

Scientific Computing

Coursework 2: Mass-Spring Simulator

Report

By

Aly Abdel Razik - 10103900

Peter Sabry - 18270045

31-January-2021

Work Description

The application uses a numerical integrator to simulate mass-spring movements by estimating the new positions of masses given the current positions and the applied forces. System behavior changes by

Besides the implemented Euler and Midpoint integrators, four more integrators are implemented:

* Heun’s
* 4th-Order Runge-Kutta
* 5th-Order Runge-Kutta
* Adaptive 4th-Order Runge-Kutta

Heun’s:

Heun’s integrator does an integration step over the whole-time interval then it can do multiple corrective steps. The slope obtained from the previous step is averaged with the current slope and then the target value is re-calculated.

To do this, the function IntegrateSystemOverTime is used to make the first prediction over the time interval, the resulted slopes are averaged with the initial slopes then the resulted average slopes are then used to make another corrective step.

To decide how many corrective steps are done, a convergence measure was used as the percentage of change from previous and current position estimation that is:

Epsilon = | current estimate – previous estimate | / current estimate %

4th Order Runge-Kutta:

Using the system integrator functions which takes in four parameters, we calculated intermediate slopes at different points. Ultimately, these slopes were averaged as per RK4, and a final slope is obtained.

This final slope is used to update the system one time step.

Also, for the goal of comparing integrators, we saved the obtained system states in a member variable of the PhyEnv class (name: ‘states’).

5th Order Runge-Kutta:

Like RK4 method, we obtained several slopes and eventually the final slope to calculate the next time step values.

Also, for the goal of comparing integrators, we saved the obtained system states in a member variable of the PhyEnv class (name: ‘states’).

Adaptive 4th-Order Runge-Kutta

We calculate RK for twice the time step. Then again, we calculate RK for the time step. Then we calculate the error between the two. This error is used to update the obtained system from RK using the time step. Basically, we obtained higher degree of accuracy with less computation.

**As we obtain more representative slopes, we get closer to the closed form solution (true solution) of the system.**

Proposed error metric:

At the end of each time step, we saved the calculated state of the system. The series of a state along a trajectory defines the simulated results. We run the simulator under constant gravitational force for all integrators and obtained the system states.

We then compared the system states for different integrators to a reference integrators (RK5). The calculation of the error metric is implemented in a function. Basically, we calculate the difference in positions between the two system states and then square that in each spatial dimension. We then sum up these numbers to obtain an error metric. The closer this number to zero, the better. (as it means it is close to the best integrator).

Integrators Comparison

*Observations*

|  |  |  |  |
| --- | --- | --- | --- |
| **Heun’s** | **RK-4** | **RK-5** | **Adaptive RK** |
| Fastest | Slower, but seems more realistic | Slower and more realistic | Slower and more realistic |
| Slowed by:   * adding more correction steps and using smaller epsilon * Using smaller step | Slowed by smaller step size |  |  |
|  |  |  |  |

Error metrics compared to reference integrator (RK5):

|  |  |  |
| --- | --- | --- |
| **Integrator** | **Error metric** | **Error metric (in y direction only)** |
| Euler | 1.38558e+07 | 4.61859e+06 |
| MidPoint | 1.38558e+07 | 4.61859e+06 |
| Heun | 1.38558e+07 | 4.61859e+06 |
| RK4 | 1.60618e+06 | 535393 |
| AdaptiveRK | 2.30032e+39 | 7.66772e+38 |

As can be seen, the integrator RK4 is closet to RK5.

Effect of “Step Size” and “Spring Constant”

Step Size

The step size can be controlled with **m\_MaxTimeStep** which is defined in **OGLView.cpp**; changing this parameter from **0.01** to **0.005** changes the step size to the half.

Decreasing the step size affected all integrators, slowed the performance. But with better more realistic simulation results.

Spring Constant

Changing the Spring Constant from the application interface resulted in the following behavior:

1. Decreasing the value from 5 to 1: the movements of the particles are larger and it took little more time to stabilize (using gravity forces)
2. Increasing the value from 5 to 10: the movements if the particles are smaller but much faster, the particles oscillate more and took much more time to stabilize.

These effects are observed on all integrators without much difference.