matlabtut3

March 7, 2019

MATLAB Tutorial 3
Outline

- Some warnings
- John Burkardt's website and few examples from there
- Native MATLAB functions

Some warnings

- If you are asked to implement something in assignments, you can not just copy paste from some source and try to pass it like your work! We expect to reasonably original work.
- If you are just asked to solve a problem using some method, you can use anything you want as long as you refer to it!

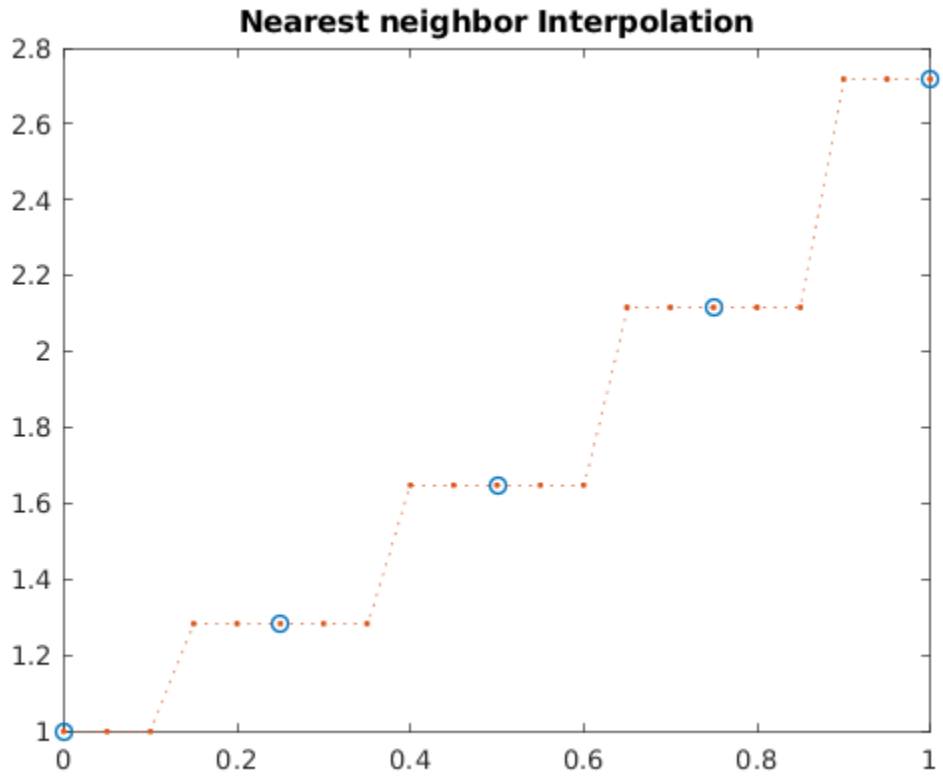
JBurkadt's website
John Burkardt over Florida State University has a good collection MATLAB source files. If you follow the link and look for keywords, you are likely to find something useful. We are going to look at the keyword interpolation.
https://people.sc.fsu.edu/~jburkardt/m_src/m_src.html
Native MATLAB functions

```
In [ ]: x = 0:0.25:1;
        y = exp(x);
        xp = 0:0.05:1;

        yp = interp1(x,y,xp);

        plot(x,y,'o',xp,yp,':.')
        title('Default (Linear) Interpolation')
```

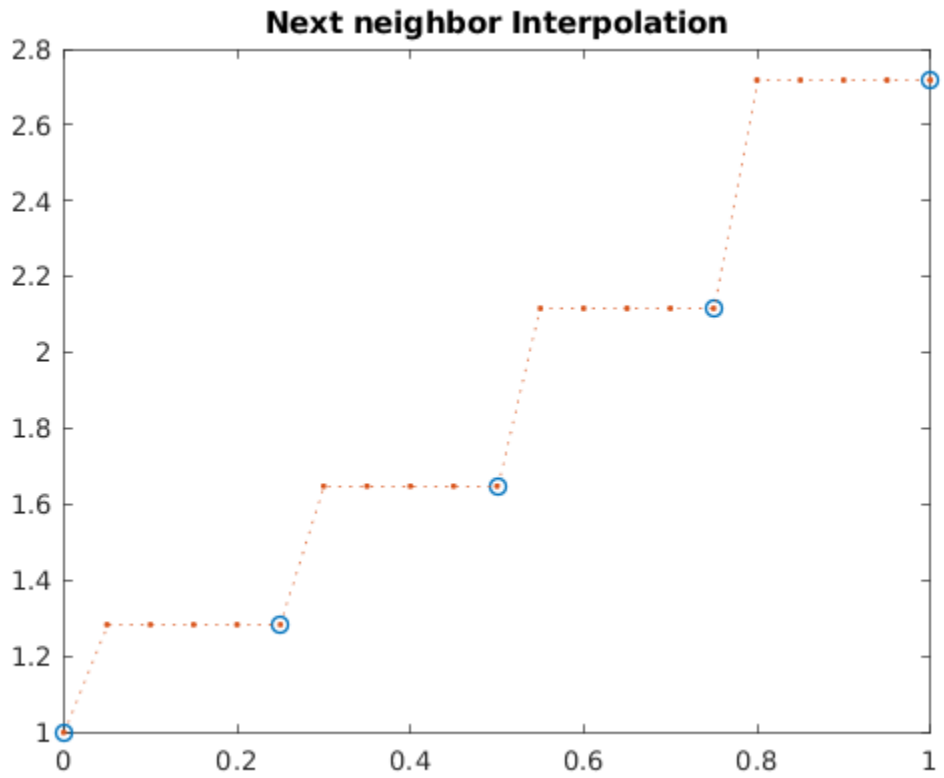
```
Warning: MATLAB has disabled some advanced graphics rendering features by switching to softwar
```

```
In [2]: x = 0:0.25:1;
        y = exp(x);
        xp = 0:0.05:1;
```

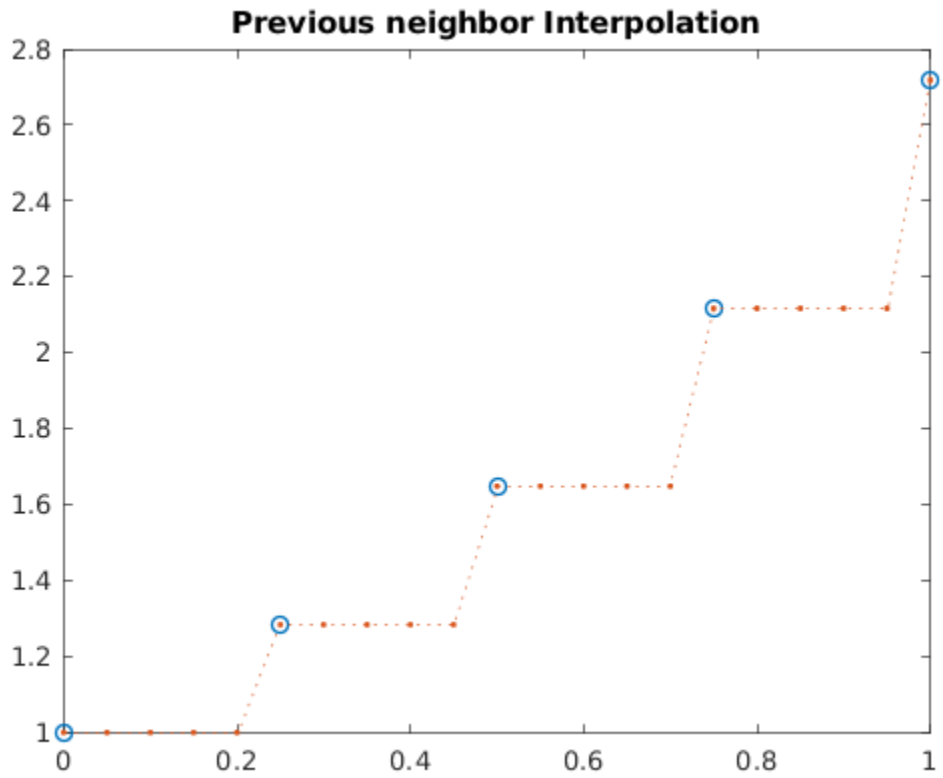
```
yp = interp1(x,y,xp,'nearest');

plot(x,y,'o',xp,yp,':.')
title('Nearest neighbor Interpolation')
```



In [3]:
```
x = 0:0.25:1;
y = exp(x);
xp = 0:0.05:1;

yp = interp1(x,y,xp,'next');

plot(x,y,'o',xp,yp,':.')
title('Next neighbor Interpolation')
```

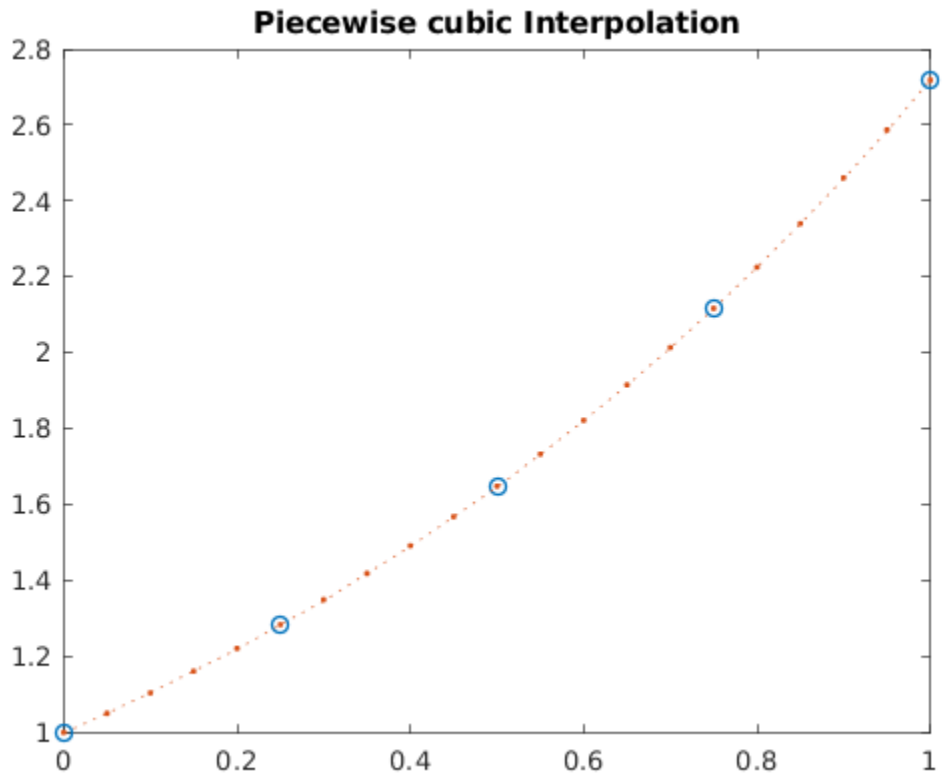



```
In [4]: x = 0:0.25:1;
        y = exp(x);
        xp = 0:0.05:1;

        yp = interp1(x,y,xp,'previous');

        plot(x,y,'o',xp,yp,':.')
        title('Previous neighbor Interpolation')
```

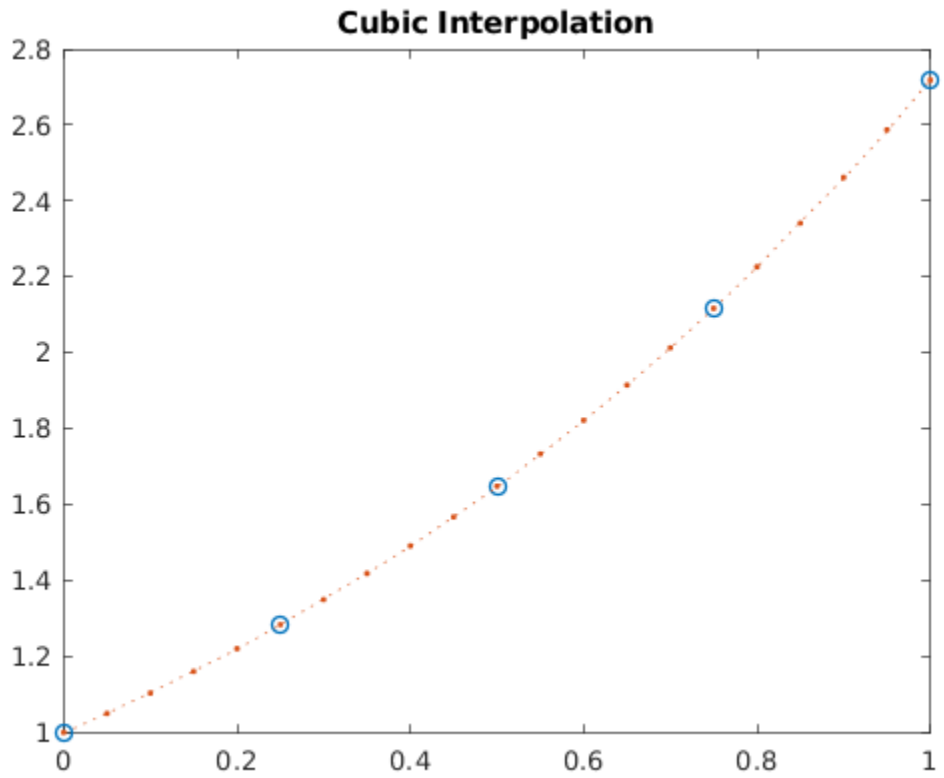



```
In [5]: x = 0:0.25:1;
        y = exp(x);
        xp = 0:0.05:1;

        yp = interp1(x,y,xp,'cubic');

        plot(x,y,'o',xp,yp,':.')
        title('Piecewise cubic Interpolation')
```

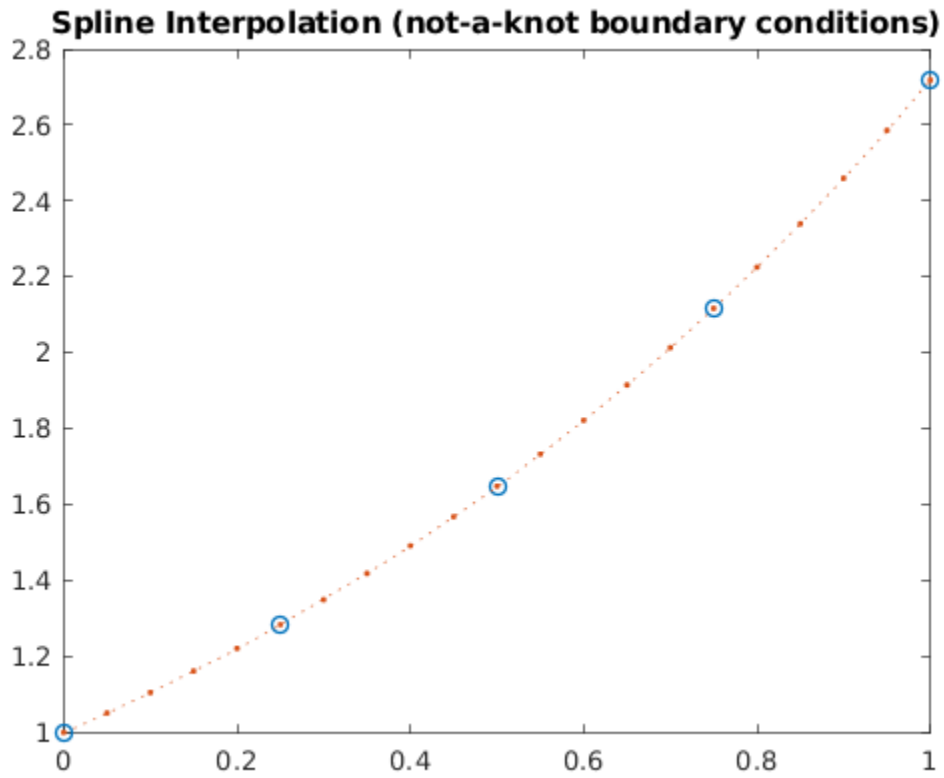
```
Warning: INTERP1(...,'CUBIC') will change in a future release. Use INTERP1(...,'PCHIP') instead
> In interp1>sanitycheckmethod (line 241)
  In interp1>parseinputs (line 376)
  In interp1 (line 78)
```



In [6]:
```
x = 0:0.25:1;
y = exp(x);
xp = 0:0.05:1;

yp = interp1(x,y,xp,'v5cubic');

plot(x,y,'o',xp,yp,':.')
title('Cubic Interpolation')
```

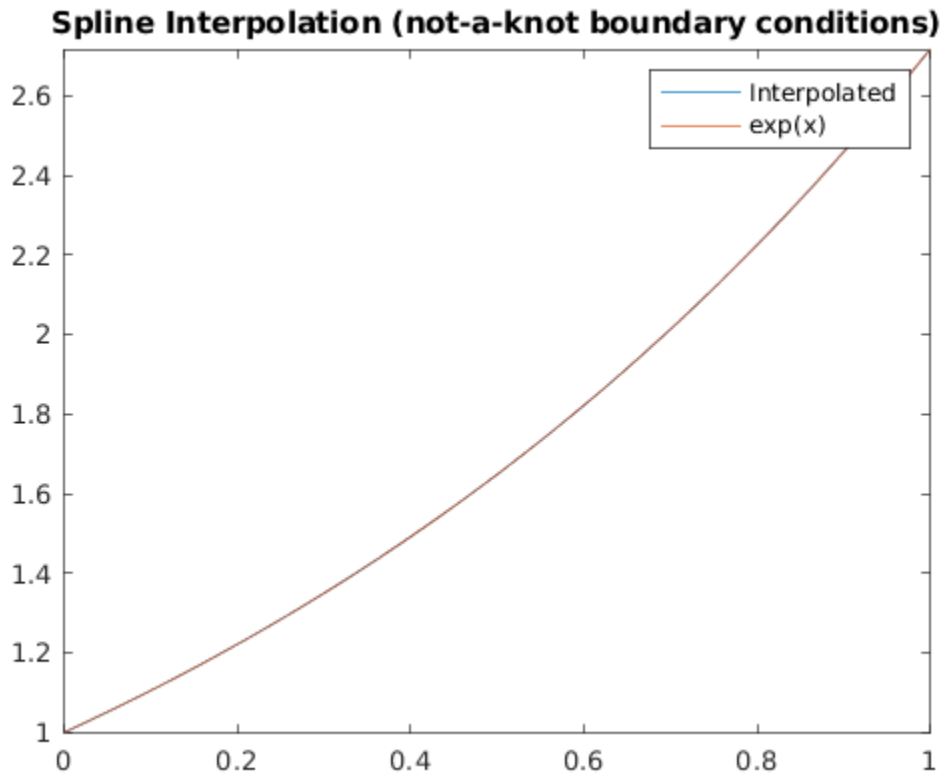


In [7]:
```
x = 0:0.25:1;
y = exp(x);
xp = 0:0.05:1;

yp = interp1(x,y,xp,'spline');

plot(x,y,'o',xp,yp,':.')
title('Spline Interpolation (not-a-knot boundary conditions)')
```

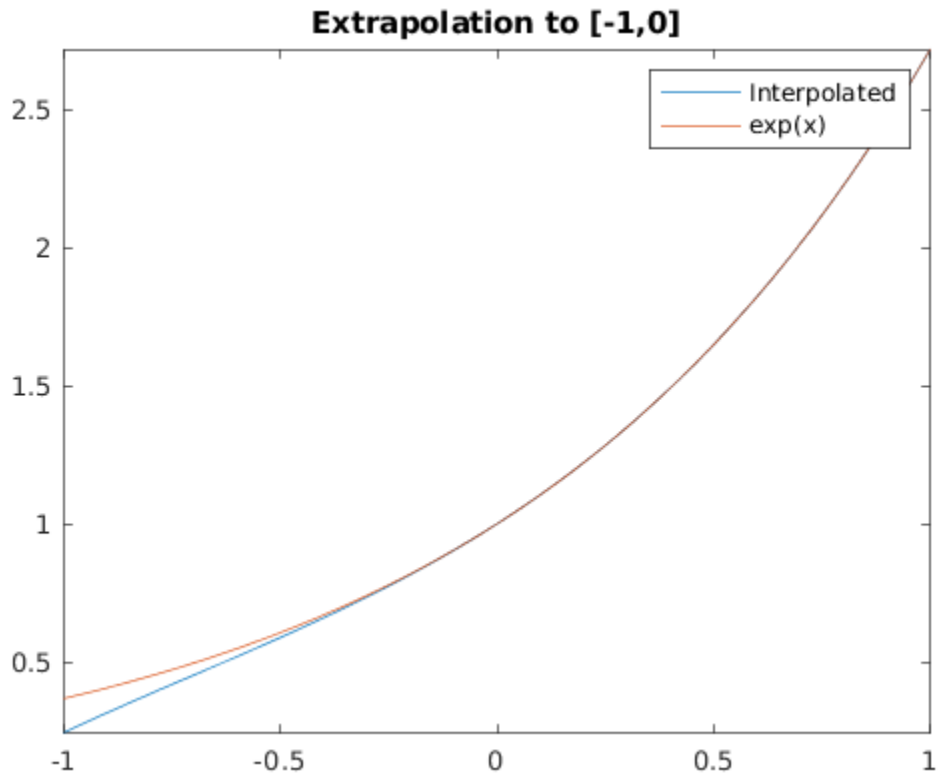


Rather than getting interpolated values at given points, we can ask for polynomial to be returned. This might be useful sometimes. For example, if you are going to use the polynomial later for some other reason.

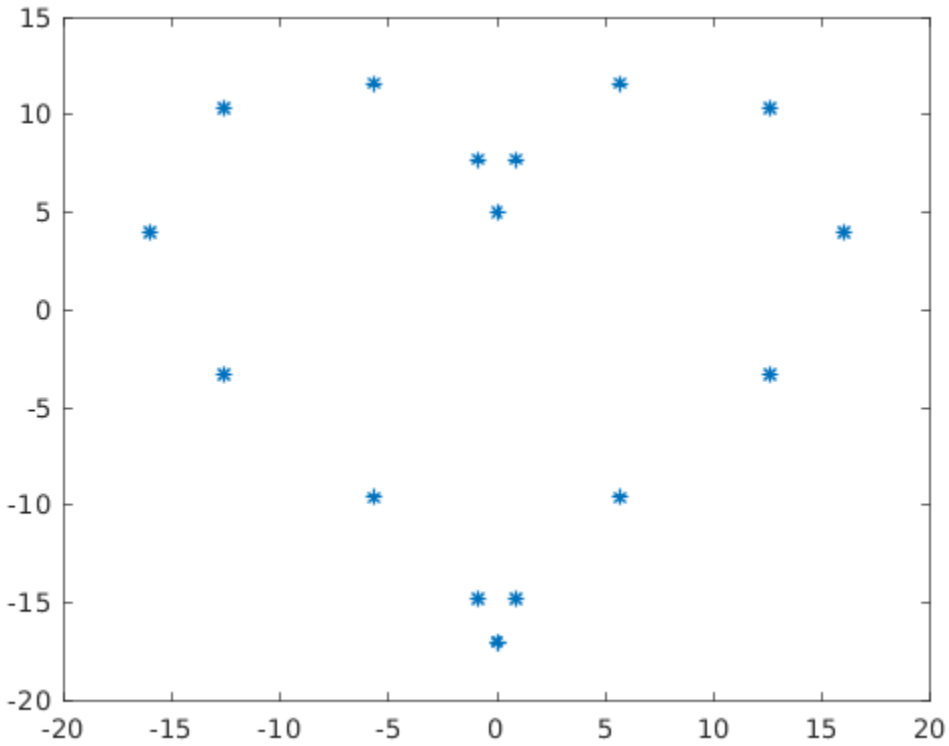
```
In [8]: x = 0:0.25:1;
        y = exp(x);
        xp = 0:0.05:1;

        pp = interp1(x,y,'spline','pp');

        fplot(@(x) ppval(pp,x),[0,1])
        hold on
        fplot(@(x) exp(x),[0,1])
        title('Spline Interpolation (not-a-knot boundary conditions)')
        legend('Interpolated','exp(x)')
```



In [9]:
```
x = 0:0.25:1;
y = exp(x);
xp = 0:0.05:1;

pp = interp1(x,y,'spline','pp');

fplot(@(x) ppval(pp,x),[-1,1])
hold on
fplot(@(x) exp(x),[-1,1])
title('Extrapolation to [-1,0]')
legend('Interpolated','exp(x)')
```

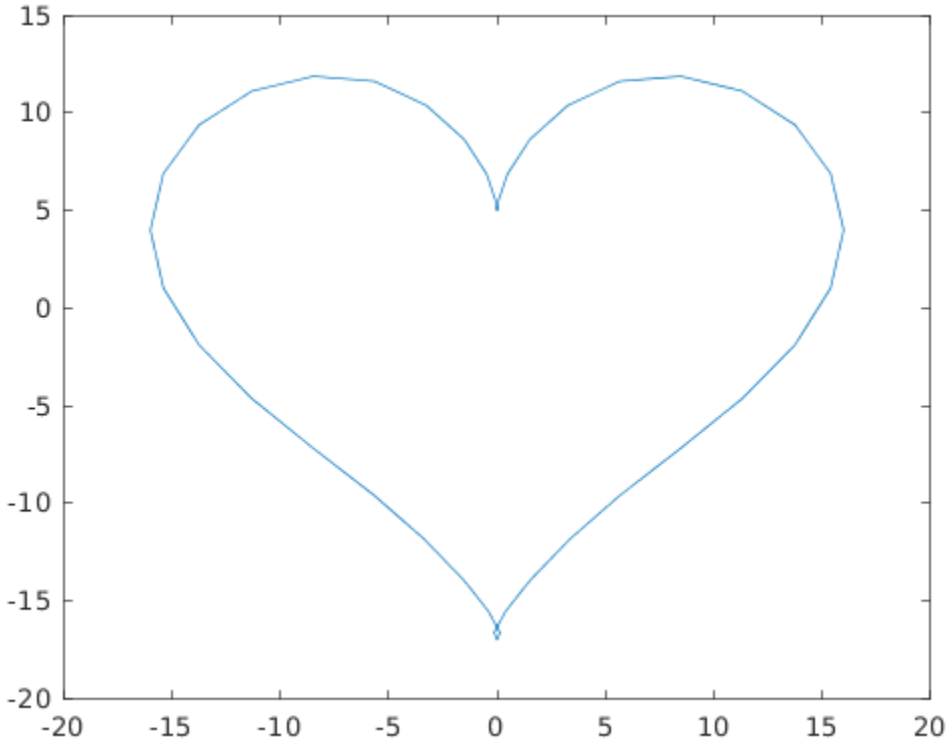



```
In [10]: t = -pi:pi/8:pi;
         x = 16*sin(t).^3;
         y = 13*cos(t) - 5*cos(2*t) - 2*cos(3*t) - cos(4*t);

         plot(x,y,'*')
```



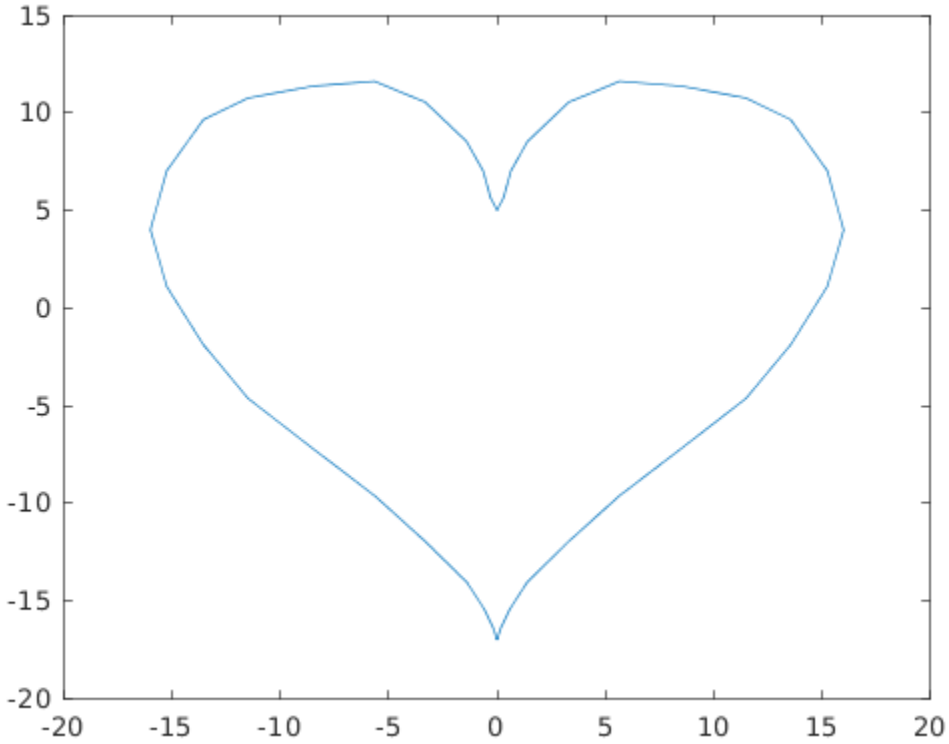
'spline' uses not-a-knot conditions so it is not appropriate to use to interpolate for this case. Below we demonstrate that.

```
In [11]: xpp = interp1(t,x,'spline','pp');
         ypp = interp1(t,y,'spline','pp');

         plot(ppval(xpp,[-pi:pi/20:pi]),ppval(ypp,[-pi:pi/20:pi]))
```



We can interpolate both x and y at the same (and it is faster than interpolating individually). We use 'pchip' as it is the correct choice for this problem.

```
In [12]: pp = interp1(t,[x',y'],'pchip','pp');

         heart = ppval(pp,[-pi:pi/20:pi]);
         plot(heart(:,1),heart(:,2))
```



Also check

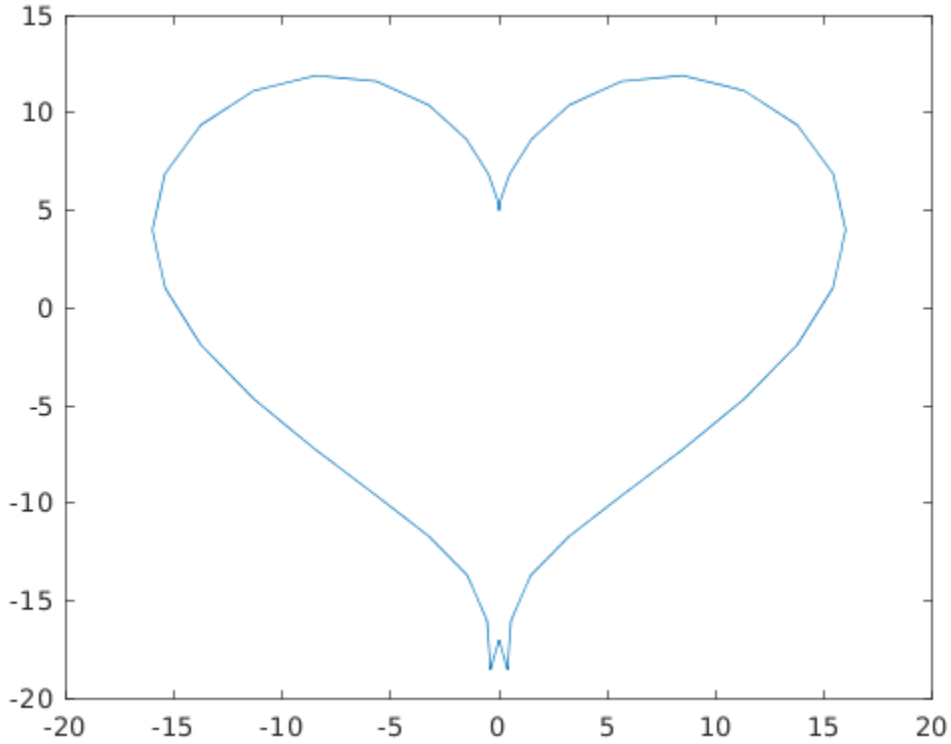
https://www.mathworks.com/help/matlab/ref/spline.html which is equivalent to calling interp1 with the option 'spline' and

https://www.mathworks.com/help/matlab/ref/pchip.html which is equivalent to calling interp1 with the option 'pchip'.

If you want to exclusively use either of these methods, the good practice is to use these commands directly so that your code is more readable.

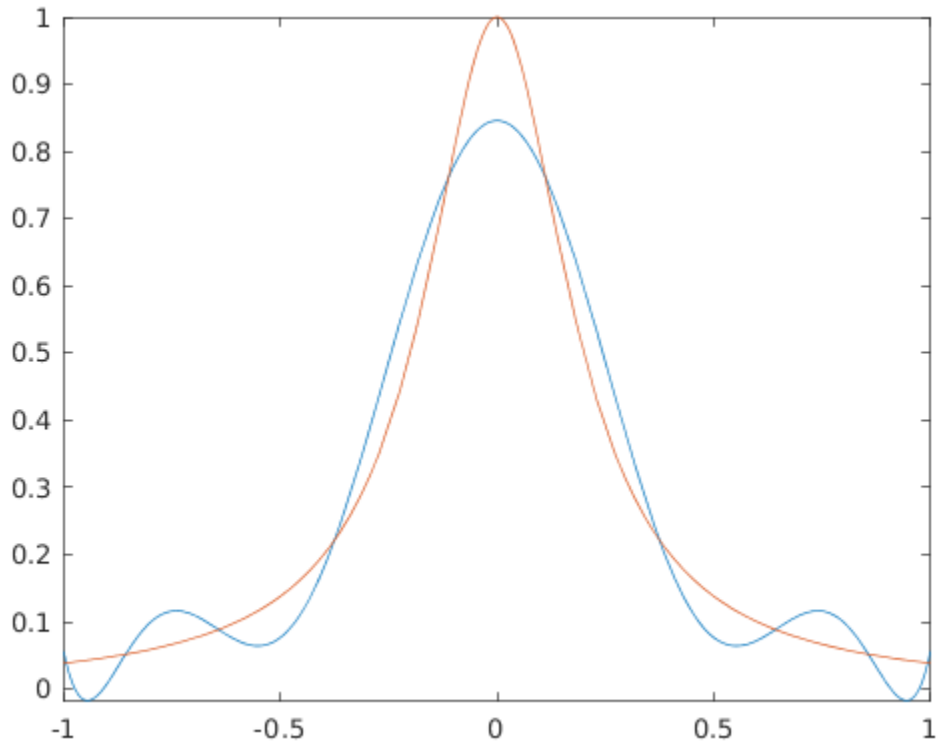
Now we try to fit single high degree polynomials to x and y coordinates.

```
In [13]: xp = polyfit(t,x,14);
         yp = polyfit(t,y,14);

         heartx = polyval(xp, [-pi:pi/20:pi]);
         hearty = polyval(yp, [-pi:pi/20:pi]);

         plot(heartx, hearty)
```



Below is the demonstration of Runge's phenomenon. It can be fixed using a different (e.g. Chebyshev) interpolation technique.

```
In [14]: f = @(x) 1./(1+25*x.^2);
         x = [-1:2/25:1];

         up = polyfit(x,f(x),9);

         fplot(@(x) polyval(up,x),[-1,1])
         hold on
         fplot(f,[-1,1])
```


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-1
-0.5
0
0.5
1