Splinetool usage summary

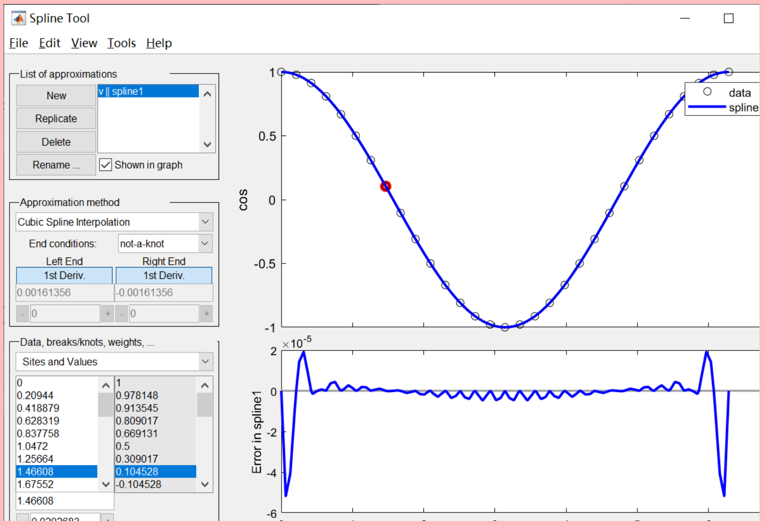
10170437 Mark Taylor

Oct. 6, 2019

Well, there are quite a few options about splinetool *MATLAB*® program when we first enter the graphical user interface (GUI) manu, including the option of importing some data from your workspace, which is real very convenient & impressive. Here I am mainly talk some some about **Import your own data** option. The rest parts , for example Titanium heat data, Census data, Richard Tapia's drag race data, are some distinct aspects in which the spline fitting applies.

OK, first, let's see the home window (see `figure 1`) if you use those default data input options. We can do many a manipulation on the fitting curves, well, in the `View` bar we can choose `Show 1st Derivative, Show 1st Derivative, Show Error` modes, in which cases the shape of the lower curve may alter when we attempt to modify (add or delete point) the data through `Edit` bar on the top. You can also turn grid on in the `Tool` bar if you like .

Then, let's proceed to the interesting parts. On the top we can see the `List of approximation` box, we first replicate a spline curve, you know, just do a copy, for the sake of comparison. After that, let's focus on the middle box `Approximation method` where we get lots of options to manipulate the curves. We have following 4 kinds of method provided by this program: `Cubic Spline Interpolation`, `smoothing Spline`, `Least-Square Approximation`, and `Spline Interpolation`. When we select the `Cubic Spline Interpolation` method, bellowing we get corresponding `End condition` which includes options: `not-a-knot, clamped, complete, second, periodic, variational, natural, Lagrange, Custom`. In our text, we have learned boundary conditions like clamped boundary (s'(a)=s'(b)), natural boundary (s"(a)=s"(b)=0), periodic boundary (s"(a)=s"(b)). The other 3 methods correspond with `Order` options



v || spline1

☑ Shown in graph

Cubic Spline Interpolation

End conditions: not-a-knot

Left End	Right End
1st Deriv.	1st Deriv.
0.00161356	-0.00161356
0	0

Sites and Values

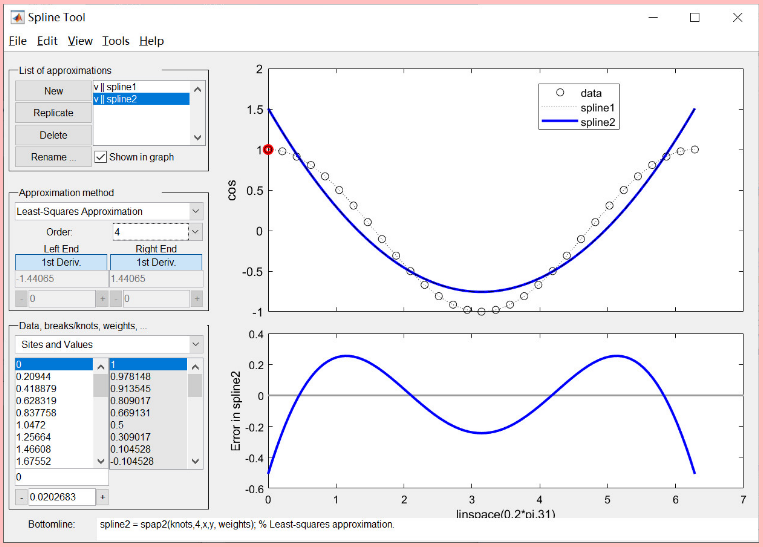
0	1
0.20944	0.978148
0.418879	0.913545
0.628319	0.809017
0.837758	0.669131
1.0472	0.5
1.25664	0.309017
1.46608	0.104528
1.67552	-0.104528

1.46608

Figure 1: Default initial window

below methods options. Take Last-Square Approximation for example, as the order increases, the spline becomes more smooth and error is decreasing exponentially! (see `figure 2` and `figure 3`). Last, in the bottom box you slowly increase (decrease) weights to see error varies graphically. You can also turn on `Breaks` which enable you see many vertical lines that intersect the spline at those data points (see `figure 4`). At the very bottom of this program box, you can see the `Bottomline` which tells you that the toolbox function csape was used to create the spline.

OK, now, it's our shot. Well, we first create a `Splinetool` test function, say $\frac{sin(x)}{x}$ (see `figure 5`), create one whichever you like. Oh, there is a break point x=0, and the program gives us a warning, click OK to ignore it, and here comes our home window (see `figure 6`).Then we can play around with those buttons. For instance, we replicate one and then select the `Least-Square Approximation` method, you'll get a terrible match, and when you ascend the order, it becomes better(see `figure 7` and `figure 8`). The other 3 spline methods have almost identical effect in drawing the fitting curves.



v || spline1
v || spline2

☑ Shown in graph

Approximation method: Least-Squares Approximation

Order: 4

Left End	Right End
1st Deriv.	1st Deriv.
-1.44065	1.44065
0	0

Data, breaks/knots, weights, ...: Sites and Values

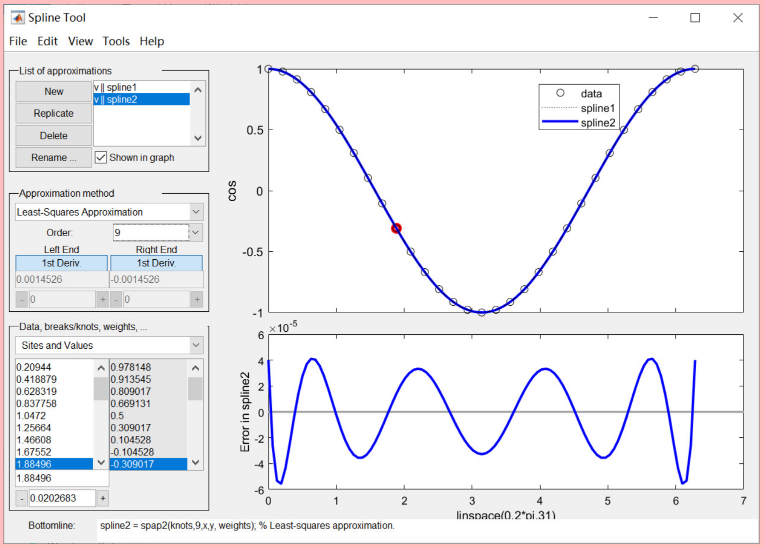
0	1
0.20944	0.978148
0.418879	0.913545
0.628319	0.809017
0.837758	0.669131
1.0472	0.5
1.25664	0.309017
1.46608	0.104528
1.67552	-0.104528

0

0.0202683

Bottomline: `spline2 = spap2(knots,4,x,y, weights); % Least-squares approximation.`

Figure 2: Last-Square Appr order 4



v || spline1
v || spline2

☑ Shown in graph

Approximation method: Least-Squares Approximation

Order: 9

Left End	Right End
1st Deriv.	1st Deriv.
0.0014526	-0.0014526
0	0

Data, breaks/knots, weights, ...: Sites and Values

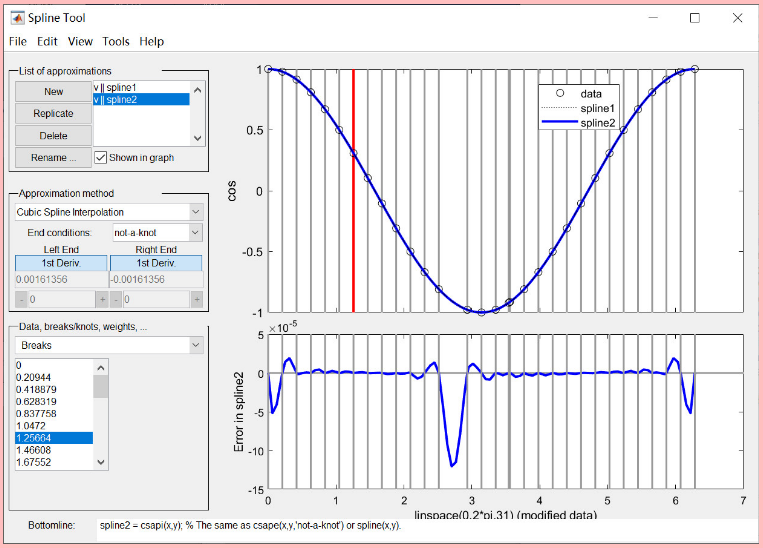
0.20944	0.978148
0.418879	0.913545
0.628319	0.809017
0.837758	0.669131
1.0472	0.5
1.25664	0.309017
1.46608	0.104528
1.67552	-0.104528
1.88496	-0.309017

1.88496

0.0202683

Bottomline: `spline2 = spap2(knots,9,x,y, weights); % Least-squares approximation.`

Figure 3: Last-Square Appr order 9



v || spline1
v || spline2

☑ Shown in graph

Cubic Spline Interpolation

End conditions: not-a-knot

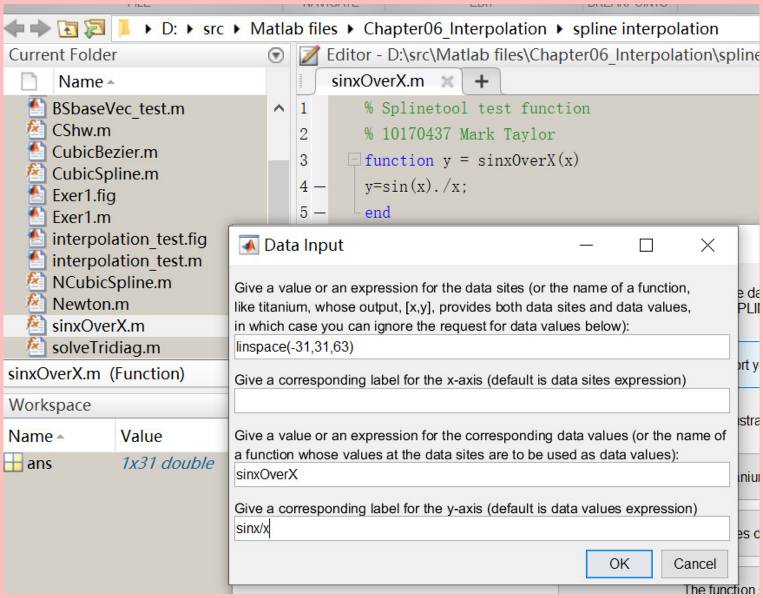
Left End	Right End
1st Deriv.	1st Deriv.
0.00161356	-0.00161356
0	0

Breaks

0
0.20944
0.418879
0.628319
0.837758
1.0472
1.25664
1.46608
1.67552

Bottomline: spline2 = csapi(x,y); % The same as csape(x,y,'not-a-knot') or spline(x,y).

Figure 4: Breaks mode



D: ▸ src ▸ Matlab files ▸ Chapter06_Interpolation ▸ spline interpolation

Editor - D:\src\Matlab files\Chapter06_Interpolation\spline

BSbaseVec_test.m
CShw.m
CubicBezier.m
CubicSpline.m
Exer1.fig
Exer1.m
interpolation_test.fig
interpolation_test.m
NCubicSpline.m
Newton.m
sinxOverX.m
solveTridiag.m

sinxOverX.m (Function)

Name	Value
ans	1x31 double

sinxOverX.m

```
% Splinetool test function
% 10170437 Mark Taylor
function y = sinxOverX(x)
y=sin(x)./x;
end
```

Data Input

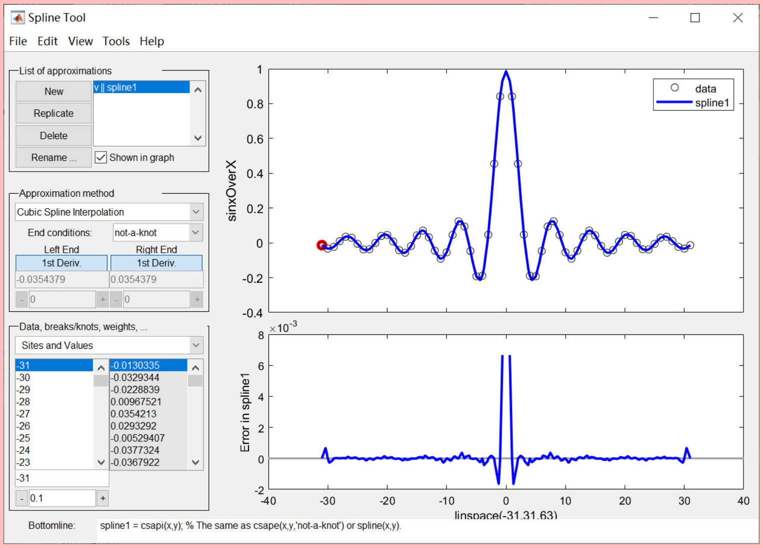
Give a value or an expression for the data sites (or the name of a function, like titanium, whose output, [x,y], provides both data sites and data values, in which case you can ignore the request for data values below):
linspace(-31,31,63)

Give a corresponding label for the x-axis (default is data sites expression)

Give a value or an expression for the corresponding data values (or the name of a function whose values at the data sites are to be used as data values):
sinxOverX

Give a corresponding label for the y-axis (default is data values expression)
sinx/x

Figure 5: My data input



Approximation method: Cubic Spline Interpolation

End conditions: not-a-knot

Left End	Right End
1st Deriv.	1st Deriv.
-0.0354379	0.0354379
0	0

Data, breaks/knots, weights, ...: Sites and Values

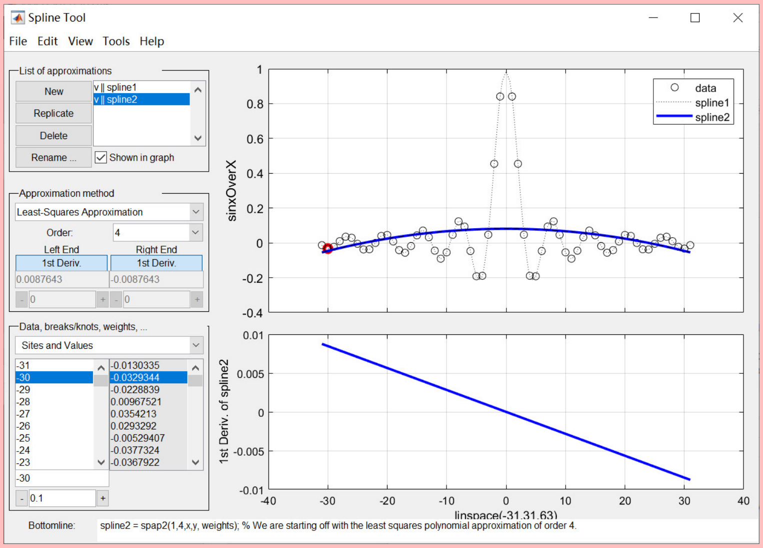
-31	-0.0130335
-30	-0.0329344
-29	-0.0228839
-28	0.00967521
-27	0.0354213
-26	0.0293292
-25	-0.00529407
-24	-0.0377324
-23	-0.0367922

-31

0.1

Bottomline: spline1 = csapi(x,y); % The same as csape(x,y,'not-a-knot') or spline(x,y).

Figure 6: sinx/x initial window



Approximation method: Least-Squares Approximation

Order: 4

Left End	Right End
1st Deriv.	1st Deriv.
0.0087643	-0.0087643
0	0

Data, breaks/knots, weights, ...: Sites and Values

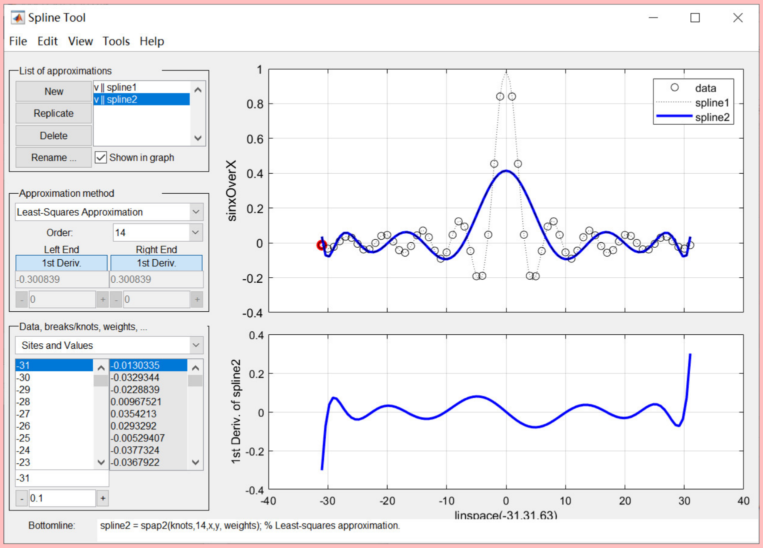
-31	-0.0130335
-30	-0.0329344
-29	-0.0228839
-28	0.00967521
-27	0.0354213
-26	0.0293292
-25	-0.00529407
-24	-0.0377324
-23	-0.0367922

-30

0.1

Bottomline: spline2 = spap2(1,4,x,y, weights); % We are starting off with the least squares polynomial approximation of order 4.

Figure 7: Last-Square Appr, bad



Least-Squares Approximation

Order: 14

Left End: 1st Deriv. -0.300839 — 0

Right End: 1st Deriv. 0.300839 — 0

Sites and Values

-31	-0.0130335
-30	-0.0329344
-29	-0.0228839
-28	0.00967521
-27	0.0354213
-26	0.0293292
-25	-0.00529407
-24	-0.0377324
-23	-0.0367922

-31

0.1

Bottomline:

```
spline2 = spap2(knots,14,x,y, weights); % Least-squares approximation.
```

Figure 8: Last-Square Appr, better