Numerican Programming

Final Project 1

Giorgi Kochlamazashvili

01/20/2024

Contents

Project Overview

This project aims to develop a robust system for analyzing and simulating ball motion by integrating computer vision and numerical methods. The goal was to create a solution capable of reliably tracking a ball's movement and predicting its trajectory while considering real-world physical constraints. To achieve this, the system combines image processing techniques for accurate ball tracking with advanced numerical methods for trajectory prediction and parameter estimation. Key components of the implementation include:

- Computer vision algorithms for precise ball detection and continuous tracking
- Numerical integration methods for accurate trajectory analysis and prediction
- Parameter estimation using Newton's shooting method to optimize shooting angles and forces
- Real-time simulation that incorporates physical constraints, allowing for dynamic interaction with the environment

The system successfully integrates these elements to simulate and predict ball trajectories in a variety of scenarios, enabling accurate control and targeted ball shooting.

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:

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

- **Forward Euler**:
  - Simplest implementation
  - Least computationally intensive
  - Requires smallest step size for stability
  - First-order accuracy
- **RK4**:
  - Higher accuracy (fourth-order)
  - Larger stability region
  - More computational cost per step
  - Good balance of accuracy and efficiency
- **Trapezoidal**:
  - Second-order accuracy
  - Unconditionally stable
  - Requires iterative solving at each step
  - Better for stiff problems

For our ball motion problem, RK4 was chosen as the primary integration method due to its:

- High accuracy for smooth trajectories
- Reasonable stability properties
- Explicit nature (no iteration required)
- Good performance for non-stiff mechanical systems

Newton's Shooting Method

Newton's shooting method is a numerical technique used for solving boundary value problems (BVPs) by converting them into an equivalent initial value problem (IVP). The method iteratively refines an initial guess for the unknown parameters to satisfy the boundary conditions.

4.1 Mathematical Derivation

The method leverages the Taylor series expansion to approximate the solution near the desired boundary. For a boundary value problem:

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \quad \mathbf{y}(t_f) = \mathbf{y}_f,$$

we introduce an unknown initial condition adjustment Δx to represent the deviation required to meet the boundary conditions.

Newton's method refines Δx iteratively to minimize the residual $\mathbf{r}(\Delta x)$:

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

where $\mathbf{y}(t_f; \mathbf{y}_0 + \Delta x)$ is the computed solution at t_f using the adjusted initial condition $\mathbf{y}_0 + \Delta x$.

Expanding $\mathbf{r}(\Delta x)$ using the Taylor series about the current estimate Δx_n:

$$\mathbf{r}(\Delta x_{n+1}) \approx \mathbf{r}(\Delta x_n) + \mathbf{J}(\Delta x_n)(\Delta x_{n+1} - \Delta x_n),$$

where $\mathbf{J}(\Delta x_n) = \frac{\partial \mathbf{r}}{\partial (\Delta x)}$ is the Jacobian matrix of the residual with respect to Δx.

Setting $\mathbf{r}(\Delta x_{n+1}) = 0$ and solving for Δx_{n+1}:

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

This iterative formula is the basis of Newton's shooting method.

4.2 Jacobian Matrix

The Jacobian matrix $\mathbf{J}(\Delta x)$ represents the sensitivity of the residual to changes in the initial condition adjustment Δx. It is defined as:

$$\mathbf{J}(\Delta x) = \frac{\partial \mathbf{r}}{\partial(\Delta x)} = \frac{\partial}{\partial(\Delta x)} \left(\mathbf{y}(t_f; \mathbf{y}_0 + \Delta x) - \mathbf{y}_f\right).$$

Computing $\mathbf{J}(\Delta x)$ often requires solving additional linearized IVPs to evaluate the partial derivatives of $\mathbf{y}(t_f)$ with respect to Δx.

4.3 Tikhonov Regularization

Tikhonov regularization is used to stabilize the solution when the Jacobian $\mathbf{J}(\Delta x)$ is ill-conditioned or near-singular. The regularized system modifies the Jacobian to include a small damping term λ:

$$\left(\mathbf{J}(\Delta x)^T \mathbf{J}(\Delta x) + \lambda \mathbf{I}\right) \Delta x = -\mathbf{J}(\Delta x)^T \mathbf{r}(\Delta x),$$

where:

- λ: Regularization parameter
- $\mathbf{I}$: Identity matrix
- $\Delta x = \Delta x_{n+1} - \Delta x_n$

The term $\lambda \mathbf{I}$ penalizes large deviations in Δx, ensuring numerical stability while maintaining convergence. The regularization parameter λ is often chosen empirically or through techniques like the L-curve method.

4.4 Advantages and Limitations

Advantages:

- High accuracy for well-posed problems.
- Flexibility to handle nonlinear boundary conditions.
- Stabilization via regularization.

Limitations:

- Sensitivity to initial guesses.
- Computational cost for large systems.
- Dependence on regularization parameter λ.

4.5 Parameter Estimation

Newton's shooting method is implemented with:

- Numerical Jacobian computation
- Tikhonov regularization for stability
- Bounded parameter constraints

Implementation Details

5.1 Computer Vision Pipeline

The vision system includes:

- Frame preprocessing using CLAHE
- Canny edge detection with adaptive thresholds
- Morphological operations for noise reduction
- Contour detection and filtering
- DBSCAN clustering for robust object tracking

5.2 Numerical Setup

Key parameters and constraints for Newton's Shooting Method:

- Integration tolerance: $[10^{-8}, 10^{-3}]$
- Parameter bounds:
  - Velocity: $[-20, 20]$ m/s
  - Mass: $[0.1, 10]$ kg
  - Drag coefficient: $[0, 1]$

Results and Errors Analysis

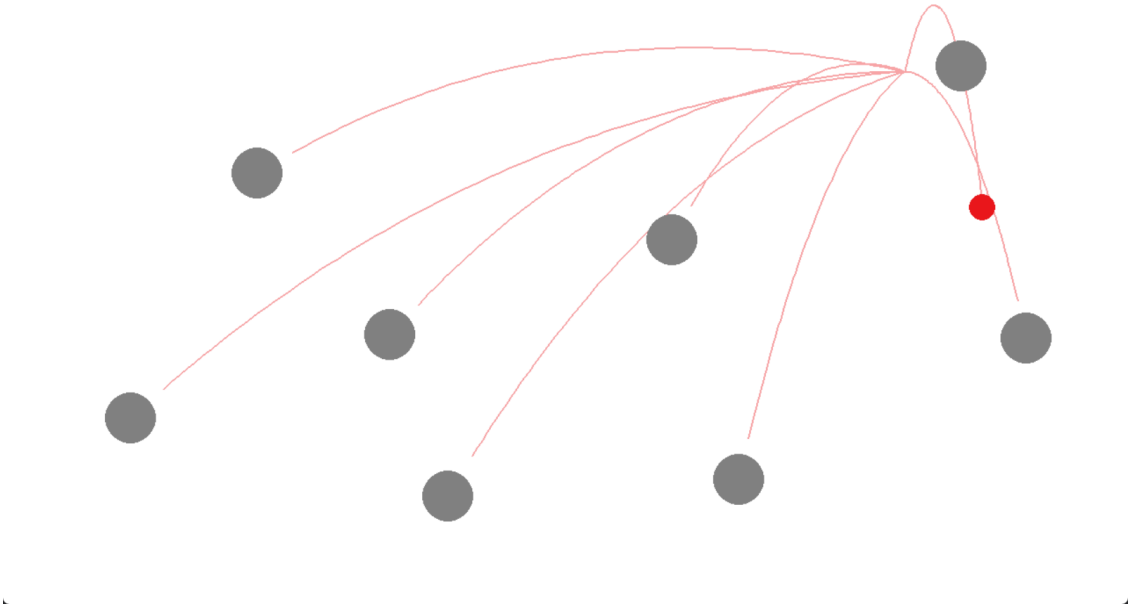

Figure 1:

6.1 Error Analysis

The performance of the numerical integration methods was evaluated based on several error metrics, including position tracking accuracy, velocity estimation error, and parameter convergence rates. Overall, the errors for all methods were relatively low, but with notable differences between the methods.

- **Forward Euler:** The Forward Euler method exhibited slightly higher errors compared to the other methods. This can be attributed to its first-order accuracy and the larger truncation error per step. While simple and computationally efficient, the method requires smaller step sizes for stable results, which can result in increased error when larger step sizes are chosen for performance optimization.
- **Runge-Kutta 4th Order (RK4):** RK4 showed very similar error characteristics to the Trapezoidal method. Both methods exhibited good accuracy with low error across most tests. RK4, being a fourth-order method, benefits from a larger stability region, which results in lower overall error for smooth trajectories. However, the performance is slightly better for RK4 due to its higher order of accuracy, allowing for more precise estimates at larger step sizes without a noticeable increase in error.
- **Trapezoidal Method:** The Trapezoidal method performed similarly to RK4. Although it is a second-order method, its implicit nature provides excellent stability for stiff problems, resulting in low error even for larger step sizes. The method's A-stability and unconditionally stable properties help prevent error growth in stiff systems, but it requires iterative solving at each step, which can increase the computational cost.

In general, all methods showed low errors under the conditions tested, with RK4 and Trapezoidal providing the most consistent and accurate results, especially for longer simulations with smoother trajectories. Forward Euler, though adequate for simple cases, demonstrated a noticeable increase in error when compared to the higher-order methods, particularly when larger step sizes were used.

Conclusion

In this project, we successfully developed a system that combines computer vision with numerical methods to track and analyze the motion of a ball. The ball tracking allowed us to extract key parameters such as velocity, mass, and drag coefficient, which were then used in numerical simulations to model the ball's trajectory.

A key part of this project was using Newton's shooting method to aim and shoot the ball with the goal of hitting other balls displayed on the screen. By leveraging the results from the ball tracking and the accuracy of our numerical methods, we applied Newton's shooting method to iteratively adjust the shooting parameters and find the correct angle and force to hit the target balls. This method demonstrated the power of combining real-time analysis with predictive simulations, allowing us to control the ball's trajectory effectively.

The results showed that using a combination of computer vision, numerical methods, and shooting simulations provided a robust and flexible system capable of hitting targets in a dynamic environment. The approach opens the door for more advanced applications, such as real-time shooting in games or sports analysis, and provides a foundation for further optimization and refinement.

References

- https://opencv.org/
- https://en.wikipedia.org/wiki/Runge-Kutta_methods
- 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