Test 2

AE\_5830 Dr. Hosder

Matt Pahayo

2021

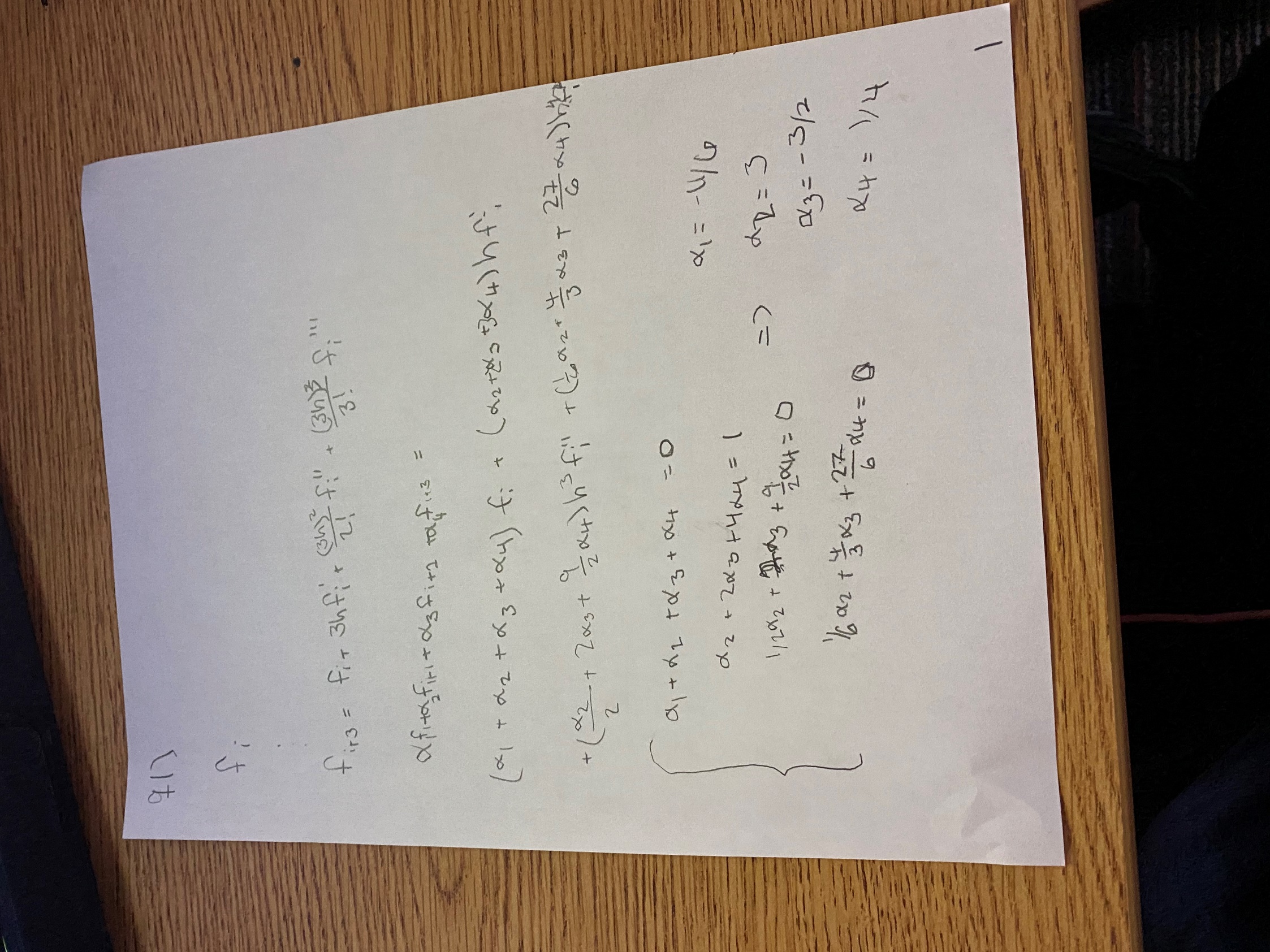
**Q1;**

Text, letter

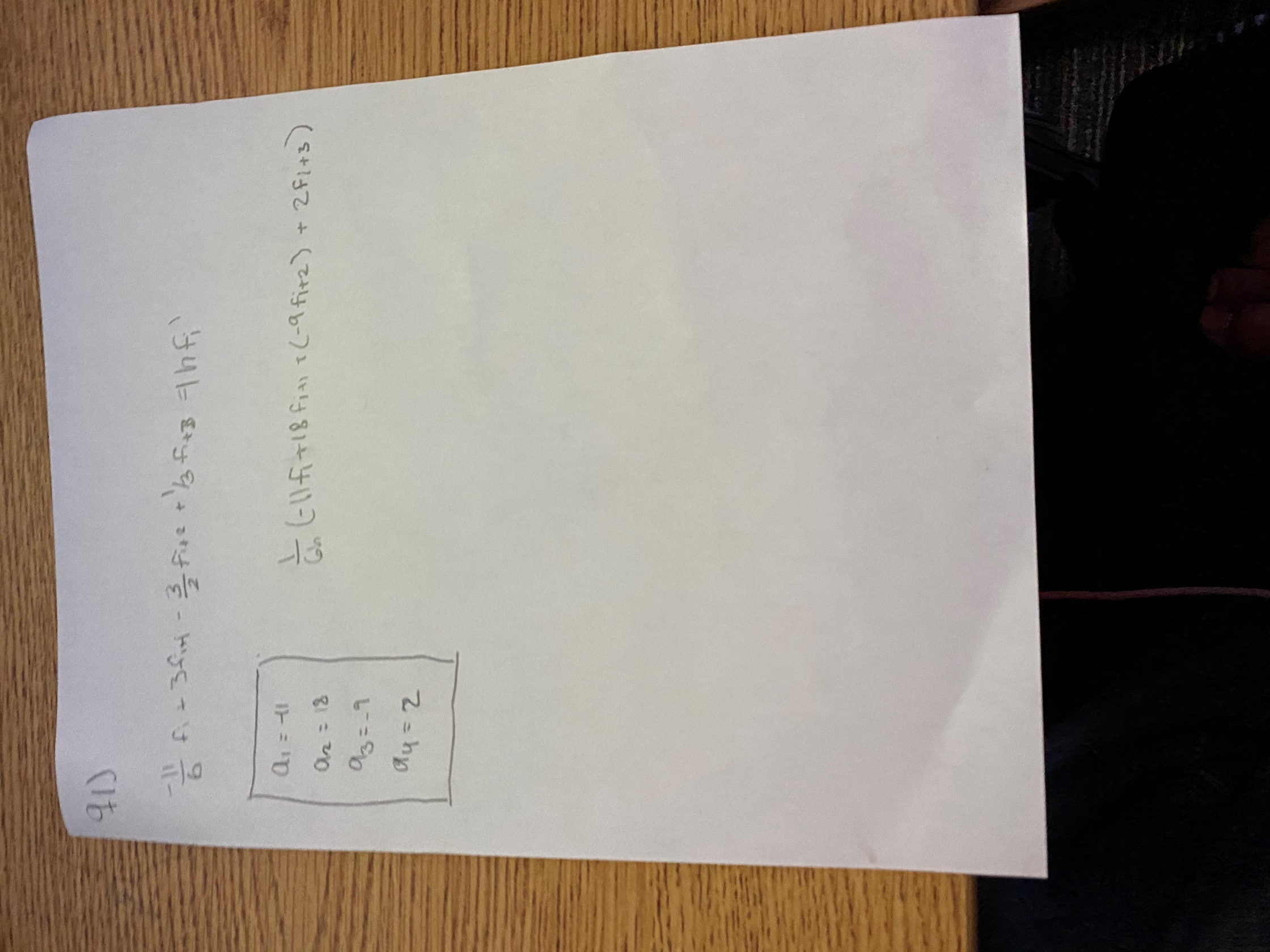
Description automatically generated



1. **fwd\_fprimeO3.m**

****

**Fig, 1 Work for Question 1, page 1**

****

**Fig, 1 Work for Question 1, page 2**

**solve with favorite method =>**

**=>**

**Table 1 question 1 B**

****

As shown in Table 1 the absolute error significantly decreases when the step size is decreased. (3 orders of magnitude per 1 order of magnitude change in step size).

Q2:

Text

Description automatically generated

Text, letter

Description automatically generated

Spline.m and gauss.m

**Chart, line chart

Description automatically generated**

**Fig 3. Spline plot**

Chart, line chart

Description automatically generated

**Fig, 4 Spline Error**

**Table 2 Question 2**

****

The Natural cubic spline has less error at the positive end and more at the negative end than the Clamped spline, so you would choose the type of spline with the least error at the desired location.

Q3:

Text, letter

Description automatically generated

The exact value of an integral can be achieved using the Gauss quadrature method if the max degree of the function times 2 minus 1is less than or equal to the number of points used in the Gauss quadrature method. The max degree of the function in the y-direction is 3, so the minimum number of points to get the exact value of the integral would be **two**. The minimum points in the x-direction would be **three**.

Where p is the point used in the quadrature method.

gaussQuad.m

**Table 3 Question 3**



Q4:

Text, letter

Description automatically generatedChart, line chart

Description automatically generated

**Fig. 5 All ODE Approximations**

**Chart, line chart

Description automatically generated**

**Fig. 6 All Errors**

**Chart, line chart

Description automatically generated**

**Fig. 7 RK4 Error**

**Table 4 Question 4**

****

An advantage of the RK4 is that it is easy to code, and it is quite accurate. It is not good at discontinuities. The two step Adams-Bashforth method is second most accurate method out of the three. A disadvantage of this is that it is not self-starting, i.e., you need a self-starting method with an order greater than or equal to this method. The Major advantage of Euler’s method is that it is the simplest algorithm, but it has the worst accuracy of the three methods; to get more reliable results the step size must be decreased further which may or may not be costly.

RK4.m, AB2.m, and Euler.m

**Appendix**

% main.m

clc

clear all

close all

format longg

%=========================================================================

%q1

%=========================================================================

fun = @(x) x^2\*sin(x)-2\*x;

fp(1,1) = fwd\_fprimeO3(fun,-3,.1);

fp(2,1) = fwd\_fprimeO3(fun,-3,.01);

fp(3,1) = fwd\_fprimeO3(fun,-3,.001);

h(1,1) = .1;

h(2,1) = .01;

h(3,1) = .001;

exact = -10.063212421;

err(1,1) = abs(exact-fp(1,1));

err(2,1) = abs(exact-fp(2,1));

err(3,1) = abs(exact-fp(3,1));

[h fp err]

%=========================================================================

%q2

%=========================================================================

f = @(x) x.^2.\*sin(x)-2.\*x;

syms p

xend = 4;

xstart = -3;

b = xend - xstart;

exact = -10.063212421;

intervals = 13;

n = intervals;

x = transpose(xstart:(xend-xstart)/intervals:xend);

g = f(x);

pClamp = Spline.Clamp(f,g,n,x,xstart,.5,xend,exact);

pNC = (Spline.NC(g,n,x));

NC = sym(zeros(b\*100+1,1));

Clamp = NC;

i = xstart ;

j=1;

while(true)

if i>=-3 && i<xstart+b/n

NC(j) = subs(pNC(1),p,i);

Clamp(j) = subs(pClamp(1),p,i);

end

if i>=xstart+b/n && i<xstart+2\*b/n

NC(j) = subs(pNC(2),p,i);

Clamp(j) = subs(pClamp(2),p,i);

end

if i>=xstart+2\*b/n && i<xstart+3\*b/n

NC(j) = subs(pNC(3),p,i);

Clamp(j) = subs(pClamp(3),p,i);

end

if i>=xstart+3\*b/n && i<xstart+4\*b/n

NC(j) = subs(pNC(4),p,i);

Clamp(j) = subs(pClamp(4),p,i);

end

if i>=xstart+4\*b/n && i<xstart+5\*b/n

NC(j) = subs(pNC(5),p,i);

Clamp(j) = subs(pClamp(5),p,i);

end

if i>=xstart+5\*b/n && i<xstart+6\*b/n

NC(j) = subs(pNC(6),p,i);

Clamp(j) = subs(pClamp(6),p,i);

end

if i>=xstart+6\*b/n && i<xstart+7\*b/n

NC(j) = subs(pNC(7),p,i);

Clamp(j) = subs(pClamp(7),p,i);

end

if i>=xstart+7\*b/n && i<xstart+8\*b/n

NC(j) = subs(pNC(8),p,i);

Clamp(j) = subs(pClamp(8),p,i);

end

if i>=xstart+8\*b/n && i<xstart+9\*b/n

NC(j) = subs(pNC(9),p,i);

Clamp(j) = subs(pClamp(9),p,i);

end

if i>=xstart+9\*b/n && i<xstart+10\*b/n

NC(j) = subs(pNC(10),p,i);

Clamp(j) = subs(pClamp(10),p,i);

end

if i>=xstart+10\*b/n && i<xstart+11\*b/n

NC(j) = subs(pNC(11),p,i);

Clamp(j) = subs(pClamp(11),p,i);

end

if i>=xstart+11\*b/n && i<xstart+12\*b/n

NC(j) = subs(pNC(12),p,i);

Clamp(j) = subs(pClamp(12),p,i);

end

if i>=xstart+12\*b/n && i<xstart+13\*b/n

NC(j) = subs(pNC(13),p,i);

Clamp(j) = subs(pClamp(13),p,i);

end

i=i+.01;

j =j+1;

if i>=b

break;

end

end

dom = transpose(xstart:.01:xend);

exact = f(dom);

errNC = abs(NC-exact);

errClamp = abs(Clamp-exact);

figure(1)

hold on

plot(dom,NC)

plot(dom,Clamp)

fplot(f)

xlim([-3 3])

xlabel('x')

ylabel('f(x)')

legend('Natural Cubic','Clamped','Exact')

title('Spline Approximations')

grid on

hold off

figure(2)

hold on

plot(dom,errNC)

plot(dom,errClamp)

xlim([-3 3])

xlabel('x')

ylabel('error')

legend('Natural Cubic','Clamped')

title('Spline Absolute Error')

grid on

hold off

sp = [-2.8;-1.4;0;1.4;2.8];

exactSP = f(sp);

NCsp = double([subs(pNC(1),p, -2.8); subs(pNC(3),p, -1.4); subs(pNC(7),p,0);subs(pNC(10),p,1.4);subs(pNC(12),p,2.8)])

Clampsp = double([subs(pClamp(1),p, -2.8); subs(pClamp(3),p, -1.4); subs(pClamp(7),p,0);subs(pClamp(10),p,1.4);subs(pClamp(12),p,2.8)])

errNCsp = abs(exactSP-NCsp)

errClampsp = abs(exactSP-Clampsp)

%=========================================================================

%q3

%=========================================================================

syms x % y-direction is going first

f = @(y) (2\*x^5+x^2/3+x)\*(y^3-y/4+1)

g(x) = gaussQuad(f,0,3,2)

I = gaussQuad(g,0,2,3)

double(I)

%=========================================================================

%q4

%=========================================================================

syms t

f(t) = sqrt(4-3\*exp(-t^2));

xi = 0;

yi = 1;

h = .1;

steps = 25+1;

[xRK4,yRK4] = RK4(xi,yi,h,steps);

[xe,ye] = Euler(xi,yi,h,steps);

[xab2,yab2] = AB2(xi,yi,h,steps);

figure (1)

hold on

fplot(f)

plot(xRK4,yRK4)

plot(xe,ye)

plot(xab2,yab2)

xlim([xi 2.5])

grid on

xlabel('t')

ylabel('f(t)')

title('ODE approximations')

legend('Exact','RK4','Euler','AB2')

x = transpose(0:.1:2.5);

exact = zeros(steps,1);

for i = 1:steps

exact(i,1) = f(x(i,1));

end

errRK4 = abs(yRK4-exact);

errAB2 = abs(yab2-exact);

errE = abs(ye-exact);

tab = [x exact yRK4 errRK4 yab2 errAB2 ye errE];

figure (2)

hold on

plot(xRK4,errRK4)

plot(xe,errE)

plot(xab2,errAB2)

xlim([xi 2.5])

grid on

xlabel('t')

ylabel('Absolute Error')

title('ODE Error')

legend('RK4','Euler','AB2')

figure (4)

hold on

plot(xRK4,errRK4)

xlim([xi 2.5])

grid on

xlabel('t')

ylabel('Absolute Error')

title('RK4 Error')

function dfdx = fwd\_fprimeO3(f,x,h)

dfdx = (-11/6\*f(x)+3\*f(x+h)-3/2\*f(x+2\*h)+1/3\*f(x+3\*h))/h;

end

function I = gaussQuad(f,a,b,n)

%myFun - Description

%

% Syntax: output = myFun(input)

%

% Long description

switch n

case 2

c = ones(2,1);

t = [-1/sqrt(3);1/sqrt(3)];

case 3

c = [5/9;8/9;5/9];

t = [-sqrt(3/5);0;sqrt(3/5)];

case 4

c = [.347855;.652145;.652145;.347855];

t = [-.861136;-.339981;.339981;.861136];

otherwise

disp('nope')

end

x = zeros(n,1);

for i = 1:n

x(i,1) = (b+a)/2-(b-a)/2\*t(i);

end

I = 0;

for i = 1:n

I = (b-a)/2\*c(i)\*f(x(i)) + I;

end

classdef Spline

%UNTITLED Summary of this class goes here

% Detailed explanation goes here

methods (Static)

function [s] = NC(g,n,x)

%sysMake will Construct a system for NC spline

% it will find the coefficents of each spline and will output

% a column vector of the spline equations as syms

% g is the vector of known points, x is the vector coresponding

% to the points

% n is the number of intervals inbetween all points

syms p

h = zeros(n,1);

for i = 1:n-1

h(i,1) = x(i+1)-x(i);

end

A = zeros(n);

A(1,1) = 1;

A(n,n) = 1;

for i = 2:n-1

A(i,i) = 2\*(h(i-1)+h(i));

end

for i = 1:n-1

A(i+1,i) = h(i);

A(i,i+1) = h(i);

end

A(1,2) = 0;

A(n,n-1) = 0 ;

b = zeros(n,1);

for i = 2:n-1

b(i,1) = 3/h(i)\*(g(i+1)-g(i))-3/h(n-1)\*(g(i)-g(i-1));

end

c = gauss(A,b);

k = zeros(n,1);

d = zeros(n,1);

for i = 1:n-1

k(i,1) = (g(i+1)-g(i))/h(i)-h(i)/3\*(2\*c(i)+c(i+1));

d(i,1) = (c(i+1)-c(i))/3/h(i);

end

s = sym(zeros(n,1));

for i = 1 :n

s(i,1) = (g(i)+k(i)\*(p-x(i))+c(i)\*(p-x(i))^2+d(i)\*(p-x(i))^3);

end

end

function [s] = Clamp(f,g,n,x,x0,h0,xn,exact)

%sysMake will Construct a system for NC spline

% it will find the coefficents of each spline and will output

% a column vector of the spline equations as syms

% g is the vector of known points, x is the vector coresponding

% to the points

% n is the number of intervals inbetween all points

syms p

h = zeros(n,1);

for i = 1:n-1

h(i,1) = x(i+1)-x(i);

end

A = zeros(n);

A(1,1) = 2\*h(i-1);

A(n,n) = A(1,1);

for i = 2:n-1

A(i,i) = 2\*(h(i-1)+h(i));

end

for i = 1:n-1

A(i+1,i) = h(i);

A(i,i+1) = h(i);

end

b = zeros(n,1);

b(1,1) = 3/h(1)\*(g(2)-g(1))-3\*fwd\_fprimeO3(f,x0,h0);

b(n,1) = -3/h(end-1)\*(g(end)-g(end-1))+3\*exact;

for i = 2:n-1

b(i,1) = 3/h(i)\*(g(i+1)-g(i))-3/h(i-1)\*(g(i)-g(i-1));

end

c = gauss(A,b);

k = zeros(n,1);

d = zeros(n,1);

for i = 1:n-1

k(i,1) = (g(i+1)-g(i))/h(i)-h(i)/3\*(2\*c(i)+c(i+1));

d(i,1) = (c(i+1)-c(i))/3/h(i);

end

s = sym(zeros(n,1));

for i = 1 :n

s(i,1) = (g(i)+k(i)\*(p-x(i))+c(i)\*(p-x(i))^2+d(i)\*(p-x(i))^3);

end

end

end

end

function dydx = Deriv(x,y)

dydx = -x\*y+4\*x/y;

% dxdy = x\*exp(3\*x)-2\*y;

% dxdy = y/x-(y/x)^2;

end

function [x,y] = RK4(xi,yi,h,steps)

x = zeros(steps,1);

y = x;

x(1,1) = xi ;

y(1,1) = yi ;

for i = 2:steps

x(i,1) = x(i-1) + h;

k1 = h\*Deriv(x(i-1),y(i-1));

k2 = h\*Deriv(x(i-1)+h/2,y(i-1)+k1/2);

k3 = h\*Deriv(x(i-1)+h/2,y(i-1)+k2/2);

k4 = h\*Deriv(x(i-1)+h,y(i-1)+k3);

y(i) = y(i-1) + 1/6\*(k1+2\*k2+2\*k3+k4);

end

end

function [x,y] = Euler(xi,yi,h,steps)

x = zeros(steps,1);

y = x;

dydx = x;

x(1,1) = xi;

y(1,1) =yi;

for i = 2:steps

x(i,1) = x(i-1) + h;

dydx(i,1) = Deriv(x(i-1),y(i-1));

y(i,1) = y(i-1)+h\*dydx(i,1);

end

end

function [x,y] = AB2(xi,yi,h,steps)

% Rk4

x = zeros(steps,1);

y = x;

x(1,1) = xi ;

y(1,1) = yi ;

for i = 2:steps

x(i,1) = x(i-1) + h;

k1 = h\*Deriv(x(i-1),y(i-1));

k2 = h\*Deriv(x(i-1)+h/2,y(i-1)+k1/2);

k3 = h\*Deriv(x(i-1)+h/2,y(i-1)+k2/2);

k4 = h\*Deriv(x(i-1)+h,y(i-1)+k3);

y(i) = y(i-1) + 1/6\*(k1+2\*k2+2\*k3+k4);

end

%AB2

for i = 3:steps

x(i,1) = x(i-1) + h;

y(i) = y(i-1) + h\*(3/2\*Deriv(x(i-1),y(i-1))-1/2\*Deriv(x(i-2),y(i-2)));

end

end

function [x] = gauss(a,b)

% gauss elimination

n = length(a);

k = 1 ;

p = k ;

big = abs(a(k,k));

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% pivoting portion

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for ii=k+1:n

dummy = abs(a(ii,k));

if dummy > big

big = dummy;

p = ii ;

end

end

if p ~= k

for jj = k:n

dummy = a(p,jj);

a(p,jj) = a(k,jj);

a(k,jj) = dummy;

end

dummy = b(p);

b(p)=b(k);

b(k) = dummy;

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% elimination step

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for k=1:(n-1)

for i=k+1:n

factor = a(i,k)/a(k,k);

for j=k+1:n

a(i,j) = a(i,j) - factor\*a(k,j);

end

b(i) = b(i) - factor\*b(k);

end

end

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

% back substitution

%\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

x(n,1) = b(n)/a(n,n);

for i = n-1:-1:1

sum = b(i);

for j = i + 1:n

sum = sum - a(i,j)\*x(j,1);

end

x(i,1) = sum/a(i,i);

end

end