MP 1 Docu

Allan Matthew B. Mariano || 2015 - 05804 || CS 131 WFY || 4/2/18

Curve Fitting

I. Introduction

Given the data from the specifications:

z (m)	0	2.3	4.9	9.1	13.7	18.3	22.9	27.2
T (^{o}C)	22.8	22.8	22.8	20.6	13.9	11.7	11.1	11.1

The specifications asks to find a curve that positions itself to the data using linear least squares regression. With that, we can determine the thermocline or the inflection point, the gradient and the heat flux. With those values, we can already plot the curve.

1.1 Finding the Cubic Polynomial

Given the data from the specifications:

z (m)	0	2.3	4.9	9.1	13.7	18.3	22.9	27.2
T (^{o}C)	22.8	22.8	22.8	20.6	13.9	11.7	11.1	11.1

The specifications states that linear least squares regression should be used in order to determine the cubic polynomial.

Mathematically, what I'm doing can be expressed as,

$$A\vec{x} = \vec{b}$$

With matrices given:

$$x = [0 \quad 2.3 \quad 4.9 \quad 9.1 \quad 13.7 \quad 18.3 \quad 22.9 \quad 27.2]$$

$$temp = [22.8 \quad 22.8 \quad 22.8 \quad 20.6 \quad 13.9 \quad 11.7 \quad 11.1 \quad 11.1]$$

We define a matrix such that it represents the values of x as a polynomial

We plug-in the values of x inside this polynomial

$$\text{Polynomial} = \begin{bmatrix} 1 & x & x\text{^}2 & x\text{^}3 \\ 1 & x1 & x1\text{^}2 & x1\text{^}3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x8 & x8\text{^}2 & x8\text{^}3 \end{bmatrix}$$

In code, I construct the polynomial matrix by making concatenationg 4 different matrices the first one being a 8 x1 matrix filled with 1s and concatenating it with x then concatenating it with y which holds the value of x^2 concatenating it with z which holds the value of x^3. The matrices y and z are also 8x1.

```
# x values z(m)
x =[
  0
  2.3
  4.9
  9.1
  13.7
  18.3
  22.9
  27.2
];
#Summer Temperature Values T(C)
temp = [
  22.8
  22.8
  22.8
  20.6
  13.9
  11.7
  11.1
  11.1
];
```

Figure 1: How my values look in code

```
#convert to vandermonde matrix first
y = x.*x;
z = x.*x.*x;

lmao = horzcat(onematrix,x);
lmao2 = horzcat(lmao,y);
polynomial = horzcat (lmao2,z);
```

Figure 2: How I make the polynomial matrix

```
polynomial =

  1.0000e+000   0.0000e+000   0.0000e+000   0.0000e+000
  1.0000e+000   2.3000e+000   5.2900e+000   1.2167e+001
  1.0000e+000   4.9000e+000   2.4010e+001   1.1765e+002
  1.0000e+000   9.1000e+000   8.2810e+001   7.5357e+002
  1.0000e+000   1.3700e+001   1.8769e+002   2.5714e+003
  1.0000e+000   1.8300e+001   3.3489e+002   6.1285e+003
  1.0000e+000   2.2900e+001   5.2441e+002   1.2009e+004
  1.0000e+000   2.7200e+001   7.3984e+002   2.0124e+004
```

Figure 3: Output of the Polynomial

Then we let A be polynomial and temp be b:

$$polynoial * x = temp$$

Solve for x which is a 4x1 matrix that holds the coefficients of the polynomial. We solve it by applying Cholesky.

1.2 **Cholesky Factorization**

Before we apply the our matrices to Cholesky LDL^T, we must first multiply A^T to both sides of the equation making it 4x4 matrix so that the factorization would work.

In my program, I made a function which computes the factorization and the converting to 4x4 matrix. The output of the program is x which is a 4x1 matrix that outputs the coefficients of the polynomial.

```
function ret = cholesky(Tot,p)
  A = Tot'*Tot;
  b = Tot'*p;
  I  = eye(columns(A));
  s = columns(I);



  D = A;
  L = I;

  for j = 1:s-1
     Ml = I;
    for k=j+1:s
       Ml(k,j) = -D(k,j)/D(j,j);
   end
    L = Ml*L;
    D = Ml*D;
  end


  D= diag(diag(D));

  L = inv(L);

  y = L\b;
  x = (D*L')\y;

  ret = x;
```

```
coefficients = cholesky(polynomial,temp)
```

Figure 4: My Cholesky Factorization function in Octave

```
================================================
1.2 = The coeficients from cholesky LDLT
coefficients =

  2.2876e+001
  3.2771e-001
  -1.0003e-001
  2.6727e-003

a0 = 22.876441
a1 = 0.327707
a2 = -0.100031
a3 = 0.002673
================================================
```

Figure 5: Output of the Cholesky LDT^T

Mathematically, coefficients can be defined as:

$$coefficients = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

The coefficients will be applied to the function such as

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

Getting the derivative and second derivative will result into

$$f'(x) = a_1 + 2a_2 x + 3a_3 x^2$$

$$f''(x) = 2a_2 + 6a_3 x$$

The first derivative is used for getting the value/s of the gradient while the second derivative is used to get the values of the thermocline.

In code, I have a function for the first derivative:

```
#Gradient (First Derivative of f(x))
function ret = gradient(x, ans)
  ret = (3*ans(4)*x*x)+ (2*ans(3)*x) + ans(2);
endfunction
```

Figure 6: Function of the Gradient

1.3 Getting the Thermocline

The thermocline is the point where maximum absolute value of the gradient and is also the point of inflection in the curve. Being an inflection point means that its second derivative is equal to 0.

Thus the equation for the thermocline would be:

$$0 = 2a_2 + 6a_3\, x$$

To solve for x, we simply isolate x since it is a polynomial of only degree 1.

$$x = -\frac{2a_2}{6a_3}$$

In my program, I saved in the variable called thermocline

```
thermocline = (-2*coefficients(3))/(6*coefficients(4));
```

Figure 7: Code for the Thermocline

```
================================================
1.3 = Get the inflection point
Thermocline = 12.475537 m
================================================
```

Figure 8: Output of the Thermocline

1.4 Getting The Heat Flux

The formula for the heat flux is:

$$J = -\alpha\rho C\frac{dT}{dz}$$

Where rho is the density of the water (1 g/cm^3) and C being the specific heat (1 cal/(g.C)). We assume alpha, which is the eddy diffusion coefficient, to be equal to 0.01cm^2/s.

The heat flux by a simple equation of

$$J = -0.01 \left[\frac{cm^2}{s}\right] * 1 \left[\frac{g}{cm^3}\right] * 1 \left[\frac{cal}{g \cdot C}\right] * \frac{dT}{dz} \left[\frac{C}{m}\right]$$

However, the answer is required to be in calories per square centimeter per day. In order to achieve this we make the following conversions:

$$J = -0.01 \left[\frac{cm^2}{s}\right] * \left[\frac{60s}{min}\right] * \left[\frac{60min}{hr}\right] * \left[\frac{24hrs}{day}\right] * 1 \left[\frac{g}{cm^3}\right] * 1 \left[\frac{cal}{g \cdot C}\right] * \frac{dT}{dz} \left[\frac{C}{m}\right] * \left[\frac{1m}{100cm}\right]$$

In the end it can be simple put to be:

$$J = -0.01 \frac{dT}{dz} * \frac{84000}{100}$$

Which its unit being cal /(cm^2*day). The gradient which dT/dz is the gradient on the thermocline which is f'(thermocline) = f'(12.476) = -0.920234 .

So all in all the whole equation would be

$$J \; or \; heatflux = -0.01(-0.920234) * \frac{84000}{100}$$

Which gets us the value 7.950821 cal/(cm^2 day)

```
printf("1.5 = Get the gradient(First Derivative)\n")
gradient1 = gradient(thermocline, coefficients);

printf("Gradient = %f C/m \n", gradient1)

printf("================================================\n")

#get heatflux

printf("1.7 = Get the heatflux using the gradient\n")

heatflux = ((-0.01*(gradient1/100))*86400);

printf("Heat flux = %f cal / (cm^2 day)\n", heatflux)
```

Figure 9 The code used to get the

```
================================================
1.5 = Get the gradient(First Derivative)
Gradient = -0.920234 C/m
================================================
1.7 = Get the heatflux using the gradient
Heat flux = 7.950821 cal / (cm^2 day)
>>
```

Figure 10: The output of the gradient and heat flux in Octave

1.5 Plotting

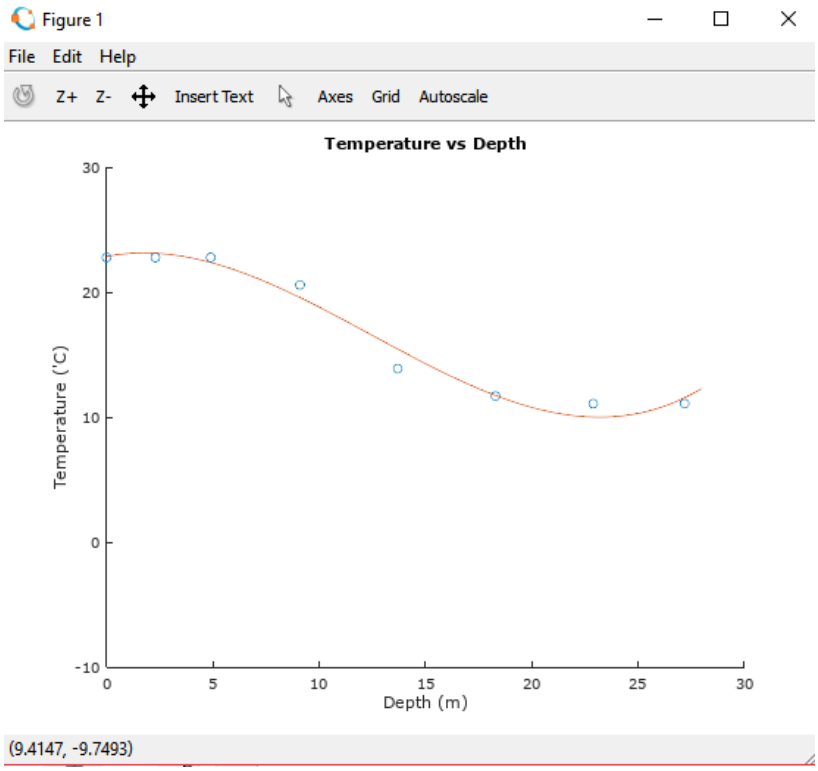


Figure 11: The Temperature vs Depth graph for Curve Fitting

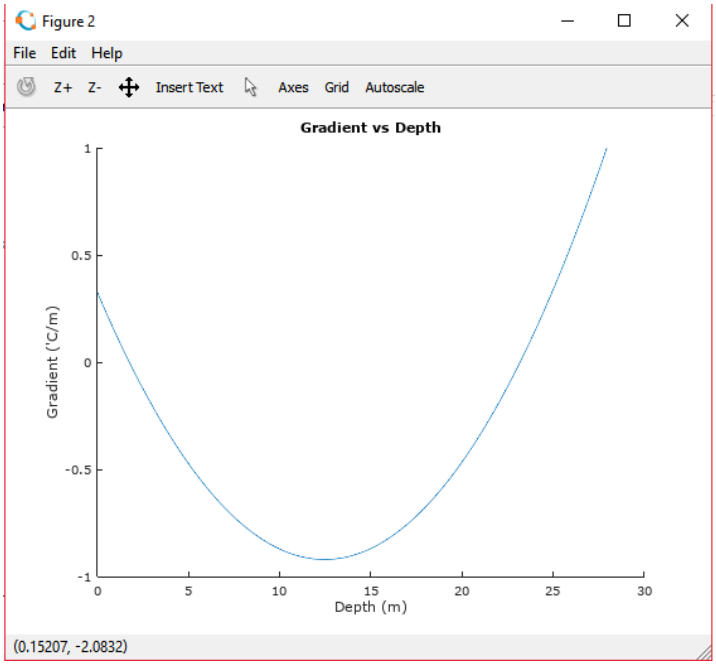


Figure 12: The Gradient vs Depth graph for Curve Fitting

Monomial Basis

I. Introduction

Given the data from the specifications:

z (m)	0	2.3	4.9	9.1	13.7	18.3	22.9	27.2
T (^{o}C)	22.8	22.8	22.8	20.6	13.9	11.7	11.1	11.1

The specifications asks to find a curve that positions itself to the data using monomial basis. With that, we can determine the thermocline or the inflection point, the gradient and the heat flux. With those values, we can already plot the curve.

2.1 Finding the Cubic Polynomial

Given the data from the specifications:

z (m)	0	2.3	4.9	9.1	13.7	18.3	22.9	27.2
T (^{o}C)	22.8	22.8	22.8	20.6	13.9	11.7	11.1	11.1

We are asked to find a 8x8 Vandermonde matrix which represents a polynomial of degree 7 so that it can position the data. Again we will be using the formula:

$$A\vec{x} = \vec{b}$$

The variable x and temp will be reused for each of the sections in this documentation

The 8x8 matrix will be defined as follows it will be named x1 as called also in the code:

$$\text{x1} = \begin{bmatrix} 1 & x & x\text{^}2 & x\text{^}3 & x\text{^}4 & x\text{^}5 & x\text{^}6 & x\text{^}7 \\ 1 & x1 & x1\text{^}2 & x1\text{^}3 & x1\text{^}4 & x1\text{^}5 & x1\text{^}6 & x1\text{^}7 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x8 & x8\text{^}2 & x8\text{^}3 & x8\text{^}4 & x8\text{^}5 & x8\text{^}6 & x8\text{^}7 \end{bmatrix}$$

```
printf("2.1 = Vandermond atrix which represents the polynomial\n")
x1 = [
  1 x(1) x(1)^2 x(1)^3 x(1)^4 x(1)^5 x(1)^6 x(1)^7
  1 x(2) x(2)^2 x(2)^3 x(2)^4 x(2)^5 x(2)^6 x(2)^7
  1 x(3) x(3)^2 x(3)^3 x(3)^4 x(3)^5 x(3)^6 x(3)^7
  1 x(4) x(4)^2 x(4)^3 x(4)^4 x(4)^5 x(4)^6 x(4)^7
  1 x(5) x(5)^2 x(5)^3 x(5)^4 x(5)^5 x(5)^6 x(5)^7
  1 x(6) x(6)^2 x(6)^3 x(6)^4 x(6)^5 x(6)^6 x(6)^7
  1 x(7) x(7)^2 x(7)^3 x(7)^4 x(7)^5 x(7)^6 x(7)^7
  1 x(8) x(8)^2 x(8)^3 x(8)^4 x(8)^5 x(8)^6 x(8)^7
]
```

Figure 13: The Vandermonde matrix in Octave

```
2.1 = Vandermond atrix which represents the polynomial
x1 =

  1.0000e+000  0.0000e+000  0.0000e+000  0.0000e+000  0.0000e+000  0.0000e+000  0.0000e+000  0.0000e+000
  1.0000e+000  2.3000e+000  5.2900e+000  1.2167e+001  2.7984e+001  6.4363e+001  1.4804e+002  3.4048e+002
  1.0000e+000  4.9000e+000  2.4010e+001  1.1765e+002  5.7648e+002  2.8248e+003  1.3841e+004  6.7822e+004
  1.0000e+000  9.1000e+000  8.2810e+001  7.5357e+002  6.8575e+003  6.2403e+004  5.6787e+005  5.1676e+006
  1.0000e+000  1.3700e+001  1.8769e+002  2.5714e+003  3.5228e+004  4.8262e+005  6.6119e+006  9.0582e+007
  1.0000e+000  1.8300e+001  3.3489e+002  6.1285e+003  1.1215e+005  2.0524e+006  3.7558e+007  6.8732e+008
  1.0000e+000  2.2900e+001  5.2441e+002  1.2009e+004  2.7501e+005  6.2976e+006  1.4422e+008  3.3025e+009
  1.0000e+000  2.7200e+001  7.3984e+002  2.0124e+004  5.4736e+005  1.4888e+007  4.0496e+008  1.1015e+010


================================================
```

Figure 14: The Vandermonde matrix output in Octave

Now going back to the formula, A will now be x1 and b will temp like last time

$$\begin{bmatrix} 1 & x & x\text{^}2 & x\text{^}3 & x\text{^}4 & x\text{^}5 & x\text{^}6 & x\text{^}7 \\ 1 & x1 & x1\text{^}2 & x1\text{^}3 & x1\text{^}4 & x1\text{^}5 & x1\text{^}6 & x1\text{^}7 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x8 & x8\text{^}2 & x8\text{^}3 & x8\text{^}4 & x8\text{^}5 & x8\text{^}6 & x8\text{^}7 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix} = temp$$

Or

$$x1x = temp$$

The 8 coefficients a_o- a_7 respectively in increasing degree. To solve the coefficients I use the same cholesky function in the previous item to solve for coefficients. It will be stored as follows:

$$ans = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{bmatrix}$$

The output is as follows:

```
1.2 = Coefficients of the function a0 - a7 respectively
ans =

  2.2800e+001
  3.7704e-001
  -3.6851e-001
  1.2678e-001
  -1.9290e-002
  1.3279e-003
  -4.1925e-005
  4.9619e-007
```

Figure 15: The output of the coefficients in Octave

Then the function f(x) would now be:

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + a_5x^5 + a_6x^6 + a_7x^7$$

$$f'(x) = a_1 + a_2x + 3a_3x^2 + 4a_4x^3 + 5a_5x^4 + 6a_6x^5 + 7a_7x^6$$

$$f''(x) = 2a_2 + 6a_3x + 12a_4x^2 + 20a_5x^3 + 30a_6x^4 + 42a_7x^5$$

$$f'''(x) = 6a_3 + 24a_4x + 60a_5x^2 + 120a_6x^3 + 210a_7x^4$$

These derivatives will later be used in finding the thermocline and gradient.

These functions are also defined in Octave, they are defined in Horner's Method form:

```
#Function to get the first derivative

function ret = first(x,a)
  ret =x*(x*(x*(x*(x*(x*7*a(8)+6*a(7))+5*a(6))+4*a(5))+3*a(4))+2*a(3))+a(2);
endfunction

#Function to get the second derivative

function ret = inflex(x,a)
  ret =x*(x*(x*(x*(x*42*a(8)+30*a(7))+20*a(6))+12*a(5))+6*a(4))+2*a(3);
endfunction

#Function to het the third derivative

function ret = inflex2(x,a)
  ret = x*(x*(x*(x*210*a(8)+120*a(7))+60*a(6))+24*a(5))+6*a(4);
endfunction
```

Figure 16: The first, second derivative defined in code in Horner's Method form

2.2 Getting the Thermocline

The thermocline is the point where maximum absolute value of the gradient and is also the point of inflection in the curve. Being an inflection point means that its second derivative is equal to 0. The second derivative of the function is of degree 5 meaning there at most 5 values of x or the roots. So in order to find the roots, we will use Newton's method in finding the roots

It is defined as follows:

$$x_1 = x_0 - \frac{f(x_0)}{f(x_1)}$$

Newton's method will get the root as x_1 gets closer and closer to the root and after a certain number of iterations it will be get the root. We place an arbritary number x_0 which can be defined from the range of values of x. The function is the second derivative so in order to use the Newton's method we use the third derivative of the function.

In this case, we will use the thermocline in the previous item (curve fitting) in order to have a more accurate answer for the root. It will be saved in thermocline (12.476). The output of the thermocline will be overwritten by the variable thermocline.

```
thermocline = 12.476;


curr = thermocline + (inflex(thermocline, ans)/inflex2(thermocline, ans));
old = curr;
err = curr;

while err > 0.01
  curr = thermocline - (inflex(thermocline, ans)/inflex2(thermocline, ans));
  err = abs(curr-old);
  old = curr;
  thermocline = curr;
endwhile

printf("2.3 = Get the inflection point\n")


thermocline = curr;

printf("Thermocline = %f m", thermocline)

printf("\n================================================\n")
```

Figure 17: Solving for the thermocline in Octave

```
================================================
2.3 = Get the inflection point
Thermocline = 11.611997 m
================================================
```

Figure 18: The output of the thermocline in Octave

2.3 Solving for the Heatflux

Again, we are asked to find the heat flux along the thermocline. From the previous part, we have derived the formula so now we just plug it in with the value we get from the gradient with the thermocline. Again, we will save the gradient in gradient and heat flux in the heatflux

```
printf("2.5 = Get the gradient(First Derivative)\n")


gradient = first(thermocline, ans);

printf("Gradient = %f C /m \n", gradient)

printf("================================================\n")
```

Figure 19: Solving for the gradient using the thermocline solved

```
printf("2.6 = Get the heatflux using the gradient\n")

heatflux = ((-0.01*(gradient/100))*86400);

printf("Heat flux = %f cal / (cm^2 day)\n", heatflux)
```

Figure 20: Solving for the heat flux in Monomial Basis Method

2.4 Plotting

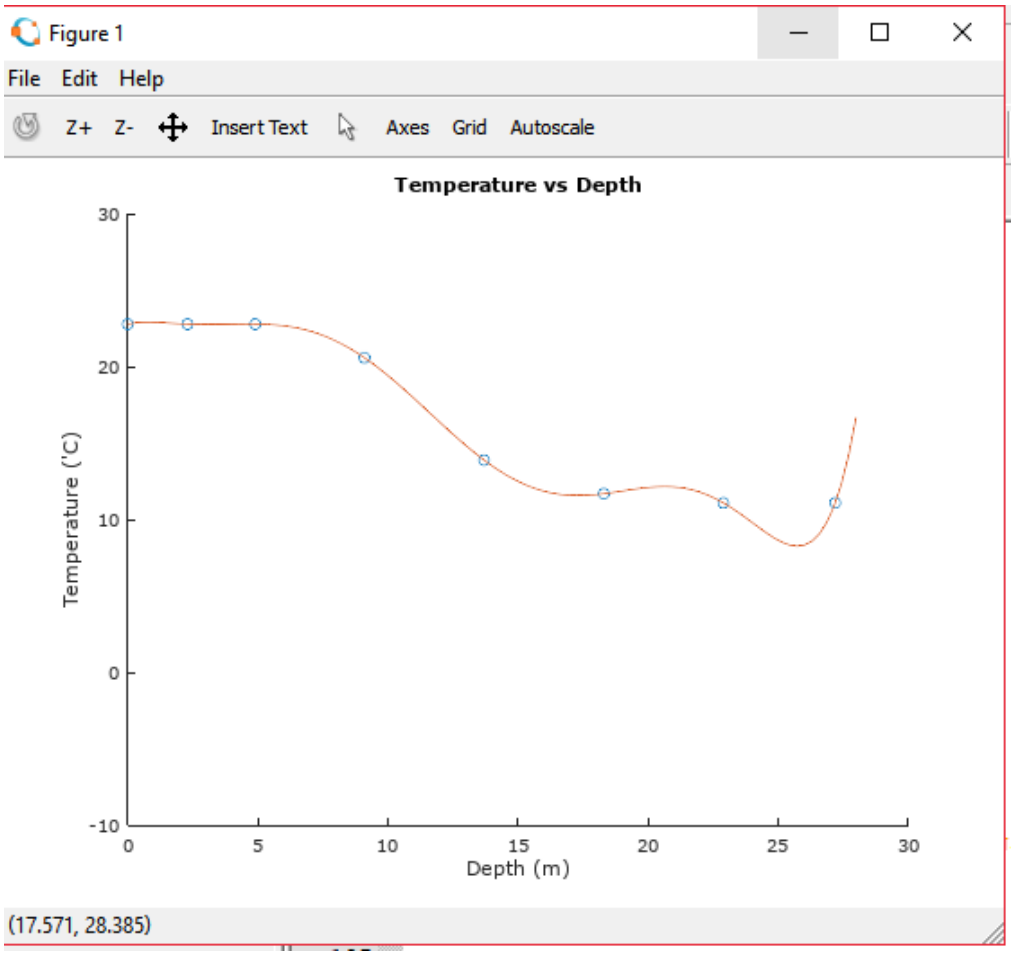


Figure 21: The Temperature vs Depth graph for Monomial Basis

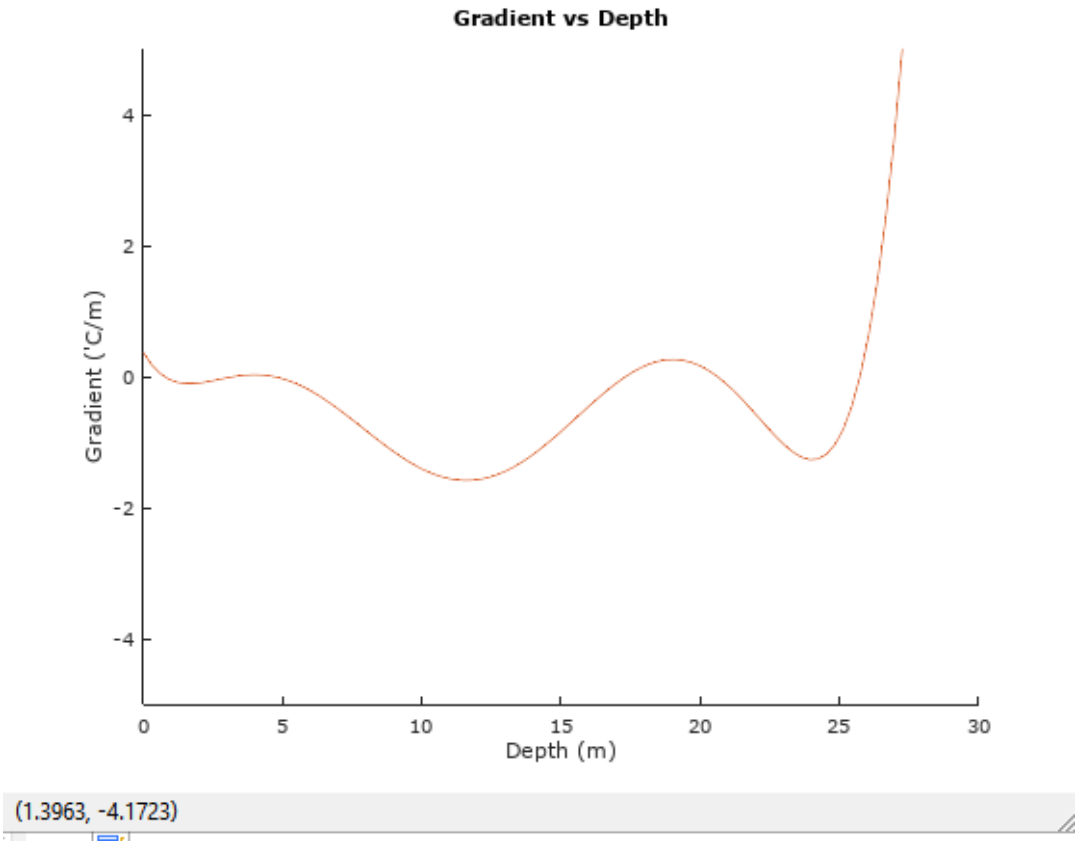


Figure 22: The Gradient vs Depth graph for Monomial Basis

Cubic Spline Interpolation

3.1 Getting the matrix

Given the data from the specifications:

z (m)	0	2.3	4.9	9.1	13.7	18.3	22.9	27.2
T (^{o}C)	22.8	22.8	22.8	20.6	13.9	11.7	11.1	11.1

The specifications asks to find a curve that positions itself to the data using cubic spline inter polation. With that, we can determine the thermocline or the inflection point, the gradient and the heat flux. With those values, we can already plot the curve.

We need the find the 7 cubic polynomials to for a piecewise function for the interpolation. It will called our cubic spline interpolant.

We have the function again f:

$$f(x) = a_{4(i-1)} + a_{4(i-1)}x + a_{4(i-1)}x^2 + a_{4(i-1)}x^3$$

$$f'(x) = a_{4(i-1)} + 2a_{4(i-1)}x + 3a_{4(i-1)}x^2$$

$$f''(x) = 2a_{4(i-1)} + 6a_{4(i-1)}x$$

$$f'''(x) = 6a_{4(i-1)}$$

Using these equations and x we will construct the matrix gg which is a 28x28 matrix:

```
gg =[
  x(1)^3 x(1)^2 x(1) 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  x(2)^3 x(2)^2 x(2) 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 x(2)^3 x(2)^2 x(2) 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 x(3)^3 x(3)^2 x(3) 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 x(3)^3 x(3)^2 x(3) 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 x(4)^3 x(4)^2 x(4) 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 x(4)^3 x(4)^2 x(4) 1 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 x(5)^3 x(5)^2 x(5) 1 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x(5)^3 x(5)^2 x(5) 1 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x(6)^3 x(6)^2 x(6) 1 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x(6)^3 x(6)^2 x(6) 1 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x(7)^3 x(7)^2 x(7) 1 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x(7)^3 x(7)^2 x(7) 1
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x(8)^3 x(8)^2 x(8) 1
  (3*(x(2)^2)) (2*(x(2))) 1 0 -(3*(x(2)^2)) -(2*(x(2))) -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 (3*(x(3)^2)) (2*(x(3))) 1 0 -(3*(x(3)^2)) -(2*(x(3))) -1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 (3*(x(4)^2)) (2*(x(4))) 1 0 -(3*(x(4)^2)) -(2*(x(4))) -1 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 (3*(x(5)^2)) (2*(x(5))) 1 0 -(3*(x(5)^2)) -(2*(x(5))) -1 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (3*(x(6)^2)) (2*(x(6))) 1 0 -(3*(x(6)^2)) -(2*(x(6))) -1 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 (3*(x(7)^2)) (2*(x(7))) 1 0 -(3*(x(7)^2)) -(2*(x(7))) -1 0
  6*x(2) 2 0 0 -6*x(2) -2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 6*x(3) 2 0 0 -6*x(3) -2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 6*x(4) 2 0 0 -6*x(4) -2 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 6*x(5) 2 0 0 -6*x(5) -2 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6*x(6) 2 0 0 -6*x(6) -2 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6*x(7) 2 0 0 -6*x(7) -2 0 0
  3*(x(1)**2) 2*x(1) 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3*(x(8)**2) 2*x(8) 1 0
```

Figure 23: How the 28x28 matrix is represented in Octave

Using the classic Ax=b equation we will now form get the coefficients again of the 7 polynomials.

We must first now construct the ‘b’ in our equation. In previous items it was b now we convert b to satisfy for a 28x28 matrix which will now be a 28x1 matrix:

$$b = \begin{bmatrix} 22.8 \\ 22.8 \\ 22.8 \\ 22.8 \\ 22.8 \\ 20.6 \\ 20.6 \\ 13.9 \\ 13.9 \\ 11.7 \\ 11.7 \\ 11.1 \\ 11.1 \\ 11.1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

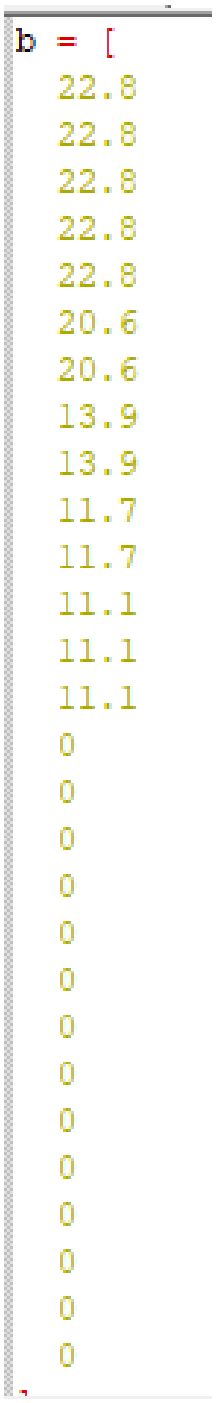


Figure 24: How b is reprented in Octave

Now we can now use Ax = b with A being gg and b being b

$$gg\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ \vdots \\ a_{27} \end{bmatrix} = b$$

The coefficient matrix will now be defined as x which is 28x1.

To solve for x we simply

$$x = gg\backslash b$$

These coefficients will now be used for the piecewise function:f(x) where x<= 2.3

$$b(x) = \begin{cases} \text{f(x)where x} <= 2.3 \\ \text{g(x) where } 2.3 < \text{ x} <= 4.9 \\ \text{h(x) where } 4.9 < \text{ x} <= 9.1 \\ \text{i(x) where } 9.1 < \text{ x} <= 13.7 \\ \text{j(x) where } 13.7 < \text{x} <= 18.3 \\ \text{k(x) where } 18.3 < \text{ x} <= 22.9 \\ \text{l(x)where x} > 27 \end{cases}$$

With a0-a3 being in f(x) a4-a7 being in g(x) and so on and so forth. I also save each coefficient interval inside separate matrices for easy access.

```
FirstPolynomial = [x(1) x(2) x(3) x(4)];
SecondPolynomial = [x(5) x(6) x(7) x(8)];
ThirdPolynomial = [x(9) x(10) x(11) x(12)];
FourthPolynomial = [x(13) x(14) x(15) x(16)];
FifthPolynomial = [x(17) x(18) x(19) x(20)];
SixthPolynomial = [x(21) x(22) x(23) x(24)];
SeventhPolynomial = [x(25) x(26) x(27) x(28)];
```

Figure 25: How I save the polynomial coeffs

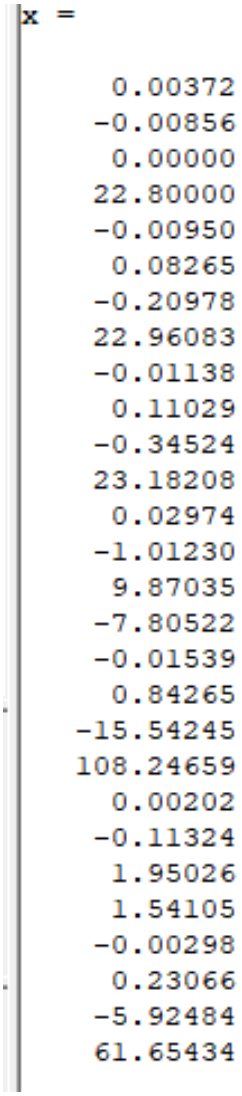


Figure 26: Output for the coefficients

3.2 Getting the thermocline

The thermocline is the point where maximum absolute value of the gradient and is also the point of inflection in the curve. Being an inflection point means that its second derivative is equal to 0. The second derivative of the function is of degree 1 meaning there at most values of 1 or the roots. So in order to find the roots, we will use Newton's method in finding the roots

It is defined as follows:

$$x_1 = x_0 - \frac{f(x_0)}{f(x_1)}$$

Newton's method will get the root as x_1 gets closer and closer to the root and after a certain number of iterations it will be get the root. We place an arbritary number x_0 which can be defined from the range of values of x. The function is the second derivative so in order to use the Newton's method we use the third derivative of the function.

In this case, we will use the thermocline in the previous item (monomial bassi) in order to have a more accurate answer for the root. It will be saved in thermocline (11.612). The output of the thermocline will be overwritten by the variable thermocline. The newton function will be used seven times for each of the polynomials and applied through the gradient to find the one true gradient to be used through the heat flux. The one true gradient is found by finding the highest absolute value in the list of gradients.

```
#Function to get the first derivative
function ret = first(x,a)
  ret = (3*(a(1)*(x^2))) + (2*a(2)*x) + a(3);
endfunction
#Function to get the second derivative

function ret = second(x,a)
  ret =  6*(a(1))*x + 2*a(2);
endfunction
#Function to het the third derivative

function ret = third(a)
  ret = 6*a(1);
endfunction
#Function to het the roots of a polynomial using Newton's Method

function ret = newton(Polynomial)
  old = 11.612;
  err = old;

  while err > 0.01
   curr = old - (second(old, Polynomial)/third(Polynomial));
   err = abs(curr-old);
    old = curr;
  endwhile

  thermocline = curr;

  ret = thermocline;
```

Figure 27: My first, second, third derivative functions and Newton function

```
================================================
3.2 = Get the inflection point
Thermocline: 11.344960 m
================================================

3.3 = Get the gradient(First Derivative)
gradient =

  -0.0065627
   0.0299753
   0.0111552
  -1.6141324
  -0.1628076
  -0.1637427
   0.0183908

The one true gradient: -1.614132 C/m
```

Figure 28: Output of the Inflection Point and the one true gradient

3.3 Getting the heat flux

Again, we are asked to find the heat flux along the thermocline. From the previous part, we have derived the formula so now we just plug it in with the value we get from the gradient with the thermocline. Again, we will save the gradient in gradient and heat flux in the heatflux.

```
printf("3.4 = Get the heatflux using the gradient\n")

heatflux = ((-0.01*(curr/100))*86400);

printf("Heat flux = %f cal / (cm^2 day)\n", heatflux)
```

Figure 29: heat flux in Octave

```
3.4 = Get the heatflux using the gradient
Heat flux = 13.946104 cal / (cm^2 day)
>>
```

Figure 30: Output of the heat flux in Octave

3.4 Plotting

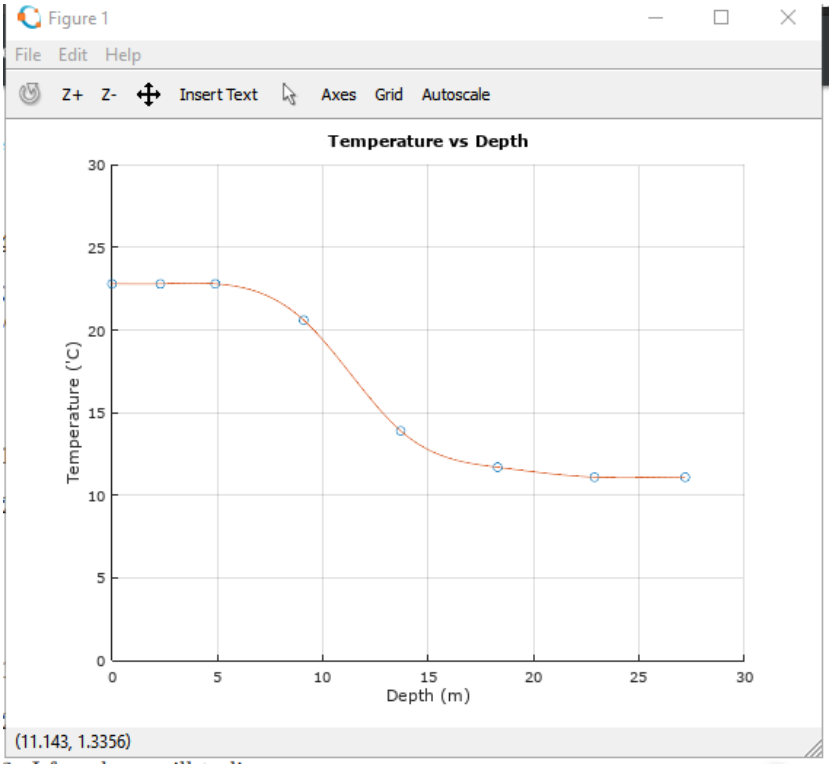


Figure 31: Temperature vs Depth graph in Cubic Splines

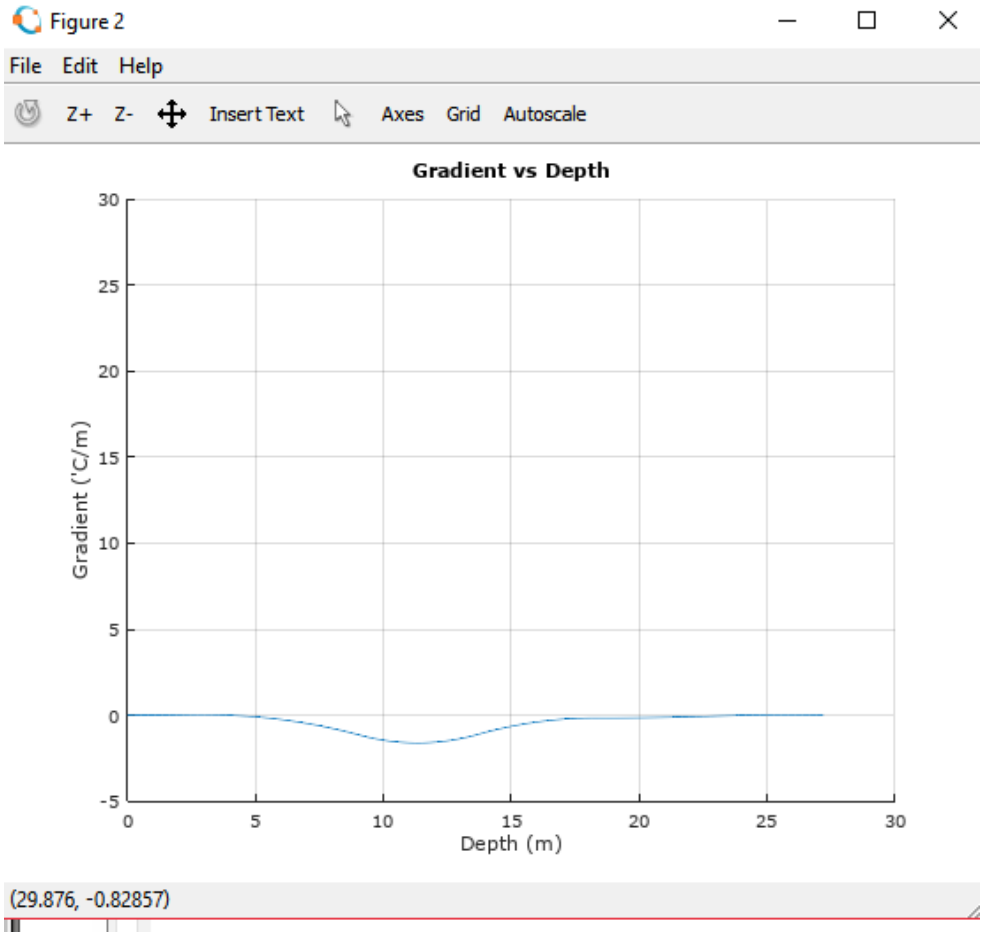


Figure 32: Gradient vs Depth in Cubic Splines

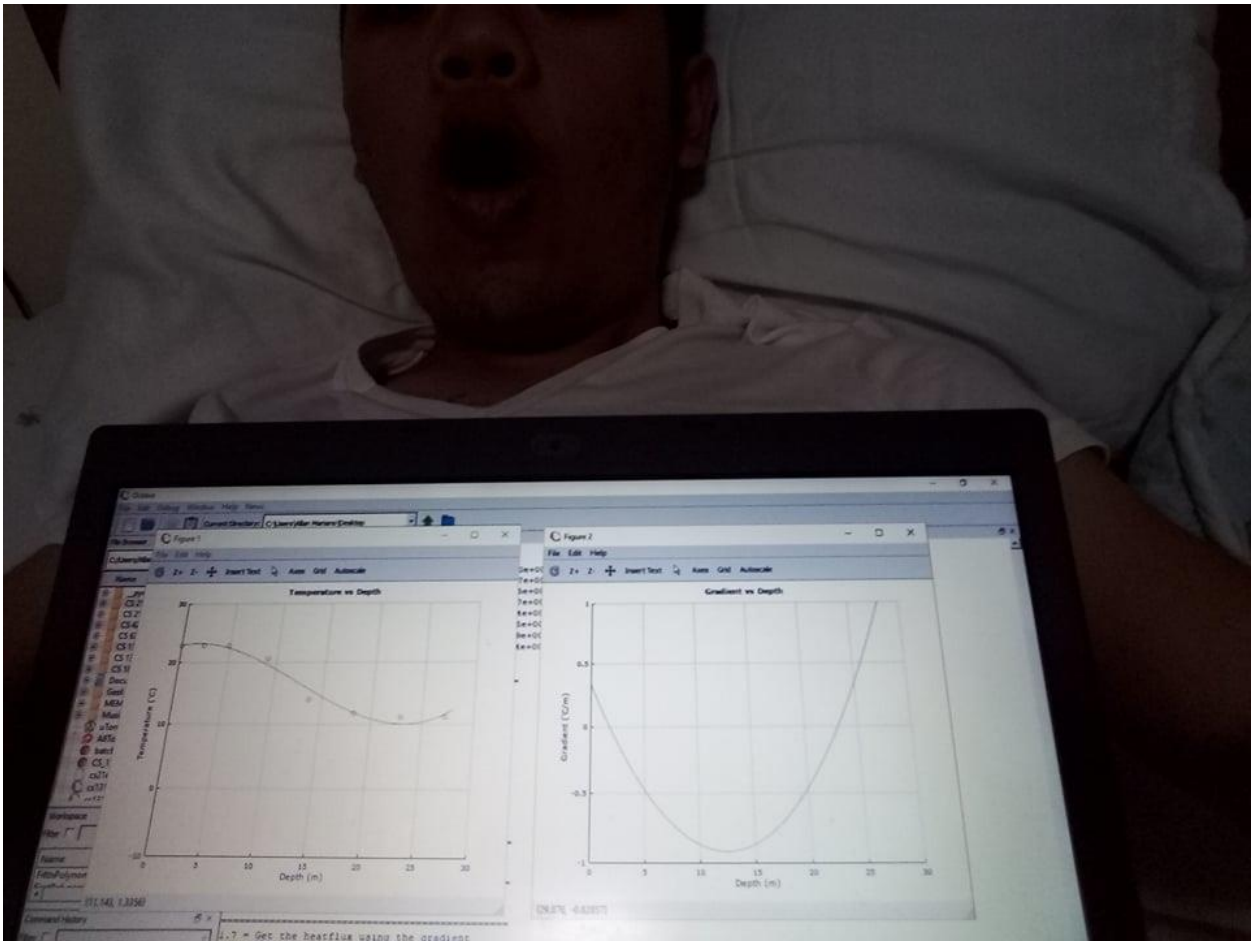
Things I learned while doing this MP

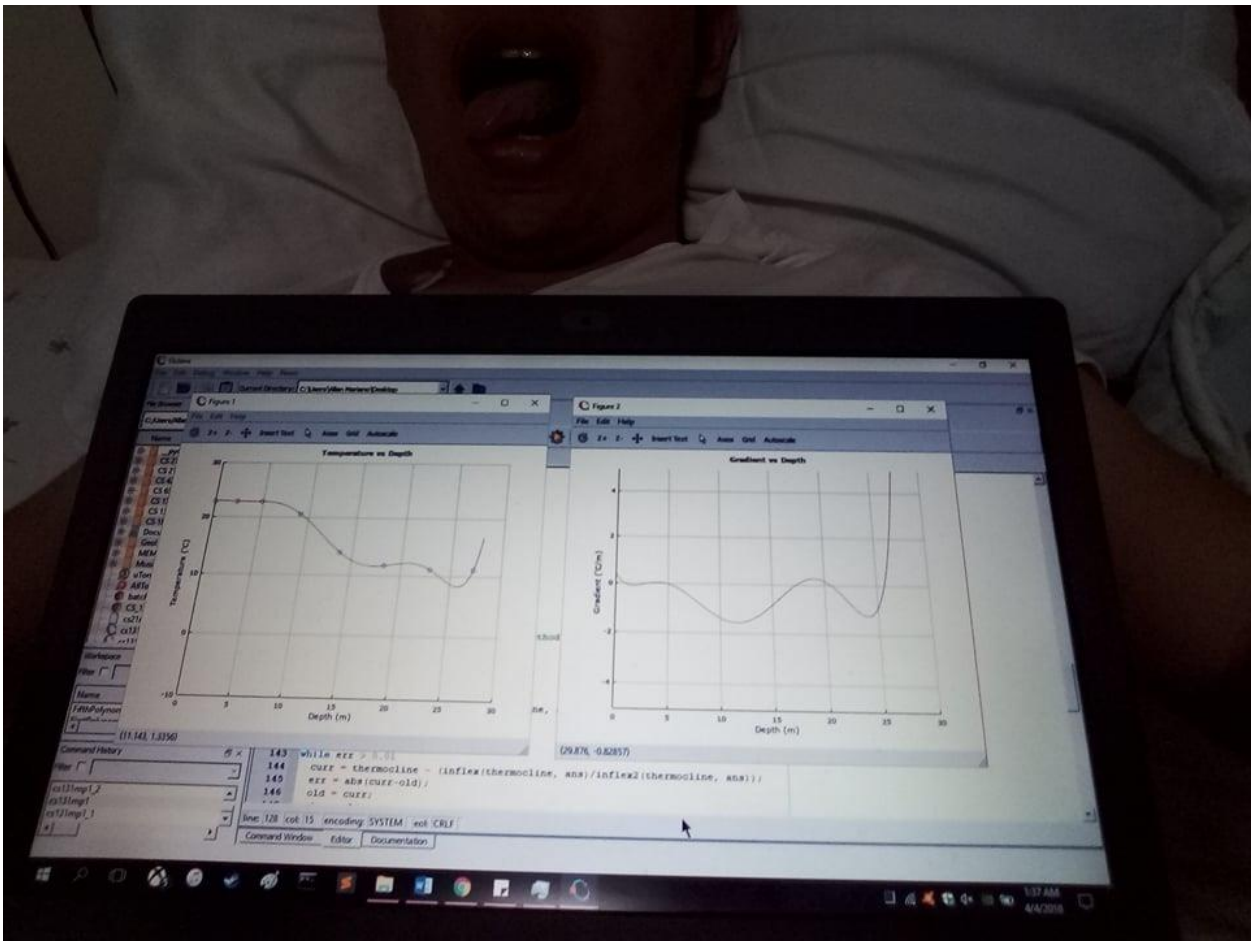
1. How to properly code in Octave
2. How to curve fitting, monomial basis, cubic splines work
3. How to really apply what you've learned from 131 to code

Things I thought of while doing the MP

1. I've been really independent on this machine problem and I'm proud of it
2. Its fun to graph
3. Its satisfying to see your code work

Selfie: Part 1,2,3 respectively




Temperature vs Depth
Temperature (°C)
Depth (m)
Gradient vs Depth
Gradient (°C/m)
Depth (m)
while err > 0.01
curr = thermocline - (inflex(thermocline, ans)/inflex2(thermocline, ans));
err = abs(curr-old);
old = curr;
Command Window
Editor
Documentation

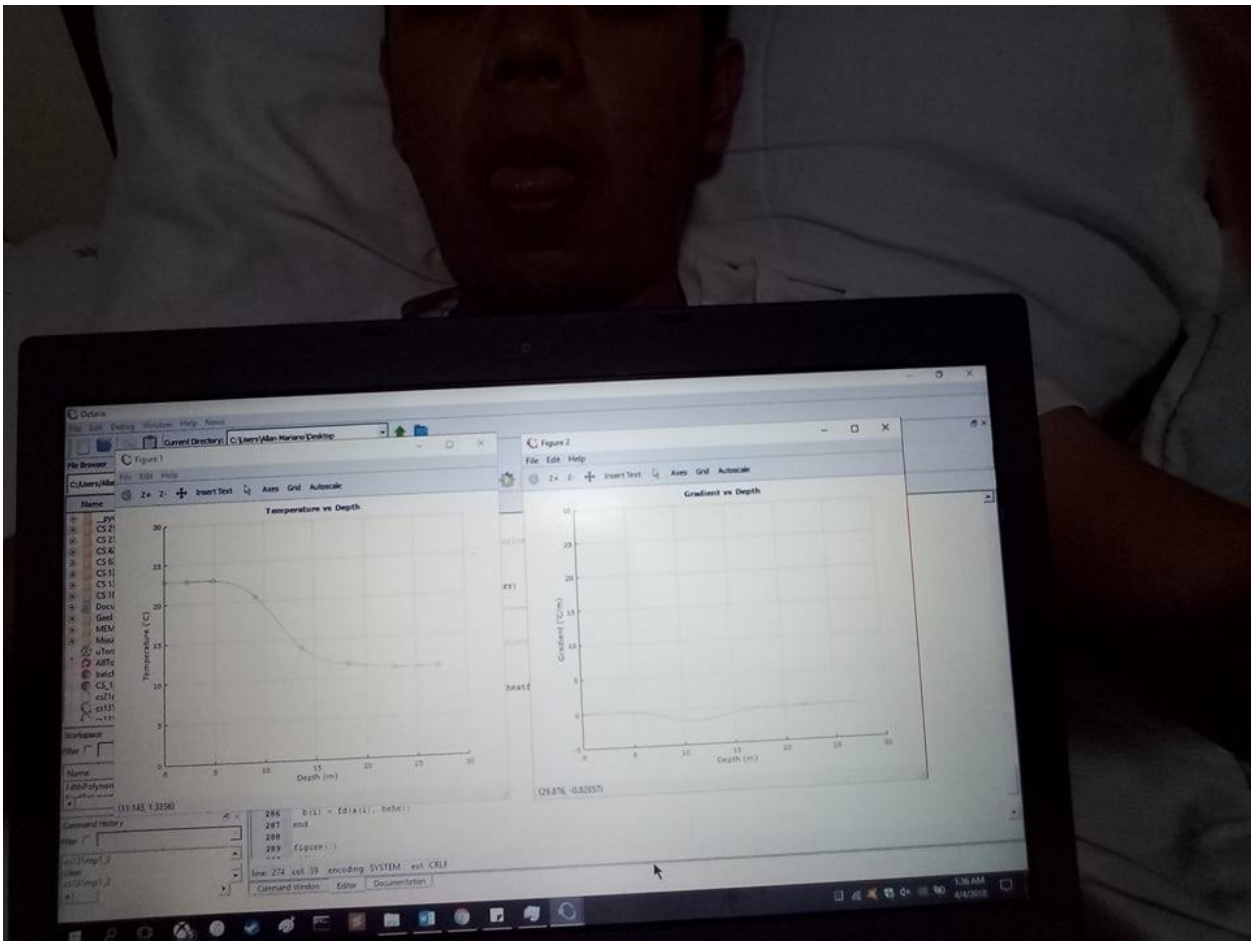


Figure 1
Temperature vs Depth
Temperature (C)
Depth (m)
Figure 2
Gradient vs Depth
Gradient (°C/m)
Depth (m)