

# CUFSM and Matlab

- The Matlab version of CUFSM allows much greater flexibility than the standalone version.
- Within the Graphical User Interface you can use mathematics, anything you could enter on the command line in Matlab you can use in the GUI examples:
  - entering `1:1:10` in the Lengths section will be interpreted as  
1 2 3 4 5 6 7 8 9 10
  - to evenly space 30 points in log space between  $10^0$  and  $10^3$  enter `logspace(0,3,30)` in the Lengths section
  - to shift a member over 2 in. just add `+2` to `x` the nodal coordinates of all nodes
  - etc.

# CUFSM and Matlab

- The real power of the Matlab version is the ability to access all the features of the program from within your own programs (m-files.)
- In your m-files you can call the CUFSM routines to do your pre- and post-processing as well as perform the analysis directly and perform parametric studies, optimization, etc..

# Useful CUFSM Matlab functions

- Plotting and Post-Processing
  - crossect.m: plot the cross-section, node numbers, springs, etc.
  - strespic.m: show the stress distribution
  - thecurve.m: single model, buckling curve plot
  - thecurve2.m: multiple model, buckling curve plot
  - dispshap.m: plot a 2D buckling mode shape
  - dispshap3d.m: plot a 3D buckling mode shape
- Model Building
  - templatecalc.m: Generate finite strip model from centerline dimensions of a C or Z
  - doubler.m: double the number of elements in a model
  - grosprop.m: calculate the properties (A, I, etc.) for a model
  - yieldMP: given  $f_y$  calculate the yield loads and moments for a model
  - stresgen.m: given a load and/or moment calculate nodal stresses for a model
- Analysis
  - strip.m: Perform finite strip calculations for a model

# Matlab Example

- An example of performing a parametric study in Matlab using your own files is presented in the following slides.
- The example m-file is: [example\\_parameter\\_study.m](#)
- The example
  - set's up the CUFSM inputs for a Cee section in compression
  - performs analyses while varying the lip length
  - saves all results
  - plots the initial cross-sections used in the parametric study
  - plots the buckling curves from the study
  - plots the local and distortional buckling modes from the study

# Parameter Study Example

```
1 %BWS
2 %6 December 2001
3 %Matlab Parameter Study Example Problem
4 %
5 %Objective:
6 %To show how CUFSM's routine can be called from within your own m-files, in order to perform
7 %specialized plotting, parameter studies, simulations, etc...
8 %
9 %
10 %Consider a channel section with varying lip length in pure compression
11 %Millimeters are selected for length, Newtons for force, MPa = N/mm^2 for stress
12 %the reference compression stress is 1.0MPa is examined
13 %
14 %
15 %The basic variables will be
16 %      _____ b _____
17 % |                               |d
18 % |                               |
19 % |                               |
20 % |                               |
21 % |                               |
22 % | h                             |
23 % |                               |
24 % |                               |
25 % |                               |
26 % |                               |
27 % | _____ b _____ |d
28 %
29 %h = the web height
30 %b = the flange width
31 %d = the lip length
32 %t = thickness
33 %
34 path(path, 'd:\cufsm\cufsm_working\cufsm2p5_matlab'); %put your appropriate path statement to cufsm here.
35 %
36 %
37 h=100; %mm
38 b=30; %mm
39 d=[5 9 13 17]; %mm
40 t=1; %mm
```

The dimensions of the model are defined to the left.

# Setup and perform analysis

```
37 h=100; %mm
38 b=30; %mm
39 d=[5 9 13 17]; %mm
40 t=1; %mm
41 %
42 %Define the material properties
43 %These are the same inputs that are required in the graphical version of CUF5M
44 prop=[1 2.03E5 2.03E5 0.3 0.3 2.03E5/(2*(1+0.3))];
45 %
46 %Define the lengths
47 % could choose lengths, like lengths=[10:10:100 150:50:1000];, or even easier
48 lengths=logspace(1,3,50); %evenly space 50 points in logspace from 10^1 to 10^3
49 %
50 %No springs or constraints.
51 springs=0;
52 constraints=0;
53 %
54 %-----
55 %Enter in a loop where the lip length is varied and analysis is run
56 for i=1:length(d)
57     node=[1 b d(i) 1 1 1 1 1.0
58           2 b 0.0 1 1 1 1 1.0
59           3 0.0 0.0 1 1 1 1 1.0
60           4 0.0 h 1 1 1 1 1.0
61           5 b h 1 1 1 1 1.0
62           6 b h-d(i) 1 1 1 1 1.0];
63     elem=[1 1 2 t 1
64           2 2 3 t 1
65           3 3 4 t 1
66           4 4 5 t 1
67           5 5 6 t 1];
68     %double the number of elements to improve the discretization
69     %this is the same as using the Double Elem button in the graphical version
70     [node,elem]=doubler(node,elem);
71     %perform the finite strip analysis
72     [curve,shapes]=strip(prop,node,elem,lengths,1,springs,constraints);
```

The nodes and elements are defined just as they would be in the graphical version of CUF5M, but now we use variables instead of numbers. Most importantly, since d is an array we can perform a loop and systematically vary the lip length 'd'.

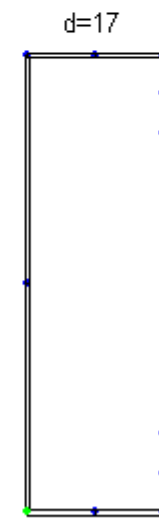
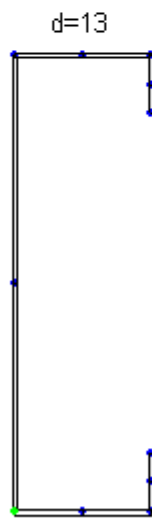
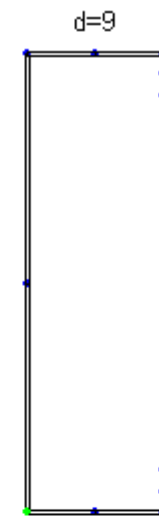
# Analysis and saved data

```
54 %-----  
55 %Enter in a loop where the lip length is varied and analysis is run  
56 for i=1:length(d)  
57     node=[1 b   d(i)   1 1 1 1 1.0  
58           2 b   0.0   1 1 1 1 1.0  
59           3 0.0 0.0   1 1 1 1 1.0  
60           4 0.0 h    1 1 1 1 1.0  
61           5 b   h    1 1 1 1 1.0  
62           6 b   h-d(i) 1 1 1 1 1.0];  
63     elem=[1 1 2 t 1  
64           2 2 3 t 1  
65           3 3 4 t 1  
66           4 4 5 t 1  
67           5 5 6 t 1];  
68     %double the number of elements to improve the discretization  
69     %this is the same as using the Double Elem button in the graphical version  
70     [node,elem]=doubler(node,elem);  
71     %perform the finite strip analysis  
72     [curve,shapes]=strip(prop,node,elem,lengths,1,springs,constraints);  
73     %Save all the inputs and the results for analysis "i"  
74     data(i).prop=prop;  
75     data(i).node=node;  
76     data(i).elem=elem;  
77     data(i).lengths=lengths;  
78     data(i).springs=springs;  
79     data(i).constraints=constraints;  
80     data(i).curve=curve;  
81     data(i).shapes=shapes;  
82     %  
83     save datafile data %save within the loop in case of abnormal termination of param study  
84 end  
85 %
```

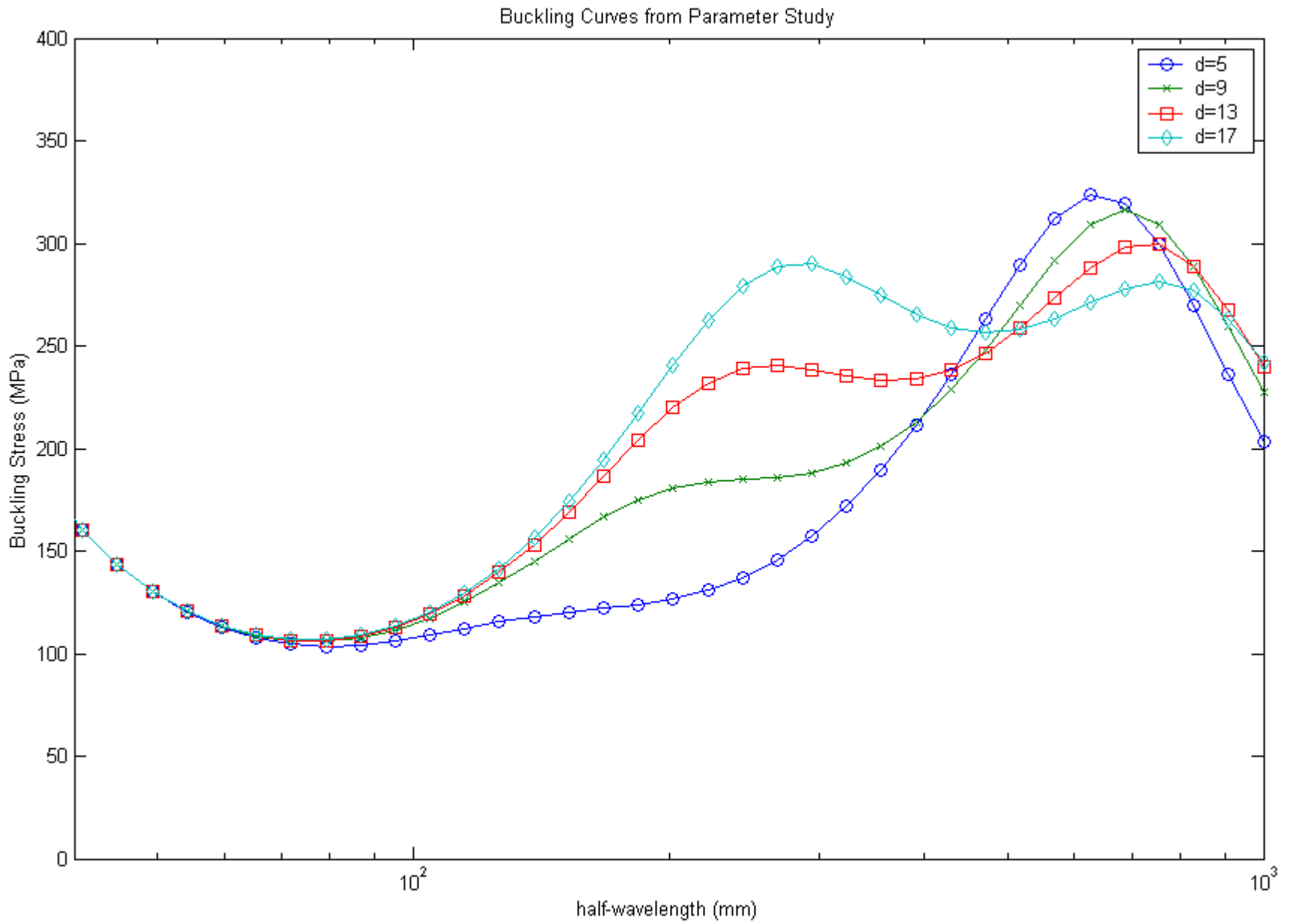
this one line performs the analysis.

# Plot sections and buckling curves

```
86 %POST-PROCESSING AND PLOTTING
87 %Let's take a look at our cross-sections
88 figure(1)
89 %flags:[node# element# mat# stress# stresspic coord constraints springs origin] 1 means show
90 flags=[0 0 0 0 0 0 1 1 1]; %these flags control what is plotted, node#, elem#
91 for i=1:4
92     axesnum=subplot(2,2,i)
93     crossect(data(i).node,data(i).elem,axesnum,data(i).springs,data(i).constraints,flags)
94     title(['d=',num2str(d(i))])
95 end
96
97 %
98 %Let's plot our buckling curve results
99 figure(2)
100 curve1=data(1).curve;,
101 curve2=data(2).curve;
102 curve3=data(3).curve;
103 curve4=data(4).curve;
104 semilogx(curve1(:,1),curve1(:,2),'-o',curve2(:,1),curve2(:,2),'-x',...
105     curve3(:,1),curve3(:,2),'-s',curve4(:,1),curve4(:,2),'-d') %plot solid lines w/ symbols
106 axis([40 1000 0 400])
107 title('Buckling Curves from Parameter Study')
108 legend('d=5','d=9','d=13','d=17')
109 xlabel('half-wavelength (mm)')
110 ylabel('Buckling Stress (MPa)') %because a referenc load of 1MPa was used this is
111     %the buckling stress instead of just the load factor
```



this plot is generated directly from the m-file!

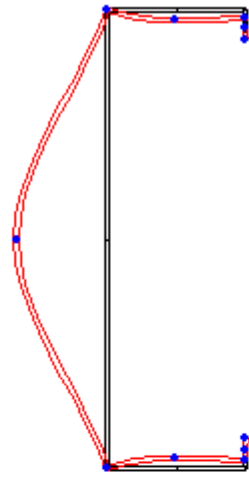


this plot is generated directly from the m-file!

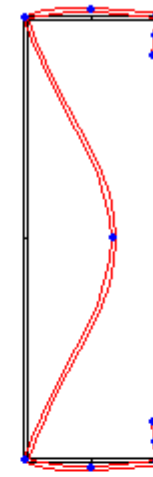
# Plot mode shapes

```
113 %Let's look at some mode shapes too.
114 modeindex=1;
115 undefv=1;
116 scale=1;
117 springs=0;
118 local_lengthindex=[23 23 23 23]; %this is the step that has the local minimum
119 dist_lengthindex=[33 37 40 42]; %this is the step that has the distortional minimum
120 figure(3)
121 for i=1:4
122     axesshape=subplot(2,2,i);
123     lengthindex=local_lengthindex(i);
124     dispshap(undefv,data(i).node,data(i).elem,data(i).shapes(:,lengthindex,modeindex),axesshape,scale,springs);
125     title(['LB, d=',num2str(d(i)),'mm \lambda=',num2str(data(i).curve(lengthindex,1)),...
126         'mm, P_{cr}=',num2str(data(i).curve(lengthindex,2)),'MPa'])
127 end
128 figure(4)
129 for i=1:4
130     axesshape=subplot(2,2,i);
131     lengthindex=dist_lengthindex(i);
132     dispshap(undefv,data(i).node,data(i).elem,data(i).shapes(:,lengthindex,modeindex),axesshape,scale,springs);
133     title(['DB, d=',num2str(d(i)),'mm \lambda=',num2str(data(i).curve(lengthindex,1)),'mm, P_{cr}=',...
134         num2str(data(i).curve(lengthindex,2)),'MPa'])
135 end
```

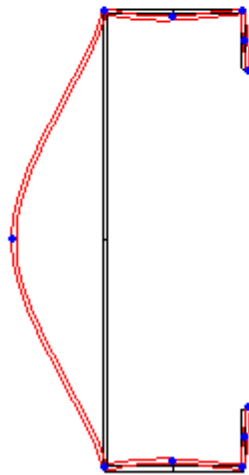
LB,  $d=5\text{mm}$   $\lambda=79.0604\text{mm}$ ,  $P_{cr}=103.5495\text{MPa}$



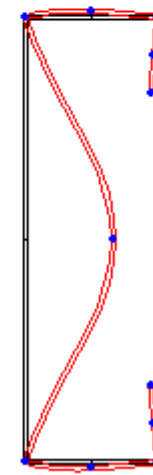
LB,  $d=9\text{mm}$   $\lambda=79.0604\text{mm}$ ,  $P_{cr}=105.8998\text{MPa}$



LB,  $d=13\text{mm}$   $\lambda=79.0604\text{mm}$ ,  $P_{cr}=106.4584\text{MPa}$

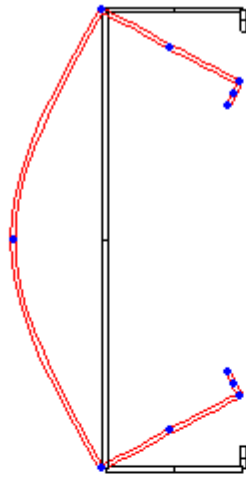


LB,  $d=17\text{mm}$   $\lambda=79.0604\text{mm}$ ,  $P_{cr}=106.6724\text{MPa}$

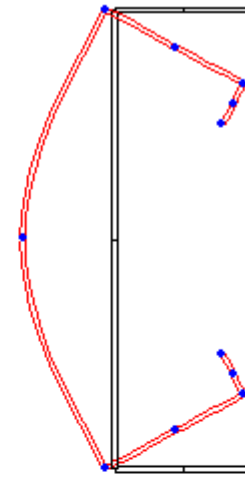


this plot is generated directly from the m-file!

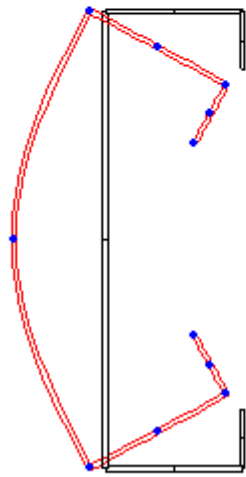
DB, d=5mm  $\lambda=202.359\text{mm}$ ,  $P_{cr}=126.7602\text{MPa}$



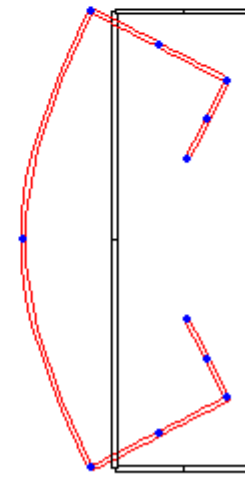
DB, d=9mm  $\lambda=294.7052\text{mm}$ ,  $P_{cr}=188.0572\text{MPa}$



DB, d=13mm  $\lambda=390.694\text{mm}$ ,  $P_{cr}=233.6511\text{MPa}$



DB, d=17mm  $\lambda=471.4866\text{mm}$ ,  $P_{cr}=256.4087\text{MPa}$



this plot is generated directly from the m-file!