**Homework 2**

Department of Mechanical and Aerospace Engineering

University of California, San Diego

MAE 150

Lawrence Custodio

A08739279

24 January 2013

**Problem #2:**

**(a)** The data appears such that the bacterial population exponentially grows over the observed period of time.

**(b) Command Window Output:**

Linear: f(t)= -5.763 + 4.0306\*t

-5.7630

4.0306

Quadratic: f(t)= 15.8161 + -1.1692\*t + 0.20799\*t^2

15.8161

-1.1692

0.2080

Cubic: f(t)= 10.2902 + 1.51\*t + -0.060463\*t^2 + 0.0071588\*t^3

10.2902

1.5100

-0.0605

0.0072

Exponential: f(t)= 10\*exp(0.10137\*t)

10.0000

0.1014

**(c) Plotted Data:**

Based on appearance, it appears to be that the growth rate of bacteria is best represented by the exponential fit function.

**(d) Command Window Output:**

LINE =

4.0306 -5.7630

QUAD =

0.2080 -1.1692 15.8161

CUBE =

0.0072 -0.0605 1.5100 10.2902

EXP =

0.1014 2.2620

The following coefficients suggest a fitting function as follows:

Which in comparison to the results shown in Part (b) are similar with the exception of the exponential; the exponential fit had to be adjusted for its initial value at (population =10 at t=0), as shown in Part (b).

**(e) Command Window Output:**

RMSLinear =

11.1090

RMSQuad =

5.2939

RMSCubic =

4.8425

RMSExp =

5.1815

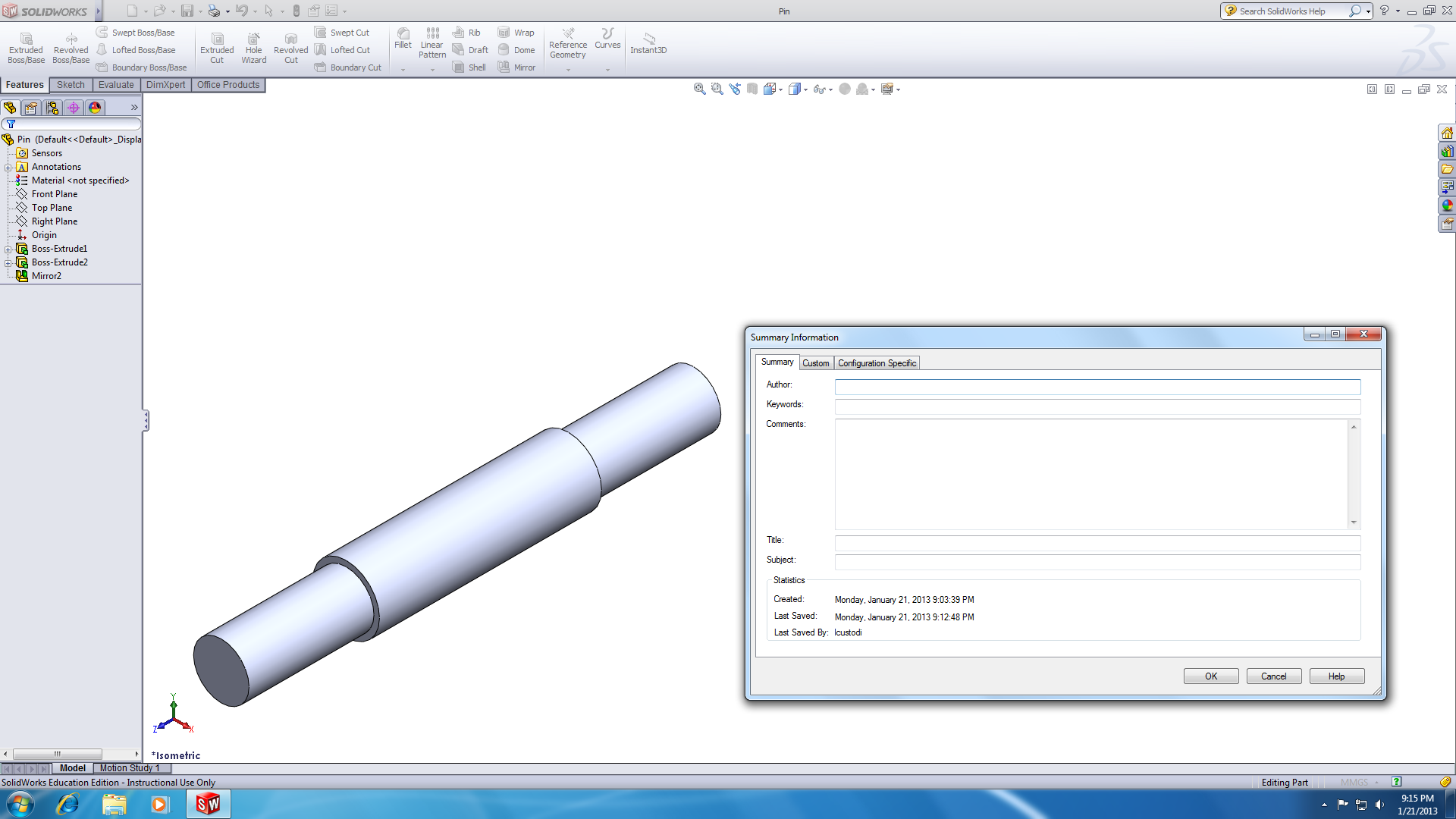
The value of RMS error is directly related to the quality of the fitted function with respect to the actual data; a higher value means a worst quality fit. Results suggest that the best functional fit for the data points is the cubic function:.

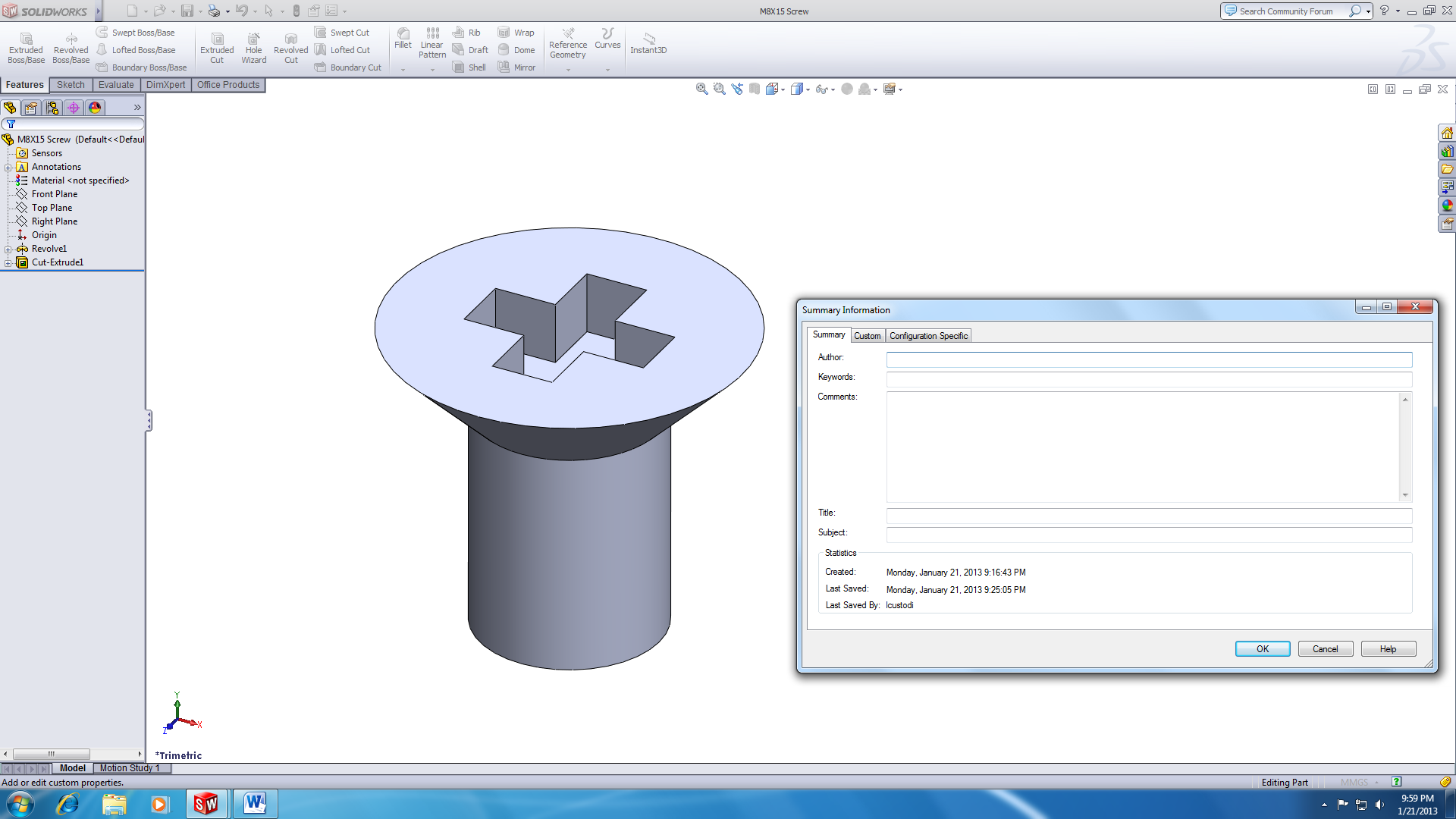
Additionally, the largest RMS Error value was the linear fit, as obviously seen in the graph.

**Problem # 3: Curve plotting**

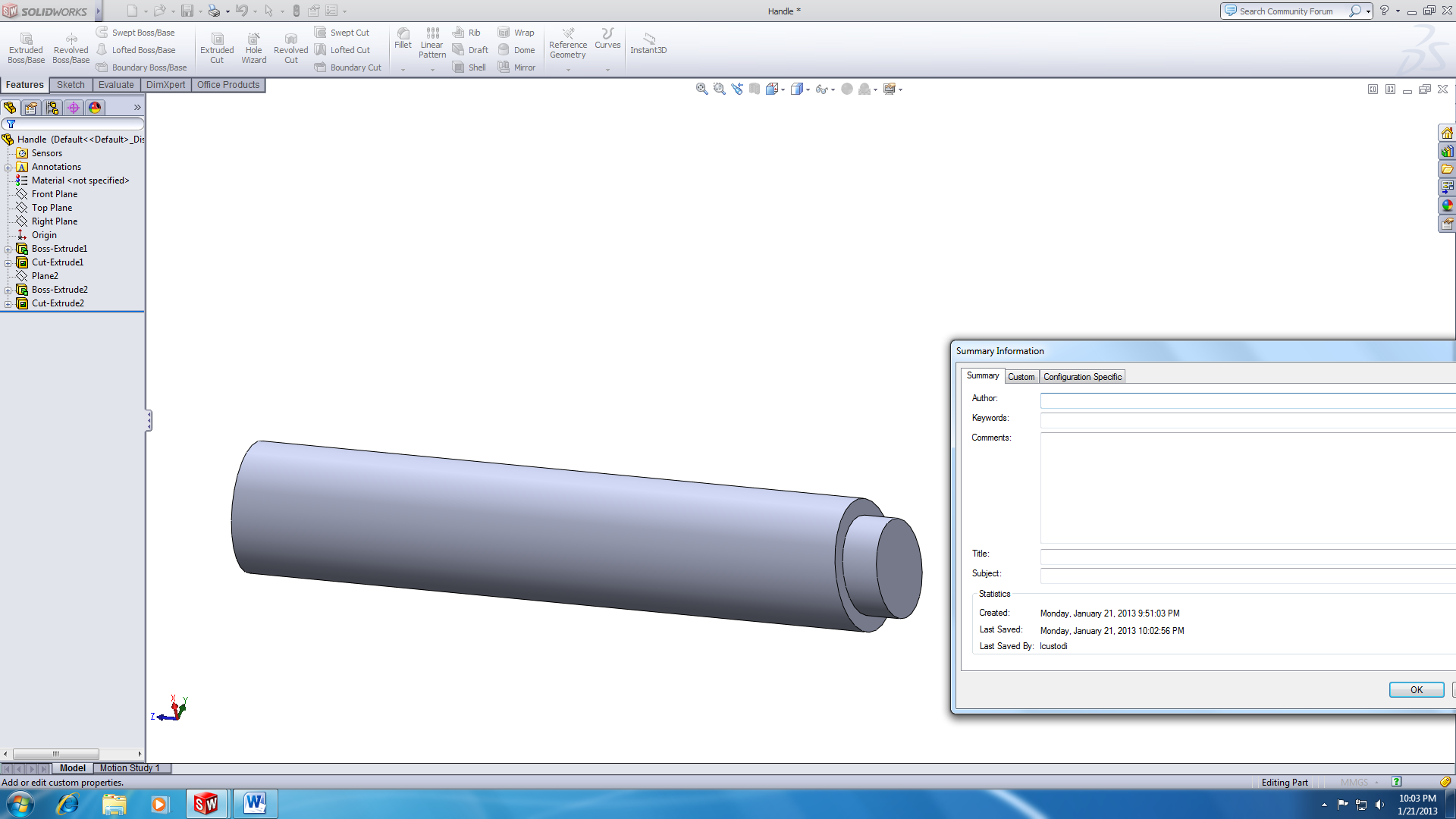
****

**Problem # 4:** SolidWorks

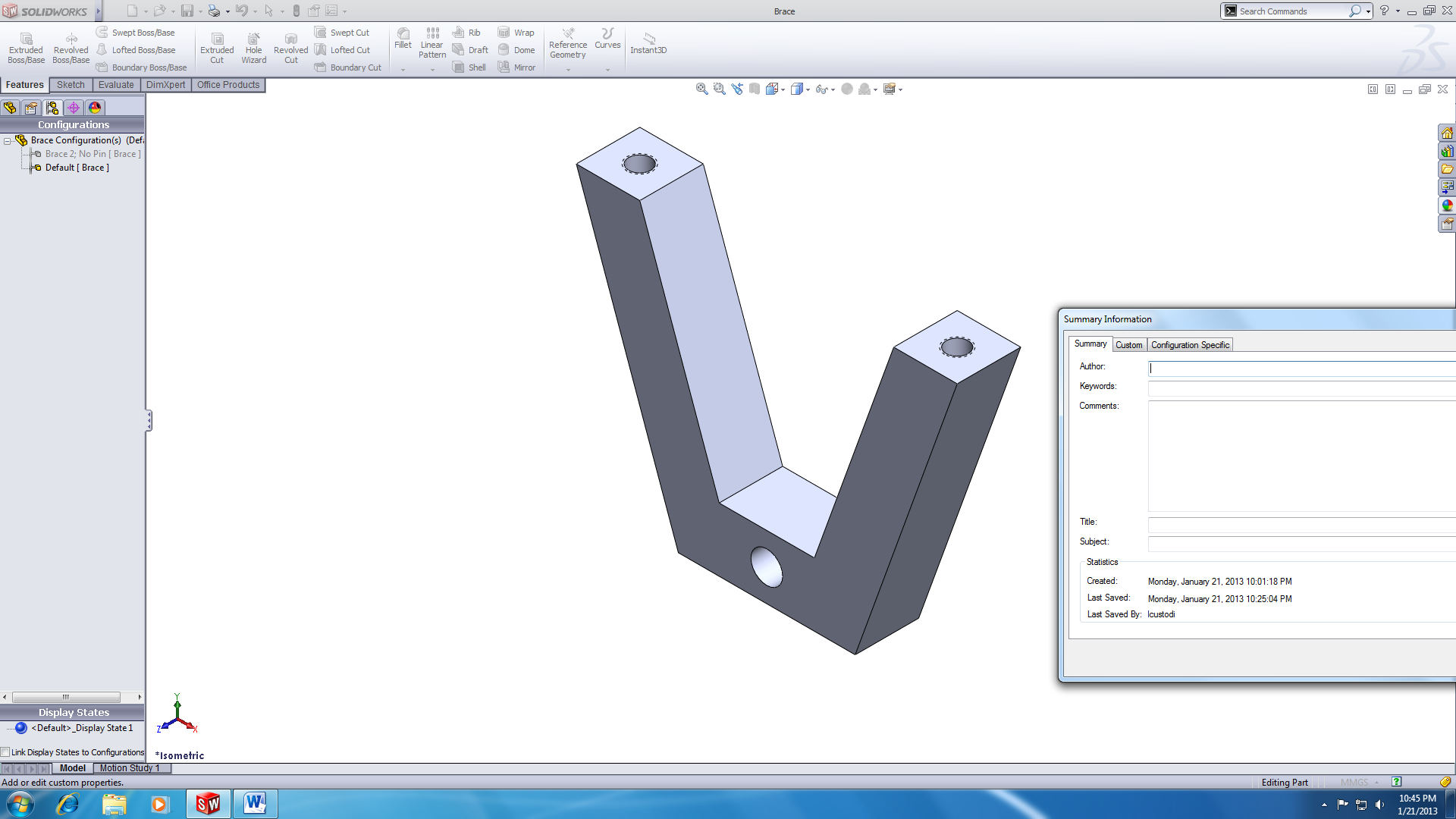
Pin.SLDPRT

M8X15 Screw.SLDPRT

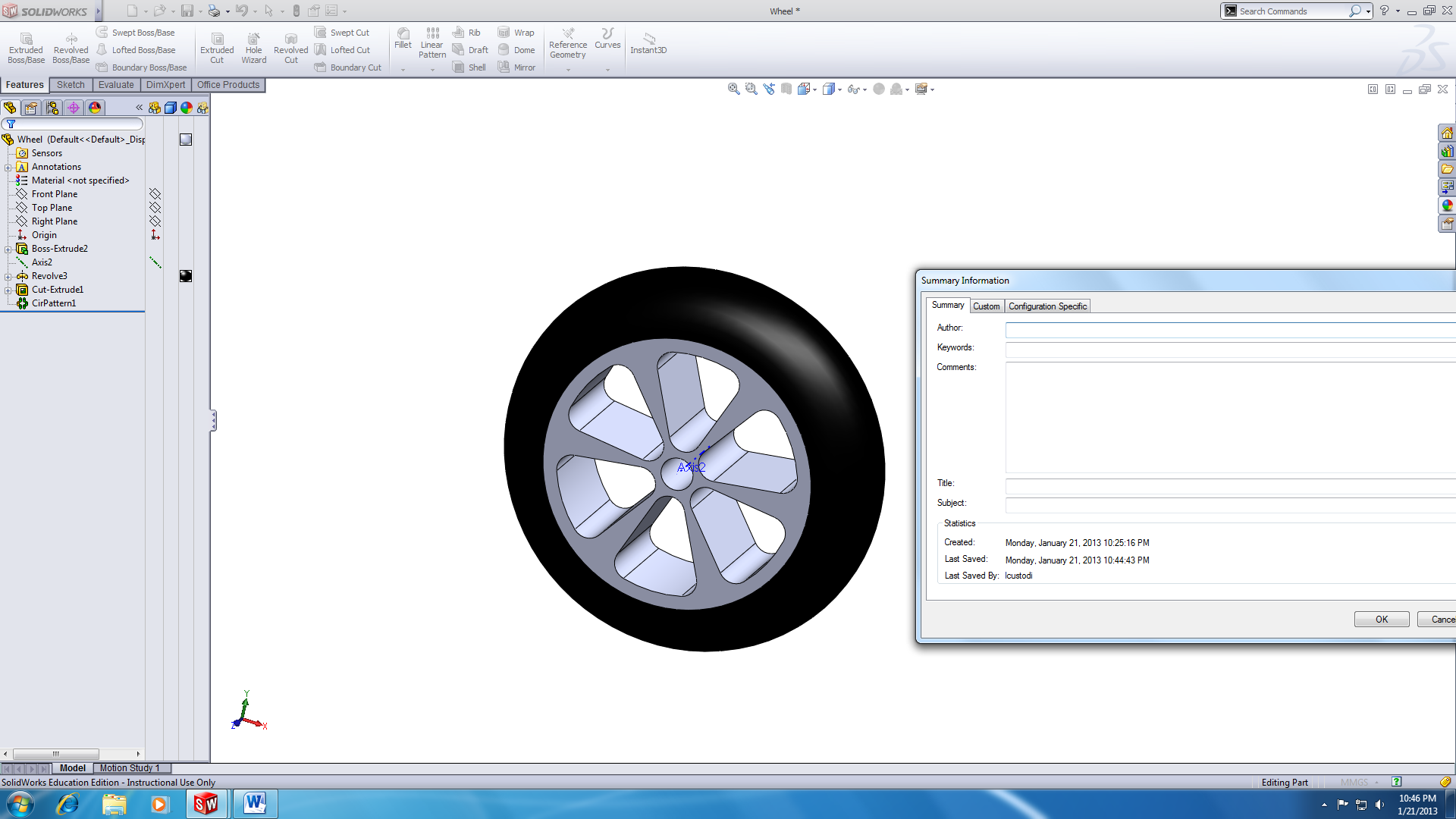
Handle.SLDPRT



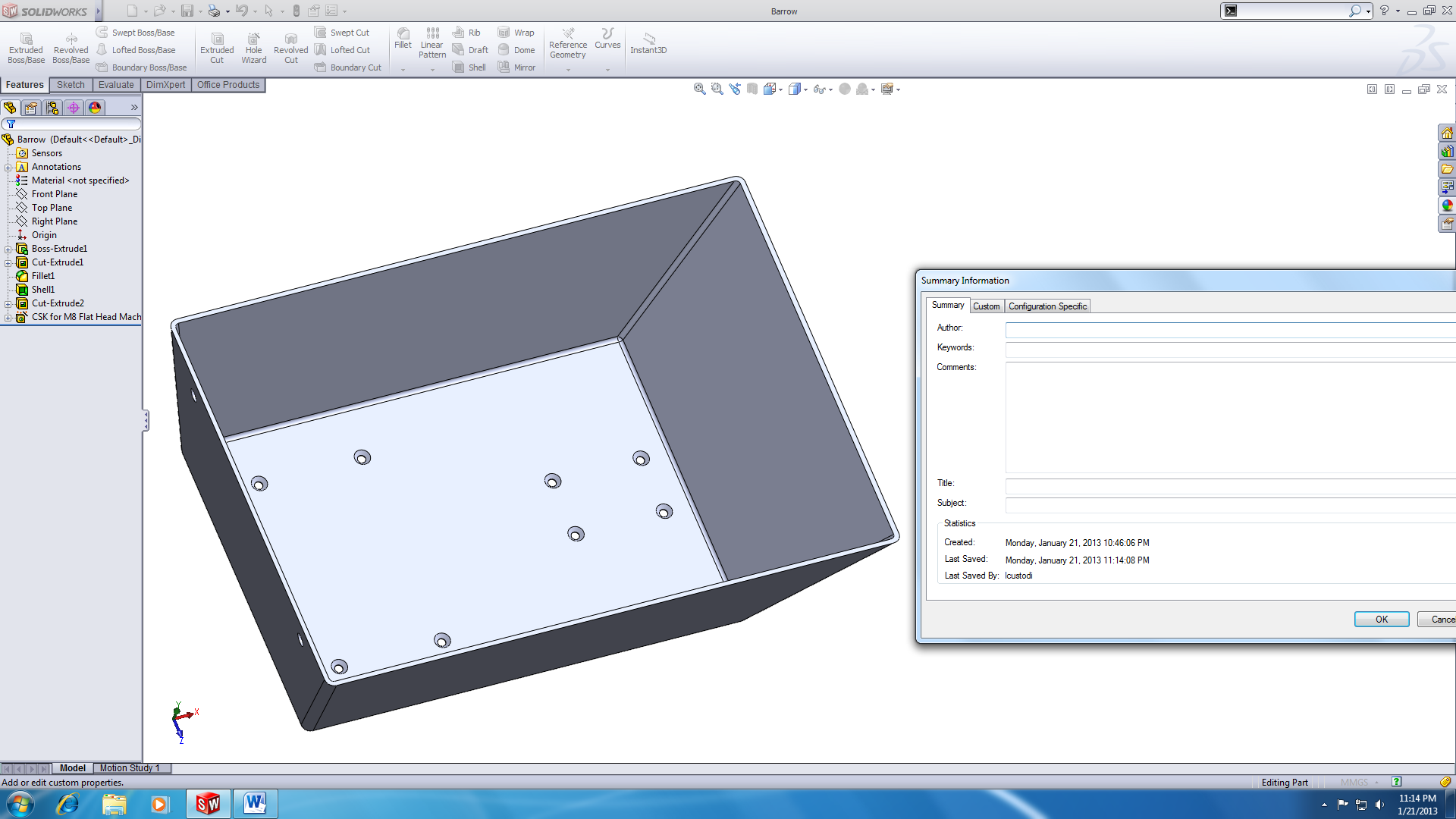
Brace.SLDPRT



Wheel.SLDPRT



Barrow.SLDPRT



**Appendix: MATLAB Code**

**Problem 2**

%Lawrence Custodio: Problem 2

clc;clear all;close all;

load('data.txt')

time = data(:,1);

pop = data(:,2);

M = length(data);

plot(time,pop)

hold on

%Linear Fit

nlinear = 1;

%A matrix: for A[i,j] = sigma[1,M](x.^(i+j-2); e.g.A(4,1) = sum(x.^3)

for rowA1=1:nlinear+1

for columnA1=1:nlinear+1

A1(rowA1,columnA1) = sum(time.^(rowA1+columnA1-2));

end

end

%b-vector follows notation: sigma[1,M](y.\*(x^(i+1))

for rowb1=1:nlinear+1

x1 = time.^(rowb1-1);

b1(rowb1,1)=sum((pop).\*(x1));

end

%{a} = inv[A]\*{b}

a1 = A1\b1;

%Equation

y1 = a1(1) + a1(2).\*time;

disp(['Linear: f(t)= ' num2str(a1(1)) ' + ' num2str(a1(2)) '\*t']);

disp([a1]);

plot(time,y1,'y')

%Quadratic Fit

nquad = 2;

%A matrix:

for rowA2=1:nquad+1

for columnA2=1:nquad+1

A2(rowA2,columnA2) = sum(time.^(rowA2+columnA2-2));

end

end

%b vector:

for rowb2=1:nquad+1

x2 = time.^(rowb2-1);

b2(rowb2,1)=sum((pop).\*(x2));

end

%Solving for constants:

a2 = A2\b2;

y2 = a2(1) + a2(2).\*time + a2(3).\*(time.^2);

disp(['Quadratic: f(t)= ' num2str(a2(1)) ' + ' num2str(a2(2)) '\*t + ' num2str(a2(3)) '\*t^2']);

disp([a2]);

plot(time,y2,'r')

%Cubic Fit (follows same logic as quadratic, but n =3)

ncubic = 3;

for rowA3=1:ncubic+1

for columnA3=1:ncubic+1

A3(rowA3,columnA3) = sum(time.^(rowA3+columnA3-2));

end

end

for rowb3=1:ncubic+1

x3 = time.^(rowb3-1);

b3(rowb3,1)=sum((pop).\*(x3));

end

a3 = A3\b3;

y3 = a3(1) + a3(2).\*time + a3(3).\*(time.^2) +a3(4).\*(time.^3);

disp(['Cubic: f(t)= ' num2str(a3(1)) ' + ' num2str(a3(2)) '\*t + ' num2str(a3(3)) '\*t^2 + ' num2str(a3(4)) '\*t^3']);

disp([a3]);

plot(time,y3,'g')

%Exponential Fit: y = Cexp^(k\*t)

%Linearly modelled at lny = ln(C) + k\*t

%A-matrix for exp. fit = linear fit A-matrix

Aexp = A1;

%b-vector: note that sum(population)->sum(log(population))

bexp(1,1) = sum(log(pop));

bexp(2,1) = sum(log(pop).\*time);

aexp = Aexp\bexp;

aexp(1) = 10; %Given that population @ t=0 is 10

k = aexp(2);

%Equation

yexp = aexp(1)\*exp(k.\*time);

disp(['Exponential: f(t)= ' num2str(aexp(1)) '\*exp(' num2str(k) '\*t)']);

disp([aexp]);

plot(time,yexp,'--k')

%Plotting

xlabel('Time (sec)')

ylabel('Bacteria Population (#)')

title('Bacterial Population Growth Rate vs. Time ')

legend('Experimental Data','Linear Fit','Quadratic Fit','Cubic Fit','Exponential Fit')

%Part D. Polyfit

LINE = polyfit(time,pop,nlinear) %Linear function (n=1)

QUAD = polyfit(time,pop,nquad) %Quadratic (n=2)

CUBE = polyfit(time,pop,ncubic) %Cubic (n=3)

EXP = polyfit(time,log(pop),1) %Exponential

%Part E. RMS Error

%Method:

%(1)Define a variable = (y fitted - actual y value)

%(2) sqrt((sum(var))^2/length(y))

%Linear

d1=(pop-y1).^2;

RMSLinear = sqrt(sum(d1)/length(pop))

%Quadratic

d2=(pop-y2).^2;

RMSQuad = sqrt(sum(d2)/length(pop))

%Cubic

d3=(pop-y3).^2;

RMSCubic = sqrt(sum(d3)/length(pop))

%Exponential

dexp=(pop-yexp).^2;

RMSExp = sqrt(sum(dexp)/length(pop))

**Problem 3**

clc;clear;close all;

%Given parameters

%NOTE: units in degrees(beta and theta-interval) are converted to rad

beta = 90\*pi/180; %radians

d= 15; %mm

theta=1:1:90;

trad=theta.\*pi/180; %in radians

%Components of displacement equation:

i=(10/(beta^3))\*(trad.^3);

ii=(15/(beta^4))\*(trad.^4);

iii=(6/(beta^5))\*(trad.^5);

%Displacement equation:

y = d\*(i-ii+iii);

ome=1; %Given value for omega

%Components of velocity equation:

di=((30\*ome)/(beta^3))\*(trad.^2);

dii=((60\*ome)/(beta^4))\*(trad.^3);

diii=((30\*ome)/(beta^5))\*(trad.^4);

%Velocity equation:

dy = d\*(di-dii+diii);

%Components of acceleration equation:

d2i=((60\*ome^2)/(beta^3))\*(trad);

d2ii=((180\*ome^2)/(beta^4))\*(trad.^2);

d2iii=((120\*ome^2)/(beta^5))\*(trad.^3);

%Acceleration equation:

d2y = d\*(d2i-d2ii+d2iii);

%Plotting

plot(trad,y)

hold on

plot(trad,dy,'r--')

plot(trad,d2y,'g-.')

xlabel('Cam angle(rad)')

legend('Displacement','Velocity','Acceleration')