Numerical Methods Homework-8

B10602110 四電子三乙 呂和軒

1.

Evaluate the following integral:

$$\int_0^{\pi/2} (8 + 4\cos x)dx$$

(a) Analytically; (b) Single application of the trapezoidal rule; (c) Composite trapezoidal rule with n=2 and n=4; (d) Single application of Simpson's 1/3 rule; (e) Composite Simpson's 1/3 rule with n=4; (f) Simpson's 3/8 rule, and (g) Composite Simpson's rule with n=5. For each of the numerical estimates (b) through (g), determine the true percent relative error based on (a).

ANS :

1.Code_function :

```
function [area] = trap_I(f,x0,x1,n)
    xx = linspace(x0,x1,n+1);
    area = 0;
    for i = 1:n
        area = area + (xx(i+1)-xx(i))*0.5*(f(xx(i+1))+f(xx(i)));
    end
end

function [area] = Simpson_1_3_I(f,x0,x1,n)
    xx = linspace(x0,x1,3*n-n+1);
    area = 0;
    for i = 1:2:3*(n-1)-(n-1-1)
        area = area + (xx(i+1)-xx(i))/3 ...
               *(f(xx(i))+4*f(xx(i+1))+f(xx(i+2)));
    end
end

function [area] = Simpson_3_8_I(f,x0,x1,n)
    xx = linspace(x0,x1,4*n-n+1);
    area = 0;
    for i = 1:3:4*(n-1)-(n-1-1)
        area = area + (xx(i+1)-xx(i))*3/8 ...
               *(f(xx(i))+3*f(xx(i+1))+3*f(xx(i+2))+f(xx(i+3)));
    end
end
```

2.Code_main:

```
close all
clear all
format long

f = @(x) (8+4.*cos(x))

x0 = 0; x1 = 0.5*pi;

%% (a)
syms x
f_s = 8+4*cos(x);
```

```
f_s_I = int(f_s)
f_I = matlabFunction(f_s_I)
area_a = f_I(x1) - f_I(x0)
fprintf("(a)real: %6.15f  \n",area_a)
%% (b)
area_b = trap_I(f,x0,x1,1);
error = abs(area_a - area_b)/area_a;
fprintf("(b) n=1: %6.15f  , error = %6.15f\n",area_b,error)
%% (c)
n = 2;
area_c_2 = trap_I(f,x0,x1,n);
error = abs(area_a - area_c_2)/area_a;
fprintf("(c) n=2: %6.15f  , error = %6.15f \n",area_c_2,error)

n = 4;
area_c_2 = trap_I(f,x0,x1,n);
error = abs(area_a - area_c_2)/area_a;
fprintf("    n=4: %6.15f  , error = %6.15f\n",area_c_2,error)
%% (d)
n = 1;
area_d = Simpson_1_3_I(f,x0,x1,n);
error = abs(area_a - area_d)/area_a;
fprintf("(d) n=1: %6.15f  , error = %6.15f\n",area_d,error)
%% (e)
n = 4;
area_e = Simpson_1_3_I(f,x0,x1,n);
error = abs(area_a - area_e)/area_a;
fprintf("(e) n=4: %6.15f  , error = %6.15f\n",area_e,error)
%% (f)
n = 1;
area_f = Simpson_3_8_I(f,x0,x1,n);
error = abs(area_a - area_f)/area_a;
fprintf("(f) n=1: %6.15f  , error = %6.15f\n",area_f,error)
%% (g)
n = 5;
area_g = Simpson_3_8_I(f,x0,x1,n);
error = abs(area_a - area_g)/area_a;
fprintf("(g) n=5: %6.15f  , error = %6.15f\n",area_g,error)
```

3.Result:

```
(a)real: 16.566370614359172
(b) n=1: 15.707963267948966  , error = 0.051816258756530
(c) n=2: 16.358608410233252  , error = 0.012541202232059
    n=4: 16.514833818250274  , error = 0.003110928597977
(d) n=1: 16.575490124328013  , error = 0.000550483276098
(e) n=4: 16.566403796455042  , error = 0.000002002979207
(f) n=1: 16.570390307616290  , error = 0.000242641756042
(g) n=5: 16.566376643005032  , error = 0.000000363908668
```

2.

Use Romberg integration to evaluate:

$$\int_0^2 \frac{e^x \sin x}{1+x^2} dx$$

to an accuracy of $\varepsilon_s = 0.5\%$. Your results should be presented in the form of $O(h^2) \to O(h^4) \to O(h^6) \to O(h^8)$.

ANS :

1.Code_function :

```
function [d,iter] = Romberg_I(f,x0,x1,maxit,es)
    %d = zeros(n,n);
    n_h = 1;
    i = 1;
    d(1,1) =  trap_I(f,x0,x1,1);
    iter = 1;
    while iter < maxit
        n_h = 2^iter;
        d(iter+1,1) =  trap_I(f,x0,x1,n_h);
        for i = 2:iter+1
            d(iter+1,i) =
(4^(i-1)*d(iter+1,i-1)-d(iter,i-1))/(4^(i-1)-1);
        end
        ea = abs(d(iter+1,iter+1)-d(iter,iter))/d(iter,iter);
        iter = iter + 1;
        if(ea <= es)
            break;
        end
    end
end
```

2.Code_main:

```
close all
clear all
format long

f = @(x) (exp(x).*sin(x))./(1+x.^2);
x0 = 0;x1 = 2;
n = 12;
es = 0.005;
[d2,iter] = Romberg_I(f,x0,x1,n,es)
str = [];
str_r = [];
for i = 1:iter
    str = [str;'O_'+string(2*i)+'_'];
    str_r = [str_r,'n ='+string(2^(i-1))];
end
T = array2table(round(d2,12));
for i = 1:iter
  T.Properties.VariableNames{i} = char((str(i)));
   T.Properties.RowNames{i} = char(str_r(i));
end
T
```

3.Result:

	O_2_	O_4_	O_6_	O_8_
n =1	1.343769939486	0	0	0
n =2	1.815562613332	1.972826837948	0	0
n =4	1.911720382902	1.943772972759	1.941836048413	0
n =8	1.933099246404	1.940225534238	1.939989038337	1.939959720717

3.

Develop a script to generate the following function in which both independent variables ranging from -3 to 3:

(a) $f(x,y) = e^{-(x^2+y^2)}$.

(b) $f(x,y) = xe^{-(x^2+y^2)}$.

ANS :

1.Code_main :

```
close all
clear all
format long

f_a = @(x,y) exp(-(x.^2+y.^2))
f_b = @(x,y) x.*exp(-(x.^2+y.^2))

[xx,yy] = meshgrid(-3:0.01:3,-3:0.01:3);
figure(1)
mesh(xx,yy,f_a(xx,yy))
figure(2)
mesh(xx,yy,f_b(xx,yy))
```

4.

Develop an *M*-file to solve a single ODE by Heun's method with iteration. Design the *M*-file so that it creates a plot of the results.

ANS:

1.Code_ function:

```
function [y,tt] = Heun_D(fy,t0,t1,y0,h)
    tt = t0:h:t1;
    n = length(tt);
    y = y0*ones(n,1);
    for i = 1:n-1
        k1 = fy(tt(i),y(i));
        k2 = fy(tt(i+1),y(i)+h*k1);
        y(i+1) = y(i) + h*0.5*(k1+k2);
    end
end
```

5.

Develop an *M*-file to solve a single ODE by the midpoint method. Design the *M*-file so that it creates a plot of the results.

ANS :

1.Code_function :

```
function [y,tt] = midpoint_D(fy,t0,t1,y0,h)
    tt = t0:h:t1;
    n = length(tt);
    y = y0*ones(n,1);
    for i = 1:n-1
        k1 = fy(tt(i),y(i));
        k2 = fy(tt(i)+0.5*h,y(i)+0.5*h*k1);
        y(i+1) = y(i) + h*(k2);
    end
end
```

6.

Given:

$$\frac{dy}{dt} = -100000y + 99999e^{-t}$$

(a) Estimate the step size required to maintain stability using the explicit Euler method.

ANS :

1.Code_function :

```
function [y,tt] = Eulr_D(fy,t0,t1,y0,h)
   tt = t0:h:t1;
   n = length(tt);
   y = y0*ones(n,1);
   for i = 1:n-1
       k1 = fy(tt(i),y(i));
       y(i+1) = y(i) + h*(k1);
   end
end
```

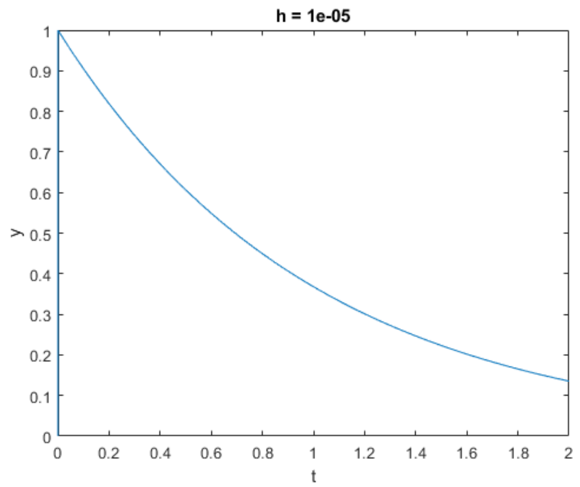
2.Code_main:

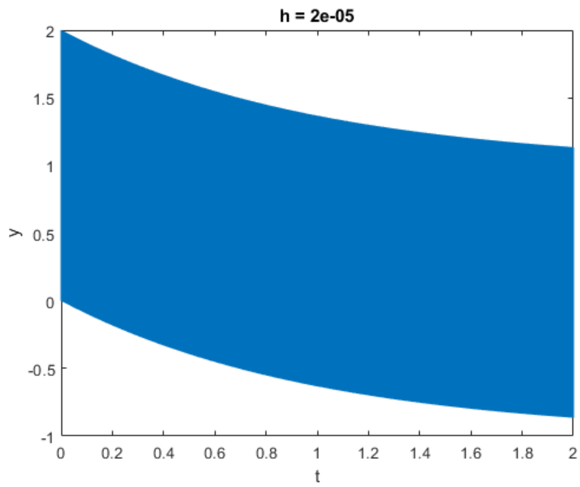
```
close all
clear all
format long

fy = @(t,y) -1e5.*y+99999.*exp(-t)

tspan = [0,2]
y0 = 0;

h = 3e-5;
[y,tt] = Eulr_D(fy,tspan(1),tspan(2),y0,h);

figure(1)
plot(tt,y);
xlabel("t")
ylabel("y")
title("h = "+string(h))

h = 2e-5;
[y,tt] = Eulr_D(fy,tspan(1),tspan(2),y0,h);
```

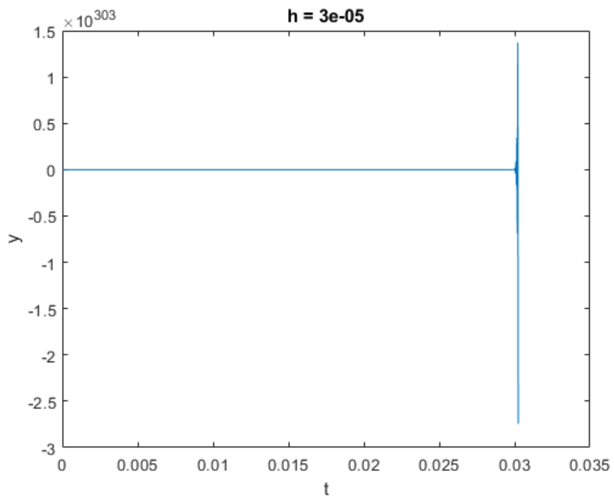
```
figure(2)
plot(tt,y);
xlabel("t")
ylabel("y")
title("h = "+string(h))

h = 1e-5;
[y,tt] = Eulr_D(fy,tspan(1),tspan(2),y0,h);
figure(3)
plot(tt,y);
xlabel("t")
ylabel("y")
title("h = "+string(h))
```

3.Result:

令$\frac{dy}{dt} \cong -100000y$, $y_n = y_{n-1} + h * (-100000y_{n-1}) = (1-100000h)y_{n-1}$
$y_n = (1-100000h)^n y_0$，為了保持穩定度，$|1-100000h| < 1$ ，因此得
$h < \frac{2}{100000}$ ，h = 1e-5 時保持穩定收斂，h=2e-5 時振盪，h=3e-5 發散。







(b) If $y(0)=0$, use the implicit Euler to obtain a solution from $t=0$ to $t=2$ using step size of 0.1.

ANS :

1. Code_function :

```
function [y,tt] = Eulr_D_back(fy,t0,t1,y0,h)
   tt = t0:h:t1;
   n = length(tt);
   y = y0*ones(n,1);
   iter = 50;
   for i = 1:n-1
       f_z = @(y_1) y_1-h.*fy(tt(i+1),y_1)-y(i);
       y(i+1) = newton_back(f_z,y(i),iter);
   end
end
```

2.Code_main:

```
close all
clear all
format long

fy = @(t,y) -1e5.*y+99999.*exp(-t)



tspan = [0,2]
y0 = 0;
h = 0.1;
[y,tt] = Eulr_D_back(fy,tspan(1),tspan(2),y0,h);
figure(1)
plot(tt,y);
xlabel("t")
ylabel("y")
```

3.Result: