Assignment 1

Colin Turner: 250956100

MME 9621: Computational Methods in Mechanical Engineering

Dr. Mohammad Z. Hossain

February 9, 2024

Q1a)

The following Figures show 9.6 and 100.2 converted to binary and expressed as floating point numbers fl(x) with IEEE rounding to nearest rule (all 52 bits)

1a) $9_{10} = 1001_2$

$0.6_{10} = 0.1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 10011_2$

$9.6_{10} = 1001.1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 10011_2$

$9.6_{10} = 1.001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 1001\ 10011_2 \times 2^3$

$3 + 2^{(11-1)} - 1 = 1026_{10} = 100\ 0000\ 0010_2$

$\therefore 9.6_{10} = 0 - 100\ 0000\ 0010 - 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011\ 0011$

Q1b)

b) $100_{10} = 110\ 0100$

$0.2_{10} = 0.0011001100110011001100110011001100110011001100110011\,0_2$

$100.2_{10} = 110\ 0100.0011001100110011001100110011001100110011001100110011\,0_2$

$100.2_{10} = 1.10\ 0100\ 0011001100110011001100110011001100110011001100110011\,0_2 \times 2^6$

1

$6 + 2^{(11-1)} - 1 = 1029_{10} = 100\ 0000\ 0101_2$

$\therefore 100.2_{10} = 0 - 100\ 0000\ 0101 - 1001\ 0000\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1100\ 1101$

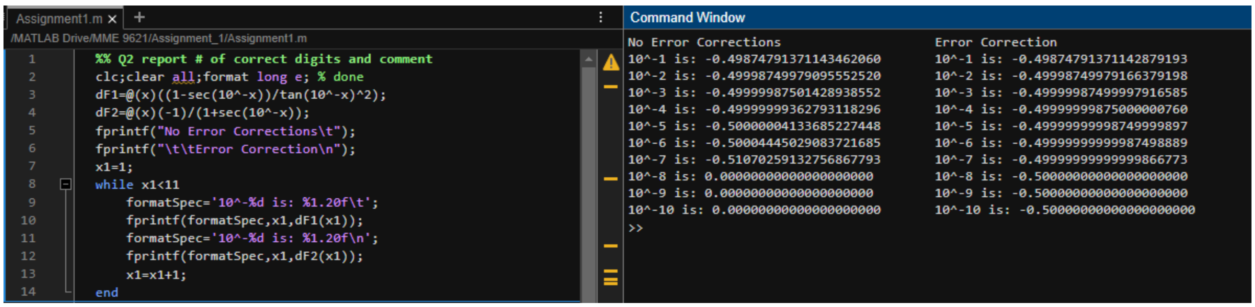
Q2)

Using the conjugate and the trig identity 1+tan^2(x)=sec^2(x) we can change the equation to negate error brought from subtracting two almost identical numbers. This also allows the equation to have an x value at 0 instead of being undefined.

2) $\frac{1-\sec x}{\tan^2 x} \times \frac{1+\sec x}{1+\sec x}$ To Reduce Error

$$1+\tan^2 x = \sec^2 x$$

$$\frac{1-\sec^2 x}{\tan^2 x + \tan^2 x \sec x} \to \frac{1-1-\tan^2 x}{\tan^2 x + \tan^2 x \sec x} \to \frac{-1}{1+\sec x}$$


Assignment1.m

/MATLAB Drive/MME 9621/Assignment_1/Assignment1.m

```
%% Q2 report # of correct digits and comment
clc;clear all;format long e; % done
dF1=@(x)((1-sec(10^-x))/tan(10^-x)^2);
dF2=@(x)(-1)/(1+sec(10^-x));
fprintf("No Error Corrections\t");
fprintf("\t\tError Correction\n");
x1=1;
while x1<11
    formatSpec='10^-%d is: %1.20f\t';
    fprintf(formatSpec,x1,dF1(x1));
    formatSpec='10^-%d is: %1.20f\n';
    fprintf(formatSpec,x1,dF2(x1));
    x1=x1+1;
end
```

Command Window

```
No Error Corrections                    Error Correction
10^-1 is: -0.49874791371143462060       10^-1 is: -0.49874791371142879193
10^-2 is: -0.49998749979095552520       10^-2 is: -0.49998749979166379198
10^-3 is: -0.49999987501428938552       10^-3 is: -0.49999987499997916585
10^-4 is: -0.49999999362793118296       10^-4 is: -0.49999999875000000760
10^-5 is: -0.50000004133685227448       10^-5 is: -0.49999999998749999897
10^-6 is: -0.50004445029083721685       10^-6 is: -0.49999999999987498889
10^-7 is: -0.51070259132756867793       10^-7 is: -0.49999999999999866773
10^-8 is: 0.00000000000000000000        10^-8 is: -0.50000000000000000000
10^-9 is: 0.00000000000000000000        10^-9 is: -0.50000000000000000000
10^-10 is: 0.00000000000000000000       10^-10 is: -0.50000000000000000000
>>
```

As x approaches zero without error correction the number of correct digits decreases, 14,12,9,9,1,1,1,1,1,1,1,1,1,1. As the function begins to suffer from subtraction error from two almost identical numbers.

As x approaches zero with error correction, we can see that more correct digits are retained until 10^-8 where rounding error occurs and it results in -0.5 as -1/(1+sec(0)) = -0.5. which is the correct solution to this problem. This method also does not suffer from subtracting almost equal numbers like the previous solution.

Q3a)

The Figure below describes how the formula was derived and how the variables were assigned.

$$F(x) = \pi d L\left[h(T_s - T_a) + \varepsilon\sigma(T_s^4 - T_{surr}^4)\right] - Q$$

$$F(x) = \underbrace{\pi d L h}T_s - \pi d L h T_a + \pi d L \varepsilon\sigma T_s^4 - \pi d L \varepsilon\sigma T_{surr}^4 - Q$$

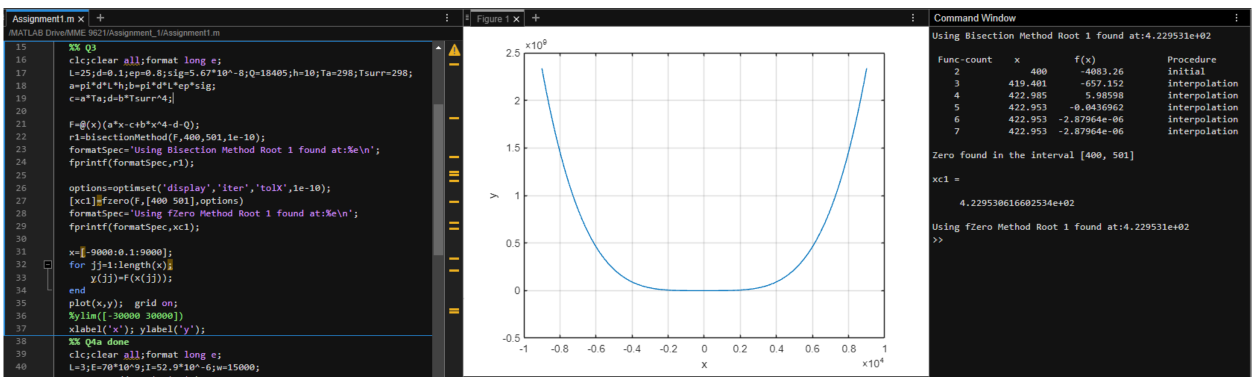
$$a = \pi d L h \qquad c = aT_a$$

$$b = \pi d L \varepsilon\sigma \qquad d = bT_{surr}^4$$

$$F(x) = aT_s - aT_a + bT_s^4 - bT_{surr}^4 - Q$$

$$F(x) = aT_s - c + bT_s^4 - d - Q$$

$$F(x) =$$



A Graph was created to help narrow down the guesses for Bisection & fZero method. Through graphical analysis we can see that one zero is on the negative x axis, as Kelvin cannot be a negative value, we can disregard it. We can than find the root between 400 & 501, yielding a root of 423K when rounded using 3 significant digits. A more accurate value can be seen in the figure above. Both methods yielded the same results.

Q4a)

The Figure below describes how the formulas was derived and how the variables were assigned.

$$y = \frac{w_0}{120LEI}\left(3L^3x^2 - 7L^2x^3 + 5Lx^4 - x^5\right)$$

$$y = \frac{3wL^2x^2}{120EI} - \frac{7wLx^3}{120EI} + \frac{5wx^4}{120EI} - \frac{wx^5}{120LEI}$$

$$z = \frac{w}{120EI}$$

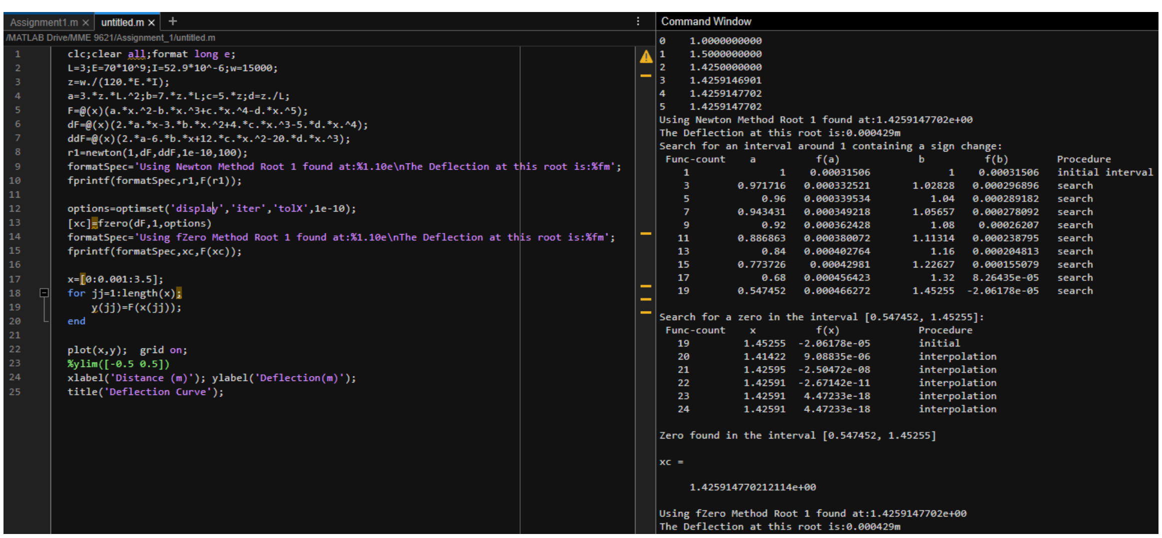
$$a = 3zL^2 \quad b = 7zL \quad c = 5z \quad d = z/L$$

$$y = ax^2 - bx^3 + cx^4 - dx^5$$

$$y' = 2ax - 3bx^2 + 4cx^3 - 5dx^4$$

$$y'' = 2a - 6bx + 12cx^2 - 20dx^3$$

A Graph was created to help narrow down the guesses for Newton Method & fZero method. Through graphical analysis we can see that one zero is located roughly in the center of the beam so we select 1 as out initial guess. This equation has 4 zeros however they fall outside of the bounds of our beam. The root was found at 1.42m rounded to 3 significant digits, with a deflection of 0.000429m rounded to 3 significant digits. A more accurate value can be seen in the figure below. Both methods yielded the same results. However, newtons method took less iterations to reach this value.



Assignment1.m × untitled.m ×

/MATLAB Drive/MME 9621/Assignment_1/untitled.m

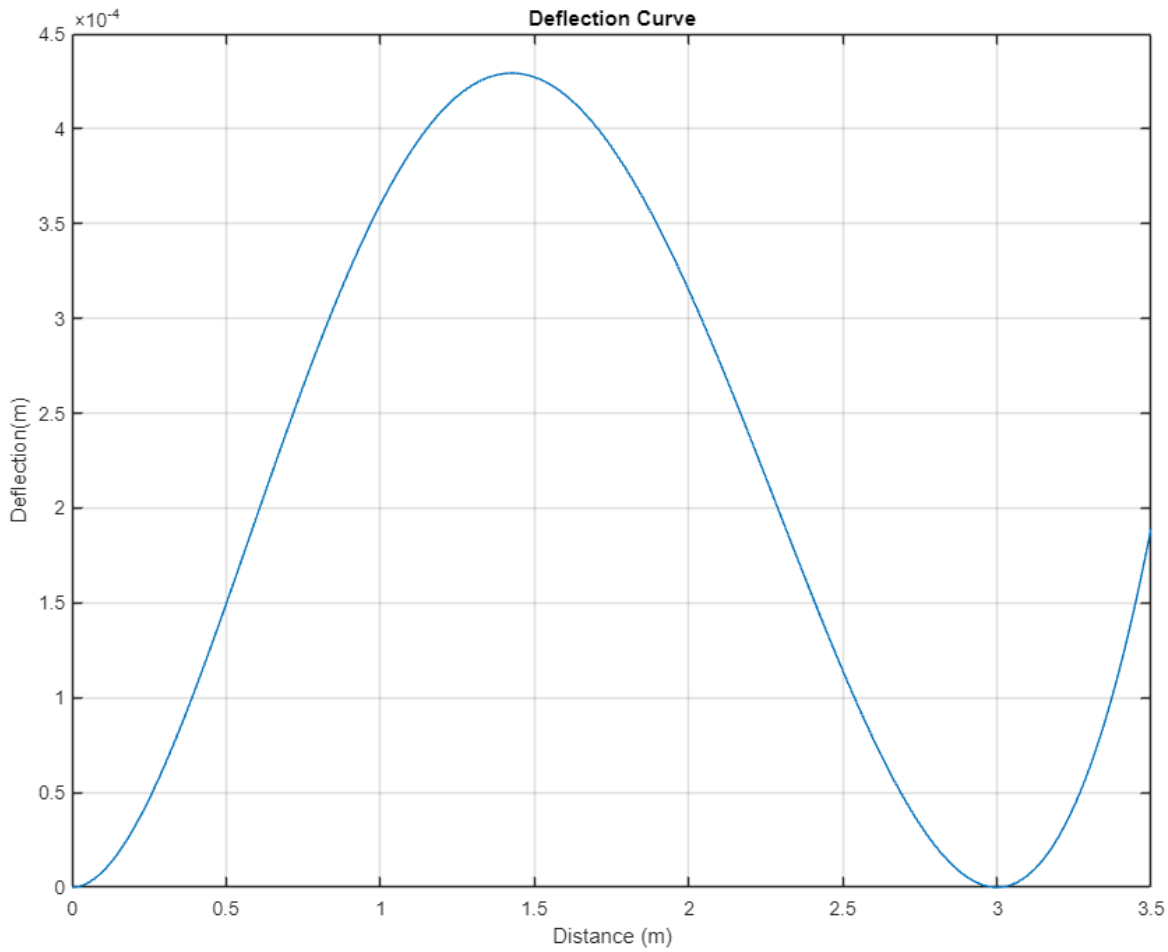
```
clc;clear all;format long e;
L=3;E=70*10^9;I=52.9*10^-6;w=15000;
z=w./(120.*E.*I);
a=3.*z.*L.^2;b=7.*z.*L;c=5.*z;d=z./L;
F=@(x)(a.*x.^2-b.*x.^3+c.*x.^4-d.*x.^5);
dF=@(x)(2.*a.*x-3.*b.*x.^2+4.*c.*x.^3-5.*d.*x.^4);
ddF=@(x)(2.*a-6.*b.*x+12.*c.*x.^2-20.*d.*x.^3);
r1=newton(1,dF,ddF,1e-10,100);
formatSpec='Using Newton Method Root 1 found at:%1.10e\nThe Deflection at this root is:%fm';
fprintf(formatSpec,r1,F(r1));

options=optimset('display','iter','tolX',1e-10);
[xc]=fzero(dF,1,options)
formatSpec='Using fZero Method Root 1 found at:%1.10e\nThe Deflection at this root is:%fm';
fprintf(formatSpec,xc,F(xc));

x=[0:0.001:3.5];
for jj=1:length(x);
    y(jj)=F(x(jj));
end

plot(x,y);  grid on;
%ylim([-0.5 0.5])
xlabel('Distance (m)'); ylabel('Deflection(m)');
title('Deflection Curve');
```

Command Window

```
0    1.0000000000
1    1.5000000000
2    1.4250000000
3    1.4259146901
4    1.4259147702
5    1.4259147702
Using Newton Method Root 1 found at:1.4259147702e+00
The Deflection at this root is:0.000429m
Search for an interval around 1 containing a sign change:
 Func-count    a          f(a)             b          f(b)        Procedure
    1            1     0.00031506           1     0.00031506   initial interval
    3     0.971716    0.000332521     1.02828    0.000296896   search
    5         0.96    0.000339534        1.04    0.000289182   search
    7     0.943431    0.000349218     1.05657    0.000278092   search
    9         0.92    0.000362428        1.08     0.00026207   search
   11     0.886863    0.000380072     1.11314    0.000238795   search
   13         0.84    0.000402764        1.16    0.000204813   search
   15     0.773726     0.00042981     1.22627    0.000155079   search
   17         0.68    0.000456423        1.32    8.26435e-05   search
   19     0.547452    0.000466272     1.45255   -2.06178e-05   search

Search for a zero in the interval [0.547452, 1.45255]:
 Func-count    x          f(x)             Procedure
   19      1.45255  -2.06178e-05           initial
   20      1.41422   9.08835e-06           interpolation
   21      1.42595  -2.50472e-08           interpolation
   22      1.42591  -2.67142e-11           interpolation
   23      1.42591   4.47233e-18           interpolation
   24      1.42591   4.47233e-18           interpolation

Zero found in the interval [0.547452, 1.45255]

xc =

    1.425914770212114e+00

Using fZero Method Root 1 found at:1.4259147702e+00
The Deflection at this root is:0.000429m
```

Q4b)

Below we can see the deflection curve of the beam. Note the y axis is *10^-4.



Q5a)

The Figure below describes how the formulas was derived and how the variables were assigned.

$$a = KA/dx \qquad b = 2a \qquad c = qA\,dx \qquad d = a+b$$

$$c + bT_A = e$$
$$c + bT_B = f$$

$$\begin{aligned} dT_1 - aT_2 - bT_A &= c \\ bT_2 - aT_1 - aT_3 &= c \\ bT_3 - aT_2 - aT_4 &= c \\ bT_4 - aT_3 - aT_5 &= c \\ dT_5 - aT_4 - bT_B &= c \end{aligned} \Rightarrow \begin{bmatrix} d & -a & 0 & 0 & 0 \\ -a & b & -a & 0 & 0 \\ 0 & -a & b & -a & 0 \\ 0 & 0 & -a & b & -a \\ 0 & 0 & 0 & -a & d \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} = \begin{bmatrix} e \\ c \\ c \\ c \\ f \end{bmatrix}$$

b) $$K\frac{d^2T}{dx^2} + q = 0$$

$$\frac{dT}{dx} = -\frac{q}{K}x + C_1$$

$$T(x) = -\frac{q}{2K}x^2 + xC_1 + C_2$$

$$T_{(0)} = C_2 = T_A$$

$$T_{(L)} = \frac{-qL^2}{2K} + C_1 L + T_A$$

$$C_1 = \frac{T_B - T_A}{L} + \frac{qL}{2K}$$

$$F(x) = \frac{-qx^2}{2K} + \left[\frac{T_B - T_A}{L} + \frac{qL}{2K}\right]x + T_A$$

Below we can see the application of \ Method, Inv Method & Tridiagonal Method all with their computation time. All Methods yielded the same results with computation time varying between each.

Assignment1.m × untitled.m ×

/MATLAB Drive/MME 9621/Assignment_1/Assignment1.m

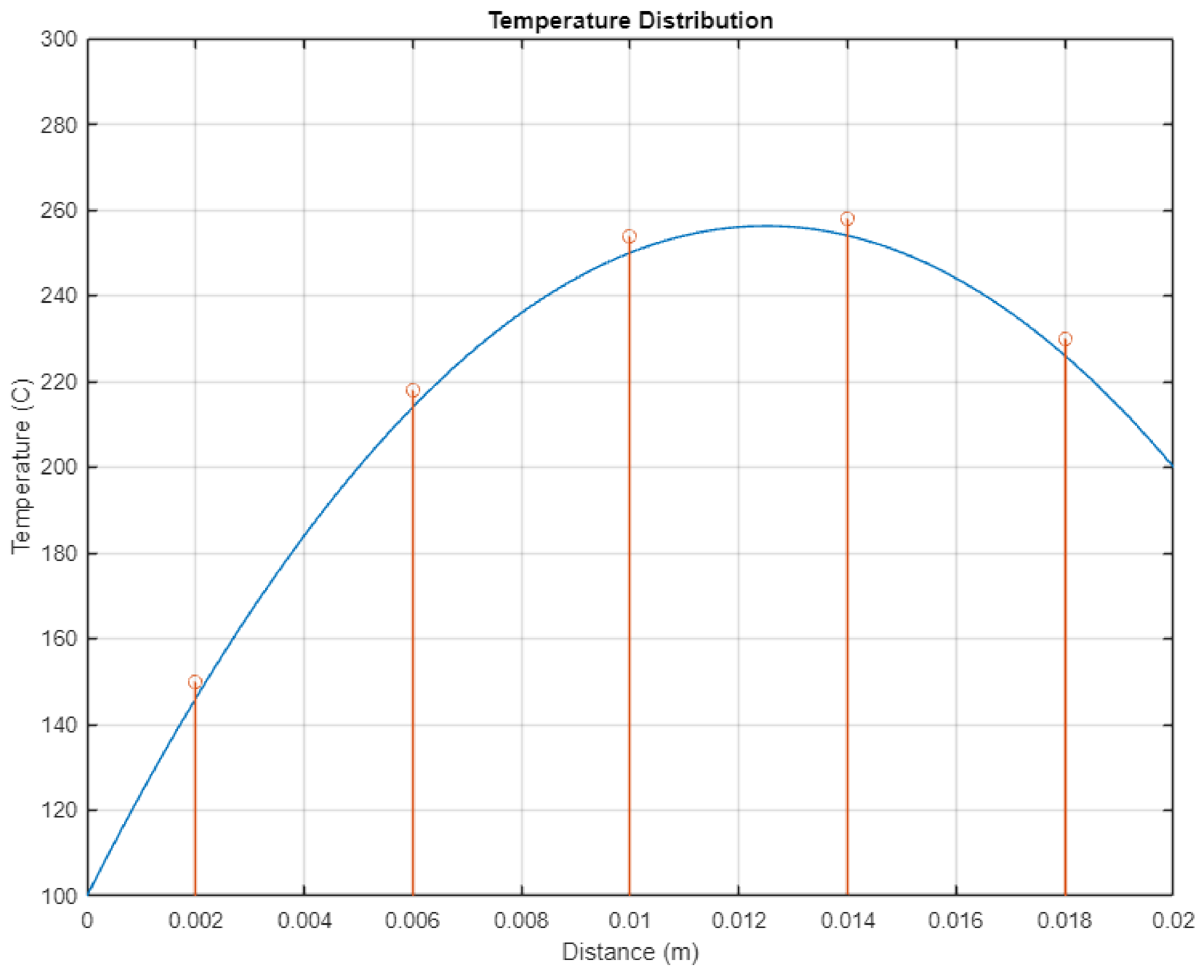
```
%% Q5a
clc;clear all;format long e;
L=0.02;k=0.5;q=10^6;Ta=100;Tb=200;dx=0.004;A=1;
a=k*A/dx;b=2*a;c=q*A*dx;d=a+b;
e=c+b*Ta;f=c+b*Tb;
% / Method
A=[d -a 0 0 0;
   -a b -a 0 0;
   0 -a b -a 0;
   0 0 -a b -a;
   0 0 0 -a d;
   ];
B=[e;c;c;c;f];
tic;
x1=A\B;
formatSpec='Using Backslash Method:\nT1=%3.0f\nT2=%3.0f\nT3=%3.0f\nT4=%3.0f\nT5=%3.0f\nComp Time:%fs\n';
fprintf(formatSpec,x1(1),x1(2),x1(3),x1(4),x1(5),toc);
% Inversion Method
tic;
xc=inv(A)*B;
formatSpec='Using Inversion Method:\nT1=%3.0f\nT2=%3.0f\nT3=%3.0f\nT4=%3.0f\nT5=%3.0f\nComp Time:%fs\n';
fprintf(formatSpec,xc(1),xc(2),xc(3),xc(4),xc(5),toc);
% Tridiagonal Method
tic;
a1=[0 -a -a -a -a];
b1=[d b b b d];
c1=[-a -a -a -a 0];
d1=[e;c;c;c;f];
[xout]=tridiagonal(a1,b1,c1,d1);
formatSpec='Using Tridiagonal Method:\nT1=%3.0f\nT2=%3.0f\nT3=%3.0f\nT4=%3.0f\nT5=%3.0f\nComp Time:%fs\n';
fprintf(formatSpec,xout(1),xout(2),xout(3),xout(4),xout(5),toc);
```

Command Window

```
Using Backslash Method:
T1=150
T2=218
T3=254
T4=258
T5=230
Comp Time:0.000285s
Using Inversion Method:
T1=150
T2=218
T3=254
T4=258
T5=230
Comp Time:0.000123s
Using Tridiagonal Method:
T1=150
T2=218
T3=254
T4=258
T5=230
Comp Time:0.002968s
>>
```

Q5b)

The figure below shows the temperature distribution in the plate as a function of x along with the plotted points derived from the previous step. The T values calculated prior are slightly higher than the integrated function.



Appendix

```
%% Q2 report # of correct digits and comment
clc;clear all;format long e; % done
F1=@(x)((1-sec(10^-x))/tan(10^-x)^2);
F2=@(x)(-1)/(1+sec(10^-x));
fprintf("No Error Corrections\t");
fprintf("\t\tError Correction\n");
x1=1;
while x1<11
    formatSpec='10^-%d is: %1.20f\t';
    fprintf(formatSpec,x1,F1(x1));
    formatSpec='10^-%d is: %1.20f\n';
    fprintf(formatSpec,x1,F2(x1));
    x1=x1+1;
end
%% Q3
clc;clear all;format long e;
L=25;d=0.1;ep=0.8;sig=5.67*10^-8;Q=18405;h=10;Ta=298;Tsurr=298;
a=pi*d*L*h;b=pi*d*L*ep*sig;
c=a*Ta;d=b*Tsurr^4;
F=@(x)(a*x-c+b*x^4-d-Q);

r1=bisectionMethod(F,400,501,1e-10);
formatSpec='Using Bisection Method Root 1 found at:%e\n';
fprintf(formatSpec,r1);

options=optimset('display','iter','tolX',1e-10);
[xc1]=fzero(F,[400 501],options)
formatSpec='Using fZero Method Root 1 found at:%e\n';
fprintf(formatSpec,xc1);

x=[-9000:0.1:9000];
for jj=1:length(x);
    y(jj)=F(x(jj));
end
plot(x,y);  grid on;
%ylim([-30000 30000])
xlabel('x'); ylabel('y');
%% Q4
clc;clear all;format long e;
L=3;E=70*10^9;I=52.9*10^-6;w=15000;
z=w./(120.*E.*I);
a=3.*z.*L.^2;b=7.*z.*L;c=5.*z;d=z./L;
F=@(x)(a.*x.^2-b.*x.^3+c.*x.^4-d.*x.^5);
dF=@(x)(2.*a.*x-3.*b.*x.^2+4.*c.*x.^3-5.*d.*x.^4);
ddF=@(x)(2.*a-6.*b.*x+12.*c.*x.^2-20.*d.*x.^3);
r1=newton(1,dF,ddF,1e-10,100);
formatSpec='Using Newton Method Root 1 found at:%1.10e\nThe Deflection at this root
is:%fm';
fprintf(formatSpec,r1,F(r1));
```

```
options=optimset('display','iter','tolX',1e-10);
[xc]=fzero(dF,1,options)
formatSpec='Using fZero Method Root 1 found at:%1.10e\nThe Deflection at this root
is:%fm';
fprintf(formatSpec,xc,F(xc));

x=[0:0.001:3.5];
for jj=1:length(x);
    y(jj)=F(x(jj));
end

plot(x,y);  grid on;
%ylim([-0.5 0.5])
xlabel('Distance (m)'); ylabel('Deflection(m)');
title('Deflection Curve');
%% Q5a
clc;clear all;format long e;
L=0.02;k=0.5;q=10^6;Ta=100;Tb=200;dx=0.004;A=1;
a=k*A/dx;b=2*a;c=q*A*dx;d=a+b;
e=c+b*Ta;f=c+b*Tb;
% / Method
A=[d -a 0 0 0;
   -a b -a 0 0;
   0 -a b -a 0;
   0 0 -a b -a;
   0 0 0 -a d;
   ];
B=[e;c;c;c;f];
tic;
x1=A\B;
formatSpec='Using Backslash
Method:\nT1=%3.0f\nT2=%3.0f\nT3=%3.0f\nT4=%3.0f\nT5=%3.0f\nComp Time:%fs\n';
fprintf(formatSpec,x1(1),x1(2),x1(3),x1(4),x1(5),toc);
% Inversion Method
tic;
xc=inv(A)*B;
formatSpec='Using Inversion
Method:\nT1=%3.0f\nT2=%3.0f\nT3=%3.0f\nT4=%3.0f\nT5=%3.0f\nComp Time:%fs\n';
fprintf(formatSpec,xc(1),xc(2),xc(3),xc(4),xc(5),toc);
% Tridiagonal Method
tic;
a1=[0 -a -a -a -a];
b1=[d b b b d];
c1=[-a -a -a -a 0];
d1=[e;c;c;c;f];
[xout]=tridiagonal(a1,b1,c1,d1);
formatSpec='Using Tridiagonal
Method:\nT1=%3.0f\nT2=%3.0f\nT3=%3.0f\nT4=%3.0f\nT5=%3.0f\nComp Time:%fs\n';
fprintf(formatSpec,xout(1),xout(2),xout(3),xout(4),xout(5),toc);

F=@(x)(Ta+((Tb-Ta)/L+q*L/2/k)*x-q*x^2/2/k);

x=[0:0.0001:0.02];
for jj=1:length(x);
    y(jj)=F(x(jj));
```

```matlab
end
plot(x,y);  grid on;

xlabel('Distance (m)'); ylabel('Temperature (C)');
title('Temperature Distribution');
hold on
x=0.002:0.004:0.018;
stem(x,x1);
ylim([100 300])
hold off

%% Bisection Method Function
function c = bisectionMethod(F,a,b,e)
c=(a+b)/2;
while abs(F(c))>e
    if F(c)<0&&F(a)<0
        a=c;
    else
        b=c;
    end
    c=(a+b)/2;
end
%% Newton Method Function
function [xout,k]=newton(x,f,fprime,tol,max_iteration)
k=0;
xnew=x;
xold=x+5*tol; % dummy, used only to start the while loop
fprintf(1,'%d %15.10f \n',k,xnew);
while abs(xnew-xold)>tol
 xold=xnew;
 xnew=xold-f(xold)/fprime(xold);
 k=k+1;
 fprintf(1,'%d %15.10f \n',k,xnew);
 if k>=max_iteration
 break;
 end
end
xout=xnew;
%% Tridiagonal Method Function
function [xout]=tridiagonal(a,b,c,d)
N=length(b);
%---forward elimination------------
for k=2:N
multiplier=a(k)/b(k-1);
b(k)=b(k)-multiplier*c(k-1);
d(k)=d(k)-multiplier*d(k-1);
end
%-----back substitution----------
x(N)=d(N)/b(N);
for k=N-1:-1:1
x(k)=(d(k)-c(k)*x(k+1))/b(k);
end
xout=x.';
```