**MA3457/CS4033** **B’14**

**FINAL EXAM ▼ PART 2**

Zach Arnold

1. (20 pts) The set of following data points is given:

***x* = [ 1 2 4 5 7 ];**

***y* = [ 52 5 -5 -40 10 ].**

Determine the 4th order polynomial in the Lagrange form that passes through the points. Use the obtained polynomial to determine the interpolated value for *x* = 3.

>> x=[ 1 2 4 5 7 ];

>> t=Lagrange\_coef(x,[52 5 -5 -40 10])

t =

0.7222 -0.1667 -0.2778 1.6667 0.0556

>> Lagrange\_Eval(3,x,t)

ans =

6.0000

As you can see here, the fourth-order polynomial is:

And the approximate value for

2. (25 pts) A tension test is conducted for determining the stress-strain behavior of rubber. The data points from the test are found to be:

***Strain * = [ 0 0.4 0.8 1.2 1.6 2.0 2.4 2.8 3.2 3.6 4.0 4.4 4.8 5.2 5.6 6.0 ];**

***Stress  (MPa)* = [ 0 3.0 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5 ].**

Find the “best fit” linear, quadratic, and cubic functions and show their numerical values for all 16 points. Determine the total squared errors for all approximations. Plot them on one graph along with the data points.

# MATLAB OUTPUT

# Linear:

>> Lin\_LS(e,s)

a =7.0393

b =-4.9493

err = 324.2502

# Quadratic:

>> Quad\_LS(e,s)

a = 1.3160

b = -0.8565

c = 2.4202

err = 71.0136

# Cubic:

>> Cubic\_LS(e,s)

a = -0.0541

b = 1.8030

c = -1.9882

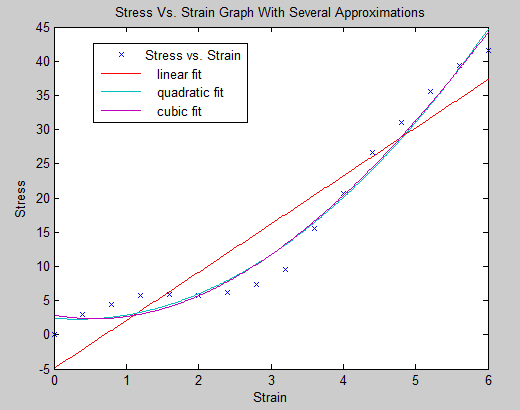
d = 2.8930

err = 69.9256

# Summary Table:

|  |  |  |
| --- | --- | --- |
| **Method** | **Polynomial** | **Error** |
| Linear (Lin\_LS) |  | err = 324.2502 |
| Quadratic (Quad\_LS) |  | err = 71.0136 |
| Cubic (Cubic\_LS) |  | err = 69.9256 |

# Graph:



3. (15 pts) Using the Composite Simpson’s Rule, find 4 decimal places of the exact values of the integral . Then evaluate it using 4th order Gaussian quadrature and find the error of this approximation.

# Exact Solution:

# MATLAB Output

The smallest n that gave the correct answer was .

>> Simpson(@(x)(exp(-x.^2)),0,3,8)

I =

0.8862

Using a 4th Order Gaussian Quadrature:

>> Gauss\_quad(@(x)(exp(-x.^2)),0,3,4)

t =

-0.5774 -0.7746 -0.8611 -0.9062

0.5774 0 -0.3400 -0.5385

0 0.7746 0.3400 0

0 0 0.8611 0.5385

0 0 0 0.9062

c =

1.0000 0.5556 0.3479 0.2369

1.0000 0.8889 0.6521 0.4786

0 0.5556 0.6521 0.5689

0 0 0.3479 0.4786

0 0 0 0.2369

I =

0.8841

ans =

0.8841

As you can see, the error of the Gaussian Quadrature is

This is a pretty good estimated integral value for the function.

4. (20 pts) Consider the initial value problem

*y*' = –1.2*y* + 7e-0.3*x*, *y*(0) = 3 on [0, 1.5].

First, use the Runge-Kutta method of order 4 with *h* = 0.5 to numerically solve this problem. Then apply the Taylor’s method of order 2. How many intervals are needed in this technique to get the same level of accuracy?

# MATLAB Output:

Using RK4:

>> [x,y]=RK4(@(x,y)(-1.2.\*y+7.\*exp(-0.3.\*x)),[0,1.5],3,3)

x =

0 0.5000 1.0000 1.5000

y =

3.0000 4.0698 4.3203 4.1676

Using Taylor2:

>> [x,y]=Taylor\_2(@(x,y)(-1.2.\*y+7.\*exp(-0.3.\*x)),@(x,y)(-1.2.\*(-1.2.\*y+7.\*exp(-0.3.\*x))-2.1.\*exp(-0.3.\*x)),[0,1.5],3,20)

x =

Columns 1 through 7

0 0.0750 0.1500 0.2250 0.3000 0.3750 0.4500

Columns 8 through 14

0.5250 0.6000 0.6750 0.7500 0.8250 0.9000 0.9750

Columns 15 through 21

1.0500 1.1250 1.2000 1.2750 1.3500 1.4250 1.5000

y =

Columns 1 through 7

3.0000 3.2376 3.4438 3.6215 3.7733 3.9018 4.0092

Columns 8 through 14

4.0975 4.1686 4.2242 4.2657 4.2947 4.3124 4.3200

Columns 15 through 21

4.3185 4.3089 4.2921 4.2689 4.2400 4.2060 4.1676

As you can see, it takes an n = 20 in order to produce the result gathered by RK4 for n = 3.

5. (20 pts) Solve the boundary value problem

, *y*(0) = 10, *y*(1) = *yb*

where *yb* = 2, 6, and 10, using the finite-difference method with *n* = 10. Present the approximate solutions numerically and by plotting them on one graph.

# How BVP\_FD.m Was Modified:

% No 3:

aa = 0; bb = 1; n = 10; h = (bb - aa)/n;

x = h:h:bb;

% Define p(x), q(x), r(x)

p = (-2)\*ones(1, n-1);

q = (4)\*ones(1, n-1);

r = -x.^2;

% Boundary conditions

ya = 10; yb = 10 or 6 or 2;

# MATLAB Output

>> BVP\_FD

0 10.0000

0.1000 7.4329

0.2000 5.6027

0.3000 4.3087

0.4000 3.4058

0.5000 2.7895

0.6000 2.3844

0.7000 2.1364

0.8000 2.0067

0.9000 1.9677

1.0000 2.0000

0 10.0000

0.1000 7.9136

0.2000 6.4942

0.3000 5.5687

0.4000 5.0131

0.5000 4.7394

0.6000 4.6855

0.7000 4.8085

0.8000 5.0796

0.9000 5.4803

1.0000 6.0000

0 10.0000

0.1000 8.3943

0.2000 7.3857

0.3000 6.8286

0.4000 6.6204

0.5000 6.6893

0.6000 6.9866

0.7000 7.4807

0.8000 8.1525

0.9000 8.9928

1.0000 10.0000

# Graph

