Joe Ingenito

3/27/19

Matlab Project 2

Exercise 1)

function [x1,x2] = quadratic(a,b,c)

disc = sqrt((b^2) - 4\*a\*c);

x1 = ((-1)\*b + disc)/(2\*a);

x2 = ((-1)\*b - disc)/(2\*a);

end

(a)

>> [x1,x2] = quadratic(1,0,-1)

x1 =

1

x2 =

-1

(b)

>> [x1,x2] = quadratic(1,0,1)

x1 =

0.0000 + 1.0000i

x2 =

0.0000 - 1.0000i

(c)

>> [x1,x2] = quadratic(1,2,1)

x1 =

-1

x2 =

-1

(d)

>> [x1,x2] = quadratic(8,10,-3)

x1 =

0.2500

x2 =

-1.5000

Exercise 2)

function [x,y] = Eul(h,x0,y0,xf)

k = 1;

x(k) = x0;

y(k) = y0;

while(x(k) + h <= xf + 10^-10)

x(k+1) = x(k) + h;

y(k+1) = y(k) + h\*(x(k) + y(k));

k = k + 1;

end

end

>> [x,y]=Eul(0.1, 0, 2, 3)

x =

Columns 1 through 11

0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000

Columns 12 through 22

1.1000 1.2000 1.3000 1.4000 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000 2.1000

Columns 23 through 31

2.2000 2.3000 2.4000 2.5000 2.6000 2.7000 2.8000 2.9000 3.0000

y =

Columns 1 through 11

2.0000 2.2000 2.4300 2.6930 2.9923 3.3315 3.7147 4.1462 4.6308 5.1738 5.7812

Columns 12 through 22

6.4594 7.2153 8.0568 8.9925 10.0317 11.1849 12.4634 13.8798 15.4477 17.1825 19.1007

Columns 23 through 31

21.2208 23.5629 26.1492 29.0041 32.1545 35.6300 39.4630 43.6893 48.3482

plot(x,y)

The value of y(3) with h = 0.1 is closer to the real value of y(3) because the step size is smaller so the tangent lines will deviate less from the real curve.

Exercise 3)

function [x,y] = Eul\_3(h,x0,y0,xf)

k = 1;

x(k) = x0;

y(k) = y0;

while(x(k) + h <= xf + 10^-10)

x(k+1) = x(k) + h;

y(k+1) = y(k) + h\*(x(k) \* exp(3\*x(k)) - 2 \* y(k));

k = k + 1;

end

end

>> [x,y] = Eul(0.25,0,0,2)

x =

0 0.2500 0.5000 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000

y =

0 0 0.1323 0.6264 2.0921 6.0675 16.3216 41.9172 104.3313

Mathematica Solution:

DSolve[y'[x] == x\*Exp[3\*x] - 2\*y[x], y[x], x]

Out: {{y[x] -> e^(3 x) (-(1/25) + x/5) + e^(-2 x) C[1]}}

y[x\_] := Exp[3\*x]\*(-(1/25) + x/5) + Exp[-2\*x]\*C

Solve[y[0] == 0]

Out: {{C -> 1/25}}

y(x) = e^(-2x)/25 + e^(3x)(-(1/25) + x/5)

>> xvec = 0:0.1:2;

>> yvec = exp(-2.\*xvec).\*(1/25) + exp(3.\*xvec).\*(-1/25) + exp(3.\*xvec).\*(xvec.\*(1/5));

>> plot(xvec,yvec,"blue",x,y,"red")



Exercise 4)

function [x,y] = Eul\_improved\_4(h, x0, y0, xf)

k = 1;

x(k) = x0;

y(k) = y0;

while (x(k)+h<=xf+10^-10)

x(k+1) = x(k)+h;

y\_predict = y(k)+h\*(x(k)\*exp(3\*x(k))-2\*y(k));

y(k+1) = y(k)+0.5\*h\*((x(k)\*exp(3\*x(k))-2\*y(k)) + (x(k+1)\*exp(3\*x(k+1))-2\*y\_predict));

k = k+1;

end

>> [x,y] = Eul\_improved\_4(0.25, 0, 0, 2)

x =

0 0.2500 0.5000 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000

y =

0 0.0662 0.3545 1.2511 3.7374 10.2351 26.5971 66.7487 163.4183

>> plot(xvec,yvec,"blue",x,y,"red")



Exercise 5)

(a)

function [x,y] = Eul\_5(h,x0,y0,xf)

k = 1;

x(k) = x0;

y(k) = y0;

while(x(k) + h <= xf + 10^-10)

x(k+1) = x(k) + h;

y(k+1) = y(k) + h\*(-2\*y(k));

k = k + 1;

end

end

>> [x1,y1] = Eul\_5(0.1, 0, 3, 5)

x1 =

Columns 1 through 11

0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000

Columns 12 through 22

1.1000 1.2000 1.3000 1.4000 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000 2.1000

Columns 23 through 33

2.2000 2.3000 2.4000 2.5000 2.6000 2.7000 2.8000 2.9000 3.0000 3.1000 3.2000

Columns 34 through 44

3.3000 3.4000 3.5000 3.6000 3.7000 3.8000 3.9000 4.0000 4.1000 4.2000 4.3000

Columns 45 through 51

4.4000 4.5000 4.6000 4.7000 4.8000 4.9000 5.0000

y1 =

Columns 1 through 11

3.0000 2.4000 1.9200 1.5360 1.2288 0.9830 0.7864 0.6291 0.5033 0.4027 0.3221

Columns 12 through 22

0.2577 0.2062 0.1649 0.1319 0.1056 0.0844 0.0676 0.0540 0.0432 0.0346 0.0277

Columns 23 through 33

0.0221 0.0177 0.0142 0.0113 0.0091 0.0073 0.0058 0.0046 0.0037 0.0030 0.0024

Columns 34 through 44

0.0019 0.0015 0.0012 0.0010 0.0008 0.0006 0.0005 0.0004 0.0003 0.0003 0.0002

Columns 45 through 51

0.0002 0.0001 0.0001 0.0001 0.0001 0.0001 0.0000

(b)

function [x,y] = Eul\_improved\_5(h, x0, y0, xf)

k = 1;

x(k) = x0;

y(k) = y0;

while (x(k)+h<=xf+10^-10)

x(k+1) = x(k)+h;

y\_predict = y(k)+h\*(-2\*y(k));

y(k+1) = y(k)+0.5\*h\*((-2\*y(k)) + (-2\*y\_predict));

k = k+1;

end

>> [x2,y2] = Eul\_improved\_5(0.1, 0, 3, 5)

x2 =

Columns 1 through 13

0 0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.8000 0.9000 1.0000 1.1000 1.2000

Columns 14 through 26

1.3000 1.4000 1.5000 1.6000 1.7000 1.8000 1.9000 2.0000 2.1000 2.2000 2.3000 2.4000 2.5000

Columns 27 through 39

2.6000 2.7000 2.8000 2.9000 3.0000 3.1000 3.2000 3.3000 3.4000 3.5000 3.6000 3.7000 3.8000

Columns 40 through 51

3.9000 4.0000 4.1000 4.2000 4.3000 4.4000 4.5000 4.6000 4.7000 4.8000 4.9000 5.0000

y2 =

Columns 1 through 13

3.0000 2.4600 2.0172 1.6541 1.3564 1.1122 0.9120 0.7479 0.6132 0.5029 0.4123 0.3381 0.2773

Columns 14 through 26

0.2274 0.1864 0.1529 0.1254 0.1028 0.0843 0.0691 0.0567 0.0465 0.0381 0.0312 0.0256 0.0210

Columns 27 through 39

0.0172 0.0141 0.0116 0.0095 0.0078 0.0064 0.0052 0.0043 0.0035 0.0029 0.0024 0.0019 0.0016

Columns 40 through 51

0.0013 0.0011 0.0009 0.0007 0.0006 0.0005 0.0004 0.0003 0.0003 0.0002 0.0002 0.0001

(c)

Mathematica Solution:

DSolve[y'[x] == -2\*y[x], y[x], x]

Out: {{y[x] -> e^(-2 x) C[1]}}

y[x\_] := Exp[-2\*x]\*C

Solve[y[0] == 3]

Out: {{C -> 3}}

y[x\_] := Exp[-2\*x]\*3

y[5]

Out: 3/e^10

N[3/e^10]

Out: 0.0001362

y(5) = 0.0001362

Euler’s Method gives a value of 0, which is an underestimate of the true value by 100%.

The improved version gives a value of 0.0001 which is an underestimate of the true value by 26.58%. The improved version did a better job at estimating the true value as expected.

(d)

>> xvec=0:0.1:5;

>> yvec = 3 .\* exp(-2.\*xvec);

>> plot(xvec,yvec,"blue",x1,y1,"green",x2,y2,"red")



Exercise 6)

(a)

function [x,y] = Eul\_6(h,x0,y0,xf)

k = 1;

x(k) = x0;

y(k) = y0;

while(x(k) + h <= xf + 10^-10)

x(k+1) = x(k) + h;

y(k+1) = y(k) + h\*((-9\*x(k)^2)/2\*y(k));

k = k + 1;

end

end

>> [x,y] = Eul\_6(0.5, 0, 7, 2)

x =

0 0.5000 1.0000 1.5000 2.0000

y =

7.0000 7.0000 6.9196 6.5945 5.8268

>> [x,y] = Eul\_6(0.25, 0, 7, 2)

x =

0 0.2500 0.5000 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000

y =

7.0000 7.0000 6.9900 6.9497 6.8587 6.6946 6.4321 6.0385 5.4680

>> [x1,y1] = Eul\_6(0.125, 0, 7, 2)

x1 =

Columns 1 through 13

0 0.1250 0.2500 0.3750 0.5000 0.6250 0.7500 0.8750 1.0000 1.1250 1.2500 1.3750 1.5000

Columns 14 through 17

1.6250 1.7500 1.8750 2.0000

y1 =

Columns 1 through 13

7.0000 7.0000 6.9987 6.9937 6.9824 6.9623 6.9307 6.8851 6.8225 6.7401 6.6344 6.5020 6.3384

Columns 14 through 17

6.1387 5.8968 5.6046 5.2518

function [x,y] = Eul\_improved\_6(h, x0, y0, xf)

k = 1;

x(k) = x0;

y(k) = y0;

while (x(k)+h<=xf+10^-10)

x(k+1) = x(k)+h;

y\_predict = y(k)+h\*((-9\*x(k)^2)/(2\*y(k)));

y(k+1) = y(k)+0.5\*h\*(((-9\*x(k)^2)/(2\*y(k))) + ((-9\*x(k+1)^2)/(2\*y\_predict)));

k = k+1;

end

>> [x,y] = Eul\_improved\_6(0.5, 0, 7, 2)

x =

0 0.5000 1.0000 1.5000 2.0000

y =

7.0000 6.9598 6.7559 6.1952 4.9499

>> [x,y] = Eul\_improved\_6(0.25, 0, 7, 2)

x =

0 0.2500 0.5000 0.7500 1.0000 1.2500 1.5000 1.7500 2.0000

y =

7.0000 6.9950 6.9698 6.9040 6.7756 6.5596 6.2244 5.7250 4.9849

>> [x2,y2] = Eul\_improved\_6(0.125, 0, 7, 2)

x2 =

Columns 1 through 13

0 0.1250 0.2500 0.3750 0.5000 0.6250 0.7500 0.8750 1.0000 1.1250 1.2500 1.3750 1.5000

Columns 14 through 17

1.6250 1.7500 1.8750 2.0000

y2 =

Columns 1 through 13

7.0000 6.9994 6.9962 6.9881 6.9723 6.9464 6.9077 6.8535 6.7806 6.6860 6.5660 6.4164 6.2323

Columns 14 through 17

6.0075 5.7344 5.4022 4.9958

Mathematica Solution:

DSolve[y'[x] == (-9\*x^2)/(2\*y[x]), y[x], x]

Out: {{y[x] -> -Sqrt[-3 x^3 + 2 C[1]]}, {y[x] -> Sqrt[-3 x^3 + 2 C[1]]}}

Taking the positive solution since y(0) = 7.

y[x\_] := Sqrt[-3 x^3 + 2 C]

Solve[y[0] == 7]

Out: {{C -> 49/2}}

y(x) = Sqrt(-3\*x^3 + 49)

>> xvec = 0:0.1:2;

>> yvec = sqrt(-3.\*xvec.\*xvec.\*xvec + 49);

plot(xvec,yvec,"blue",x1,y1,"green",x2,y2,"red")



Exercise 7)

>> [x,y] = ode45(@(x,y) x\*exp(3\*x)-2\*y,[0,2],0)

y(2) = 145.2357

>> [x,y] = ode45(@(x,y) -2\*y,[0,5],3)

y(5) = 0.0001

>> [x,y] = ode45(@(x,y) (-9\*x^2)/(2\*y),[0,2],7)

y(2) = 5.0000