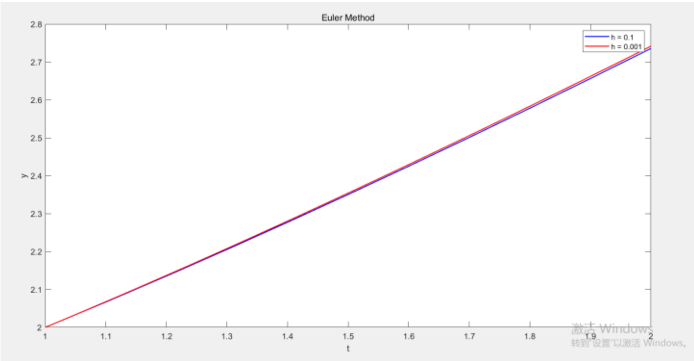
MATLAB 第 7 次作业参考答案

几点反馈：

- 作业的参考答案来自同学们的作业，只是稍作调整，为保护隐私，隐去姓名。
- 对于绘制多条图线比较的题目，大家可以注意一下用不同的线型（宽）、marker 来区分，不要都叠在一起没法区分。

第一题

做出图像如下：



从图中可以看出，随着 t 的增大，选用不同步长进行求解得到的结果间的差距也会变大，即误差会具有一定的累积效应。

代码：Hw7_1_1.m

```
h1 = 0.1;
h2 = 0.001;
t1 = 1:h1:2;
t2 = 1:h2:2;
y_prime = @(t,y) (1+t)./(1+y);
y1(1) = 2;
y2(1) = 2;
for i = 1:length(t1)-1
    y1(i+1) = y1(i)+h1*y_prime(t1(i),y1(i));
end
for i = 1:length(t2)-1
    y2(i+1) = y2(i)+h2*y_prime(t2(i),y2(i));
end
figure;
plot(t1,y1,'b','LineWidth',1.2);
hold on;
plot(t2,y2,'r','LineWidth',1.2);
xlabel('t');
ylabel('y');
title('Euler Method');
legend('h = 0.1','h = 0.001');
```

(2)用分离变量法求出解析解，要求给出求解过程。取步长为 0.25，使用 Euler’s Method 进行求解，在同一幅图中作出解析解与 Euler 解。在另一幅图中作出 Euler 解的 Local Truncation Error 和 Global Truncation Error。要求给出代码与两幅图像，注意线型、图例、坐标轴和标题。

$y^{'}=\frac{dy}{dt}=\frac{1+t}{1+y}$，$(1+y)dy=(1+t)dt$，两边取积分得 $\int(1+y)dy=\int(1+t)dt$，

解得 $\frac{(y+1)^2}{2}=\frac{(t+1)^2}{2}+C$，代入初值解得 $(y+1)^2=(t+1)^2+5$，根据初值

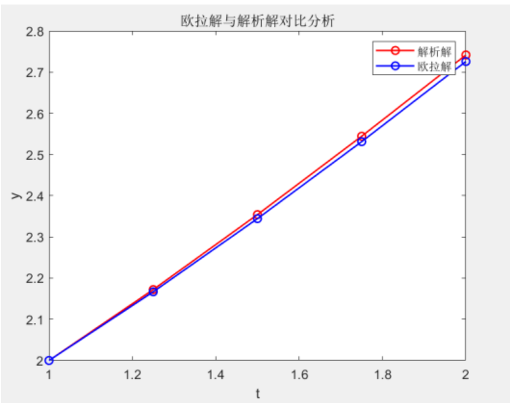
$y(1)>0, t(1)>0,$ 则 $y^{'}(1)>0,$ 则 y 为增函数，又根据 y 的连续性，在 $1\le t\le 2$ 上 $y>0$

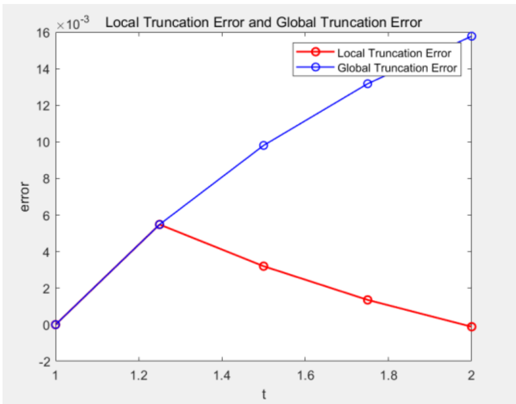
因此解析解为 $y=\sqrt{(1+t)^2+5}-1$

又记 $f(t,y)=y^{'}, \frac{\partial f}{\partial y}=-\frac{1+t}{(1+y)^2}$，则 $\left|\frac{\partial f}{\partial y}\right|\le\frac{3}{4}$，即 f 在所求解区域上满足 Lipschitz 条件，$L=\frac{3}{4}$

因此 $|y_{i+1}-z_{i+1}|=e^{Lh}\lfloor y_i-w_i\rfloor$，$g_{i+1}=|y_{i+1}-w_{i+1}|$，$e_{i+1}=|z_{i+1}-w_{i+1}|=g_{i+1}-|y_{i+1}-z_{i+1}|$

做出解析解与欧拉解的图像和误差图如下：





从图中可以看出，随着 t 的增加，Local Truncation Error 是逐渐减小的，而 Global Truncation Error 由于误差的累积效应却在逐渐增大。

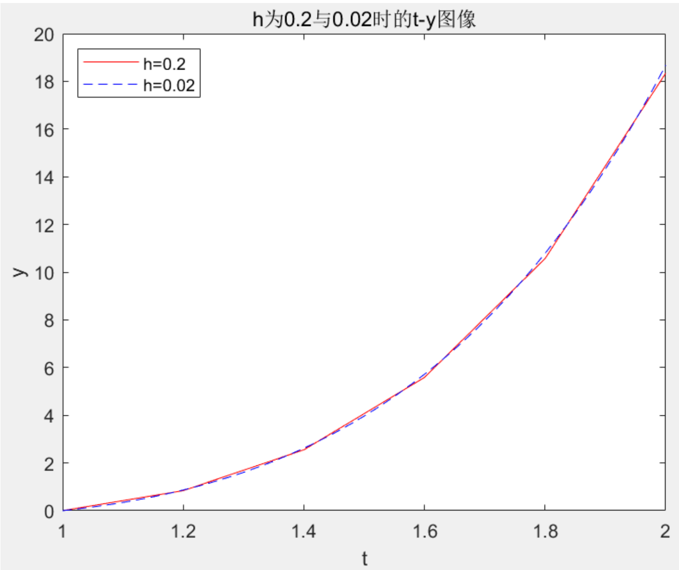
代码：Hw7_1_2.m

```
h = 0.25;
y_prime = @(t,y) (1+t)./(1+y);
y = @(t) sqrt((1+t).^2+5)-1;
t = 1:h:2;
y_t = y(t);
y_e = zeros(1,length(t));
y_e(1) = 2;
for i = 1:length(t)-1
    y_e(i+1) = y_e(i) + h*y_prime(t(i),y_e(i));
end
figure;
plot(t,y_t,'r-o','LineWidth',1.2);
hold on;
plot(t,y_e,'b-o','LineWidth',1.2);
```

```
xlabel('t');
ylabel('y');
title('欧拉解与解析解对比分析');
legend('解析解','欧拉解');
g = abs(y_t-y_e);
L = 3/4;
Local_error = zeros(1,length(t));
for i = 1:length(t)-1
    Local_error(i+1) = g(i+1) - exp(L*h)*g(i);
end
figure;
plot(t,Local_error,'r-o','LineWidth',1.4);
hold on;
plot(t,g,'b-o','LineWidth',1);
xlabel('t');
ylabel('error');
title('Local Truncation Error and Global Truncation Error');
legend('Local Truncation Error','Global Truncation Error');
```

第二题

（1）



Trapezoid.m

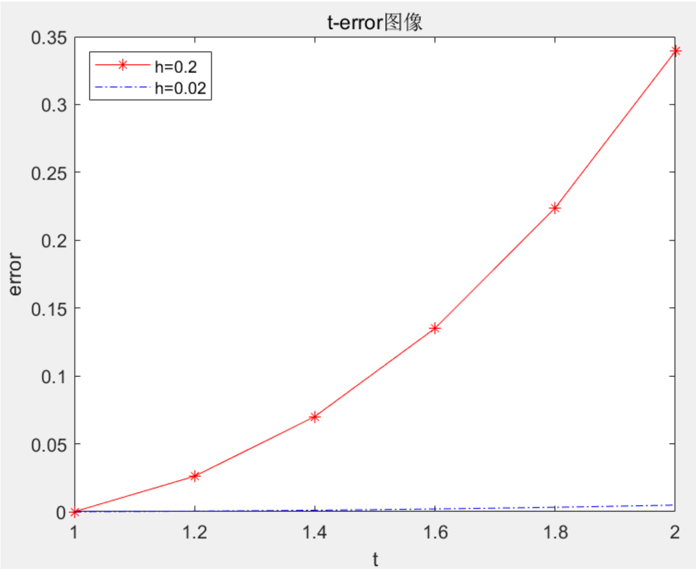
```
function [t,w]=Trapezoid(f,t0,tf,y0,h)
t=t0:h:tf;
if t(end)~=tf
    t(end+1)=tf;
end
n=length(t);
w=[y0,zeros(1,n-1)];
for i=1:n-1
    if i==n-1
        h=t(end)-t(end-1);
    end
```

```
    w(i+1)=w(i)+h/2*(f(t(i),w(i))+f(t(i+1),w(i)+h*f(t(i),w(i))));
end
end
```

Hw7_2_1.m

```
clear,clc;
f=@(t,y)2./t.*y+t.^2.*exp(t);
t0=1;tf=2;y0=0;
[t1,w1]=Trapezoid(f,t0,tf,y0,0.2);
[t2,w2]=Trapezoid(f,t0,tf,y0,0.02);
plot(t1,w1,'-r',t2,w2,'--b');
legend('h=0.2','h=0.02','Location','northwest');
xlabel 't';ylabel 'y';
title('h 为 0.2 与 0.02 时的 t-y 图像');
```

（2）



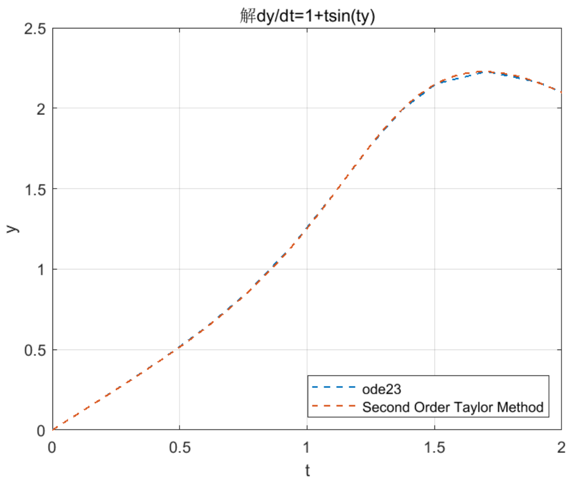
Hw7_2_2.m

```
syms y(t);
eqn = diff(y,t) == 2/t*y+t^2*exp(t);
cond = y(1) == 0;
ySol(t) = dsolve(eqn,cond);
y_exa=matlabFunction(ySol);
y=plot(t1,y_exa(t1)-w1,'-*r',t2,y_exa(t2)-w2,'-.b');
legend('h=0.2','h=0.02','Location','northwest');
xlabel 't';ylabel 'error';
title('t-error 图像');
```

第三题

（1）Euler's Method 是 Taylor Method 的一阶形式。

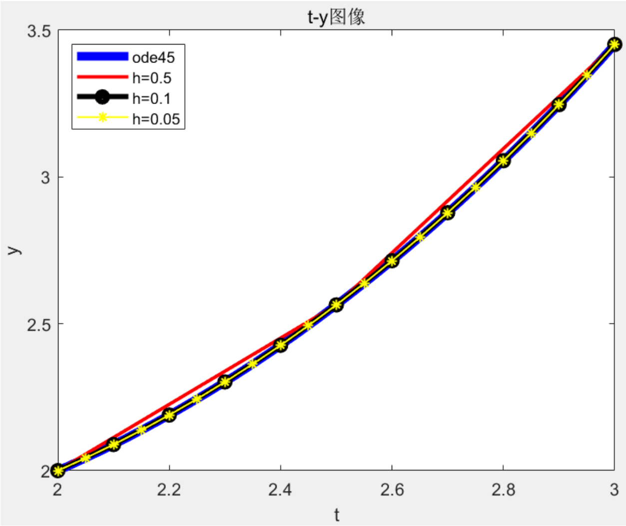
（2）得到的结果如下图所示。



Hw7_3.m

```
tspan = [0 2];
y0 = 0;
[t1,y1] = ode23(@(t,y) 1+t.*sin(t.*y), tspan, y0); %ode23 求解方程
h2 = 0.02;
t2 = 0:h2:2;
ydt =@(t,y) 1+t.*sin(t.*y);
ydt2 = @(t,y) sin(t.*y)+t.*(cos(t.*y).*(y+t.*ydt(t,y)));
y2 = zeros(1,length(t2));
y2(1) = 0;
for i = 2:1:length(t2)
    y2(i) = y2(i-1) + h2*ydt(t2(i-1),y2(i-1)) + h2^2/2*ydt2(t2(i-1),y2(i-1));
end %Taylor 二阶求解微分方程
figure
plot(t1,y1,'k--','color',[0 0.4470 0.7410],'LineWidth',1);
hold on
plot(t2,y2,'k--','color',[0.8500 0.3250 0.0980],'LineWidth',1);
hold on
grid on
legend('ode23','Second Order Taylor Method','Location','Southeast');
xlabel('t');
ylabel('y');
title('解 dy/dt=1+tsin(ty)'); %作图
```

第四题



Hw7_4.m

```
f=@(t,y)-y+t.*sqrt(y);
t0=2;tf=3;y0=2;
[t1,w1]=RK4(f,t0,tf,y0,0.5);
[t2,w2]=RK4(f,t0,tf,y0,0.1);
[t3,w3]=RK4(f,t0,tf,y0,0.05);
[t,y]=ode45(f,[t0,tf],y0);
plot(t,y,'-b','LineWidth',5);hold on;
plot(t1,w1,'-r','LineWidth',2);
plot(t2,w2,'-ok','LineWidth',3);
plot(t3,w3,'-*y','LineWidth',1);
legend('ode45','h=0.5','h=0.1','h=0.05','Location','northwest');
xlabel 't';ylabel 'y';
title('t-y 图像');
```

RK4.m

```
function [t,w] = RK4(f,t0,tf,y0,h)
%func 为函数句柄，[a,b]为求解区间，h 为步长，y0 为 y(a)的值（列向量）
t=t0:h:tf;
if t(end)~=tf
    t(end+1)=tf;
end
n=length(t); m=length(y0);
w=zeros(m,n);
w(:,1)=y0;
for i=1:n-1
    if i==n-1
        h=t(end)-t(end-1);
    end
    k1=f(t(i),w(:,i));
    k2=f(t(i)+h/2,w(:,i)+h/2*k1);
    k3=f(t(i)+h/2,w(:,i)+h/2*k2);
    k4=f(t(i)+h,w(:,i)+h*k3);
    w(:,i+1)=w(:,i)+h/6*(k1+2*k2+2*k3+k4);
```

```
end
end
```

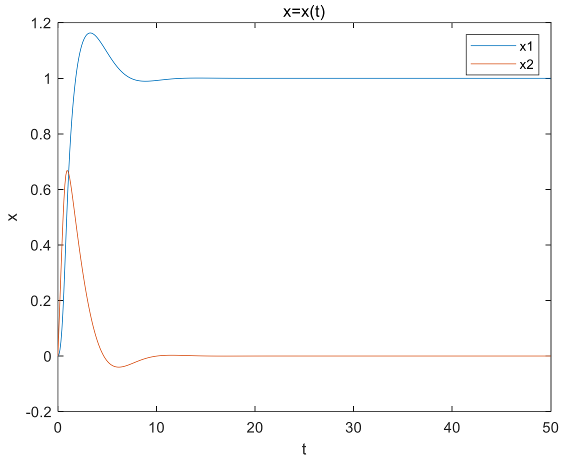
第五题

将方程整理为如下形式：

$$\begin{cases} \dfrac{dx_1}{dt} = \dfrac{(1-x_1)\sin x_1 + x_2\cos x_2}{\sin^2 x_1 + \cos^2 x_2} \\ \dfrac{dx_2}{dt} = \dfrac{(1-x_1)\cos x_2 - x_2\sin(x_1)}{\sin^2 x_1 + \cos^2 x_2} \end{cases}$$

取步长 h=0.1

图像如下：



Hw7_5.m

```
clear
clc
f=@(x1,x2) [((1-x1).*sin(x1)+x2.*cos(x2))./((sin(x1)).^2+(cos(x2)).^2);
                      (((1-x1).*cos(x2))-x2.*sin(x1))./((sin(x1)).^2+(cos(x2)).^2)];
h=0.1;
t=0:h:50;
y=[0;0];
for i=1:length(t)-1
        y(:,i+1)=y(:,i)+h*RK4(f,y(1,i),y(2,i),h);
end
plot(t,y(1,:))
hold on
plot(t,y(2,:))
legend('x1','x2')
xlabel('t')
ylabel('x')
title('x=x(t)')
hold off
```

RK4.m

```
function k=RK4(f,x1,x2,h)
        k1=f(x1,x2);
        k2=f(x1+h/2*k1(1),x2+h/2*k1(2));
        k3=f(x1+h/2*k2(1),x2+h/2*k2(2));
        k4=f(x1+h*k3(1),x2+h*k3(2));
        k=1/6*(k1+2*(k2+k3)+k4);
end
```

第六题

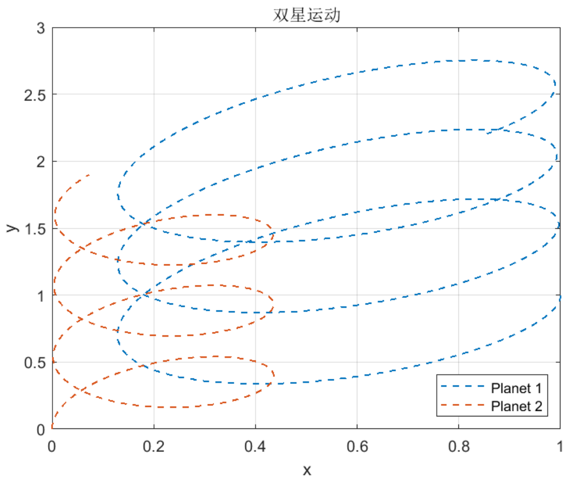
（1）建立物理模型：

$$\frac{dx_1}{dt}=v_{x1},\quad \frac{dy_1}{dt}=v_{y1},\quad \frac{dv_{x1}}{dt}=-G\frac{m_2(x_1-x_2)}{r^3},\quad \frac{dv_{y1}}{dt}=-G\frac{m_2(y_1-y_2)}{r^3}$$

$$\frac{dx_2}{dt}=v_{x2},\quad \frac{dy_2}{dt}=v_{y2},\quad \frac{dv_{x2}}{dt}=-G\frac{m_1(x_2-x_1)}{r^3},\quad \frac{dv_{y2}}{dt}=-G\frac{m_1(y_2-y_1)}{r^3}$$

定义变量：$\boldsymbol{y}=[x_1;y_1;v_{x1};v_{y1};x_2;y_2;v_{x2};v_{y2}]$，$\boldsymbol{y}(\boldsymbol{0})=[1;1;0;-1;0;0;0;1]$，。根据上述微分方程组即可求解。

计算得到结果如下图所示。



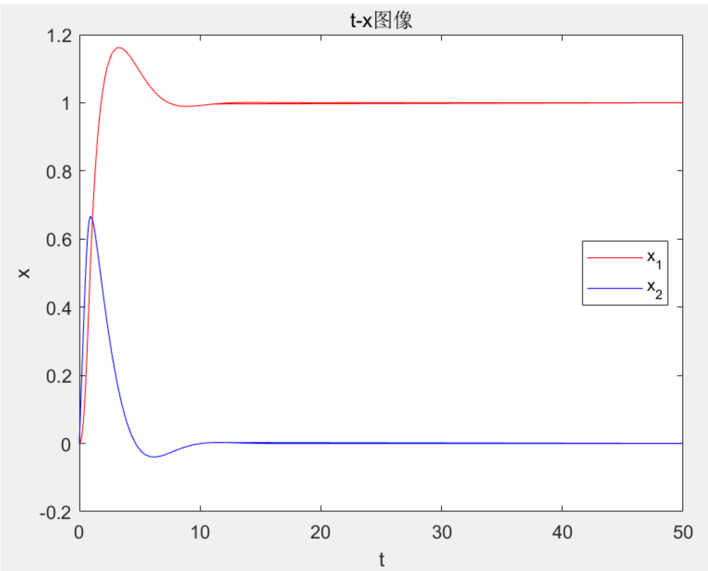
（2）通过 pause 函数实现动画，代码见下。

Hw7_6.m

```
%%%%%% PART ONE
G = 8;
m1 = 0.5;
m2 = 1; %定义参数
ydt = @(t,y) [y(3);...
              y(4);...
              G*m2*(y(5)-y(1))./(((y(1)-y(5)).^2+(y(2)-y(6)).^2).^1.5);...
              G*m2*(y(6)-y(2))./(((y(1)-y(5)).^2+(y(2)-y(6)).^2).^1.5);...
              y(7);...
              y(8);...
              G*m1*(y(1)-y(5))./(((y(1)-y(5)).^2+(y(2)-y(6)).^2).^1.5);...
              G*m1*(y(2)-y(6))./(((y(1)-y(5)).^2+(y(2)-y(6)).^2).^1.5)]; %定义微分方程组
y0 = [1;1;0;-1;0;0;0;1]; %赋予初值
h = 0.01;
tspan = [0 5];
[t1,y1] = hw7_5RK4(ydt,tspan,y0,h); %利用第五题中的四阶 RK 算法函数对微分方程组进行求解
figure
plot(y1(1,:),y1(2,:),'k--','color',[0 0.4470 0.7410],'LineWidth',1);
hold on
plot(y1(5,:),y1(6,:),'k--','color',[0.8500 0.3250 0.0980],'LineWidth',1);
```

```
hold on
grid on
legend('Planet 1','Planet 2','Location','Southeast');
xlabel('x');
ylabel('y');
title('双星运动'); %作图
%%%%%% PART TWO
figure
for i = 1:1:length(t1)
    h1 = plot(y1(1,i),y1(2,i),'o','color',[0 0.4470 0.7410],'LineWidth',1);
    hold on
    h2 = plot(y1(5,i),y1(6,i),'o','color',[0.8500 0.3250 0.0980],'LineWidth',1);
    axis([0,1,0,3]);
    pause(0.1);
    delete(h1);
    delete(h2);
end %通过 pause 函数实现动画
```

第七题



hw7_7.m

```
f=@(t,y)[(sin(y(1))+y(2)*cos(y(2))-y(1)*sin(y(1)))/((sin(y(1)))^2+(cos(y(2)))^2);
    (cos(y(2))-y(2)*sin(y(1))-y(1)*cos(y(2)))/((sin(y(1)))^2+(cos(y(2)))^2)];
y0=[0;0];
h=0.1;
t0=0;tf=50;
w(:,1)=y0;w0=y0;
i=1;t(i)=t0;
TOL=1e-5;
while 1
    if t(i)+h>tf
        h=tf-t(i);
```

```
        end
        w1=oneStepRK4(f,h,w0,t(i));
        wtmp=oneStepRK4(f,h/2,w0,t(i));
        w2=oneStepRK4(f,h/2,wtmp,t(i)+h/2);
        if norm(w2-w1)<=TOL
            while 1
                if t(i)+h>=tf
                    h=tf-t(i);break;
                end
                h=h*2;
                w1=oneStepRK4(f,h,w0,t(i));
                wtmp=oneStepRK4(f,h/2,w0,t(i));
                w2=oneStepRK4(f,h/2,wtmp,t(i)+h/2);
                if norm(w2-w1)>TOL
                    h=h/2;break;
                end
            end
            t(i+1)=t(i)+h;
            w(:,i+1)=oneStepRK4(f,h,w0,t(i));
        else
            while 1
                h=h/2;
                w1=oneStepRK4(f,h,w0,t(i));
                wtmp=oneStepRK4(f,h/2,w0,t(i));
                w2=oneStepRK4(f,h/2,wtmp,t(i)+h/2);
                if norm(w2-w1)<=TOL
                    break;
                end
            end
            t(i+1)=t(i)+h;
            w(:,i+1)=w1;
        end
        w0=w(:,i+1);
        if t(i+1)==tf
            break;
        end
        i=i+1;
    end
    plot(t,w(1,:),'-r',t,w(2,:),'-b');
    legend('x_1','x_2','Location','east');
    xlabel 't';ylabel 'x';
    title('t-x 图像');
    t_adaptivestep=toc
    function w1=oneStepRK4(f,h,w0,t)
```

```
    k1=f(t,w0);
    k2=f(t+h/2,w0+h/2*k1);
    k3=f(t+h/2,w0+h/2*k2);
    k4=f(t+h,w0+h*k3);
    w1=w0+h/6*(k1+2*k2+2*k3+k4);
end
```

从结果的图像比较，发现两种算法的结果一致。

将两题的程序中画图部分的代码注释，只留求解数值解部分的代码。运行变步长算法后发现其最小的步长为 0.2，正好与第五题中条件一致，具有可比性。下面对其效率进行比较，通过以下程序进行比较：

hw7_7_2.m

```
clear;
t1=zeros(1000,1);
for k=1:1000
    tic;
    hw7_5
    t1(k)=toc;
    clearvars -except t1 k;
end
t1_ave=sum(t1)/1000;
fprintf('fixed step(h=0.2),t=%f\n',t1_ave);
clear;
t2=zeros(1000,1);
for j=1:1000
    tic;
    hw7_7
    t2(j)=toc;
    clearvars -except t2 j;
end
t2_ave=sum(t2)/1000;
fprintf('adaptive step(min h=0.2),t=%f\n',t2_ave);
```

运行结果为

```
>> hw7_7_2
fixed step(h=0.2),t=0.000686
adaptive step(min h=0.2),t=0.001372
>> hw7_7_2
fixed step(h=0.2),t=0.000654
adaptive step(min h=0.2),t=0.001357
```

可以发现变步长法比固定步长法要慢，猜测可能是因为要求的时间差距不够大，下面将两个程序的终止时间调到 1000，进行比较

```
>> hw7_7_2
fixed step(h=0.2),t=0.009533
adaptive step(min h=0.2),t=0.007918
>> hw7_7_2
fixed step(h=0.2),t=0.009636
adaptive step(min h=0.2),t=0.008788
```

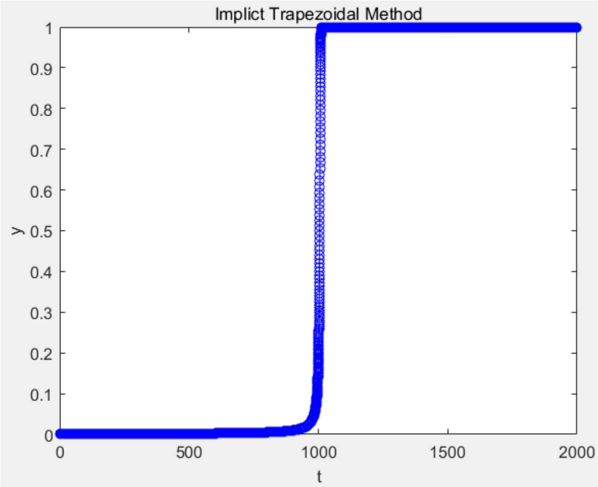
则变步长法的优势在所求时间的时间长度较长时可以体现，同时其实在存在突变过程的问题中也有一定的优势。

第八题

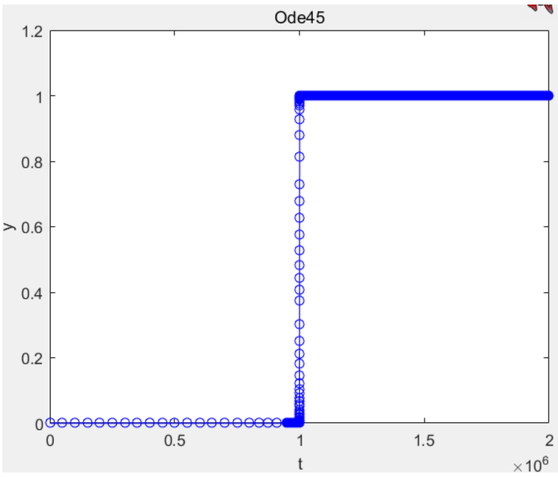
参考 1：

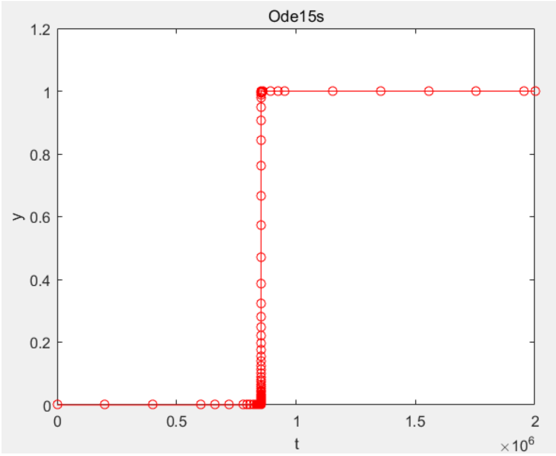
第八题

(1)该题目区间十分长，步长过小，计算时间长。因此修改$\delta = 10^{-3}$



(2)该题目为 stiff problem，在$t = \frac{1}{\delta}$时，发生突变，因此显式的方法需要采用极小的步长才能保证精度，因此 ode45 需要花费更长的时间





可以看到 ode15s 计算的步长分布比 ode45 更为合理，隐式方法计算时在突变点处误差更小

hw7_8.m

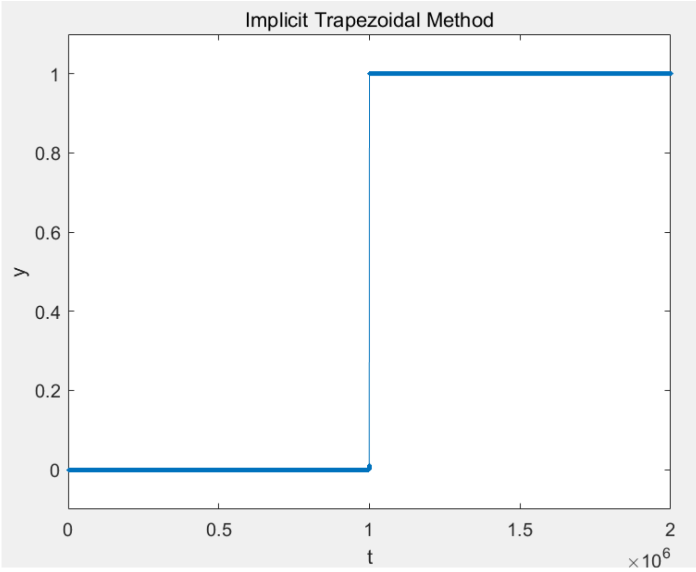
```
clear all; close all; clc
dy = @(t,y)(y.^2-y.^3);
dyy = @(t,y)(2*y-3*y.^2);
delta = 1e-3; h = 0.1e3;
t1 = 0:h:2/delta;
y1 = Trape_ode_implict_Newton(dy, t1, delta, 1e-5, dyy);
figure()
plot(t1, y1, '-ob')
xlabel('t'); ylabel('y'); title('Implict Trapezoidal Method')
delta = 1e-6;
tspan = [0, 2/delta];
opts = odeset('AbsTol', 1e-5);
[t2, y2] = ode45(dy, tspan, delta, opts);
figure()
plot(t2, y2, '-ob')
xlabel('t'); ylabel('y'); title('Ode45')
fprintf('steps taken by ode45: %d\n', length(t2))
[t3, y3] = ode15s(dy, tspan, delta, opts);
figure()
plot(t3, y3, '-or')
xlabel('t'); ylabel('y'); title('Ode15s')
fprintf('\nsteps taken by ode15s: %d\n\n', length(t3))
```

Trape_ode_implict_Newton.m

```
function y = Trape_ode_implict_Newton(dy, t, y0, TOL, dyy)
    y = zeros(size(t)); y(1) = y0;
    for k = 2:length(t)
        h = (t(k)-t(k-1));
        y(k) = imp_step(dy, t, y(k-1), h, TOL, dyy);
    end
end
function y=imp_step(dy, t, y0, h, TOL, dyy)
    fun = @(w)(y0+h/2*(dy(t,y0) + dy(t+h,w))-w);
    dfun = @(w)(h/2*(dyy(t+h,w))-1);
    y = fpi(@(w)(w-fun(w)/dfun(w)), y0, TOL);
end
```

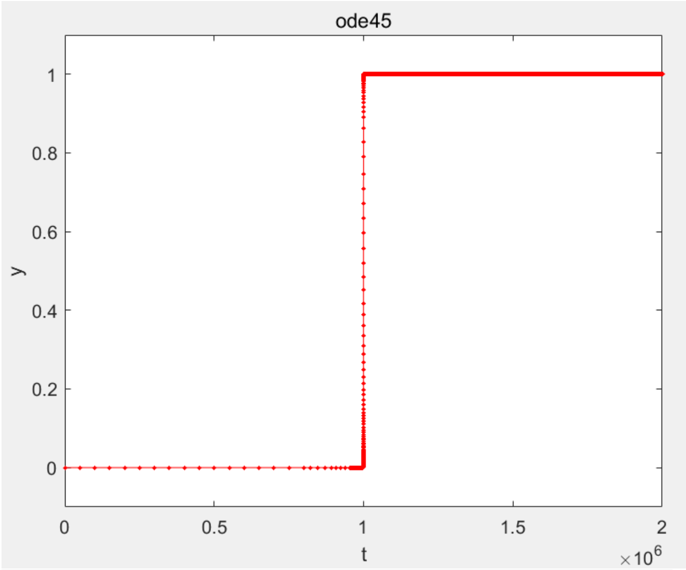
示例 2：

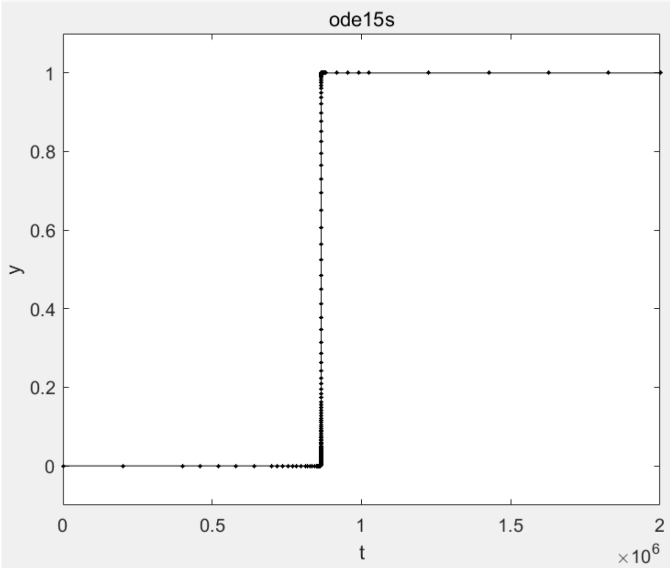
（1）




```
Hw7_8_1.m
clc,clear;
delta=1e-6;
t0=0; tf=2/delta;
h=0.1;TOL=1e-5;
t=t0:h:tf;
n=length(t);
w=zeros(1,n);
w(1)=delta;
g=@(x,y)x-h/2*x^2+h/2*x^3-y-h/2*y^2+h/2*y^3;
dg=@(x)1-h*x+3*h/2*x^2;
for i=1:n-1
    w0=w(i);
    w1=w0-g(w0,w0)/dg(w0);
    while abs(w1-w0)>TOL
        w0=w1;
        w1=w0-g(w0,w0)/dg(w0);
    end
    w(i+1)=w1;
end
plot(t,w,'-*','markerSize',2);
axis([t0 tf -0.1 1.1])
xlabel t;
ylabel y;
title 'Implicit Trapezoidal Method';
```

（2）



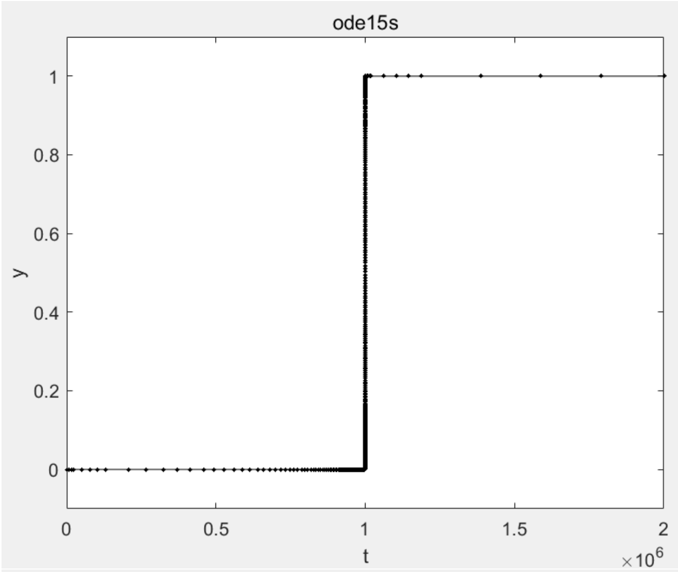


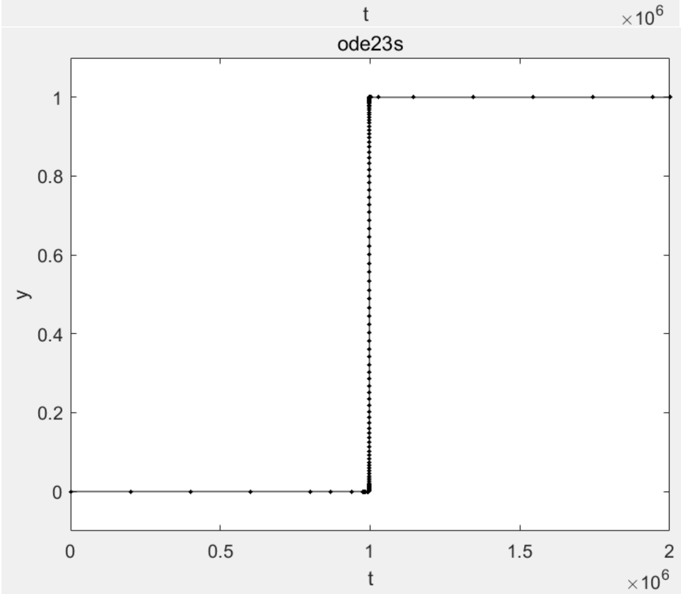

```
clc,clear;
f=@(t,y)y.^2-y.^3;
delta=1e-6;
t0=0; tf=2/delta;y0=delta;TOL=1e-5;
opts = odeset('RelTol',TOL,'AbsTol',TOL); % 这里，设置绝对误差、相对误差、都设，都对
[t2,w2]=ode45(f,[t0,tf],y0,opts);
[t3,w3]=ode15s(f,[t0,tf],y0,opts);
plot(t2,w2,'-r*','markerSize',2);
axis([t0 tf -0.1 1.1])
xlabel t; ylabel y; title 'ode45';
figure;
plot(t3,w3,'-k*','markerSize',2);
axis([t0 tf -0.1 1.1])
xlabel t; ylabel y; title 'ode15s';
```

结果比较：

（1）Implicit Trapezoidal Method 求解时间最长，ode45 次之，ode15s 的效率最高，且其点数最少。这是因为原方程是刚性方程，这类方程的特点就是其解时而变化缓慢时而变化十分迅速，用常规的微分方程数值解法需要用极小的步长，也就导致了求解时间的延长。

（2）但是，我们发现 ode15s 求解的结果并不准确，通过减小相对误差或绝对误差才可以让其结果准确度更高，或者在相同的误差条件下，我们使用 ode23s 也可以得到更准确的结果。


ode15s
y
t
×10^6


ode23s
y
t
×10^6