Roller Follower Cam Design Program

Kalin D. Gibbons
Student
Boise State University
Boise, Idaho 83725
KalinGibbons@u.boisestate.edu

Braden King
Student
Boise State University
Boise, Idaho 83725
BradenKing@u.boisestate.edu

CONTENTS

1 Introduction **2**
Objectives 3

2 Theory of Cam Design **3**
Prime Circle Radius R_b 3
Eccentricity ε 3
Radius of Curvature ρ 3
Design Process 4

3 Programming Results **4**

4 Discussion and Conclusions **7**

A main.m **8**

B main.m **9**

C main.m **10**

D main.m **11**

E getInput.m **12**

F timingDiagram.m **13**

G svaj.m **14**

H plotProfile.m **15**

I camChecker.m **16**

J Standard Cam Motion Equations **17**
J.1 Cycloidal equations 17
J.2 Simple harmonic equations 18

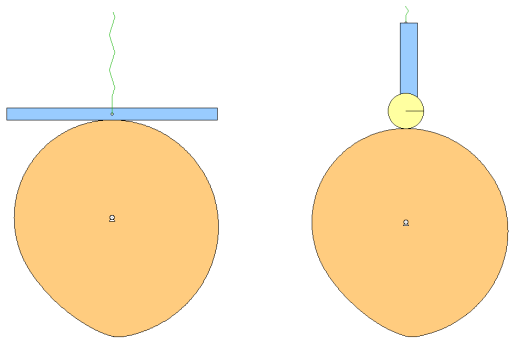
List of Figures

1 Simple Cams 2
2 Cam Definitions 3
3 Cam Timing Diagram 4
4 svaj - pchip() 4
5 pchip() cam profile 4
6 Control equation svaj diagrams 5
7 svaj - spline() 6
8 spline() cam profile 6
9 svaj - altered spline() 6
10 velspline() cam profile 6

1 Introduction

A cam is a device that converts rotational motion into translational movement. A cam assembly consists of a rotating cam and a translating follower. There are many types of followers in use and some of their names are

- Flat-Faced Followers
- Roller Followers



(a) Flat-faced follower (b) Roller follower

Fig. 1: Examples of the two most simple follower types. A roller follower has the benefit of being able to move through a concave cam profile

- Offset Roller Followers
- Mushroom Followers
- Knife Edge Followers
- Oscillating Followers

The most simple of which, from an analysis perspective are the flat-faced and roller followers. A flat-faced follower has the disadvantage that it is unable to traverse a concave cam segment, but the force applied to the cam is always in one direction, which is beneficial when analyzing the forces developed on the bearings holding a shaft in place. The roller followers curved surface allows the point of contact to occur at an angle from the desired axis of translation, which puts the follower member under bending stress. This *pressure angle* should be minimized to increase the life of the cam.

They are used extensively in the engineering field for specifying the locations or velocities of mechanical devices, as well as synchronizing the timing of a mechanical assembly. The two most used methods of cam design are called *critical extreme position* (CEP) or *critical path motion* (CPM). When designing the cams controlling the pistons within an automobile engine, one would typically use CEP design techniques. Control systems in manufacturing plants are an excellent example of the devices that benefit from CPM design. This is because manufacturing throughput is increased by having the assembly robots move with the product as they perform their operations, which makes matching conveyor velocity a critical design parameter.

With either choice of design technique, cam life is increased by minimizing sudden accelerations of the follower. This is done by ensuring that the first and second derivatives of follower position are continuous functions. This is articulated in the *Fundamental Law of Cams*. This also requires that the 4th derivative of the cam position function, known as the *jerk* is bounded [1]. This law of cam design, coupled with both CEP and CPM methods using piecewise position or velocity requirements, makes computer software the most suitable tool for cam design. This report was produced after carrying out an investigation of the usability of Matlab computation software as a design tool for producing cam profiles.

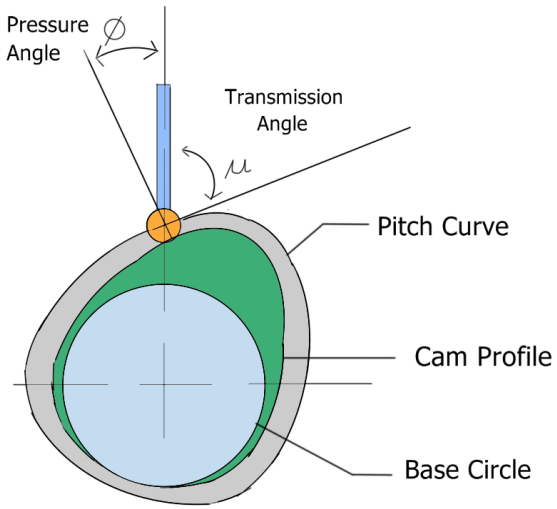


Fig. 2: Terminology used in designing a roller follower cam system

Objectives

The goal of this project was to create a Matlab program that would aid the user in generating a data set of a working cam profile which ensured that the fundamental law of cams was obeyed and would allow for smooth follower motion. A secondary objective was to investigate the feasibility of Matlab's various interpolation functions in generating the profile of a cam. The program was intended to be capable of the following:

1. Prompt the user to input angular displacement and position vectors
2. Plot the *cam timing diagram* generated from the users vectors
3. Generate kinematic plots for the follower in the form of an *svaj diagram*
4. Check that the base radius was large enough to allow smooth follower motion
5. Animate several cam profiles for varying base radii
6. Plot a final cam profile based on the minimum calculated base radius

In exploring the best way to ensure the fundamental law of cams was obeyed, the following comparison svaj diagrams would be created:

1. Cycloidal
2. Simple harmonic
3. Matlab *pchip()*
4. Matlab *spline()*

Cycloidal and simple harmonic equations represented the control plots from which the others could be compared as they were created using published standard [2] cam motion equations.

2 Theory of Cam Design

When designing a roller follower type cam, there are four design parameters of which you have control. They are

1. Prime Circle Radius (R_p)
2. Pressure angle (ϕ)
3. Eccentricity (ε)
4. Radius of curvature (ρ)

of which all except eccentricity may be seen in Fig.2.

Prime Circle Radius R_b

Increasing R_b will decrease the pressure angle while increasing the radius of curvature. Both of these are beneficial and will be discussed in subsequent sections. The equation for the prime circle radius is

$$R_p = R_f + R_b \tag{1}$$

Pressure Angle ϕ

In designing a cam, it is suggested that the pressure angle be kept below 30° for a translating follower [3]. The formula used to calculate the pressure angle is

$$\phi = \tan^{-1}\left(\frac{v-\varepsilon}{s+\sqrt{R_p^2-\varepsilon^2}}\right) \tag{2}$$

Eccentricity ε

Eccentricity is the offset between the follower and cam axes. It may be used to decrease the pressure angle in a disproportionate way. This is useful when the magnitude of pressure angles is skewed more when a cam is undergoing a position rise or fall.

Radius of Curvature ρ

Radius of curvature is the parameter that controls undercutting, which occurs when the base circle of the cam is too small to accommodate the required displacement equation. When undercutting occurs a normally smooth cam profile will appear to have a segment that looks like a balloon has been tied off, which is impossible to machine. The radius of curvature is a function of R_p, s, a, and v. It is governed by

$$\rho = \frac{\left[(R_p+s)^2+v^2\right]^{3/2}}{(R_p+s)^2+2v^2-a(R_p+s)} \tag{3}$$

Where $R_p = R_b + R_f$. Increasing the radius of curvature will help to avoid undercutting, but will also make the machine larger and heavier, which is undesirable in engineering. Cams are generally designed to be as small as is possible to avoid undercutting.

In addition to the mentioned pressure angle requirements, another rule to follow when designing a cam is that the minimum radius of curvature is at least 3 times larger than the follower radius. In equation form

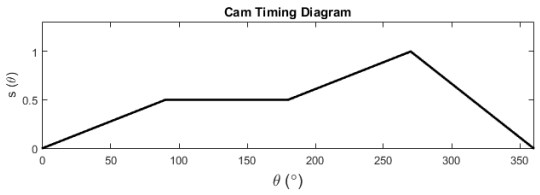


Fig. 3: A sample cam timing diagram produced by Matlab

$$|\rho_{min}| \geq 3R_f \tag{4}$$

Once these rules are followed, the cam profile may be generated and inspected. The profiles are generated in xy coordinates, which allow the machining of the cam using a CNC machine. The equations used to generate the pitch curve are

$$x = \cos(\lambda)\sqrt{(d+s)^2+\varepsilon^2} \tag{5}$$

$$x = \sin(\lambda)\sqrt{(d+s)^2+\varepsilon^2} \tag{6}$$

where λ and d are defined by

$$\lambda = (2\pi-\phi) - \tan^{-1}\left(\frac{\varepsilon}{d+s}\right) \tag{7}$$

$$d = \sqrt{R_p^2-\varepsilon^2} \tag{8}$$

Design Process

The general design process for generating a cam is broken down into 6 steps. They are

1. Produce a simple cam timing diagram
2. Generate position functions for each domain
3. Differentiate the functions 3 times, using boundary conditions as needed
4. Produce svaj diagrams to ensure continuity in the position and velocity functions
5. Generate a plot of pressure angle, and vary the design parameters until its maximum magnitude is below 30°
6. Generate the cam profile plot, increasing the pitch circle radius until undercutting is eliminated

These steps are best left to a spreadsheet or some other software, such as Matlab. A relatively simple program can be made to generate the majority of these plots, and certain design parameters may be automatically sized in an iterative fashion.

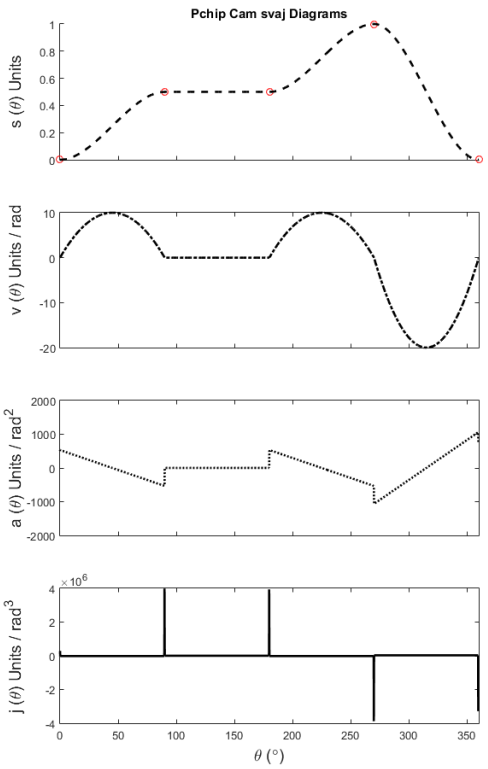


Fig. 4: svaj diagram produced using the pchip() function to interpolate over the entire domain.

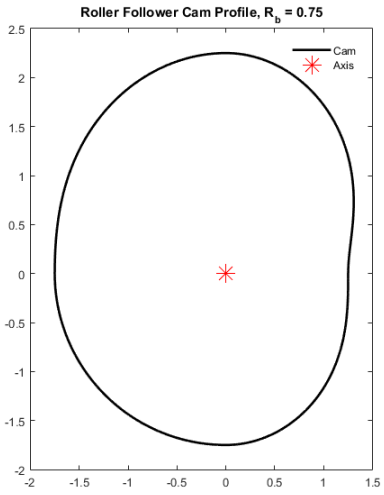
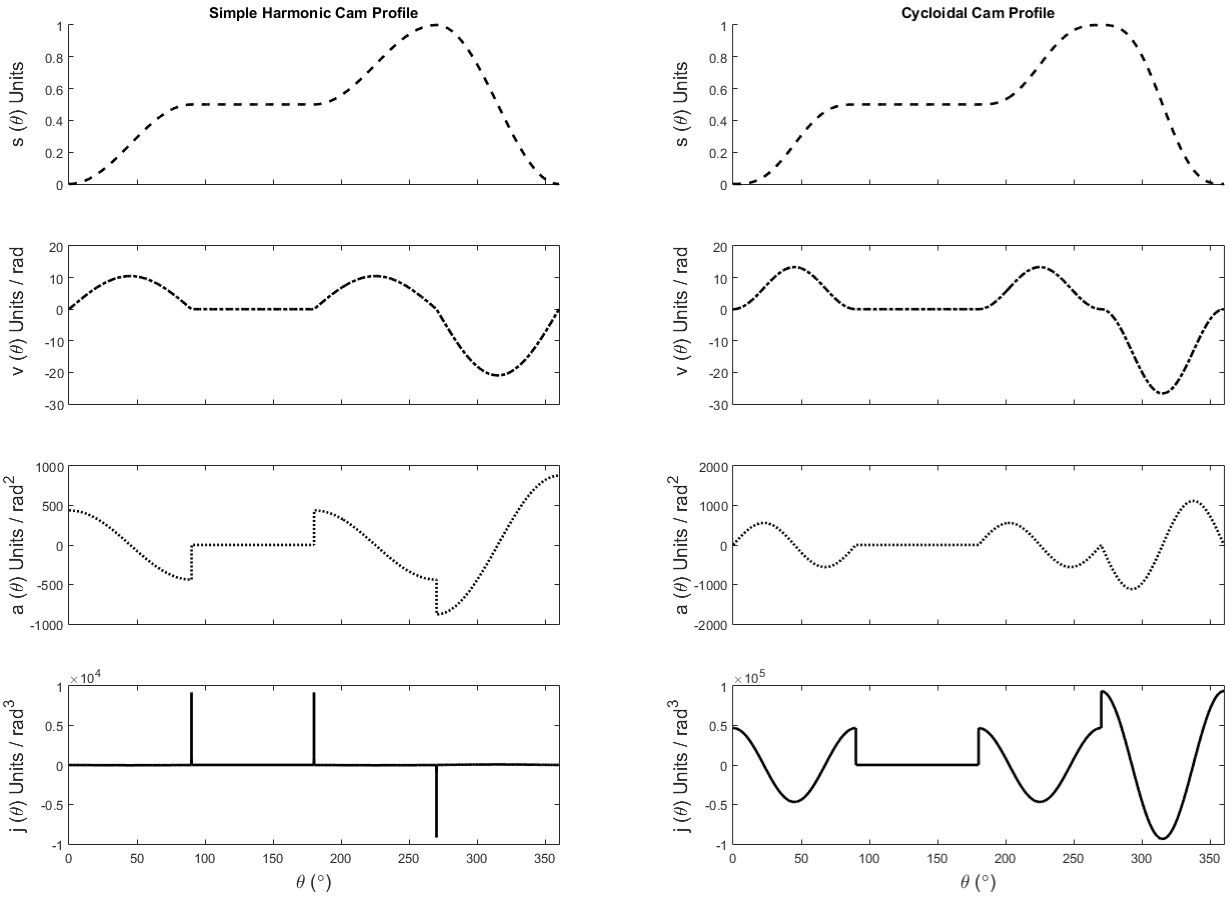


Fig. 5: Cam profile generated using the pchip() function

3 Programming Results

The plots from the various interpolation methods are included here. Each method produced a svaj diagram and cam profile based on the timing diagram seen in Fig.3.



(a) Simple Harmonic

(b) Cycloidal

Fig. 6: svaj diagrams created using standard cam design equations from reference [2]

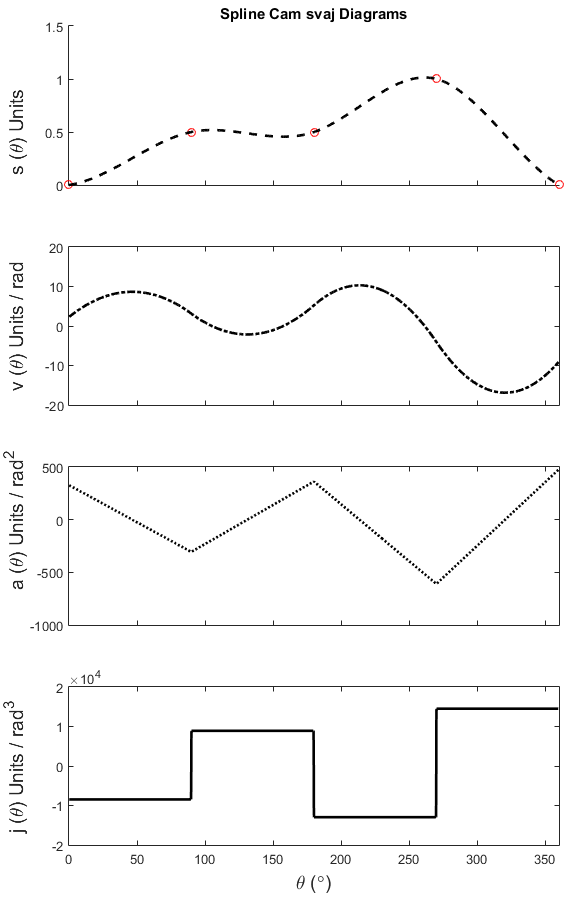


Fig. 7: svaj diagram produced using the spline() function. While the velocity function is perfectly continuous, the dwell portion of the position diagram has been lost

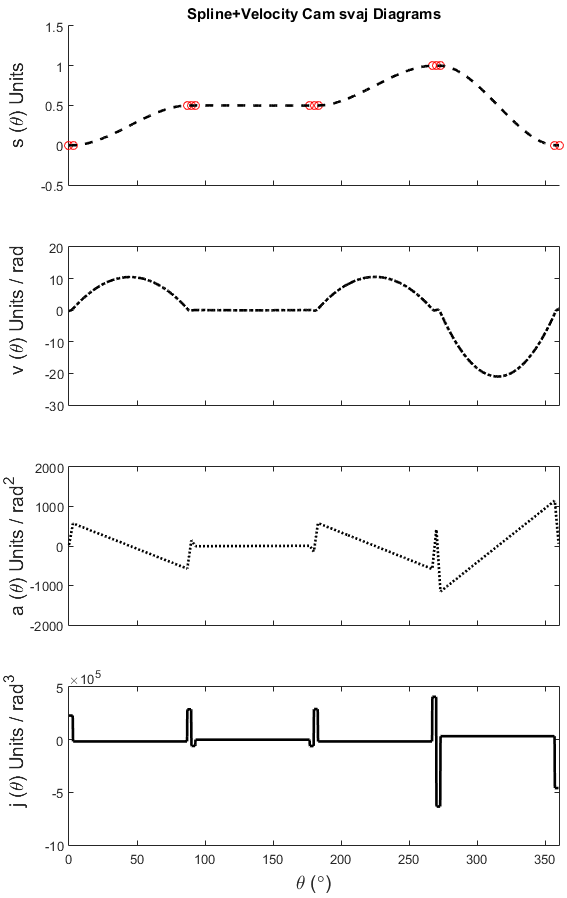


Fig. 9: svaj diagram produced by forcing velocity requirements into data points before applying the spline() function

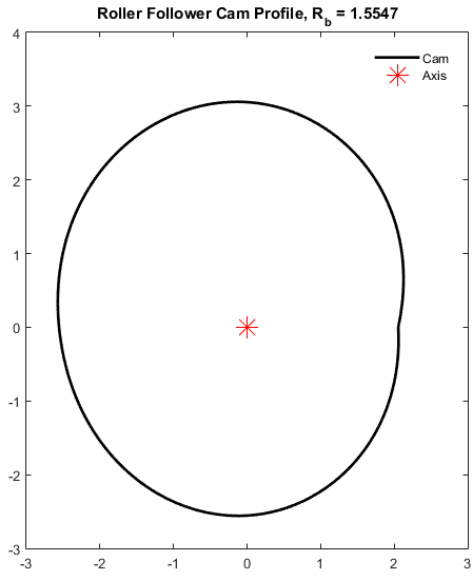


Fig. 8: Cam profile generated using the spline() function

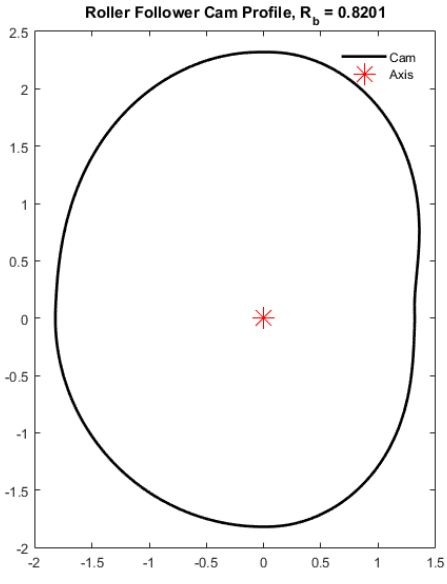


Fig. 10: Cam profile generated using the velspline() function

4 Discussion and Conclusions

When designing a cam, certain considerations must be taken. To minimize mechanical wear and stress on machine elements, it is very desirable to have as small of acceleration and jerk as possible. Also imperative in high-speed cam situations is to have continuous functions for velocity and acceleration across boundaries of different portions of the cam. These requirements are generally achieved through use of certain standard cam motion equations, including simple harmonic motion and cycloidal motion, among others. To verify the validity and effectiveness of the cam profile design code outlined in this paper, the resulting displacement, velocity, acceleration, and jerk diagrams were compared to ones derived from harmonic and cycloidal motion equations for a set of relatively simple rise data with standard cam features: multiple rises, a large fall, and a dwell in Appendix J.

Important conclusions can be drawn as to the characteristics of each interpolation method through comparison with standard cam motions. When looking at the shape of the displacement diagrams in Figs. [4 - 10], it is clear that the basic Spline technique displayed in Fig. 7, while having the best acceleration and jerk profiles, does not match the requirement for a dwell between θ_1 and θ_2, as is visible in both Figs 6a and 6b. The Pchip function, while meeting this design criteria, , has discontinuities in the acceleration function that result in infinite jerk values, although in Fig. 4, they do not appear infinite due to the limited precision over which the derivatives were evaluated. Infinite jerk creates significant wear on a mechanism, which is the primary reason that sinusoidal functions are used rather than polynomial functions.

This leaves the enhanced Spline interpolation in which small data modifications were made to enforce dwells in the rise data given by the user. It has a profile very similar to that of derived cycloidal cam motion functions (see Equations J.1 and Fig. 6b). It also has the lowest overall jerk and acceleration values of the methods tested, and thus is determined to be the most effective method tested for cam profile design.

Often when cams are being developed, polynomials of an 8th or higher order are used. The functions derived in this program, using MatLab's build in interpolation algorithms, are only of the 3rd order. This means that once a third derivative is taken, as is done when finding the jerk in a cam profile, the resultant curve reveals discontinuities. A better process for developing a cam profile would be to create a different version of the Pchip interpolation formula that has a higher order. This would allow for the interpolation algorithm to set higher derivatives of the piecewise sections of the function to be continuous, thereby allowing for a more effective automated cam design. This, however, was beyond the scope of this program design and as such, not used in this report.

References

[1] Guarino, J., 2015. Cam lecture 1 - the road to cam conciousness. Lecture Notes. Word Doc, Accessed 2015.

[2] John J. Uicker, J., Pennock, G. R., and Shigley, J. E., 2011. *Theory of Machines and Mechanisms*. Oxford University Press.

[3] Guarino, J., 2015. Cam profile design: Translating followers. Lecture Notes. Word Doc, Accessed 2015.

A main.m

```
%% Roller Follower Cam Design Program
%
%  Final Project for MATH 365
%
%  Prepared for: Professor Grady Wright
%
%  Authored by: Kalin Gibbons
%               Braden King

clear variables; close all; clc
%% Define global variables
% Just a few vars for editing the plot parameters
global myWidth myFormat myFontSize myXLim
myWidth = 2;
myFormat = '%3.2f';
myFontSize = 14;
myXLim = [0,360];

%% Define the Cam CEP values and design constraints
%
welcome;

% This allows to get user inputs for rise,theta,vel sets
%     [theta,rise,Vels,w] = getInput;

% Example set 1
%     theta = [0, 30,40, 50,60, 90, 110, 180, 270, 360];
%     rise = [0, 1, 1, 1.5, 1.5, 1, 1, 2, 2.3, 0];
%     Vels = [0, 0, 0,   0,   0, 0, 0, 0,   0, 0];

%  Example set 2
    theta = [0 90 180 270 360];
    rise = [0 .5 .5 1 0];
    Vels = [0 0 0 0 0];

% This sets the design constraints for all cams
    rFollow = 0.5; %Roller-follower radius
    rBase = .5; %Initial base radius (checked in CamChecker)
    ecc = 0; %
    myFactor = 0.01;

% Choose what the rotation speed of the cam is
w = 200*2*pi/60;

% Choose how many points we want to interpolate over
targetPoints = 3600;

%% Plot the desired cam timing diagram
%
fig1 = timingDiagram(theta,rise);

%% Plot the svaj curves using Pchip interpolation
%
[t,s,v,a,j,fig2] = svaj(theta,rise,targetPoints,w);

%% Plot the svaj curves using Spline interpolation
```

B main.m

```
%
[t2,s2,v2,a2,j2,fig3] = svajSpline(theta,rise,targetPoints,w);

%% Plot the svaj curves using Spline interpolation with defined velocities
%
[theta2, rise2] = AddSlopes(theta,rise,theta,Vels,w);
[t3,s3,v3,a3,j3,fig4] = svajSplineV(theta2,rise2,targetPoints,w);

%% Development of a cam profile
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calculate minimum base radius (Roller Follower)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pchip cam evaluation
[rBase1, rPitch1, rCurve1] =...
    CamChecker(rBase, rFollow, ecc, myFactor, s, v, a, t);
fprintf(['The minimum base radius for a Pchip-derived cam is '...
    num2str(rBase1) ' Units.\n'])
pause;
% Show Pchip's minimum allowable cam size
plotProfile(rBase1,rFollow,t,s,ecc);
daspect([1,1,1]);
savefig PchipCam;
pause;

% Spline cam evaluation
[rBase2, rPitch2, rCurve2] =...
    CamChecker(rBase, rFollow, ecc, myFactor, s2, v2, a2, t2);
fprintf(['The minimum base radius for a Spline-derived cam is '...
    num2str(rBase2) ' Units.\n'])
% Show Spline's minimum allowable cam size
plotProfile(rBase2,rFollow,t2,s2,ecc);
daspect([1,1,1]);
savefig SplineCam;
pause;

% Spline+Velocities cam evaluation
[rBase3, rPitch3, rCurve3] =...
    CamChecker(rBase, rFollow, ecc, myFactor, s3, v3, a3, t3);
fprintf(['The minimum base radius for a Spline-derived cam is '...
    num2str(rBase3) ' Units.\n'])
% Show VelSpline's minimum allowable cam size
plotProfile(rBase3,rFollow,t3,s3,ecc);
daspect([1,1,1]);
savefig VelSplineCam;

%% Comparison of Pchip, Spline, and Spline+Velocities profiles
%

close all;

% Load saved Cam Profiles
PchipP=hgload('PchipCam.fig');
SplineP=hgload('SplineCam.fig');
VelSplineP=hgload('VelSplineCam.fig');
```

C main.m

```
% Prepare subplots
ProfileFig = figure('units','normalized','outerposition',[0 0 1 1]);
h(1)=subplot(1,3,1);
daspect([1 1 1])
title('Pchip Cam Profile')
h(2)=subplot(1,3,2);
daspect([1 1 1])
title('Spline Cam Profile')
h(3)=subplot(1,3,3);
daspect([1 1 1])
title('Spline+Velocities Cam Profile')

copyobj(allchild(get(PchipP,'CurrentAxes')),h(1));
copyobj(allchild(get(SplineP,'CurrentAxes')),h(2));
copyobj(allchild(get(VelSplineP,'CurrentAxes')),h(3));
pause; % When user is done studying profiles, program moves on to next step

%% Comparison of SVAJ diagrams for 3 interpolation methods
%
close all;
% Load saved svaj diagrams (made fullscreen)
PchipC=load('Pchip.fig');
SplineC=load('Spline.fig');
VelSplineC=load('VelSpline.fig');
pause;

%% Extra code for comparison of examples
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Comparison with established cam profile techniques using Example 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plot Harmonic Cam Profile for Example 2
close all;
load HarmonicS
plotProfile(rBase1,rFollow,th,s,ecc);
daspect([1,1,1]);
savefig HarmonicCam;

% Plot Cycloidal Cam Profile for Example 2
close all;
load CycloidalS
plotProfile(rBase1,rFollow,th,s,ecc);
daspect([1,1,1]);
savefig CycloidalCam;

% Open the derived Kinematic Plots for Example 2
% Lets us compare our methods to cycloidal and harmonic profile cam profile
% techniques for sample data set

close all
open Harmonic.fig
open Cycle.fig
open('Pchip.fig')
pause
open('Spline.fig')
pause
```

D main.m

```
open('VelSpline.fig')

% Open the profiles for Example 2
close all;

% Load saved Cam Profiles
PchipP=hgload('PchipCam.fig');
SplineP=hgload('SplineCam.fig');
VelSplineP=hgload('VelSplineCam.fig');

% Prepare subplots
ProfileFig = figure('units','normalized','outerposition',[0 0 1 .5]);
h(1)=subplot(1,3,1);
daspect([1 1 1])
title('Pchip Cam Profile')
h(2)=subplot(1,3,2);
daspect([1 1 1])
title('Spline Cam Profile')
h(3)=subplot(1,3,3);
daspect([1 1 1])
title('Spline+Velocities Cam Profile')

copyobj(allchild(get(PchipP,'CurrentAxes')),h(1));
copyobj(allchild(get(SplineP,'CurrentAxes')),h(2));
copyobj(allchild(get(VelSplineP,'CurrentAxes')),h(3));
pause; % When user is done studying profiles, program moves on to next step
```

Published with MATLAB® R2015a

E getInput.m

```
function [ theta, rise, vels, w ] = getInput(  )
%GETINPUT prompts the user for arrays of values
%        theta = set of angles for which position and velocity values are
%                defined
%        rise = folleo
%theta = [0, 30, 50, 90, 110, 180, 270, 360];%N-1 segments
%rise = [0, 1, 1.5, 1, 1, 2, 2.3, 0];
exampleString = ...
    [...
    'Please enter a matrix describing your cam \n',...
    'requirements in order as ordered pairs.\n',...
    'The ordered pairs should have degree angles in ascending order'...
    'as the\n',...
    '1st row dimension afollower displacement as the 2nd, and'...
    'follower velocity as the 3rd\n',...
    '\nExample:\n\n',...
    '[0,90,180,360;0,1,4,5;0,1,0,0] <enter>'...
    ];
flag = 1;
fprintf(exampleString);
while flag == 1

    promptString ='What is your array of angle values? \n\n';
    theta = input( promptString );
    promptString ='What is your array of displacement values? \n\n';
    rise = input( promptString );
    promptString ='What is your array of velocity values? \n\n';
    vels = input( promptString );
%
%     promptString ='What is your array of velocity values? \n\n';
%     vels = input( promptString );
%     promptString ='What is your array of velocity values? \n\n';
%     vels = input( promptString );

    warnString = [...
        'Warning! Angles must be between 0 and 360 degrees.\n',...
        'Your input values have been reset.\n'...
        ];

    if min(theta) ~= 0
        fprintf(warnString)
    elseif max(theta) > 360
        fprintf(warnString)
    else

        flag = 0;
    end
end

end
```

Published with MATLAB® R2015a

F timingDiagram.m

```
function [ fig ] = timingDiagram( theta,rise )

fig = figure('units','normalized','outerposition',[0 0 .5 .15]);
global myWidth myFontSize myXLim
p1 = plot(theta,rise);
    xlabel('\theta (\circ)','FontSize',myFontSize)
    ylabel('s (\theta)')
    title('Cam Timing Diagram')
    xlim(myXLim)
    ylim([0,1.3*max(rise)])
    p1.LineWidth = myWidth;
    p1.Color = 'k';
end
```

Published with MATLAB® R2015a

G svaj.m

```
function [dom,sFun,vFun,aFun,jFun,fig] = svaj( theta, rise, numPoints, w)

% Each row is a linear spacing of points between thetas
targetPoints = numPoints;
global myWidth myFormat myFontSize myXLim
theta = [-15 theta 395]*pi/180;
rise = [rise(1) rise rise(end)];
dom = linspace(theta(1),theta(end),targetPoints);
sFun = pchip(theta,rise,dom);

% Remove the extra values used to make endpoints continuous
remove = find( dom < 0 );
remove2 =  find( dom > 2*pi );
remove = [remove remove2];
dom(dom<0)=[];
dom(dom>2*pi)=[];
dom(1) = 0;
dom(end) = 2*pi;
sFun(remove)=[];
sFun(1) = rise(1);
sFun(end) = rise(end);

% Find the velocity, acceleration, and jerk as functions of angle
step = 2*pi / numel(dom);
vFun = diff(sFun)/step;
aFun = diff(vFun)/step;
jFun = diff(aFun)/step;

dom2 = dom;
dom = dom*180/pi;

% Generate plots of the SVAJ data
fig= figure('units','normalized','outerposition',[0 0 1/3 1]);
subplot(4,1,1)
hold on
plot(theta*180/pi,rise,'or')
spS = plot( dom,sFun );
    xlim(myXLim)
    spS.LineStyle = '--';
    spS.LineWidth = myWidth;
    spS.Color = 'k';%[.2 .2 .2];
ylabel('s (\theta) Units','FontSize',myFontSize)
title('Pchip Cam svaj Diagrams')
hold off
 ax = gca;
 ax.XTickLabel = {};
 ax.YLabel.Position = [-30,.5,-1];

subplot(4,1,2)
spV = plot(dom(1:numel(vFun)),w*vFun);
    xlim(myXLim)
    spV.LineStyle = '-.';
    spV.LineWidth = myWidth;
    spV.Color = 'k';
```

H plotProfile.m

```
function [p3,x,y] = plotProfile( rBase,rFollow,t, s, e )

rPitch = rBase + rFollow;
global myWidth
d = (rPitch.^2 - e.^2).^.5;
lambda = (2*pi - t) - atan2(e,d+s);

x = cos(lambda) .* ((d + s).^2 + e.^2).^.5;
y = sin(lambda) .* ((d + s).^2 + e.^2).^.5;

p3 = plot(x,y,0,0,'r*');
title(['Roller Follower Cam Profile, R_b = ' num2str(rBase)])
l = legend('Cam','Axis');
    l.Box = 'off';
p3(1).LineWidth = myWidth;
p3(1).Color = 'k';
p3(1).LineStyle = '-';
p3(2).MarkerSize = 15;


end
```

Published with MATLAB® R2015a

I camChecker.m

```
function [rBaseFound,rPitch,rCurve ] = CamChecker(rBase,rFollow,ecc,myFactor,s,v,a,t)
%CamChecker - Finds minimum cam size & plots it
% Cam must be of a certain size to meet undercutting constraints inherent
% to roller-follower cams

% anonymous function to compute the pitch radius
rP = @(rBase,rFollow) rBase + rFollow;
rPitch = rP(rBase,rFollow);

% anonymous function to compute the curve radius
Ind = 1:numel(a);   % fixes the lost data point from diff
rC = @(rPitch,s,v,a) ( ( rPitch + s(Ind) ).^2 + v(Ind).^2 ) .^(3/2) ...
    / ( (rPitch + s(Ind)).^2 ...
    + 2 .* v(Ind).^2 - a.*(rPitch + s(Ind) )  );
rCurve = rC(rPitch,s,v,a);

% anonymous function to compute the pressure angle
phi = @(rPitch,s,v,ecc) atan2d(v-ecc,s( 1:numel(v) ) + (rPitch.^2 - ecc.^2).^.5);

% Make sure that the specified radii follow some rules
% Curve radius must be 3 times follower radius
while ( min(abs(rCurve)) < (3 * rFollow)) || ...
        (max( abs( phi(rPitch,s,v,ecc) ) )   > 30)

    % smooth the cam by increasing the base radius
    rBase = rBase + myFactor*max(s);
    rPitch = rP(rBase,rFollow);
    rCurve = rC(rPitch,s,v,a);
end

rBaseFound = rBase;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the parametric curves
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
timePause = .5;
fig3 = figure;
myScaler = 120;
posVec = [10*myScaler+10 (1080-5*myScaler-70) 5*myScaler 5*myScaler];
fig3.Position = posVec;
rBaseMin = rBase/2;
rBaseMax = 3*rBase;
rBaseStep = rBase/4;
for rBase = rBaseMin:rBaseStep:rBaseMax
    plotProfile(rBase,rFollow,t,s,ecc);
    xlim([-rBaseMax-2*max(s),rBaseMax+2*max(s)])
    ylim([-rBaseMax-2*max(s),rBaseMax+2*max(s)])
    pause(timePause)
end

end
```

J Standard Cam Motion Equations

The published cam equations from which we may compare our results are shown here.

J.1 Cycloidal equations

$$s(\theta)=\begin{cases} L_1\left(\frac{\theta_1}{\beta_1}-\frac{1}{2\pi}\sin\frac{2\pi\theta_1}{\beta_1}\right),\ 0\le\theta_1<\beta_1 & \text{for } 0^\circ\le\theta<90^\circ,\ \beta_1=90^\circ \\ L_1,\ 0\le\theta_2<\beta_2 & \text{for } 90^\circ\le\theta<180^\circ,\ \beta_2=90^\circ \\ L_1+(L_2-L_1)\left(\frac{\theta_3}{\beta_3}-\frac{1}{2\pi}\sin\frac{2\pi\theta_3}{\beta_3}\right),\ 0\le\theta_3<\beta_3 & \text{for } 180^\circ\le\theta<270^\circ,\ \beta_3=90^\circ \\ L_2\left(1-\frac{\theta_4}{\beta_4}+\frac{1}{2\pi}\sin\frac{2\pi\theta_4}{\beta_4}\right),\ 0\le\theta_4<\beta_4 & \text{for } 270^\circ\le\theta<360^\circ,\ \beta_4=90^\circ \end{cases}$$

$$v(\theta)=\begin{cases} \frac{L_1}{\beta_1}\left(\frac{\theta_1}{\beta_1}-\cos\frac{2\pi\theta_1}{\beta_1}\right) & \text{for } 0^\circ\le\theta<90^\circ \\ 0 & \text{for } 90^\circ\le\theta<180^\circ \\ \frac{L_2-L_1}{\beta_3}\left(1-\cos\frac{2\pi\theta_3}{\beta_3}\right) & \text{for } 180^\circ\le\theta<270^\circ \\ -\frac{L_2}{\beta_4}\left(1-\cos\frac{2\pi\theta_4}{\beta_4}\right) & \text{for } 270^\circ\le\theta<360^\circ \end{cases}$$

$$a(\theta)=\begin{cases} \frac{2\pi L_1}{{\beta_1}^2}\sin\frac{2\pi\theta_1}{\beta_1} & \text{for } 0^\circ\le\theta<90^\circ \\ 0 & \text{for } 90^\circ\le\theta<180^\circ \\ \frac{2\pi(L_2-L_1)}{{\beta_3}^2}\sin\frac{2\pi\theta_3}{\beta_3} & \text{for } 180^\circ\le\theta<270^\circ \\ -\frac{2\pi L_2}{{\beta_4}^2}\sin\frac{2\pi\theta_4}{\beta_4} & \text{for } 270^\circ\le\theta<360^\circ \end{cases}$$

$$j(\theta)=\begin{cases} \frac{4\pi^2 L_1}{{\beta_1}^3}\cos\frac{2\pi\theta_1}{\beta_1} & \text{for } 0^\circ\le\theta<90^\circ \\ 0 & \text{for } 90^\circ\le\theta<180^\circ \\ \frac{4\pi^2(L_2-L_1)}{{\beta_3}^3}\cos\frac{2\pi\theta_3}{\beta_3} & \text{for } 180^\circ\le\theta<270^\circ \\ -\frac{4\pi^2 L_2}{{\beta_4}^3}\cos\frac{2\pi\theta_4}{\beta_4} & \text{for } 270^\circ\le\theta<360^\circ \end{cases}$$

J.2 Simple harmonic equations

$$s(\theta) = \begin{cases} \frac{L_1}{2}\left(1-\cos\frac{\pi\theta_1}{\beta_1}\right), 0 \le \theta_1 < \beta_1 & \text{for } 0^\circ \le \theta < 90^\circ, \beta_1 = 90^\circ \\ L_1, 0 \le \theta_2 < \beta_2 & \text{for } 90^\circ \le \theta < 180^\circ, \beta_2 = 90^\circ \\ L_1 + \frac{L_2 - L_1}{2}\left(1-\cos\frac{\pi\theta_3}{\beta_3}\right), 0 \le \theta_3 < \beta_3 & \text{for } 180^\circ \le \theta < 270^\circ, \beta_3 = 90^\circ \\ \frac{L_2}{2}\left(1-\cos\frac{\pi\theta_4}{\beta_4}\right), 0 \le \theta_4 < \beta_4 & \text{for } 270^\circ \le \theta < 360^\circ, \beta_4 = 90^\circ \end{cases}$$

$$v(\theta) = \begin{cases} \frac{\pi L_1}{2\beta_1}\sin\frac{\pi\theta_1}{\beta_1} & \text{for } 0^\circ \le \theta < 90^\circ \\ 0 & \text{for } 90^\circ \le \theta < 180^\circ \\ \frac{\pi(L_2 - L_1)}{2\beta_3}\sin\frac{\pi\theta_3}{\beta_3} & \text{for } 180^\circ \le \theta < 270^\circ \\ -\frac{\pi L_2}{2\beta_4}\sin\frac{\pi\theta_4}{\beta_4} & \text{for } 270^\circ \le \theta < 360^\circ \end{cases}$$

$$a(\theta) = \begin{cases} \frac{\pi^2 L_1}{2{\beta_1}^2}\cos\frac{\pi\theta_1}{\beta_1} & \text{for } 0^\circ \le \theta < 90^\circ \\ 0 & \text{for } 90^\circ \le \theta < 180^\circ \\ \frac{\pi^2 (L_2 - L_1)}{2{\beta_3}^2}\cos\frac{\pi\theta_3}{\beta_3} & \text{for } 180^\circ \le \theta < 270^\circ \\ \frac{\pi^2 L_2}{2{\beta_4}^2}\cos\frac{\pi\theta_4}{\beta_4} & \text{for } 270^\circ \le \theta < 360^\circ \end{cases}$$

$$j(\theta) = \begin{cases} -\frac{\pi^3 L_1}{2{\beta_1}^3}\sin\frac{\pi\theta_1}{\beta_1} & \text{for } 0^\circ \le \theta < 90^\circ \\ 0 & \text{for } 90^\circ \le \theta < 180^\circ \\ -\frac{\pi^3 (L_2 - L_1)}{2{\beta_3}^3}\sin\frac{\pi\theta_3}{\beta_3} & \text{for } 180^\circ \le \theta < 270^\circ \\ \frac{\pi^3 L_2}{2{\beta_4}^3}\sin\frac{\pi\theta_4}{\beta_4} & \text{for } 270^\circ \le \theta < 360^\circ \end{cases}$$