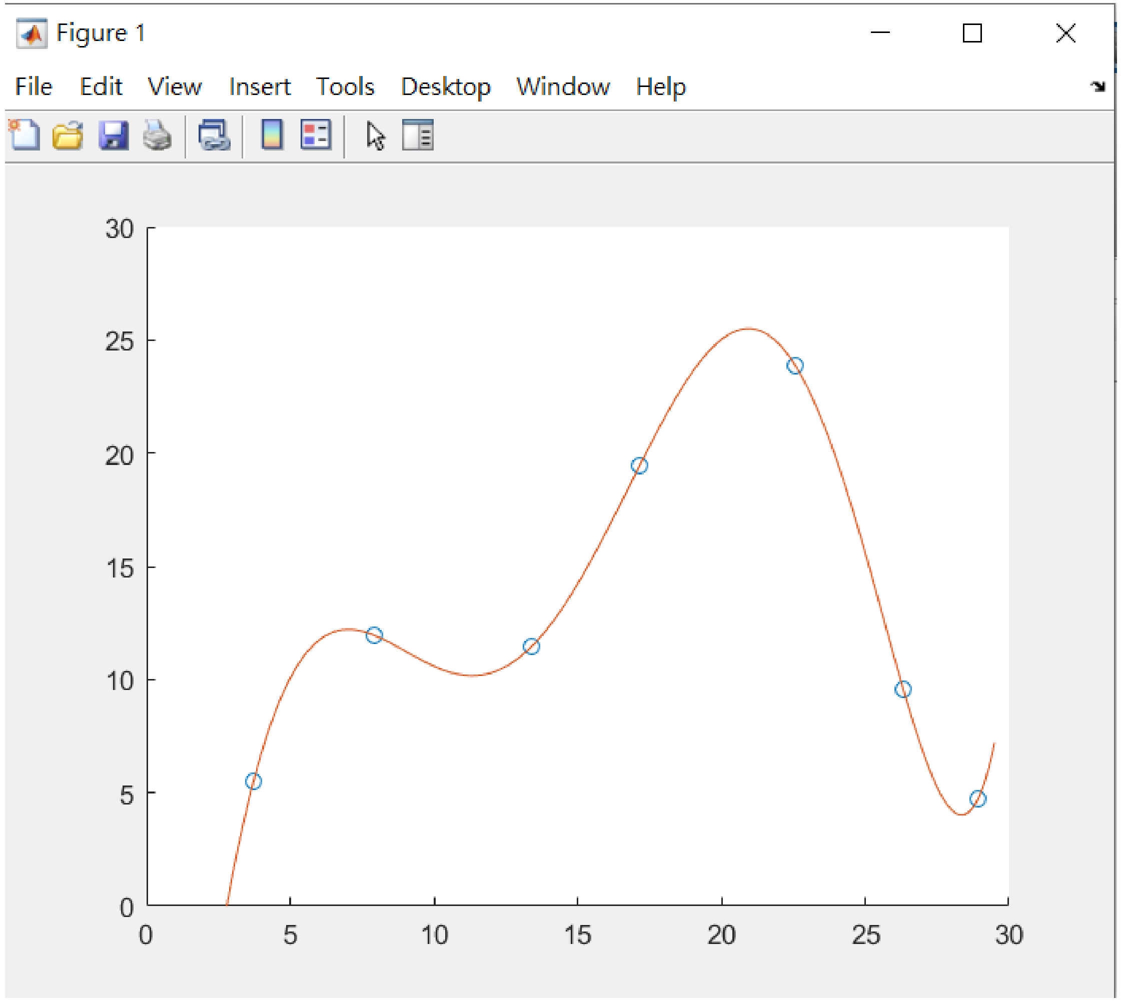
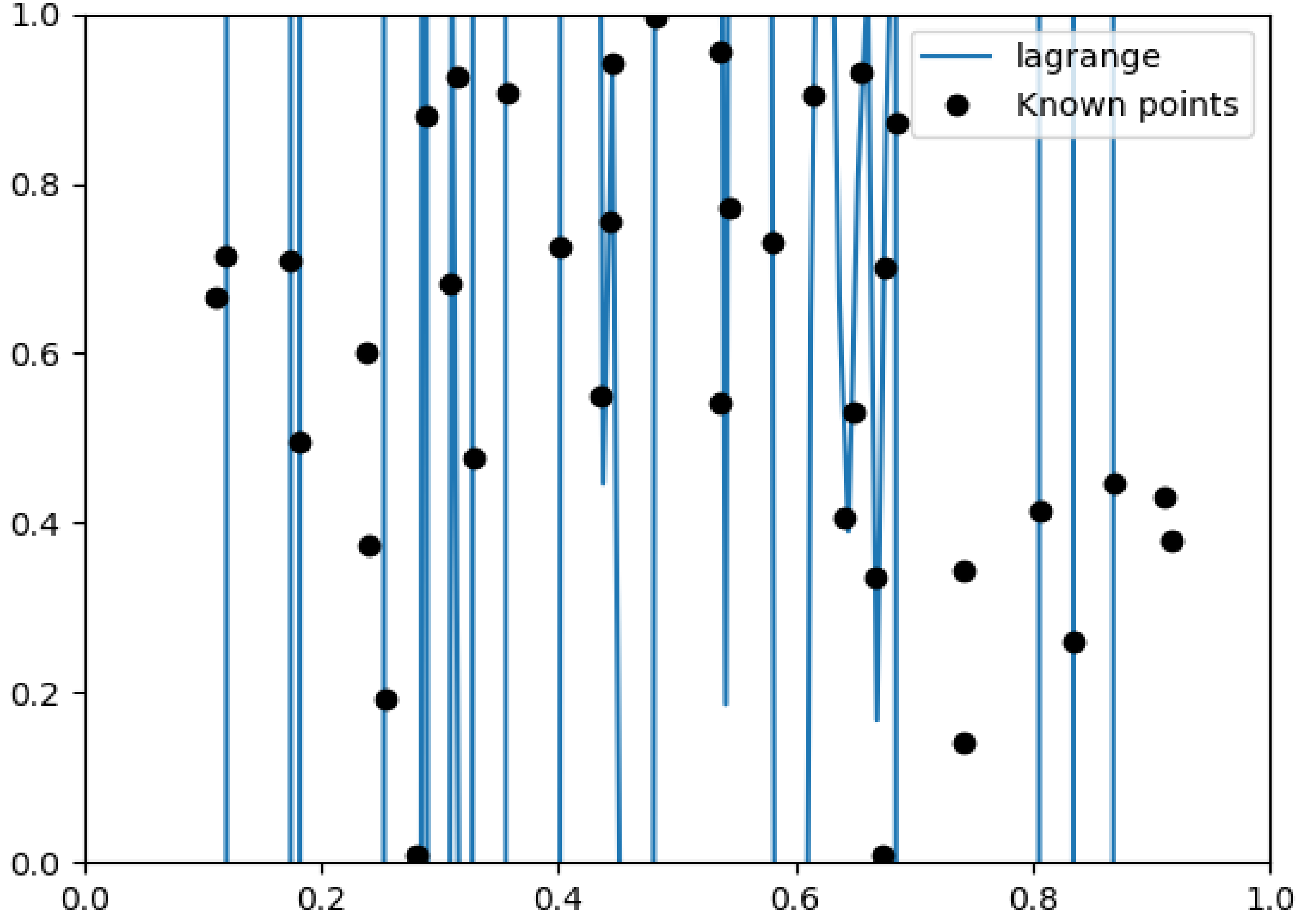
Interpolating Polynomial

- Power form
- $P(x) = c_1 x^{n-1} + c_2 x^{n-2} + \cdots + c_{n-1} x + c_n$
- $P(x_k) = y_k, k = 1 \ldots n$
- Solve

$$\begin{pmatrix} {x_1}^{n-1} & {x_1}^{n-2} & \cdots & 1 \\ {x_2}^{n-1} & {x_2}^{n-2} & \cdots & 1 \\ \vdots & & \ddots & \vdots \\ {x_n}^{n-1} & {x_n}^{n-2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$



Use Lagrange method to plot hand



Newton interpolation

- f(x)=$[y_0] + [y_0, y_1](x - x_0) + \cdots + [y_0, \ldots, y_k](x - x_0)(x - x_1) \ldots (x - x_{k-1})$
- $[y_0] = y_0$

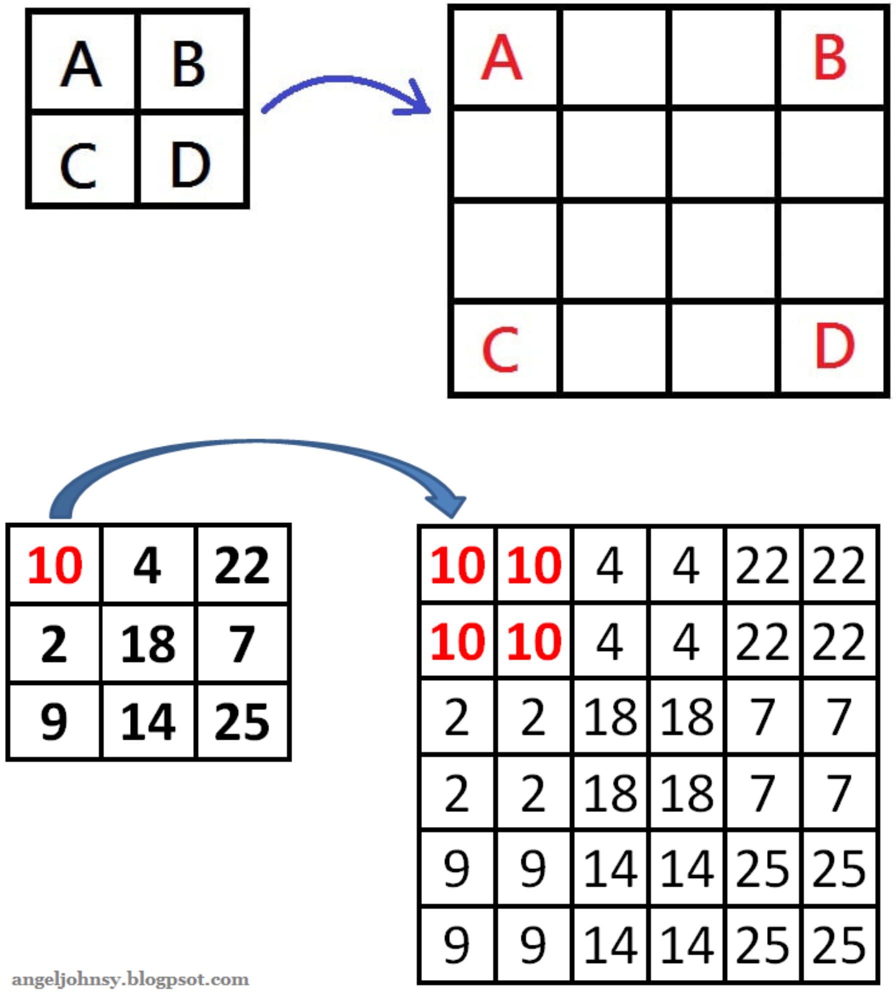
  $[y_0, y_1]$=$\frac{y_1 - y_0}{x_1 - x_0}$

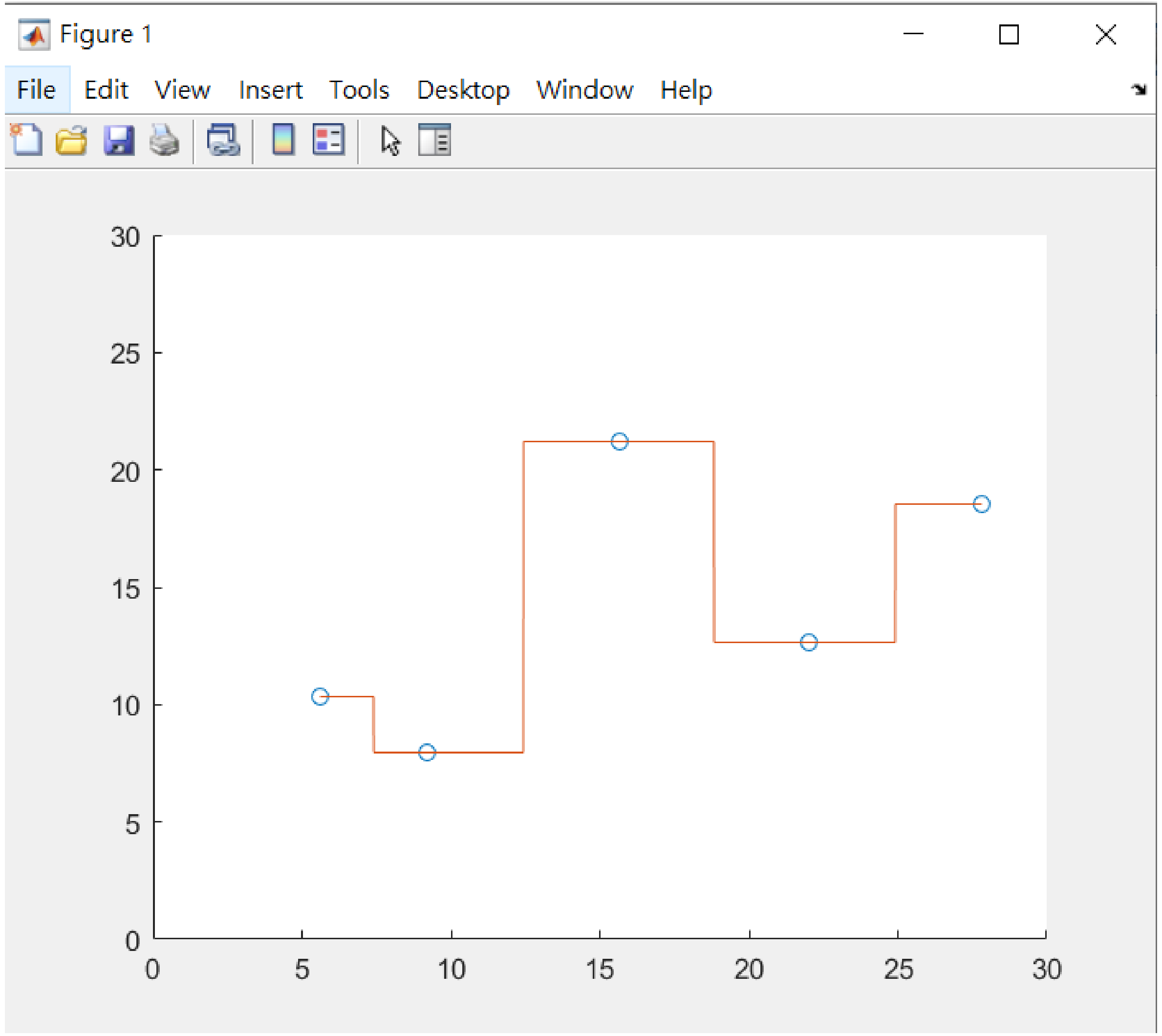
  $[y_0, y_1, y_2] = \frac{[y_1, y_2] - [y_0, y_1]}{x_2 - x_0}$

  $[y_0, y_1, y_2, y_3]$=$\frac{[y_{1,} y_2, y_3] - [y_0, y_1, y_2]}{x_3 - x_0}$

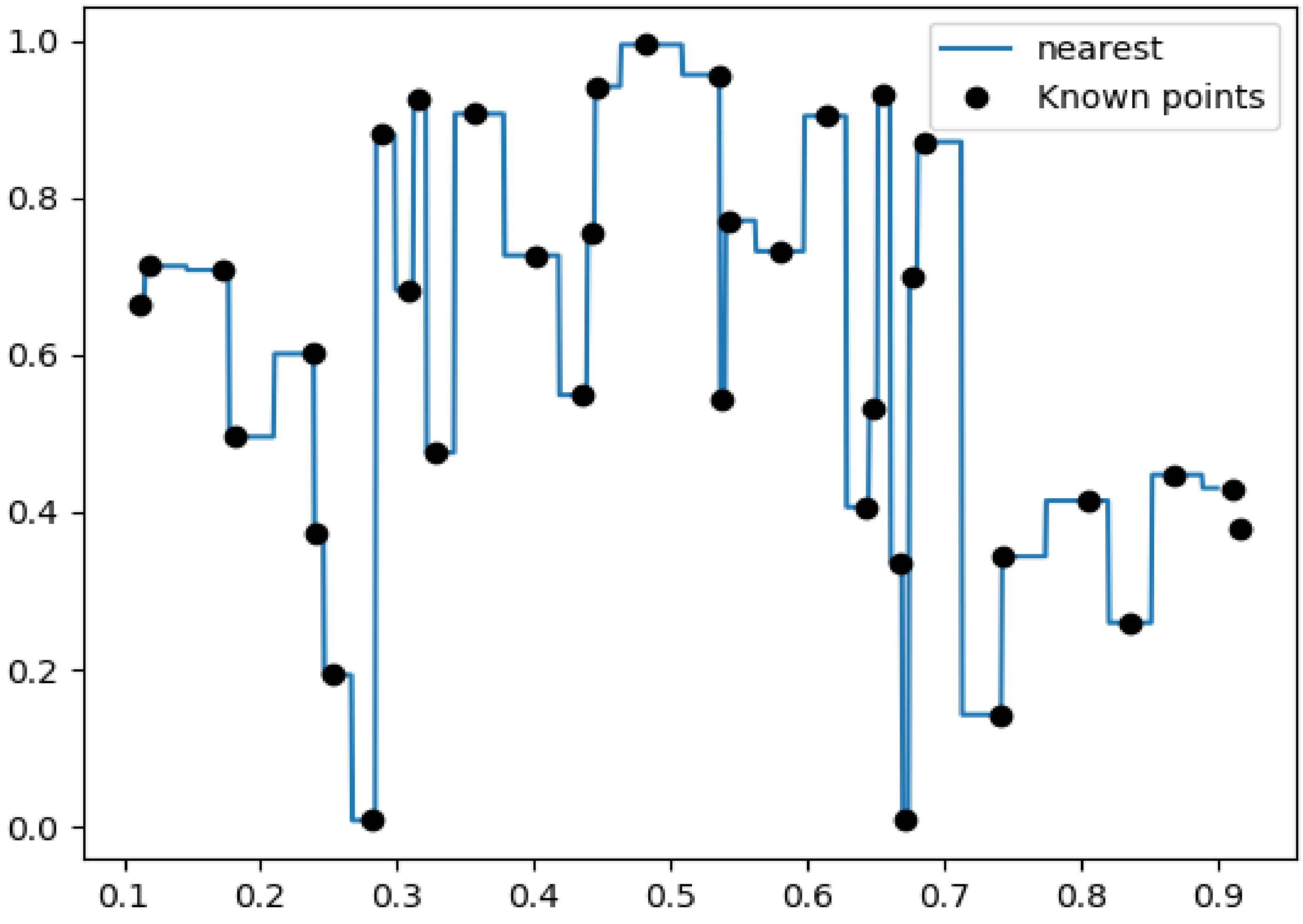
  $[y_0, y_1, y_2, y_3, \ldots., y_n] = \frac{[y_1, y_{2,\ldots,} y_n] - [y_0, y_{1,\ldots,} y_{n-1}]}{x_n - x_0}$

Nearest neighbor interpolation



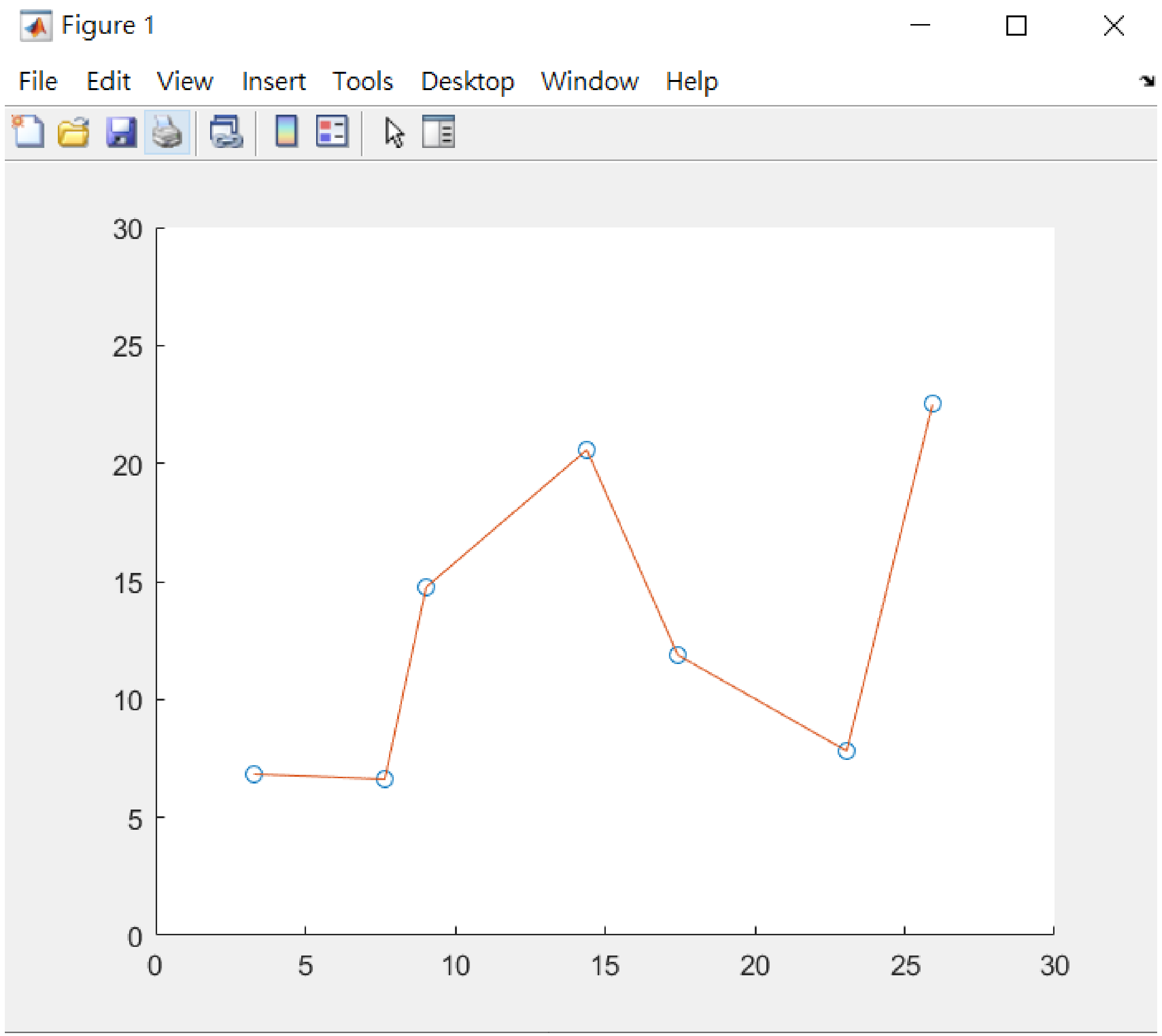


Use nearest method to plot hand

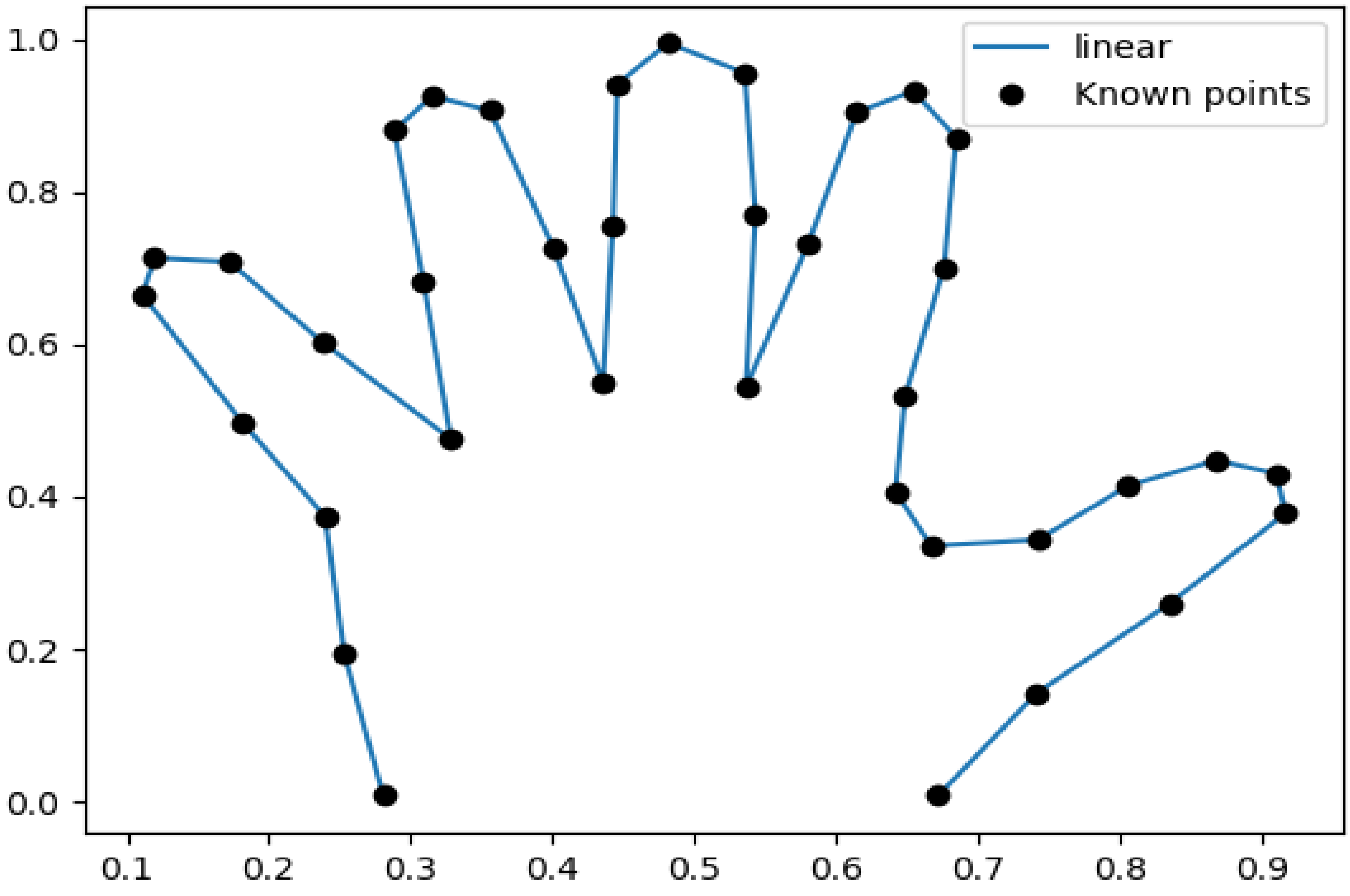


Piecewise linear interpolation

- $x_k \le x < x_{k+1}$
- $s = x - x_k$
- $\delta_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$
- $L(x) - y_k = \delta_k(x - x_k) = \delta_k * s$



Use linear method to plot hand



Piecewise Cubic Hermite Interpolation

- Let h_k denote the length of the kth subinterval:
$$h_k = x_{k+1} - x_k$$
- Then the first divided difference, δ_k, is given by
$$\delta_k = \frac{y_{k+1} - y_k}{h_k}$$
- Let d_k denote the slope of the interpolant at x_k :
$$d_k = P'(x_k)$$

```
function v = splinetx(x,y,u)
% First derivatives
  h = diff(x);
  delta = diff(y)./h;
  d = splineslopes(h,delta);
function v = pchiptx(x,y,u)
% First derivatives
  h = diff(x);
  delta = diff(y)./h;
  d = pchipslopes(h,delta);
```

Piecewise Cubic Hermite Interpolation

- Once the slopes have been computed, the interpolant can be efficiently evaluated using the power form with the local variable s:

$$P(x) = y_k + sd_k + s^2 c_k + s^3 b_k$$

where the coefficients of the quadratic and cubic terms are

$$c_k = \frac{3\delta_k - 2d_k - d_{k+1}}{h}$$

$$b_k = \frac{d_k - 2\delta_k + d_{k+1}}{h}$$

```
function v = splinetx(x,y,u)
% Piecewise polynomial coefficients
  n = length(x);
  c = (3*delta - 2*d(1:n-1) - d(2:n))./h;
  b = (d(1:n-1) - 2*delta + d(2:n))./h.^2;
function v = pchiptx(x,y,u)
% Piecewise polynomial coefficients
  n = length(x);
  c = (3*delta - 2*d(1:n-1) - d(2:n))./h;
  b = (d(1:n-1) - 2*delta + d(2:n))./h.^2;
```

Spline Interpolation

- The two functions differ in the way they compute the slopes, d_k.

With the two end conditions included, we have n linear equations in n unknowns:

$$Ad = r.$$

The vector of unknown slopes is

$$d = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{pmatrix}.$$

The coefficient matrix A is tridiagonal:

$$A = \begin{pmatrix} h_2 & h_2 + h_1 & & & & \\ h_2 & 2(h_1 + h_2) & h_1 & & & \\ & h_3 & 2(h_2 + h_3) & h_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} \\ & & & & h_{n-1} + h_{n-2} & h_{n-2} \end{pmatrix}.$$

```
% Diagonals of tridiagonal system
n = length(h)+1;
a = zeros(size(h)); b = a; c = a; r = a;
a(1:n-2) = h(2:n-1);
a(n-1) = h(n-2)+h(n-1);
b(1) = h(2);
b(2:n-1) = 2*(h(2:n-1)+h(1:n-2));
b(n) = h(n-2);
c(1) = h(1)+h(2);
c(2:n-1) = h(1:n-2);
```

Spline Interpolation

- The right-hand side is

$$r = 3\begin{pmatrix} r_1 \\ h_1\delta_2 + h_2\delta_1 \\ h_2\delta_3 + h_3\delta_2 \\ \vdots \\ h_{n-2}\delta_{n-1} + h_{n-1}\delta_{n-2} \\ r_n \end{pmatrix}$$

```
% Right-hand side
  r(1) = ((h(1)+2*c(1))*h(2)*delta(1)+h(1)^2*delta(2))/c(1);
  r(2:n-1) = 3*(h(2:n-1).*delta(1:n-2)+h(1:n-2).*delta(2:n-1));
  r(n) = (h(n-1)^2*delta(n-2)+(2*a(n-1)+h(n-1))*h(n-2)*delta(n-1))/a(n-1);
```

Pchip Interpolation

- The two functions differ in the way they compute the slopes, d_k.
- If δ_k and δ_{k-1} have the same sign, but the two intervals have different lengths, then d_k is a weighted harmonic mean

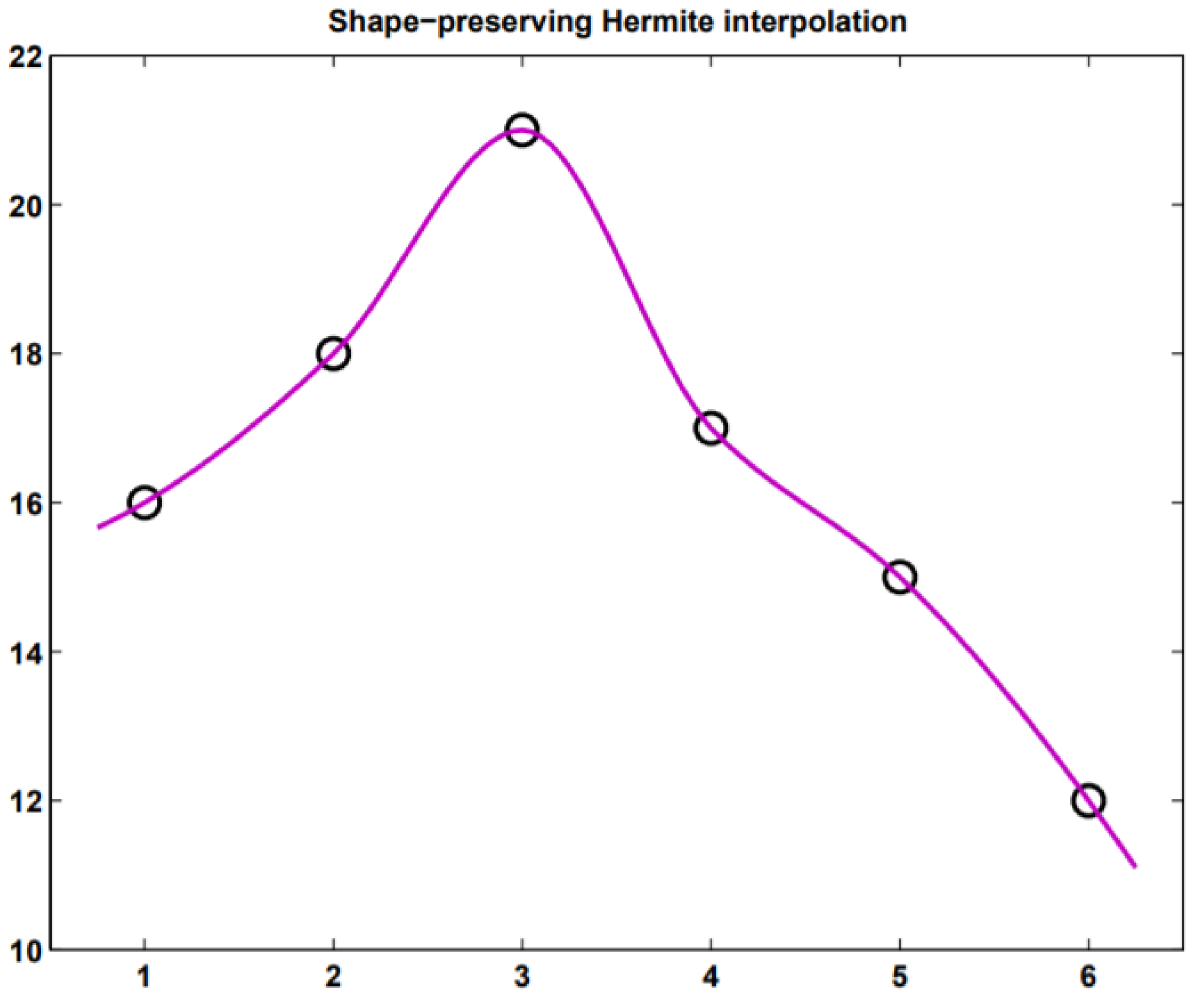
$$\frac{w_1 + w_2}{d_k} = \frac{w_1}{\delta_{k-1}} + \frac{w_2}{\delta_k},$$

where

$$w_1 = 2h_k + h_{k-1}$$
$$w_2 = h_k + 2h_{k-1}$$

```
% delta(k) if they have the same sign.
n = length(h)+1;
d = zeros(size(h));
k = find(sign(delta(1:n-2)).*...
                    sign(delta(2:n-1))>0)+1;
w1 = 2*h(k)+h(k-1);
w2 = h(k)+2*h(k-1);
d(k) = (w1+w2)./...
          (w1./delta(k-1) + w2./delta(k));
```

Pchip Interpolation



- If δ_k and δ_{k-1} have the opposite sign , d_k=0

- If δ_k and δ_{k-1} have the same sign, two intervals have same lengths, $d_k = \frac{2}{(\frac{1}{\delta_{k-1}} - \frac{1}{\delta_k})}$

Pchip Interpolation

- With periodicity, these formulas can also be used at the endpoints where k = 1 and k = n because

$$\delta_0 = \delta_{n-1}\,, \delta_1 = \delta_n$$

```
% Slopes at endpoints
d(1) = pchipend(h(1),h(2),delta(1),delta(2));
d(n) = pchipend(h(n-1),h(n-2),delta(n-1),delta(n-2));
```

Remark

- spline produces a smoother result.
- spline produces a more accurate result if the data consists of values of a smooth function.
- pchip has no overshoots and less oscillation if the data are not smooth.
- pchip is less expensive to set up.
- The two are equally expensive to evaluate.

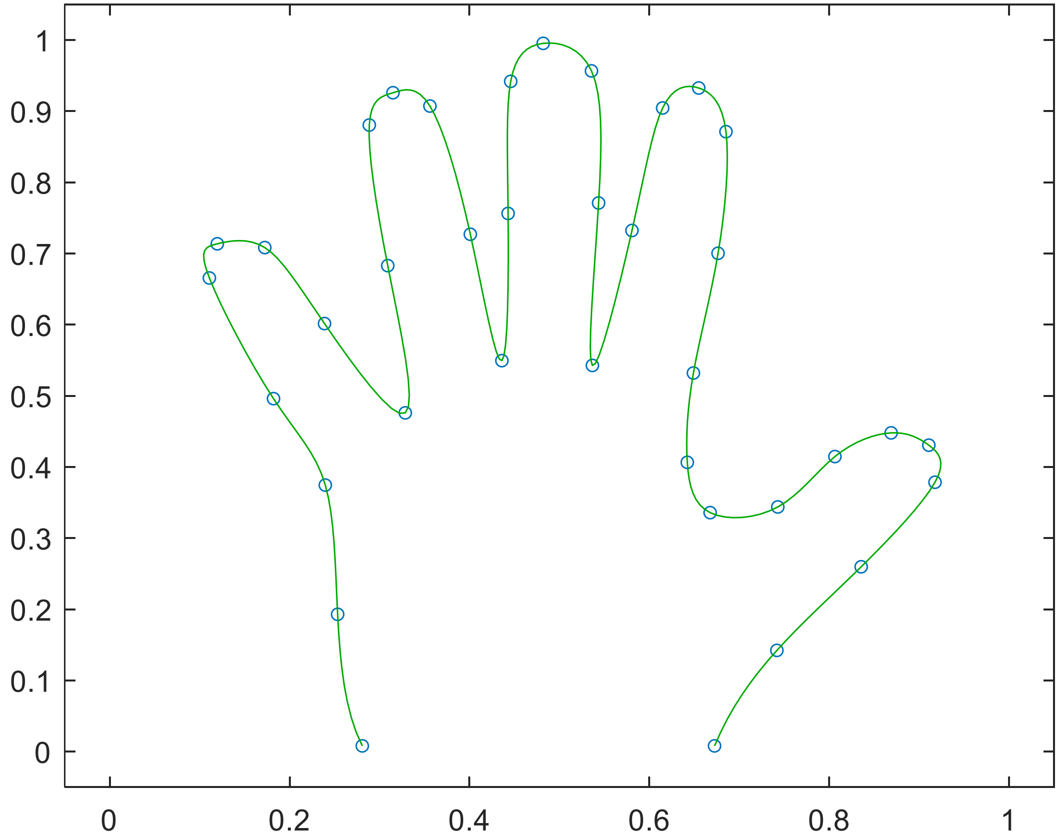
Hand (Exer. 3.4, Spline Interp., Cartesian Coor.)

% Load data.

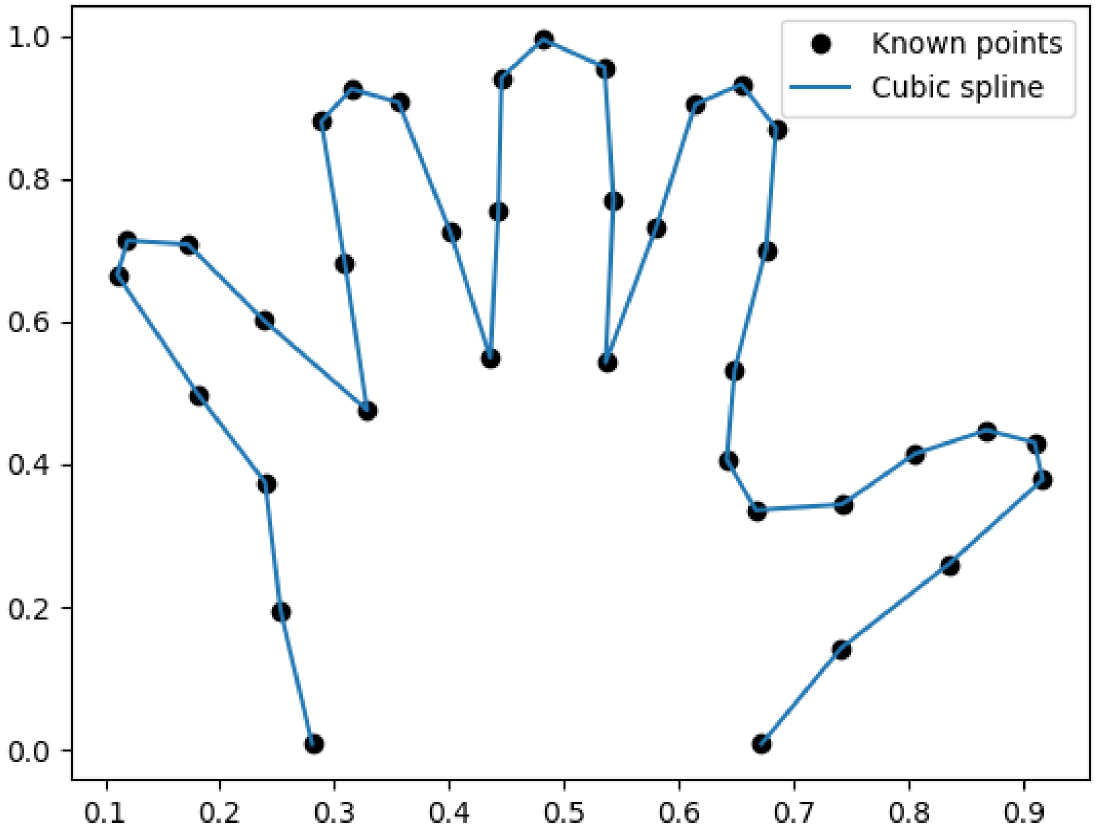
```
load myhand.dat
x = myhand(:,1);
y = myhand(:,2);
n = length(x);
```

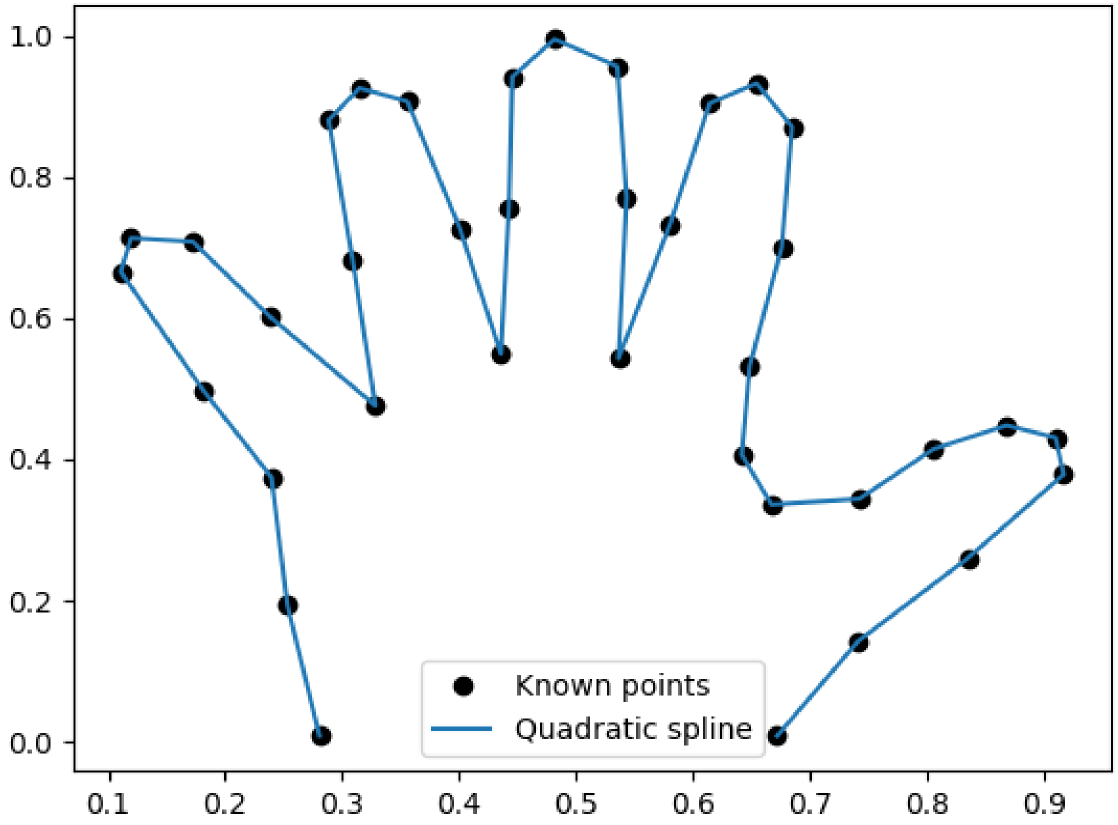
% Plot

```
p = plot(x,y,‘o’,u,v,‘-’); %點標、線型
set(p(1),‘markersize’,4) %點標大小
set(p(2),'color',[0 2/3 0]) %RGB
axis([-.05 1.05 -.05 1.05]) %兩軸範圍
```



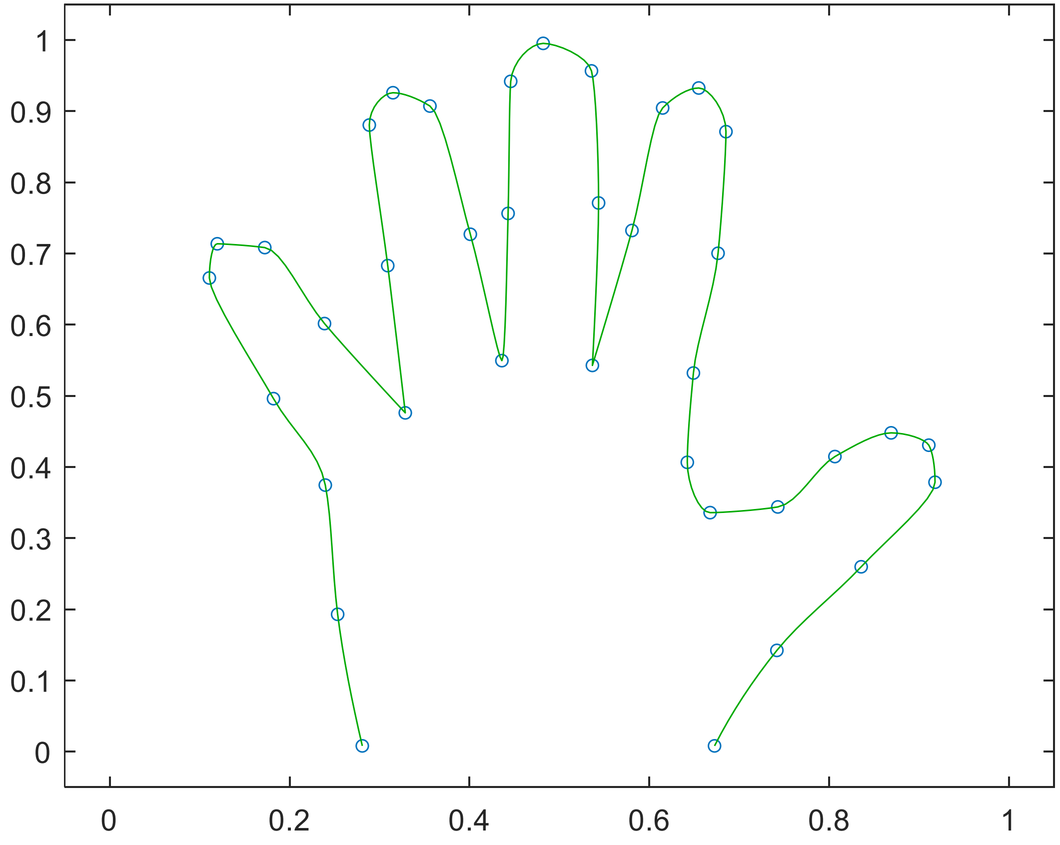
Cubic Spline and Quadratic Spline



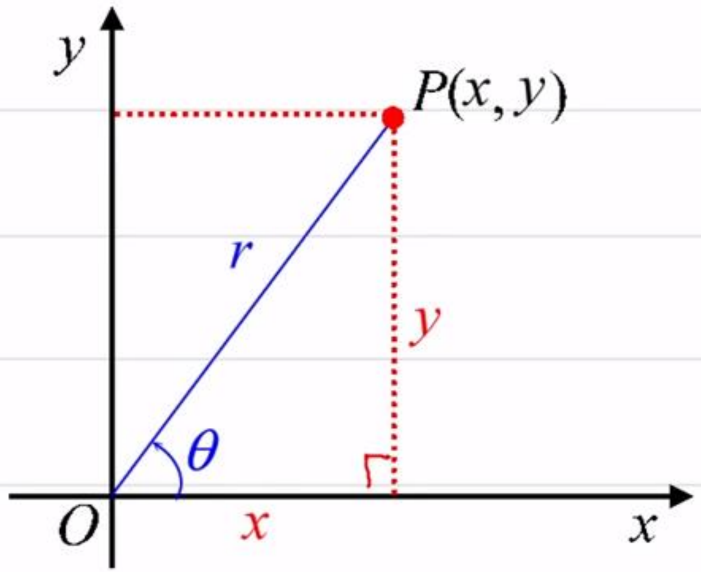


Hand (Exer. 3.4, Pchip Interp., Cartesian Coor.)

```
% Load data.
  load myhand.dat
  x = myhand(:,1);
  y = myhand(:,2);
  n = length(x);
% Plot
  p = plot(x,y,‘o’,u,v,‘-’); %點標、線型
  set(p(1),‘markersize’,4) %點標大小
  set(p(2),'color',[0 2/3 0]) %RGB
  axis([-.05 1.05 -.05 1.05]) %兩軸範圍
```



Polar Coordinates



$$\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$$

$$\begin{cases} x^2 + y^2 = r^2 \\ \tan\theta = \dfrac{y}{x} \end{cases}$$

```
% Translate to make starlike w.r.t. origin.
  x0 = (x(1) + x(n))/2;
  y0 = (y(1) + y(n))/2;
  x = x - x0;
  y = y - y0;
  theta = atan2(y,x);
  r = sqrt(x.^2 + y.^2);
```

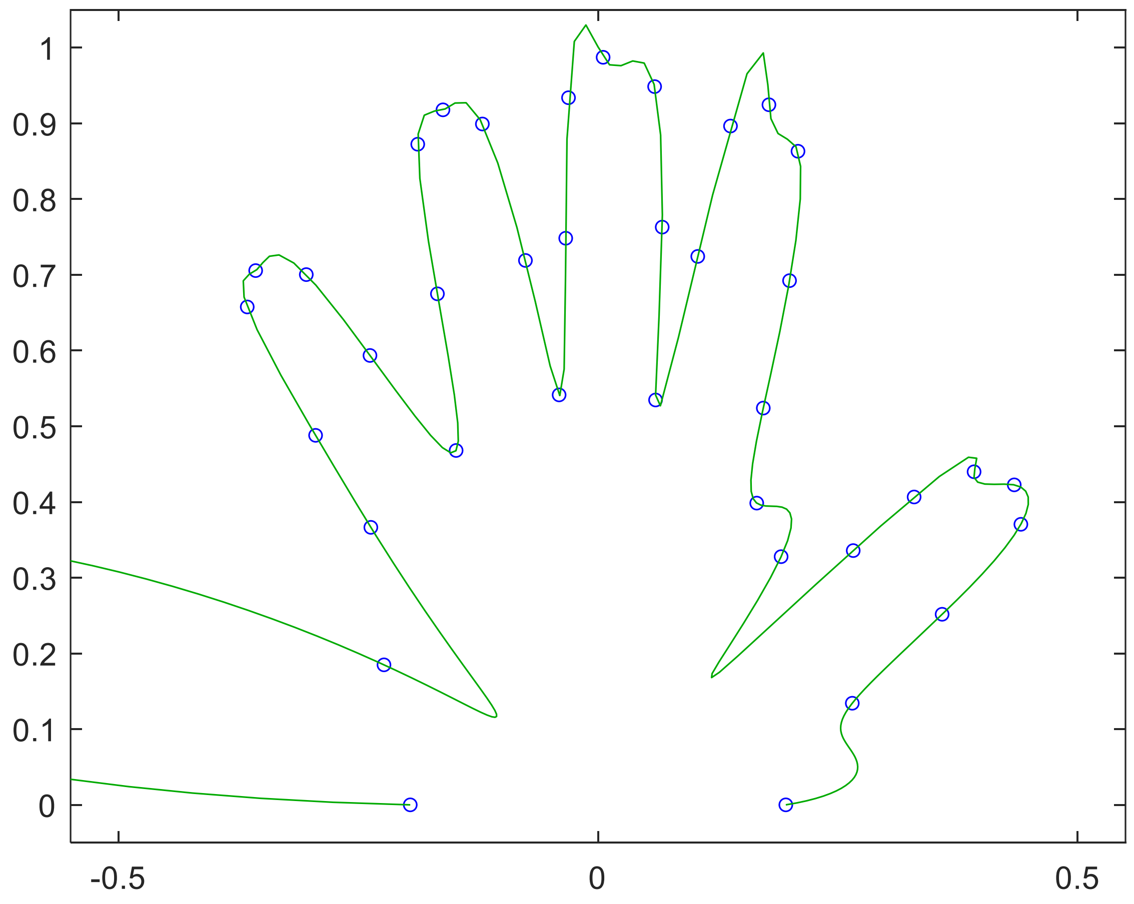
Hand (Exer. 3.5, Spline Interp., Polar Coor.)

% Load data.

```
load myhand.dat
x = myhand(:,1);
y = myhand(:,2);
n = length(x);
```

% Plot

```
S = splinetx(theta,r,t).*exp(i*t);
p = plot(x,y,‘o’,real(S),imag(S),‘k-’);
set(p(1),‘markersize‘,4) %點標大小
set(p(2),‘color’,[0 2/3 0]) %RGB
axis([-.55 .55 -.05 1.05]) %兩軸範圍
```



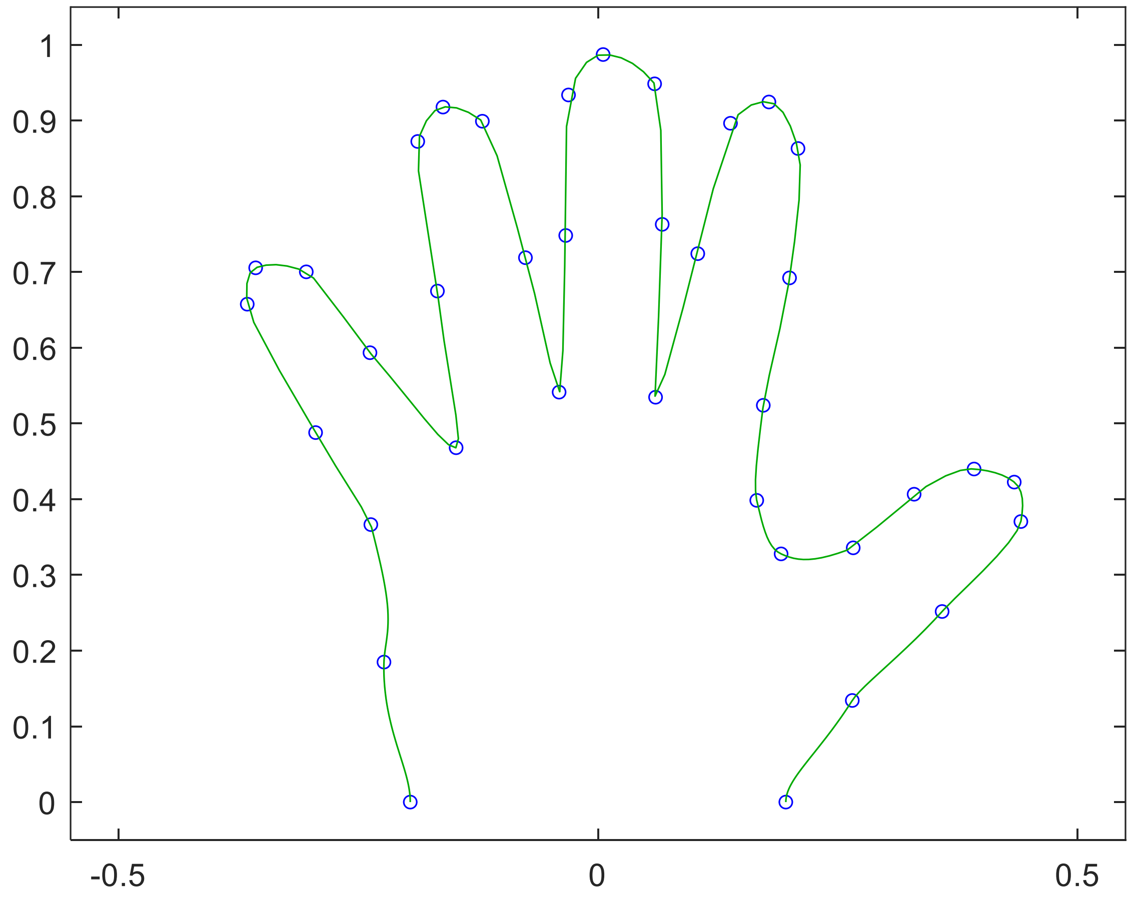
Hand (Exer. 3.5, Pchip Interp., Polar Coor.)

% Load data.

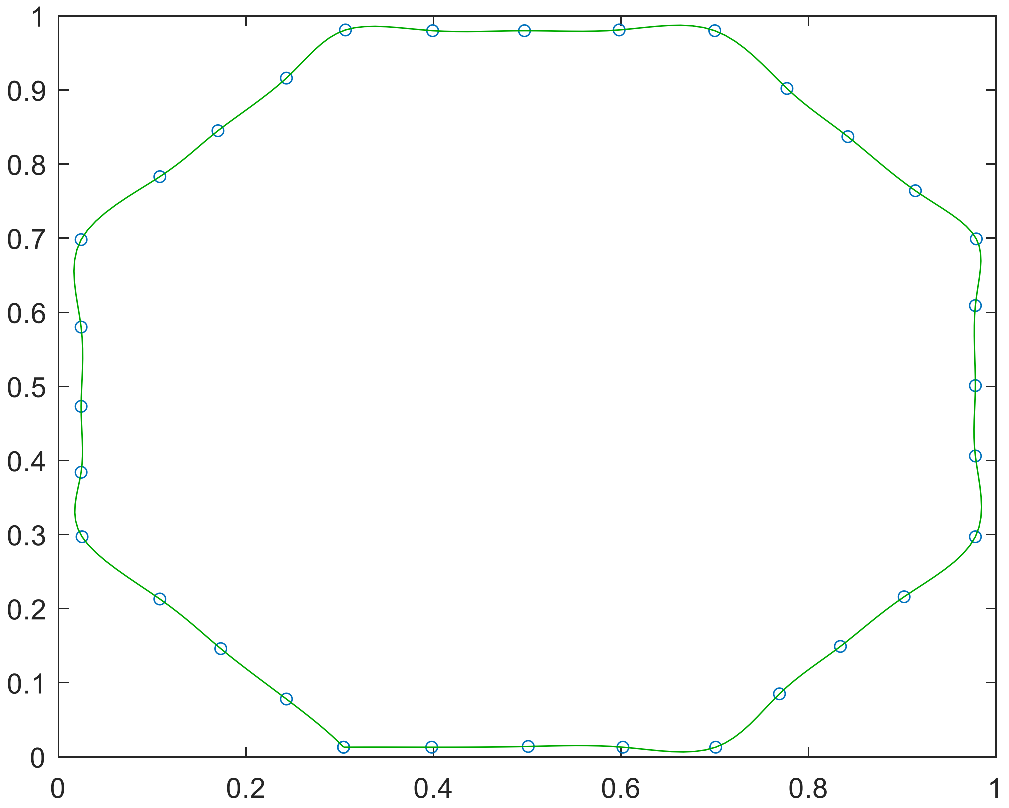
```
load myhand.dat
x = myhand(:,1);
y = myhand(:,2);
n = length(x);
```

% Plot

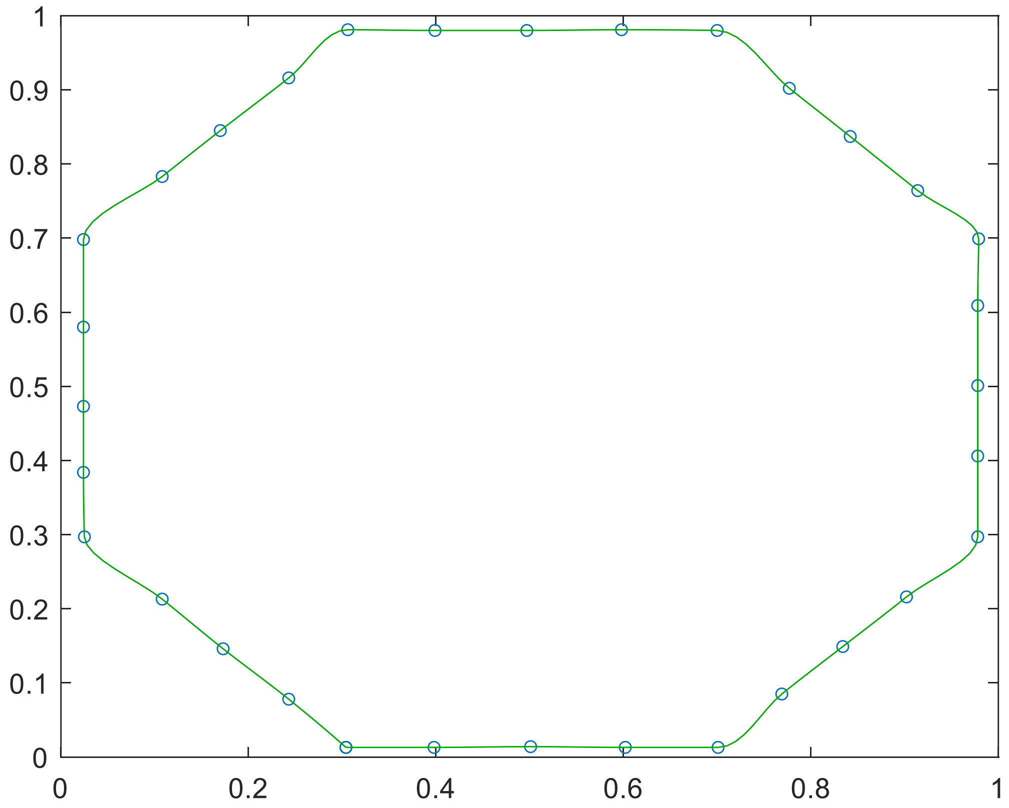
```
Z = pchiptx(theta,r,t).*exp(i*t);
p = plot(x,y,‘o’,real(Z),imag(Z),‘k-’);
set(p(1),‘markersize‘,4) %點標大小
set(p(2),'color',[0 2/3 0]) %RGB
axis([-.55 .55 -.05 1.05]) %兩軸範圍
```



Regular Octagon (Cartesian Coor.)

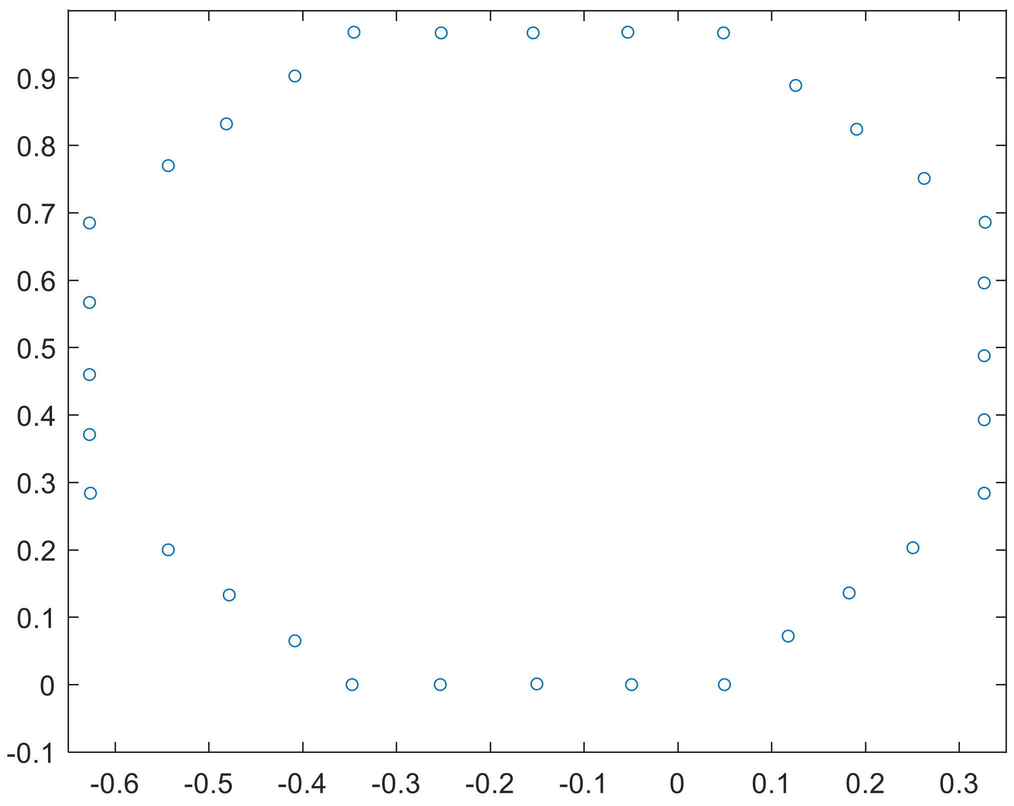


Spline Interp.



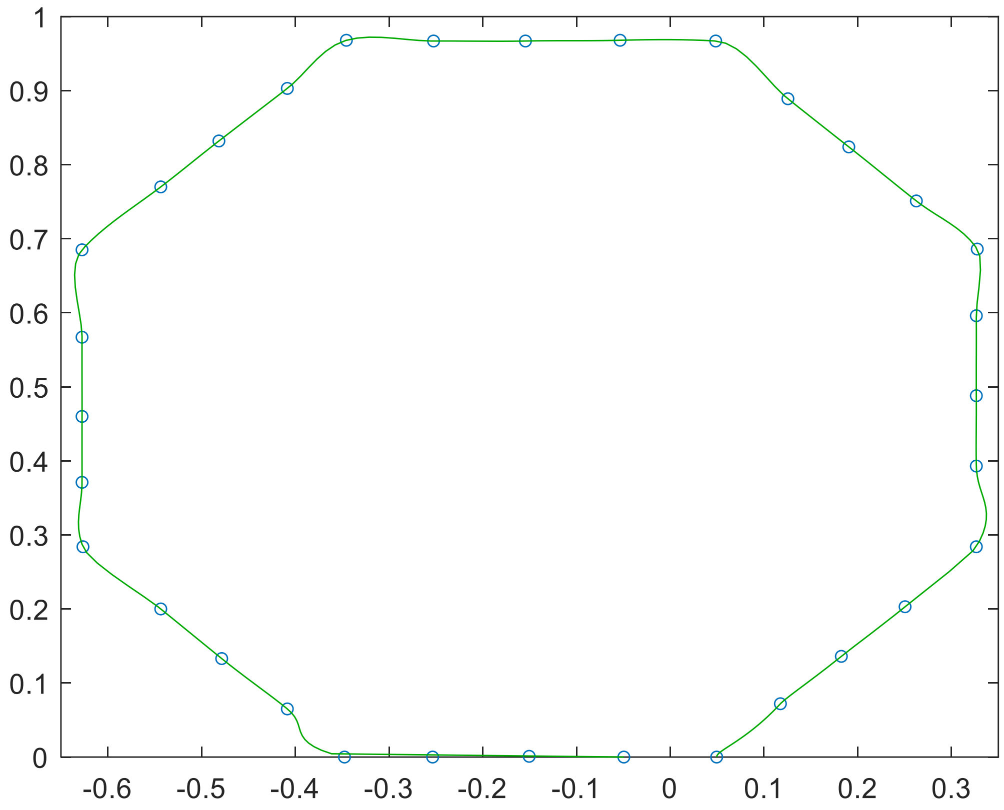
Pchip Interp.

Regular Octagon (Polar Coor.)



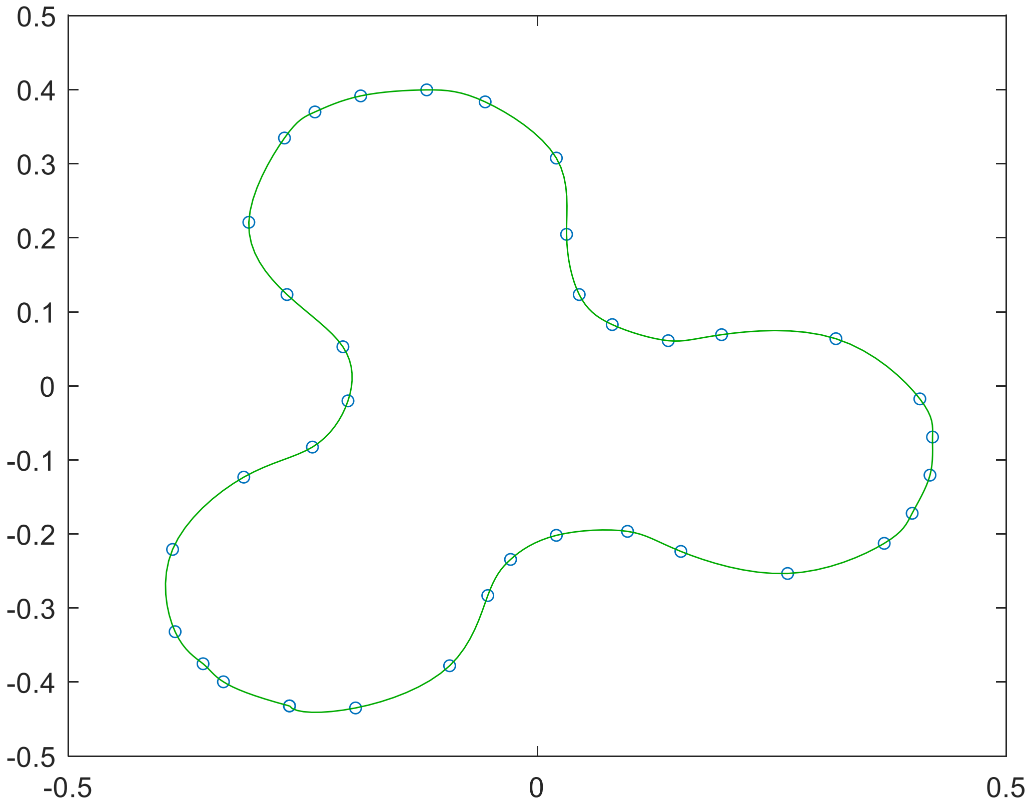
Spline Interp.

Matrix is singular to working precision!!!

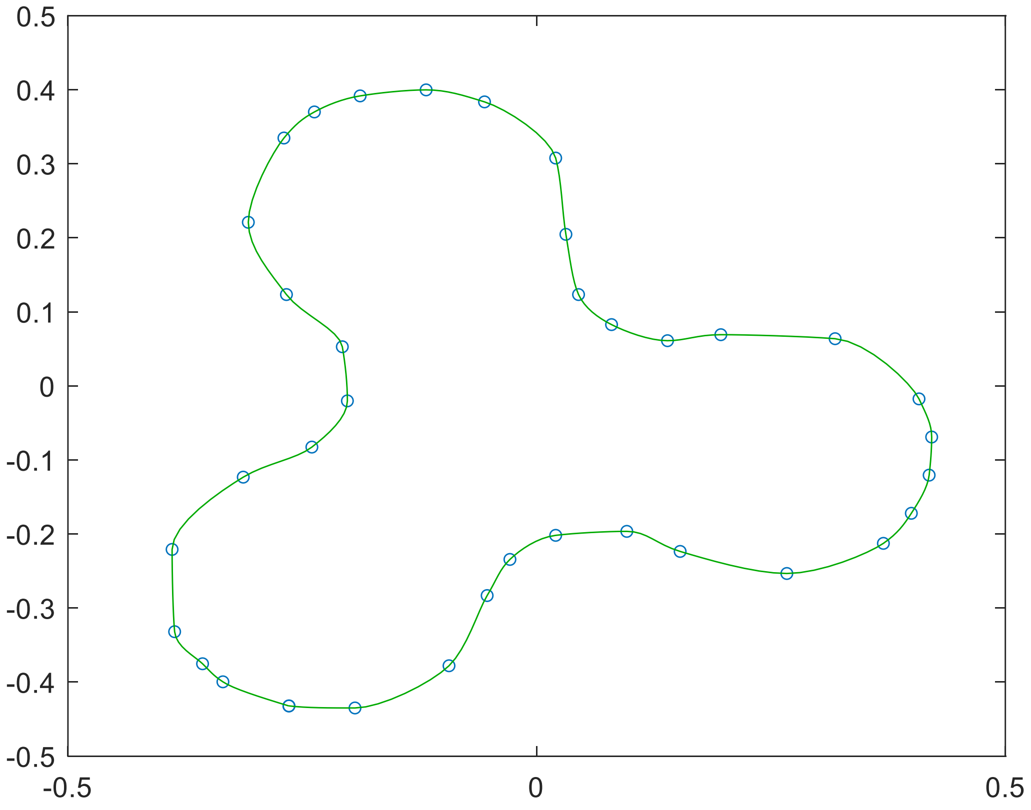


Pchip Interp.

Fidget Spinner (Cartesian Coor.)

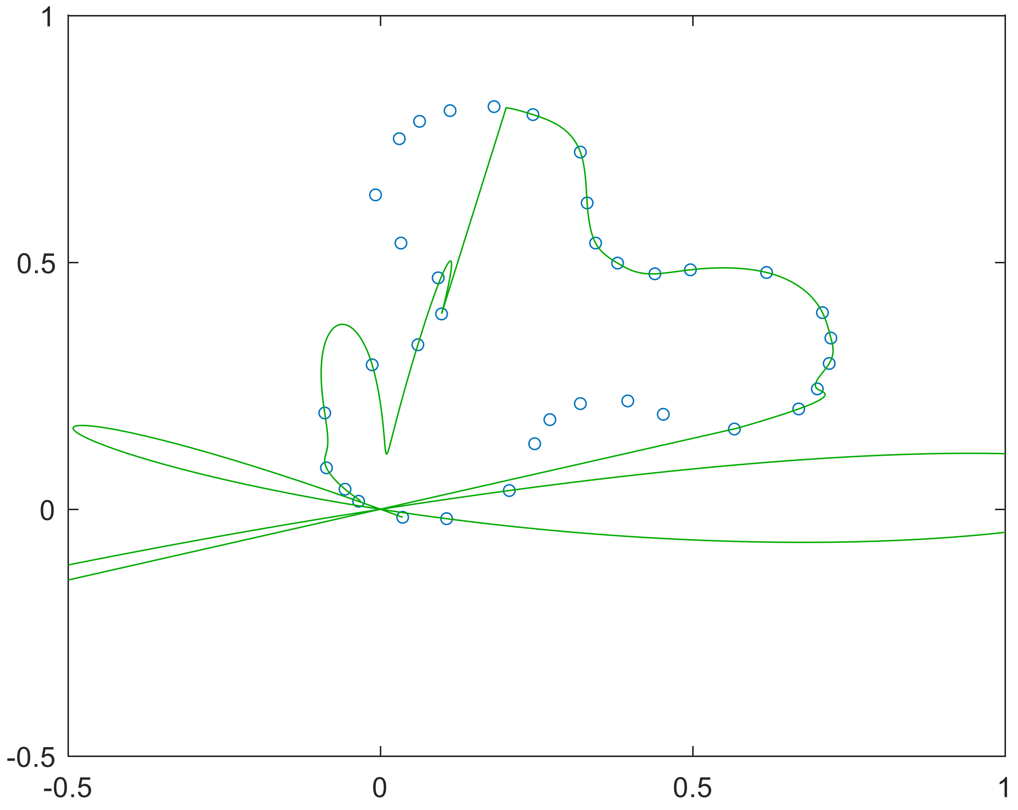


Spline Interp.

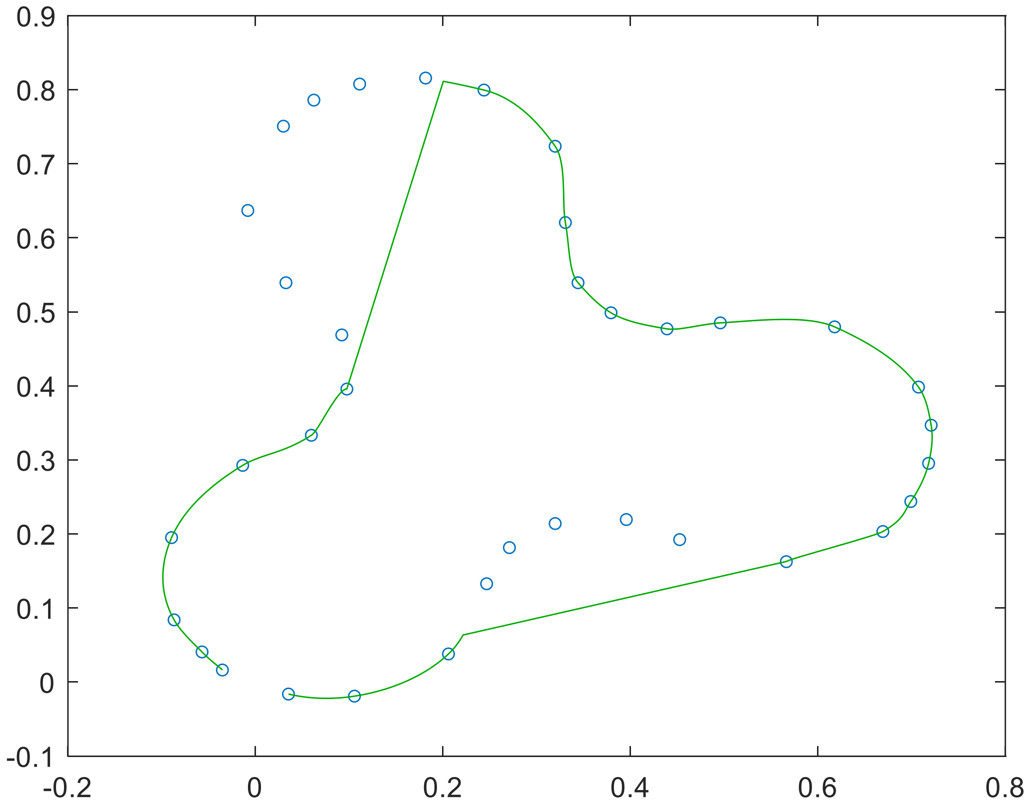


Pchip Interp.

Fidget Spinner (Polar Coor.)



Spline Interp.



Pchip Interp.