AERSP 313 Computer Project
Fall 2018

THE PENNSYLVANIA STATE UNIVERSITY

Kostandin Gjergo
Trent Dempsey
Alena Kim
Ashley Bitto
Kriston Ramdass
Brody Clark
Patrick Julich

Statement of the Problem

The displacement y(t) of a simple harmonic oscillator with a natural frequency w= sqrt(k/m)= 2rad/sec and damping c/m =1/sec is described by the differential equation

$$\frac{d^2y}{dt^2}+\frac{dy}{dt}+4y=0,\quad y(0)=1\text{ cm.},\quad \frac{dy}{dt}(0)=0.$$

For this problem we need to find the exact solution for the displacement. Also, we need to write a computer program to predict the displacement as a function of time t, for the set boundary conditions of 0<t<10 seconds.

After we gather the exact values from the exact function and the predicted values from the Improved Euler Method using a h=0.1 step size, we will compare the values to see the accuracy of the estimated to the exact. We will then repeat the test with estimated values that come from a higher step value, h=0.5. We predict this should be less accurate than the h=0.1 step value and the error should be larger.

The Exact Solution

$$\frac{d^2y}{dt^2} + \frac{dy}{dt} + 4y = 0 \; ; \; y(0) = 1 \; ; \; \frac{dy}{dt}(0) = 0$$

$$\lambda^2 + \lambda + 4 = 0$$

$$\lambda = \frac{-1 \pm \sqrt{1-16}}{2} = -\frac{1}{2} \pm \frac{\sqrt{15}}{2} i$$

$$y(t) = e^{-\frac{1}{2}t}\left(c_1 \cos\frac{\sqrt{15}}{2}t + c_2 \sin\frac{\sqrt{15}}{2}t\right)$$

Plug $t=0$, $y(t) = 1$

$$1 = e^{-\frac{1}{2}(0)}\left(c_1 \cos\frac{\sqrt{15}}{2}(0) + c_2 \sin\frac{\sqrt{15}}{2}(0)\right)$$

$$= 1\left(c_1(1) + c_2(0)\right)$$

$$= c_1$$

$$y(t) = e^{-\frac{1}{2}t}\left(\cos\frac{\sqrt{15}}{2}t + c_2 \sin\frac{\sqrt{15}}{2}t\right)$$

$$\frac{dy}{dt} = -\frac{1}{2}e^{-\frac{1}{2}t}\left(\cos\frac{\sqrt{15}}{2}t + c_2 \sin\frac{\sqrt{15}}{2}t\right) + e^{-\frac{1}{2}t}\left\{\frac{\sqrt{15}}{2}\left(-\sin\frac{\sqrt{15}}{2}t + c_2\cos\frac{\sqrt{15}}{2}t\right)\right\}$$

Plug $t=0$, $\frac{dy}{dt}(0) = 0$

$$0 = -\frac{1}{2}e^{-\frac{1}{2}(0)}\left(\cos\frac{\sqrt{15}}{2}(0) + c_2 \sin\frac{\sqrt{15}}{2}(0)\right) + e^{-\frac{1}{2}(0)}\left\{\frac{\sqrt{15}}{2}\left(-2\sin\frac{\sqrt{15}}{2}(0) + c_2\cos\frac{\sqrt{15}}{2}(0)\right)\right\}$$

$$= -\frac{1}{2}(1)\left(1 + c_2(0)\right) + (1)\frac{\sqrt{15}}{2}(0 + c_2)$$

$$= -\frac{1}{2} + \frac{\sqrt{15}}{2}c_2$$

$$\therefore c_2 = \frac{1}{\sqrt{15}}$$

$$y(t) = e^{-\frac{1}{2}t}\left(\cos\frac{\sqrt{15}}{2}t + \frac{1}{\sqrt{15}}\sin\frac{\sqrt{15}}{2}t\right)$$

$$y'(t) = -\frac{1}{2}e^{-\frac{1}{2}t}\left(\cos\frac{\sqrt{15}}{2}t + \frac{1}{\sqrt{15}}\sin\frac{\sqrt{15}}{2}t\right) + e^{-\frac{1}{2}t}\left(-\frac{\sqrt{15}}{2}\sin\frac{\sqrt{15}}{2}t + \frac{1}{2}\cos\frac{\sqrt{15}}{2}t\right)$$

$$= e^{-\frac{1}{2}t}\left(-\frac{1}{2}\cos\frac{\sqrt{15}}{2}t + \frac{1}{2}\cos\frac{\sqrt{15}}{2}t - \frac{8}{\sqrt{15}}\sin\frac{\sqrt{15}}{2}t\right)$$

$$= -\frac{8}{\sqrt{15}}e^{-\frac{1}{2}t}\sin\left(\frac{\sqrt{15}}{2}t\right)$$

Therefore, the exact solution for the displacement is

$$y_1 = y(t) = e^{-\frac{1}{2}t}\left(\cos\left(\frac{\sqrt{15}}{2}t\right) + \frac{1}{\sqrt{15}}\sin\left(\frac{\sqrt{15}}{2}t\right)\right)$$

And the exact solution for the velocity is

$$y_2 = y'(t) = -\frac{8}{\sqrt{15}}e^{-\frac{1}{2}t}\sin\left(\frac{\sqrt{15}}{2}t\right)$$

Comparison of the Numerical and Exact Solution

1. h=0.1

n	x	Numerical y(x)	actual y(x)	numerical y'(x)	actual y'(x)	% error y(x)	y'(x)
0	0	1	1	0	0	0	0
1	0.1	0.9243	0.980714	-0.7087	-0.378118	0.073	0.498
2	0.2	0.838488	0.926057	-0.978433	-0.705908	0.19	0.396
3	0.3	0.728767	0.841468	-1.18454	-0.975724	0.354	0.278
4	0.4	0.60166	0.733005	-1.32525	-1.18285	0.578	0.143
5	0.5	0.463728	0.607055	-1.40148	-1.32538	0.889	0.01
6	0.6	0.321314	0.47006	-1.41652	-1.40407	1.347	0.184
7	0.7	0.180318	0.328268	-1.37572	-1.42201	2.118	0.386
8	0.8	0.0460177	0.187515	-1.28603	-1.38428	3.838	0.618
9	0.9	-0.077075	0.0530432	-1.15563	-1.29763	13.245	0.894
10	1	-0.185319	-0.070644	-0.993441	-1.17	9.104	1.228
11	1.1	-0.275989	-0.179882	-0.808774	-1.01012	3.023	1.651
12	1.2	-0.347303	-0.271899	-0.610889	-0.82714	1.504	2.22
13	1.3	-0.398392	-0.344845	-0.408662	-0.630189	0.713	3.063
14	1.4	-0.429247	-0.397766	-0.210277	-0.428072	0.157	4.534
15	1.5	-0.440638	-0.43056	-0.022981	-0.228949	0.305	8.156
16	1.6	-0.434008	-0.443899	0.147104	-0.040088	0.735	42.674
17	1.7	-0.411353	-0.439127	0.29511	0.13232	1.166	11.173
18	1.8	-0.375091	-0.41815	0.417487	0.283302	1.625	4.168
19	1.9	-0.327928	-0.383305	0.51201	0.409168	2.143	2.033
20	2	-0.272728	-0.337235	0.577742	0.507534	2.76	0.882
21	2.1	-0.212388	-0.282754	0.614938	0.577283	3.546	0.08
22	2.2	-0.149721	-0.22273	0.624928	0.618494	4.643	0.575
23	2.3	-0.087358	-0.159969	0.609955	0.632318	6.406	1.169
24	2.4	-0.027665	-0.097111	0.573007	0.620838	10.043	1.753
25	2.5	0.0273232	-0.036551	0.517624	0.586897	24.309	2.367
26	2.6	0.075951	0.0196332	0.447714	0.533919	39.168	3.052

27	2.7	0.116965	0.0697257	0.367366	0.465728	8.928	3.868
28	2.8	0.149525	0.112406	0.280672	0.386356	4.056	4.915
29	2.9	0.173199	0.14676	0.191575	0.29988	1.884	6.405
30	3	0.187934	0.172277	0.103729	0.210257	0.535	8.885
31	3.1	0.19403	0.18883	0.0203848	0.121185	0.475	14.404
32	3.2	0.192086	0.196644	-0.055691	0.035992	1.329	43.363
33	3.3	0.182953	0.196254	-0.122279	-0.042459	2.124	31.163
34	3.4	0.167678	0.188456	-0.177739	-0.111834	2.92	9.34
35	3.5	0.147439	0.174249	-0.221017	-0.170364	3.771	4.329
36	3.6	0.123494	0.154785	-0.251627	-0.216861	4.746	1.916
37	3.7	0.0971193	0.1313	-0.269617	-0.250706	5.945	0.367
38	3.8	0.0695633	0.105068	-0.275517	-0.271818	7.565	0.81
39	3.9	0.041998	0.0773472	-0.270266	-0.280609	10.064	1.815
40	4	0.0154827	0.0493296	-0.255145	-0.277918	14.862	2.753
41	4.1	-0.009066	0.0221068	-0.231687	-0.264939	29.964	3.697
42	4.2	-0.030895	-0.003364	-0.201598	-0.243144	169.479	4.712
43	4.3	-0.049429	-0.026284	-0.166674	-0.214199	17.543	5.883
44	4.4	-0.064274	-0.046025	-0.128724	-0.179881	7.395	7.342
45	4.5	-0.075217	-0.062141	-0.089496	-0.142005	3.432	9.352
46	4.6	-0.082215	-0.074367	-0.050622	-0.102346	1.144	12.555
47	4.7	-0.08538	-0.082607	-0.013559	-0.062578	0.474	19.106
48	4.8	-0.084960	-0.086929	0.0204451	-0.024221	1.783	44.021
49	4.9	-0.081319	-0.087542	0.0503788	0.0114056	2.95	79.255
50	5	-0.074906	-0.084776	0.0754864	0.0432089	4.078	16.594
51	5.1	-0.066237	-0.079057	0.0952699	0.0703464	5.25	7.307
52	5.2	-0.05586	-0.070882	0.109484	0.0922326	6.553	3.293
53	5.3	-0.044343	-0.060796	0.118121	0.108537	8.117	0.873
54	5.4	-0.032235	-0.049364	0.121387	0.119172	10.17	0.882
55	5.5	-0.020058	-0.037146	0.119677	0.124274	13.222	2.323
56	5.6	-0.008288	-0.024682	0.113537	0.124175	18.734	3.622
57	5.7	0.0026636	-0.012468	0.103629	0.119375	33.527	4.89

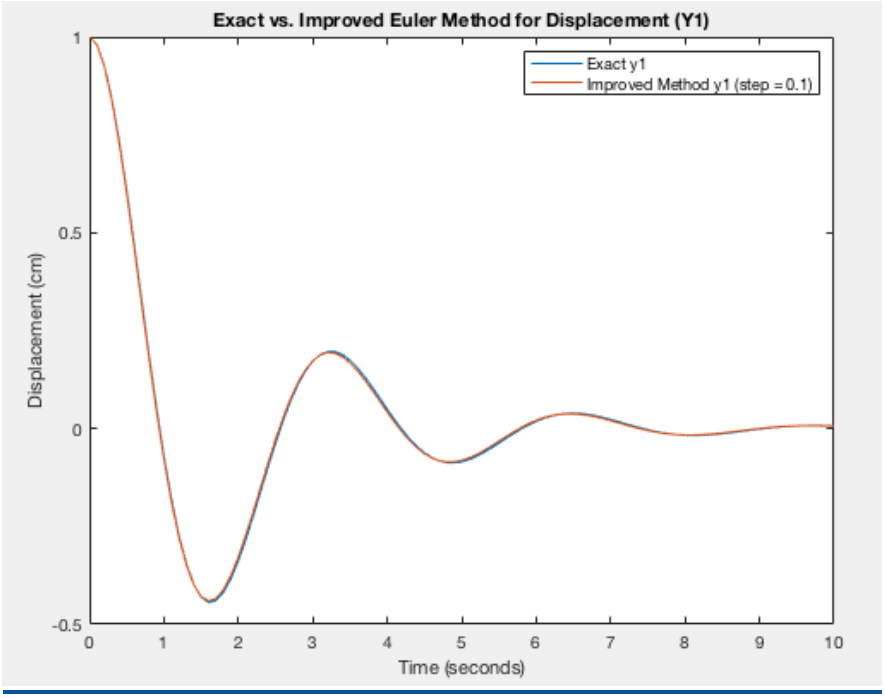
58	5.8	0.0124551	-0.000943	0.0906998	0.110503	382.322	6.221
59	5.9	0.0208225	0.0095205	0.0755364	0.0982828	30.824	7.715
60	6	0.027582	0.0186276	0.0589372	0.0834968	11.783	9.534
61	6.1	0.0326294	0.0261612	0.0416782	0.0669502	5.431	11.969
62	6.2	0.0359362	0.0319855	0.0244861	0.0494382	2.013	15.696
63	6.3	0.0375437	0.036042	0.0080144	0.0317171	0.294	22.798
64	6.4	0.0375542	0.0383451	-0.007174	0.0144791	2.09	44.648
65	6.5	0.0361216	0.0389743	-0.020619	-0.001667	3.644	330.137
66	6.6	0.0334403	0.038065	-0.031974	-0.016215	5.105	27.164
67	6.7	0.0297339	0.0357981	-0.041004	-0.028762	6.586	11.167
68	6.8	0.0252438	0.0323888	-0.047588	-0.039024	8.197	5.074
69	6.9	0.0202181	0.0280752	-0.051708	-0.046829	10.085	1.619
70	7	0.0149014	0.0231069	-0.053444	-0.052115	12.502	0.782
71	7.1	0.0095262	0.0177348	-0.052961	-0.054921	15.976	2.689
72	7.2	0.0043043	0.0122011	-0.050490	-0.055376	21.924	4.362
73	7.3	-0.000578	0.0067310	-0.046319	-0.053687	36.052	5.954
74	7.4	-0.004967	0.0015262	-0.040773	-0.050121	137.89	7.586
75	7.5	-0.008741	-0.003241	-0.034196	-0.044994	53.251	9.382
76	7.6	-0.011815	-0.007432	-0.026942	-0.038647	17.617	11.517
77	7.7	-0.014138	-0.010942	-0.019354	-0.031438	7.979	14.302
78	7.8	-0.015694	-0.013702	-0.011756	-0.023723	3.178	18.416
79	7.9	-0.016497	-0.015681	-0.004440	-0.015841	0.084	25.792
80	8	-0.016589	-0.016876	0.0023392	-0.008109	2.246	45.246
81	8.1	-0.016035	-0.017317	0.0083740	-0.000805	4.206	390.461
82	8.2	-0.014918	-0.017059	0.0135044	0.0058343	6.005	43.532
83	8.3	-0.013337	-0.016179	0.0176205	0.0116208	7.789	16.209
84	8.4	-0.011396	-0.014768	0.0206624	0.0164157	9.689	7.339
85	8.5	-0.009205	-0.012931	0.022617	0.0201316	11.871	2.637
86	8.6	-0.006873	-0.010779	0.0235143	0.0227301	14.597	0.498
87	8.7	-0.004501	-0.008422	0.023422	0.0242189	18.398	2.909
88	8.8	-0.002186	-0.005971	0.0224391	0.0246473	24.605	4.971

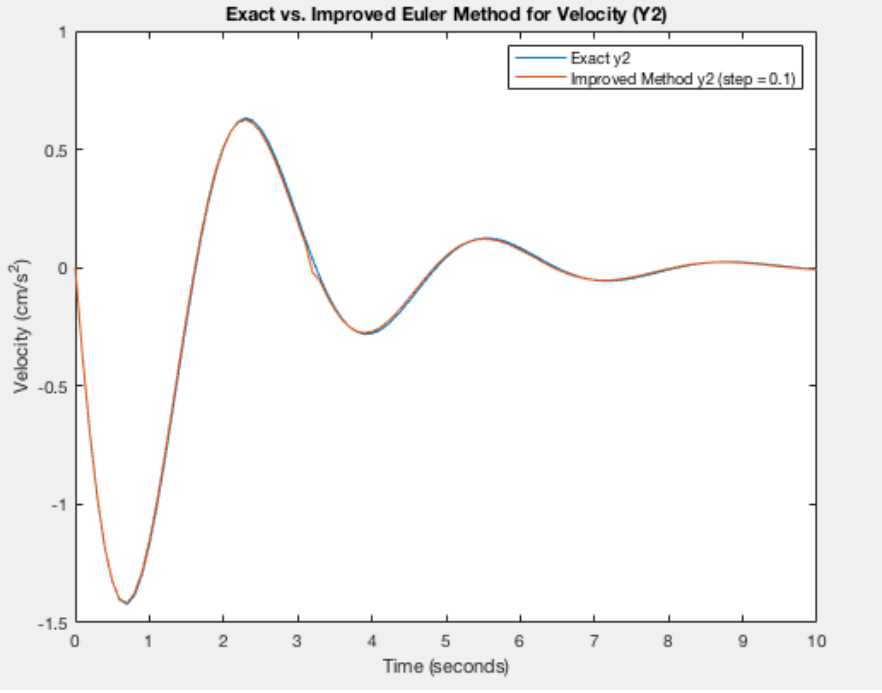
89	8.9	-1.13E-05	-0.003525	0.0206896	0.0240998	37.982	6.891
90	9	1.95E-03	-0.001179	0.0183146	0.0226901	99.042	8.817
91	9.1	0.0036552	0.0009878	0.0154657	0.0205538	97.856	10.894
92	9.2	0.0050513	0.0029117	0.0122982	0.0178409	25.536	13.313
93	9.3	0.0061186	0.0045420	0.0089643	0.0147094	11.214	16.392
94	9.4	0.0068479	0.0058449	0.0056083	0.0113178	4.684	20.794
95	9.5	0.0072437	0.0068020	0.0023612	0.0078196	0.674	28.278
96	9.6	0.0073231	0.0074100	-0.000662	0.0043576	2.244	45.815
97	9.7	0.0071137	0.0076790	-0.003369	0.0010597	4.634	162.556
98	9.8	0.0066513	0.0076311	-0.005685	-0.001964	6.78	71.528
99	9.9	0.0059782	0.0072982	-0.007558	-0.004626	8.864	22.89
100	10	0.0051405	0.0067202	-0.008961	-0.006859	23.51	30.64

2. h=0.5

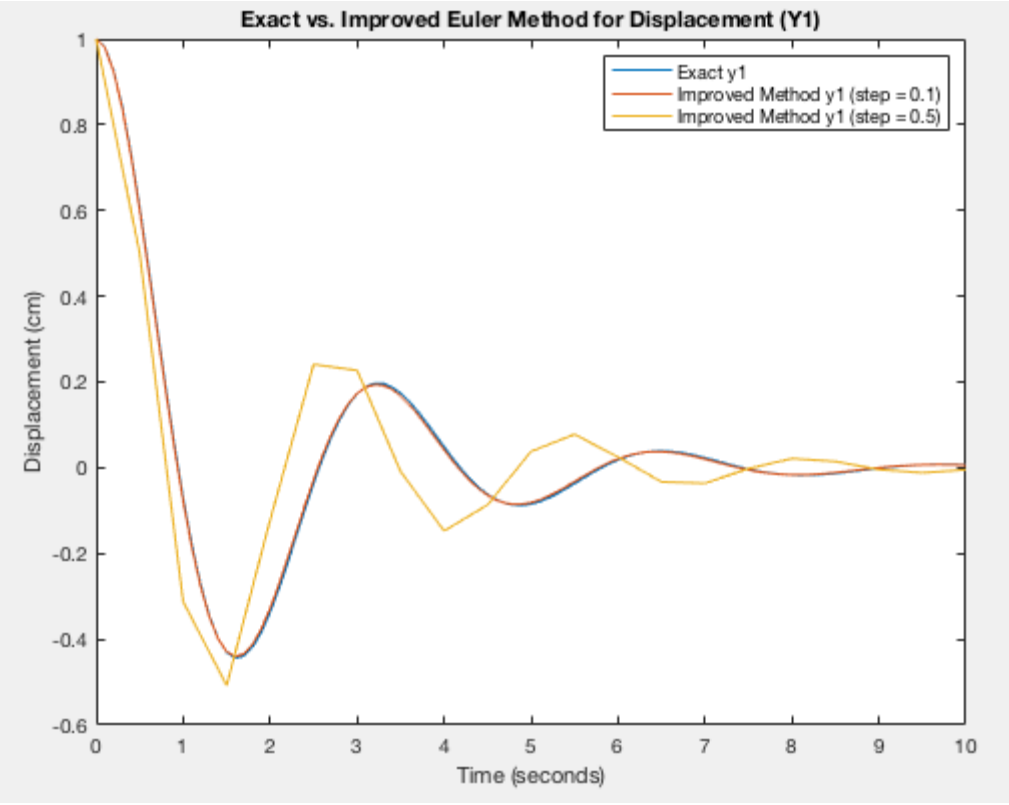
n	x	Numerical y(x)	actual y(x)	numerical y'(x)	actual y'(x)	error y(x)	y'(x)
0	0	1	1	0	0	0	0
1	0.5	0.5	0.607055	-1.5	-1.32538	17.635	13.175
2	1	-0.3125	-0.070644	-0.9375	-1.17	342.355	19.872
3	1.5	-0.507813	-0.43056	0.351563	-0.228949	17.942	253.555
4	2	-0.12207	-0.337235	0.805664	0.507534	63.803	58.741
5	2.5	0.241089	-0.036550	0.283813	0.586897	759.6	51.642
6	3	0.226974	0.172277	-0.326157	0.210257	31.749	255.123
7	3.5	-0.008821	0.174249	-0.381231	-0.170364	105.063	123.774
8	4	-0.147372	0.0493296	-0.034421	-0.277918	398.75	87.614
9	4.5	-0.086594	-0.062141	0.216756	-0.142005	39.351	252.64
10	5	0.0379863	-0.084776	0.156986	0.0432089	144.808	263.319
11	5.5	0.0778629	-0.037146	-0.037356	0.124274	309.61	130.06
12	6	0.0249229	0.0186276	-0.121464	0.0834968	33.796	245.471
13	6.5	-0.033087	0.0389743	-0.052567	-0.001667	184.896	3051.876
14	7	-0.036256	0.0231069	0.0430604	-0.052115	256.908	182.625
15	7.5	-0.001980	-0.003241	0.0597673	-0.044994	38.892	232.833

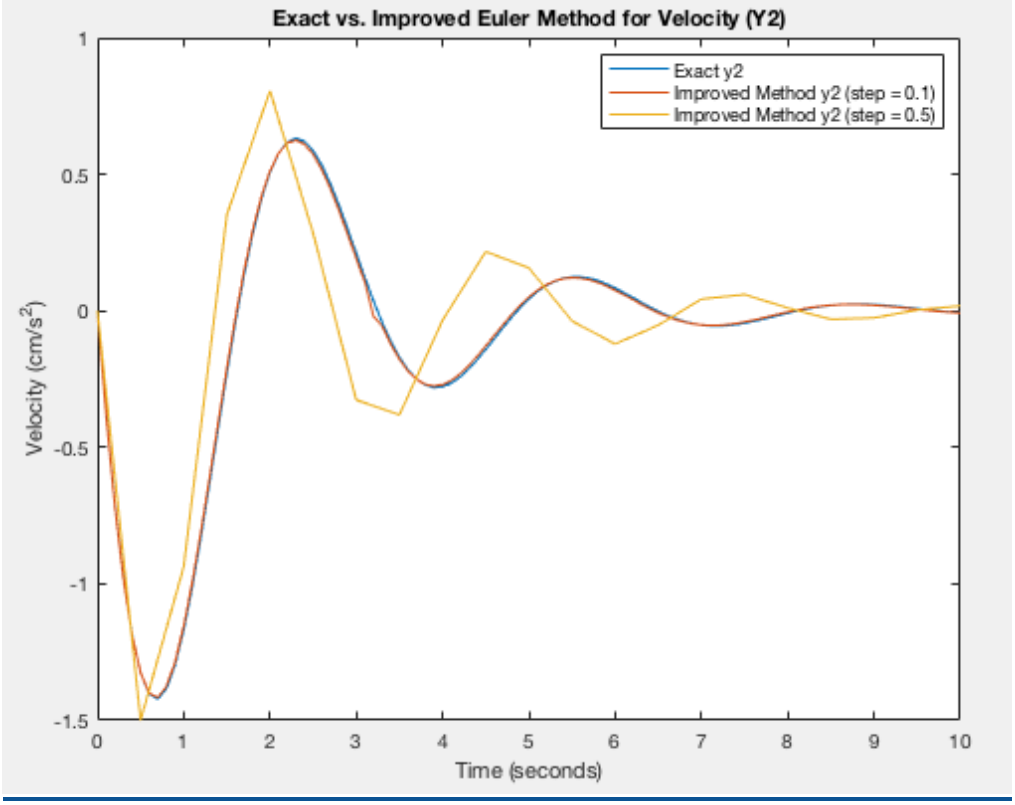
16	8	0.0214224	-0.016876	0.0104418	-0.008109	226.939	228.759
17	8.5	0.0146269	-0.012931	-0.030828	0.0201316	213.107	253.134
18	9	-0.004247	-0.001179	-0.025793	0.0226901	259.985	213.679
19	9.5	-0.011796	0.0068020	0.0031465	0.0078196	273.423	59.76
20	10	-0.004718	0.0067202	0.0180878	-0.006859	170.209	363.694

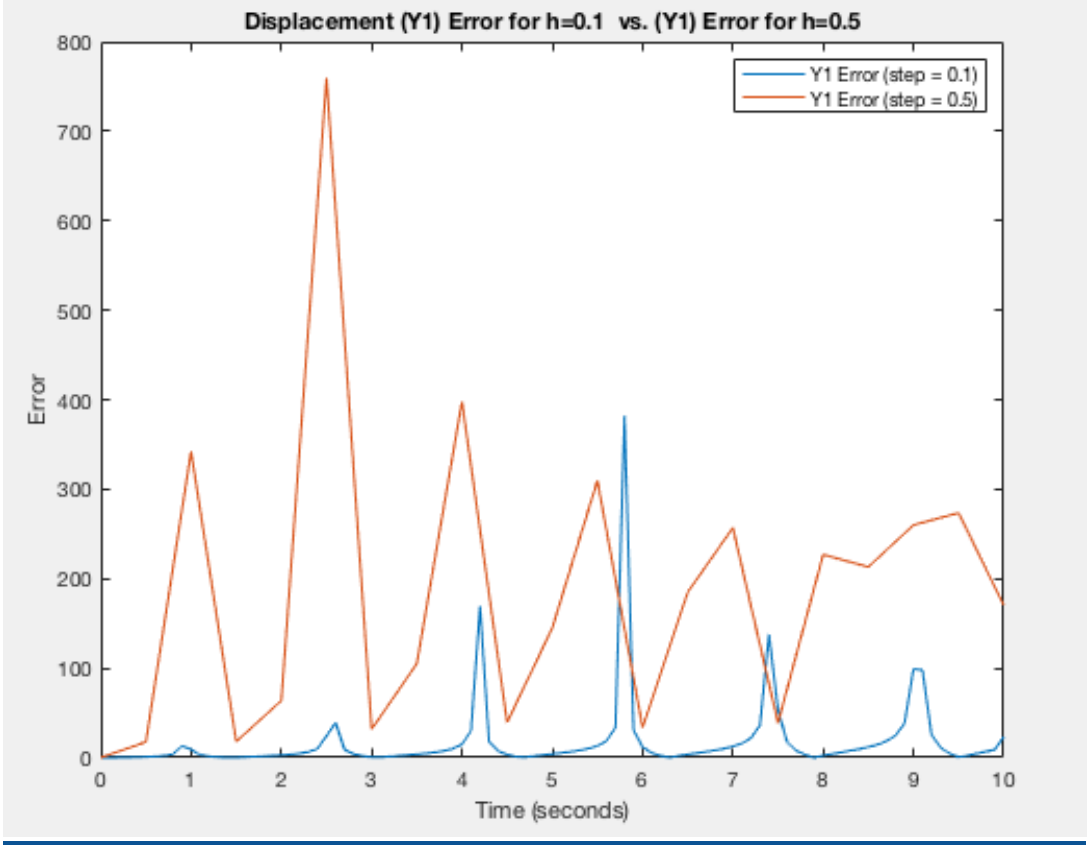


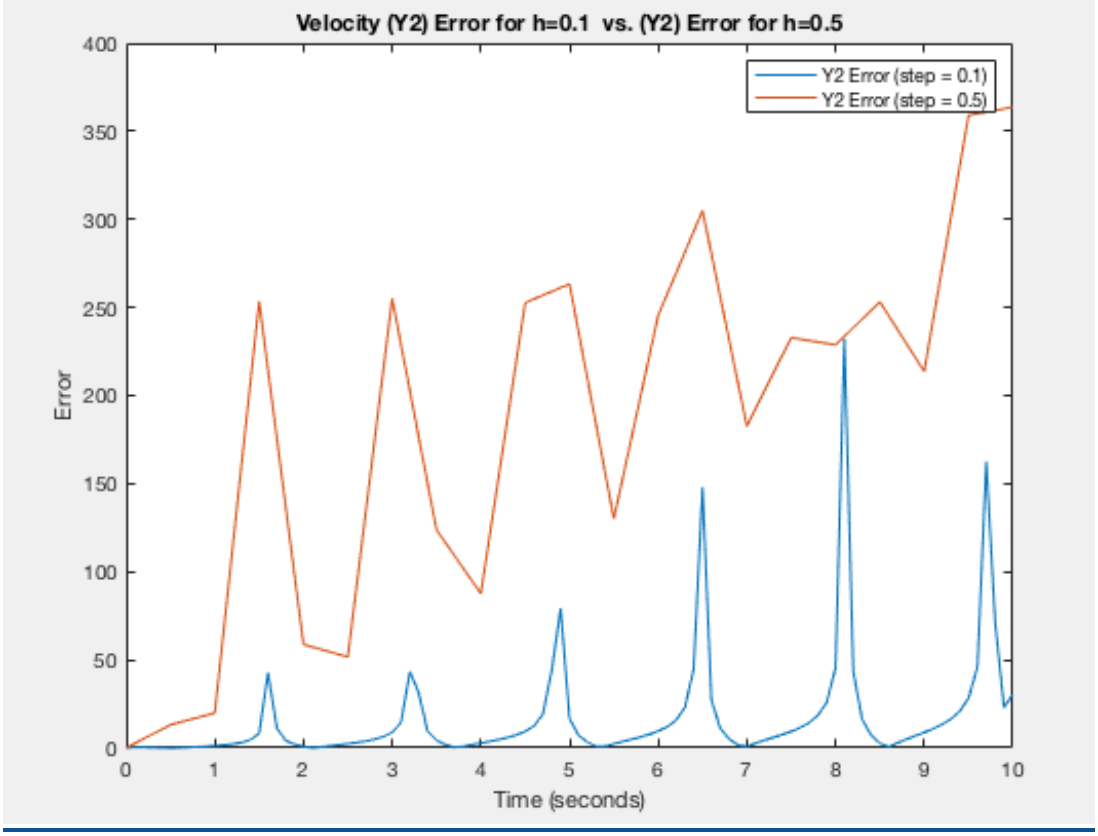


The Result Showing Effect of Time Step Size






Displacement (Y1) Error for h=0.1 vs. (Y1) Error for h=0.5
Y1 Error (step = 0.1)
Y1 Error (step = 0.5)
Error
Time (seconds)
0
100
200
300
400
500
600
700
800
0
1
2
3
4
5
6
7
8
9
10


Velocity (Y2) Error for h=0.1 vs. (Y2) Error for h=0.5
Y2 Error (step = 0.1)
Y2 Error (step = 0.5)
Error
Time (seconds)
0
50
100
150
200
250
300
350
400
0
1
2
3
4
5
6
7
8
9
10

Discussion of the Results

- Process

Once given the project, we began with finding the exact solution for the displacement of the simple harmonic oscillator by computing it by hand using constant coefficients. Then we used C++ to predict the displacement as a function of time between zero and ten seconds—this used the Improved Euler method. The C++ code gave us the values for the differential equation which we compared with the exact solution. In order to compare these values, we used graphical comparison and error analysis. We continued to use graphical comparison to highlight the effect of time step size on the predicted displacement.

- Discuss values and meaning

The graphs provided give a visual representation of what was calculated numerically in the exact solution and by the code with the Improved Euler Method (IEM). We ran the tests at three values: the exact at h=0.1, the IEM at h=0.1, and the IEM at h=0.5. The point of doing so was to show the comparison between the exact and the IEM when h is small, and IEM when h is larger and show how it is less accurate—and that is what we found. The errors between the h=0.1 step and the exact ranged primarily from 0% - 25% which is very reasonable. When we compared the values with a h=0.5 step size and the exact, we received errors reaching over 200%, which goes to show the effectiveness of a small step size. Additionally, the h=0.1 step values were very similar to the exact values, however, there were still areas of large error, and we believe that can be fixed by making the step size smaller, like h=0.05 for example.

- Methods used

We coded the improved Euler method in C++ and used the constant coefficient method to solve the equation for the exact solution. We did this together to check each other's work and make sure everyone agreed that the solution was correct. Then we copied the work down neatly to be scanned onto the submission. Once we gathered the values for the exact solution and the Improved Euler Method, we used MATLAB to quickly graph the trends to we could see a visual representation of the differences between the values.

- Divided and conquered

We broke the project up into small steps so that all members in the groups had something to work on—this allowed us to easily complete all the tasks. We had parts of the group creating the code, others wrote down the report, some copied the solution, and everyone participated in formatting the results. This allowed us to get the work done quickly even though we had a few scheduling conflicts.

Appendix: Copy of Computer Code

```
#include <iostream>
#include <iomanip>
#include <cmath>
#include <fstream>
using namespace std;

void exact(double i, double &ex1, double &ex2);                 //prototype for exact equation function
void IEM(double h, double &y1, double &y2);   //prototype for improved euler method function

int main()
{
double y1, y2, ex1, ex2;        //variable for storing initial conditions. will be changed during process
double h;                       //variable for step size
double e1, e2;          //variable for error

        //asks user to enter initial conditions
        cout << "please enter the initial conditions:" << endl;
        cout << "Y(0) = ";
        cin >> y1;
        cout << endl;
        cout << "Y'(0) = ";
        cin >> y2;
        cout << endl;
        cout << "please enter the step size: ";
        cin >> h;


        //creates files for outputting estimated and exact values
        ofstream output1;
        output1.open("output1.txt");
        ofstream output2;
        output2.open("output2.txt");
        ofstream exactans1;
        exactans1.open("exact solutions1.txt");
        ofstream exactans2;
        exactans2.open("exact solutions2.txt");
        ofstream errors1;
        errors1.open("percent errors.txt");
        ofstream errors2;
        errors2.open("percent errors2.txt");
```

```
for (double t = 0; t <= 10; t += h)  //loop for reiterating functions to produce values each step
	{


	IEM(h, y1, y2);		//improved euler method function

	output1 << y1 << endl;			//outputs estimated answers for Y1 and Y2 to a file
	output2 << y2 << endl;
exact(t, ex1, ex2);	//exact equation function



	exactans1 << ex1 <<endl;		//outputs exact answers for Y1 and Y2 to a file
	exactans2 << ex2 << endl;



	}

	output1.close();			//closes files
	exactans1.close();
	exactans2.close();
	output2.close();



ifstream errorvalues;  //creates file for error percentages
errorvalues.open("output1.txt");


double estimated1[100], estimated2[100];	//creates two arrays whose elements will be compared for calculation

estimated1[0] = 1;	//initializes the estimated values for Y1 and Y2 to 1 and 0 for comparing to exact values.
estimated2[0] = 0;

for (int i = 1; i < (10 / h); i++)
	{

	errorvalues >> estimated1[i];  //reads the estimated values file and puts values in array
```

```
	}
	errorvalues.close();
	errorvalues.open("output2.txt");
		for (int i = 1; i < (10 / h); i++)
		{

			//reads the estimated values file and puts values in array
		errorvalues >> estimated2[i];

		}
	errorvalues.close();
	errorvalues.open("exact solutions1.txt");	//opens exact solutions file
	double exacts1[100], exacts2[100];			//creates exact solution array

	for (int i = 0; i <( 10 / h); i++)
	{

	errorvalues >> exacts1[i];	//reads file and puts values from exact solution file to array


	}
	errorvalues.close();
	errorvalues.open("exact solutions2.txt");
	for (int i = 0; i <(10 / h); i++)
	{

		//reads file and puts values from exact solution file to array
		errorvalues >> exacts2[i];

	}

	errorvalues.close();




for (int i = 0; i < (10 / h); i ++)
{
```

```
e1 = 100 * (abs(exacts1[i] - estimated1[i]) / abs(exacts1[i]));          //creates percent error value for Y1 values

errors1 << setw(4)<<fixed<<setprecision(3) << e1<<endl;   //outputs to file

e2 = 100 * (abs(exacts2[i] - estimated2[i]) / abs(exacts2[i]));          //creates percent error value for Y2 values
errors2 << setw(4) << fixed<<setprecision(3)<<e2<< endl;


        }

        errors1.close();          //closes files
        errors2.close();

                return 0;
}




void exact(double i, double &ex1, double &ex2)    //exact equation file, accepts the step value, and two reference variables as parameters
{



                double eq1 = exp(-i / 2)*(cos(pow(15, .5)*i / 2) + (1 / pow(15, .5))*sin(pow(15, .5)*i / 2));        //calculates actual Y1 based on actual equation
                ex1 = eq1;//changes value of ex1
                double eq2 =( ((-1) / pow(15, .5))*(8 * exp(-i / 2)*sin(pow(15, .5)*i / 2)));
        //calculates actual value of Y2 based on derivative of actual equation
                ex2 = eq2;      //changes value of ex2



}

void IEM(double h, double &y1, double &y2)          //improved euler method function. accepts step size and two reference variables as parameters.
{

        double y11, y22;                    //new values of Y1 and Y2
        double a = y2; //first function created by differential equation
```

```
        double b = -a - 4 * y1;             //second function created by differential equation


        double k1[2] = { a, b };//K1 array, has element K11 and K12. K11 = a function and K12 =
b function

        double astar = a + h*b;          //placeholder for new function called astar. increases b by
step size
        double bstar = -astar - 4 * (y1 + h*a);          //placeholder for new function called bstar.
replaces a with astar and increases Y1

        double k2[2] = { astar, bstar };          //new K values based on astar and bstar

        y11 = y1 + (h / 2)*(k1[0] + k2[0]);                //solves for new values of Y1 and Y2
        y22 = y2 + (h / 2)*(k1[1] + k2[1]);

        y1 = y11;        //changes vale of y1 and y2 for next iteration to use.
        y2 = y22;

}
```