Pick and Place Simulation of Parallel End Effector Grasping Fruits Midterm Project Report

Jessica Anz, Trevor Oshiro

***Abstract*— This project aims to develop a simulated model of a parallel end effector grasping a piece of fruit. This model should allow for the analysis of how various applied forces affect the deformation of the fruit. Current progress includes the development of a collision detection algorithm and implementation of the modified mass method as a contact model.**

I. INTRODUCTION

The usage of robotics to automate the agricultural industry is a developing field. A key element of automating crop harvesting is the picking up and placing of various crops. This is a complex task to automate, as the amount of force applied to the crop has to be precise enough to grasp the crop without damaging it [1]. In order to test various forces applied to various crops, a simulation model is needed. This project shall develop a simulation that can be used to test different forces applied to a fruit model.

II. PROJECT OBJECTIVES

The primary objective of this project is to develop a simulated model of the robotic harvesting of a piece of fruit. This model should be composed of a parallel end effector which closes in on the piece of fruit in a linear motion [2]. This model should accurately simulate the contact forces between the robot gripper and the fruit. It should also predict the fruit deformation caused by applied forces [3]. This simulation shall be developed in a 3 stage process. The first stage will create a 2D simplified model of a gripper closing on a circular fruit model. The second stage will further this model by adding complexities to both the gripper and fruit geometries. The final stage will expand this simulation into a 3D model if possible, however the majority of analysis will be done on the model from stage 2. The final simulation should facilitate the analysis of forces from a soft gripper geometry applied to a fruit. This will help prevent the crushing and dropping of fruits in real world robotic harvesting applications, which will improve the quality and efficiency of harvesting [4].

III. COLLISION DETECTION

The first step in developing the simulation is to create a working collision detection system. As a starting point, a predictor-corrector style algorithm was used to simulate basic bouncing with working contact detection. This algorithm uses the Newton-Raphson iterative method to solve the equation of motion at each time step and calculate the current position. This iterative method is used for the predictor step in the predictor-corrector method. Then the node is tested for collisions with any defined obstacles such as the ground. If a collision or reaction force from a collision is detected, then the corrector step is triggered. In the corrector method, the iterative solver is ran again using the updated position based on the detected collision. Using this algorithm, a simple simulation of a single node falling under gravity and bouncing off both the ground and a wall was created.

For clarity, pseudocode of the bouncing simulation is given below in algorithm 1. This algorithm begins by defining the basic parameters of a simple bouncing simulation, such as setting the number of nodes to 1, defining the time step, mass, and total time, and giving initial conditions. Next, the iterative Newton-Raphson solver is defined. This function takes position and velocity inputs as well as the basic simulation parameters. While the error is greater than a set tolerance and the number of iterations is less than the maximum, the solver updates the position and velocity. The equation of motion is used to solve for the force and Jacobian. These values are used to calculate the new position guess. If the error of the new guess is within the predefined tolerance, then this value is used and the loop is broken. This function returns the new position, a flag for if the function succeeded, and the reaction force.

Next, the main time-stepping is started. This loops through all time steps in the total time and uses the current position as a guess for the next position. The iterative objective function is then used to calculate the current position and reaction forces. The loop then checks each node for a collision. This is done by brute force checking if the position of the node exceeds the ground or wall, which are set to $y = 0$ and $x = 0.4$ in this example. If the node is free and a collision is detected, then the node is fixed and reset to the position of the detected obstacle. The reaction force is also checked, and if there is a reaction force that indicates a collision has happened, then the node is freed and reset to the position of the collision. If any collisions or reaction forces were detected, then the corrector step is run. This step updates the free index based on any newly fixed nodes, and then inputs the updated positions into the objective function to obtain the corrected positions. Then the velocity and current position are updated using the found values, and the loop is incremented. This simulation ends by plotting the x and y data of all time steps.

Algorithm 1 Ball Bouncing Simulation

Simulation Parameters:
  n_v, m ▷ Nodes and Mass
  Δt, T ▷ Time step and Total time
  tol, max_iter ▷ Tolerance and Max iterations
  $\mathbf{q}_0$, $\mathbf{u}_0$, $\mathbf{W}$ ▷ Initial positions, velocities, Forces
  isFixed, free_index ▷ Fixed and Free DOFs

Objective Function:
objfun $(q_{guess},\ q_{old},\ u_{old},\ parameters)$:
  $\mathbf{q}_{new},\ e,\ i,\ flag \leftarrow$ initial values
  While $e >$ tol **AND** $i <$ max_iter **do**
    $\mathbf{f} \leftarrow \frac{\mathbf{m}}{\Delta t}\left(\frac{\mathbf{q}_{new}-\mathbf{q}_{old}}{\Delta t} - \mathbf{u}_{old}\right) - \mathbf{W}$
    $\mathbf{J} \leftarrow \frac{\mathbf{mMat}}{\Delta t^2}$
    $\mathbf{q_{new}} \leftarrow \mathbf{q_{new}} - \Delta\mathbf{q}$
    $e \leftarrow \|\Delta\mathbf{q}\|$
    $i \leftarrow i+1$
  End While
  return $\mathbf{q}_{new}$, flag, $\mathbf{f}$

Time-Stepping Loop:
for timeStep $= 1$ to N_{steps} **do**
  Predictor Step:
    $\mathbf{q}_{guess} \leftarrow \mathbf{q}_0$
    $\mathbf{q}$, error, $r_{force} \leftarrow$ objfun$(q_{guess},\ q_0,\ u)$
  Collision Detection:
  for each node c **do**
    if isFixed$[c] = 0$ **AND** $\mathbf{q}[2c+1] < 0$ **then**
      isFixed$[c] \leftarrow 1$
      $\mathbf{q}_{guess}[2c+1] \leftarrow 0$
      $\mathbf{u}[2c+1] \leftarrow -\text{elasticity} \cdot \mathbf{u}[2c+1]$
    end if
    if isFixed$[c] = 0$ **AND** $\mathbf{q}[2c] > 0.4$ **then**
      isFixed$[c] \leftarrow 1$
      $\mathbf{q}_{guess}[2c] \leftarrow 0.4$
      $\mathbf{u}[2c] \leftarrow -\text{elasticity} \cdot \mathbf{u}[2c]$
    end if
    if isFixed$[c] = 1$ **AND** $r_{force}[2c+1] < 0$ **then**
      isFixed$[c] \leftarrow 0$
      $\mathbf{q}_{guess}[2c+1] \leftarrow 0$
    end if
    if isFixed$[c] = 1$ **AND** $r_{force}[2c] > 0.4$ **then**
      isFixed$[c] \leftarrow 0$
      $\mathbf{q}_{guess}[2c] \leftarrow 0.4$
    end if
  end for
  Corrector Step:
  if Correction is needed **then**
    free_index $\leftarrow$ getFreeIndex(isFixed)
    $\mathbf{q}$, error, $r_force \leftarrow$ objfun$(\mathbf{q}_{guess}, \mathbf{q}_0, \mathbf{u})$
  end if
  $\mathbf{u} \leftarrow \frac{\mathbf{q}-\mathbf{q}_0}{\Delta t}$
  $\mathbf{q}_0 \leftarrow \mathbf{q}$
  ctime $\leftarrow$ ctime $+ \Delta t$
end for

Post-Processing:
  Plot ball trajectory: (x, y) positions

The resulting plots of the ball's trajectory with different starting positions and velocities are given below in figure 1. The plot shows that the ball bounces of the ground and loses some energy with each collision. The ball also hits the right wall once and continues bouncing in the other direction. This basic simulation allowed for the development and debugging of simple collision detection, which can be expanded upon for other simulations.

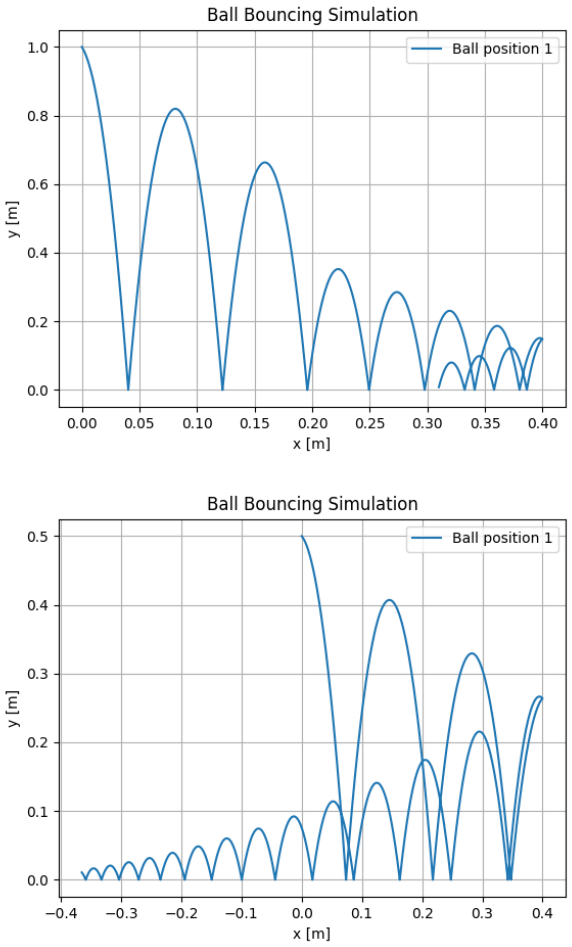


Fig. 1: Trajectory of Bouncing Simulation

IV. MODIFIED MASS METHOD ALGORITHM

As this project calls for a need to work with contact between surfaces of different types of geometry, a robust method for handling constraint enforcement of the nodes involved becomes necessary. The Implicit Contact model and the Incremental Potential Contact model incorporate these constraints from calculations of contact energy between points [5] [6]. However, within the scope of this project, the model is able to be simplified between contact between a soft body and a rigid body. Thus, the modified mass method is applicable for this project. With this simplification of the system, the simulation is then able to both enforce constraints and enforce movement by updating S and $\vec{z}$ variables within the run time [7]. Below is the application

of the program for this project.

Algorithm 2 Modified Mass Method (MMM)

Simulation Parameters:
  $n_v,\ m$ ▷ Nodes and Mass
  $\Delta t,\ T$ ▷ Time step and Total time
  tol, max_iter ▷ Tolerance and Max iterations
  $\mathbf{q}_0,\ \mathbf{u}_0,\ \mathbf{F}$ ▷ Initial positions, velocities, Forces

Initialize Loop Values:
  $Nsteps$ ▷ Number of time steps
  r_{force} ▷ Reaction Force
  s_{mat} ▷ S Matrix
  z_{vec} ▷ z Vector

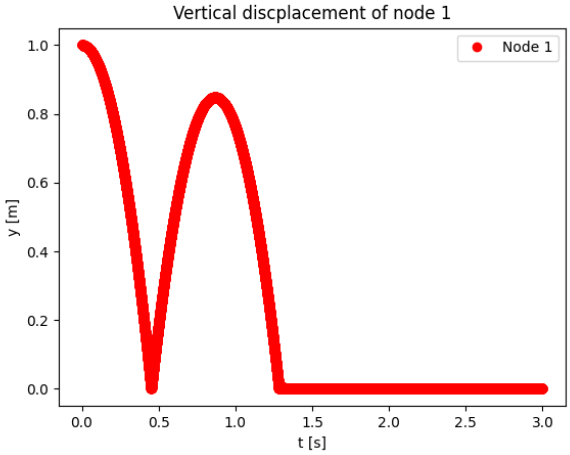
Define Modified Mass Method:
MMM $(parameters)$:
  $e, i \leftarrow$ initial values ▷ Error and iteration count
  While $e >$ tol **AND** $i <$ max_iter **do**
    $\mathbf{F}_e \leftarrow$ Compute elastic forces
    $\mathbf{f}_n \leftarrow \frac{1}{\Delta t}\left(\frac{\mathbf{q}_{\text{new}}-\mathbf{q}_{\text{old}}}{\Delta t} - \mathbf{u}_{\text{old}}\right) - \frac{1}{m}\mathbf{SF} - \mathbf{z}$ ▷ Mass
    $\mathbf{J}_n \leftarrow \frac{\mathbf{I}}{\Delta t^2}$ ▷ Jacobian matrix
    $\Delta\mathbf{q} \leftarrow \mathbf{J}_n^{-1}\mathbf{f}_n$ ▷ Solve for position update
    $\mathbf{q}_{\text{new}} \leftarrow \mathbf{q}_{\text{new}} - \Delta\mathbf{q}$ ▷ Update positions
    $e \leftarrow \|\Delta\mathbf{q}\|$ ▷ Update error
    $i \leftarrow i+1$ ▷ Increment iteration count
  End While
  Return: $r_{force}, q, flag$

Time-Stepping Loop:
for $timeStep = 1$ to $Nsteps$ **do**
    $r_{force}, q, flag \leftarrow MMM(\dots)$ ▷ MMM function
    $con_{ind}, free_{ind}$ ▷ Check Collisions
    $S_{mat},\ z_{vec}$ ▷ Update S matrix and z vector
  if $flag == 1$ **then**
      $r_{force}, q, flag \leftarrow MMM(\dots)$ ▷ MMM function
  end if
    $q_0 \leftarrow q$ ▷ Update displacement
    $\mathbf{u} \leftarrow \frac{\mathbf{q}_{\text{new}}-\mathbf{q}_{\text{old}}}{\Delta t}$ ▷ Update velocities
    $ctime \leftarrow ctime + dt$ ▷ Update time
end for

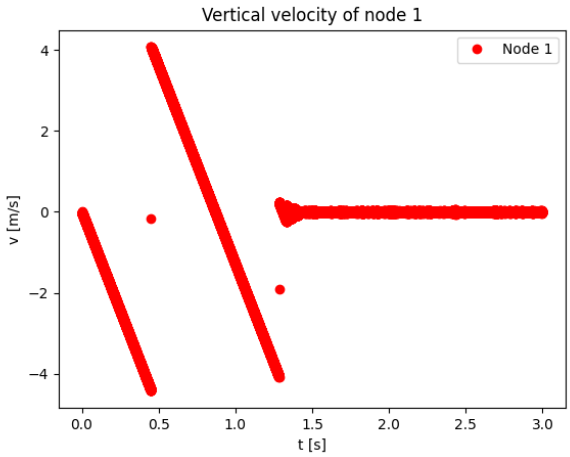
Post-Processing:
  Store and plot data

This program was tested with a system containing only one node, where the time step was $1e-4s$ during the run time. Figure 2 shows plots obtained during the simulation time:

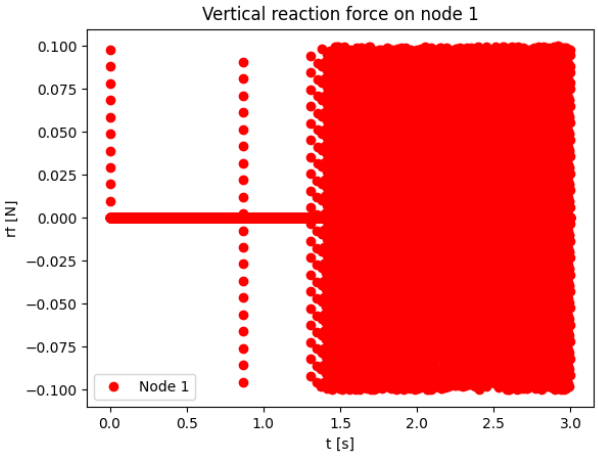
Within the run of the simulation, the bouncing motion is obtained with the enforcement of acceleration from the contact with the floor of $y = 0$. This then gives values for S and $\vec{z}$, which then changes the velocity of the node upon contact. However, after the second bounce in figure 2, there is a noticeable leveling off of the bouncing simulation.



(a) Displacement



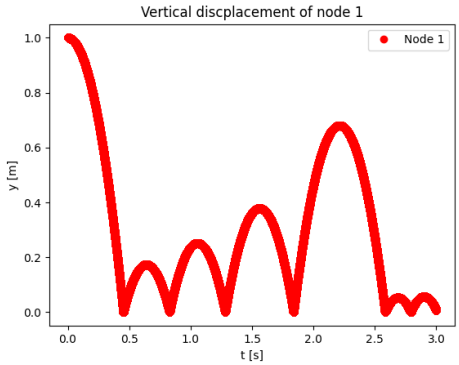
(b) Velocity



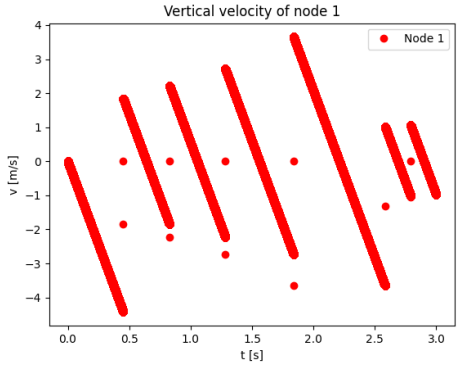
(c) Reaction force

Fig. 2: The plots show vertical displacement 2a, vertical velocity 2b, and vertical reaction force 2c plotted against time in the simulation for a time step of $1e-4s$.

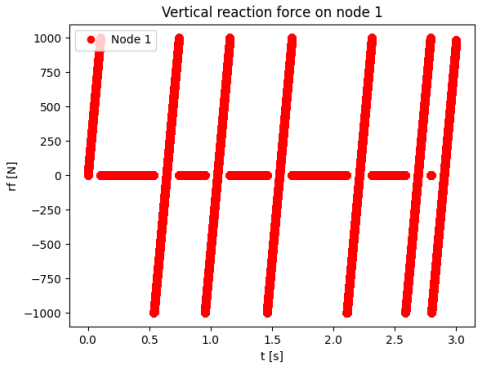
When also referencing the plot of reaction force in 2c, the high frequency of oscillation in the reaction force indicates an oscillation of small amplitude bounces after the second bounce. This is likely due to the computation of the change in velocity upon contact being significantly lower during the time step. Effect of the time step in the collision simulation was further investigated by testing the same simulation with a smaller time step of $1e-6s$, where plots were obtained shown in figure 3.



(a) Displacement



(b) Velocity



(c) Reaction force

Fig. 3: The plots show vertical displacement 3a, vertical velocity 3b, and vertical reaction force 3c plotted against time in the simulation for a time step of $1e-6s$.

Within the plots of figure 3, there is not high frequency collisions as seen with figure 2, but the bounces don't consistently decrease in amplitude as would be expected. The smaller time step used for the simulation has also likely impacted the reaction force calculation, as there is a noticeable increase of overall values in 3c compared to 2c.

V. FUTURE PROGRESS

Within this stage of the project, the both initial contact simulations and MMM methods were developed for single node cases. Currently, the MMM implementation can include multiple nodes in a straight beam under the contact between a flat surface as well.

As the model within the simulation solely involves forces from the inertial effects, verification of the implementation of other bending and stretching forces would need to be done. As seen with the results of our MMM implementation of a single node, possible tests to adjust time steps of the simulation may be of interest. Potentially altering the time step size for points closer to contact could be investigated to improve the precision of the reaction forces and $\vec{z}$ quantity obtained.

This would then lead into testing the effectiveness of the MMM implementation with multiple nodes over varied surface geometries. This would then serve as a control to further base our observations we would have for more involved contact between surfaces.

VI. CONCLUSIONS

Overall, both collision detection and modified mass method contact model algorithms have been developed. These algorithms provide the basis for more complex simulations that include both collision detection and contact between objects. While these basis algorithms provide a good starting point, they need to be expanded to include multiple nodes and various structures. These algorithms should also be integrated in order to be used for simulation. Future work in this project aims to integrate and expand the basic algorithms in order to develop a simulated parallel end effector grasping a piece of fruit.

REFERENCES

[1] L. Mu, G. Cui, Y. Liu, Y. Cui, L. Fu, and Y. Gejima, "Design and simulation of an integrated end-effector for picking kiwifruit by robot," *Information Processing in Agriculture*, vol. 7, no. 1, pp. 58–71, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214317318304372

[2] P.-L. Chang, I.-T. Chi, N. D. K. Tran, and D.-A. Wang, "Design and modeling of a compliant gripper with parallel movement of jaws," *Mechanism and Machine Theory*, vol. 152, p. 103942, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X20301634

[3] K. Chen, T. Li, T. Yan, F. Xie, Q. Feng, Q. Zhu, and C. Zhao, "A soft gripper design for apple harvesting with force feedback and fruit slip detection," *Agriculture*, vol. 12, no. 11, 2022. [Online]. Available: https://www.mdpi.com/2077-0472/12/11/1802

[4] E. Navas, R. R. Shamshiri, V. Dworak, C. Weltzien, and R. Fernández, "Soft gripper for small fruits harvesting and pick and place operations," *Frontiers in Robotics and AI*, vol. 10, 2024. [Online]. Available: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2023.1330496

[5] D. Tong, A. Choi, J. Joo, and M. K. Jawed, "A fully implicit method for robust frictional contact handling in elastic rods," *Extreme Mechanics Letters*, vol. 58, p. 101924, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352431622002000

[6] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: Intersection- and inversion-free large deformation dynamics," *ACM Trans. Graph. (SIGGRAPH)*, vol. 39, no. 4, 2020.

[7] D. Baraff and A. Witkin, "Large steps in cloth simulation," in *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 43–54. [Online]. Available: https://doi.org/10.1145/280814.280821