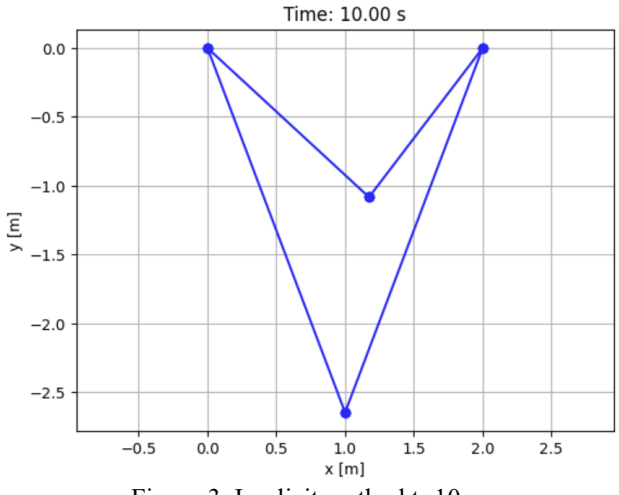

Figure 3. Implicit method t=10s

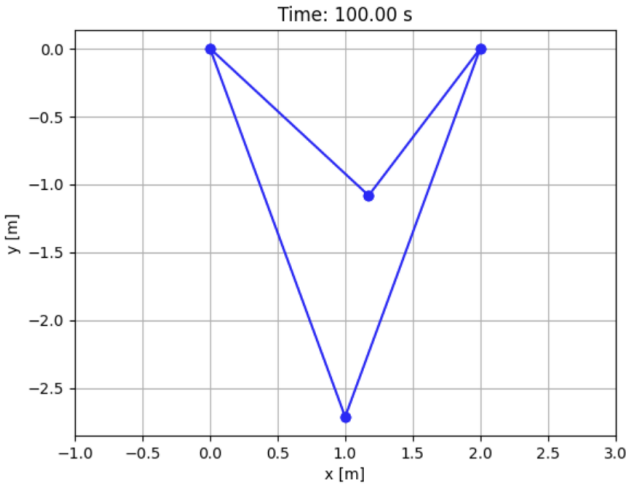

Figure 4. Implicit method t=100s

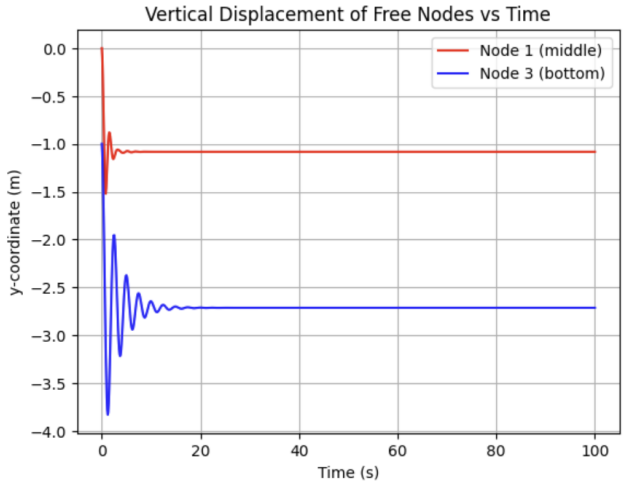

Figure 5. Implicit method vertical displacements of nodes 1 and 3

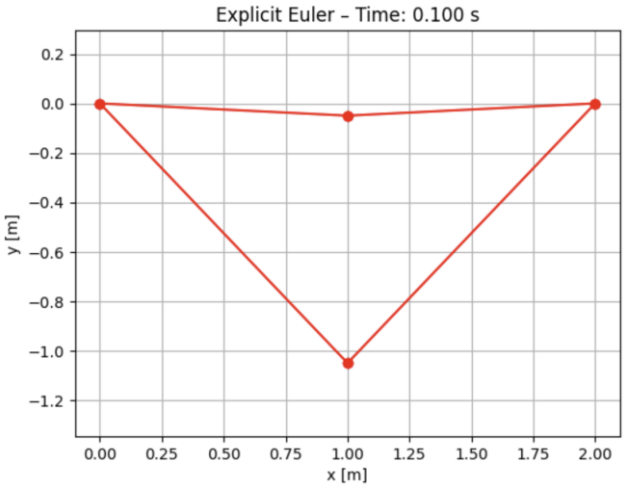

Figure 6. Explicit method t=0.1s

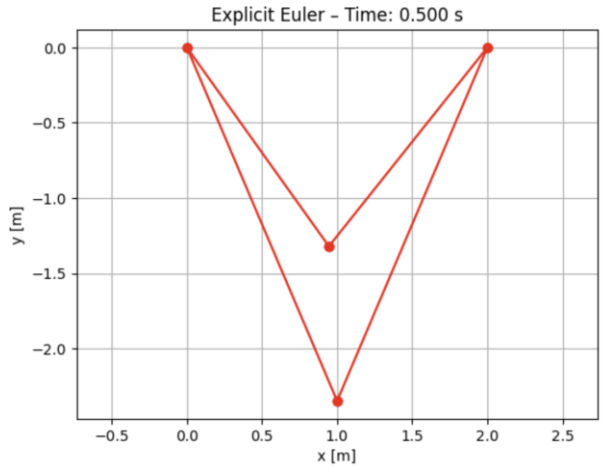

Figure 7. Explicit method t=0.5s

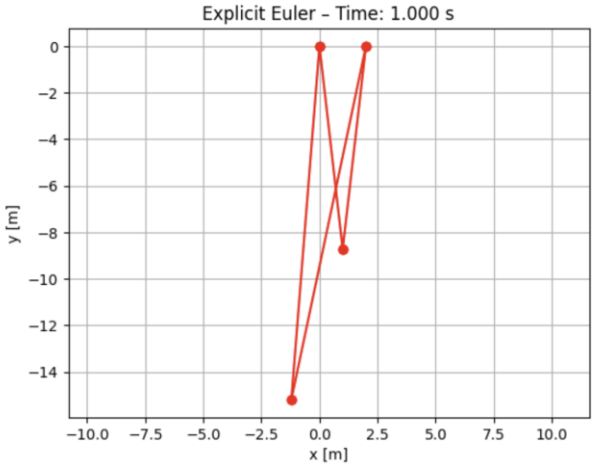

Figure 8. Explicit method t=1s

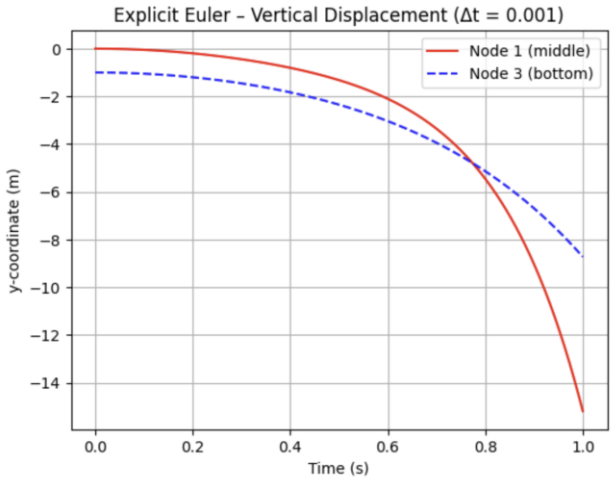

Figure 9. Explicit method vertical displacements of nodes 1 and 3

REFERENCES

[1] M. Khalid Jawed, MAE263F Lecture

[2] S. S. Rao, *Mechanical Vibrations*, 6th ed. Hoboken, NJ, USA: Pearson Education, 2017.

[3] K.-J. Bathe, *Finite Element Procedures*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.

[4] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu, *The Finite Element Method: Its Basis and Fundamentals*, 7th ed. Oxford, U.K.: Butterworth-Heinemann, 2013.