به نام خدا

تمرین سوم متلب (امتیازی)

محاسبات عددی دکتر آریانیان
مجید صادقی نژاد - ۸۱۰۱۰۱۴۵۹

سوال اول

مقدمه:

خط ۲: پاک کردن تمام داده های قبلی بستن تمام پنجره ها
خط ۳: تعیین دقت اعشار
خطوط ۵ تا ۷: تعریف جواب معادله دیفرانسیل و مقدار دقیق آن
خطوط ۹ تا ۱۳: تعریف نقاط اولیه و مقادیر باقی نقاط و تعداد نقاط داده شده
خطوط ۱۵ تا ۱۷: تعریف ماتریس های جواب

بخش الف:

خطوط ۲۱ تا ۲۳: با استفاده از یک حلقه و تابع تعریف شده ی روش اویلر که مقادیر اولیه و طول گام را دریافت می کند جواب معادله را به ازای طول گام های مختلف محاسبه می کنیم. و میزان خطا را با استفاده از مقدار دقیق جواب محاسبه می کنیم سپس ماتریس های مقدار خطا و جواب روش اویلر به ازای h های مختلف را نمایش می دهیم

خطوط ۲۸ تا ۳۲: حال نمودار میزان خطا بر حسب طول گام را که یک نمودار با مقیاس لگاریتمی است را با استفاده از فانکشن پلات متلب رسم می کنیم.

تابع روش اویلر، خطوط ۶۳ تا ۷۵: ابتدا معادله دیفرانسیل را به صورت جبری تعریف می کنیم می دانیم که روش اویلر یک روش است که با تکرار جواب را محاسبه می کند پس در یک حلقه فرمول اویلر را نوشته و تعداد اجرا حلقه از تقسیم بازه بر طول گام به دست می آید. و در نهایت جواب را بر می گردانیم.

بخش ب:

خطوط ۳۶ تا ۴۱: با استفاده از یک حلقه و تابع تعریف شده ی روش رانگه کوتای مرتبه دوم که مقادیر اولیه و طول گام را دریافت می کند جواب معادله را به ازای طول گام های مختلف محاسبه می کنیم. و میزان خطا را با استفاده از مقدار دقیق جواب محاسبه می کنیم و در ماتریسی ذخیره می کنیم. سپس ماتریس های مقدار خطا و جواب روش رانگه کوتای مرتبه دوم را به ازای h های مختلف نمایش می دهیم.

خطوط ۴۳ تا ۴۷: حال نمودار میزان خطا بر حسب طول گام را که یک نمودار با مقیاس لگاریتمی است را با استفاده از فانکشن پلات متلب رسم می کنیم.

تابع روش رانگه کوتای مرتبه دوم، خطوط ۷۷ تا ۹۳: ابتدا معادله دیفرانسیل را به صورت جبری تعریف می کنیم و از آنجا که رانگه کوتای مرتبه دوم یک روش دارای تکرار است فرمول های رانگه کوتای مرتبه دوم را درون یک حلقه می گذاریم تا در هر مرحله از حلقه یک تکرار محاسبه شود تعداد تکرار این حلقه نیز از تقسیم بازه بر طول گام به دست می آید.

بخش ج:

خطوط ۵۱ تا ۵۶: با استفاده از یک حلقه و تابع تعریف شده ی روش رانگه کوتای مرتبه چهارم که مقادیر اولیه و طول گام را دریافت می کند جواب معادله را به ازای طول گام های مختلف محاسبه می کنیم. و میزان خطا را با استفاده از مقدار دقیق جواب محاسبه می کنیم و در ماتریسی ذخیره می کنیم. سپس ماتریس های مقدار خطا و جواب روش رانگه کوتای مرتبه چهارم را به ازای h های مختلف نمایش می دهیم.

خطوط ۵۸ تا ۶۲: حال نمودار میزان خطا بر حسب طول گام را که یک نمودار با مقیاس لگاریتمی است را با استفاده از فانکشن پلات متلب رسم می کنیم.

تابع روش رانگه کوتای مرتبه چهارم، خطوط ۹۵ تا ۱۱۳: ابتدا معادله دیفرانسیل را به صورت جبری تعریف می کنیم و از آنجا که رانگه کوتای مرتبه چهارم یک روش دارای تکرار است فرمول های رانگه کوتای مرتبه چهارم را درون یک حلقه می گذاریم تا در هر مرحله از حلقه یک تکرار محاسبه شود تعداد تکرار این حلقه نیز از تقسیم بازه بر طول گام به دست می آید.

```
error1 =
 0.689885485377136   0.373618105077135

eulerAns =
 9.46496   9.7812273803   9.95989311543326

error2 =
 0.0467209957771359   0.0126029455524623

rungeKutta2Ans =
 10.1081244896   10.1422425398247

error3 =
(0.0000920755593298746   0.00000625297163913201

rungeKutta4Ans =
(10.1547534098178   10.1548392324055   10.15484507
```

```m
%---- Information ---
clc, clear , close all
digits(15)

syms x f(x)
f(x)= 2*x+3*exp(x);
exact= f(1);

y0 = 3 ;
x0 = 0 ;

hValue=[0.2, 0.1, 0.05, 0.025, 0.02, 0.0125, 0.01, 0.00625, 0.003125, 0.002,
hNum=1;size(hValue,2);

eulerAns= zeros(1,hNum);
rungeKutta2Ans= zeros(1,hNum);
rungeKutta4Ans= zeros(1,hNum);

%---- Part A - Euler Method ---

for i=1:hNum
    eulerAns(1,i) = euler(x0,y0,hValue(1,i));
end

error1= vpa(abs(eulerAns-exact))
eulerAns= vpa(eulerAns)

f1=figure;
loglog(hValue,error1,'o','color','black','LineWidth',2)
title('Euler error per h')
xlabel('h')
ylabel('Error')

%---- Part B - rungeKutta 2 Method ---

for i=1:hNum
    rungeKutta2Ans(1,i) = rungeKutta2(x0,y0,hValue(1,i));
end

error2= vpa(abs(rungeKutta2Ans-exact))
rungeKutta2Ans= vpa(rungeKutta2Ans)

f2=figure;
loglog(hValue,error2,'o','color','red','LineWidth',2)
title('rungeKutta 2 error per h')
xlabel('h')
ylabel('Error')

%---- Part C - rungeKutta 4 Method ---

for i=1:hNum
    rungeKutta4Ans(1,i) = rungeKutta4(x0,y0,hValue(1,i));
end

error3= vpa(abs(rungeKutta4Ans-exact))
rungeKutta4Ans= vpa(rungeKutta4Ans)

f3=figure;
loglog(hValue,error3,'o','color','blue','LineWidth',2)
title('rungeKutta 4 error per h')
xlabel('h')
ylabel('Error')

function result = euler(xn,yn,h)
    syms x y f(x,y)
    f(x,y)= y-2*x+2;

    n= 1/h ;

    for i= 1:n
        yn = yn + h*f(xn,yn);
        xn= xn+h;
    end

    result= yn ;
end

function result = rungeKutta2(xn,yn,h)
    syms x y f(x,y)
    f(x,y)= y-2*x+2;

    n= 1/h ;

    for i= 1:n
        k1=h*f(xn,yn);

        k2=h*f(xn+h, yn+k1);

        yn = yn + 0.5*(k1+k2);
        xn= xn + h;
    end

    result= yn ;
end

function result = rungeKutta4(xn,yn,h)
    syms x y f(x,y)
    f(x,y)= y-2*x+2;

    n= 1/h ;

    for i= 1:n

        k1=h*f(xn,yn);
        k2=h*f(xn+(h/2), yn+(k1/2));
        k3=h*f(xn+(h/2), yn+(k2/2));
        k4=h*f(xn+h, yn+k3);

        yn = yn + (1/6)*(k1+2*k2+2*k3+k4);
        xn= xn + h;
    end

    result= yn ;
end
```

سوال دوم

مقدمه:

خطوط ۱ تا ۵: تمامی داده های قبلی پاک می شود و پنجره های قبلی بسته می شوند دقت اعشار تعیین می شود در ادامه متغیر x و تابع f(x) به صورت جبری تعریف می شوند

خطوط ۷ تا ۱۰: تابع انتگرال ده و ابتدا و انتهای بازه انتگرال گیری تعیین می شوند همچنین طول گام مورد نیاز برای انتگرال گیری عددی تعیین می شود.

بخش الف:

خط ۱۴: با استفاده از تابع روش انتگرال گیری ذوزنقه ای که تابع انتگرال ده ، ابتدا و انتهای بازه ی انتگرال گیری و طول گام را دریافت می کند، مقدار انتگرال محاسبه و نمایش داده می شود.

تابع روش ذوزنقه ای، خطوط ۲۴ تا ۳۹: ابتدا تابع انتگرال ده به صورت یک تابع جبری دریافت می شود سپس تعداد نقاطی که برای این انتگرال گیری لازم است از تقسیم طول بازه بر طول گام یافت می شود و در نهایت با استفاده از یک حلقه جمع نقاطی که در فرمول ذوزنقه وجود دارد انجام می شود لازم به ذکر است که نقطه ی ابتدایی و انتهایی که ضریب آن ها متفاوت است به صورت جدا گانه جمع می شوند و در پایان جواب بازگردانده می شود.

بخش ب:

خط ۱۸: با استفاده از تابع روش انتگرال گیری سیمپسون که تابع انتگرال ده ، ابتدا و انتهای بازه ی انتگرال گیری و طول گام را دریافت می کند، مقدار انتگرال محاسبه و نمایش داده می شود.

تابع روش سیمپسون، خطوط ۴۱ تا ۶۰: ابتدا تابع انتگرال ده به صورت یک تابع جبری دریافت می شود سپس تعداد نقاطی که برای این انتگرال گیری لازم است از تقسیم طول بازه بر طول گام یافت می شود و در نهایت با استفاده از یک حلقه جمع نقاط با اندیس فرد که در فرمول سیمپسون وجود دارد انجام می شود و با استفاده از حلقه ی دیگری جمع نقاط با اندیس زوج انجام می شود. لازم به ذکر است که نقطه ی ابتدایی و انتهایی که ضریب آن ها متفاوت است به صورت جدا گانه جمع می شوند و در پایان جواب بازگردانده می شود.

بخش ج:

خطوط ۲۲ تا ۲۳: با استفاده از تابع روش انتگرال گیری گاوس که تابع انتگرال ده ، ابتدا و انتهای بازه ی انتگرال گیری، طول گام و نوع روش گاوس (دو نقطه ای یا سه نقطه ای) را دریافت می کند، مقدار انتگرال محاسبه و نمایش داده می شود.

تابع روش سیمپسون، خطوط ۶۲ تا ۷۸: ابتدا متغیر های x و t را به صورت جبری تعریف می کنیم تا بتوانیم تغییر متغیر گاوس را انجام دهیم سپس این تغییر متغیر انجام می شود و تابع جدید بر حسب t تعریف می شود. و مقدار های عددی که در فرمول گاوس استفاده می شوند هم تعریف می شوند. سپس با دو شرط بررسی می شود که لازم است فرمول گاوس دو نقطه ای اجرا شود یا فرمول گاوس سه نقطه ای ، در نهایت جواب باز گردانده می شود.

```
trapeziumAns = 0.629838064980291
```

```
SimpsonAns = 0.632131438789757
```

```
gauss2Ans = 0.625507241177351
```

```
gauss3Ans = 0.632280553856818
```

```
%---- Information ---
clc, clear , close all
digits(15)

syms x f(x)

integralFunc= 2*x*exp(-(x^2));
top=1;
floor=0;
h=0.1;

%--- Part A - trapezium Method ---

trapeziumAns= vpa(trapezium(floor,top,h,integralFunc))

%--- Part B - Simpson Method ---

SimpsonAns= vpa(Simpson(floor,top,h,integralFunc))

%--- Part C - gauss 2 & 3 Method ---

gauss2Ans= vpa(gauss(floor,top,integralFunc,2))
gauss3Ans= vpa(gauss(floor,top,integralFunc,3))

function result = trapezium(floor,top,h,integralFunc)
    syms x f(x)
    f(x)= integralFunc;

    n= (top-floor)/h;

    result= f(floor);

    for i= 1:(n-1)
        result= result+ 2*f(floor+(i*h)) ;
    end

    result= result + f(top) ;

    result= (h/2)*result ;
end

function result = Simpson(floor,top,h,integralFunc)
    syms x f(x)
    f(x)= integralFunc;

    n= (top-floor)/h;

    result= f(floor);

    for i= 1:2:(n-1)
        result= result+ 4*f(floor+(i*h)) ;
    end

    for i= 2:2:(n-1)
        result= result+ 2*f(floor+(i*h)) ;
    end

    result= result + f(top) ;

    result= (h/3)*result ;
end

function result = gauss(floor,top,integralFunc,kind)
    syms x t f(x) a(t)
    f(x)= integralFunc ;

    a(t)= ((top-floor)/2)*t+(top+floor)/2 ;

    A=(3^(1/2))/3;
    B=(3/5)^(1/2);

    if kind==2
        result= diff(a(t),t)*(f(a(A))+f(a(-A))) ;
    end

    if kind==3
        result= diff(a(t),t)*(1/9)*(5*f(a(B))+5*f(a(-B))+8*f(a(0))) ;

    end
end

```