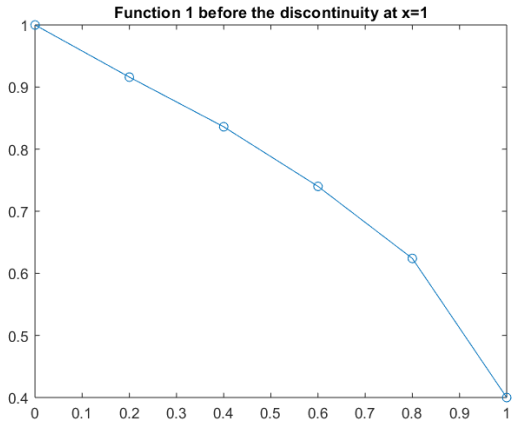
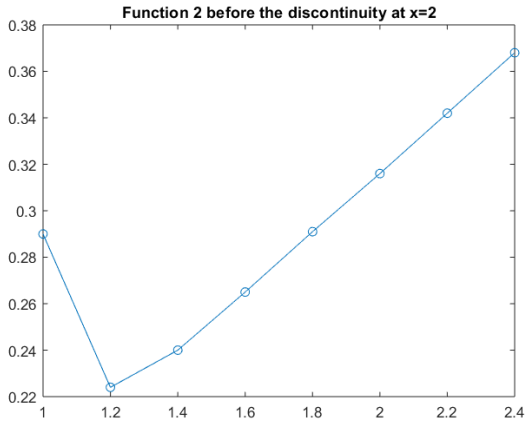
Bryan Jay
260738764

ECSE 443 - Assignment 3

***For the following questions, I used the data included in the assignment document. The data was treated as 2 different function to handle the discontinuity located at x = 1. Therefore, I put each equation into its own .txt files ('Ass_3_data_funct1', 'Ass_3_funct2').

Refer to Appendix F, for the Matlab code to generate the graphs.





Question 1

** For the following parts of I used the function 1 associated with the data points before x = 1.

a) Refer to Appendix A, part a and Appendix C, part a, for the corresponding Matlab code and the cubic splines equation that was written.

In order to find the $f(0.23)$ we will use all the points. Therefore, we can find the coefficients by constructing the following system of equations/matrices:

$$\begin{bmatrix} 1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & \\ & \ddots & \ddots & \ddots \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & 1 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{Bmatrix} = \begin{Bmatrix} 0 \\ 3(f[x_3,x_2]-f[x_2,x_1]) \\ \vdots \\ 3(f[x_n,x_{n-1}]-f[x_{n-1},x_{n-2}]) \\ 0 \end{Bmatrix}$$

Using the following functions, the cofficients for the polynomial were found.

$$d_i = \frac{c_{i+1}-c_1}{3h_i}$$

$$b_i = \frac{f_{i+1}-f_i}{h_i} - \frac{h_1}{3}(2c_i + c_{i+1})$$

$\boldsymbol{a_1}$	0.916
$\boldsymbol{b_1}$	-0.396459330143541
$\boldsymbol{c_1}$	0.176555023923443
$\boldsymbol{d_1}$	-0.971291866028701

With the calculate variables we find the following piecewise function:

$$\texttt{f} = \frac{369\left(t_c-\frac{1}{5}\right)^2}{2090}-\frac{4143\,t_c}{10450}-\frac{203\left(t_c-\frac{1}{5}\right)^3}{209}+\frac{26002}{26125}$$

To find $f(0.23)$ we will us the case where $x \in [0.2, 0.4]$ and we calculate the following.

$$\boldsymbol{f(0.23) = 0.904238894736842}$$

Using the built in spline method in Matlab the answer was found to be similar.

$$\boldsymbol{f(0.23) = 0.904474866666667}$$

In other cases, it is better to use the cubic spline along all points rather than just 3 points to get a better value. This is due to the fact that there would be more points allowing for a better fit of the function which would lead to a closer more precise answer. Refer to Appendix E, part A, I demonstrated the difference between the case of 3 points and all the points so the answers can be compared. Using all the pts provides a closer approximate to the Newton's and Lagrange methods.

b) Refer to Appendix A, part b and Appendix C, part b, for the corresponding Matlab code since there is a function in Appendix B to the function used. To use Lagrange Polynomial interpolation the following matrix/system of equations is constructed and solved for the unknowns:

$$Z = \begin{bmatrix} \prod_{k=1,k\neq1}^{n}\frac{x_1-x_k}{x_1-x_k} & \prod_{k=1,k\neq2}^{n}\frac{x_1-x_k}{x_2-x_k} & \cdots & \prod_{k=1,k\neq n}^{n}\frac{x_1-x_k}{x_n-x_k} \\ \prod_{k=1,k\neq1}^{n}\frac{x_2-x_k}{x_1-x_k} & \prod_{k=1,k\neq2}^{n}\frac{x_2-x_k}{x_2-x_k} & \cdots & \prod_{k=1,k\neq n}^{n}\frac{x_2-x_k}{x_n-x_k} \\ \prod_{k=1,k\neq1}^{n}\frac{x_3-x_k}{x_1-x_k} & \prod_{k=1,k\neq2}^{n}\frac{x_3-x_k}{x_2-x_k} & \cdots & \prod_{k=1,k\neq n}^{n}\frac{x_3-x_k}{x_n-x_k} \\ \vdots & \vdots & \ddots & \vdots \\ \prod_{k=1,k\neq1}^{n}\frac{x_n-x_k}{x_1-x_k} & \prod_{k=1,k\neq2}^{n}\frac{x_n-x_k}{x_2-x_k} & \cdots & \prod_{k=1,k\neq n}^{n}\frac{x_n-x_k}{x_n-x_k} \end{bmatrix}$$

The Lagrange Polynomials and the interpolating polynomial were found to be:

$$\texttt{L\_coeffs} = \begin{pmatrix} -\frac{125\,(5x_l-1)\,(x_l-1)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{24} \\ \frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{24} \\ -\frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{12} \\ \frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{4}{5}\right)}{12} \\ -\frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)}{24} \\ \frac{625\,x_l\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{24} \end{pmatrix}$$

$$f = -\frac{125\,x_l^5}{48} + \frac{45\,x_l^4}{8} - \frac{73\,x_l^3}{16} + \frac{61\,x_l^2}{40} - \frac{7\,x_l}{12} + 1$$

Then, we plug the point 0.23 to find $f(0.23)$.

$$\boldsymbol{f(0.23) = 0.90505882109375}$$

c) Refer to Appendix A, part c and Appendix C, part c, for the corresponding Matlab code since there is a function in Appendix B to the function used. To use Newton's Polynomial interpolation the following matrix/system of equations is constructed and solved for the unknowns:

$$\begin{bmatrix} 1 & & \cdots & & 0 \\ 1 & x_1 - x_0 & & & \\ 1 & x_2 - x_0 & (x_2 - x_0)(x_2 - x_1) & & \vdots \\ \vdots & \vdots & & \ddots & \\ 1 & x_k - x_0 & \cdots & \cdots & \prod_{j=0}^{k-1}(x_k - x_j) \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_k \end{bmatrix}$$

The matrices are calculated to be the following:

```
x_mat = 6×6
    1.0000         0         0         0         0         0
    1.0000    0.2000         0         0         0         0
    1.0000    0.4000    0.0800         0         0         0
    1.0000    0.6000    0.2400    0.0480         0         0
    1.0000    0.8000    0.4800    0.1920    0.0384         0
    1.0000    1.0000    0.8000    0.4800    0.1920    0.0384

y_mat = 6×1
    1.0000
    0.9160
    0.8360
    0.7400
    0.6240
    0.4000
```

We solve this matrix to find the unknowns.

$\boldsymbol{a_1}$	1.0000
$\boldsymbol{a_2}$	-0.4200
$\boldsymbol{a_3}$	0.5000
$\boldsymbol{a_4}$	-0.4167
$\boldsymbol{a_5}$	0.4167
$\boldsymbol{a_6}$	-2.6042

So, the corresponding function to the unknowns is found and then we input $f(0.23)$:

f(t)= $0.05\,t\,(t-0.2) - 0.42\,t - 0.41667\,t\,(t-0.2)\,(t-0.4) + 0.41667\,t\,(t-0.6)\,(t-0.2)\,(t-0.4) - 2.6042\,t\,(t-0.6)\,(t-0.2)\,(t-0.4)\,(t-0.8) + 1.0$

Simplified function:

$$f\_N(t) = -\frac{125\,t^5}{48} + \frac{303992974847507863\,t^4}{54043195528445952} - \frac{205476732998778263\,t^3}{45035996273704960} + \frac{2060396829521991593\,t^2}{1351079888211148800} - \frac{1970324836974588379\,t}{3377699720527872000} + 1$$

$$\boldsymbol{f(0.23) = 0.905058821093750}$$

The results of the Newton's Polynomial and Lagrange Polynomial are both exactly the same because they work up to the 5th order polynomial which looks much greater than order of the function 1. Therefore, the use of these interpolation methods is overkill on the type of method as they are more complex then our given function points. That is why the functions are able to produce the same results because of their higher level of complexity.

Question 2

** For the following parts of I used the function 1 associated with the data points before x = 1.

a) Refer to Appendix B, part d and Appendix C, part a, for the corresponding Matlab code and the cubic splines equation that was written.

In order to find the $f(0.23)$ we will all the points. Therefore, as we saw in question 1, a, we followed the same procedure as in question 1 to calculate the polynomial. Using the following functions, the cofficients for the polynomial were found.

$$d_i = \frac{c_{i+1} - c_1}{3h_i}$$
$$b_i = \frac{f_{i+1} - f_i}{h_i} - \frac{h_1}{3}(2c_i + c_{i+1})$$

$\boldsymbol{a_3}$	0.74
$\boldsymbol{b_3}$	-0.473971291866028
$\boldsymbol{c_3}$	0.248325358851676
$\boldsymbol{d_3}$	-3.892344497607658

With the calculate variables we find the following piecewise function:

$$f = \frac{519\left(t_c - \frac{3}{5}\right)^2}{2090} - \frac{4953\,t_c}{10450} - \frac{1627\left(t_c - \frac{3}{5}\right)^3}{418} + \frac{26762}{26125}$$

$$\boldsymbol{f(0.78) = 0.640030755980861}$$

Using the built in spline method in Matlab the answer was found to be similar.

$$\boldsymbol{f(0.78) = 0.638620800000000}$$

In other cases, it is better to use the cubic spline along all points rather than just 3 points to get a better value. This is due to the fact that there would be more points allowing for a better fit of the function which would lead to a closer more precise answer. Refer to Appendix E, part 2, for

the results of the cubic splines done using 3 points instead of all the points, and it is clear that using all the points provides a better function.

b) Refer to Appendix B, part b and Appendix C, part b, for the corresponding Matlab code since there is a function in Appendix B to the function used. To calculate the Lagrange Polynomial, we followed the same procedure as in question 1, b.
The Lagrange Polynomials and the interpolating polynomial were found to be:

L_coeffs =

$$\begin{pmatrix} -\frac{125\,(5x_l-1)\,(x_l-1)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{24} \\ \frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{24} \\ -\frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{12} \\ \frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{4}{5}\right)}{12} \\ -\frac{3125\,x_l\,(x_l-1)\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)}{24} \\ \frac{625\,x_l\left(x_l-\frac{1}{5}\right)\left(x_l-\frac{2}{5}\right)\left(x_l-\frac{3}{5}\right)\left(x_l-\frac{4}{5}\right)}{24} \end{pmatrix}$$

f =

$$-\frac{125\,x_l^5}{48}+\frac{45\,x_l^4}{8}-\frac{73\,x_l^3}{16}+\frac{61\,x_l^2}{40}-\frac{7\,x_l}{12}+1$$

Then, we plug the point 0.78 to find $f(0.78)$.

$$\boldsymbol{f(0.78) = 0.637895075}$$

c) Refer to Appendix B, part c and Appendix C, part c, for the corresponding Matlab code since there is a function in Appendix B to the function used. To calculate Newton's Polynomial interpolation, we followed the same procedure as question 1, c.

The matrices are calculated to be the following:

```
x_mat = 6×6
    1.0000         0         0         0         0         0
    1.0000    0.2000         0         0         0         0
    1.0000    0.4000    0.0800         0         0         0
    1.0000    0.6000    0.2400    0.0480         0         0
    1.0000    0.8000    0.4800    0.1920    0.0384         0
    1.0000    1.0000    0.8000    0.4800    0.1920    0.0384

y_mat = 6×1
    1.0000
    0.9160
    0.8360
    0.7400
    0.6240
    0.4000
```

We solve this matrix to find the unknowns.

a_1	1.0000
a_2	-0.4200
a_3	0.5000
a_4	-0.4167
a_5	0.4167
a_6	-2.6042

So, the corresponding function to the unknowns is found and then we input $f(0.78)$:

$$f\_N(t) = -\frac{125\,t^5}{48} + \frac{303992974847507863\,t^4}{54043195528445952} - \frac{205476732998778263\,t^3}{45035996273704960} + \frac{2060396829521991593\,t^2}{1351079888211148800} - \frac{1970324836974588379\,t}{3377699720527872000} + 1$$

$$\boldsymbol{f(0.78) = 0.637895074999998}$$

The results of the Newton's Polynomial and Lagrange Polynomial are both the same because they work up to the 5th order polynomial which looks much greater than order of the function 2 (which seems linear). Therefore, the use of these interpolation methods is overkill on the type of method as they are more complex then our given function points. That is why the functions are able to produce the same results because of their higher level of complexity.

Question 3

** For the following question, I used the function 2 associated with the data points after x = 1.

Refer to the Appendix D and Appendix C, part A for the Matlab code and the function used to calculate the cubic spline. Similarly, to what was done in Question 1a and 2a, we used the cubic spline method on the set of three points to get a function. However, since we are extrapolating a point not on the curve, we are using a the last 3 given points of function 2.

a_1	0.916
b_1	-0.396459330143541
c_1	0.176555023923443
d_1	-0.971291866028701

In order to find the $f(3.0)$ we all the points. With the calculated variables we find the following function:

$$f = \frac{4724091904558005\,t_c}{36028797018963968} - \frac{2420897525561351\left(t_c - \frac{11}{5}\right)^2}{288230376151711744} + \frac{8069658418537847\left(t_c - \frac{11}{5}\right)^3}{5764607523034234888} + \frac{241105798807258257}{4503599627370496000}$$

To find $f(3.0)$ we will us the case where $x \in [2.4, \infty)$ and we calculate the following.

$$\boldsymbol{f(3.0) = 0.448687736173136}$$

The answer found for this case was used with all points. However, I compared it with the method of finding 3 points which can be found in Appendix E, part C. We found that the use of all points is a closer

approximation to the extrapolated value but using 3 points requires less computation and gives a fairly accurate result.

Appendix

Appendix A – Question 1 Matlab Code

```
spline(x_cs, y_cs, x_unkn)
```

```
ans =
   0.904474866666667
```

part b

```
% use the same data as above
% transfer the data to its x,y coordinates
x = data(:,1);
y = data(:,2);
L_coeffs = sym(ones(length(x), 1));
syms x_l
% call the Lagrange polynomial function to get the function polynomial
f = LagrangePolynomial(x_l, x, y)
```

```
L_coeffs =
```

$$\begin{pmatrix} -\frac{125\,(5x_l-1)\,(x_l-1)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{24} \\ \frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{24} \\ -\frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{12} \\ \frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{12} \\ -\frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)}{24} \\ \frac{625\,x_l\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{24} \end{pmatrix}$$

f =

$$-\frac{125\,x_l^5}{48}+\frac{45\,x_l^4}{8}-\frac{73\,x_l^3}{16}+\frac{61\,x_l^2}{40}-\frac{7\,x_l}{12}+1$$

```
vpa(subs(f, x_l, x_unkn))
```

ans = 0.90505882109375

part c - Newton's Polynomial

```
% use the same data as above
% transfer the data to its x,y coordinates
% use the same data as above
% transfer the data to its x,y coordinates
x = data(:,1);
y = data(:,2);
% call the newtons polynomial funct
vars = NewtonsPolynomial(x, y)
```

```
vars = 6×1
   1.000000000000000
  -0.420000000000000
   0.049999999999997
  -0.416666666666658
   0.416666666666655
  -2.604166666666694
```

```
% plug points into the corresponding function
f_N(t) = vars(1) + vars(2)*(t-x(1)) + vars(3)*(t-x(1))*(t-x(2)) + vars(4)*(t-
x(1))*(t-x(2))*(t-x(3)) + vars(5)*(t-x(1))*(t-x(2))*(t-x(3))*(t-x(4)) +
vars(6)*(t-x(1))*(t-x(2))*(t-x(3))*(t-x(4))*(t-x(5));
vpa(f_N(t), 5)
```

```
ans
=
```

$$0.05\,t\,(t-0.2)-0.42\,t-0.41667\,t\,(t-0.2)\,(t-0.4)+0.41667\,t\,(t-0.6)\,(t-0.2)\,(t-0.4)-2.6042\,t\,(t-0.6)\,(t-0.2)\,(t-0.4)\,(t-0.8)+1.0$$

```
f_N(t) = simplify(f_N(t))
f_N(t) =
```

$$-\frac{125\,t^5}{48}+\frac{303992974847507863\,t^4}{54043195528445952}-\frac{205476732998778263\,t^3}{45035996273704960}+\frac{2060396829521991593\,t^2}{1351079888211148800}-\frac{1970324836974588379\,t}{3377699720527872000}+1$$

```
double(f_N(x_unkn))
ans =
   0.905058821093750
```

Appendix B – Question 2 Matlab Code

Question 2 - part a - Cubic Spline Interpolation

```
format long
syms f_N(t)
% import the data
data = importdata(['Ass_3_data_funct1.txt']);
% calculate the unknown point 0.78
x_unkn = 0.78;
% data points that we will use for the splines
x_cs = data(:,1);
y_cs = data(:,2);
% calculate the cubic spline and our unknown point
f = CubicSplineInter(x_cs, y_cs, x_unkn);
c = 6×1
                   0
   0.176555023923443
  -0.406220095693778
   0.248325358851676
  -2.087081339712920
                   0
d = 5×1
   0.294258373205738
  -0.971291866028701
   1.090909090909090
  -3.892344497607658
   3.478468899521534
b = 5×1
  -0.431770334928229
  -0.396459330143541
  -0.442392344497608
  -0.473971291866028
  -0.841722488038277
f =
```

$$\frac{519\left(t_c-\frac{3}{5}\right)^2}{2090}-\frac{4953\,t_c}{10450}-\frac{1627\left(t_c-\frac{3}{5}\right)^3}{418}+\frac{26762}{26125}$$

```
f = double(f)
```

```
f =
   0.640030755980861
```

```
% compare with the built in spline equation
spline(x_cs, y_cs, x_unkn)
```

```
ans =
   0.638620800000000
```

part b

```
% use the same data as above
% transfer the data to its x,y coordinates
x = data(:,1);
y = data(:,2);
L_coeffs = sym(ones(length(x), 1));
syms x_l
% call the Lagrange polynomial function to get the function polynomial
f = LagrangePolynomial(x_l, x, y)
```

```
L_coeffs =
```

$$\begin{pmatrix} -\frac{125\,(5\,x_l-1)\,(x_l-1)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{24} \\ \frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{24} \\ -\frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{12} \\ \frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{12} \\ -\frac{3125\,x_l\,(x_l-1)\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)}{24} \\ \frac{625\,x_l\,\left(x_l-\frac{1}{5}\right)\,\left(x_l-\frac{2}{5}\right)\,\left(x_l-\frac{3}{5}\right)\,\left(x_l-\frac{4}{5}\right)}{24} \end{pmatrix}$$

```
f =
```

$$-\frac{125\,x_l^5}{48}+\frac{45\,x_l^4}{8}-\frac{73\,x_l^3}{16}+\frac{61\,x_l^2}{40}-\frac{7\,x_l}{12}+1$$

```
vpa(subs(f, x_l, x_unkn))
```

```
ans = 0.637895075
```

part c - Newton's Polynomial

```
% use the same data as above
% transfer the data to its x,y coordinates
% use the same data as above
% transfer the data to its x,y coordinates
```

```
x = data(:,1);
y = data(:,2);
% call the newtons polynomial funct
vars = NewtonsPolynomial(x, y)
```

```
vars = 6×1
   1.000000000000000
  -0.420000000000000
   0.049999999999997
  -0.416666666666658
   0.416666666666655
  -2.604166666666694
```

```
% plug points into the corresponding function
f_N(t) = vars(1) + vars(2)*(t-x(1)) + vars(3)*(t-x(1))*(t-x(2)) + vars(4)*(t-
x(1))*(t-x(2))*(t-x(3)) + vars(5)*(t-x(1))*(t-x(2))*(t-x(3))*(t-x(4)) +
vars(6)*(t-x(1))*(t-x(2))*(t-x(3))*(t-x(4))*(t-x(5));
vpa(f_N(t), 5)
```

```
ans
=
```

$$0.05\,t\,(t-0.2)-0.42\,t-0.41667\,t\,(t-0.2)\,(t-0.4)+0.41667\,t\,(t-0.6)\,(t-0.2)\,(t-0.4)-2.6042\,t\,(t-0.6)\,(t-0.2)\,(t-0.4)\,(t-0.8)+1.0$$

```
f_N(t) = simplify(f_N(t))
```

```
f_N(t) =
```

$$-\frac{125\,t^5}{48}+\frac{303992974847507863\,t^4}{54043195528445952}-\frac{205476732998778263\,t^3}{45035996273704960}+\frac{2060396829521991593\,t^2}{1351079888211148800}-\frac{1970324836974588379\,t}{3377699720527872000}+1$$

```
double(f_N(x_unkn))
```

```
ans =
   0.637895074999998
```

Appendix C – Question 1 & 2 Matlab Functions

PART A – Cubic Splines Interpolation Function

Function to Calculate the Cubic Spline Interpolation.

inputs: t_c: symbolic variable that will be used

x: x points (3-pts)

y: y points (3-pts)

pt: the desired point

return: polynomial function

```
function val = CubicSplineInter(x, y, pt)
len = length(y);
% find all the h step values
h = zeros(len-1, 1);
for i=1:length(h)
    h(i) = x(i+1)-x(i);
end
f_int = zeros(len-1, 1);
```

```
for i=1:length(f_int)
    f_int(i) = ((y(i+1)-y(i))/h(i));
end
% create the A and B matrix for the system of equations
A = zeros(len);
B = zeros(len,1);
% set the two conditions to 1
A(1,1) = 1;
A(len,len) = 1;
% fill the A and B matrices
for i=2:len-1
    A(i,i-1) = h(i-1);
    A(i,i) = 2*(h(i-1)+h(i));
    A(i,i+1) = h(i);

    B(i) = 3*(f_int(i)-f_int(i-1));
end
% calculate the unknown c variable
c = inv(A)*B
% initialize the b,d variables and calculate them with the given equations
b = zeros(len-1, 1);
d = zeros(len-1,1);
for i=1:len-1
    b(i) = (y(i+1)-y(i))/h(i) - h(i)*(2*c(i) + c(i+1))/3;
    d(i) = (c(i+1)-c(i))/(3*h(i));
end
d
b
syms t_c
% determine what index to use for the function
if pt < min(x)
    % if the points is before min use the first point
    index = 1;
elseif pt > max(x)
    % if the point is after min use the second to last point for function
    index = len-1;
else
    % Else use  the coordinate right below our pt
    temp = find(pt>x);
    index = max(temp);
end
f = y(index) + b(index)*(t_c-x(index)) + c(index)*(t_c-x(index))^2 +
d(index)*(t_c-x(index))^3
val = subs(f,t_c,pt);
end
```

PART B – Lagrange Polynomial Function

Function to Calculate the Lagrange Polynomial Interpolation.

inputs: x_l: symbolic variable that will be used

x: x points

y: y points

return: polynomial function

```
function f = LagrangePolynomial(x_l, x, y)
L_coeffs = sym(ones(length(x),1));
syms x_l
% find all the lagrange polynomials associated with the data
for i=1:length(x)
    temp = 1;
    for j=1:length(x)
        if j ~= i
            temp = temp * (x_l - x(j))/(x(i)-x(j));
        end
    end
    L_coeffs(i) = temp;
end
L_coeffs
f = 0;
for i=1:length(x)
    temp = y(i)*L_coeffs(i);
    f = f + temp;
end
f = simplify(f);
end
```

PART C – Newton Polynomial Function

Function to Calculate the Newton's Polynomial Interpolation.

inputs: x: x points

y: y points

return: coefficients of the newtons polynomial

```
function f = NewtonsPolynomial(x, y)
% initialize matrix of the data size
x_mat = zeros(length(x));
y_mat = zeros(length(x),1);
sz = length(x_mat);
for i=1:sz
    x_mat(i, 1) = 1;
    y_mat(i) = y(i);
    for j=2:sz
        if i >= j
            temp = 1;
            % find the array points val
```

```
            for k=1:j-1
                temp = temp * (x(i)-x(k));
            end
            x_mat(i,j) = temp;
        end
    end
end
% calculate the unknowns
f = inv(x_mat)*y_mat;
end
```

Appendix D – Question 3 Matlab Code

Question 3 - Cubic Spline extrapolation

```
format long
% import the data
data = importdata('Ass_3_data_funct2.txt');
x_unkn = 3.0;
% split the data to get the desired points
x = data(:,1);
y = data(:,2);
% call the cubic spline function and get the function correspoing to the
% second interval
f = CubicSplineInter(x,y,x_unkn);
```

```
c = 8×1
                   0
   1.601356922019911
  -0.255427688079691
   0.095353830298861
  -0.050987633115765
   0.033596702164203
  -0.008399175541051
                   0
d = 7×1
   2.668928203366505
  -3.094641016832671
   0.584635863964255
  -0.243902439024376
   0.140973892133281
  -0.069993129508757
   0.013998625901751
b = 7×1
  -0.436757128134659
  -0.116485743730675
   0.152700103057369
   0.120685331501203
   0.129558570937822
```

```
   0.126080384747510
   0.131119890072140
f =
```

$$\frac{4724091904558005\,t_c}{36028797018963968}-\frac{2420897525561351\left(t_c-\frac{11}{5}\right)^2}{288230376151711744}+\frac{8069658418537847\left(t_c-\frac{11}{5}\right)^3}{576460752303423488}+\frac{241105798807258257}{4503599627370496000}$$

```
f = double(f)
f =
   0.448687736173136
```

Appendix E – Alternative Spline method for 3 points, Results and Matlab code

PART A – Question 1, a – Using 3 pts for spline

With the calculate variables we find the following piecewise function:

$$f(t)=\begin{cases}1-0.425t+0.125t^3, & x\in[0,0.2]\\ 1.002-0.455t+0.15t^2-0.125t^3, & x\in[0.2,0.4]\end{cases}$$

To find $f(0.23)$ we will us the case where $x\in[0.2,0.4]$ and we calculate the following.

$$\boldsymbol{f(0.23)=0.903764125000000}$$

Using the built in spline method in Matlab the answer was found to be similar.

$$\boldsymbol{f(0.23)=0.903745000000000}$$

PART B – Question 2, a – Using 3 pts for the cubic spline method

$\boldsymbol{a_0}$	0.74
$\boldsymbol{b_0}$	-0.445
$\boldsymbol{c_0}$	0
$\boldsymbol{d_0}$	-3.375
$\boldsymbol{a_1}$	0.624
$\boldsymbol{b_1}$	-0.85
$\boldsymbol{c_1}$	-2.025
$\boldsymbol{d_1}$	3.375

With the calculate variables we find the following piecewise function:

$$f(t)=\begin{cases}1.736-0.409t+6.075t^2-3.375t^3, & x\in[0.6,0.8)\\ -1.72+8.87t-10.125t^2+3.375t^3, & x\in[0.8,1.0)\end{cases}$$

$$\boldsymbol{f(0.78)=0.640217000000000}$$

Using the built in spline method in Matlab the answer was found to be similar.

$$\boldsymbol{f(0.78)=0.640460000000000}$$

PART C – Question 3, extrapolating the function using only 3 points as minimum.

Since we are extrapolating a point not on the curve, we are using a the last 3 given points of function 2. In order to find the $f(3.0)$ we will use the points [2,0.316], [2.2,0.342], [2.4, 0.368]. The matrices were found to be:

```
A = 3×3
   1.000000000000000                   0                   0
   0.200000000000000   0.800000000000000   0.200000000000000
                   0                   0   1.000000000000000

B = 3×1
                   0
   0.059999999999998
                   0
```

With the calculate variables we find the following function:

$$f = 0.13\,t_c + 0.056$$

To find $f(3.0)$ we will us the case where $x \in [2.4, \infty)$ and we calculate the following.

$$\boldsymbol{f(3.0) = 0.446}$$

Appendix F – Matlab code for generating graphs

```
funct1 = importdata('Ass_3_data_funct1.txt');
funct2 = importdata('Ass_3_data_funct2.txt');
plot(funct1(:,1), funct1(:,2), '-o')
title('Function 1 before the discontinuity at x=1')

plot(funct2(:,1), funct2(:,2),'-o')
title('Function 2 before the discontinuity at x=2')
```