Homework

Jingcheng Lu [42326170], Yunpeng Hu [13606024]

March 26, 2018

1 Task1

1.1 Euler method

First, we assume the mass is 1(same for below questions), then we can get the following equation:

$$\begin{aligned}&x_{i+1} = x_i + hv_i\\&v_i = v_{i-1} + ha_{i-1}\\&a_{i-1} = x''_{i-1} = -x_{i-1}\end{aligned}$$

Also we know:

$$\begin{aligned}&v_{i-1} = x'_{i-1}\\&x_{i+1} = x_i + h \cdot (v_{i-1} + ha_{i-1})\\&= x_i + h \cdot (x'_{i-1} - h \cdot x_{i-1})\\&= x_i + h \cdot x'_{i-1} - h^2 \cdot x_{i-1})\end{aligned}$$

So, we can conclude that:

$$x_{i+1} = x(t_i) + h \cdot x'(t_{i-1}) - h^2 \cdot x(t_{i-1}))$$

Applying Taylor expansion for $x(t_{i+1})$ at t_i, we can get:

$$\begin{aligned}&x(t_{i+1}) = x(t_i) + h \cdot x'(t_i) + \tfrac{h^2}{2!}x''(t_i) + O(h^3)\\&x(t_{i+1}) - x_{i+1} = h(x'(t_i) - x'(t_{i-1})) + \tfrac{h^2}{2}x''(t_i) + h^2x(t_{i-1}) + O(h^3)\end{aligned}$$

Applying Taylor expansion for $x'(t_i)$ at t_{i-1}, we can get:

$$x'(t_i) = x'(t_{i-1}) + h \cdot x''(t_{i-1}) + O(h^2)$$

Then we applying above equation:

$$\begin{aligned}&x(t_{i+1}) - x_{i+1} = h(h \cdot x''(t_{i-1}) + O(h^2)) + \tfrac{h^2}{2}x''(t_i) + h^2x(t_{i-1}) + O(h^3)\\&= h^2(x''(t_{i-1}) + \tfrac{1}{2}x''(t_i) + x(t_{i-1})) + O(h^3) = O(h^2)\end{aligned}$$

So the error of Euler method is $O(h^2)$. Thus the global error is $O(h)$.
The picture 1 shown below is the result of Euler method.
As we can see in the picture, the slope is 1, so again it proves our conclusion that Euler method is 1st order.

1.2 Leapfrog method

First, we can get:

$$\begin{aligned}x_{i+1} &= x_i + hv_{i+0.5}\\ v_{i+0.5} &= v_{i-0.5} + ha_i\end{aligned}$$

so:

$$\begin{aligned}x_{i+1} &= x_i + h\cdot(v_{i-0.5} + ha_i)\\ &= x_i + h\cdot x'(t_{i-0.5}) + h^2\cdot x''(t_{i-0.5})\end{aligned}$$

Applying Taylor expansion for $x(t_{i+1})$ at t_i, we can get:

$$\begin{aligned}&x(t_{i+1}) = x(t_i) + h\cdot x'(t_i) + \tfrac{h^2}{2}x''(t_i) + \tfrac{h^3}{3}x'''(t_i) + O(h^4)\\ &x(t_{i+1}) - x_{i+1} = h\cdot(x'(t_i) - x'(t_{i-0.5})) + \tfrac{h^2}{2}x''(t_i) - h^2\cdot x''(t_{i-0.5}) + \tfrac{h^3}{3}x'''(t_i) + O(h^4)\end{aligned}$$

Applying Taylor expansion for $x'(t_i)$ at $t_{i-0.5}$, we can get:

$$\begin{aligned}&x'(t_i) = x'(t_{i-0.5}) + \tfrac{h}{2}x''(t_{i-0.5}) + (\tfrac{h}{2})^2\tfrac{x'''(t_{i-0.5})}{2!} + O(h^3)\\ &x(t_{i+1}) - x_{i+1} = h(\tfrac{h}{2}x''(t_{i-0.5}) + (\tfrac{h}{2})^2\tfrac{x'''(t_{i-0.5})}{2!} + O(h^3)) + \tfrac{h^2}{2}x''(t_i) - h^2\cdot x''(t_{i-0.5}) + \tfrac{h^3}{3}x'''(t_i) + O(h^4)\end{aligned}$$

Applying Taylor expansion for $x''(t_i)$ at $t_{i-0.5}$, we can get:

$$x''(t_i) = x''(t_{i-0.5}) + \frac{h}{2}x''(t_{i-0.5}) + O(h^2)$$

Thus we can get:

$$\begin{aligned}x(t_{i+1}) - x_{i+1} &= h^3(\tfrac{3}{4}x'''(t_{i-0.5}) + \tfrac{1}{6}x'''(t_i)) + O(h^4)\\ &= O(h^3)\end{aligned}$$

Thus the step error of Leapfrog is $O(h^3)$, then the global error will be $O(h^2)$.

The picture 2 shown below is the result of Leapfrog method.

As we can see in the picture, the slope is 2, so it proves our conclusion that Leapfrog method is 2nd order.

The irregular shape at the beginning is because of the machine error for using single.

1.3 Runge-Kutta method

The picture 3 shown below is the result of Runge-Kutta method.

As we can see in the picture, the slope is 4, so it proves our conclusion that Leapfrog method is 4th order.

The irregular shape at the beginning is because of the machine error for using single.

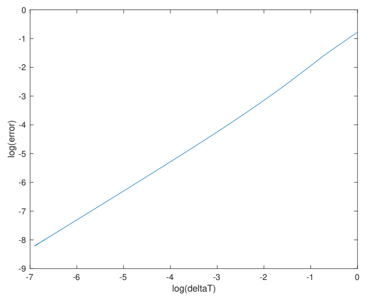

Figure 1: Euler method

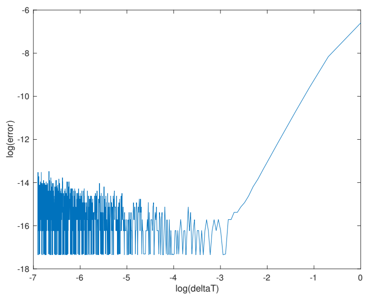

Figure 2: leapfrog method

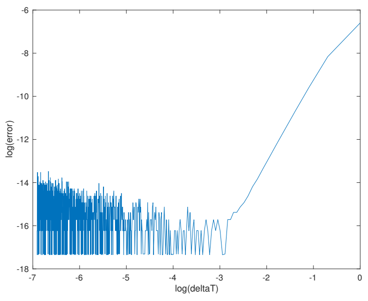

Figure 3: Runge-Kutta method

2 Task 2

In this task, we set the total time(duration) to 1000 and time step to 0.01(using smaller number the error would be smaller too, but would slow down the computer).
The picture 4 shown below is the result of Euler method.
The picture 5 shown below is the result of Leapfrog method.
The picture 6 shown below is the result of Runge-Kutta method.
AS the result shown in these pictures, we can see that when the time step is same, the Runge-Kutta method(which error is about 10^{-5}) is better then Leapfrog method(which error is about 10^{-3}); and the Leapfrog method is better then Euler method(which error is about 10^{2}).

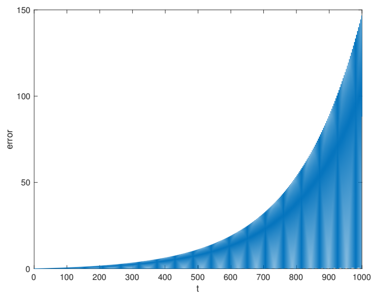

Figure 4: Euler method

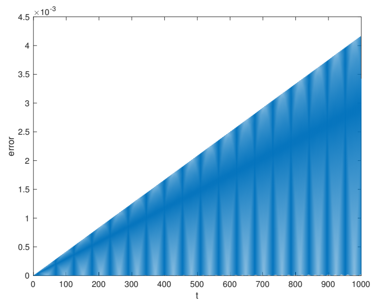

Figure 5: leapfrog method

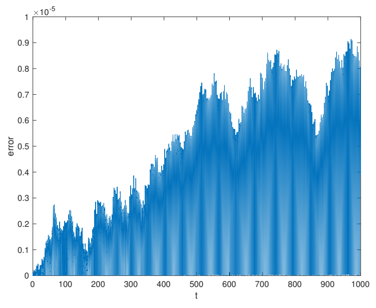

Figure 6: Runge-Kutta method

3 Task 3

In this task, we calculate the energy when using 3 different method and compare them with the real energy value(0.5). Generally, the Euler method is the worst and will violate conservation of energy by at least 100% even when time step is 0.001. For Leapfrog method and Runge-Kutta method, we set duration to 10000000, but it runs forever so we didn't find which one is better. But from the picture, we can see that the error of Runge-Kutta keep getting larger but the error of Leapfrog method it actually come back forth from 0.5 which is the exact value of energy. so we guess that after the duration getting very large the Leapfrog method may get a better result(smaller error) then Runge-Kutta method.

3.1 Euler method

For Euler method, we set the duration to 1000 and time step to 3 different value of $[0.01, 0.005, 0.001]$. The picture 7 shown below is the result of Euler method.

3.2 Leapfrog method

For Euler method, we set the duration to 100000 and time step to 3 different value of $[0.01, 0.005, 0.001]$. The picture 8 shown below is the result of Leapfrog method.

3.3 Runge-Kutta method

For Euler method, we set the duration to 100000 and time step to 3 different value of $[0.01, 0.005, 0.001]$. The picture 9 shown below is the result of Runge-Kutta method.

A task1 code

A.1 Euler method

```
clc;                                                    % Clears the
    screen
clear all;

totalT = 1;
k = 1;
```

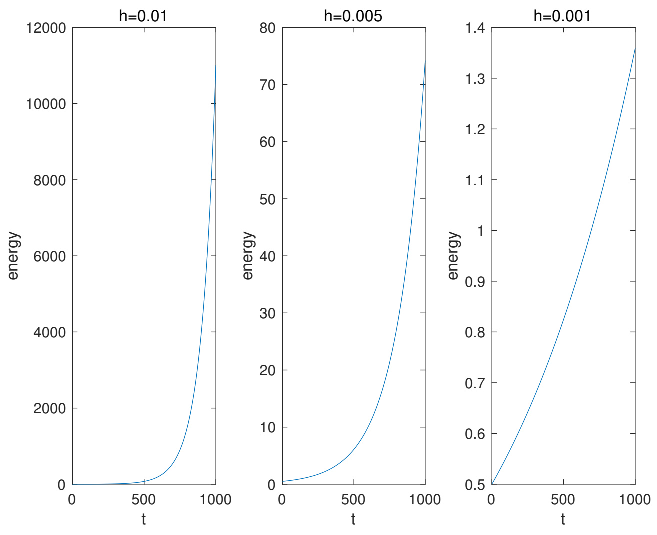

Figure 7: Euler method

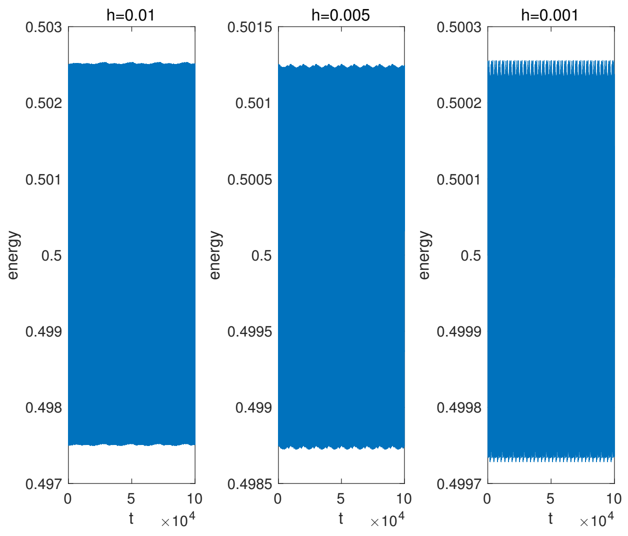

Figure 8: leapfrog method

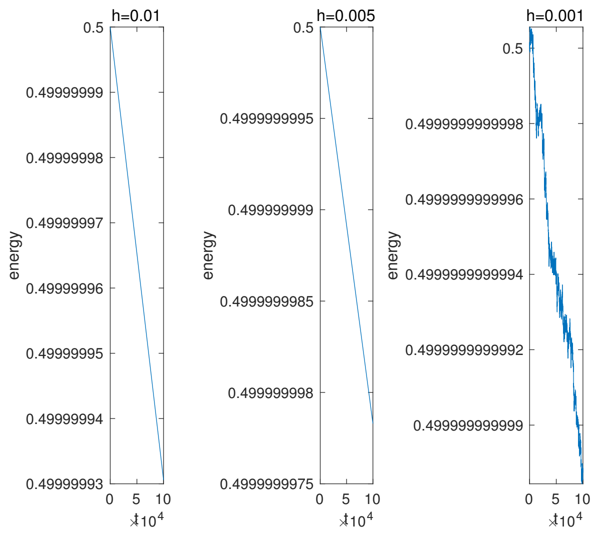

Figure 9: Runge-Kutta method

```
for n = 1 : 1000
    deltaT(n) =single(totalT / n);
    x = 1;
    v = 0;
    for i = 1 : n
        a = single(-k * x);
        x = single(x + v * deltaT(n));
        v = single(v + a * deltaT(n));
    end
    error(n) = abs(cos(totalT) - x);
end

plot(log(deltaT), log(error));
xlabel('log(deltaT)');ylabel('log(error)');
p = polyfit(log(deltaT), log(error),1);
p(1)
```

A.2 Leapfrog method

```
clc;                                                  % Clears the
   screen
clear all;

totalT = 1;
k = 1;

for n = 1 : 1000
    deltaT(n) =single(totalT / n);
    x = 1;
    v = 0;
    a = single(- k * x);
    v = single(v + 1/2 *deltaT(n) * a);
    for i = 1 : n
        x = single(x + v * deltaT(n));
        a = single(-k * x);
        v = single(v + a * deltaT(n));
    end
    error(n) = abs(cos(totalT) - x);
end

x=log(deltaT);
y=log(error);
plot(x,y);
xlabel('log(deltaT)');ylabel('log(error)');
index = find(x>-4);
p = polyfit(x(index), y(index),1);
p(1)
```

A.3 Runge-Kutta method

```
clc;                                                    % Clears the
   screen
clear all;

totalT = 1;
k = 1;

for n = 1 : 1000
    deltaT(n) =single(totalT / n);
    x = 1;
    v = 0;
    for i = 1 : n
        v_1 = v;
        a_1 = -k * x;

        v_2 = v + 0.5 * deltaT(n) * a_1;
        a_2 = -k * (x + deltaT(n) / 2 * v_1);

        v_3 = v + deltaT(n) / 2 * a_2;
        a_3 = -k * (x + deltaT(n) / 2 * v_2);

        v_4 = v + deltaT(n) * a_3;
        a_4 = -k * (x + deltaT(n) * v_3);

        x = x + deltaT(n) / 6 * ( v_1 + 2 * v_2 + 2 * v_3 + v_4);
        v = v + deltaT(n) / 6 * ( a_1 + 2 * a_2 + 2 * a_3 + a_4);
    end
    error(n) = abs(cos(totalT) - x);
end
```

```
x=log(deltaT);
y=log(error);
plot(x,y);
xlabel('log(deltaT)');ylabel('log(error)');
index = find(x>-3);
p = polyfit(x(index), y(index),1);
p(1)
```

B task2 code

B.1 Euler method

```
clc;                                            % Clears the
   screen
clear all;

totalT = 1000;
k = 1;

x(1) = 1;
v = 0;
h = 0.01;
t(1) = 0;
n = totalT / h;
error(1) = 0;
for i = 2 : n
    a = single(-k * x(i - 1));
    x(i) = single(x(i - 1) + v * h);
    v = single(v + a * h);
    t(i) = t(i - 1) + h;
    error(i) = abs(cos(t(i)) - x(i));
end
plot(t, error);
xlabel('t');ylabel('error');
```

B.2 Leapfrog method

```
clc;                                            % Clears the
   screen
clear all;

totalT = 1000;
k = 1;

x(1) = 1;
v = 0;
h = 0.01;
t(1) = 0;
```

```
n = totalT / h;
a = single(- k * x(1));
v = single(v + 1/2 *h * a);
error(1) = 0;
for i = 2 : n
    x(i) = single(x(i - 1) + v * h);
    a = single(-k * x(i));
    v = single(v + a * h);
    t(i) = t(i - 1) + h;
    error(i) = abs(cos(t(i)) - x(i));
end
plot(t, error);
xlabel('t');ylabel('error');
```

B.3 Runge-Kutta method

```
clc;                                                    % Clears the
   screen
clear all;

totalT = 1000;
k = 1;

x(1) = 1;
v = 0;
h = 0.01;
t(1) = 0;
n = totalT / h;
error(1) = 0;
for i = 2 : n
    v_1 = single(v);
    a_1 = single(-k * x(i - 1));
    v_2 = single(v + h / 2 * a_1);
    a_2 = single(-k * (x(i - 1) + h / 2 * v_1));
    v_3 = single(v + h / 2 * a_2);
    a_3 = single(-k * (x(i - 1) + h / 2 * v_2));
    v_4 = single(v + h * a_3);
    a_4 = single(-k * (x(i - 1) + h * v_3));
    x(i) = single(x(i - 1) + h / 6 * ( v_1 + 2 * v_2 + 2 * v_3 +
       v_4));
    v = single(v + h / 6 * ( a_1 + 2 * a_2 + 2 * a_3 + a_4));
    t(i) = t(i - 1) + h;
    error(i) = abs(cos(t(i)) - x(i));
end
plot(t, error);
xlabel('t');ylabel('error');
```

C task3 code

C.1 Euler method

```
clc;                                                         % Clears the
   screen
clear all;

totalT = 1000;
k = 1;

figureN = 1;
for h = [0.01, 0.005, 0.001]
    x = 1;
    v = 0;
    n = totalT / h;
    t(1) = 0;
    energy(1) = 0.5;
    for i = 2 : n
        a = single(-k * x);
        x = single(x + v * h);
        v = single(v + a * h);
        energy(i) = 0.5 * v * v + 0.5 * k * x * x;
        t(i) = t(i - 1) + h;
    end
    subplot(1,3,figureN);
    plot(t,energy);
    xlabel('t');ylabel('energy');
    str=['h=',num2str(h)];
    title(str);
    figureN = 1+figureN;
end
```

C.2 Leapfrog method

```
clc;                                                         % Clears the
   screen
clear all;

totalT = 100000;
k = 1;

figureN = 1;
for h = [0.01, 0.005, 0.001]
    x = 1;
    v = 0;
    n = totalT / h;
    t(1) = 0;
    energy(1) = 0.5;
    a = single(- k * x);
    v = single(v + 1/2 *h * a);
    for i = 2 : n
        x = single(x + v * h);
        a = single(-k * x);
        v = single(v + a * h);
        energy(i) = 0.5 * v * v + 0.5 * k * x * x;
        t(i) = t(i - 1) + h;
    end
    subplot(1,3,figureN);
    plot(t,energy);
    xlabel('t');ylabel('energy');
    str=['h=',num2str(h)];
    title(str);
    figureN = 1+figureN;
end
```

C.3 Runge-Kutta method

```
clc;                                                        % Clears the
   screen
clear all;

totalT = 100000;
k = 1;

figureN = 1;
for h = [0.01, 0.005, 0.001]
    x = 1;
    v = 0;
    n = totalT / h;
    t(1) = 0;
    energy(1) = 0.5;
    for i = 2 : n
        v1 = v;
        a1 = -k * x;
        v2 = v + h / 2 * a1;
        a2 = -k * (x + h / 2 * v1);
        v3 = v + h / 2 * a2;
        a3 = -k * (x + h / 2 * v2);
        v4 = v + h * a3;
        a4 = -k * (x + h * v3);
        x = x + h / 6 * ( v1 + 2 * v2 + 2 * v3 + v4);
        v = v + h / 6 * ( a1 + 2 * a2 + 2 * a3 + a4);
        t(i) = t(i - 1) + h;
        energy(i) = 0.5 * v * v + 0.5 * x * x;
    end
    subplot(1,3,figureN);
    plot(t,energy);
    xlabel('t');ylabel('energy');
    str=['h=',num2str(h)];
    title(str);
    figureN = 1+figureN;
end
```

C.4 Compare 3 method

```
clc;                                                    % Clears the
   screen
clear all;
k = 1;
s = ones(1,3);
h=0.01;
for totalT = [10000000];
%       figureN=1;
%       for h = [0.01]
          x_e = 1;
          x_lf = 1;
          x_rk = 1;
          v_e = 0;
          v_lf = 0;
          v_rk = 0;
          t(1) = 0;
          n = totalT / h;

          energyEuler(1) = 0.5;
          energyLf(1) = 0.5;
          energyRk(1) = 0.5;

          a_lf = single(- k * x_lf);
          v_lf = single(v_lf + 1/2 *h * a_lf);
          for i = 2 : n
              %1
              if s(1)==1
                  a_e = single(-k * x_e);
                  x_e = single(x_e + v_e * h);
                  v_e = single(v_e + a_e * h);
                  energyEuler(i) = 0.5 * v_e * v_e + 0.5 * k * x_e
                     * x_e;
              end
              %2
              if s(2)==1
                  x_lf = single(x_lf + v_lf * h);
                  a_lf = single(-k * x_lf);
                  v_lf = single(v_lf + a_lf * h);
                  energyLf(i) = 0.5 * v_lf * v_lf + 0.5 * k * x_lf
                     * x_lf;
              end
```

```
            %3
            if s(3)==1
                v_1 = v_rk;
                a_1 = -k * x_rk;
                v_2 = v_rk + h / 2 * a_1;
                a_2 = -k * (x_rk + h / 2 * v_1);
                v_3 = v_rk + h / 2 * a_2;
                a_3 = -k * (x_rk + h / 2 * v_2);
                v_4 = v_rk + h * a_3;
                a_4 = -k * (x_rk + h * v_3);
                x_rk = x_rk + h / 6 * ( v_1 + 2 * v_2 + 2 * v_3 +
                    v_4);
                v_rk = v_rk + h / 6 * ( a_1 + 2 * a_2 + 2 * a_3 +
                    a_4);
                energyRk(i) = 0.5 * v_rk * v_rk + 0.5 * x_rk *
                   x_rk;
            end
            t(i) = t(i - 1) + h;
            if abs(max(energyEuler)-0.5)/0.5>1
                s(1)=0;
            end
            if abs(max(energyLf)-0.5)/0.5>1
                s(2)=0;
            end
            if abs(max(energyRk)-0.5)/0.5>1
                s(3)=0;
            end

            if(sum(s)<2)
                break;
            end
%           end
%       subplot(1,3,figureN);
%       plot(t,energyEuler);
%       xlabel('t');ylabel('energy');
%       str=['h=',num2str(h)];
%       title(str);
%       figureN = 1+figureN;
        end
        if(sum(s)<2)
            break;
        end
end
```