ASSIGNMENT 1

Prachit Gupta 210100111

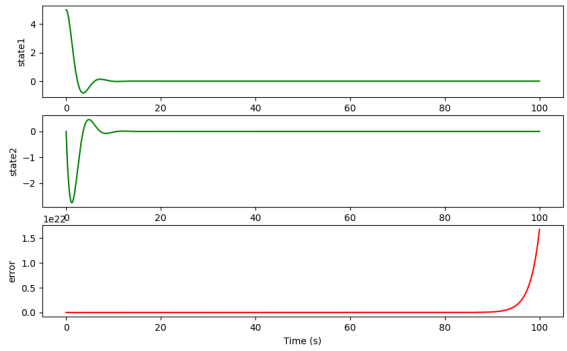
CODE :
https://github.com/prachitgupta/estimation-of-lie-groups/blob/main/spring-mass.py

Ans1)

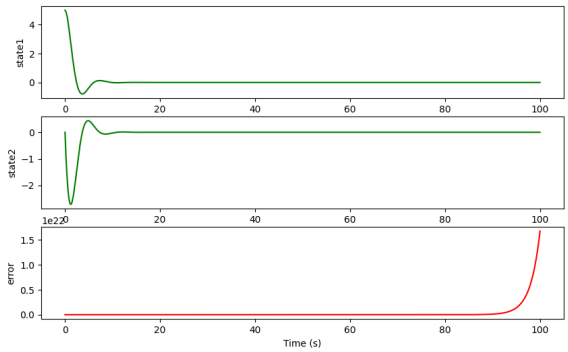
$$(\text{Spring-Mass-Damper System})\ m\ddot{x} + c\dot{x} + kx = 0$$

3 numerical integrators have been tested forward/explicit euler , backward/implicit euler And symplectic euler methods and results are plotted with errors against analytical solutions . Plots are also shown for various time periods of simulation and step size Check out code for reference , all numerical integrators are defined in a separate class for smoother analysis

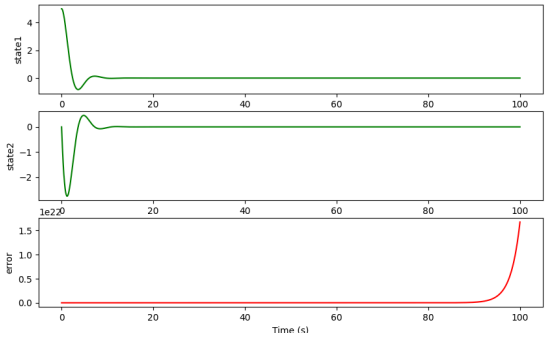
Time simulation 100s , step 0.01 , initial state = released from 5m at 0 speed
Forward euler: analysis : error(red plot) jumps exponentially towards end



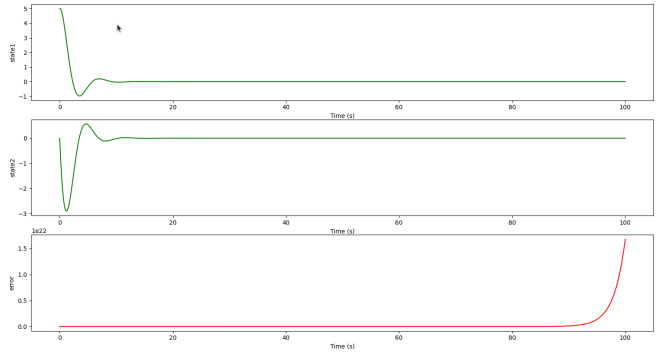
Backward euler



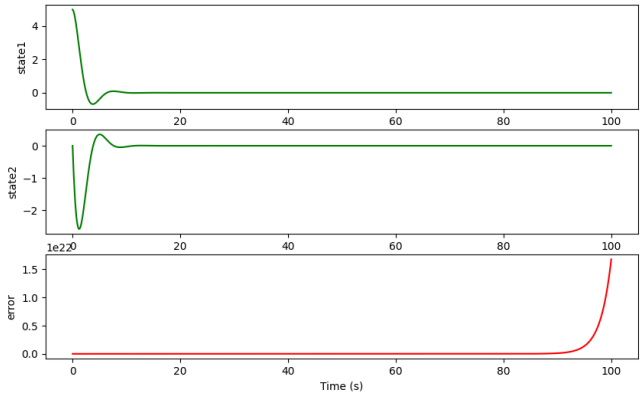
Symplectic:



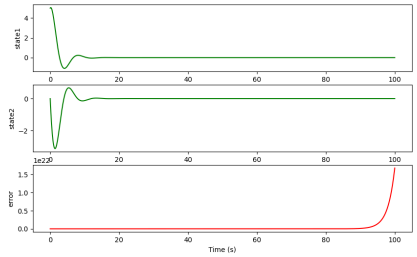
Time simulation 100s , step 0.1 , initial state = released from 5m at 0 speed
Forward



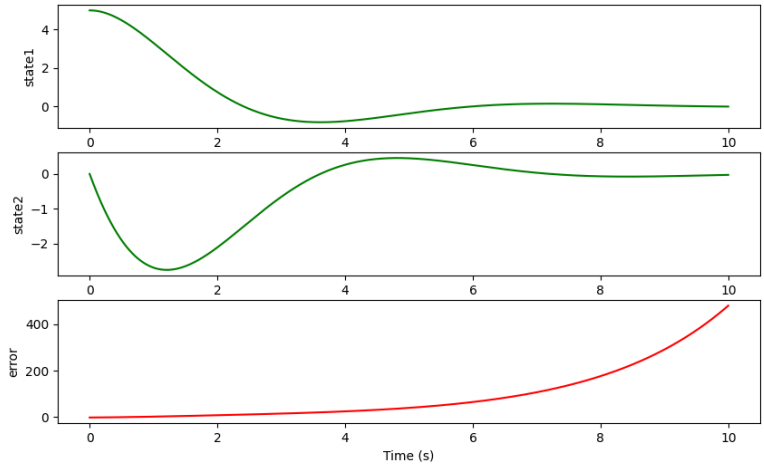
Backward



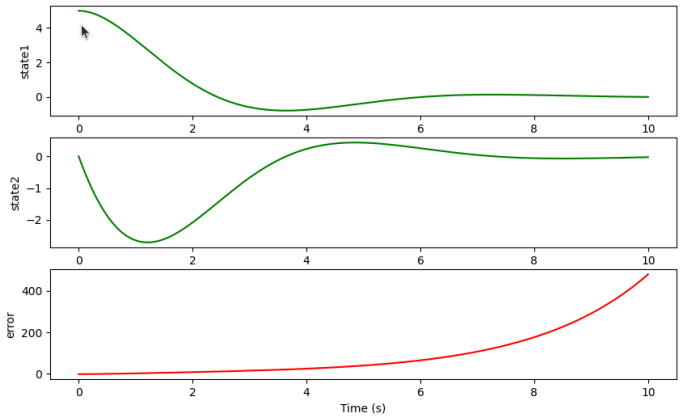
Symplectic



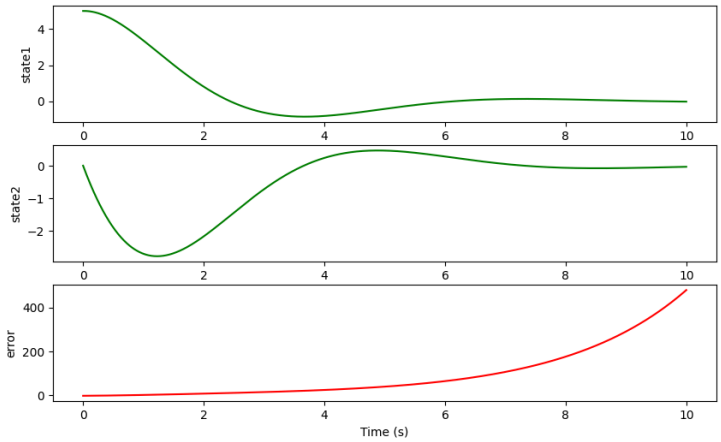
Time simulation 10s , step 0.01 , initial state = released from 5m at 0 speed



Backward



Symplectic



Analysis
Fidelity analysis : for a significant period of time the numerical integrators worked quite well shown by error with analytical solution in range of at most 0.1 then sudden leap to extremely large value

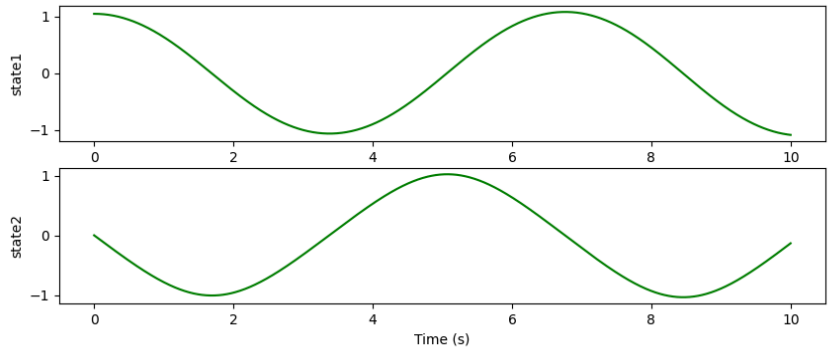
Comparison of 3 integrators : according to theory and insight methods deployed (refer code) its expected Euler forward may exhibit increasing error with smaller step sizes due to accumulation of truncation errors. Euler backward may be more stable for smaller step sizes but may introduce some phase lag compared to Euler forward. Symplectic methods should preserve energy better over long simulations but may require larger step sizes to achieve acceptable accuracy. To large extend these conclusions can be made from our plots as well showing propagation of x,v and deviation with step size and simulation time but difference is negligible for such simple system

Effect of step size : It can be seen Using a large time step (dt) can lead to inaccuracies, and error assumes large values much faster especially when dynamics gets complicated

Effect of time step : Numerical integration methods, especially explicit methods like Euler's method, introduce truncation errors at each time step which propagate over time. Over a long simulation, these errors can accumulate, leading to significant deviations from the true solution.

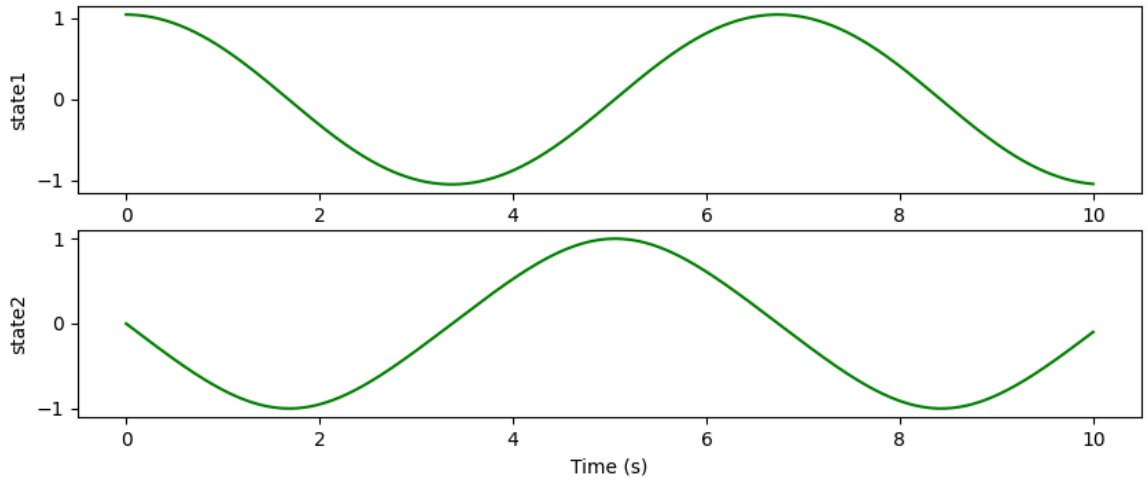
Pendulum system given
Time simulation 10s , step 0.01 , initial state = released from 60 deg at 0 speed

Forward



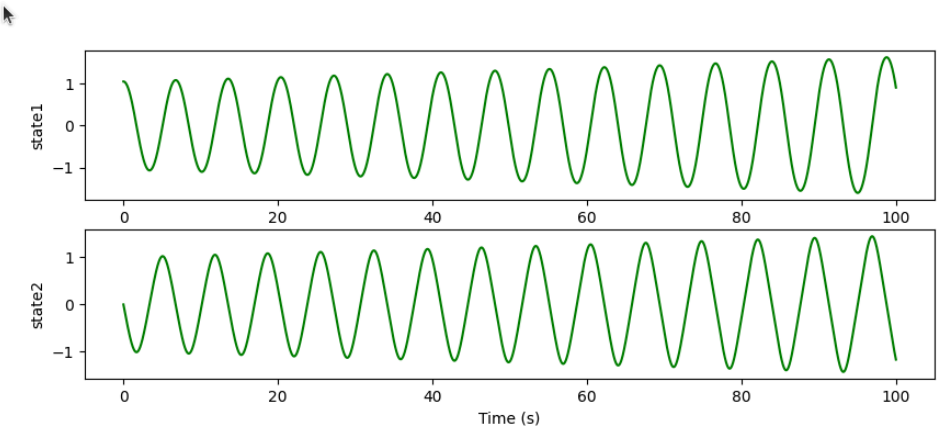
Note: The system given is non linear non affine and **doesn't have an exact analytical solution ,** Calculation of analytical solutions itself assumed some infinite series assumptions .Also the implicit euler involved solving an implicit non solvable equation unlike spring mass in which the transition function was a simple invertible matrix , so this would have involved calling some other numerical solver maybe newton finite method which would additionally propagate errors so for fidelity analysis only plots of explicit and symplectic euler have been compared

Symplectic

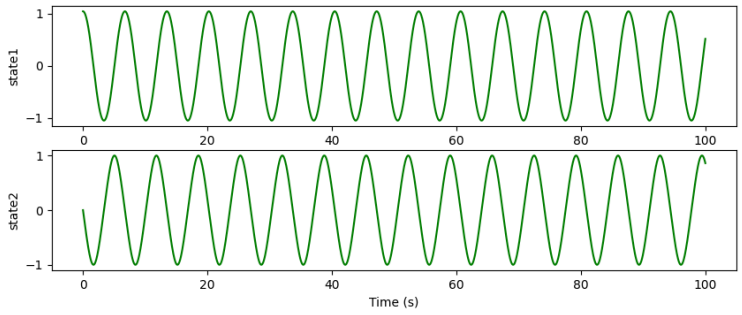


Time simulation 100s , step 0.01 , initial state = released from 60 deg at 0 speed

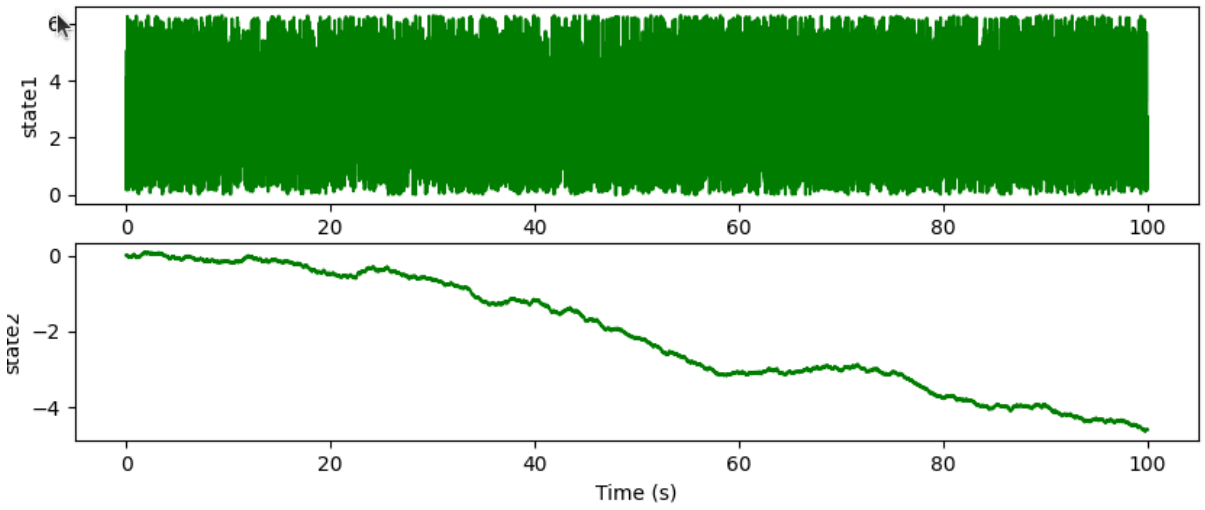
Forward



Symplectic



Effect of restricting theta between 0 -2pi



Analysis of effect of step size and time simulation, and method used remains same

Fundamental difference between 2 systems

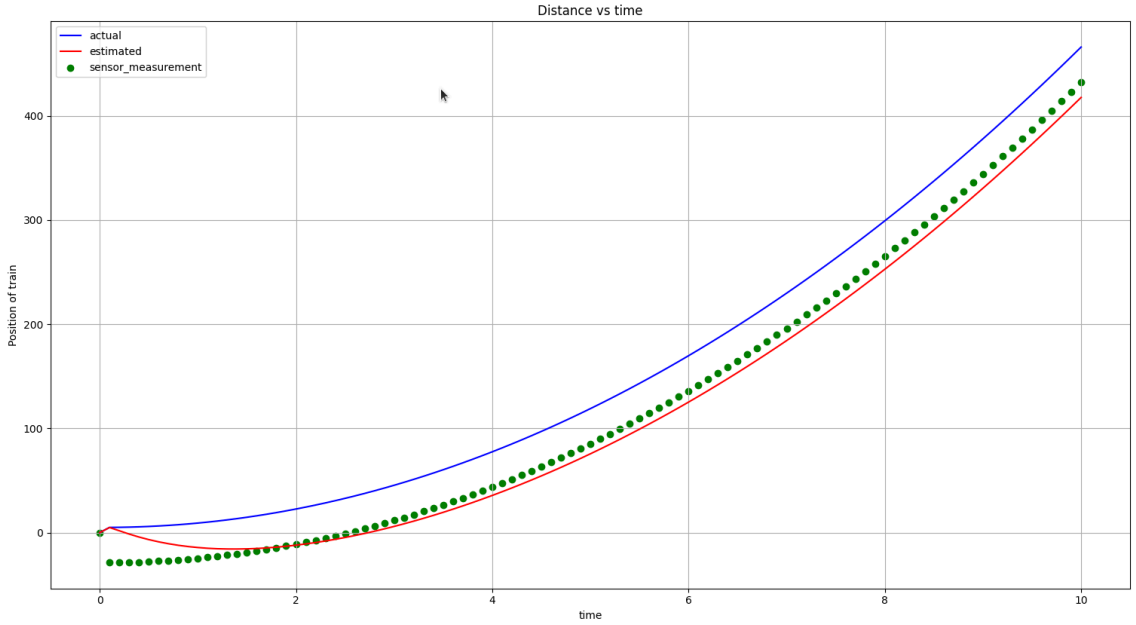
The pendulum's motion is periodic, oscillating between 0 and 2π, depending on the representation of the angle. Ensuring that numerical integration methods preserve this periodicity can be challenging, especially when using fixed-step methods**.**

This is clearly seen when we try to restrict theta and system just crashes

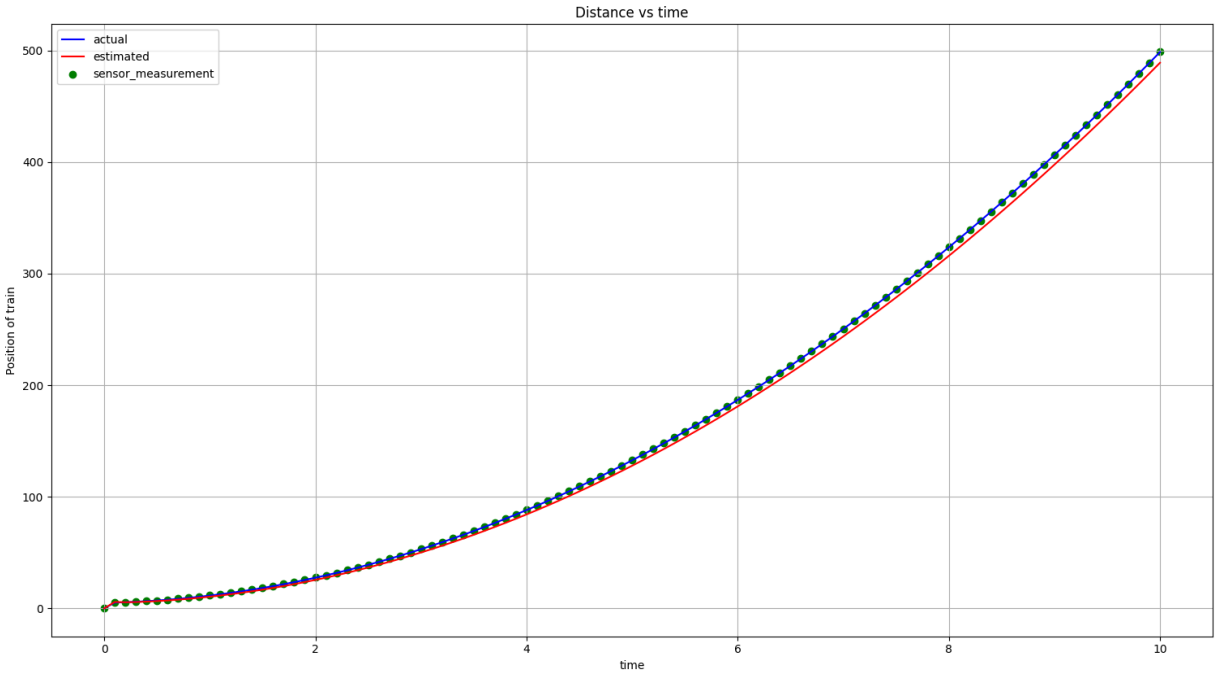
As 2pi + delta theta is very close to delta theta because of circular space

While spring mass system is defined in a linear space and this periodicity issue is not encountered

ANS 3) kalman filter implementation of ramsay paper : refer code for implementation details , commented out



As seen when introduced significant gaussian noise between measurements and desired pose, the estimate tends to follow the one more close to ground truth which validates prediction and update function working just fine



Part 4) This problem is tackled by output injection
In which a simplest state estimator is defined which mimics the state dynamics plus there some additional term proportional to output - measurement of output
L can be treated as tuning parameter and adjusted to make net state transition matrix schur stable i.e all real parts of eigen values are less than one

Mathematical formulation is as follows

To study the performance of this state estimator, we define the *state estimation error*

$$e := \hat{x} - x.$$

Taking derivatives, we conclude that

$$\dot{e} = A\hat{x} + Bu - (A\hat{x} + Bu) = Ae.$$

Therefore, when A is a stability matrix, the open-loop state estimator (16.6) results in an error that converges to zero exponentially fast, *for every input signal u*.

When the matrix A is not a stability matrix, it is still possible to construct an asymptotically correct state estimate, but to achieve this we need a closed-loop estimator of the form

$$\dot{\hat{x}} = A\hat{x} + Bu - L(\hat{y} - y), \qquad \hat{y} = C\hat{x} + Du, \tag{16.7}$$

for some *output injection matrix gain* $L \in \mathbb{R}^{n\times m}$. Now the state estimation error evolves according to

$$\dot{e} = A\hat{x} + Bu - L(\hat{y} - y) - (A\hat{x} + Bu) = (A - LC)e.$$

Theorem 16.7. *Consider the closed-loop state estimator* (16.7). *If the output injection matrix gain* $L \in \mathbb{R}^{n\times m}$ *makes* $A - LC$ *a stability matrix, then the state estimation error e converges to zero exponentially fast, for every input signal u.* □

Reference:
Implicit euler stability :
https://math.libretexts.org/Bookshelves/Differential_Equations/Numerically_Solving_Ordinary_Differential_Equations_(Brorson)/01%3A_Chapters/1.03%3A_Backward_Euler_method
linear_systems_theory_Joao_P_Hespanha part 4 directly taken