6.11 Implement Heun's method to estimate the solution of the initial value problem in Exercise 6.8b. Use h = 1/40 and h = 1/80. Compute the errors at x = 0.5 and x = 1.0 to see if they are roughly quartered as h is halved. How small an h would you estimate is needed in order for the absolute error to be less than 10^{-6} in magnitude?

El método de Heun se define para $i = 0, \cdots, n-1$ como

$$y_0 = a$$
$$y_{i+1} = y_i + \frac{h}{2}\Big(f(x_i, y_i) + f\big(x_i + h, y_i + h\, f(x_i, y_i)\big)\Big)$$

El problema por resolver es $\frac{dy}{dx} = -\frac{y^3}{2}, y(0) = 1, y(x) = \frac{1}{\sqrt{1+x}}$, por lo tanto $\boldsymbol{\alpha} = \mathbf{1}$ (valor inicial) y $f(x, y) = -\frac{y^3}{2}$ (función que define la EDO). Se implementó el método en un script en Matlab y se resolvió la EDO dada para los tamaños de paso dados.

Command Window

Método de Heun con tamaño de paso 0.025000

x	y	error
0.5	0.816510918598554	1.4338e-05
1.0	0.70712073488976	1.3954e-05

Command Window

Método de Heun con tamaño de paso 0.012500

x	y	error
0.5	0.816500145287085	3.5644e-06
1.0	0.707110251903514	3.4707e-06

Podemos ver que los errores disminuyen en aproximadamente un cuarto de magnitud

$$\frac{1.43376708274801 \times 10^{-5}}{3.56435935933153 \times 10^{-6}} = 4.02250990488485$$

$$\frac{1.39537032122217 \times 10^{-5}}{3.47071696582546 \times 10^{-6}} = 4.02040942825858$$

Si los errores disminuyen en un cuarto de su magnitud cada vez que se disminuye a la mitad el tamaño de paso, bastaría con hacer $h = \frac{1}{160}$ para obtener errores de menos de 1×10^{-6}.

```
%Definimos la función f(x,y)
f=@(x,y) -y^3/2;
%Definimos la función solución
y_sol=@(x,y) 1/sqrt(1+x);
%Definimos el valor inicial (se asume que empiezas en x=0)
Y_i=1; X_i=0;
%Tamaño de paso
h=1/160;
%Numero de iteraciones (opcionalmente introduce límite sup)
limite=1; n=limite/h;
%METODO DE HEUN
sol=zeros(n+1,3);
sol(1,1)=X_i; sol(1,2)=Y_i;
for i=2:n+1
    %Calculo de solución
    Y_i=Y_i+(h/2)*(f(X_i,Y_i)+f(X_i+h,Y_i+h*f(X_i,Y_i)));
    X_i=X_i+h;
    sol(i,1)=X_i; sol(i,2)=Y_i;
    %Calculo de error local
    sol(i,3)=abs(Y_i-y_sol(X_i,Y_i));
end
%Mostramos soluciones de x=0.5 y x=1
fprintf('Método de Heun con tamaño de paso %f\n',h);
disp(sol(n/2+1,:));
disp(sol(n+1,:))
```

6.12 Implement the classical Runge-Kutta algorithm to estimate the solution of the initial value problem in Exercise 6.8b. Use h = 1/40 and h = 1/80. Compute the errors at x = 0.5 and x = 1.0. By what factor are the errors reduced as h is halved? Roughly speaking, what is the largest value of h for which the absolute error will be less than 10^{-6} in magnitude?

El método de Runge-Kutta clásico se define para $i = 0, \cdots, n-1$ como

$$y_0 = a$$
$$K_0 = f(x_i, y_i)$$
$$K_1 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_0\right)$$
$$K_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}K_1\right)$$
$$K_3 = f(x_i + h, y_i + hK_2)$$
$$y_{i+1} = y_i + \frac{h}{6}(K_0 + 2K_1 + 2K_2 + K_3)$$

El problema por resolver es $\frac{dy}{dx} = -\frac{y^3}{2}, y(0) = 1, y(x) = \frac{1}{\sqrt{1+x}}$, por lo tanto $\boldsymbol{\alpha} = \mathbf{1}$ (valor inicial) y $f(x,y) = -\frac{y^3}{2}$ (función que define la EDO). Se implementó el método en un script en Matlab y se resolvió la EDO dada para los tamaños de paso dados.

Command Window		
Método de Runge-Kutta con tamaño de paso 0.025000		
x	y	error
0.5	0.816496581014829	8.7103102508479e-11
1.0	0.707106781257412	7.0864647483404e-11

Command Window		
Método de Runge-Kutta con tamaño de paso 0.012500		
x	y	error
0.5	0.816496580933499	5.77282666114343e-12
1.0	0.707106781191226	4.6779247142581e-12

Podemos ver que los errores disminuyen en un factor de

$$\frac{8.7103102508479 \times 10^{-11}}{5.77282666114343 \times 10^{-12}} = 15.0884666423063$$

$$\frac{7.0864647483404 \times 10^{-11}}{4.6779247142581 \times 10^{-12}} = 15.1487362050552$$

Si los errores disminuyen en $\frac{1}{15}$ de su magnitud cada vez que se disminuye a la mitad el tamaño de paso, bastaría con hacer $h = \frac{1}{5} = 0.2$ para obtener errores de menos de 1×10^{-6}.

```
%Definimos la función f(x,y)
f=@(x,y) -y^3/2;
%Definimos la función solución
y_sol=@(x,y) 1/sqrt(1+x);
%Definimos el valor inicial (se asume que empiezas en x=0)
Y_i=1; X_i=0;
%Tamaño de paso
h=0.275;
%Numero de iteraciones (opcionalmente introduce límite sup)
limite=1; n=ceil(limite/h);
%METODO DE HEUN
sol=zeros(n+1,3);
sol(1,1)=X_i; sol(1,2)=Y_i;
for i=2:n+1
    K_0=f(X_i,Y_i);
    K_1=f(X_i+h/2,Y_i+(h/2)*K_0);
    K_2=f(X_i+h/2,Y_i+(h/2)*K_1);
    K_3=f(X_i+h/2,Y_i+h*K_2);
    %Calculo de solución
    Y_i=Y_i+(h/6)*(K_0+2*K_1+2*K_2+K_3);
    X_i=X_i+h;
    sol(i,1)=X_i; sol(i,2)=Y_i;
    %Calculo de error local
    sol(i,3)=abs(Y_i-y_sol(X_i,Y_i));
end
%Mostramos soluciones de x=0.5 y x=1
fprintf('Método de Runge-Kutta con tamaño de paso %f\n',h);
disp(sol(floor(n/2+1),:));
disp(sol(floor(n+1),:))
```

6.15 Implement Euler's method and a local error estimator based on Heun's method. Apply it to the problem

$$\frac{dy}{dx} = 10(y - x), y(0) = \frac{1}{10}$$

and compare the estimated local error to the true local error. Also compare the global error at several points to the general size of the local errors made in the computations up to this point.

Se programó el método correspondiente el Matlab y se computaron los errores sabiendo que la solución exacta de esta ecuación diferencial es $y(x) = x + \frac{1}{10}$. Un hecho muy interesante es que el método de Euler parece funcionar exactamente para intervalos de $[0, a]$ con $a \leq 1$, sin embargo, después los errores cometidos por el método incrementan exponencialmente, incluso el estimador (método de Heun) comete errores increíblemente grandes.

```
%Definimos la función f(x,y)
f=@(x,y) 10*(y-x);
%Definimos la función solución
y_sol=@(x,y) x+(1/10);
%Tamaño de paso
h=1/80;
%Numero de iteraciones (opcionalmente introduce límite sup)
limite=10; n=ceil(limite/h);
%METODO DE HEUN
Y_i=1/10; X_i=0;
sol_H=zeros(n+1,2);
sol_H(1,1)=X_i; sol_H(1,2)=Y_i;
for i=2:n+1
    %Calculo de solución
    Y_i=Y_i+(h/2)*(f(X_i,Y_i)+f(X_i+h,Y_i+h*f(X_i,Y_i)));
    X_i=X_i+h;
    sol_H(i,1)=X_i; sol_H(i,2)=Y_i;
end
%METODO DE EULER
Y_i=1/10; X_i=0;
sol_E=zeros(n+1,2);
sol_E(1,1)=X_i; sol_E(1,2)=Y_i;
for i=2:n+1
    Y_i=Y_i+h*f(X_i,Y_i);
    X_i=X_i+h;
    sol_E(i,1)=X_i; sol_E(i,2)=Y_i;
end
%Calculamos el error con de Euler con respecto a Heun y a la sol
error_EH=abs(sol_E(:,2)-sol_H(:,2));
error_real=abs(sol_E(:,2)-y_sol(sol_E(:,1),sol_E(:,2)));
disp(error_real);
```