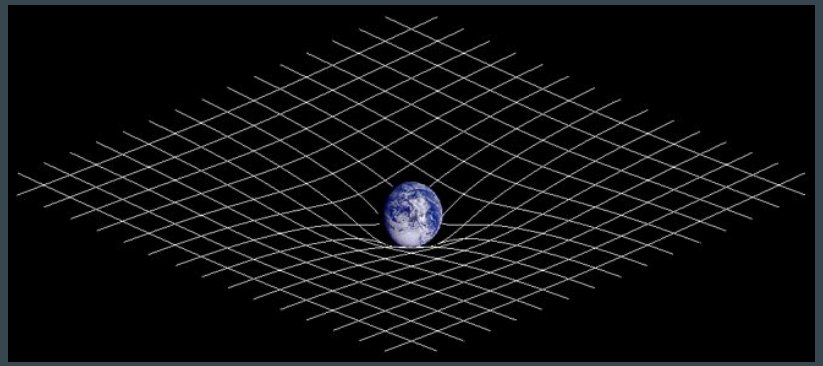
Elastic Sheet Universe Model

Harrison Sandstrom

Project Overview

Goal:

- Create a model that simulates the interaction of a deformable elastic sheet with rigid objects.
- Recreate Educational Demonstrations of Gravity

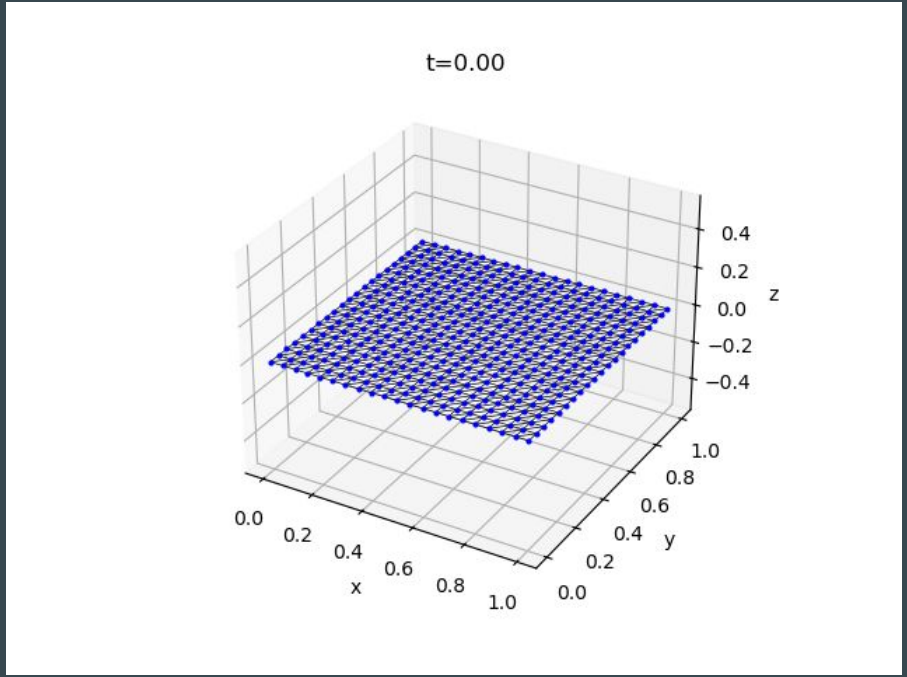


Geometric Representation

Simulated a 1x1m sheet which has NxN nodes.

Edges: Connect nodes to resist stretching (springs).

Hinges: Pairs of adjacent triangles used to calculate bending angles.



Elastic Energy Calculation

Stretching Energy

- Linear Springs as the Edges between Nodes
- Based stiffness on Young's modulus and cross-sectional area

Bending Energy

- Modeled using the change in dihedral angle between triangle normals.

Contact Mechanics

- Sphere-to-Node contact
- Checks distance between ball center and shell nodes against the ball radius.
- Repulsion Force Application:
  - Takes the penetration depth d<r and a repulsive force is applied proportional to the penetration depth
- $F_{Ccontact} = -K_{Contact} d * normal$
- Forces are applied equally and oppositely to the shell node and the falling ball.

Time Integration

Implicit Euler is used for its stability

$M\ddot{x}+C\dot{x}+\nabla E(x)=F_{ext}$.

Newton-Raphson Solver:

- We solve for the equilibrium state at t+dt by minimizing the objective function (Energy + Inertial potentials).
- Requires the global Jacobian (Hessian) matrix, combining stiffness from stretching, bending, and contact.

Simulation Setup

The Shell:

- 1.0m×1.0m fabric sheet.
  - (NxN nodes)
- Fixed boundary conditions on all four sides

The Projectiles:

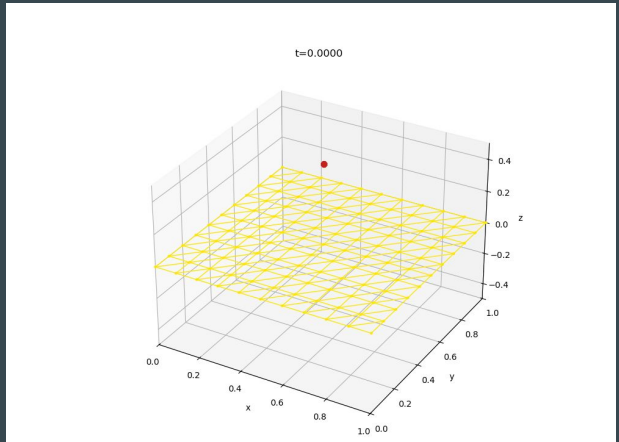
- Ball 1: Spawned at t=0.0s, Radius 0.05, Mass 1.0kg.
- Ball 2: Spawned at t=0.3s, Radius 0.08, Mass 3.0kg. (in one example)
  - Dynamic updates of the mass matrix as new objects are spawned.

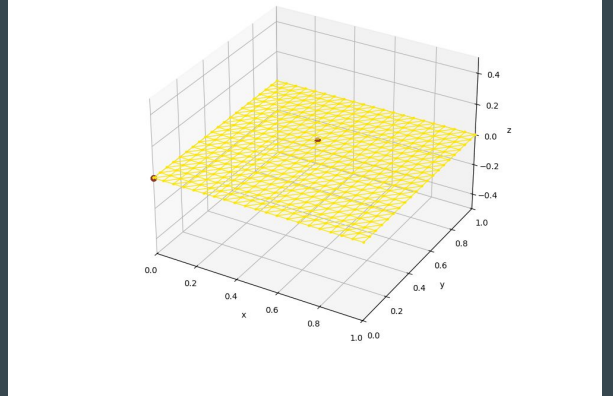
Contact Code Implementation

```
def gradFc_hessFc_BallToNode(q_ball, q_node, ball_radius, K_contact, mu_contact):
    # Force magnitude and direction
    delta = ball_radius - dist_eff      # penetration depth (>= 0)
    n = r / dist_eff                    # unit vector from node to ball
    Fn_mag = K_contact * delta          # normal force magnitude
    Fn = Fn_mag * n                     # force on ball along n
    # Force on node is opposite
    Fc[0:3] = -Fn
    Fc[3:6] = Fn
    # Jacobian (stiffness)
    # H = d(F_ball)/d(q_ball)
    H = -K_contact * np.outer(n, n) - (Fn_mag / dist_eff) * (Id3 - np.outer(n, n))
    # Block structure:
    # [ dF_node/dq_node   dF_node/dq_ball ]
    # [ dF_ball/dq_node   dF_ball/dq_ball ]
    Jc[0:3, 0:3] = H
    Jc[3:6, 3:6] = H
    Jc[0:3, 3:6] = -H
    Jc[3:6, 0:3] = -H
    return Fc, Jc
```

Initial results

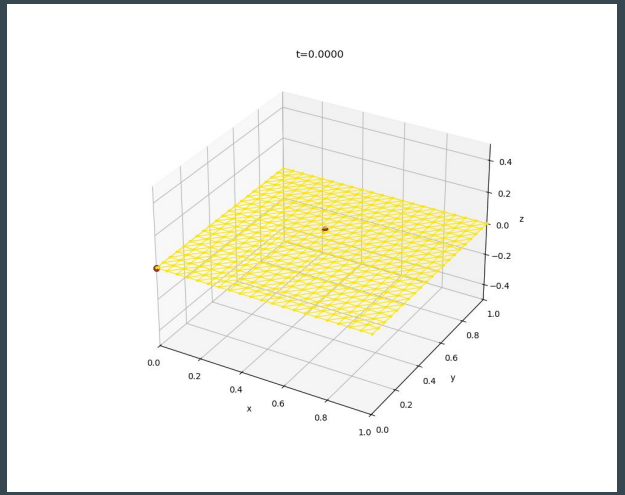
- Could handle one ball bouncing on the trampoline
- A second ball being introduce caused errors where the spike in energy would phase one ball through the mesh while the other got launched
-





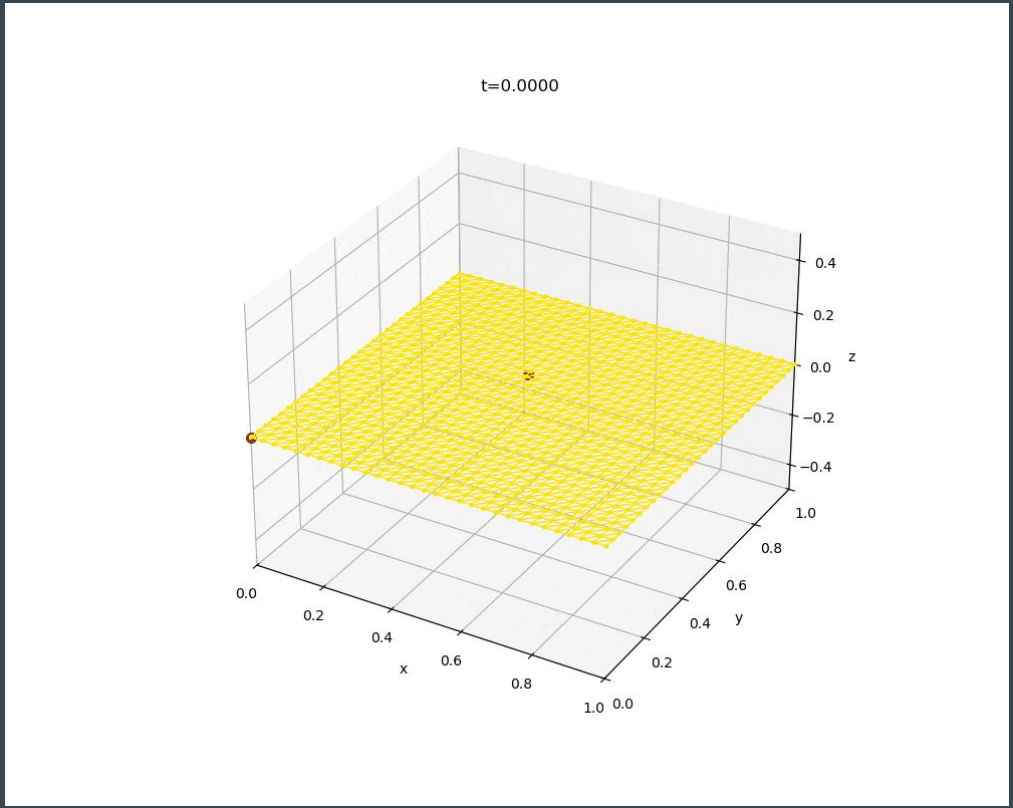
Initial Working Model

- Ball 1: Spawned at t=0.0s, Radius 0.05, Mass 1.0kg.
- Ball 2: Spawned at t=0.3s, Radius 0.08, Mass 3.0kg.

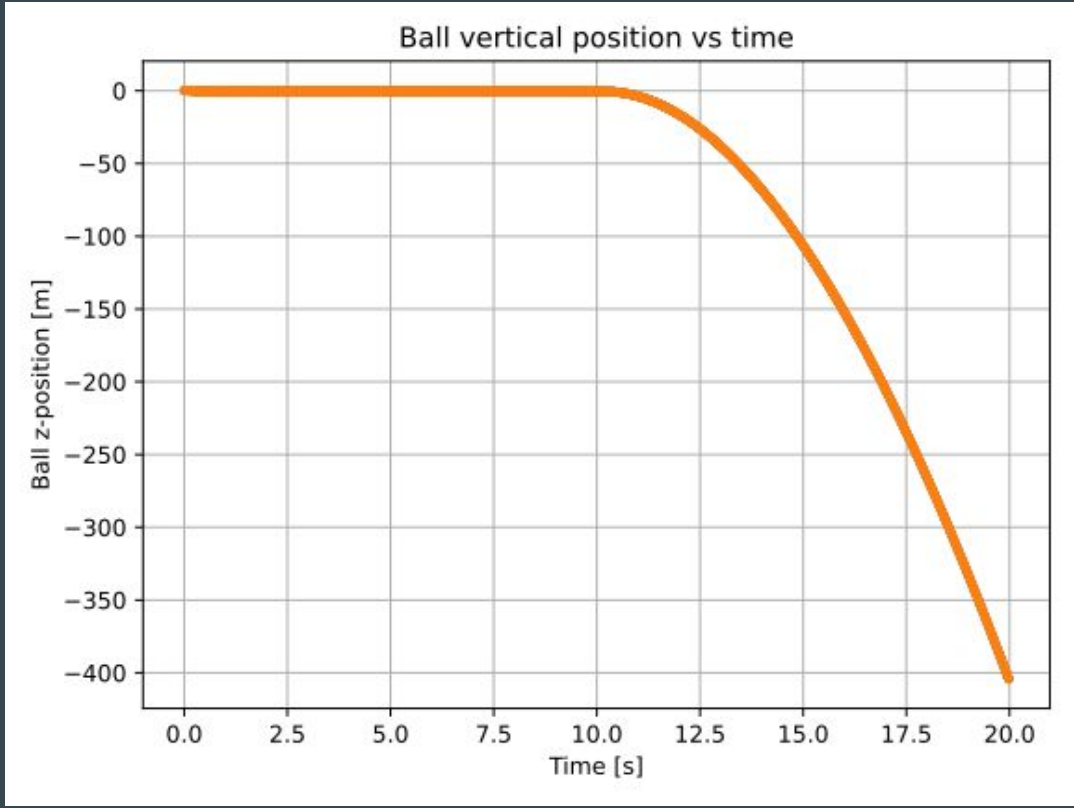


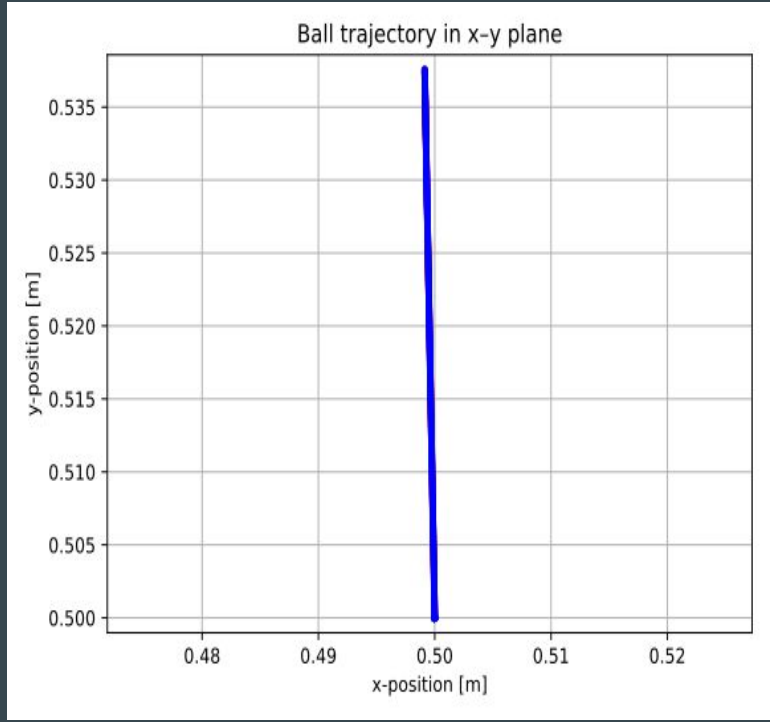
More Errors Arise

- The introduction of the second ball causes the first ball to slip through the gaps in the deformed mesh



Movement Diagram





Further Improvements

- Friction: Implementing the tangential forces in the Jacobian
- Surface Collions: Determining Collisions with the balls based off of the triangles of the mesh, not just the nodes allowing for sparser mesh
- Implementation of Better Energy Maintenance