Set 3: Numerical methods for ODEs

Aritra Biswas

1 Implementation of explicit Euler Method

We have:

$$F = m\frac{d^2x}{dt^2} = -kx$$
$$\frac{d^2x}{dt^2} = -\frac{k}{m}x$$

Under our simplification that $\frac{k}{m} = 1$:

$$\frac{d^2x}{dt^2} = -x$$

We use the explicit Euler method to approximate a solution to this ODE:

$$x(t+h) \approx x(t) + h\frac{dx}{dt}$$
$$x_{i+1} = x_i + hv_i.$$
$$v(t+h) \approx v(t) + h\frac{dv}{dv}$$
$$v_{i+1} = v_i - hx_i.$$

```
def explicit_Euler(x_0, v_0, h, s, plot=False, plotname='explicit.eps'):
    '''Given a starting position x_0 and velocity v_0, plots the position x
    and velocity v of a spring over time using the explicit Euler method
    with a step size h from t = 0 to t = s.'''

    x = np.array([x_0])
    v = np.array([v_0])
    t = np.arange(0, s, h)

    for i in xrange(len(t) - 1):
        x = np.append(x, x[i] + h*v[i])
        v = np.append(v, v[i] - h*x[i])

    if plot:
        plotter.figure(figsize=(10, 4))
        plotter.plot(t, x, color='blue', label='x')
        plotter.plot(t, v, color='red', label='v')
        plotter.xlabel('t')
        plotter.legend()
        plotter.savefig(plotname, format='eps', bbox_inches='tight', pad_inches=0.1)

    return (t, x, v)
```

With initial conditions $x_0 = 0$, $v_0 = 5$, and a step size $h = 0.05$ with a stop time $s = 50$, the above subroutine produces the numerical solutions for $x(t)$ and $v(t)$ shown in figure 1.

2 Comparison with analytic solution

We show that a generic sinusoidal function is a solution to the given ODE.

$$\begin{aligned} x &= A\sin t + B\cos t \\ v = \frac{dx}{dt} &= A\cos t - B\sin t \\ \frac{d^2x}{dt^2} &= -A\sin t - B\cos t = -x \end{aligned}$$

Using our initial values $x = x_0$ and $v = v_0$ at $t = 0$:

$$\begin{aligned} x_0 &= A\sin 0 + B\cos 0 \\ &= B. \\ v_0 &= A\cos 0 - B\sin 0 \\ &= A. \end{aligned}$$

Thus, the complete analytical solution is:

$$\begin{aligned} x &= v_0 \sin t + x_0 \cos t. \\ v &= v_0 \cos t - x_0 \sin t. \end{aligned}$$

Figure 2 shows the global trunctation error, the difference between the explicit Euler method used in section 1 and the analytic method derived here. The error is computed in the `global_error` function:

explicit-eps-converted-to.pdf

Figure 1: Numerical solutions for $x(t)$ and $v(t)$ as calculated by the explicit Euler method.

```
def global_error(x_0, v_0, h, s, method_a, method_b, plot=False, plotname=None):
    '''Generates arrays of the global error method_b - method_a for x and v.
    Returns maximum error in x and v.'''

    (t_a, x_a, v_a) = method_a(x_0, v_0, h, s)
    (t_b, x_b, v_b) = method_b(x_0, v_0, h, s)

    x_diff = x_b - x_a
    v_diff = v_b - v_a

    assert np.array_equal(t_a, t_b)
    t = t_a

    if plotname is None:
        plotname = '%s_error.eps' % method_a

    if plot:
        plotter.figure(figsize=(10, 4))
        plotter.plot(t, x_diff, color='blue', label='x error')
        plotter.plot(t, v_diff, color='red', label='v error')
        plotter.xlabel('t')
        plotter.legend()
        plotter.savefig(plotname, format='eps', bbox_inches='tight', pad_inches=0.1)

    max_x_error = np.max(np.absolute(x_diff))
    max_v_error = np.max(np.absolute(v_diff))

    return (max_x_error, max_v_error)
```

3 Relationship between truncation error and step size h

We examine the effect of changing the step size h on the truncation error with the following subroutine error_vs_h, which plots the maximum truncation error for various h ranging from a given h_0 to $\frac{h_0}{16}$.

```
def error_vs_h(x_0, v_0, h_0, s, method_a, method_b, plot=False, plotname=None):
    '''Compares global error between method_a and method_b for x and v
    for h ranging from h_0 to h_0/16.'''

    coeff = np.logspace(0, 4, base=2)
    h = h_0 / coeff

    # allow global_error function to take an array of h and s
    global_error_vec = np.vectorize(global_error, excluded=['x_0', 'v_0', \
    's', 'method_a', 'method_b', 'plot'])

    errors = global_error_vec(x_0, v_0, h, s, method_a, method_b)
    x_errors = errors[0]
    v_errors = errors[1]

    if plotname is None:
        plotname = '%s_error_v_h.eps' % method_a
```

explicit_error-eps-converted-to.pdf

Figure 2: Truncation error: difference between the analytic solution and the solution calculated from the explicit Euler method.

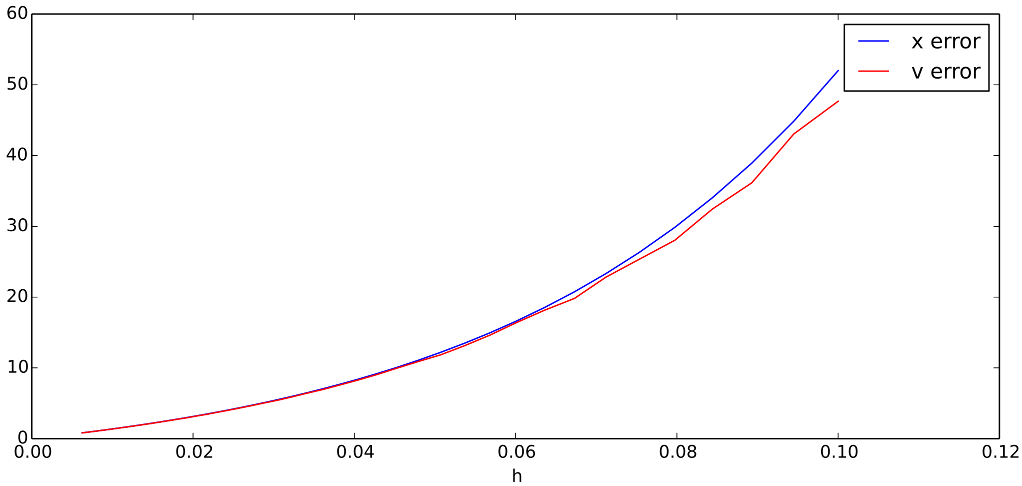


Figure 3: Truncation error vs. h for large values of h.

```
if plot:
    plotter.figure(figsize=(10, 4))
    plotter.plot(h, x_errors, color='blue', label='x error')
    plotter.plot(h, v_errors, color='red', label='v error')
    plotter.xlabel('h')
    plotter.legend()
    plotter.savefig(plotname, format='eps', bbox_inches='tight', pad_inches=0.1)
```

Figure 3 shows an exponential trend for various, somewhat large h ($h_0 = 0.1$). If we restrict the domain to small h, we expect to see a small portion of this exponential curve, which will appear linear. As expected, figure 4 shows a roughly linear trend when h is small ($h_0 = 0.01$).

4 Energy conservation in explicit Euler method

Given $E = x^2 + v^2$, we can plot energy as a function of time with our numerical solutions for x and v. Figure 5 shows that the numerical solution does not exhibit energy conservation, instead showing energy increasing over time.

Visually, the trend appears to be quadratic or exponential. A quadratic trend is expected since the error trend in x and v (figure 2), while sinusoidal, appears to be bound above and below by linear curves.

5 Implicit Euler method

The implicit Euler method uses values at $t + h$ to updates values at $t + h$, as opposed to the explicit Euler method which used values at t to updates values at $t + h$.

$$x_{i+1} = x_i + hv_{i+1}.$$
$$v_{i+1} = v_i - hx_{i+1}.$$

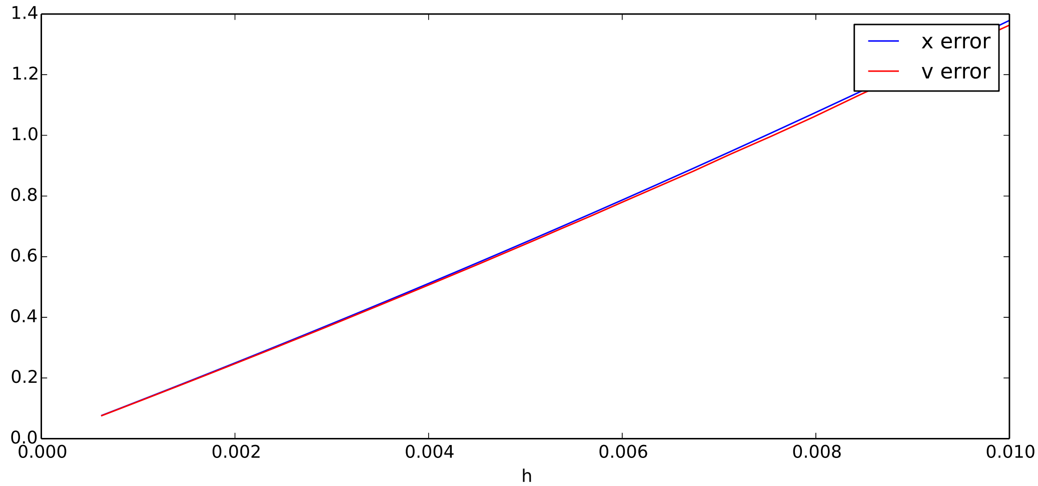


Figure 4: Truncation error vs. h for small values of h.

We obtain $x(t+h)$ and $v(t+h)$ in terms of $x(t)$ and $v(t)$ with some algebraic manipulation. As demonstrated, the two above equations are equivalent to the linear system:

$$\begin{pmatrix} 1 & -h \\ h & 1 \end{pmatrix} \begin{pmatrix} x_{i+1} \\ v_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ v_i \end{pmatrix}$$

Using *Mathematica*, we find that:

$$x_{i+1} = \frac{x_i + hv_i}{h^2+1}.$$
$$v_{i+1} = \frac{v_i - hx_i}{h^2+1}.$$

Using this method and the same conditions as in figure 1, we can generate new numerical solutions for x and v in figure 6. We also show the global error between the implicit and analytic solutions in figure 7.

With the new numerical solutions, we also generate a new solution for $E(t)$, plotted in figure 8.

6 Phase-space geometry and the non-symplectic Euler methods

Since $E = x^2 + v^2$, and energy should be conserved in an ideal spring, plotting the spring's trajectory in the xv-plane should yield a circle of radius $\sqrt{E}$. Of course, we have already seen that the explicit and implicit Euler methods do not conserve energy, so their phase-space trajectories will be spirals rather than closed circles, as shows in figure 9. These plots are generated with the same conditions as before: $x_0 = 0, v_0 = 5, h = 0.05, s = 50$.

explicit_E-eps-converted-to.pdf

Figure 5: $E(t)$ as determined from explicit-Euler-method numerical solutions to $x(t)$ and $v(t)$.

implicit-eps-converted-to.pdf

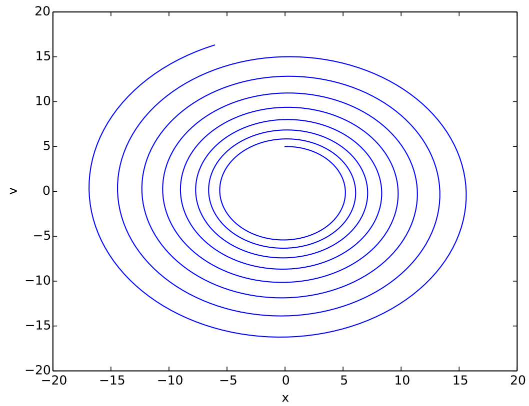
Figure 6: Numerical solutions for $x(t)$ and $v(t)$ as calculated by the implicit Euler method.

implicit_error-eps-converted-to.pdf

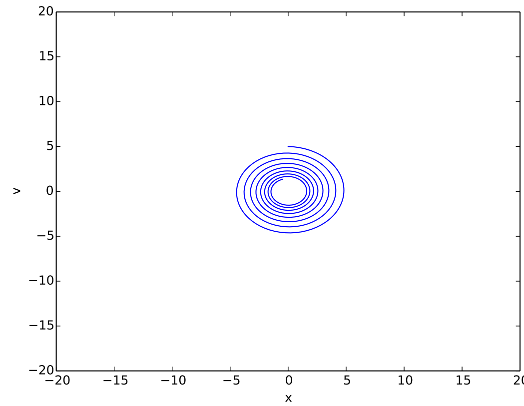
Figure 7: Truncation error for the implicit Euler method.

implicit_E-eps-converted-to.pdf

Figure 8: $E(t)$ as determined from implicit-Euler-method numerical solutions to $x(t)$ and $v(t)$.



(a) explicit Euler method



(b) implicit Euler method

Figure 9: Phase-space trajectories for explicit and implicit Euler methods.

7 Implementation of symplectic Euler method

To design a numerical method that conserves area in phase space (and thus conserves energy), we combine the explicit and implicit Euler methods:

$$\begin{aligned} x_{i+1} &= x_i + hv_i. \\ v_{i+1} &= v_i - hx_{i+1} \\ &= v_i - h(x_i + hv_i) \\ &= v_i - hx_i - h^2 v_i. \end{aligned}$$

Figure 10 shows the numerical solutions for $x(t)$ and $v(t)$ obtained through this method. We immediately notice that, as expected from the analytic solution to a spring's motion, the amplitude does not change over time in our time frame ($t = 0$ to $t = 50$).

Figure 11 shows the phase-space trajectory, which appears closed in our time frame.

8 Energy evolution of symplectic Euler method

Though the phase-space trajectory (figure 11) suggests that energy is conserved with the symplectic Euler method, the calculated energy undergoes small oscillations around the expected constant value of 25.0 (see figure 12).

9 Long-term error in symplectic Euler method

In addition to the small oscillations in energy, the symplectic Euler method exhibits another imperfection: the calculated solutions for $x(t)$ and $v(t)$ lag behind the analytic solutions. Figure 13 shows this phenomenon. Note that the figure is plotted from $t = 4950$ to $t = 5000$; the error is not appreciable at small times.

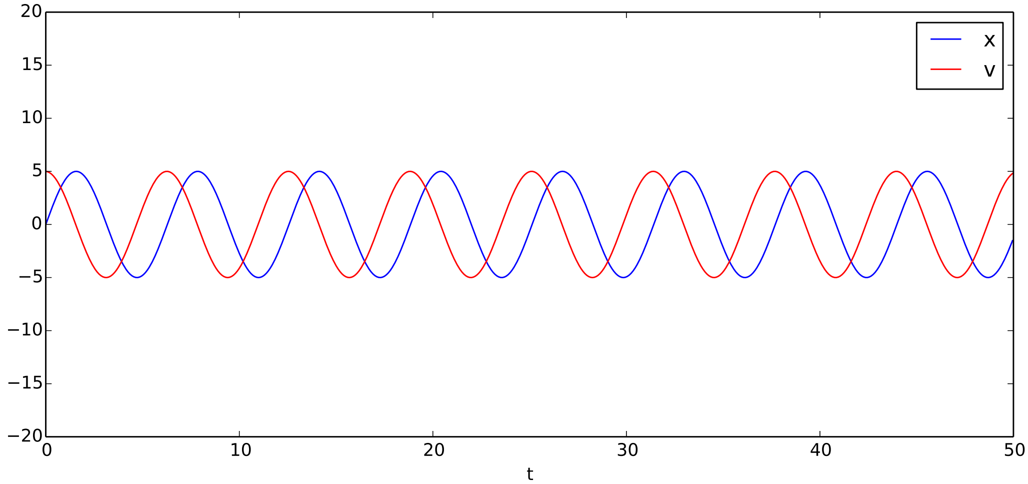


Figure 10: Numerical solutions for $x(t)$ and $v(t)$ as calculated by the implicit Euler method.

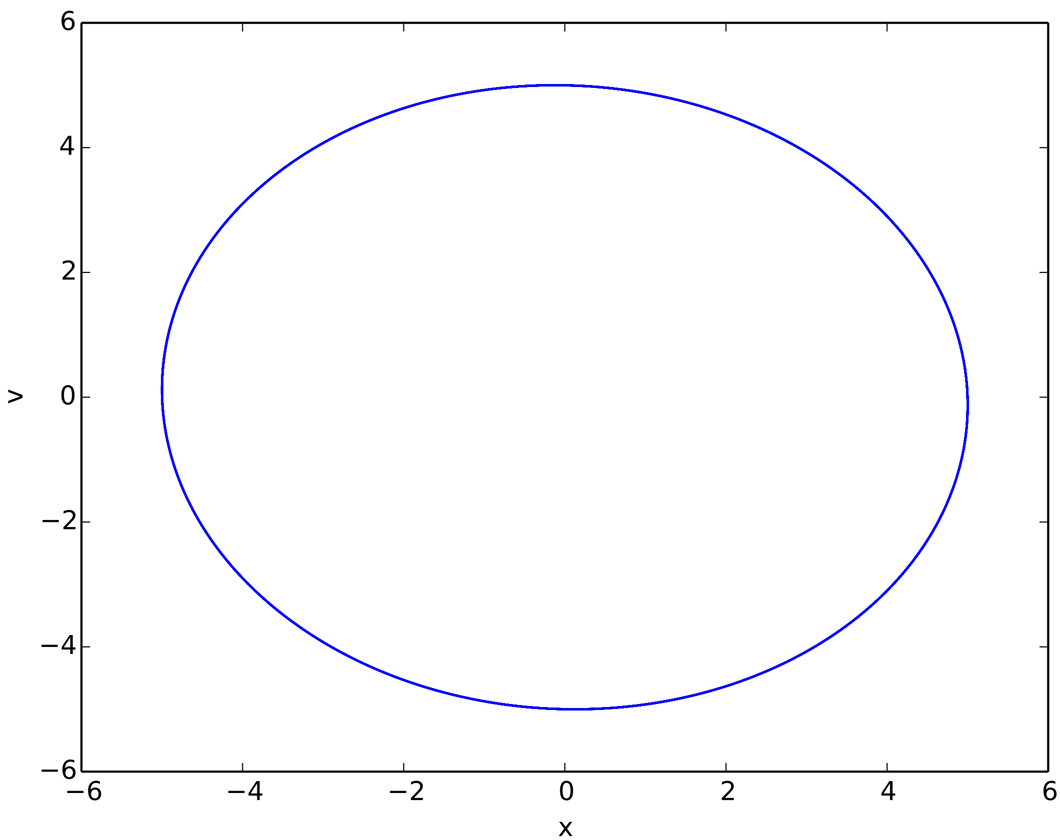


Figure 11: Phase-space trajectory for symplectic Euler method.

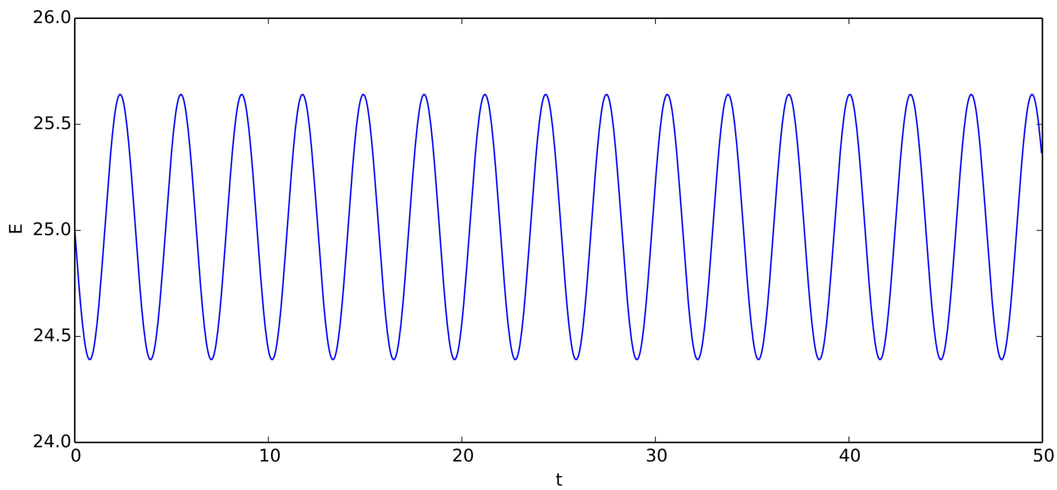


Figure 12: $E(t)$ for symplectic Euler method.

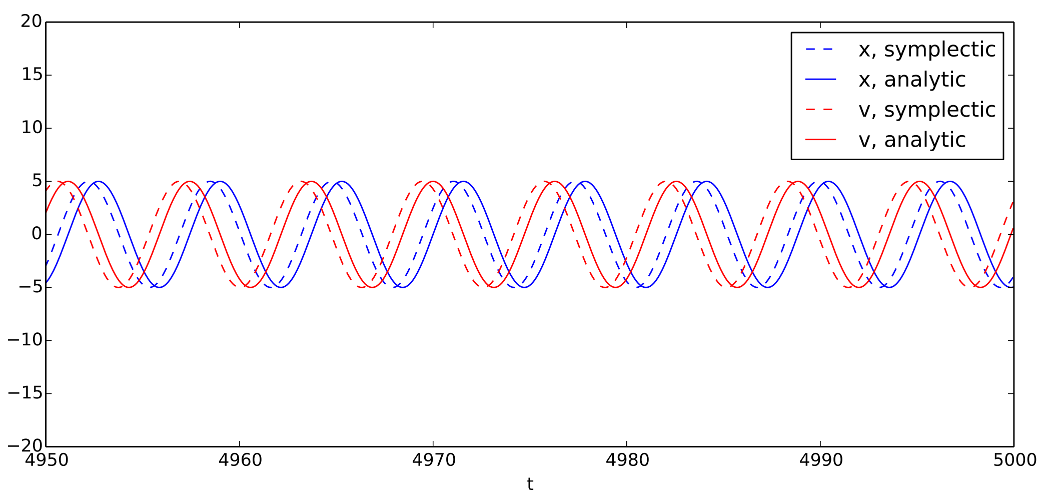


Figure 13: Comparison of numerical and analytic solutions for $x(t)$ and $v(t)$, showing that at large t, the symplectic solution exhibits a noticeable lag.