1.2

1.2 Ordinary Differential Equations

1.2 Ordinary Differential Equations

1.2.1 First-order ODEs

In the first part of the report we focus on the first order differential equation

$$\frac{dy}{dx} = -4y + 3e^{-x},\ y(0) = 0 \tag{1}$$

This has the exact solution

$$y(x) = e^{-x} - e^{-4x} \tag{2}$$

Question 1: The table below shows the result of the Leapfrog method starting with $x_0 = 0,\ Y_0 = 0$. A program that generates the table is listed on page .

Table 1: Leapfrog method with $x_0 = 0,\ Y_0 = 0$, $n = 25$, and $h = 0.4$

n	x_n	Y_n	$y(x_n)$	E_n
1	4.000000E-01	1.200000E+00	4.684235E-01	7.315765E-01
2	8.000000E-01	-2.231232E+00	4.085668E-01	-2.639799E+00
3	1.200000E+00	9.418332E+00	2.929645E-01	9.125367E+00
4	1.600000E+00	-3.164703E+01	2.002350E-01	-3.184726E+01
5	2.000000E+00	1.111734E+02	1.349998E-01	1.110384E+02
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
21	8.400000E+00	5.300271E+10	2.248673E-04	5.300271E+10
22	8.800000E+00	-1.848096E+11	1.507331E-04	-1.848096E+11
23	9.200000E+00	6.443936E+11	1.010394E-04	6.443936E+11
24	9.600000E+00	-2.246869E+12	6.772874E-05	-2.246869E+12
25	1.000000E+01	7.834375E+12	4.539993E-05	7.834375E+12

From Table 1 we see that the global error grows exponentially, which indicates that the Leapfrog method is unstable.

To estimate the growth rate γ, we assume that

$$|E| = ke^{\gamma x} \text{ for some constant } k \tag{3}$$

Note that for $n = 24$ and $n = 25$, we have $\dfrac{|E_{24}|}{|E_{25}|} = \dfrac{2.246869}{7.834375} = \dfrac{e^{\gamma x_{24}}}{e^{\gamma x_{25}}} = e^{-0.4\gamma}$, hence $\gamma = 3.12246$.

We could carry out the same procedure for different n and h. The following

Table 2: Maximal instability and growth rate for different n and h

(n, h)	Maximal Instability	Growth Rate
(50,0.2)	-8.281515E+14	3.663330E+00
(100,0.1)	-2.907417E+15	3.900380E+00
(200,0.05)	-1.647960E+15	3.974347E+00
(1000,0.01)	-8.722706E+15	3.998545E+00

table shows the results obtained. We see from Table 2 that reducing h does not have a noticeable effect on the size of the instability and increases the growth rate. In fact we can conjecture that, in the limit $h \to 0$, the growth rate tends to 4.

Question 2:

Note that the difference equation gives the numerical solution Y_n.

(i): Rewrite the difference equation into the standard form

$$Y_{n+1} + 8hY_n - Y_{n-1} = 6he^{-nh}$$

Using method of characteristic equation, the general solution is

$$Y_n = A(-4h + \sqrt{16h^2+1})^n + B(-4h - \sqrt{16h^2+1})^n + \frac{3he^{-nh}}{4h - \sinh(h)}$$

for some A, B to be determined.

By substituting the initial condition we find that

$$Y_n = \frac{3h}{2\sinh(h) - 8h}(1 + \frac{\cosh(h)}{\sqrt{16h^2+1}})(-4h + \sqrt{16h^2+1})^n +$$
$$\frac{3h}{2\sinh(h) - 8h}(1 - \frac{\cosh(h)}{\sqrt{16h^2+1}})(-4h - \sqrt{16h^2+1})^n + \frac{3he^{-nh}}{4h - \sinh(h)}$$

(ii): The instability comes from the term $(-4h - \sqrt{16h^2+1})^n$: Since $|-4h - \sqrt{16h^2+1}| > 1$ for all $h > 0$, this term blows up as $n \to \infty$.

The growth rate is estimated by $\gamma = \dfrac{\log(|-4h - \sqrt{16h^2+1}|)}{h}$. It can be seen that, in the limit $h \to 0$, $\gamma \to 4$, which confirms our conjecture in Question 1.

(iii): In the limit $h \to 0$, $n \to \infty$ with $x_n = nh$ fixed,

$$
\begin{aligned}
&\lim_{h\to 0} Y_n \\
&= \lim_{h\to 0}\Big(\frac{3h}{2\sinh(h)-8h}\big(1+\frac{\cosh(h)}{\sqrt{16h^2+1}}\big)(-4h+\sqrt{16h^2+1})^n \\
&+ \frac{3h}{2\sinh(h)-8h}\big(1-\frac{\cosh(h)}{\sqrt{16h^2+1}}\big)(-4h-\sqrt{16h^2+1})^n + \frac{3he^{-nh}}{4h-\sinh(h)}\Big) \\
&= -e^{-4x_n} + 0*(-1)^n * e^{4x_n} + e^{-x_n} \text{ (by L'Hopital's Rule)} \\
&= e^{-x_n} - e^{-4x_n}\text{, as required.}
\end{aligned}
$$

The size of instability is indeed suppressed by taking small h: In evaluating the limit as $h \to 0$, $1-\dfrac{\cosh(h)}{\sqrt{16h^2+1}}$ term vanishes, and the exploding term e^{4x_n} is bounded by e^{40}. By taking h sufficiently small we could reduce the magnitude of instability.
To further exemplify the above, we choose $n = 100000$, $h = 0.0001$. In this case the maximal instability is $8.8270 * 10^7$, which is significantly smaller than all cases in Question 1. This also shows that for an ideally sufficiently small h, the cost of computation is very large.

Question 3: The numerical solutions of the ODE (1) using the Euler and the RK4 method are tabulated in Table 3.

Table 3: Numerical Solutions Using Euler and the RK4 Method

n	x_n	$Y_{n,Euler}$	$Y_{n,RK4}$
1	0.40000000	1.20000000	0.40585600
2	0.80000000	0.08438406	0.38179688
3	1.20000000	0.48856432	0.28560073
4	1.60000000	0.06829446	0.19946792
5	2.00000000	0.20129915	0.13587704
6	2.40000000	0.04162285	0.09166779
7	2.80000000	0.08388783	0.06160540
8	3.20000000	0.02263938	0.04133823
9	3.60000000	0.03533102	0.02772144
10	4.00000000	0.01158986	0.01858537

A plot of these numerical solutions is given by Figure 1.

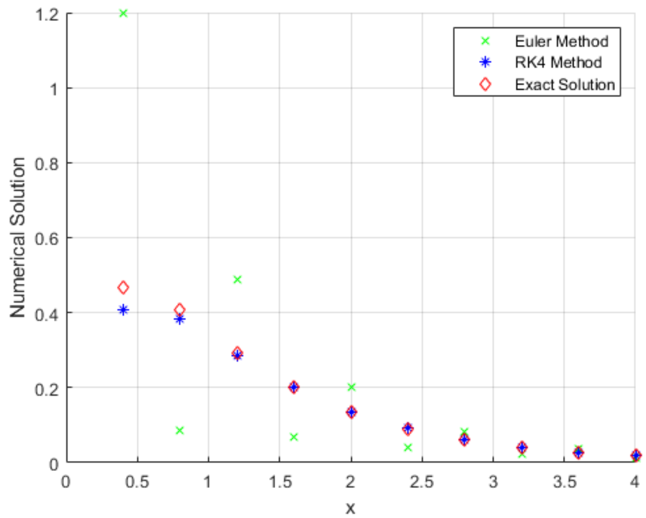

Figure 1: The Numerical Solutions

Question 4: The global error E_n at $x_n = 0.4$ for each of the Euler, LF, and RK4 method with various (n, h) is tabulated in Table 4.

Table 4: Global Error for Various Methods

(n, h)	Euler Method	LeapFrog Method	RK4 Method
$(2^0, 0.4/2^0)$	7.315765E-01	7.315765E-01	-6.256753E-02
$(2^1, 0.4/2^1)$	1.428149E-01	-4.459466E-01	-1.923277E-03
$(2^2, 0.4/2^2)$	6.371591E-02	-1.601633E-01	-8.420115E-05
$(2^3, 0.4/2^3)$	3.003888E-02	-4.479847E-02	-4.413362E-06
$(2^4, 0.4/2^4)$	1.459857E-02	-1.153981E-02	-2.526839E-07
$(2^5, 0.4/2^5)$	7.198271E-03	-2.906985E-03	-1.511655E-08
$(2^6, 0.4/2^6)$	3.574394E-03	-7.281358E-04	-9.243529E-10
$(2^7, 0.4/2^7)$	1.781075E-03	-1.821210E-04	-5.714446E-11
$(2^8, 0.4/2^8)$	8.890148E-04	-4.553569E-05	-3.551992E-12
$(2^9, 0.4/2^9)$	4.441277E-04	-1.138426E-05	-2.215450E-13
$(2^{10}, 0.4/2^{10})$	2.219690E-04	-2.846087E-06	-1.393330E-14
$(2^{11}, 0.4/2^{11})$	1.109608E-04	-7.115230E-07	-1.054712E-15
$(2^{12}, 0.4/2^{12})$	5.547450E-05	-1.778808E-07	-5.551115E-17
$(2^{13}, 0.4/2^{13})$	2.773577E-05	-4.447022E-08	-4.996004E-16
$(2^{14}, 0.4/2^{14})$	1.386752E-05	-1.111755E-08	1.165734E-15
$(2^{15}, 0.4/2^{15})$	6.933665E-06	-2.779388E-09	-5.551115E-16

A log-log graph of $|E_n|$ against h is provided in Figure 2.

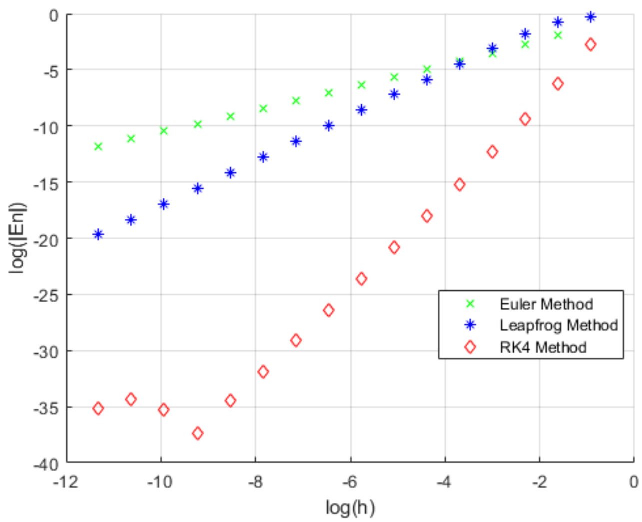

Figure 2: Log-log Graph of $|E_n|$ against h

From Figure 2 it can be seen that, for small h(corresponding to small $\log(h)$), RK4 method is the most accurate method(corresponding to small $\log(|E_n|)$), Leapfrog is the second best method, and Euler method is the least accurate method. This is consistent with the theoretic accuracy of these methods: RK4 has fourth-order accuracy, Leapfrog has second-order accuracy, and Euler has first-order accuracy.

1.2.2 Second-order ODEs

The second part of the report studies the equation

$$\frac{d^2y}{dt^2} + \frac{d}{dt}(\gamma y + \frac{1}{3}\delta^3 y^3) + \Omega^2 y = a\sin(\omega t) \tag{4}$$

where $\gamma, \delta, \Omega, \omega$ and a are real non-negative constants.

Question 5: For $\delta = 0$, $0 < \gamma < 2\Omega$, the equation (4) reduces to

$$\frac{d^2y}{dt^2} + \gamma\frac{dy}{dt} + \Omega^2 y = a\sin(\omega t) \tag{5}$$

Using method of characteristic equation, the general solution is

$$y = e^{-\frac{\gamma}{2}t}(A\cos(\frac{\sqrt{4\Omega^2 - \gamma^2}}{2})t + B\sin(\frac{\sqrt{4\Omega^2 - \gamma^2}}{2})t) + A_s\sin(\omega t - \phi_s) \tag{6}$$

where $A_s = \frac{\frac{\Omega^2 - \omega^2}{|\Omega^2 - \omega^2|}a}{\sqrt{(\Omega^2 - \omega^2)^2 + \omega^2\gamma^2}}$, $\phi_s = \arctan(\frac{\omega\gamma}{\Omega^2 - \omega^2})$, and A, B are some constants to be determined.

Question 6: Take as initial conditions

$$y = \frac{dy}{dt} = 0 \text{ at } t = 0 \tag{7}$$

By substituting $t = 0$ into (6) we find that, for $\omega < 1$,

$$\begin{aligned} y = e^{-\frac{\gamma}{2}t}(&\frac{\sin(\arctan(\frac{\omega\gamma}{1-\omega^2}))}{\sqrt{(1-\omega^2)^2+\omega^2\gamma^2}}\cos(\frac{\sqrt{4-\gamma^2}}{2})t \\ +& \frac{\gamma\sin(\arctan(\frac{\omega\gamma}{1-\omega^2})) - 2\omega\cos(\arctan(\frac{\omega\gamma}{1-\omega^2}))}{\sqrt{(1-\omega^2)^2+\omega^2\gamma^2}\sqrt{4-\gamma^2}}\sin(\frac{\sqrt{4-\gamma^2}}{2})t) \\ +& \frac{\sin(\omega t - \arctan(\frac{\omega\gamma}{1-\omega^2}))}{\sqrt{(1-\omega^2)^2+\omega^2\gamma^2}} \end{aligned} \tag{8}$$

for $\omega = 1$,

$$y = e^{-\frac{\gamma}{2}t}(\frac{1}{\gamma}\cos(\frac{\sqrt{4-\gamma^2}}{2})t + \frac{1}{\sqrt{4-\gamma^2}}\sin(\frac{\sqrt{4-\gamma^2}}{2})t) + \frac{\sin(t-\frac{\pi}{2})}{\gamma} \tag{9}$$

and for $\omega > 1$,

$$\begin{aligned} y = e^{-\frac{\gamma}{2}t}(&-\frac{\sin(\arctan(\frac{\omega\gamma}{1-\omega^2}))}{\sqrt{(1-\omega^2)^2+\omega^2\gamma^2}}\cos(\frac{\sqrt{4-\gamma^2}}{2})t \\ +& \frac{-\gamma\sin(\arctan(\frac{\omega\gamma}{1-\omega^2})) + 2\omega\cos(\arctan(\frac{\omega\gamma}{1-\omega^2}))}{\sqrt{(1-\omega^2)^2+\omega^2\gamma^2}\sqrt{4-\gamma^2}}\sin(\frac{\sqrt{4-\gamma^2}}{2})t) \\ -& \frac{\sin(\omega t - \arctan(\frac{\omega\gamma}{1-\omega^2}))}{\sqrt{(1-\omega^2)^2+\omega^2\gamma^2}} \end{aligned} \tag{10}$$

In the case where $\gamma = 1$, $\omega = \sqrt{3}$ and $(n, h) = (25,0.4)$, the numerical solution Y_n, the analytic solution $y(t_n)$, and the global error $E_n = Y_n - y(t_n)$ are tabulated in Table 5.

Table 5: the RK4 Numerical Solution to (5) with $\gamma = \Omega = a = 1$ and $\omega = \sqrt{3}$

n	Numerical Solution Y_n	Analytic Solution $y(t_n)$	Global Error E_n
1	1.629712E-02	1.622784E-02	6.928741E-05
2	1.070963E-01	1.070522E-01	4.408538E-05
3	2.773505E-01	2.773567E-01	-6.186866E-06
4	4.631759E-01	4.631934E-01	-1.746659E-05
5	5.700184E-01	5.699823E-01	3.616040E-05
.	.	.	.
.	.	.	.
.	.	.	.
21	-1.518953E-01	-1.517073E-01	-1.880424E-04
22	1.017122E-01	1.019410E-01	-2.287525E-04
23	3.120639E-01	3.122205E-01	-1.565432E-04
24	3.815799E-01	3.815868E-01	-6.894475E-06
25	2.774153E-01	2.772664E-01	1.489114E-04

Of course we could carry out the same procedure for $(n, h) = (50{,}0.2)$ and (100,0.1). We see that, for $h = 0.4$, 0.2 and 0.1, the global error has order $10^{-4}, 10^{-5}$ and 10^{-7}. This is consistent with the fact that the RK4 method has fourth-order accuracy.

Question 7:

Using the RK4 method with $h = 0.1$, we can plot the numerical solutions up to $t = 40$ for $\omega = 1$ and various γ. This is shown in Figure 3, 4, 5, and 6.

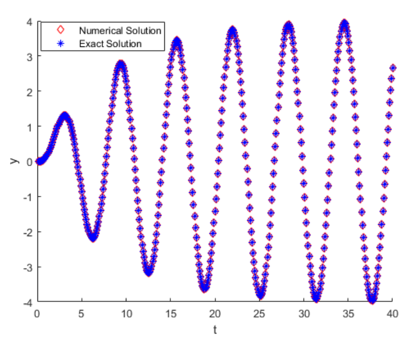

Figure 3: Numerical Solution for $\gamma = 0.25$

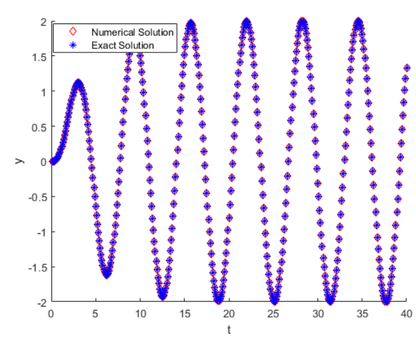

Figure 4: Numerical Solution for $\gamma = 0.5$

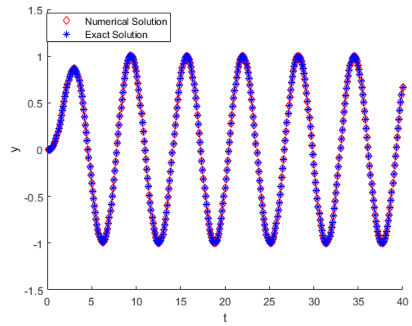

Figure 5: Numerical Solution for $\gamma = 1.0$

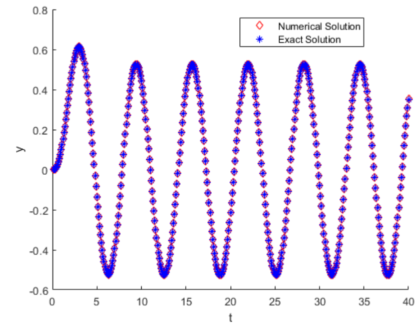

Figure 6: Numerical Solution for $\gamma = 1.9$

Using the same program we could generate plots of the RK4 numerical solutions for the same values of γ with $\omega = 2$. This is shown in Figure 7, 8, 9, 10.

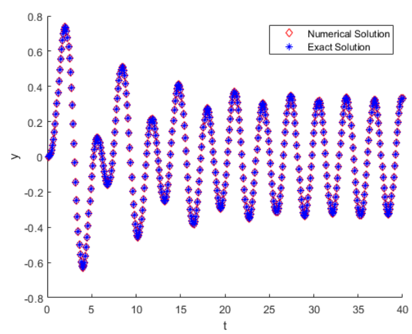

Figure 7: Numerical Solution for $\gamma = 0.25$

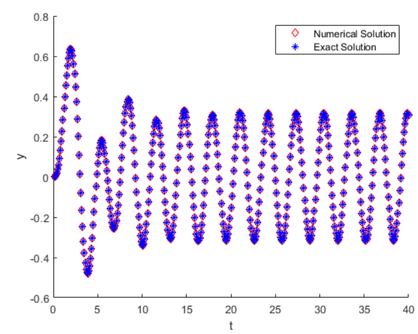

Figure 8: Numerical Solution for $\gamma = 0.5$

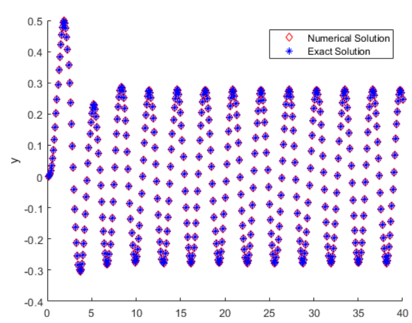

Figure 9: Numerical Solution for $\gamma = 1.0$

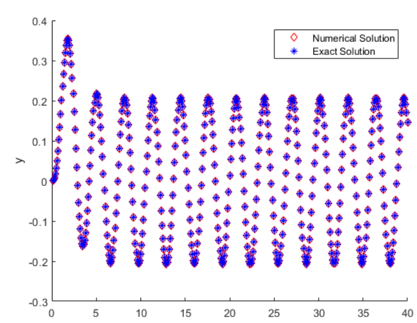

Figure 10: Numerical Solution for $\gamma = 1.9$

It can be seen from these figures that the numerical solutions generated by the

RK4 method agree with the analytic solution, and the behaviour of the plots is sinusoidal for large t. This can be expected: the decaying term with $e^{-\frac{\gamma}{2}t}$ will eventually be negligible, and the sine term will dominate the solution as t becomes large.

Two remarkable differences can be noted from Figure 7, 8, 9, and 10:

• The solutions corresponding to $\omega = 2$ are approximately two times denser than those corresponding to $\omega = 1$. This is anticipated since the forcing term $\sin(\omega t)$ has higher(double) frequency when $\omega = 2$.

• The solutions corresponding to $\omega = 2$ have smaller amplitude. This can be explained by the analytic solution (10) and (9): In the case where $\omega = 2$, the solution has amplitude $|A_s|$, where $|A_s| = \dfrac{1}{\sqrt{9+4\gamma^2}}$. In the case where $\omega = 1$, the solution has amplitude $\frac{1}{\gamma}$.

Question 8:

The RK4 scheme applied to (4) with $\gamma = 0$, $\Omega = \omega = a = 1$ uses the vector field

$$f(t, \mathbf{y}) = \begin{bmatrix} \mathbf{y}_2 \\ -\delta^3 \mathbf{y}_1^2 \mathbf{y}_2 - \mathbf{y}_1 + \sin(t), \end{bmatrix} \tag{11}$$

where $\mathbf{y} = \binom{y}{y'}$ and the subscript i denotes the i^{th} entry of $\mathbf{y}$.

Let y be the exact solution and h be the step size, consider the error introduced in each step of the RK4 scheme:

$$\begin{aligned}
&|(\mathbf{y}_{n+1})_1 - y(t_{n+1})| = |(\mathbf{y}_{n+1})_1 - y(t_n + h)| \\
&= |(\mathbf{y}_n)_1 + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)_1 - y(t_n) - hy'(t_n))| \\
&\leq |(\mathbf{y}_n)_1 - y(t_n)| + h|\frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)_1 - y'(t_n)| \\
&= |(\mathbf{y}_n)_1 - y(t_n)| + \frac{h}{6}|(\mathbf{y}_n)_2 + 2(\mathbf{y}_n + \frac{1}{2}\mathbf{k}_1)_2 + 2(\mathbf{y}_n + \frac{1}{2}\mathbf{k}_2)_2 + (\mathbf{y}_n + \mathbf{k}_3)_2 - 6y'(t_n)| \\
&= |(\mathbf{y}_n)_1 - y(t_n)| + \frac{h}{6}|(\mathbf{k}_1)_2 + (\mathbf{k}_2)_2 + (\mathbf{k}_3)_2| \qquad (12)
\end{aligned}$$

Note that, for the second derivative to exist, y is continuously differentiable. Hence by the Extreme Value Theorem y and y' are bounded, which implies f_2 is also bounded.

Since $(\mathbf{k}_i)_2$'s are just $hf_2(t, \mathbf{y})$ for different t and $\mathbf{y}$, they are bounded, say by hC. Therefore the term $\frac{h}{6}|(\mathbf{k}_1)_2 + (\mathbf{k}_2)_2 + (\mathbf{k}_3)_2|$ is bounded by $\dfrac{h^2C}{2}$. The inequality (12) gives

$$|(\mathbf{y}_{n+1})_1 - y(t_{n+1})| \leq |(\mathbf{y}_n)_1 - y(t_n)| + \frac{h^2C}{2} \tag{13}$$

By applying inequality (13) inductively and use the relation $t_n = nh$ we see that

$$|(\mathbf{y}_{n+1})_1 - y(t_{n+1})| \leq \frac{hC}{2} t_n \tag{14}$$

For moderate forcing and damping(which corresponds to ω and γ being not too large nor too small), $C = 60$ is sufficient. Suppose we want to approximate the solution at $t = 60$ to 1 decimal place, we require that $1800h \leq 10^{-1}$. This is satisfied by choosing $h = 10^{-5}$.

Plots of numerical solutions for various δ are given by Figure 11, 12, 13, and 14:

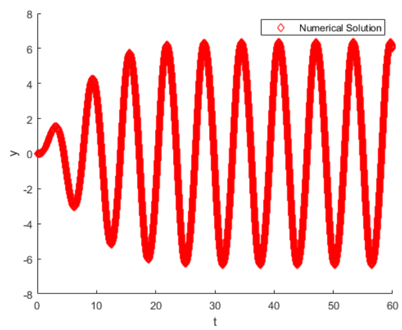

Figure 11: Numerical Solution for $\delta = 0.25$

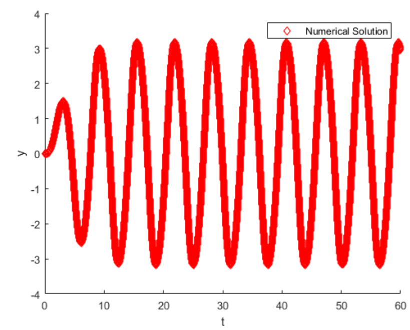

Figure 12: Numerical Solution for $\gamma = 0.5$

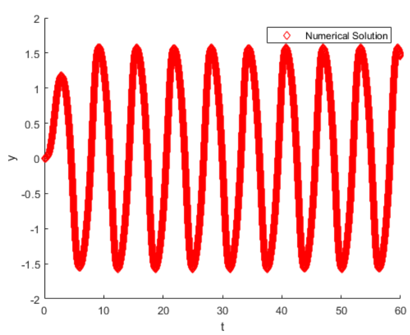

Figure 13: Numerical Solution for $\gamma = 1.0$

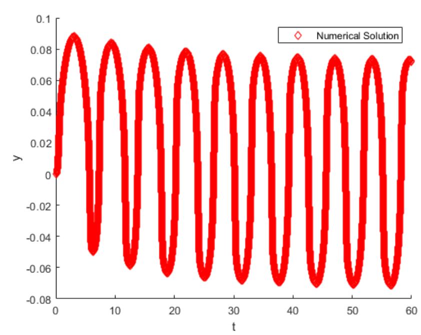

Figure 10: Numerical Solution for $\gamma = 20$

From Figure 11, 12, 13, and 14 we see that the numerical solutions corresponding to non-linear forcing are still approximately sinusoidal but with smaller amplitude. Indeed, with relatively light forcing (corresponding to small γ) the amplitude is slightly smaller, and with huge forcing (corresponding to large γ) the amplitude is significantly smaller.

Reference

Source Code

Simple callings of functions:

```
f = @(x,y) -4*y+3*exp(-x)
g = @(x) exp(-x) - exp(-4*x)
%Q1
[x,K,E,L,R,EE,EL,ER] = Tabulate(0,0,25,0.4,f,g)
[x2,K2,E2,L2,R2,EE2,EL2,ER2] = Tabulate(0,0,50,0.2,f,g)
[x3,K3,E3,L3,R3,EE3,EL3,ER3] = Tabulate(0,0,100,0.1,f,g)
[x4,K4,E4,L4,R4,EE4,EL4,ER4] = Tabulate(0,0,200,0.05,f,g)
%Q3
[x,K,E,L,R,EE,EL,ER] = Tabulate(0,0,10,0.4,f,g)
xlabel('x');
ylabel('Numerical Solution')
hold on
plot(x,E,'gx')
plot(x,R,'b*')
plot(x,K,'rd')
hold off
grid on
```

Code for Programming Task 1(Euler):

```
function [Y] = Euler(x0,Y0,n,h,f)%set initial value to be x1 Y1 for consistency
of indices in the program
x = zeros(1,n);
Y = zeros(1,n);
for i = 1:n
format long
x(i) = x0 + i * h;
end
format long
```

```
Y(1) = Y0 + h * f(x0,Y0);
for i = 2:n
format long Y(i) = Y(i-1) + h * f(x(i-1),Y(i-1));
end
x;
Y;
end
```

Code for Programming Task 1(Leapfrog):

```
function [Y] = Leapfrog(x0,Y0,n,h,f)
x = zeros(1,n);
Y = zeros(1,n);
for i = 1:n
format long
x(i) = x0 + i * h;
end
format long
Y(1) = Y0 + h * f(x0,Y0);
Y(2) = Y0 + 2 * h * f(x(1),Y(1));
for i = 2:n-1
Y(i+1) = Y(i-1) + 2 * h * f(x(i),Y(i));
end
x;
Y;
end
```

Code for Programming Task 1(RK4):

```
function [Y] = RK4(x0,Y0,n,h,f)
x = zeros(1,n);
Y = zeros(1,n);
for i = 1:n
format long
x(i) = x0 + i * h;
end
format long
a = h * f(x0,Y0);
format long
b = h * f(x0 + 0.5 * h,Y0 + 0.5 * a);
format long
c = h * f(x0 + 0.5 * h,Y0 + 0.5 * b);
format long
d = h * f(x0 + h,Y0 + c);
format long
```

```
Y(1) = Y0 + (a + 2 * b + 2 * c + d)/6;
for i = 2:n
format long
a = h * f(x(i-1),Y(i-1));
format long
b = h * f(x(i-1) + 0.5 * h,Y(i-1) + 0.5 * a);
format long
c = h * f(x(i-1) + 0.5 * h,Y(i-1) + 0.5 * b);
format long
d = h * f(x(i-1) + h,Y(i-1) + c);
format long
Y(i) = Y(i-1) + (a + 2 * b + 2 * c + d)/6;
end
x;
Y;
end
```

Code for Question 1 and 3:

```
function [x,K,E,L,R,EE,EL,ER] = Tabulate(x0,Y0,n,h,f,g)%K is the exact solution, x is xn.
E = Euler(x0,Y0,n,h,f);
L = Leapfrog(x0,Y0,n,h,f);
R = RK4(x0,Y0,n,h,f);
K = zeros(1,n);
for i = 1:n
format long
x(i) = x0 + i * h;
K(i) = g(x(i));
end
EE = E-K;
EL = L-K;
ER = R-K;
end
```

Code for Question 4:

```
function [T] = Error(f,g)
T = zeros(16,3);
for i=0:15
[x,K,E,L,R,EE,EL,ER] = Tabulate(0,0,2 ^ i,0.4/(2^i),f,g);
T(i+1,1) = EE(2^i);
T(i+1,2) = EL(2^i);
T(i+1,3) = ER(2^i);
end
```

Code for Programming Task 2(RK4):

```
function [x,Y] = RK42D(x0,Y0,n,h,f)
x = zeros(1,n);
Y = zeros(1,n);
S = Y0;
for i = 1:n
format long
x(i) = x0 + i * h;
end
format long
a = h * f(x0,Y0);
format long
b = h * f(x0 + 0.5 * h,Y0 + 0.5 * a);
format long
c = h * f(x0 + 0.5 * h,Y0 + 0.5 * b);
format long
d = h * f(x0 + h,Y0 + c);
format long
S = S + (a + 2 * b + 2 * c + d)/6;
Y(1) = S(1);
for i = 2:n
format long
a = h * f(x(i-1),S);
format long
b = h * f(x(i-1) + 0.5 * h,S + 0.5 * a);
format long
c = h * f(x(i-1) + 0.5 * h,S + 0.5 * b);
format long
d = h * f(x(i-1) + h,S + c);
format long
S = S + (a + 2 * b + 2 * c + d)/6;
Y(i) = S(1);
end
x;
Y;
end
```

Code for Question 6:

function [f] = Analytic(a,b)%a for gamma b for omega
if b == 1
$f = @(x)exp(-a*x/2)*((1/a)*(cos(sqrt(4-a^2)*x/2))+(1/(sqrt(4-a^2)))*(sin(sqrt(4-a^2)*x/2)))+sin(x-(pi/2))/a;$

$elseif b < 1$
$l = atan(a * b/(1 - b * b));$
$f = @(x)exp(-a * x/2) * ((sin(l) * cos(sqrt(4 - a^2) * x/2)/sqrt((1 - b^2)^2 + a^2 * b^2)) + ((a * sin(l) - 2 * b * cos(l)) * sin(sqrt(4 - a^2) * x/2)/(sqrt((1 - b^2)^2 + a^2 * b^2) * sqrt(4 - a^2)))) + (sin(b * x - l)/sqrt((1 - b^2)^2 + a^2 * b^2));$
else
$l = atan(a * b/(1 - b * b));$
$f = @(x) - (exp(-a * x/2) * ((sin(l) * cos(sqrt(4 - a^2) * x/2)/sqrt((1 - b^2)^2 + a^2 * b^2)) + ((a * sin(l) - 2 * b * cos(l)) * sin(sqrt(4 - a^2) * x/2)/(sqrt((1 - b^2)^2 + a^2 * b^2) * sqrt(4 - a^2)))) + (sin(b * x - l)/sqrt((1 - b^2)^2 + a^2 * b^2)));$
end
end

```
h = @(x,y)[y(2),-y(2)-y(1)+sin(sqrt(3)*x)];
[x,X] = RK42D(0,[0,0],25,0.4,h);
[y,Y] = RK42D(0,[0,0],50,0.2,h);
[z,Z] = RK42D(0,[0,0],100,0.1,h);
f = Analytic(1,sqrt(3));

EX =zeros(1,25);
fX = zeros(1,25);
EY = zeros(1,50);
fY = zeros(1,50);
EZ = zeros(1,100);
fZ = zeros(1,100);

for i = 1:25
fX(i) = f(x(i));
EX(i) = X(i) - f(x(i));
end

for i = 1:50
fY(i) = f(y(i));
EY(i) = Y(i) - f(y(i));
end

for i = 1:100
fZ(i) = f(z(i));
EZ(i) = Z(i) - f(z(i));
end
```

Code for Question 7:

```
function Question7(a,b)%a for gamma b for omega
h = @(x,y)[y(2),-a*y(2)-y(1)+sin(b*x)];
f = Analytic(a,b)
```

```
[x, X] = RK42D(0,[0,0],400,0.1,h)
EX =zeros(1,400);
fX = zeros(1,400);
xlabel('t')
ylabel('y')
for i = 1:400
fX(i) = f(x(i))
EX(i) = X(i) - f(x(i));
end
hold on
plot(x,X,'rd')
plot(x,fX,'b*')
hold off
```

Code for Question 8:

```
function Question8(a)%a delta
h = @(x,y)[y(2),-(a ^ 3)*((y(1)) ^ 2)*(y(2)) - y(1) + sin(x)];
[x,X] = RK42D(0,[0,0],6*10 ^ 6,10 ^ (-5),h);
xlabel('t')
ylabel('y')
hold on
plot(x,X,'rd')
hold off
```