Num. #1: ordinary differential equations (ODEs) - Correction

The programs are written with the MATLAB software.
For the exercise, the following functions are needed

- **Euler method** :

```
% T is the final time, dt the time step
% uinit is the initial value, f is the function of the ODE
% Euler method
function u=Euler(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(1)=uinit;
        % Euler method
    for i=1:Nt-1,
        u(i+1)=u(i)+dt*f(u(i));
    end
    % Plot a graphic
    plot(time,u);
```

- **Midpoint (or Runge-Kutta 2) method** :

```
% RK2 method
function[u]=RK2(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(1)=uinit;
        % RK2 method
    for i=1:Nt-1,
        v=u(i)+dt*f(u(i))/2;
        u(i+1)=u(i)+dt*f(v);
    end
    % Plot a graphic
    plot(time,u,'r');
```

- **Heun method** :

```
% Heun method
function[u]=Heun(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(1)=uinit;
        % Heun method
    for i=1:Nt-1,
        p1=f(u(i));
        p2=f(u(i)+dt*p1);
        u(i+1)=u(i)+dt*(p1+p2)/2;
    end
    % Plot a graphic
    plot(time,u,'g');
```

- **Runge-Kutta 4 method**

```
% RK4 method
function[u]=RK4(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(1)=uinit;
        % Heun method
    for i=1:Nt-1,
        p1=f(u(i));
        p2=f(u(i)+dt*p1/2);
        p3=f(u(i)+dt*p2/2);
        p4=f(u(i)+dt*p3);
        u(i+1)=u(i)+dt*(p1+2*p2+2*p3+p4)/6;
    end
    % Plot a graphic
    plot(time,u,'k');
```

- **Forward Euler method for a system** :

```
% T is the final time, dt the time step
% uinit is the initial value, f is the function of the ODE
```

```
% The output u is a matrix
% Column n corresponds to a vector at a discrete time t^n
% Euler method
function[u]=EulerSystem(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(:,1)=uinit;
        % Euler method
    for i=1:Nt-1,
        u(:,i+1)=u(:,i)+dt*f(u(:,i));
    end
```

- **Midpoint (or Runge-Kutta 2) method for a system** :

```
% RK2 method
function[u]=RK2System(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(:,1)=uinit;
        % RK2 method
    for i=1:Nt-1,
        v=u(:,i)+dt*f(u(:,i))/2;
        u(:,i+1)=u(:,i)+dt*f(v);
    end
```

- **Heun method for a system** :

```
% Heun method
function[u]=HeunSystem(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(:,1)=uinit;
        % Heun method
    for i=1:Nt-1,
```

```
        p1=f(u(:,i));
        p2=f(u(:,i)+dt*p1);
        u(:,i+1)=u(:,i)+dt*(p1+p2)/2;
    end
```

- **Runge-Kutta 4 method for a system** :

```
% RK4 method
function[u]=RK4System(T,dt,uinit,f)
    % Time discretization vector
    time=0:dt:T;
    Nt=length(time);
    % Initial datum
        u(:,1)=uinit;
        % Heun method
    for i=1:Nt-1,
        p1=f(u(:,i));
        p2=f(u(:,i)+dt*p1/2);
        p3=f(u(:,i)+dt*p2/2);
        p4=f(u(:,i)+dt*p3);
        u(:,i+1)=u(:,i)+dt*(p1+2*p2+2*p3+p4)/6;
    end
```

Exercise

1. Implement (with MATLAB) the resolution of the following equation :

$$\partial_t u = 1 - u^2 \text{ with } u(0) = 0 \tag{1}$$

using the four methods presented above and a time step $\Delta t = 0.1$ until time $T = 1$.

```
% First example
T=1;
dt=0.1;
f=inline('1-x^2');
uinit=0;
Euler(T,dt,uinit,f)
RK2(T,dt,uinit,f)
Heun(T,dt,uinit,f)
RK4(T,dt,uinit,f)
```

2. Remark that the exact solution of equation (1) can be computed exactly and is equal to :

$$u(t) = tanh(t) = \frac{e^{2t}-1}{e^{2t}+1}.$$

Enhance the convergence property of the Heun method by letting Δt go to 0.

```
clear;
uinit=0;
f=inline('1-x^2');
% Different time steps
T=1;
TimeStep=[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001];
for k=1:length(TimeStep),
   dt=TimeStep(k);
   time=0:dt:T;
   % Exact solution
   uexact=(exp(2*time)-ones(size(time)))./(exp(2*time)+ones(size(time)));
   % Approximated solution
   u=Heun(T,dt,uinit,f);
   % error
   Error(k)=sqrt(dt)*norm(uexact-u);
end
% Display the error
format long;
Error
```

3. We still consider equation (1). Compare the order of the four methods by plotting a graph in a log-log scale, which represents the evolution of L^2 error with respect to the time step Δt.

```
 clear;
uinit=0;
f=inline('1-x^2');
% Different time steps
T=1;
TimeStep=[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001];
for k=1:length(TimeStep),
   dt=TimeStep(k);
   time=0:dt:T;
   % Exact solution
   uexact=(exp(2*time)-ones(size(time)))./(exp(2*time)+ones(size(time)));
   % Approximated solutions
```

```
    uEuler=Euler(T,dt,uinit,f);
    uHeun=Heun(T,dt,uinit,f);
    uRK2=RK2(T,dt,uinit,f);
    uRK4=RK4(T,dt,uinit,f);
    % errors between exact and approximated solutions
    ErrorEuler(k)=sqrt(dt)*norm(uexact-uEuler);
    ErrorHeun(k)=sqrt(dt)*norm(uexact-uHeun);
    ErrorRK2(k)=sqrt(dt)*norm(uexact-uRK2);
    ErrorRK4(k)=sqrt(dt)*norm(uexact-uRK4);
end
% Clear the figure
clf;
% Graph of the errors
loglog(TimeStep,ErrorEuler)
hold on;
loglog(TimeStep,ErrorHeun,'r')
loglog(TimeStep,ErrorRK2,'g')
loglog(TimeStep,ErrorRK4,'k')
% Legend for the graph
legend('Euler','Heun','RK2','RK4');
```

4. Same question using now the following equation

$$\partial_t u = \mathrm{e}^{-u} - 1 + u \text{ with } u(0) = 1, \tag{2}$$

for which no exact solution is known. The error will be defined as the L^2 norm of the difference between the solution computed with a time step Δt and the solution computed with a time step $\frac{\Delta t}{2}$.

```
clear;
uinit=1;
f=inline('exp(-x)-1+x');
% Different time steps
T=1;
TimeStep=0.1*0.5.^(0:6);
NumberStep=length(TimeStep);
% Computation for the 1st time step
dt=TimeStep(1);
uEuler=Euler(T,dt,uinit,f);
uHeun=Heun(T,dt,uinit,f);
uRK2=RK2(T,dt,uinit,f);
uRK4=RK4(T,dt,uinit,f);
```

```
for k=2:NumberStep,
   dt=TimeStep(k);
   % Save previous approximated solutions for dt
   uEulerOld=uEuler;
   uHeunOld=uHeun;
   uRK2Old=uRK2;
   uRK4Old=uRK4;
  % Approximated solutions for dt/2
  uEuler=Euler(T,dt,uinit,f);
  uHeun=Heun(T,dt,uinit,f);
  uRK2=RK2(T,dt,uinit,f);
  uRK4=RK4(T,dt,uinit,f);
  % errors between approximated solutions
  ErrorEuler(k-1)=sqrt(dt)*norm(uEulerOld-uEuler(1:2:end));
  ErrorHeun(k-1)=sqrt(dt)*norm(uHeunOld-uHeun(1:2:end));
  ErrorRK2(k-1)=sqrt(dt)*norm(uRK2Old-uRK2(1:2:end));
  ErrorRK4(k-1)=sqrt(dt)*norm(uRK4Old-uRK4(1:2:end));
end
% Clear the figure
clf;
% Graph of the errors
loglog(TimeStep(1:NumberStep-1),ErrorEuler')
hold on;
loglog(TimeStep(1:NumberStep-1),ErrorHeun','r')
loglog(TimeStep(1:NumberStep-1),ErrorRK2','g')
loglog(TimeStep(1:NumberStep-1),ErrorRK4','b')
% Legend for the graph
legend('Euler','Heun','RK2','RK4');
```

5. Implement the resolution of the following system of equations, called the Lorenz system :

$$\begin{cases} \partial_t x = 10(y-x) \\ \partial_t y = x(28-z) - y \\ \partial_t z = xy - \dfrac{8}{3}z \end{cases} \tag{3}$$

using the four methods presented above.

```
   clear;
T=30;
dt=0.001;
% Initial datum for a system
```

```
xinit=1;yinit=1;zinit=1;
uinit=[xinit;yinit;zinit];
% Function for a system
f=inline('[10*(u(2)-u(1));u(1)*(28-u(3))-u(2);u(1)*u(2)-8*u(3)/3]','u');
u=EulerSystem(T,dt,uinit,f);
%RK2System(T,dt,uinit,f);
%HeunSystem(T,dt,uinit,f);
%RK4System(T,dt,uinit,f);
% Plot a graphic
clf;
plot3(u(1,:),u(2,:),u(3,:))
```