UNIVERSITY OF BIRMINGHAM

Computational Modelling of Physical Systems
Worksheet 3

Samuel Delacruz
sjd054@bham.ac.uk
1090154

Issue: 3
Date: November 18, 2013

School of Physics and Astronomy
University of Birmingham
Birmingham, B15 2TT

Contents

List of Figures

List of Tables

1 Introduction

In this assignment, methods of integrating ordinary differential equations will be explored.

Firstly, Euler's method will be explained and applied to a basic problem. The errors associated with the method will be discussed.

Following this, the Runge-Kutta method will be described, and used to solve a problem, using two different levels of precision in the second-order algorithm. The respective results will be compared, and errors discussed.

2 Euler's Method

2.1 Theory

Euler's method is a basic, but inefficient method of numerical approximation of differential equations.
It is described as such, when considering a general first order differential equation,

$$\frac{dy}{dx}(x) = f(x, y) \tag{1}$$

Can also be written as,

$$\frac{dy}{dx}(x) = \lim_{h \to 0} \frac{y(x+h) - y(x)}{h} \tag{2}$$

This leads to the definitions of $y_0 = y(x_0)$ and $y_1 = y(x_0 + h)$.
Expanding y_1 as a series gives,

$$y_1 = y(x_0) + y'(x_0)h + \frac{1}{2}y''(x_0)h^2 + ... \tag{3}$$

Omitting powers of h^3 and above. Referring to equation (1), we can see that this may be re-written as,

$$y_1 = y(x_0) + hf(x_0, y_0) + O(h^2) \tag{4}$$

So we can see that the error in using one step of Euler's method is of $O(h^2)$.

However, for a good numerical approximation of the solution to a differential equation, we should aim for a set of values for many values of our dependent variable.

Hence, if we apply the method n times in succession, we will have,

$$y_n = y(x_0) + nhf(x_0, y_0) + O(nh^2) \tag{5}$$

But, $n \propto h^{-1}$, so the $O(nh^2)$ term becomes $O(h)$. This means that the error expected from implementations of Euler's method is of $O(h)$, and will be investigated.

2.2 Procedure

To demonstrate Euler's method, a C++ program can be written to solve the differential equation

$$\frac{dy}{dt} = 1 + y^2, \tag{6}$$

with initial condition $y(0) = 0$, for $y(\pi/4)$.

This equation has the solution

$$y(t) = \tan(t) \tag{7}$$

So $y(\pi/4) = \mathbf{1}$. This value will be used as a reference value, to compare to computed values at different values of h.

A program, named "euler.cpp" has been written to solve the described problem, and may be compiled using G++, and run with the terminal command:
`./euler.o [n] [filename]`
Where [n] is the maximum number of intervals to use, and [filename] is the name of the file to save results to. **Warning: ensure that [filename] is unique, or risk overwriting previous results**.

2.3 Results

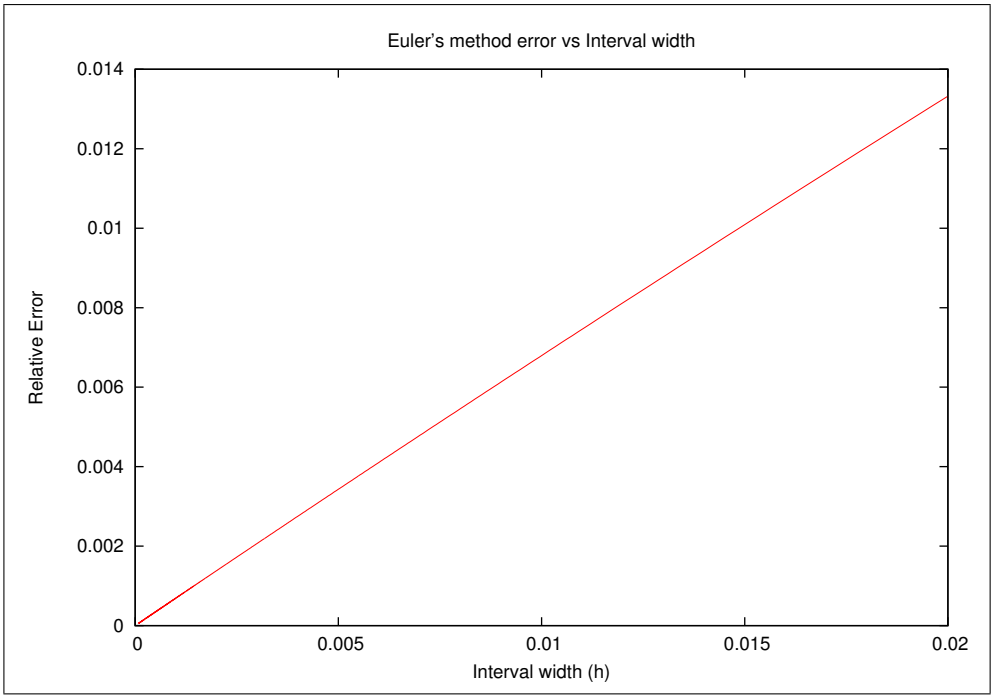

Figure 1: Plot of Euler's method relative error vs interval width

Figure (1) shows the evolution of errors in the program written as h increases. As expected, the error in the approximate solution is $O(h)$, as demonstrated by the straight line on a linear scale graph.
Even when using $n = 10000$, the value for $y(\pi/4)$ is calculated as $9.999455690423819 \cdot 10^{-1}$, which has a relative error $\epsilon = 5.443095761814565 \cdot 10^{-5}$. This confirms that relative error is $O(h)$, or alternatively $O(n^{-1})$.

Clearly, Euler's method is inefficient at solving this particular problem, and more efficient algorithms must be explored which have faster convergence.

3 The Runge-Kutta method

3.1 Theory

The Runge-Kutta method improves upon Euler's method by averaging the slope of the function between x and $x + h$. In the second order Runge Kutta method, the value of the slope at the midpoint of the interval is used to reach the end of the interval from the beginning. This results in a closer fitting to the curve, generating an overall relative error $O(h^3)$. This method is therefore far more efficient than the basic Euler's method.

For solving systems of first order differential equations, using the second-order Runge-Kutta method, the following procedure applies

$$\mathbf{k}_1 = h\mathbf{f}(x_n, \mathbf{y}_n) \tag{8}$$
$$\mathbf{k}_2 = h\mathbf{f}(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{\mathbf{k}_1}{2}) \tag{9}$$
$$\mathbf{y}_{n+1} = \mathbf{y}_n + \mathbf{k}_2 + O(h^3) \tag{10}$$

To put this into practice, another problem shall be solved. In this case, the Simple Harmonic Oscillator will be considered, with unit mass and potential

$$V(x) = \frac{1}{2}x^2 \tag{11}$$

The Simple Harmonic Oscillator with unit mass can be described by

$$F = \frac{d^2x}{dt^2} \tag{12}$$
$$= -x \tag{13}$$

Which may be decomposed into a system of two coupled first order equations

$$\frac{dx}{dt} = p \tag{14}$$
$$\frac{dp}{dt} = -x \tag{15}$$

These equations can be solved together using the Runge-Kutta method as described previously, and they have an analytical solution of

$$x(t) = x_0 \cos(t) \tag{16}$$

3.2 Procedure

To demonstrate the second order Runge-Kutta method, a C++ program can be written to solve the coupled differential equations previously discussed,

$$\frac{dx}{dt} = p \tag{17}$$
$$\frac{dp}{dt} = -x \tag{18}$$

with initial condition $x(0) = 1$, and $p(0) = 0$.

Since energy is conserved, it is expected that with this setup, $E = \frac{p^2}{2} + \frac{x^2}{2} = 1$, and any deviations from this are indicative of errors arising from the Runge-Kutta method algorithm.

A program, named "sho-rk2.cpp" has been written to solve the described problem, and may be compiled using G++, and run with the terminal command:

```
./sho-rk2.o [dt] [n] [filename]
```

Where [dt] is the size of timestep to use, [n] is the maximum number of intervals to use, and [filename] is the name of the file to save results to. **Warning: ensure that [filename] is unique, or risk overwriting previous results**.

The program was run with parameters

$$dt = 0.5, n = 100 \tag{19}$$
$$dt = 0.02, n = 2500 \tag{20}$$

Their respective behaviours are discussed in the next subsection.

3.3 Results

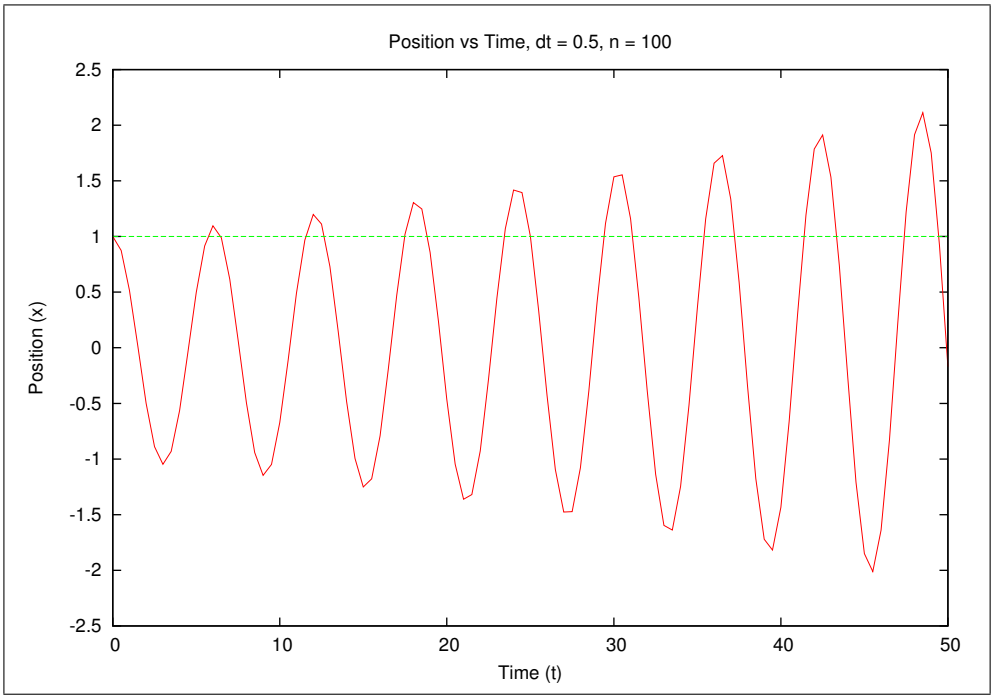

Figure 2: Second order Runge-Kutta method. Position vs. Time for $dt = 0.5, n = 100$

Figure (2) shows the output graph of position vs time of the oscillator, using parameters $dt = 0.5$ and $n = 100$. This shows the oscillator's behaviour over 50 units of time. As shown by the graph, the amplitude of oscillation appears to be increasing with time. This is clearly not physically possible, as this would indicate an increase in energy in the system, which must remain constant.

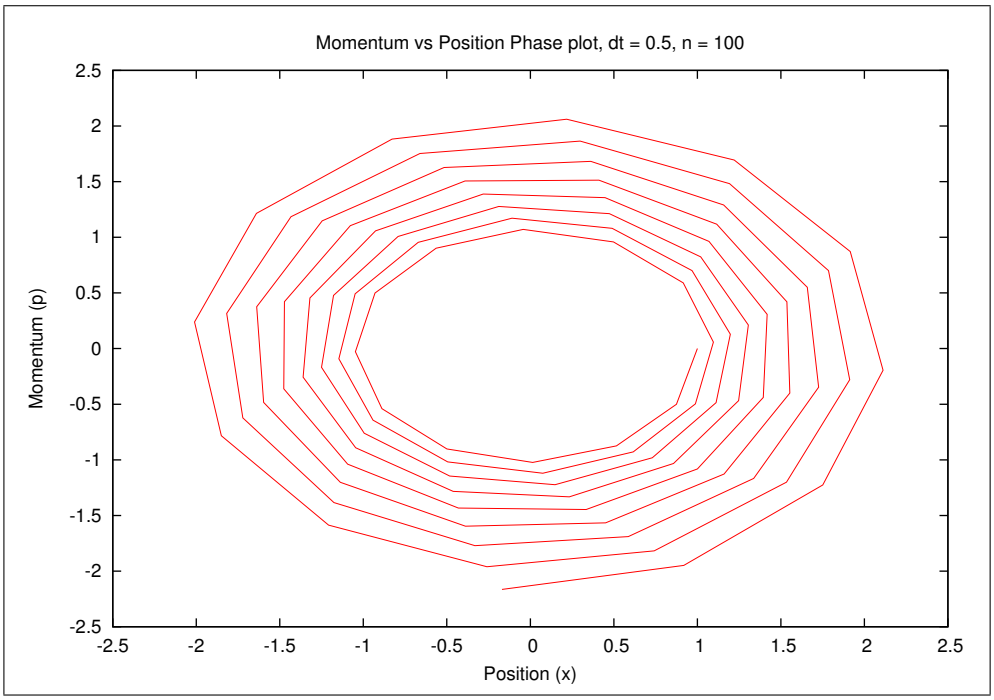

Figure 3: Second order Runge-Kutta method. Phase plot of Momentum vs. Position for $dt = 0.5, n = 100$

Figure (3) shows the output graph of momentum vs. position in a phase space plot. The radius of the spiral, $|r|$, is $\sqrt{E}$, since $E = \frac{p^2}{2} + \frac{x^2}{2}$. This should remain constant, however it is shown to spiral outwards, indicating that momentum and position is being amplified over time.

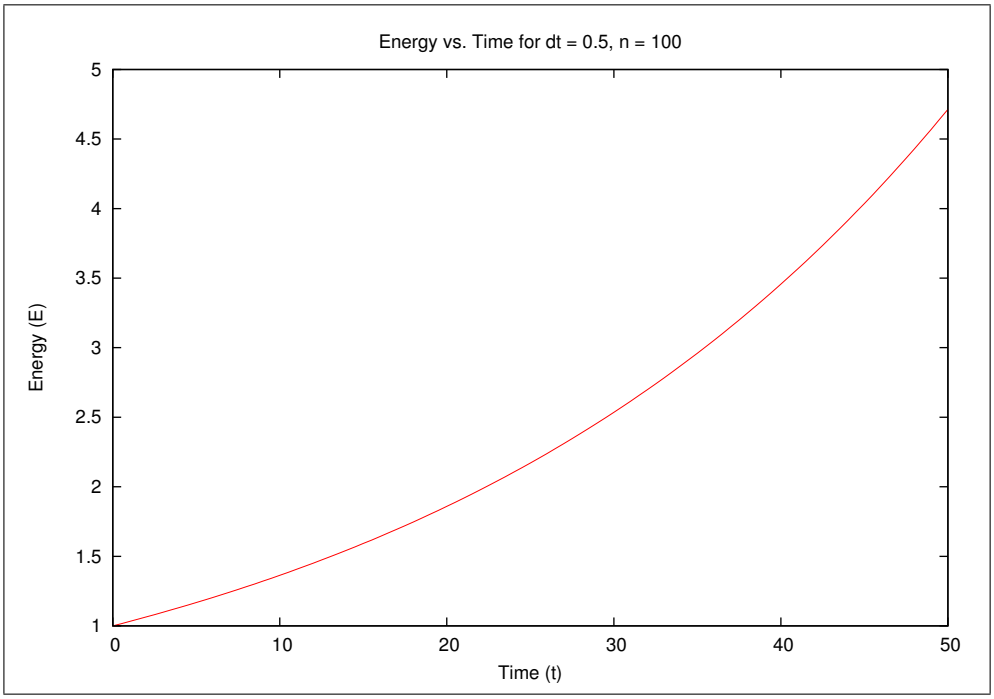

Figure 4: Second order Runge-Kutta method. Phase plot of Momentum vs. Position for $dt = 0.5, n = 100$

Figure (4) shows the increase of energy over time, the rate at which energy increases is also increasing over

time.

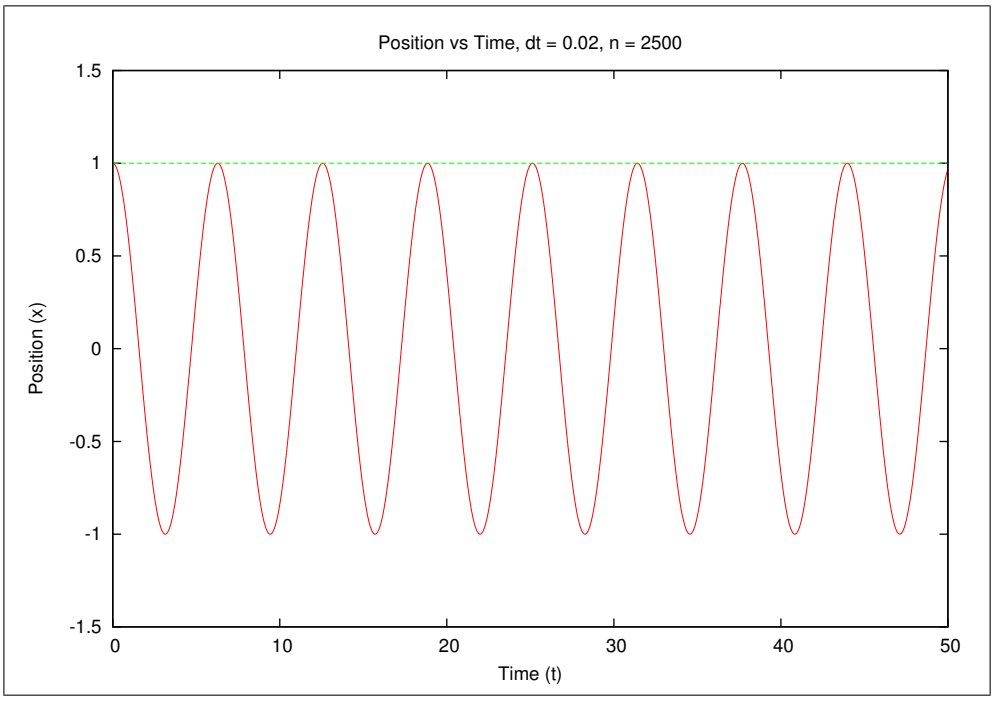

Figure 5: Second order Runge-Kutta method. Position vs. Time for $dt = 0.02, n = 2500$

Figure (5) shows the output graph of position vs time of the oscillator, using parameters $dt = 0.02$ and $n = 2500$. This shows the oscillator's behaviour over 50 units of time. As shown by the graph, the amplitude of oscillation appears to remain close to constant with time. This is a more physical representation of the system than that of the results obtained from a larger time step. This indicates that energy in the system is remaining close to constant.

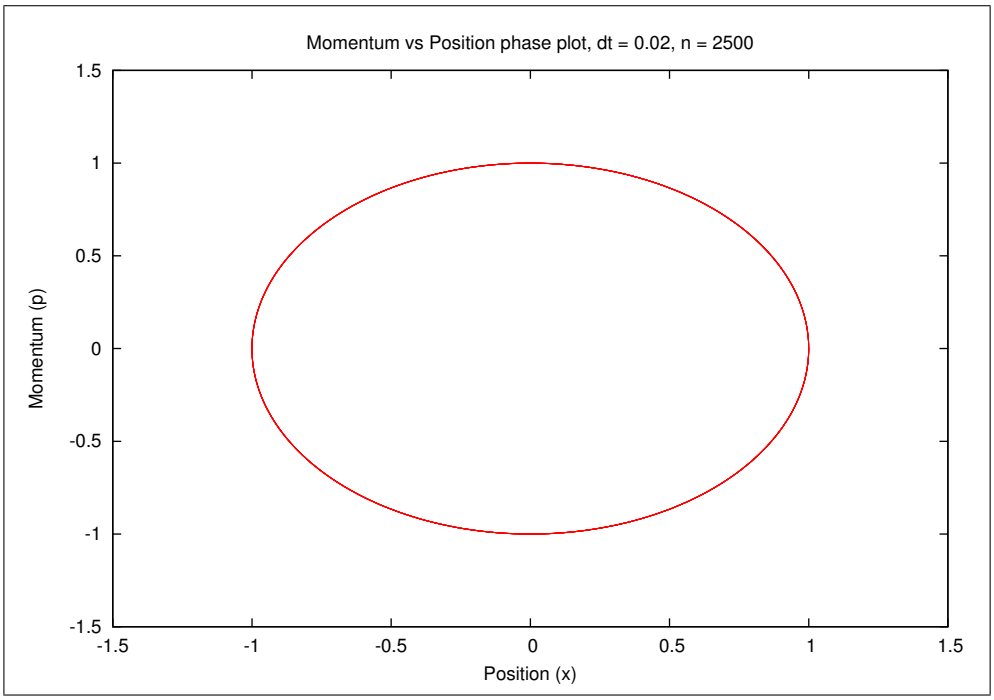

Figure 6: Second order Runge-Kutta method. Phase plot of Momentum vs. Position for $dt = 0.02, n = 2500$

Figure (6) shows the output graph of momentum vs. position in a phase space plot. The radius, $|r|$ is $\sqrt{E}$, since $E = \frac{p^2}{2} + \frac{x^2}{2}$, and appears to remain constant.

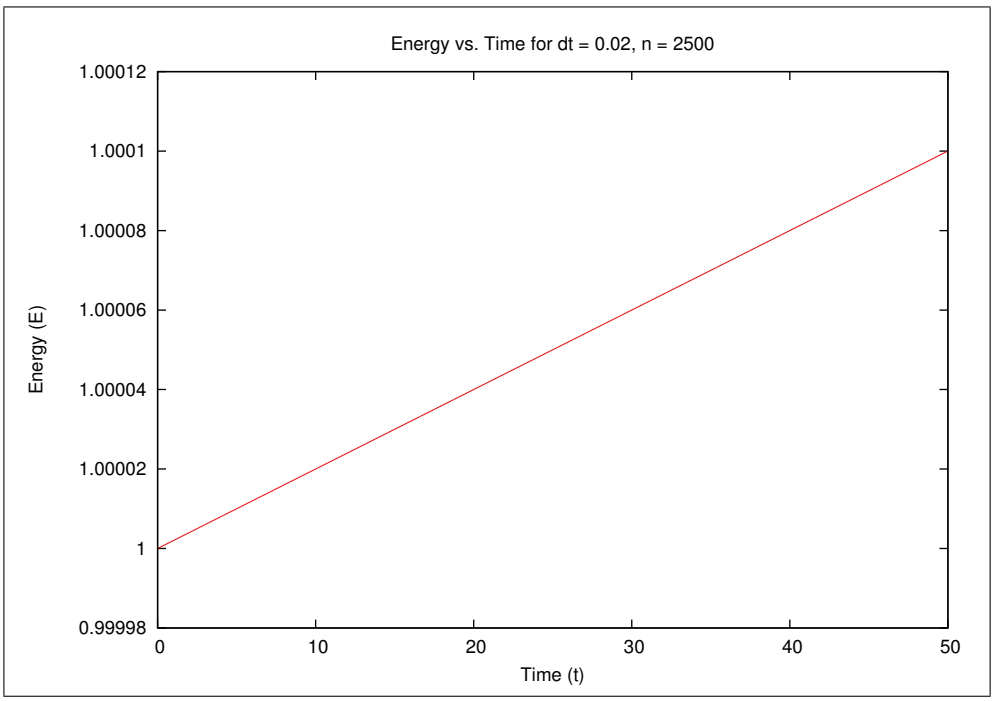

Figure 7: Second order Runge-Kutta method. Phase plot of Momentum vs. Position for $dt = 0.02, n = 2500$

Figure (7) shows the increase of energy over time. The energy is still increasing as in the previous example, however it is still much closer to $E = 1$ than with the larger time step. After 50 units of time, the energy of the

system increases by just ≈ 0.0001 units of energy.

4 Conclusions

In this assignment, two methods of numerical approximation of ordinary differential equations were introduced - Euler's Method, and the second order Runge-Kutta method.

It was found that Euler's method is a relatively inefficient algorithm, producing large errors, even at 10000 subdivisions.

The second order Runge-Kutta method was then investigated, at two different precision levels. It was found that the error on a result was inversely proportional to the number of subdivisions used. However compared with Euler's method, the Runge-Kutta method is more efficient.

Unfortunately, the due to being unable to develop a working implementation of the GSL routines for fourth order Runge-Kutta methods, it was not possible to explore the error behaviours of these methods.

5 Appendices

5.1 Tables of Data

5.1.1 Euler's Method Data

Table 1: Euler's Method, $n = 10000$ (Reduced set)

Intervals (n)	**Approximation (y)**	**Relative Error (E)**
1	7.853981633974483e-01	2.146018366025517e-01
2	8.459572975386589e-01	1.540427024613411e-01
3	8.801189879195483e-01	1.198810120804517e-01
4	9.018822813149980e-01	9.811771868500196e-02
5	9.169419341833210e-01	8.305806581667896e-02
6	9.279826868386274e-01	7.201731316137261e-02
7	9.364257661502460e-01	6.357423384975402e-02
8	9.430927338362762e-01	5.690726616372377e-02
9	9.484914877726021e-01	5.150851222739794e-02
10	9.529529514142623e-01	4.704704858573772e-02
11	9.567020195541525e-01	4.329798044584754e-02
12	9.598969005441126e-01	4.010309945588741e-02
13	9.626521443407428e-01	3.734785565925725e-02
14	9.650527402357407e-01	3.494725976425928e-02
15	9.671630805568465e-01	3.283691944315348e-02
16	9.690328475730891e-01	3.096715242691095e-02
17	9.707009895207200e-01	2.929901047927996e-02
18	9.721984725322474e-01	2.780152746775255e-02
19	9.735502268486828e-01	2.644977315131725e-02
20	9.747765498288290e-01	2.522345017117100e-02
21	9.758941348377806e-01	2.410586516221935e-02
22	9.769168374909493e-01	2.308316250905074e-02
23	9.778562543070635e-01	2.214374569293653e-02
24	9.787221652667678e-01	2.127783473323219e-02
9995	9.999455418176355e-01	5.445818236449185e-05
9996	9.999455472647586e-01	5.445273524140593e-05
9997	9.999455527107965e-01	5.444728920345199e-05
9998	9.999455581557488e-01	5.444184425118515e-05
9999	9.999455635996066e-01	5.443640039337616e-05
10000	9.999455690423819e-01	5.443095761814565e-05

5.1.2 Second Order Runge-Kutta Method Data - $dt = 0.5, n = 100$

Table 2: Second Order Runge-Kutta Method - $dt = 0.5, n = 100$

Time step (t)	Position (x)	Momentum (p)
0	1.000000000000000e+00	0.000000000000000e+00
5.000000000000000e-01	8.750000000000000e-01	-5.000000000000000e-01
1.000000000000000e+00	5.156250000000000e-01	-8.750000000000000e-01
1.500000000000000e+00	1.367187500000000e-02	-1.023437500000000e+00
2.000000000000000e+00	-4.997558593750000e-01	-9.023437500000000e-01
2.500000000000000e+00	-8.884582519531250e-01	-5.396728515625000e-01
3.000000000000000e+00	-1.047237396240234e+00	-2.798461914062500e-02
3.500000000000000e+00	-9.303250312805176e-01	4.991321563720703e-01
4.000000000000000e+00	-5.644683241844177e-01	9.019031524658203e-01
4.500000000000000e+00	-4.295820742845535e-02	1.071399420499802e+00
5.000000000000000e+00	4.981112787500024e-01	9.589535966515541e-01
5.500000000000000e+00	9.153241672320291e-01	5.900287576951087e-01
6.000000000000000e+00	1.095923025175580e+00	5.861307936720550e-02
6.500000000000000e+00	9.882391867122351e-01	-4.966750681414851e-01
7.000000000000000e+00	6.163717543024632e-01	-9.287102779799170e-01
7.500000000000000e+00	7.497014602469676e-02	-1.120807370383659e+00
8.000000000000000e+00	-4.948048074202198e-01	-1.018191522098050e+00
8.500000000000000e+00	-9.420499675417173e-01	-6.435151781256838e-01
9.000000000000000e+00	-1.146051310661845e+00	-9.205079708911468e-02
9.500000000000000e+00	-1.048820295373671e+00	4.924812078779470e-01
1.000000000000000e+01	-6.714771545129889e-01	9.553312045800393e-01
1.050000000000000e+01	-1.098769079088457e-01	1.171653381264029e+00
1.100000000000000e+01	4.896843962117744e-01	1.080135162560448e+00
1.150000000000000e+01	9.685414279655267e-01	7.002760691345048e-01
1.200000000000000e+01	1.197611784037088e+00	1.284708465099283e-01
1.250000000000000e+01	1.112145734287417e+00	-4.863939013223569e-01
1.300000000000000e+01	7.299305668403111e-01	-9.816675308007706e-01
1.350000000000000e+01	1.478554805848868e-01	-1.223924372870830e+00
1.400000000000000e+01	-4.825886409236388e-01	-1.144861566554420e+00
1.450000000000000e+01	-9.946958440853937e-01	-7.604595502732976e-01
1.500000000000000e+01	-1.250588638711368e+00	-1.680541844464385e-01
1.550000000000000e+01	-1.178292151095667e+00	4.782469079650504e-01
1.600000000000000e+01	-7.918821782261830e-01	1.007612120017253e+00
1.650000000000000e+01	-1.890908459392838e-01	1.277601694128188e+00
1.700000000000000e+01	4.733463568672204e-01	1.212446905331806e+00
1.750000000000000e+01	1.020401514924721e+00	8.242178637317201e-01
1.800000000000000e+01	1.304960257424991e+00	2.109898733028948e-01
1.850000000000000e+01	1.247335161898314e+00	-4.678639895724626e-01
1.900000000000000e+01	8.574862718747938e-01	-1.033048571825062e+00
1.950000000000000e+01	2.337762019779136e-01	-1.332660636284326e+00
2.000000000000000e+01	-4.617761414114886e-01	-1.282966157737742e+00
2.050000000000000e+01	-1.045537202603923e+00	-8.917073173147799e-01
2.100000000000000e+01	-1.360698710935823e+00	-2.574753013484707e-01
2.150000000000000e+01	-1.319349022743080e+00	4.550584667879995e-01
2.200000000000000e+01	-9.269011615061956e-01	1.057850669811040e+00
2.250000000000000e+01	-2.821131814124013e-01	1.389069916837758e+00
2.300000000000000e+01	4.476859246830277e-01	1.356492767939238e+00
2.350000000000000e+01	1.069971568067269e+00	9.630882096053198e-01
2.400000000000000e+01	1.417769226861520e+00	3.077163993710206e-01
2.450000000000000e+01	1.394406273189340e+00	-4.396327639811168e-01

5.1.3 Second Order Runge-Kutta Method Data - $dt = 0.5, n = 100$

Table 3: Second Order Runge-Kutta Method - $dt = 0.5, n = 100$ (Continued)

Time step (t)	Position (x)	Momentum (p)
2.500000000000000e+01	1.000289107050114e+00	-1.081881805078147e+00
2.550000000000000e+01	3.343120661297762e-01	-1.446791132968436e+00
2.600000000000000e+01	-4.308725086206638e-01	-1.433098274412270e+00
2.650000000000000e+01	-1.093562582249216e+00	-1.038524735800404e+00
2.700000000000000e+01	-1.476129627368266e+00	-3.619278527007457e-01
2.750000000000000e+01	-1.472577350297605e+00	4.213779425709803e-01
2.800000000000000e+01	-1.077816210224914e+00	1.104994374898410e+00
2.850000000000000e+01	-3.905919964975950e-01	1.505778183148566e+00
2.900000000000000e+01	4.111210946388876e-01	1.512851908503793e+00
2.950000000000000e+01	1.116156912060923e+00	1.118184872621375e+00
3.000000000000000e+01	1.535729734363995e+00	4.203333075132416e-01
3.050000000000000e+01	1.553930171325117e+00	-4.000732231079113e-01
3.100000000000000e+01	1.159652288355521e+00	-1.127029155881980e+00
3.150000000000000e+01	4.511811743700910e-01	-1.565976655574494e+00
3.200000000000000e+01	-3.882048002134172e-01	-1.595820160812727e+00
3.250000000000000e+01	-1.137589280593104e+00	-1.202240240604428e+00
3.300000000000000e+01	-1.596510740821179e+00	-4.831655702323224e-01
3.350000000000000e+01	-1.638529683334693e+00	3.754854964573076e-01
3.400000000000000e+01	-1.245970724689203e+00	1.147814651067491e+00
3.450000000000000e+01	-5.163170585693070e-01	1.627323182028656e+00
3.500000000000000e+01	3.618841647661842e-01	1.682066313559727e+00
3.550000000000000e+01	1.157681800950275e+00	1.290865941981669e+00
3.600000000000000e+01	1.658404546822325e+00	5.506667987588232e-01
3.650000000000000e+01	1.726437377848946e+00	-3.473688244971922e-01
3.700000000000000e+01	1.336948293369232e+00	-1.167166410359516e+00
3.750000000000000e+01	5.862465515183196e-01	-1.689744755749192e+00
3.800000000000000e+01	-3.319066452960665e-01	-1.771649937039703e+00
3.850000000000000e+01	-1.176243283153910e+00	-1.384240372261707e+00
3.900000000000000e+01	-1.721333058890524e+00	-6.230886841520388e-01
3.950000000000000e+01	-1.817710768605228e+00	3.154639308122283e-01
4.000000000000000e+01	-1.432764957123461e+00	1.184886323763314e+00
4.050000000000000e+01	-6.612261756013711e-01	1.753158011854630e+00
4.100000000000000e+01	2.980061022761153e-01	1.864626348173487e+00
4.150000000000000e+01	1.193068513578344e+00	1.482545003513743e+00
4.200000000000000e+01	1.785207451137923e+00	7.006926212853533e-01
4.250000000000000e+01	1.912402830388359e+00	-2.794976819442773e-01
4.300000000000000e+01	1.533603635617676e+00	-1.200761886895422e+00
4.350000000000000e+01	7.415222377177551e-01	-1.817468468842332e+00
4.400000000000000e+01	-2.599022764181305e-01	-1.961046029095918e+00
4.450000000000000e+01	-1.207937506413823e+00	-1.585964137249863e+00
4.500000000000000e+01	-1.849927386737027e+00	-7.837498668867189e-01
4.550000000000000e+01	-2.010561396838258e+00	2.391825598426345e-01
4.600000000000000e+01	-1.639649942312159e+00	1.214565438281434e+00
4.650000000000000e+01	-8.274109803824219e-01	1.882569729652334e+00
4.700000000000000e+01	2.173002569915481e-01	2.060954003637004e+00
4.750000000000000e+01	1.220614726686106e+00	1.694684624686604e+00
4.800000000000000e+01	1.915380198193645e+00	8.725416832577255e-01
4.850000000000000e+01	2.112228515048303e+00	-1.942161262463129e-01
4.900000000000000e+01	1.751091887544108e+00	-1.226053367989675e+00
4.950000000000000e+01	9.191787176062574e-01	-1.948342640763020e+00
5.000000000000000e+01	-1.698899424760347e-01	-2.164389169470771e+00

5.1.4 Second Order Runge-Kutta Method Data - $dt = 0.02, n = 2500$

Table 4: Second Order Runge-Kutta Method - $dt = 0.02, n = 2500$ (Reduced set)

Time step (t)	Position (x)	Momentum (p)
0	1.000000000000000e+00	0.000000000000000e+00
2.000000000000000e-02	9.998000000000000e-01	-2.000000000000000e-02
4.000000000000000e-02	9.992000400000001e-01	-3.999200000000000e-02
6.000000000000000e-02	9.982003599920001e-01	-5.996800240000000e-02
8.000000000000000e-02	9.968013598720017e-01	-7.992001599935999e-02
1.000000000000000e-01	9.950035992800401e-01	-9.984005919360016e-02
1.200000000000000e-01	9.928077973763121e-01	-1.197201631673622e-01
1.400000000000000e-01	9.902148325534896e-01	-1.395523750822550e-01
1.600000000000000e-01	9.872257420853339e-01	-1.593287612583084e-01
1.800000000000000e-01	9.838417217117507e-01	-1.790414103477634e-01
2.000000000000000e-01	9.800641251604530e-01	-1.986824364999288e-01
2.200000000000000e-01	9.758944636054224e-01	-2.182439825158379e-01
2.400000000000000e-01	9.713344050623846e-01	-2.377182229914432e-01
2.600000000000000e-01	9.663857737215432e-01	-2.570973674480926e-01
2.800000000000000e-01	9.610505492178371e-01	-2.763736634490338e-01
3.000000000000000e-01	9.553308658390128e-01	-2.955393997007008e-01
3.200000000000000e-01	9.492290116718310e-01	-3.145869091375409e-01
3.400000000000000e-01	9.427474276867459e-01	-3.335085719891500e-01
3.600000000000000e-01	9.358887067614255e-01	-3.522968188284871e-01
3.800000000000000e-01	9.286555926435035e-01	-3.709441335999499e-01
4.000000000000000e-01	9.210509788529758e-01	-3.894430566261000e-01
4.200000000000000e-01	9.130779075246832e-01	-4.077861875918343e-01
4.400000000000000e-01	9.047395681913416e-01	-4.259661885048096e-01
4.600000000000000e-01	8.960392965076072e-01	-4.439757866309355e-01
4.800000000000000e-01	8.869805729156869e-01	-4.618077774037614e-01
4.952000000000000e+01	7.370323268553897e-01	6.759307613008724e-01
4.954000000000000e+01	7.504035356160360e-01	6.610549286115044e-01
4.956000000000000e+01	7.634745534811429e-01	6.459146469134613e-01
4.958000000000000e+01	7.762401515087159e-01	6.305159729144558e-01
4.960000000000000e+01	7.886952229367034e-01	6.148650666896986e-01
4.962000000000000e+01	8.008347852259100e-01	5.989681892176266e-01
4.964000000000000e+01	8.126539820532174e-01	5.828316998752650e-01
4.966000000000000e+01	8.241480852543120e-01	5.664620538942255e-01
4.968000000000000e+01	8.353124967151457e-01	5.498657997783605e-01
4.970000000000000e+01	8.461427502113699e-01	5.330495766841019e-01
4.972000000000000e+01	8.566345131950096e-01	5.160201117645377e-01
4.974000000000000e+01	8.667835885276614e-01	4.987842174782846e-01
4.976000000000000e+01	8.765859161595215e-01	4.813487888642357e-01
4.978000000000000e+01	8.860375747535744e-01	4.637208007832724e-01
4.980000000000000e+01	8.951347832542891e-01	4.459073051280442e-01
4.982000000000000e+01	9.038739024001992e-01	4.279154280019328e-01
4.984000000000000e+01	9.122514361797578e-01	4.097523668683284e-01
4.986000000000000e+01	9.202640332298884e-01	3.914253876713596e-01
4.988000000000000e+01	9.279084881766696e-01	3.729418219292275e-01
4.990000000000000e+01	9.351817429176188e-01	3.543090638013083e-01
4.992000000000000e+01	9.420808878450614e-01	3.355345671301956e-01
4.994000000000000e+01	9.486031630100963e-01	3.166258424598684e-01
4.996000000000000e+01	9.547459592266916e-01	2.975904540311745e-01
4.998000000000000e+01	9.605068191154698e-01	2.784360167558344e-01
5.000000000000000e+01	9.658834380867634e-01	2.591701931701739e-01