Applied Computational Science I

Lab: Symplectic Integrators

Anand Kamble

amk23j@fsu.edu

1st December 2023

Task 1

$$x''(t) + x(t) = 0$$

$$x(t) = x(0)\cos t + v(0)\sin t$$

Substituting $t=0$,
we get,

$x(0) = x(0)$ $\therefore$ Initial condition is satisfied.

Since, $v(0) = x'(0)$,

$$v(0) = -x(0)\sin\cancel{x}(0) + v(0)\cos(0)$$

$\therefore v(0) = v(0)$

$\therefore v(0) = x'(0)$ is satisfied

from eqⁿ ①,

$$x''(t) + x(t) = 0$$

$$-[\cancel{x(0)\cos(t)} + \cancel{v(0)\sin t}] + \cancel{x(0)\cos(t)} + \cancel{v(0)\sin t} = 0$$

$$= 0$$

$x(t)$ satisfies the initial & differential eqⁿ.

Task 2

$$E(t) = \frac{1}{2}x^2(t) + \frac{1}{2}v^2(t)$$

Using $x(t)$ and $v(t)$ from previous question,

$$E(t) = \frac{1}{2}\left(x(0)\cos(t) + v(0)\sin(t)\right)^2$$
$$+ \frac{1}{2}\left(-x(0)\sin(t) + v(0)\cos(t)\right)^2$$
$$= \frac{1}{2}\left[x(0)^2\cos^2(t) + 2x(0)v(0)\cos(t)\sin(t) + v(0)^2\sin^2(t)\right]$$
$$+ \frac{1}{2}\left[x(0)^2\sin^2(t) - 2x(0)v(0)\cos(t)\sin(t) + v(0)^2\cos^2(t)\right]$$
$$= \frac{1}{2}\; x(0)^2\left[\cos^2(t) + \sin^2(t)\right] + v(0)^2\left[\sin^2(t) + \cos^2(t)\right]$$
$$= \frac{1}{2}\left[x(0)^2 + v(0)^2\right]$$

Substituting, $x(0) = 1$ and $v(0) = 0$

$$E(t) = \frac{1}{2}(1 + 0) = \frac{1}{2}$$

Task 3

1. Forward Euler:

$$G_{FE} = \begin{bmatrix} 1 & h \\ -h & 1 \end{bmatrix}$$

2. Backward Euler:

$$G_{BE} = \begin{bmatrix} 1 & h \\ -h & 1 \end{bmatrix}$$

3. Implicit Trapezoidal:

$$G_{IT} = \begin{bmatrix} 1 - \frac{h}{2} & \frac{h}{2} \\ -\frac{h}{2} & 1 - \frac{h}{2} \end{bmatrix}$$

4. Leap-Frog:

$$G_{LF} = \begin{bmatrix} 1 & h \\ -h & 1 \end{bmatrix}$$

Programming Tasks

1.

In [23]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [5]:
```
def integrate(method, h, T):
    num_steps = int((T[1] - T[0]) / h) + 1
    time_points = np.linspace(T[0], T[1], num_steps)
    x = np.zeros(num_steps)
    v = np.zeros(num_steps)
    energy = np.zeros(num_steps)
    x[0] = 1
    for i in range(num_steps - 1):
        x[i + 1], v[i + 1] = method(x[i], v[i], h)
        energy[i] = 0.5 * (x[i]**2 + v[i]**2)

    return time_points, x, v, energy
```

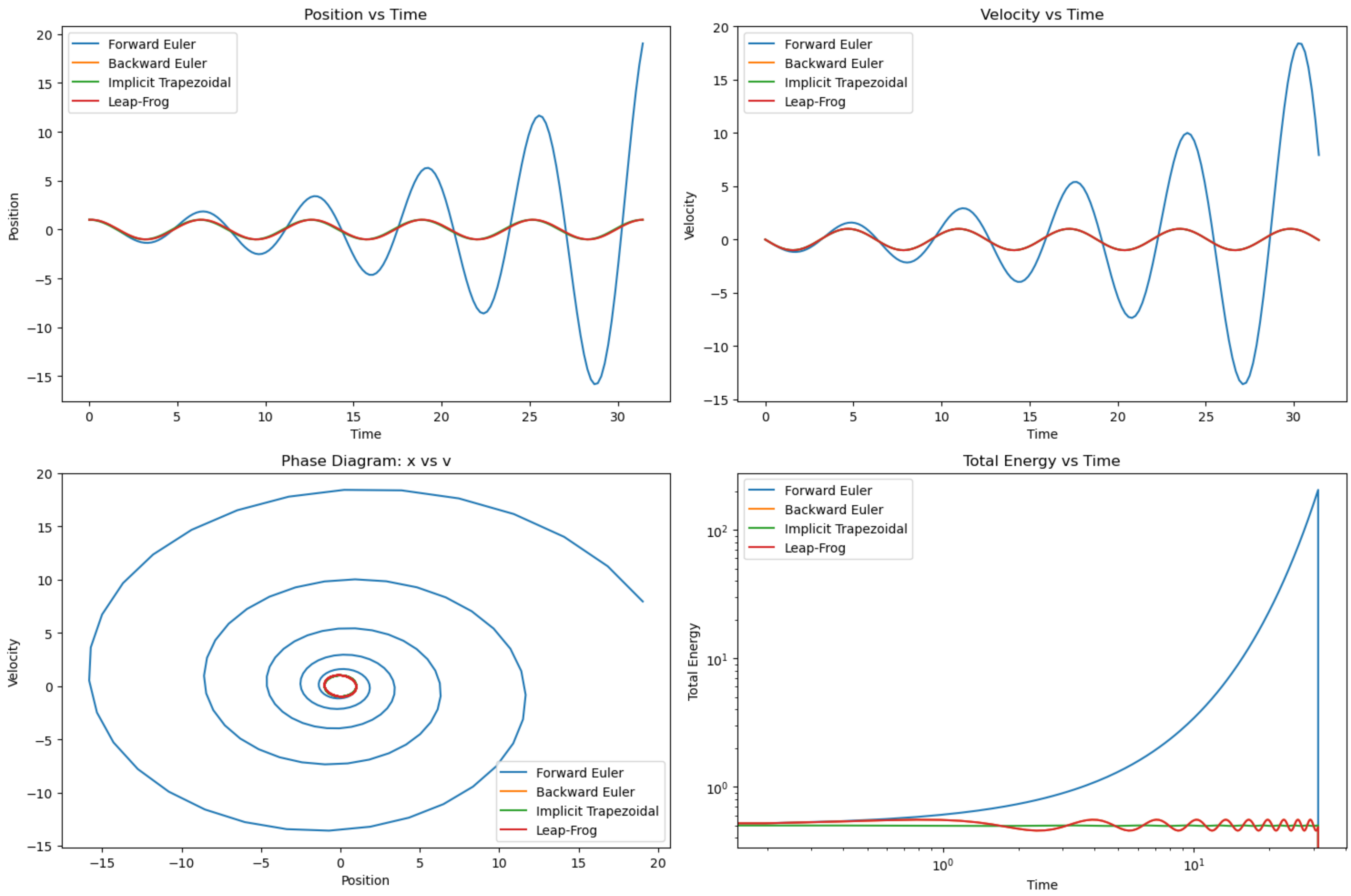
In [6]:
```
def forward_euler(x, v, h):
    x_next = x + h * v
    v_next = v - h * x
    return x_next, v_next
```

In [7]:
```
def backward_euler(x, v, h):
    x_next = x + h * v
    v_next = v - h * x_next
    return x_next, v_next
```

In [8]:
```
def implicit_trapezoidal(x, v, h):
    x_next = x + 0.5 * h * (v + v - h * x)
    v_next = v - 0.5 * h * (x + x_next)
    return x_next, v_next
```

In [9]:
```
def leap_frog(x, v, h):
    x_next = x + h * v
    v_next = v - h * x_next
    return x_next, v_next
```

In [22]:
```
h = 2 * np.pi / 32
T = [0, 10 * np.pi]

time_fe, x_fe, v_fe, energy_fe = integrate(forward_euler, h, T)
time_be, x_be, v_be, energy_be = integrate(backward_euler, h, T)
time_it, x_it, v_it, energy_it = integrate(implicit_trapezoidal, h, T)
time_lf, x_lf, v_lf, energy_lf = integrate(leap_frog, h, T)

plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
plt.plot(time_fe, x_fe, label='Forward Euler')
plt.plot(time_be, x_be, label='Backward Euler')
plt.plot(time_it, x_it, label='Implicit Trapezoidal')
plt.plot(time_lf, x_lf, label='Leap-Frog')
plt.title('Position vs Time')
plt.xlabel('Time')
plt.ylabel('Position')
plt.legend()

plt.subplot(2, 2, 2)
plt.plot(time_fe, v_fe, label='Forward Euler')
plt.plot(time_be, v_be, label='Backward Euler')
plt.plot(time_it, v_it, label='Implicit Trapezoidal')
plt.plot(time_lf, v_lf, label='Leap-Frog')
plt.title('Velocity vs Time')
plt.xlabel('Time')
plt.ylabel('Velocity')
plt.legend()

plt.subplot(2, 2, 3)
plt.plot(x_fe, v_fe, label='Forward Euler')
plt.plot(x_be, v_be, label='Backward Euler')
plt.plot(x_it, v_it, label='Implicit Trapezoidal')
plt.plot(x_lf, v_lf, label='Leap-Frog')
plt.title('Phase Diagram: x vs v')
plt.xlabel('Position')
plt.ylabel('Velocity')
plt.legend()

plt.subplot(2, 2, 4)
plt.loglog(time_fe, energy_fe, label='Forward Euler')
plt.loglog(time_be, energy_be, label='Backward Euler')
plt.loglog(time_it, energy_it, label='Implicit Trapezoidal')
plt.loglog(time_lf, energy_lf, label='Leap-Frog')
plt.title('Total Energy vs Time')
plt.xlabel('Time')
plt.ylabel('Total Energy')
plt.legend()

plt.tight_layout()
plt.show()
```



2.

Leap-frog and Implicit Trapezoidal are the Symplectic Methods and Forward Euler and Backward Euler are not Symplectic Methods.

3.

Eigen Values

Forward Euler - $\lambda_1 = 1 + h$ and $\lambda_2 = 1 - h$

Backward Euler - $\lambda_1 = 1 + h$ and $\lambda_2 = 1 - h$

Implicit Trapezoidal - $\lambda_1 = 1$ and $\lambda_2 = 1 - h$

Leap-Frog - $\lambda_1 = 1 + h$ and $\lambda_2 = 1 - h$

Forward Euler and Backward Euler:

The eigenvalues have magnitudes greater than 1 for certain values of h, indicating potential instability and growth of the solution over time.

Implicit Trapezoidal:

Both eigenvalues have magnitudes less than or equal to 1 for all values of h, indicating stability. This is consistent with the symplectic nature of the method.

Leap-Frog:

Similar to Forward and Backward Euler, the eigenvalues have magnitudes greater than 1 for certain values of h, indicating potential instability.

4.

The Implicit Trapezoidal and Leap-Frog are the two symplectic methods. The Implicit Trapezoidal method is time reversible and will benefit for scenarios where time reversibility is required. While the Leap-Frog method is simple and computationally efficient.