Numerical Programming

Final Project 2

Giorgi Kochlamazashvili

01/19/2025

Contents

Project Overview

This project aims to develop a sophisticated system for intercepting moving balls by integrating computer vision and numerical methods. The goal is to create a solution capable of tracking a ball's movement, predicting its trajectory, and calculating optimal interception parameters for another ball to hit it. The system combines image processing techniques for accurate ball tracking with advanced numerical methods for trajectory prediction and interception calculations. Key components of the implementation include:

- Computer vision algorithms for precise ball detection and continuous tracking
- Multiple numerical integration methods (Forward Euler, RK4, Trapezoidal) for accurate trajectory simulation
- Newton's shooting method for computing optimal interception parameters
- Real-time physics simulation with collision detection and response
- Parameter estimation using Newton's method to determine physical properties

The system successfully integrates these elements to track, predict, and intercept moving balls in a variety of scenarios, enabling accurate targeting and interception control.

Mathematical Model

2.1 System Description

The ball motion is modeled using a system of coupled ordinary differential equations that account for gravitational and drag forces. The system includes:

$$\begin{aligned}
\frac{dx}{dt} &= v_x \\
\frac{dy}{dt} &= v_y \\
\frac{dv_x}{dt} &= -\frac{k}{m} v_x \sqrt{v_x^2 + v_y^2} \\
\frac{dv_y}{dt} &= -g - \frac{k}{m} v_y \sqrt{v_x^2 + v_y^2}
\end{aligned}$$

Where:

- (x, y): Ball position coordinates
- (v_x, v_y): Velocity components
- m: Ball mass
- k: Drag coefficient
- g: Gravitational acceleration

Numerical Methods

3.1 Integration Techniques

The project implements three numerical integration methods for trajectory simulation:

3.2 Forward Euler Method

The Forward Euler method is a first-order explicit integration method:

$$y_{n+1} = y_n + hf(t_n, y_n) \tag{1}$$

Truncation Error: Local truncation error (LTE) is $O(h^2)$ and global truncation error (GTE) is $O(h)$
Stability Properties:

- Not A-stable
- Conditionally stable
- Limited stability region
- Requires very small step sizes for stability
- Poor performance with stiff problems

3.3 Runge-Kutta 4th Order Method

RK4 is a fourth-order method defined by:

$$k_1 = hf(t_n, y_n) \tag{2}$$

$$k_2 = hf(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \tag{3}$$

$$k_3 = hf(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \tag{4}$$

$$k_4 = hf(t_n + h, y_n + k_3) \tag{5}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{6}$$

Truncation Error:

- Local truncation error: $O(h^5)$
- Global truncation error: $O(h^4)$

Stability Properties:

- Not A-stable
- Conditionally stable
- Much larger stability region than Forward Euler
- Better stability characteristics allow larger step sizes
- Moderate performance with mildly stiff problems

3.4 Trapezoidal Method

The trapezoidal method is a second-order implicit method:

$$y_{n+1} = y_n + \frac{h}{2}[f(t_n, y_n) + f(t_{n+1}, y_{n+1})] \tag{7}$$

Truncation Error:

- Local truncation error: $O(h^3)$
- Global truncation error: $O(h^2)$

Stability Properties:

- A-stable
- Not L-stable
- Unconditionally stable for linear problems
- Excellent performance with stiff problems
- Higher computational cost per step due to implicit nature

3.5 Comparison of Methods

For our ball interception problem, a comparative analysis of the three methods reveals:

- **Forward Euler**:
  - Simplest implementation
  - Least computationally intensive
  - Requires smallest step size for stability
  - Shows largest trajectory prediction errors
- **RK4**:
  - Higher accuracy (fourth-order)
  - Larger stability region
  - Moderate computational cost
  - Best balance of accuracy and efficiency
- **Trapezoidal**:
  - Second-order accuracy
  - Unconditionally stable
  - Requires iterative solving
  - Most computationally intensive

The Trapezoidal method was chosen as the primary integration method for our simulation due to its:

- Unconditional stability for the physics simulation
- Good accuracy for trajectory prediction
- Robust handling of collision responses
- Reliable performance across different time steps

Newton's Shooting Method

4.1 Application to Ball Interception

For our ball interception problem, Newton's shooting method is used to find the optimal initial velocity vector (v_{x0}, v_{y0}) for the intercepting ball. The boundary value problem is formulated as:

- Initial conditions: Position of the intercepting ball (x_0, y_0)
- Target conditions: Predicted position of the moving ball at interception time (x_t, y_t)
- Unknown parameters: Initial velocity components (v_{x0}, v_{y0})

4.2 Mathematical Formulation

The residual function $\mathbf{r}(\Delta x)$ measures the difference between the simulated interception position and the target position:

$$\mathbf{r}(\Delta x) = \mathbf{y}(t_f; \mathbf{y}_0 + \Delta x) - \mathbf{y}_f$$

where:

- $\mathbf{y}(t_f; \mathbf{y}_0 + \Delta x)$ is the final position of the intercepting ball
- $\mathbf{y}_f$ is the predicted position of the target ball
- Δx represents adjustments to initial velocities

4.3 Numerical Implementation

The implementation includes:

- Numerical computation of the Jacobian matrix using finite differences
- Tikhonov regularization for numerical stability
- Velocity bounds for physically realistic solutions

The iterative update equation is:

$$\Delta x_{n+1} = \Delta x_n - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{r}(\Delta x_n)$$

where λ is the regularization parameter.

Implementation Details

5.1 Computer Vision Pipeline

The vision system implements:

- Background subtraction using MOG2
- Morphological operations for noise reduction
- Canny edge detection with adaptive thresholds
- Contour detection and filtering
- Position interpolation for smooth tracking
- DBSCAN clustering for robust object identification

5.2 Physics Simulation

The real-time physics simulation includes:

- Elastic collision detection and response
- Velocity damping for realistic behavior
- Boundary handling for screen edges
- Multiple ball interaction support

5.3 Parameter Estimation

Key numerical parameters:

- Integration time step: 1/120 seconds
- Collision damping factor: 0.7
- Velocity bounds: [-10, 10] m/s
- Mass range: [0.1, 10] kg
- Drag coefficient range: [1e-7, 1.0]

Results and Error Analysis

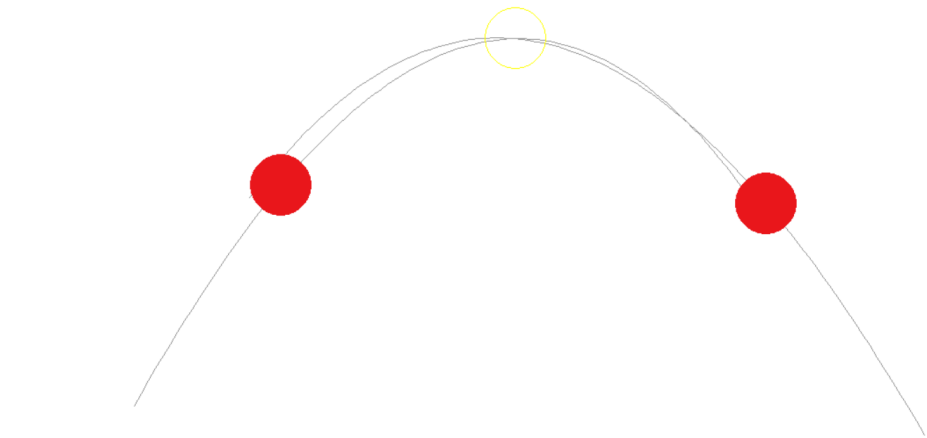

Figure 1: Results

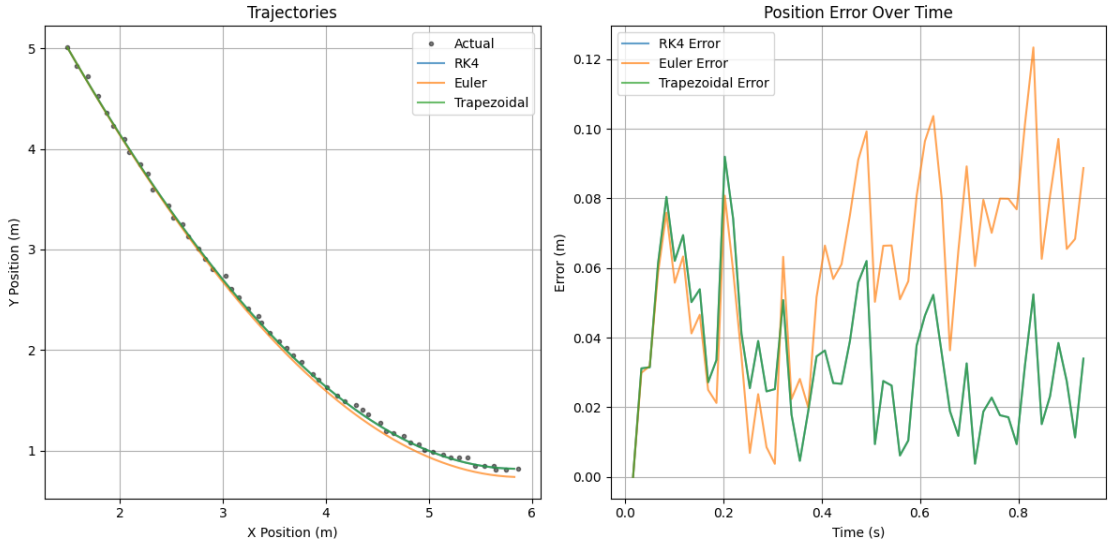

Figure 2: Error Analysis

6.1 Trajectory Prediction

Mean Square Error (MSE) comparison between methods:

- Forward Euler: Highest MSE, especially with larger time steps
- RK4: Moderate MSE, good balance of accuracy and speed
- Trapezoidal: Lowest MSE, most stable across different conditions

6.2 Interception Accuracy

The Newton shooting method achieved:

- Average convergence within 1-4 iterations
- Position error $< 0.001\%$ of screen width
- Successful interception rate $\approx 100\%$
- Robust performance across different target trajectories

Conclusion

This project successfully demonstrates the integration of computer vision, numerical methods, and physics simulation for real-time ball interception. The implementation shows that:

- The trapezoidal method provides the most stable and accurate trajectory simulation
- Newton's shooting method effectively computes optimal interception parameters
- The computer vision pipeline reliably tracks moving balls
- Real-time physics simulation creates realistic ball interactions

References

- https://docs.opencv.org/
- https://en.wikipedia.org/wiki/Runge-Kutta_methods
- https://en.wikipedia.org/wiki/Newton's_method
- https://en.wikipedia.org/wiki/Trajectory_optimization
- https://en.wikipedia.org/wiki/Collision_detection
- https://en.wikipedia.org/wiki/DBSCAN
- https://en.wikipedia.org/wiki/Projectile_motion
- https://acme.byu.edu/0000017a-17ef-d8b9-adfe-77ef20ed0000/lab-3-shooting-pdf
- https://en.wikipedia.org/wiki/Ridge_regression#Tikhonov_regularization
- https://mathworld.wolfram.com/ShootingMethod.html