

Istanbul Technical University

Department of Computer Engineering

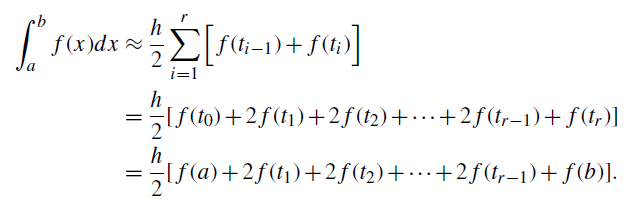
BLG 202E – Numerical Methods

Assignment 4

Solutions

**Solution 1**

The composite trapezoidal method is



So that in this question, trapezoidal equation is for interval [0, 3] with N = 10

(In this case step size h must be (b-a)/N = 0.3)

syms x

f = inline((2.718182.^-x)\*sin(x), 'x');

h = 0.3

xi = 0;

yn = 0; %% new approximate

yp = f(xi); %% previous approximate

sum = 0;

while xi < 3

xi = xi + h;

sum = sum + 2\*f(xi)

yp = yn;

end

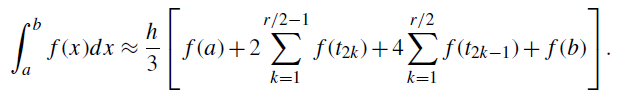
sum = sum + f(3);

result = sum \* h/2

, where number of interval is 10. For small number of interval, error is nor smaller than For just ten interval, absolute error is 0.0018(small than )

**Solution 2**

The composite Simpson’s rule is



And exact integration is:

interval 0 and 4, exact value is

5216.426477

Absolute relative true error = 0.0075

syms x

h = 1;

xi = 0;

xs = 4;

yn = 0; %% new approximate

yp = f(xi); %% previous approximate

sum = 0;

f = inline((2.718182.^2\*x)\*x, 'x');

%%Isimp = h/3\*(f(xi) + 2\*f(xi+2\*h) + 4\*f(xi + h) + 4\*f(xi+3\*h)+f(xs))

Isimp = h/3\*(f(0) + 2\*f(2) + 4\*f(1) + 4\*f(3)+f(4))

fun = @(x) exp(2\*x).\*x;

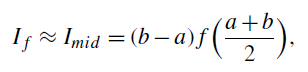
ExactIntegral = integral(fun, 0, 4)

yu = 5256

error = abs(ExactIntegral - yu)/ExactIntegral

**Solution 3**

The basic midpoint rule as like following



So we need to find all midpoint which given number of midpoints. For example m=2, number of midpoint is 2 and

m0 = 0.5

m1 = 1.5

General formula of MATLAB code for every midpoint is

syms x

f = inline(1+(2.718182.^-x)\*sin(8\*x^(2/3)), 'x');

m = [2,4,8,16,32,60,70,100];

h = 2./m;

i = 1;

first = 0;

second = 0 + h;

mid\_first = (first + second) / 2;

result = 0;

mid = 0;

while i < 9

mid = mid\_first(i);

while mid < 2

result = result + f(mid);

mid = mid + h(i);

end

result = result \* h(i);

results(i) = result;

mid = 0;

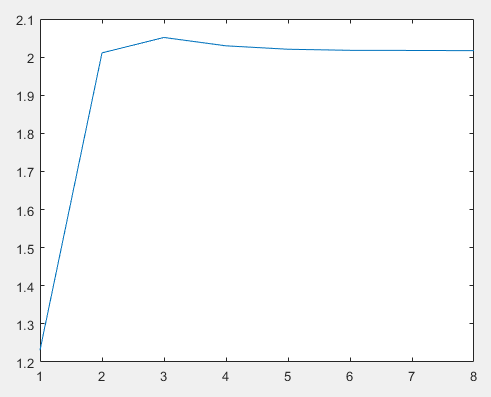
result = 0;

i = i + 1

end

results

plot(results)



*approximation results according to the number of sample values*

**Solution 4**

Runge Kutta 2 stage MATLAB code as following:

clear;

to = 0; %%integrated from 0

tf = 100; %%integrated to 100

yo = [80 30]; %%initial value

options = odeset('RelTol', 1e-5);

[t y] = ode45('y1y2',[to tf],yo,options);

plot(y(:,1),y(:,2))

ylabel('prey')

xlabel('predator')

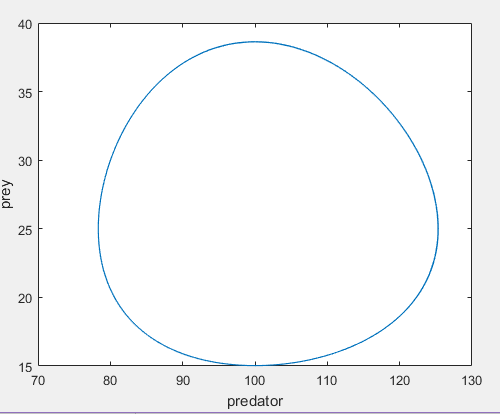
function y1y2 =y1y2(t,y)

y1y2(1) = .25\*y(1) - .01\*y(1)\*y(2);

y1y2(2) = -y(2) +.01\*y(1)\*y(2);

y1y2 = [y1y2(1) y1y2(2)]';

end



*predator-prey solution in phase plane: the curve of y1(t) vs. y2(t) yields a limit cycle*

**Solution 5**

Calculation of error in each step size value for Forward Euler Method.

s = [.00005,.0001,.0005,.001];

h = [.00005,.0001,.0005,.001]\*pi;

N = 1./[.00005,.0001,.0005,.001];

N = N/2;

y0 = 1;

result\_forward = [0,0,0,0];

i = 1;

while i < 5

y = y0; t = 0;

for k=1:N(i)

y = y + h(i)\*(-1000\*(y - cos (t)) - sin (t));

t = t + h(i);

end

result\_forward(i) = y;

i

i = i + 1;

end

plot(result\_forward)

i = 1;

while i < 5

fprintf('Approximation Result for h = %i: %i\n', s(i), result\_forward(i))

i = i + 1;

end

After the calculation, error results will be as following:

Approximation Result for h = 5.000000e-05: 7.421676e-11

Approximation Result for h = 1.000000e-04: 1.405548e-10

Approximation Result for h = 5.000000e-04: 3.741929e-10

Approximation Result for h = 1.000000e-03: -3.669057e+159

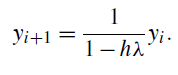
It can be seen that after the value h = .0005π, we get the error as -3.6e+159 for h = .001π which means that a blowup has occurred. This result from stability difference between forward and backward Euler method.

In Forward Euler Method,



the constant step size h must not exceed a certain bound that depends on the problem which is being approximately solved.

However; in Backward Euler Method, there is no restriction about step size. Because,



For Backward Euler Method for any h > 0 and λ < 0, there is no absolute stability restriction.

**For h = 0.01π,**

Error in Forward Euler = -3.6e+159

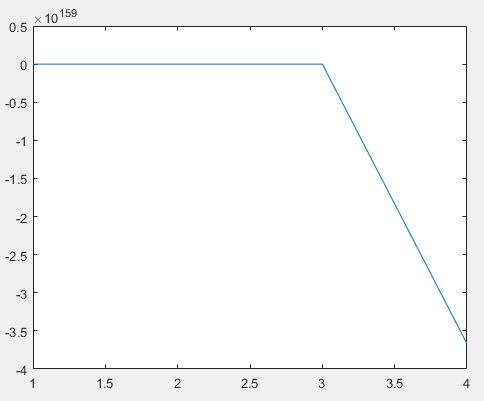
Error in Backward Euler = -3.2e-9

Where for this problem , then

Conclusion, h ≥0.01π, the result blows up

h ≤ 0.002, there is no problem for Forward Euler

For Backward Euler, there is no any problem for any h or .



In this figure, we can see blowup point.