Matlab Processing of FSAE TTC Tire Test Data


```
Bill Cobb
Vehicle Dynamics Center
General Motors Corp.
(248) 515-5145
william.a.cobb@gm.com
```


Contents

Introduction

This Matlab code reads a user-selected TIRF raw text file for a Pure Slip test condition and produces Splined Surface Structures for all dependent variables.

It is intended to be a starting point for FSAE Tire Test Consortium members wishing to produce their own tire models. The resulting Matlab spline structures are compatible with various commercial and home grown handling simulations via a Simulink interface. Knowledgeable users will quickly be able to engage in tire sizing, brand, wheel width, and pressure optimization activities by integrating this data into vehicle dynamics models performing open and closed loop maneuvers.

Fetch data file name using UIGETFILE function

`uigetfile` is handy to fetch the path and filename of your test file. You will probably want to make a change to the next line to match your TIRF file locations:

```
[filename pathname]= uigetfile('*.dat','Enter TIRF Test File','C:\Documents and Settings\zzvyb6\My Documents\FSAE\' )
```

```
filename =
B1175run1.dat
pathname =
C:\Documents and Settings\zzvyb6\My Documents\FSAE\
```

Import a data file

`importdata` is a built-in Matlab data reading routine. The resulting structure `t` will have a set of text and numeric fields.

```
t = importdata([pathname filename])
```

```
t =
          data: [25955x21 double]
      textdata: {3x21 cell}
    colheaders: {1x21 cell}
```

Determination of Channel Names:

Since TIRF files can contain any number of arbitrary data channels, we don't assume which ones are there or the order they are in.

By the way, the TIRF channels we are assuming to be here are:

```
SA = Slip angle
IA = Camber Angle
FZ = Vertical Load
FY = Lateral Force
MX = Overturning Moment
MZ = Aligning Moment
```

If any of these channels are not in the file, Houston, we're gonna have a problem.

Here we go:

```
names  = t.textdata{2}
nchans = size(t.data,2)
```

```
names =
ET      V       N       SA      IA      RL      RE      P       FX      FY      FZ      MX      MZ      NFX     NFY     RST     TSTI    TST
nchans =
    21
```

Eliminate Unused Data

Here, we toss out the 1st 1500 pts (warmup). Brackets mean just go away:

```
t.data(1:1500,:)=[];
```

Demultiplex Data Array into Unique Channels

The next step is to pluck out the names of channels in the file and assign the data array elements to these names.

```
for n=1:nchans          % demultiplex
    [name,names]=strtok(names);
    eval([upper(name) '= t.data(:,' num2str(n) ');']);
end
```

Spline the Slip Channel to locate Points of Interest

Here we are going to sniff out the key locations in the slip stream corresponding to the beginning and ending of the slip sweep.

First, define a point vector corresponding to the SA index. Then use the builtin spline routine to give us a continuous function. By adding 3.5 to the slip angle data, we shift the function up towards the zero level. Remember that the test procedure starts the slip sweep beginning at -4 degrees.

The reason for this will be revealed in the next step.

```
m  = 1:length(SA);     % point counter
sp = spline(m,SA+3.5); % fit a generic spline to locate zeros.
```

Locate Zeros Indicating Start and Stop Positions

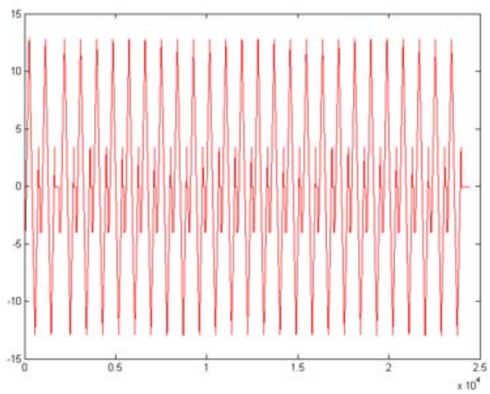
We now have a splined string of shifted slip angle. The zeros of this function include the beginning and ending of the slip sweep. There is some extra stuff in there, which may be of interest to the diehards:

```
z=fnzeros(sp);      % location of zero crossings
z=round(z(1,:));   % no dups and integer indices are required.
```

Check your results with a plot

Plotting the slip angle channel with the zero points indicated is a good check to see if there has been a problem. The full plot is a bit to muddy, so we zoom into the first few seconds of the run.

```
figure('Name','Locations of Test Slip Sweep  william.a.cobb@gm.com','NumberTitle','Off')  % Just checkin'
plot(m,SA,'r')
```



Eliminate unnecessary zero conditions

Because there are some 'zero' points that we don't need, send them to the trash:

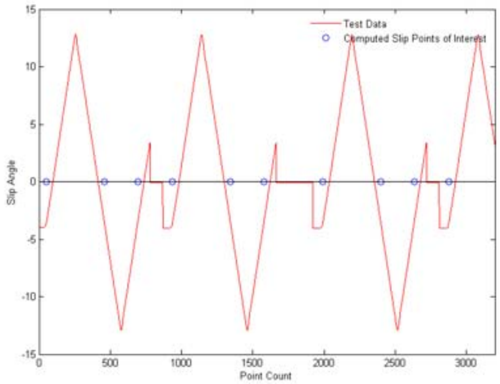
```
z([4:4:length(z)])=[];  % drop kick the shutdown points;
```

Show Abreviated Slip sequence with indicated Zeros

This tells us we're on the right track:

```
hold on
xlim([0 3200])  % Don't need to see All the data...
plot(z,zeros(length(z)),'bo')
line([0 m(end)], [0 0],'color','k')
xlabel('Point Count')
ylabel('Slip Angle')
legend('Test Data','Computed Slip Points of Interest'),legend Boxoff
% Don't want to remember any data from apprevious processing session:
```

```
clear fmdata    % not a speck of cereal..
```

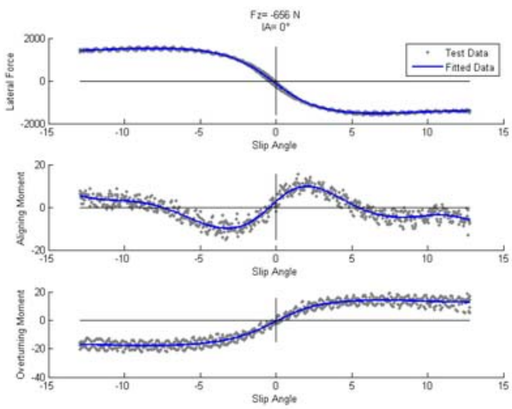


Outputting scans of data for each slip, load, and camber condition

Now we need to spew out the data as a sequential data vector for each separate slip, load and camber scan: q is the scan pointer

```
q = 0;
% Subset the data:
for n=7:3:length(z) % for some reason there are some repeat runs here. Skip them
    sa=SA(z(n):z(n+2));
    fz=FZ(z(n):z(n+2));
    fy=FY(z(n):z(n+2));
    mz=MZ(z(n):z(n+2));
    mx=MX(z(n):z(n+2));
    rl=RL(z(n):z(n+2));
    ia=IA(z(n):z(n+2));

    % Now we have collected the tire channels for each full slip sweep.
    %
    % Next step is to capture the rational data between the max and minimum
    % values, peek at the endpoints, fix up some problems at the MZ endpoints,
    % and proceed with data fitting.

    [tmp,imn]=min(sa);
    [tmp,imx]=max(sa);
    p=1:length(sa);
    rng=imx-50:imx+50;  % This is a range of our data at maximum MZ
    %
    % Being careful not to use a Matlab reserved word for a variable name.
    %
    warning off     % Keep down the chatter over multiple observations

    % fit this data to a polynomial.  Crude but fair.  We are only using it
    % to look for outliers.
    pp=polyfit(p(rng),mz(rng)',3);
    warning on
    mzf=polyval(pp,p(rng));

    % This step spots data values for MZ that are greater than an arbitray
    % level.  I believe these spikes are related to the MZ transient response.
    % A smarter approach would be to use normalized residuals, but who did
    % I just hear volunteer for that task?
    %
    ind=find(abs(mzf-mz(rng)') > 7);
    mz(rng(ind))=mzf(ind);

    rng=imn-50:imn+50;% This is a range of our data at minimum MZ
    warning off
    pp=polyfit(p(rng),mz(rng)',3);
    warning on
    mzf=polyval(pp,p(rng));

    ind=find(abs(mzf-mz(rng)') > 7);
    mz(rng(ind))=mzf(ind);

    %% Spline fitting the continuous data to subset it with 1 Degrre increments
    % with some tighter tension:
    sp_fy=csaps(sa,fy,.1);
    sp_mz=csaps(sa,mz,.1);
    sp_mx=csaps(sa,mx,.1);
    sp_rl=csaps(sa,rl,.1);

    %% Check out Segment 10
    % Just out of curiosity, what kind of data are we dealing ?
    if isequal(n,10)
        figure('Name',[upper(filename) ': Aligning Moment vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off
        subplot(3,1,1)
        hold on
        plot(sa,fy,'.','color',[.5 .5 .5])
        fnplt(sp_fy,'b')
        title({['Fz= ' num2str(round(mean(fz))) ' N'];['IA= ' num2str(round(mean(ia))) '°']})
        xlabel('Slip Angle')
        ylabel('Lateral Force')
        line([min(sa) max(sa)],[0 0],'color','k')
        line([0 0],[min(fy) max(fy)],'color','k')
        legend('Test Data','Fitted Data')
```

```
        subplot(3,1,2)
        hold on
        plot(sa,mz,'.','color',[.5 .5 .5])
        fnplt(sp_mz,'b')
        xlabel('Slip Angle')
        ylabel('Aligning Moment')
        line([min(sa) max(sa)],[0 0],'color','k')
        line([0 0],[min(mz) max(mz)],'color','k')
        subplot(3,1,3)
        hold on
        plot(sa,mx,'.','color',[.5 .5 .5])
        fnplt(sp_mx,'b')
        xlabel('Slip Angle')
        ylabel('Overturning Moment')
        line([min(sa) max(sa)],[0 0],'color','k')
        line([0 0],[min(mz) max(mz)],'color','k')
    end

    for sl=floor(min(sa)):1:ceil(max(sa));  % This is the only pushup step required:
        q=q+1;
        fmdata(q,1)=sl;
        fmdata(q,2)=round(mean(ia));
        fmdata(q,3)=mean(fz);
        fmdata(q,4)=fnval(sp_fy,sl);
        fmdata(q,5)=fnval(sp_mz,sl);
        fmdata(q,6)=fnval(sp_mx,sl);
    end
end
% All done with the hard work,
```



Sort Data Array by Camber, Slip and Load Groups

Use `sortrows` to preserve the array correspondence.

```
fmdata  = sortrows(fmdata,[2,1,3]);
```

Determining Data Sets (Slip, Load, Camber):

Get the distinct values and counts of each data set:

```
incls   = unique(round(fmdata(:,2)))'
nincls  = length(incls)

slips   =  unique(round(fmdata(:,1)))'
nslips  = length(slips)


incls =
     0     1     2     3     4
nincls =
     5
slips =
  Columns 1 through 17
   -13   -12   -11   -10    -9    -8    -7    -6    -5    -4    -3    -2    -1     0     1     2     3
  Columns 18 through 27
     4     5     6     7     8     9    10    11    12    13
nslips =
    27
```

Check Visuals for Zero Camber Subset:

Stuck living in a 3-D world, we can only look at a graph of w=f(u,v). First sniff out indices of the zero camber rows:

```
inx0    = find(fmdata(:,2) == 0);  % zero camber points
% then grab the rest of the zero camber condition data:
fmdata0 = fmdata(inx0,:);
%  Next, we transpose the arrays to make our spline functions happy:
loads   = mean(reshape(fmdata0(:,3),[],nslips),2)'
nloads  = length(loads)
%  Take a look at FZ:
fz0     = reshape(fmdata0(:,3),nloads,nslips)'
fy0     = reshape(fmdata0(:,4),nloads,nslips)';
mz0     = reshape(fmdata0(:,5),nloads,nslips)';
```

```
mx0     = reshape(fmdata0(:,6),nloads,nslips)';
%
%  Normalized FY for you folks headed to the tire industry:
nfy0    = fy0./fz0

% now we are ready to rock 'n roll.
```

```
loads =
      -1557.1      -1097.4      -656.26       -434.3      -214.41
nloads =
     5
fz0 =
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
      -1557.1      -1097.4      -656.26       -434.3      -214.41
nfy0 =
      -2.0113      -2.1595      -2.1571      -2.2706      -2.3373
       -2.026      -2.1769      -2.2001      -2.3145      -2.3655
      -2.0441      -2.1988      -2.2406      -2.3564      -2.3944
      -2.0638       -2.224      -2.2747       -2.389      -2.4179
       -2.078      -2.2419       -2.299      -2.4083      -2.4249
      -2.0784      -2.2402      -2.3091      -2.4093      -2.4114
       -2.055       -2.206      -2.2965      -2.3738      -2.3767
      -1.9937      -2.1272      -2.2479      -2.2931      -2.3104
      -1.8785      -1.9944      -2.1479      -2.1707      -2.2035
      -1.6931      -1.7994      -1.9764      -1.9924      -2.0387
        -1.41      -1.5135      -1.6922      -1.7033      -1.7643
     -0.98999       -1.084      -1.2301      -1.2372      -1.3078
     -0.43636     -0.49187     -0.56307     -0.57101     -0.62809
      0.19053      0.20067      0.22988       0.2163      0.20361
      0.79607        0.868      0.98268      0.95305      0.98468
       1.2937       1.4026       1.5585       1.5076       1.5595
       1.6404       1.7729       1.9306       1.8685       1.9225
       1.8525       2.0092       2.1504       2.0965       2.1366
       1.9713       2.1548       2.2714       2.2438       2.2652
       2.0341       2.2385       2.3244       2.3392       2.3521
       2.0655       2.2777       2.3289       2.3968       2.4139
       2.0791        2.285       2.3006       2.4215       2.4504
       2.0813       2.2737       2.2563       2.4199       2.4639
        2.074       2.2537       2.2124       2.4008       2.4606
       2.0586       2.2266       2.1809       2.3667       2.4443
       2.0424       2.2004       2.1595       2.3198       2.4176
       2.0278       2.1797       2.1433       2.2682       2.3871
```

Beer break...

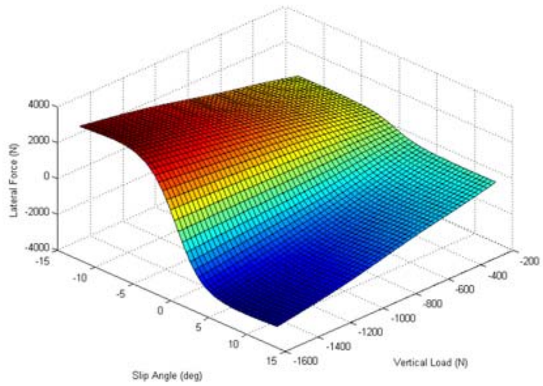
```
%The test data has been reduced to a reasonable grid of fixed slips, cambers and loads.
% All that remains is to do something with it in the tire model world.
```

FY Surface Fit (Zero Camber):

This process uses the Spline toolbox functions to generate fitted surfaces. CSAPS is a cubic - smoothed spline allowing multiple obervations at grid points with default or user prescribed tension

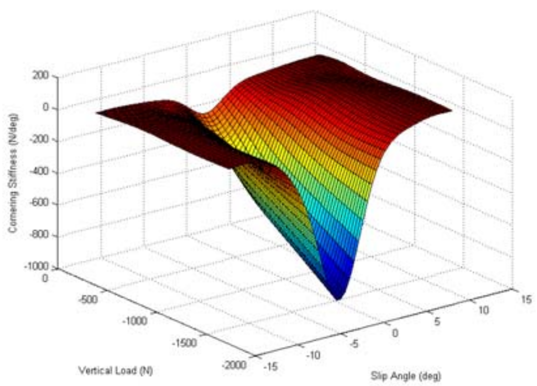
```
LATE_SLIP_VERT = csaps({slips,loads},fy0,.9)
figure('Name',[upper(filename) ': Lateral Force vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(LATE_SLIP_VERT)
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Lateral Force (N)')
view(45,45)
```

```
LATE_SLIP_VERT =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x104x16 double]
    pieces: [26 4]
     order: [4 4]
       dim: 1
```



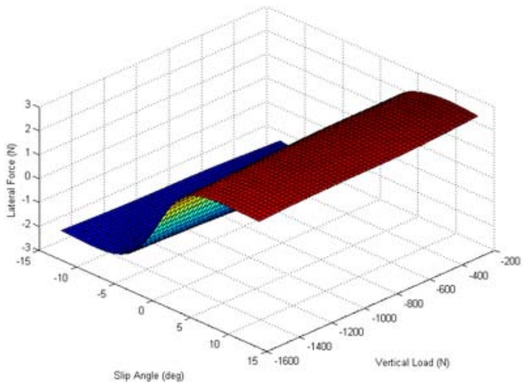
Cornering Stiffness Surface Fit (Zero Camber):

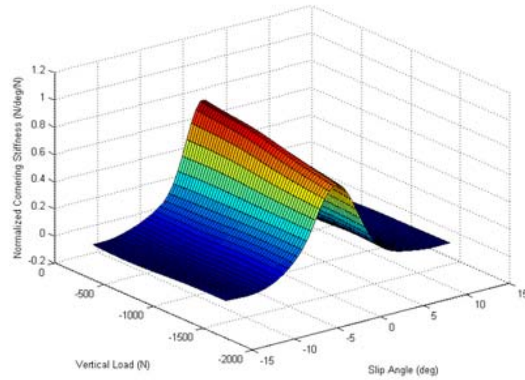
```
CS=fnder(LATE_SLIP_VERT,[1,0])
figure('Name',[upper(filename) ': Cornering Stiffness vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(CS)
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Cornering Stiffness (N/deg)')


CS =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x78x16 double]
    pieces: [26 4]
     order: [3 4]
       dim: 1
```



Normalized Lateral Force Surface fit

```
NLATE_SLIP_VERT = csaps({slips,loads},nfy0)
figure('Name',[upper(filename) ': Load Normalized Lateral Force vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle
fnplt(NLATE_SLIP_VERT)
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Lateral Force (N)')
view(45,45)
% Wow, look at the mu on that baby, good as a Sprint Cup Left Side Tire !!
% Here's the traditional normalized cornering stiffness used in industry:
NCS=fnder(NLATE_SLIP_VERT,[1,0])
figure('Name',[upper(filename) ': Normalized Cornering Stiffness vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitl
fnplt(NCS)
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Normalized Cornering Stiffness (N/deg/N)')


NLATE_SLIP_VERT =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x104x16 double]
    pieces: [26 4]
     order: [4 4]
       dim: 1
NCS =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x78x16 double]
    pieces: [26 4]
     order: [3 4]
       dim: 1
```

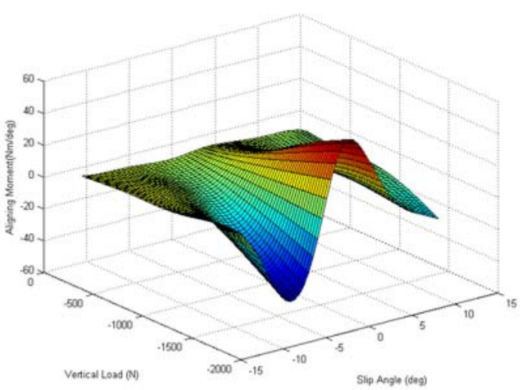




MZ Surface Fit (Zero Camber):

```
ALNT_SLIP_VERT = csaps({slips,loads},mz0,.9)
figure('Name',[upper(filename) ': Aligning Moment vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(ALNT_SLIP_VERT)
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Aligning Moment(Nm/deg)')
```

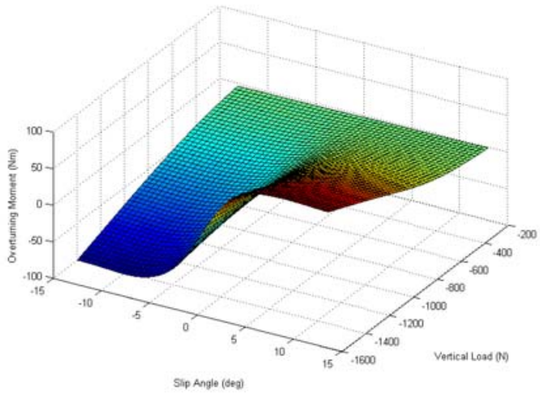
```
ALNT_SLIP_VERT =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x104x16 double]
    pieces: [26 4]
     order: [4 4]
       dim: 1
```



MX Surface Fit (Zero Camber):

```
OVTM_SLIP_VERT = csaps({slips,loads},mx0,.9)
figure('Name',[upper(filename) ': Overturning Moment vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(OVTM_SLIP_VERT)
view(30,45)
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Overturning Moment (Nm)')
```

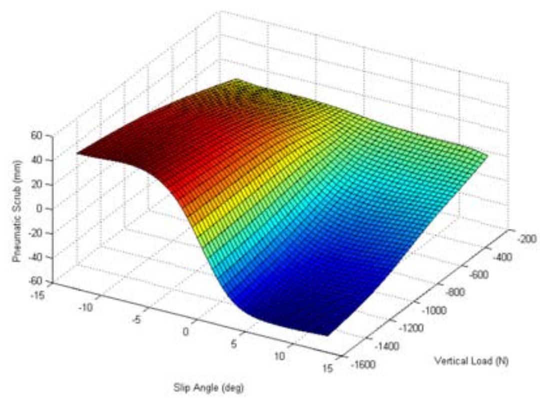
```
OVTM_SLIP_VERT =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x104x16 double]
    pieces: [26 4]
     order: [4 4]
       dim: 1
```



Pneumatic Scrub Surface Fit (Zero Camber):

```
PSCRUB_SLIP_VERT = csaps({slips,loads},1000*mx0./fz0,.9)
figure('Name',[upper(filename) ': Pneumatic Scrub vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(PSCRUB_SLIP_VERT)
view(30,45)
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Pneumatic Scrub (mm)')
```

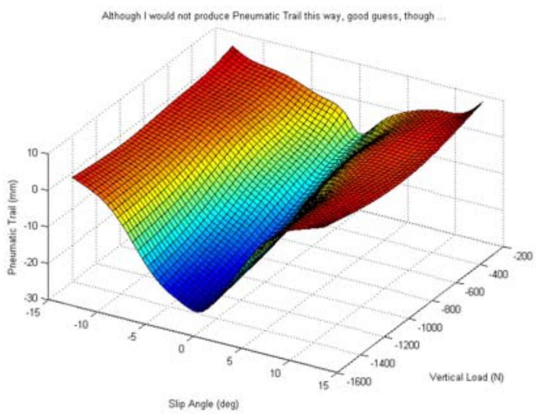
```
PSCRUB_SLIP_VERT =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x104x16 double]
    pieces: [26 4]
     order: [4 4]
       dim: 1
```



Pneumatic Trail Surface Fit (Zero Camber):

```
PTRAIL_SLIP_VERT = csaps({slips,loads},1000*mz0./fy0,.707)
figure('Name',[upper(filename) ': Pneumatic Trail vs. Slip Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(PTRAIL_SLIP_VERT)
view(30,45)
title('Although I would not produce Pneumatic Trail this way, good guess, though ...' )
xlabel('Slip Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Pneumatic Trail (mm)')
```

```
PTRAIL_SLIP_VERT =
      form: 'pp'
    breaks: {1x2 cell}
     coefs: [1x104x16 double]
    pieces: [26 4]
     order: [4 4]
       dim: 1
```



Subset Scans of Zero Slip Angle

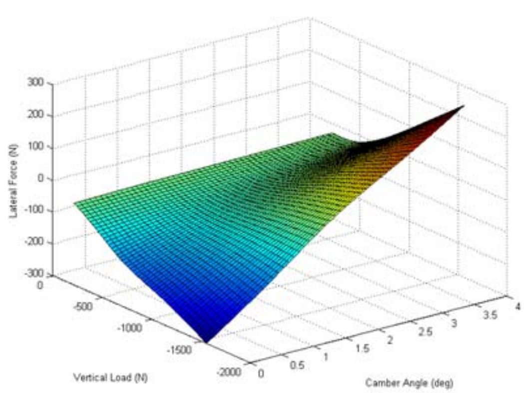
```
inx0    = find(fmdata(:,1) == 0);  % zero slip points
fmdata0 = fmdata(inx0,:);
loads   = mean(reshape(fmdata0(:,3),[],nincls),2)'   % We may have new load groups for camber
nloads  = length(loads)
fy0     = reshape(fmdata0(:,4),nloads,nincls)';
mz0     = reshape(fmdata0(:,5),nloads,nincls)';
mx0     = reshape(fmdata0(:,6),nloads,nincls)';
```

```
loads =
      -1559.8      -1107.3      -665.51      -442.64      -222.53
nloads =
     5
```

FY Surface Fit (Zero Slip):

```
LATE_INCL_VERT = csaps({incls,loads},fy0,.9)
figure('Name',[upper(filename) ': Lateral Force vs. Camber Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(LATE_INCL_VERT)
xlabel('Camber Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Lateral Force (N)')
```

```
LATE_INCL_VERT =
      form: 'pp'
    breaks: {[0 1 2 3 4]  [-1559.8 -1107.3 -665.51 -442.64 -222.53]}
     coefs: [1x16x16 double]
    pieces: [4 4]
     order: [4 4]
       dim: 1
```

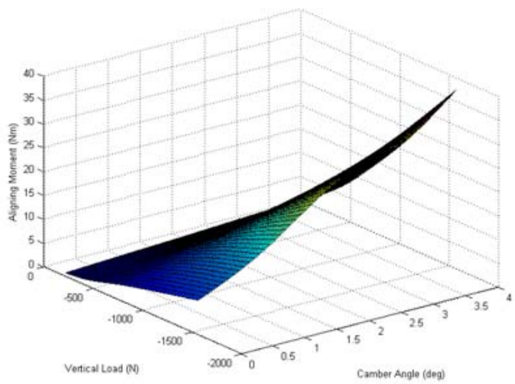


MZ Surface Fit (Zero Slip):

```
ALNT_INCL_VERT = csaps({incls,loads},mz0,.9)
figure('Name',[upper(filename) ': Aligning Moment vs. Camber Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(ALNT_INCL_VERT)
xlabel('Camber Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Aligning Moment (Nm)')
```

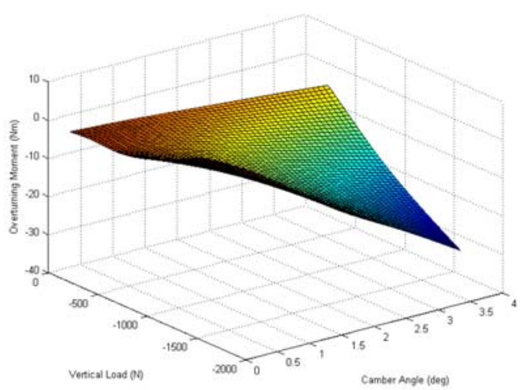
```
ALNT_INCL_VERT =
      form: 'pp'
    breaks: {[0 1 2 3 4]  [-1559.8 -1107.3 -665.51 -442.64 -222.53]}
     coefs: [1x16x16 double]
    pieces: [4 4]
     order: [4 4]
```

```
       dim: 1
```



MX Surface Fit (Zero Slip):

```
OVTM_INCL_VERT = csaps({incls,loads},mx0,.9)
figure('Name',[upper(filename) ': Overturning Moment vs. Camber Angle & Vertical Load  ' '   william.a.cobb@gm.com'],'numbertitle','off')
fnplt(OVTM_INCL_VERT)
xlabel('Camber Angle (deg)')
ylabel('Vertical Load (N)')
zlabel('Overturning Moment (Nm)')


OVTM_INCL_VERT =
      form: 'pp'
    breaks: {[0 1 2 3 4]  [-1559.8 -1107.3 -665.51 -442.64 -222.53]}
     coefs: [1x16x16 double]
    pieces: [4 4]
     order: [4 4]
       dim: 1
```



Published with MATLAB® 7.4