Data driven function approximation

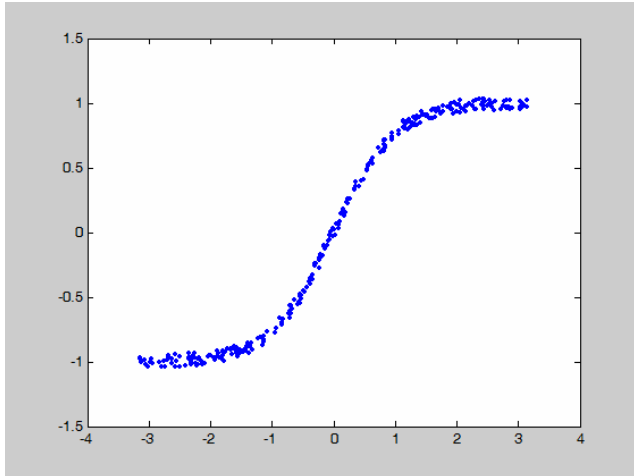
Forward kinematics

Inverse kinematics

Exercise

- Download ex1.dat

```
z=load('ex1.dat');
plot(z(:,1),z(:,2),'.');
```

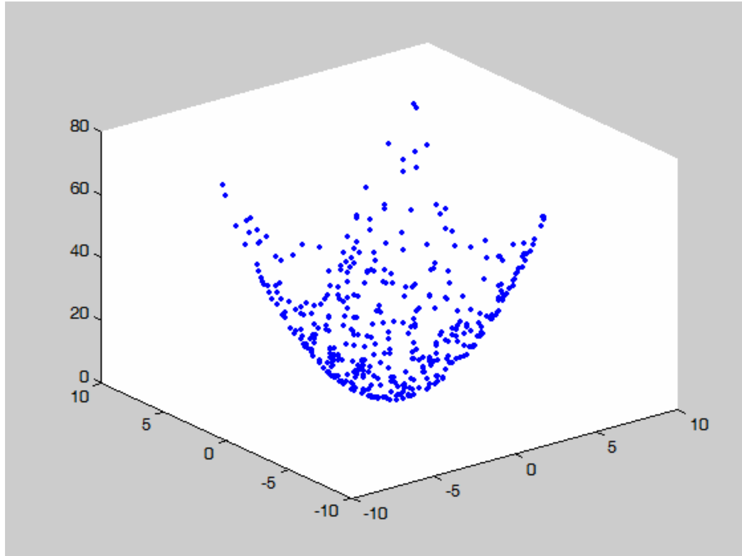


- Train an MLP network subject to paired data in ex1.dat
- Plot the approximating function

Exercise

- Download ex2.dat

```
z=load('ex2.dat');
>> plot3(z(:,1),z(:,2),z(:,3),'.');
```



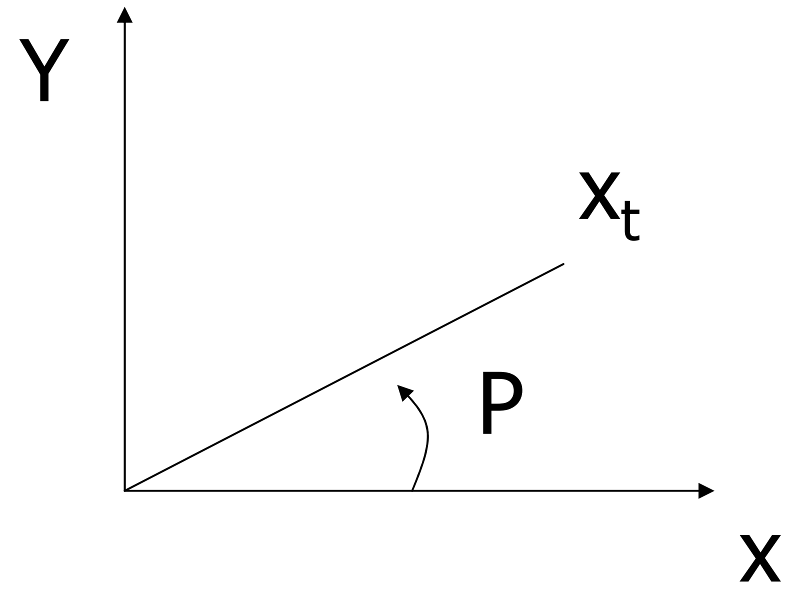
- Train an MLP network subject to paired data in ex2.dat
- Plot the approximating function

Forward kinematics of one link robot

- Given X_t what is the joint angle P

$$x_t = l\cos(P)$$

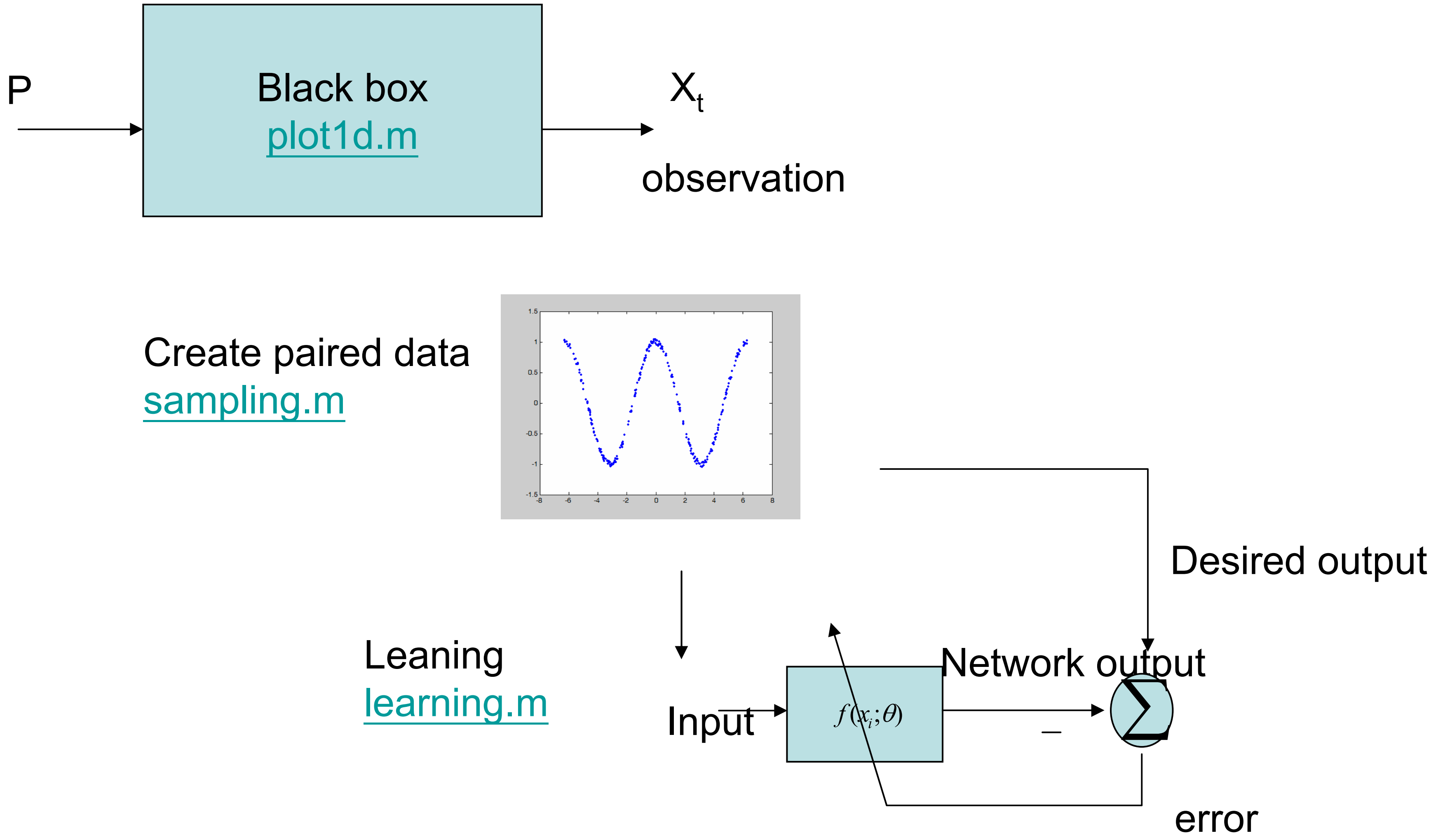
forward position solution



Matlab code

1	eval_MLP.m	Evaluate an MLP function
2	mean_square_error2.m	Calculate E
3	The NNSYSID Toolbox	
4	learn_MLP.m	Seek parameters

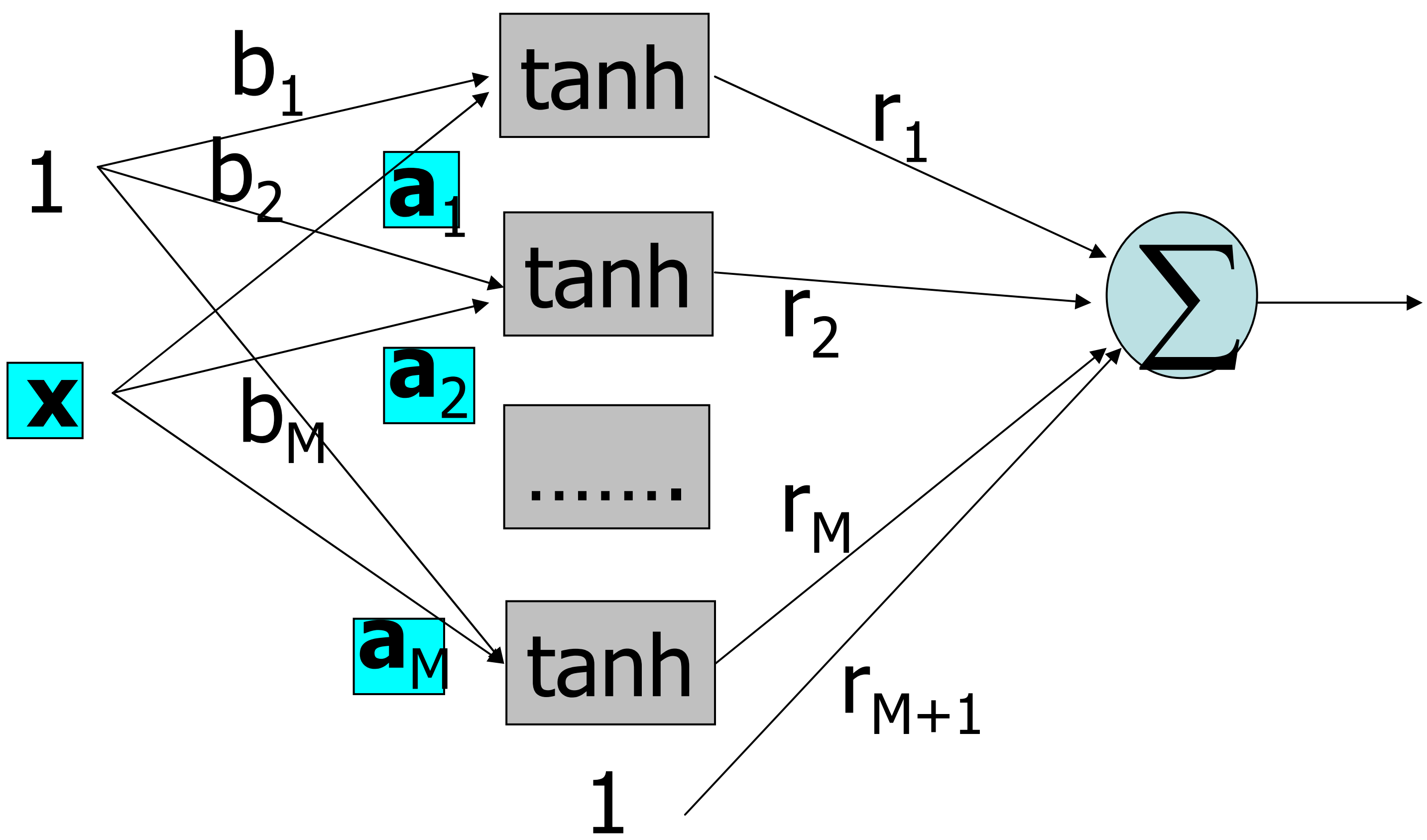
Application



Matlab code

Function approximation for an arbitrary target function
fa1d.m

Network



Network function

$$f(x_i;\theta) = \sum_{m=1}^{M} r_m \tanh(\mathbf{a}_m^T \mathbf{x} + b_m) + r_0$$

$$\theta = \{\mathbf{a}_m\} \cup \{b_m\} \cup \{r_m\}$$

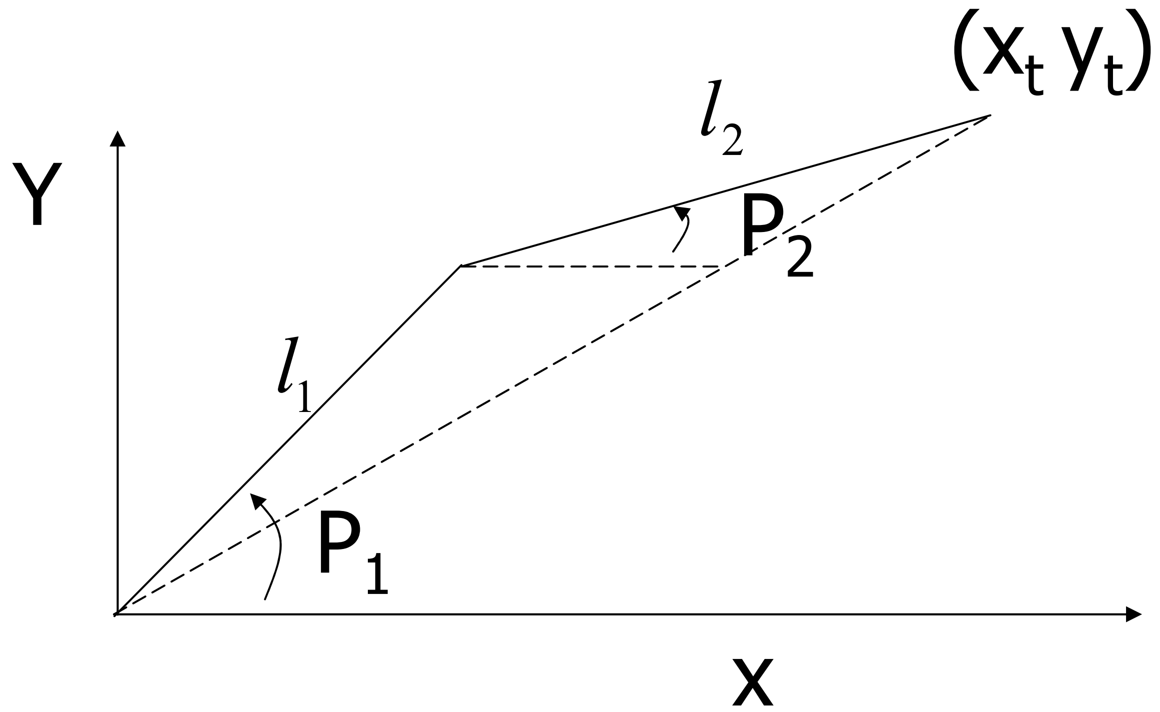
MLP learning

```
[a,b,r]=learn_MLP(x',y',M);
```

Matlab code

1	eval_MLP2.m	Evaluate MLP function
2	mean_square_error2.m	Calculate E
3	The NNSYSID Toolbox	
4	learn_MLP.m	Seek parameters

Two links



Forward kinematics of two-link robot

$$P_1, P_2$$

$$\downarrow$$

$$x_t = l_1 \cos(P_1) + l_2 \cos(P_2)$$

$$y_t = l_1 \sin(P_1) + l_2 \sin(P_2)$$

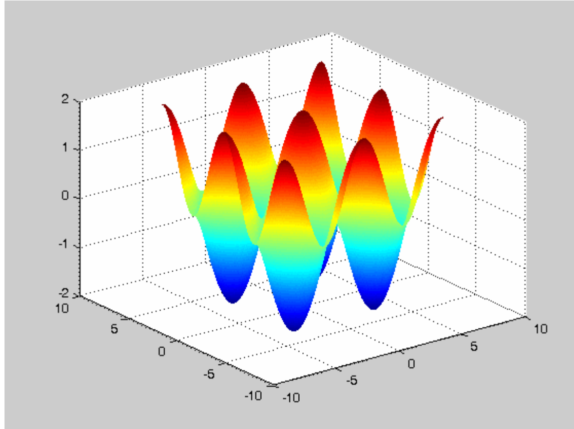
$$\downarrow$$

$$x_t, y_t$$

Data preparation

P_1, P_2 → **Black box** plot2d.m → x_t

observations

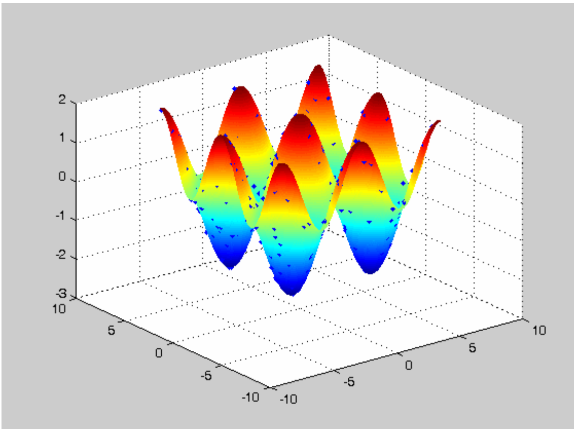


$$x_t = l_1 \cos(P_1) + l_2 \cos(P_2)$$

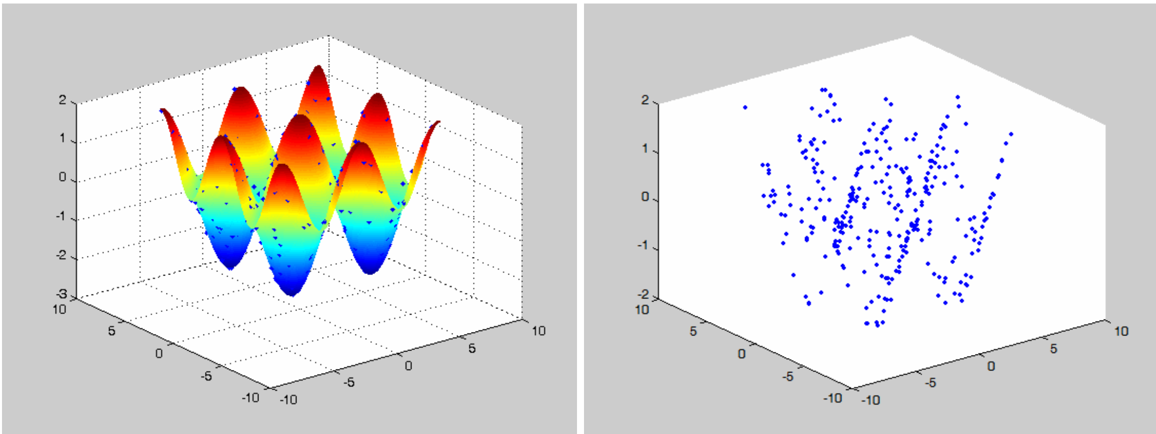
⇩

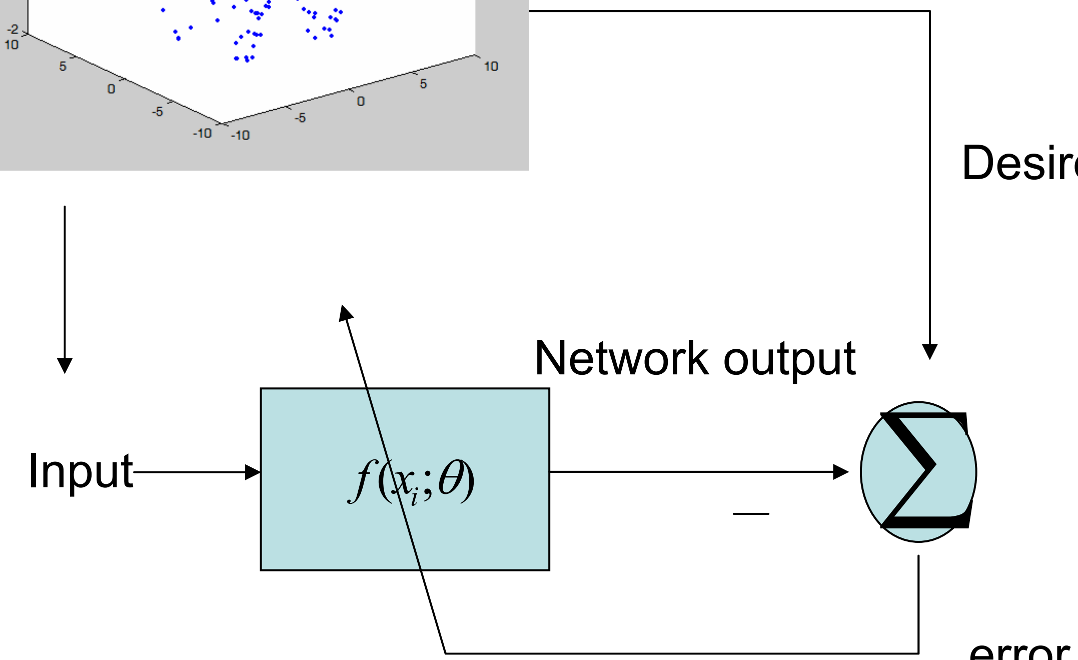
Create paired data
sampling2.m

⇩



Learning forward kinematics



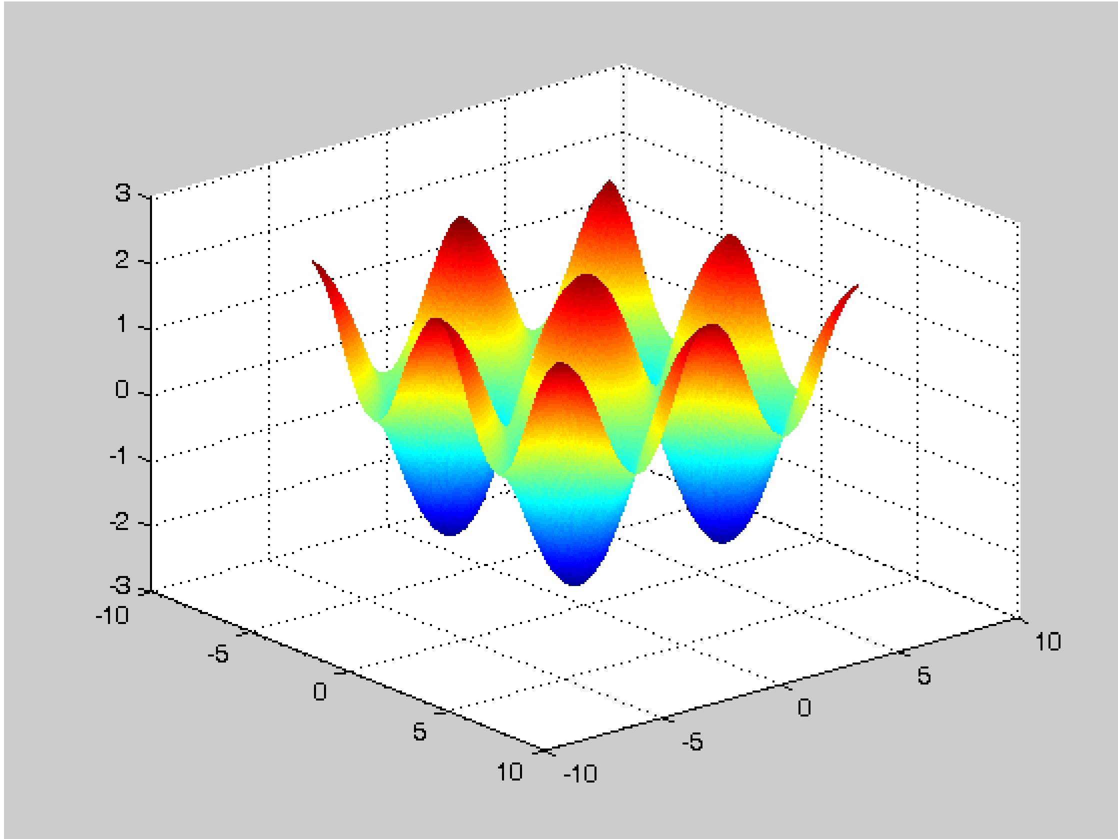


M=input('keyin the number of hidden units:');
[a,b,r]=learn_MLP(x',y',M);

fa2d.m

Reconstructed forward Kinematics

$$x_t = l_1 \cos(P_1) + l_2 \cos(P_2)$$

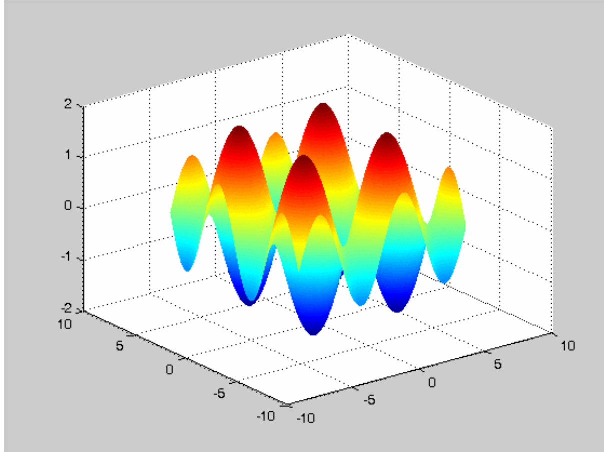


MSE for training data 0.001075
ME for training data 0.026292

Data preparation

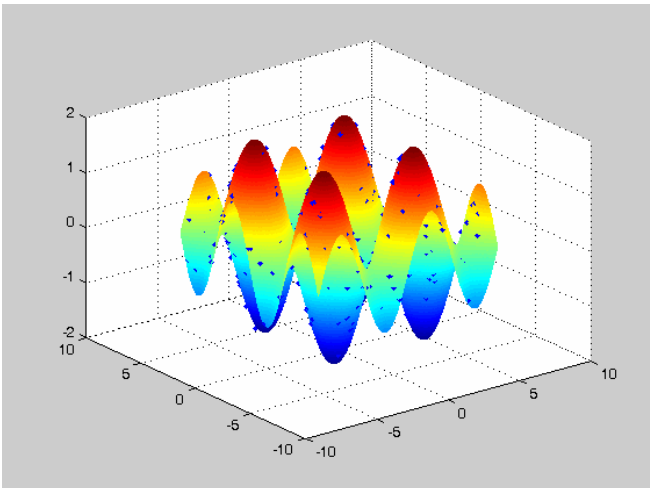
P_1, P_2 → **Black box** plot2d.m → y_t

observations

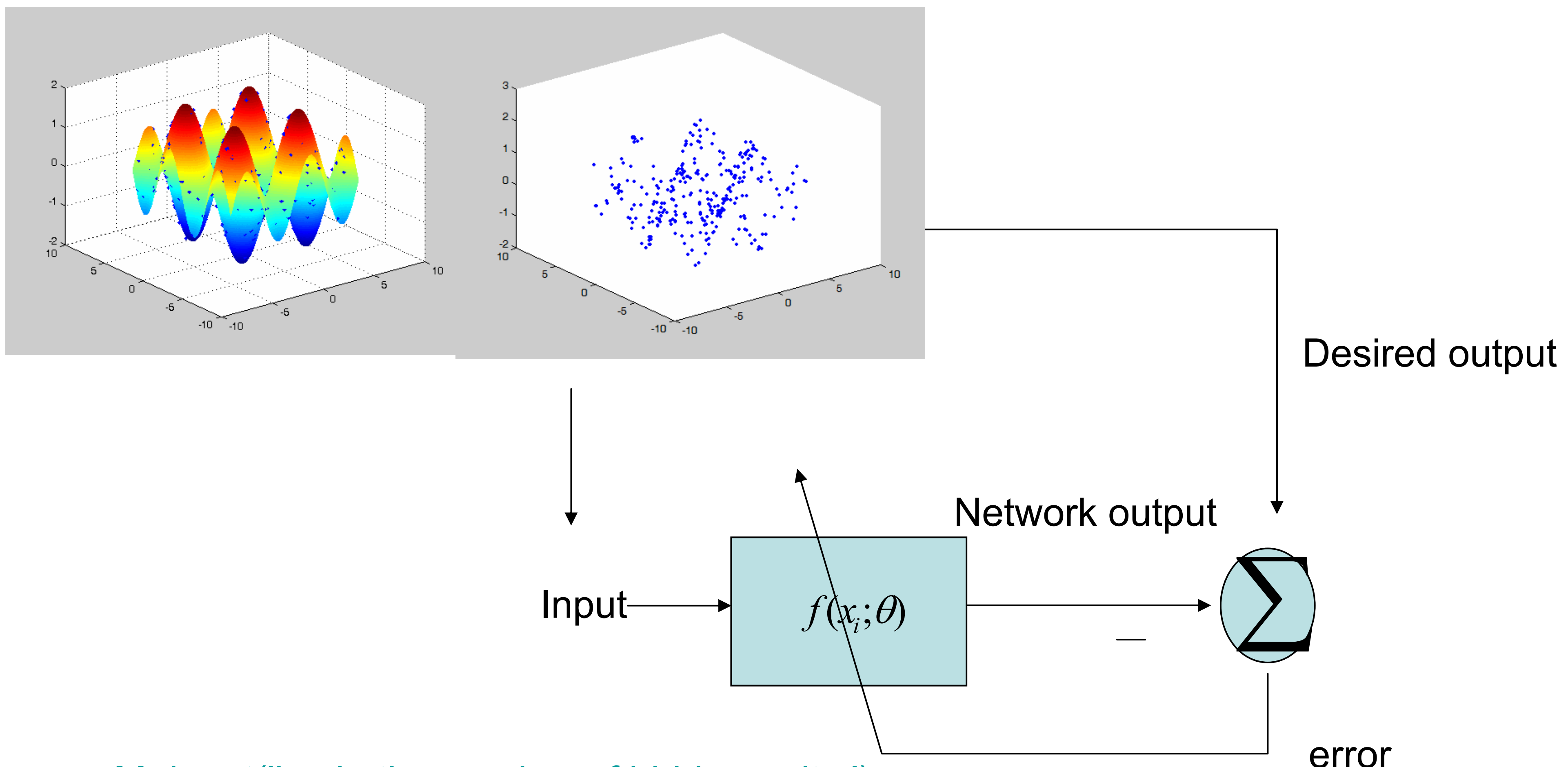


$$y_t = l_1 \sin(P_1) + l_2 \sin(P_2)$$

Create paired data
sampling2.m



Learning forward kinematics

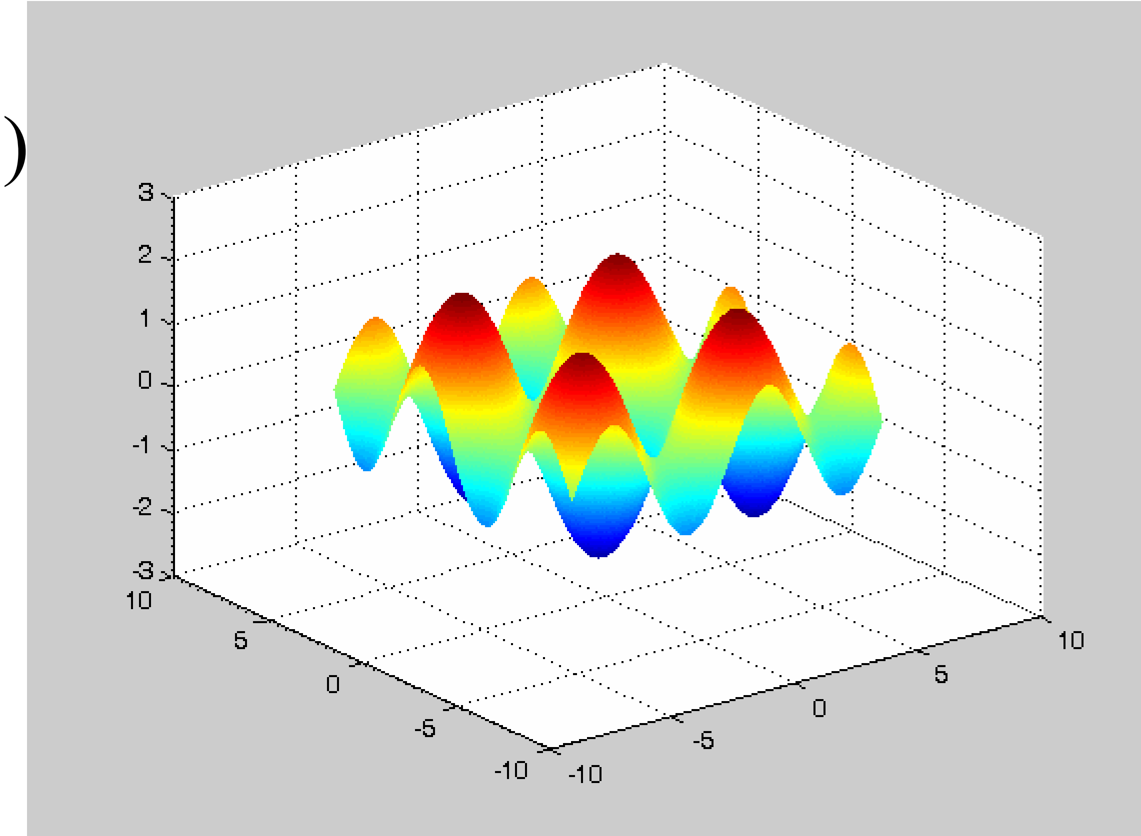


M=input('keyin the number of hidden units:');
[a,b,r]=learn_MLP(x',y',M);

fa2d.m

Reconstructed forward Kinematics

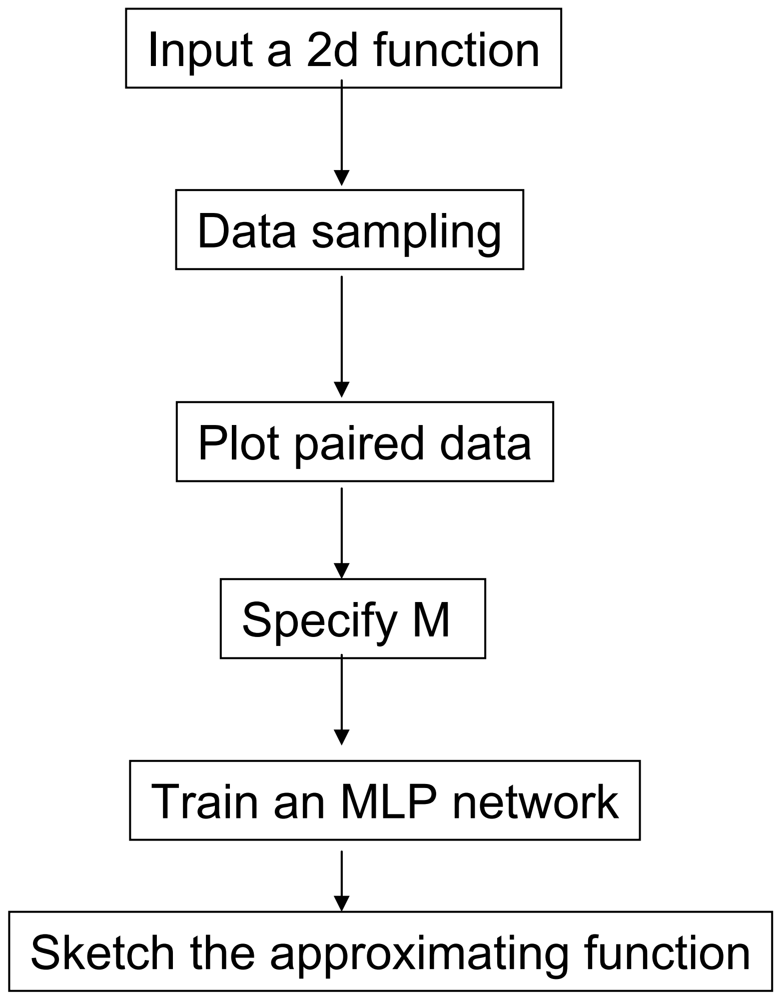
$$y_t = l_1 \sin(P_1) + l_2 \sin(P_2)$$



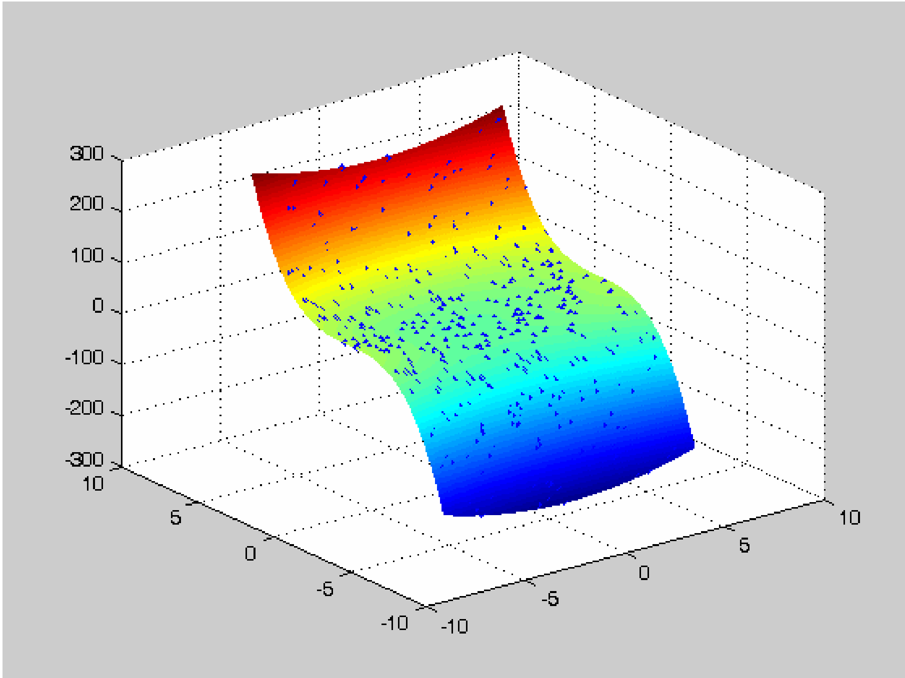
MSE for training data 0.001232
ME for training data 0.027646

Data driven function approximation

demo_fa2d.m



Example

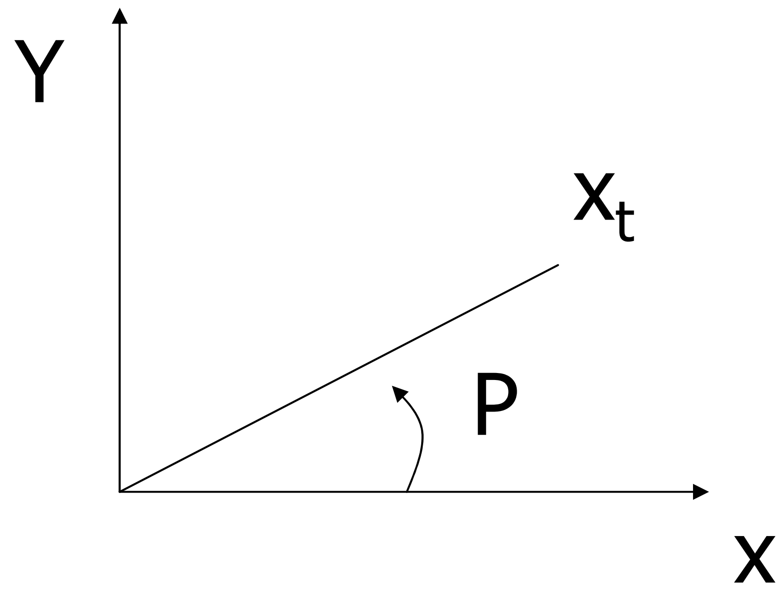


x1.^2+x2.^3

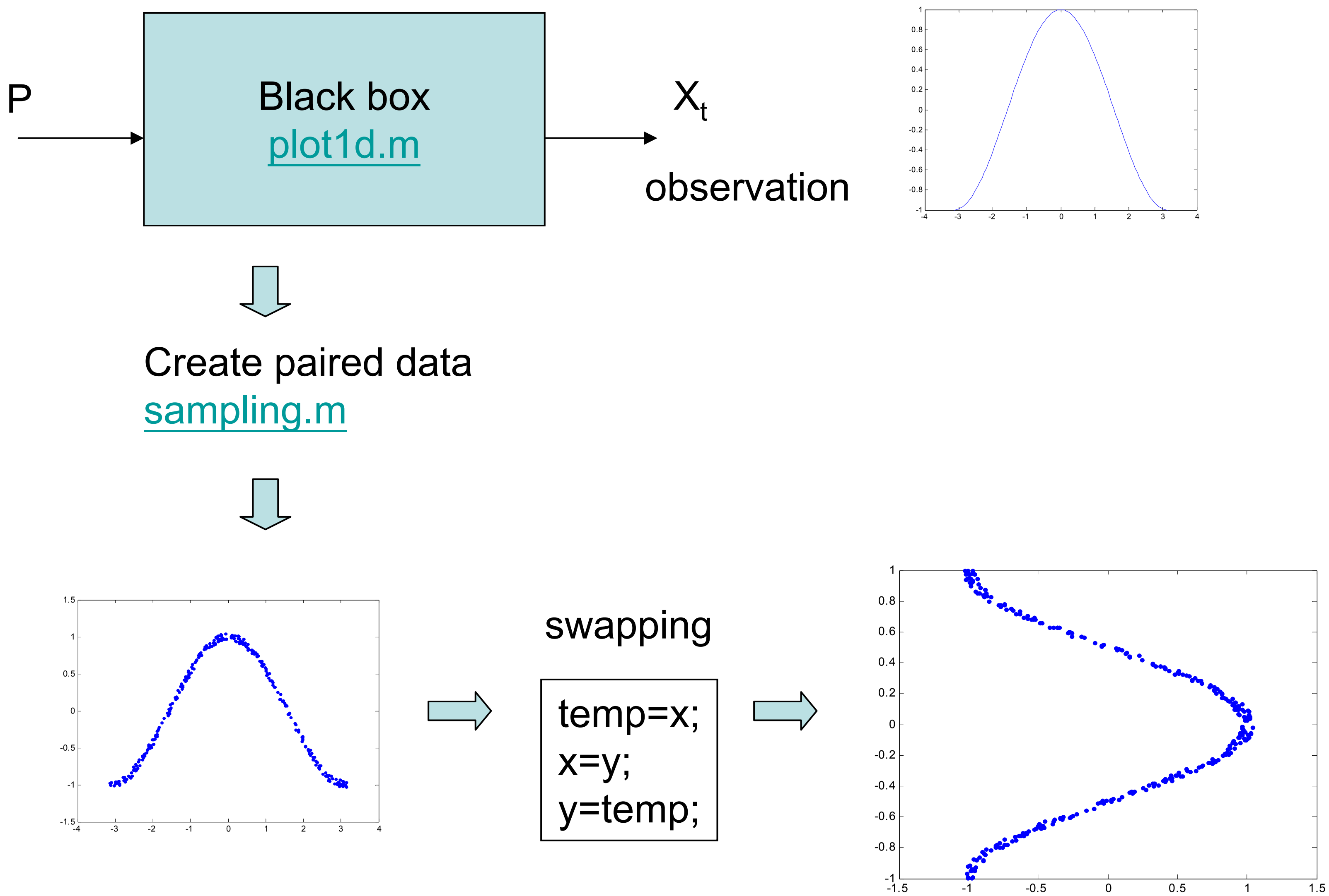
Inverse kinematics of one-link robot

$$\cos(P) = \frac{x_t}{l}$$

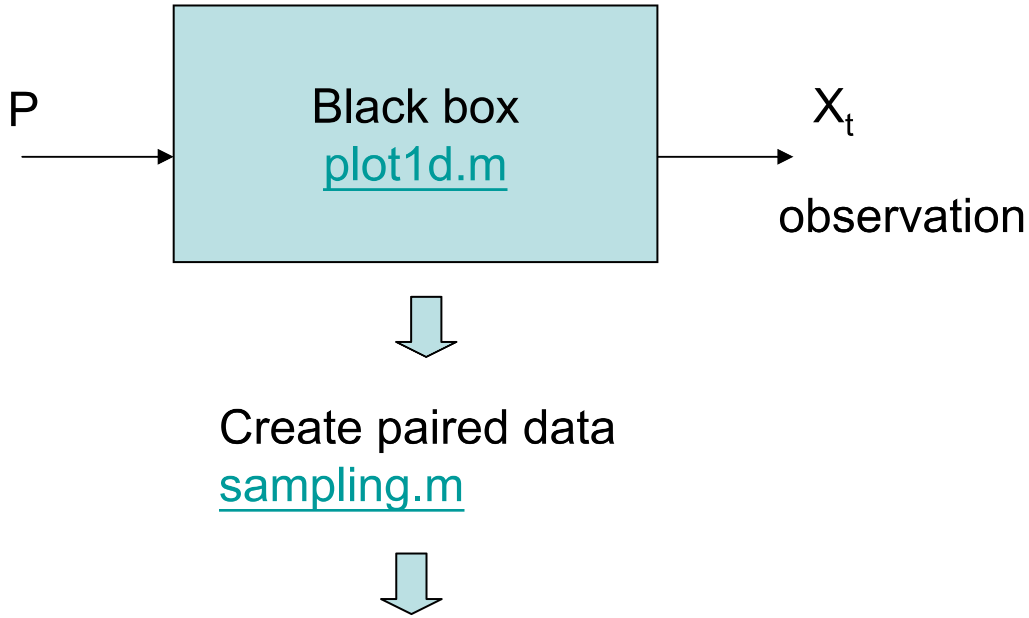
$$P = \arccos\frac{x_t}{l}$$

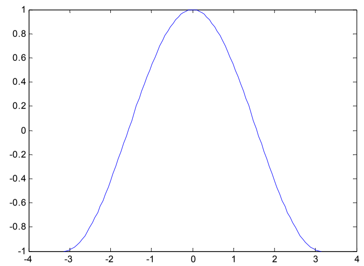


Data preparation

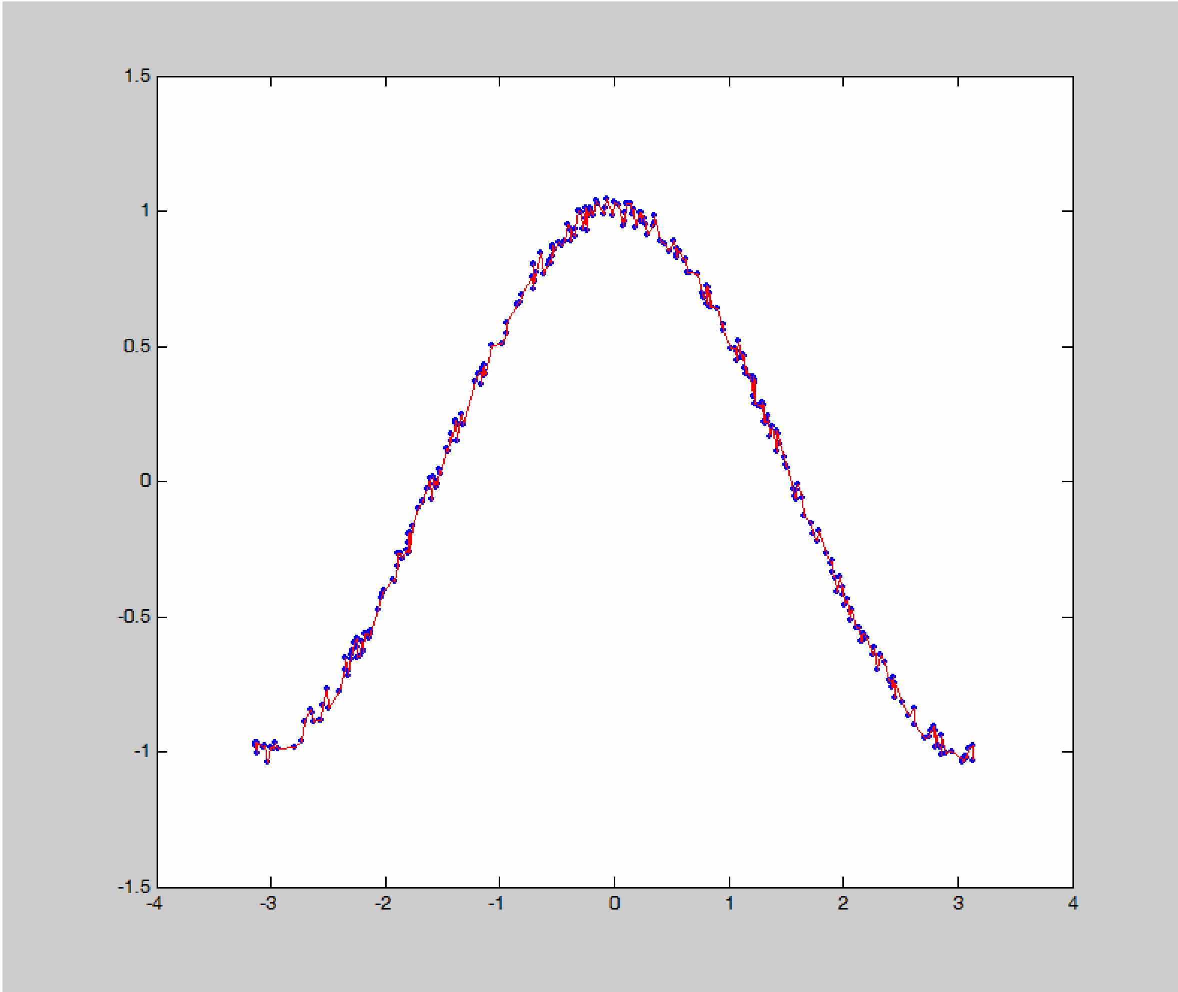


Sorting



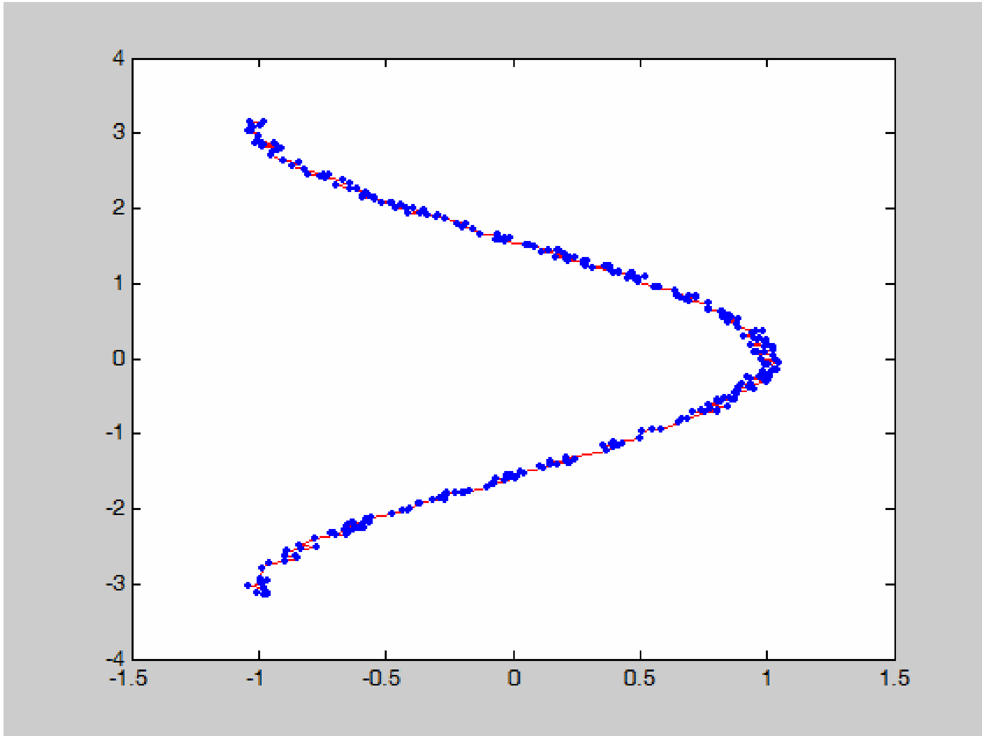


```
[x,ind]=sort(x);
y=y(ind);
z=[x;y];
```

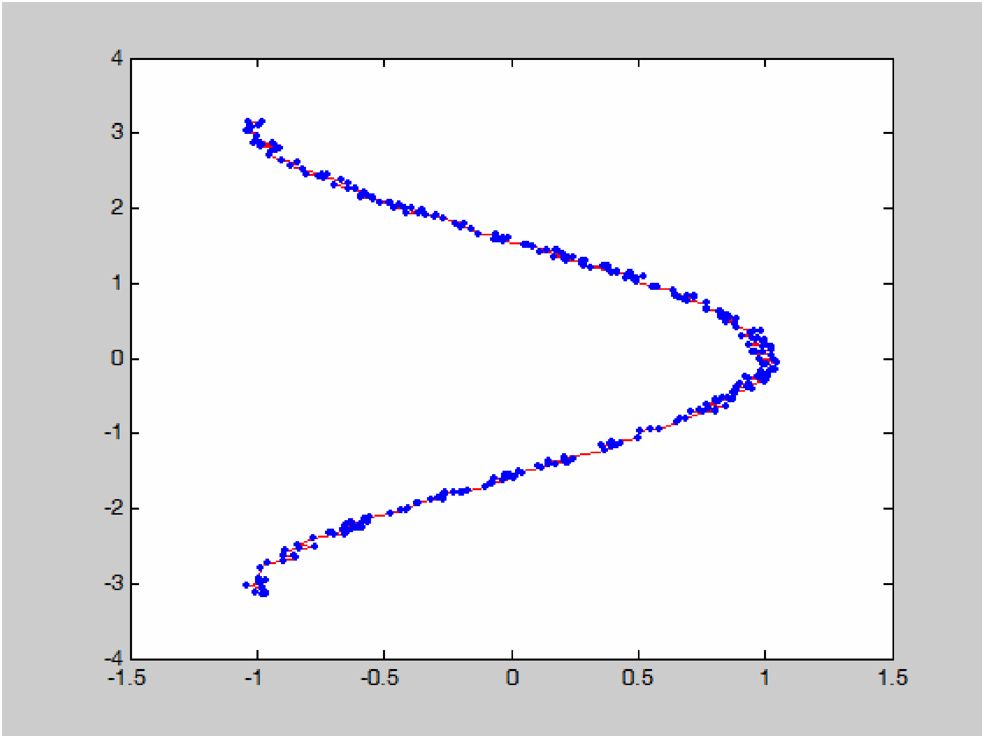

1.5
1
0.5
0
-0.5
-1
-1.5
-4
-3
-2
-1
0
1
2
3
4

swapping

```
temp=x;
x=y;
y=temp;
```



Polar systems



P

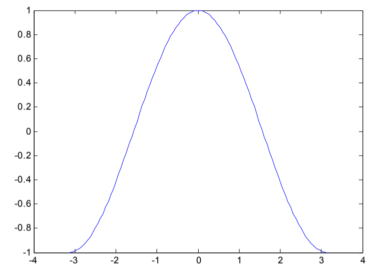
X

$$P = f(t;\theta_1)$$

$$X = f(t;\theta_2)$$

Construction of a polar system

P → **Black box** plot1d.m → X_t observation



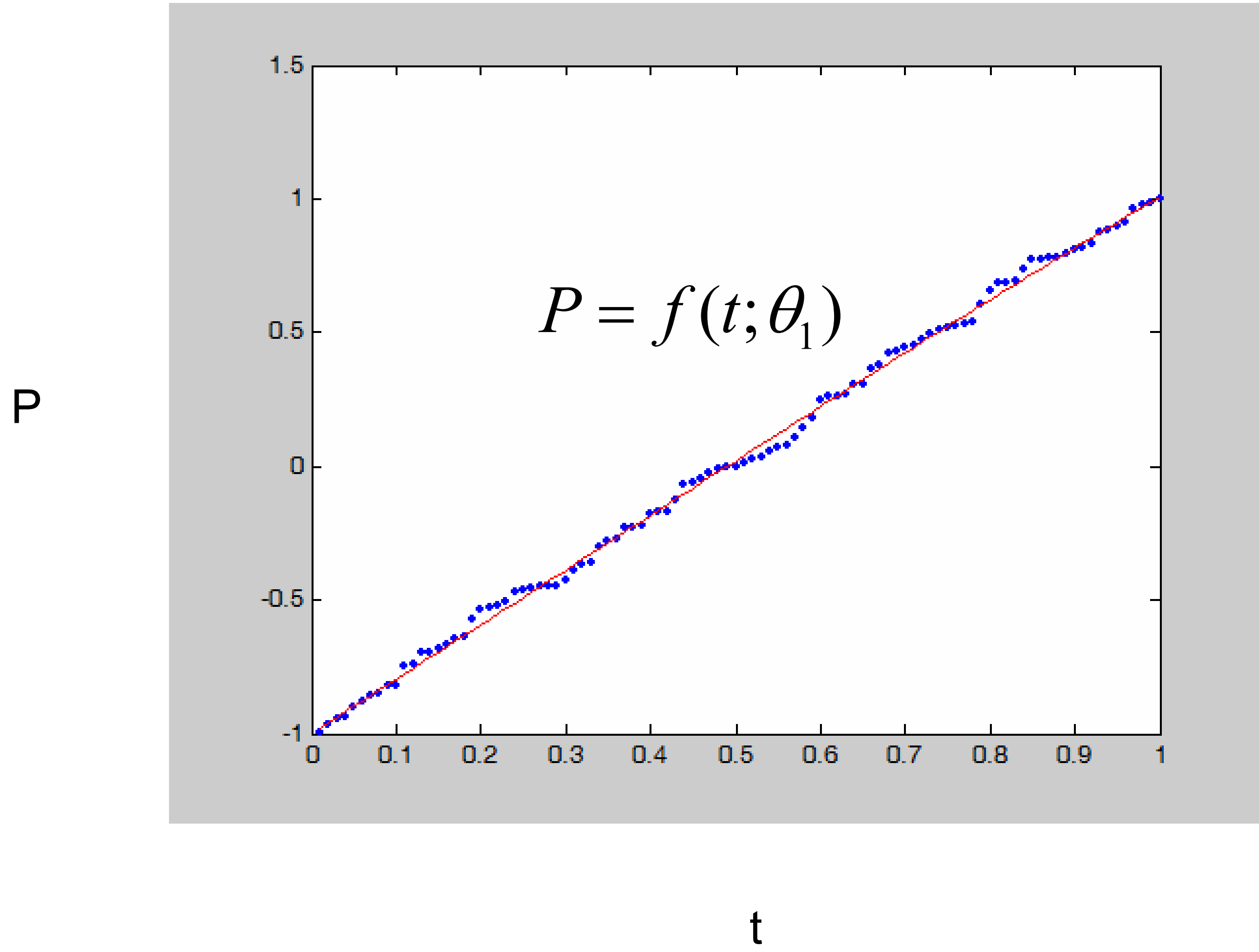
⇩

Create paired data
sampling.m

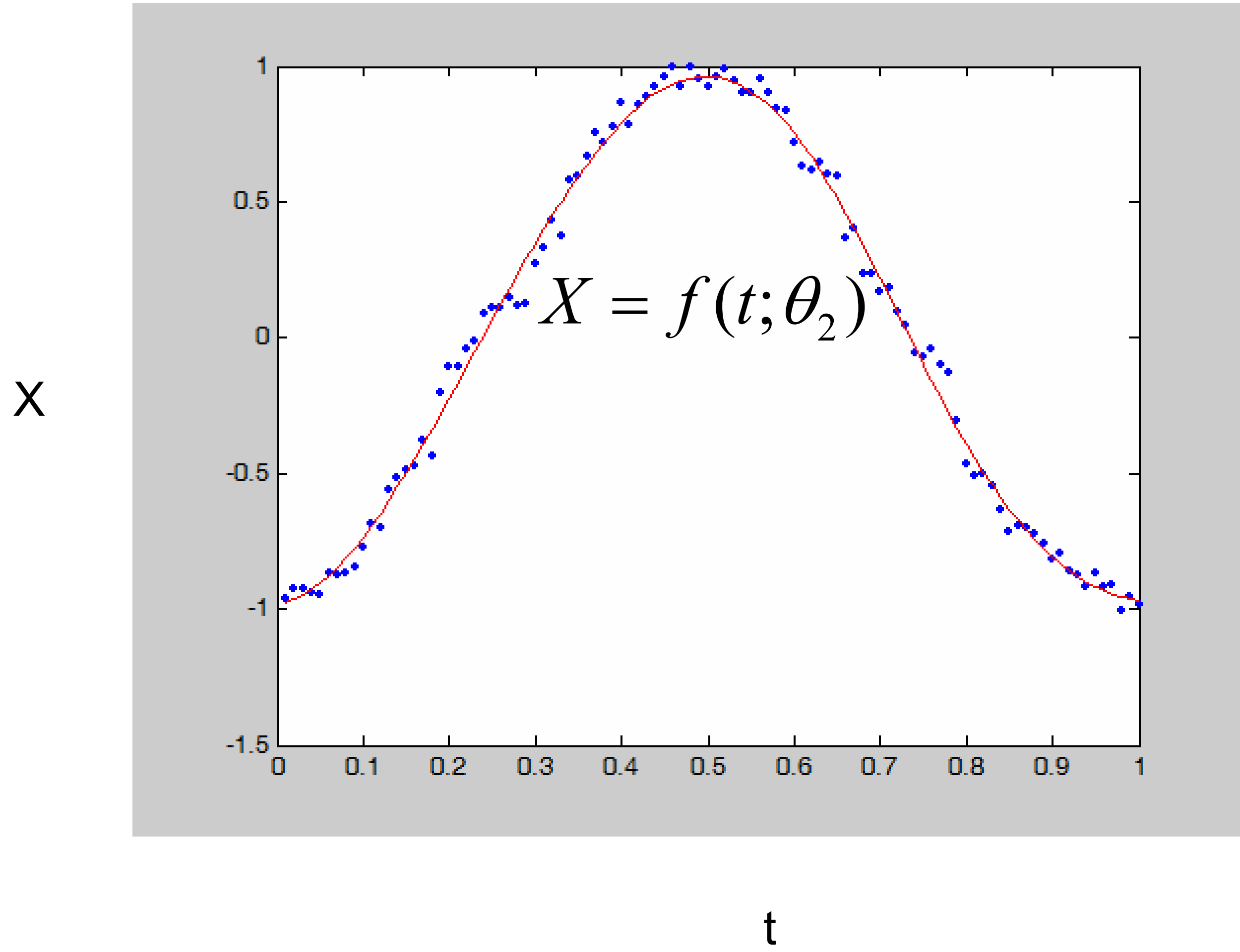
⇩

```
[x,ind]=sort(x);
y=y(ind);
z=[x;y];
```

⇨ fa1d_inv1.m

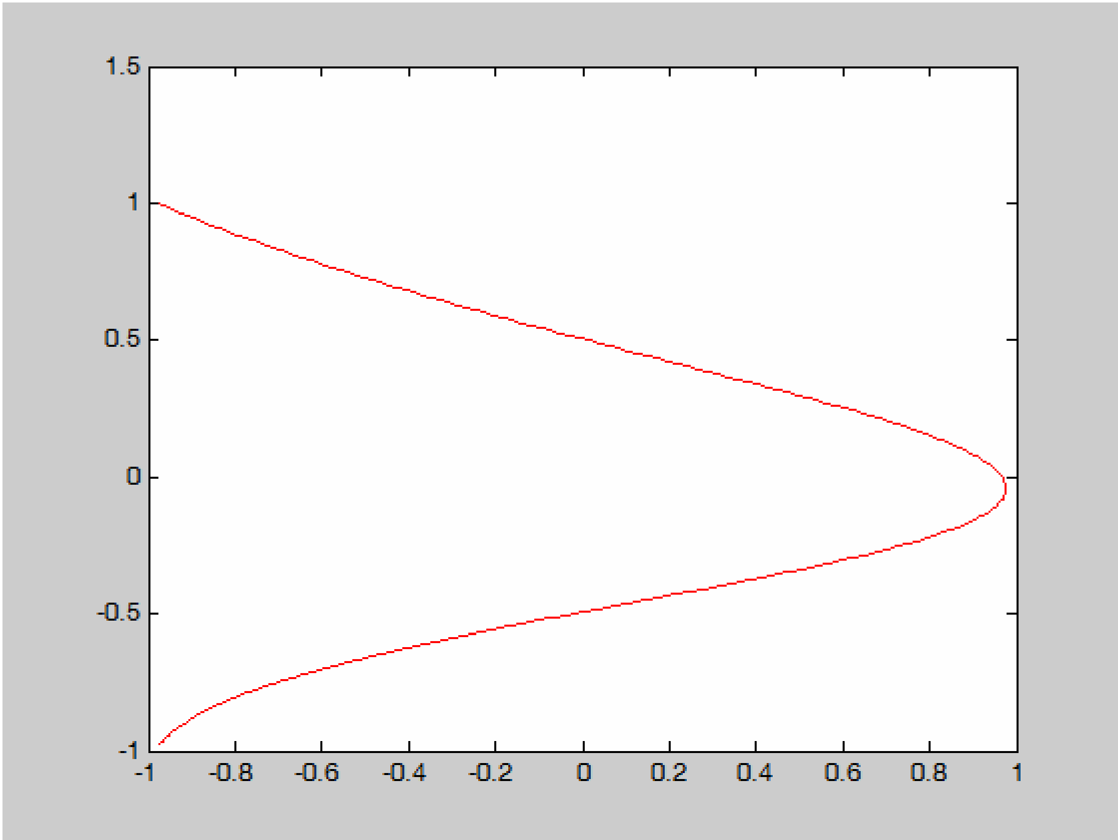


P
t
$P = f(t;\theta_1)$
1.5
1
0.5
0
-0.5
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



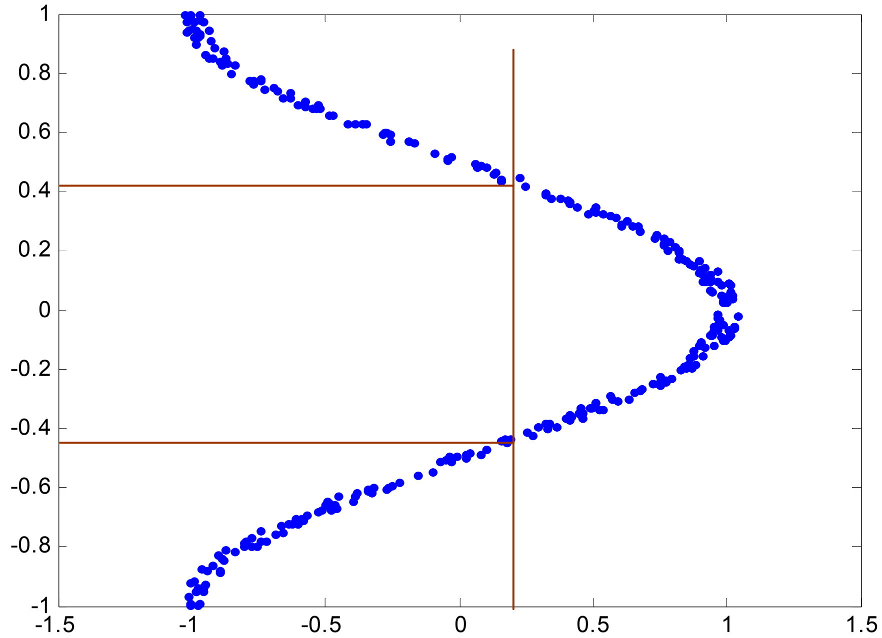
X
t
$X = f(t;\theta_2)$
1
0.5
0
-0.5
-1
-1.5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Reconstructed inverse kinematics



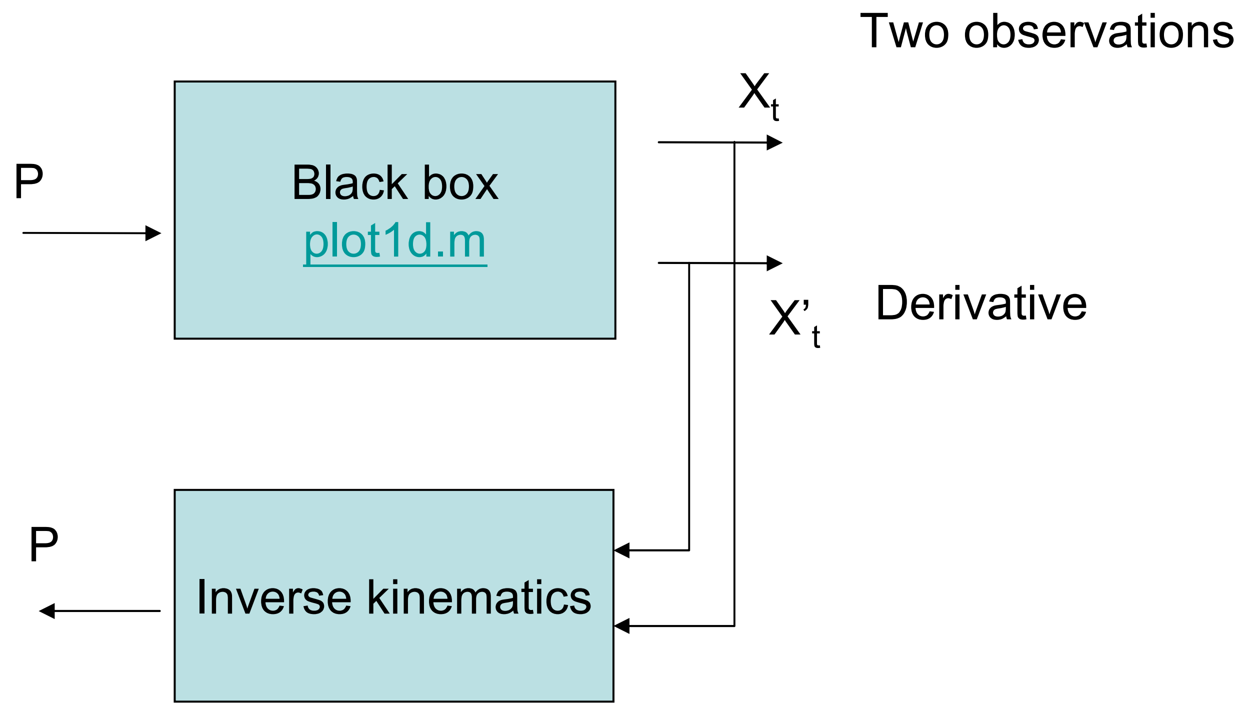
Multiple outputs

- Inverse kinematics
- Conflicts of I/O relation
- Two distinct outcomes for the same input



Invertible kinematics

- Recruit first order derivative



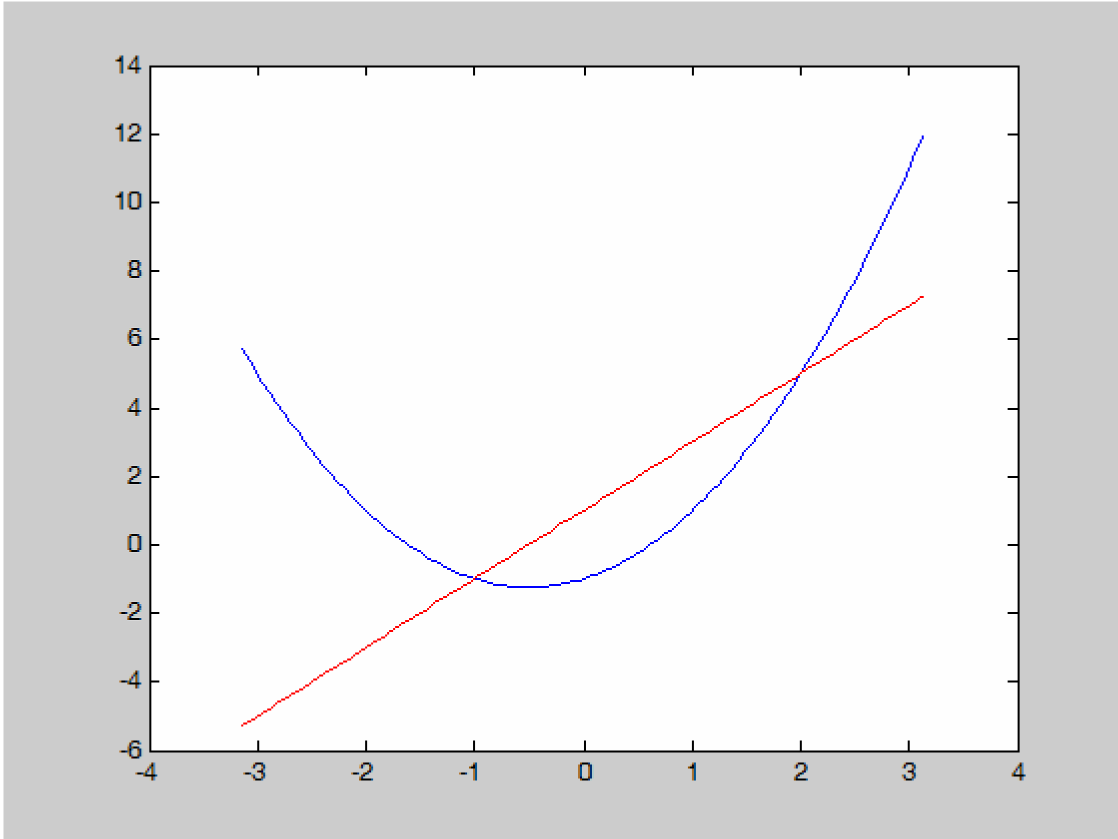
Symbolic differentiation

demo_diff.m

```
function demo_diff()
% input a string to specify a function
% plot its derivative
ss=input('function of x:','s');
fx=inline(ss);
x=sym('x');
ss=['diff(' ss ')'];
ss1=eval([sprintf(ss)]);
fx1=inline(ss1)
x=linspace(-pi,pi);
plot(x,fx(x),'b');hold on;
plot(x,fx1(x),'r');
return
```

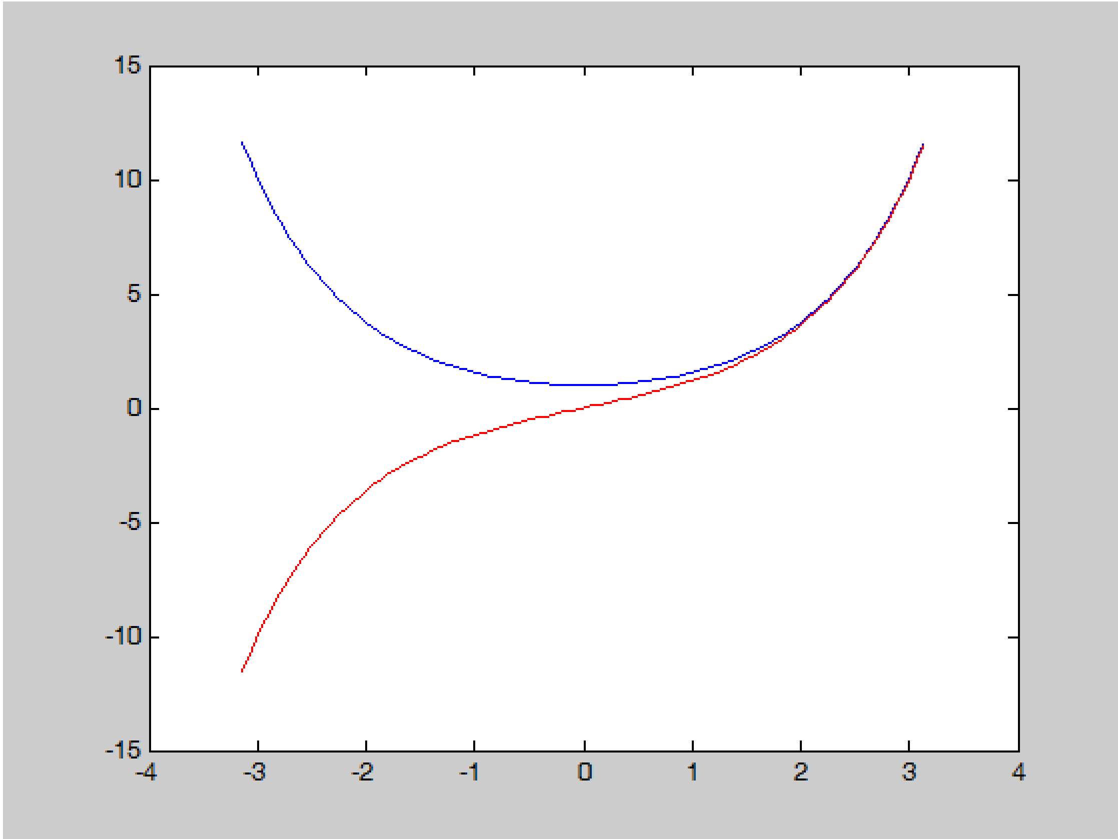
Example

```
>> demo_diff
function of x:x.^2+x-1

fx1 =

     Inline function:
     fx1(x) = 2.*x+1
```



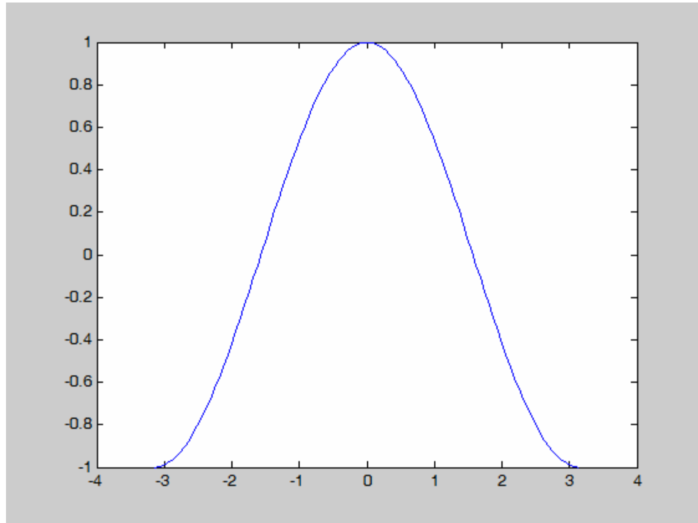
Example

```
>> demo_diff
function of x:cosh(x)

fx1 =

     Inline function:
     fx1(x) = sinh(x)
```

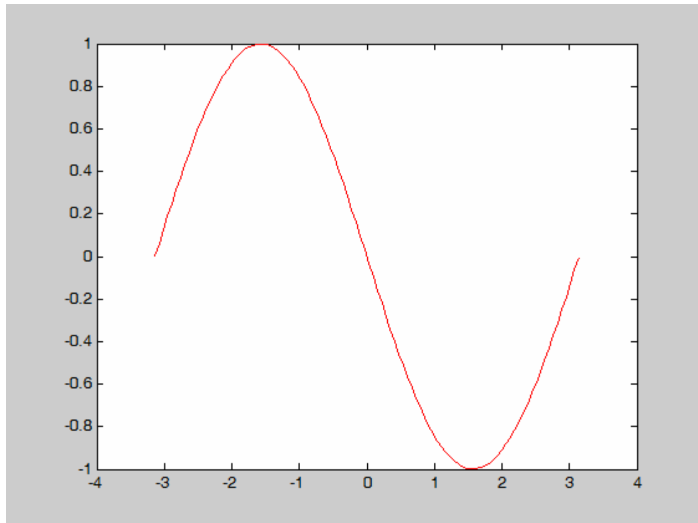


Data preparation

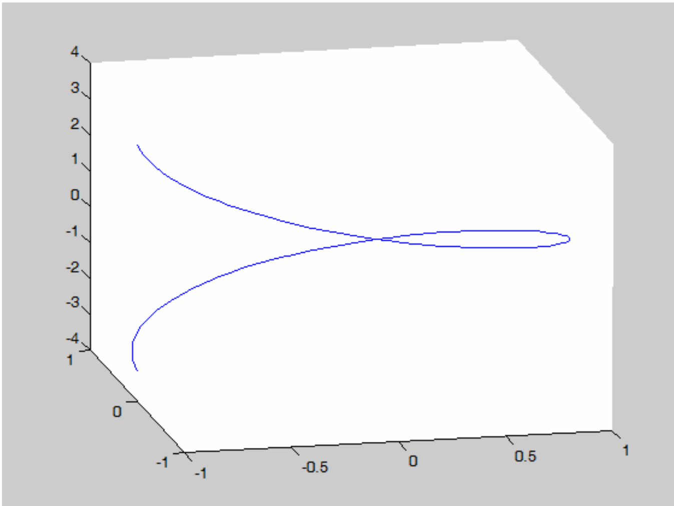
Two observations



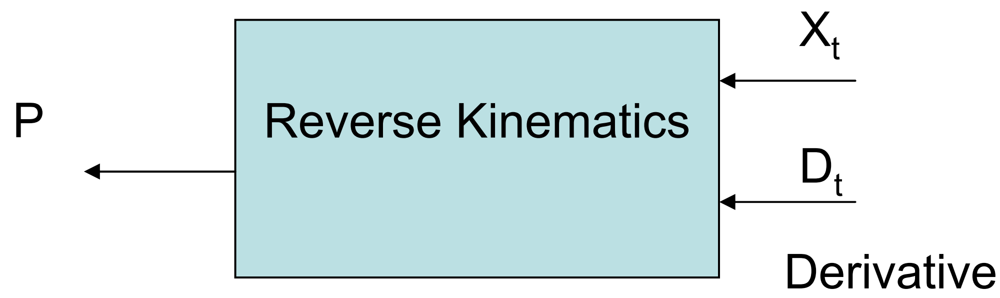
Derivative



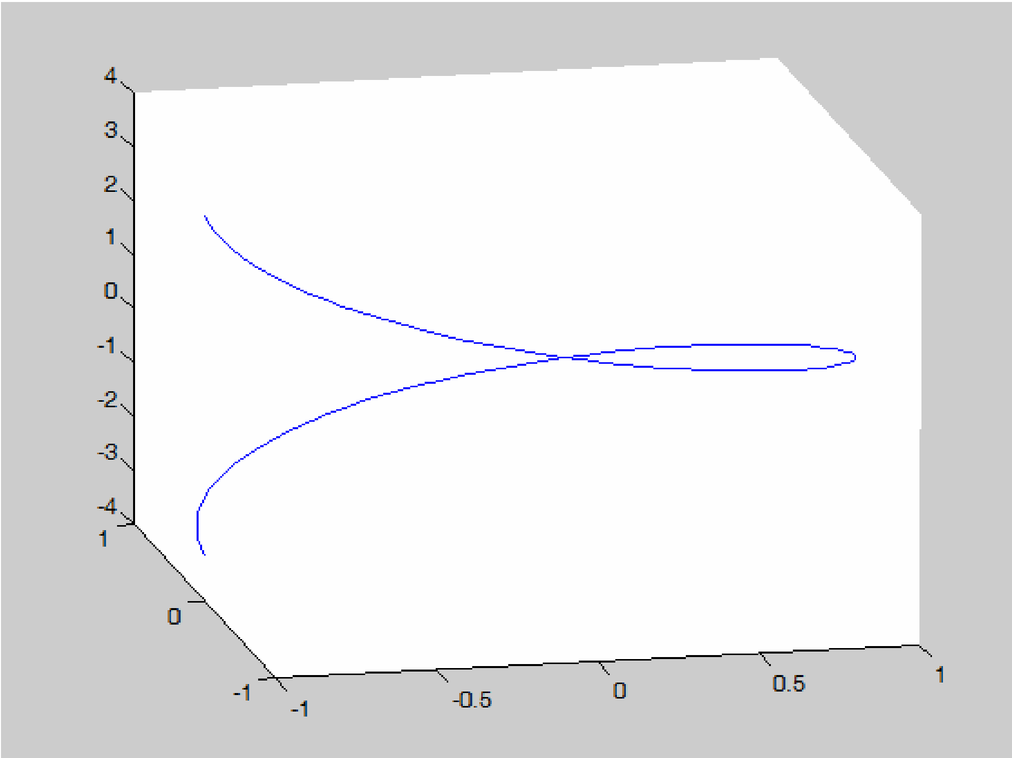
Reverse kinematics



plot3(y1,y2,ix)



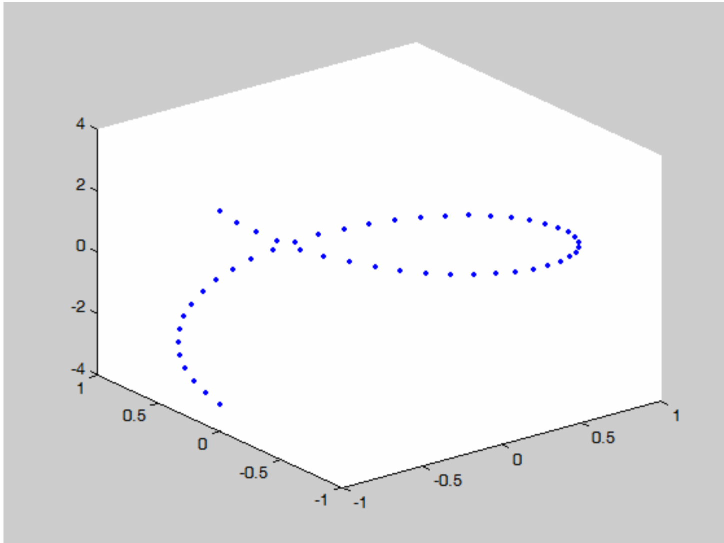
Dimensionality



- 3d curve
- Representation of a 3d curve in a polar system.
- Approximation of a 3d curve by learning MLP networks

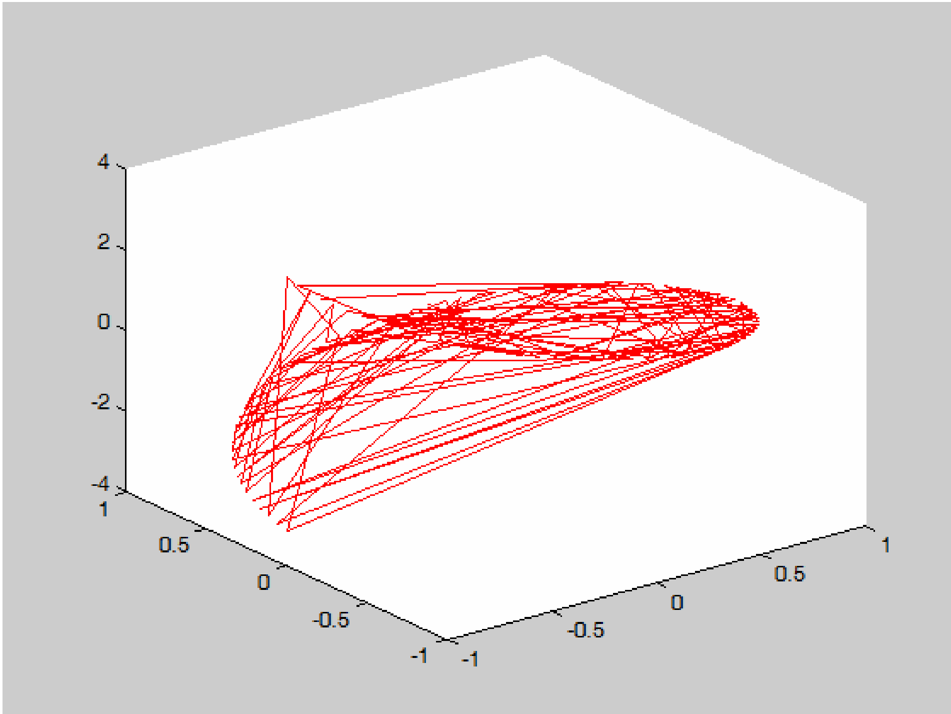
plot1dd.m

plot3(y1,y2,ix,'.')



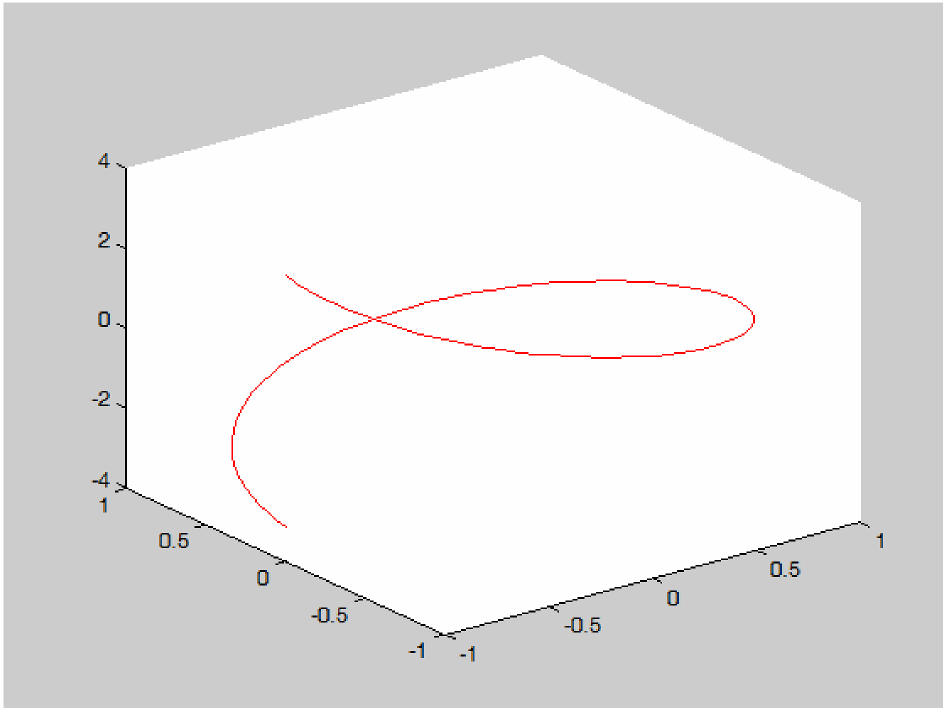
randperm

```
ind=randperm(length(ix));
plot3(y1(ind),y2(ind),ix(ind),'r');
```



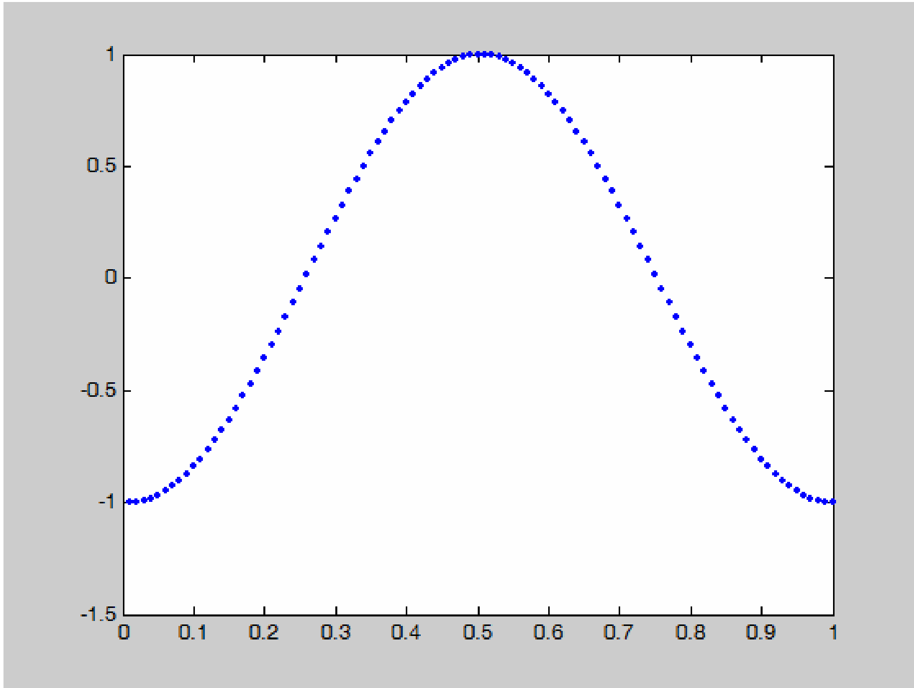
Sorting

```
[v ind]=sort(ix);
plot3(y1(ind),y2(ind),ix(ind),'r');
```

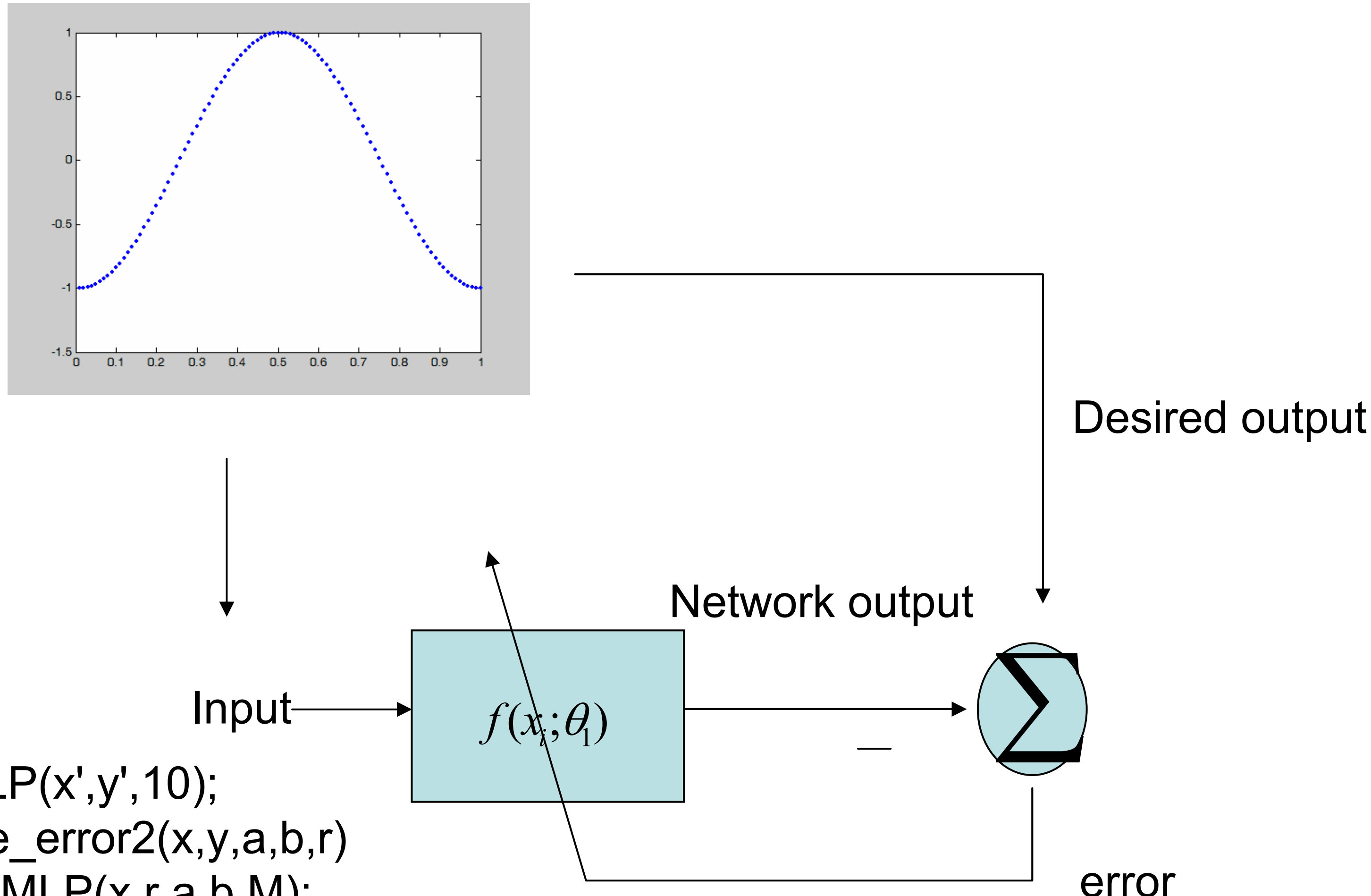


First coordinate

```
z=[y1;y2;ix];k=1;
y=z(k,:)/max(z(k,:));
x=(1:1:length(y))/length(y);
figure
plot(x,y,'.')
```

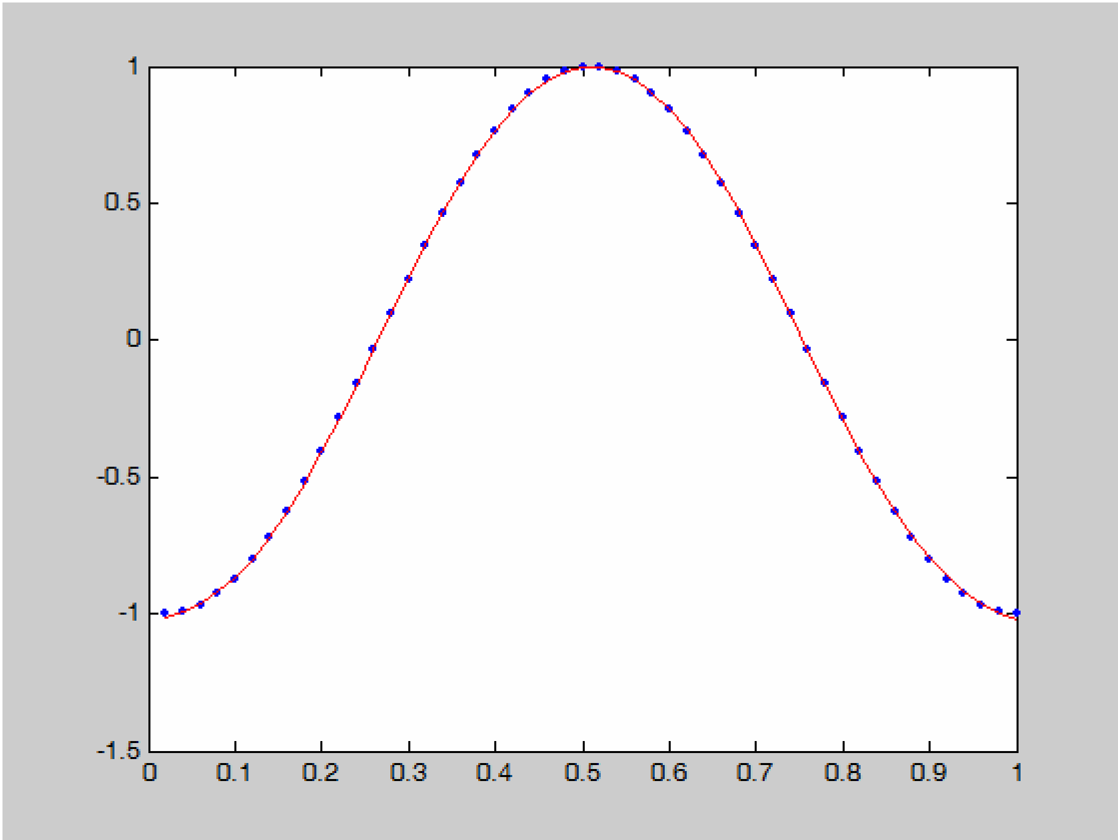


MLP learning




```
[a,b,r]=learn_MLP(x',y',10);
E=mean_square_error2(x,y,a,b,r)
y_hat(k,:)=eval_MLP(x,r,a,b,M);
hold on;
plot(x,y_hat,'r')
```

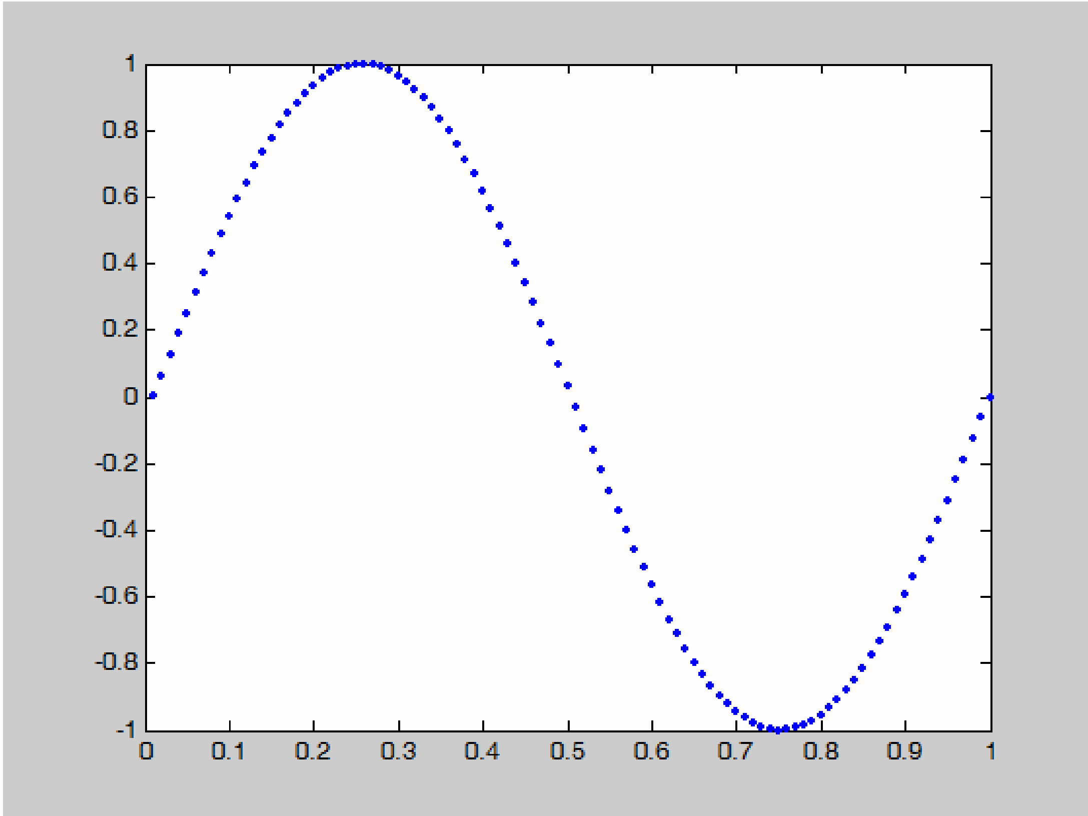
K=1



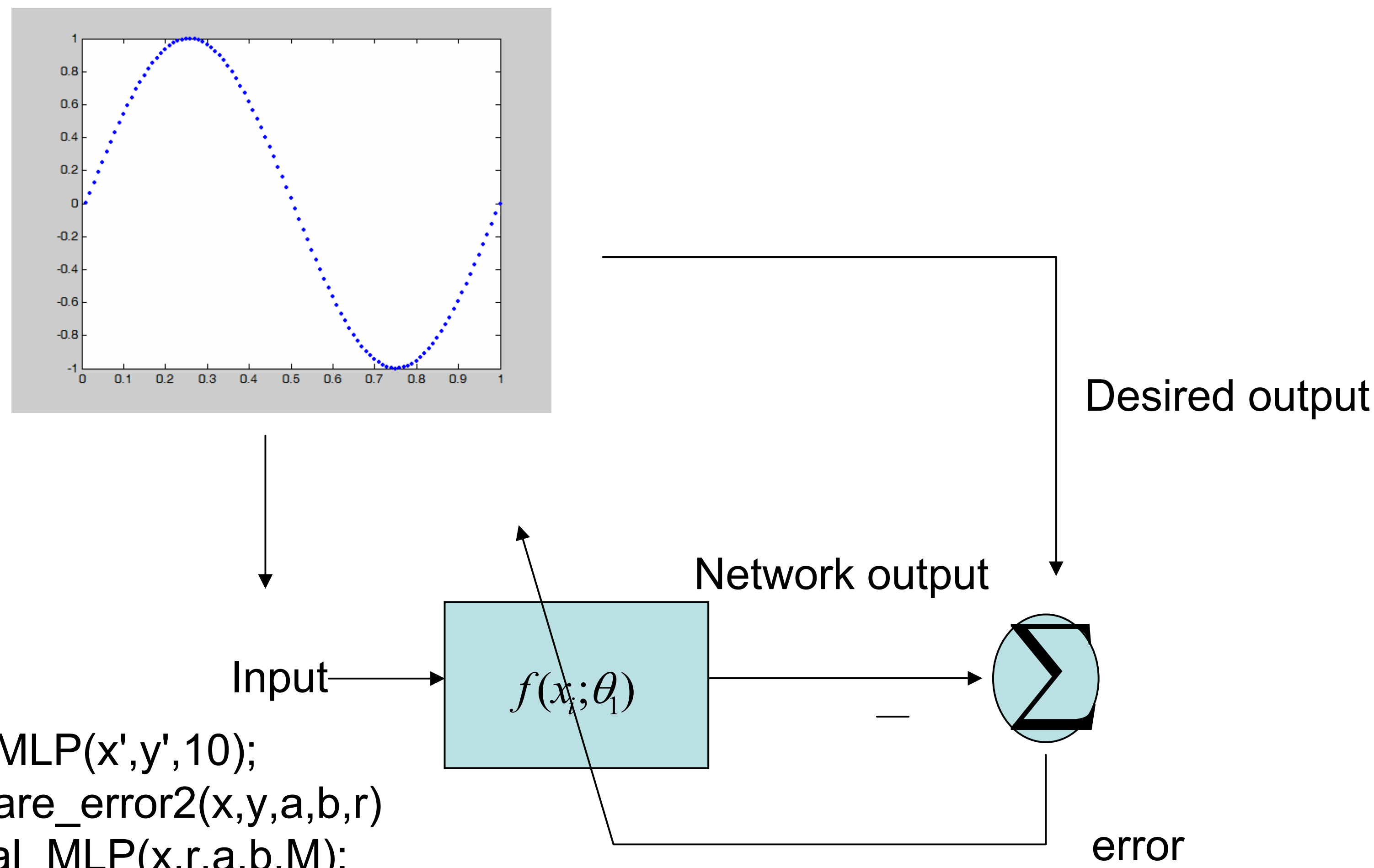
$$z_1 = f(x;\theta_1)$$

Second coordinate

```
z=[y1;y2;ix];k=2;
y=z(k,:)/max(z(k,:));
x=(1:1:length(y))/length(y);
figure
plot(x,y,'.')
```

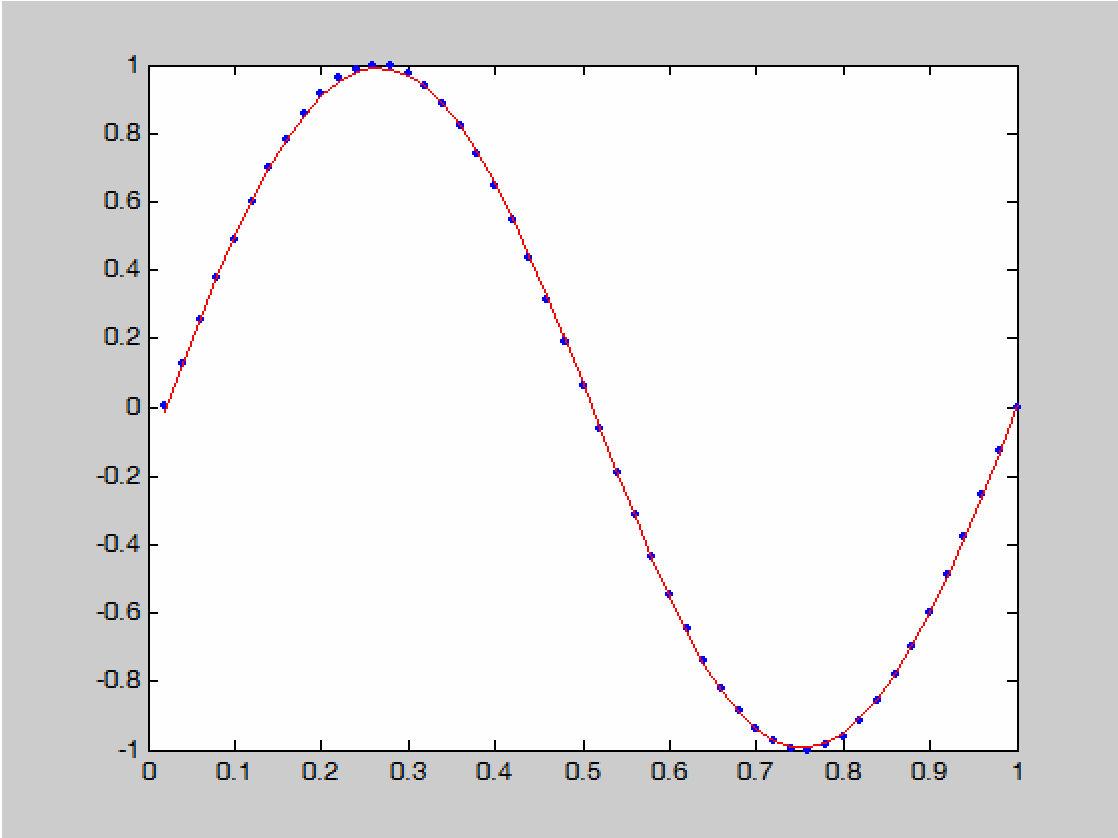


MLP learning




```
[a,b,r]=learn_MLP(x',y',10);
E=mean_square_error2(x,y,a,b,r)
y_hat(k,:)=eval_MLP(x,r,a,b,M);
hold on;
plot(x,y_hat,'r')
```

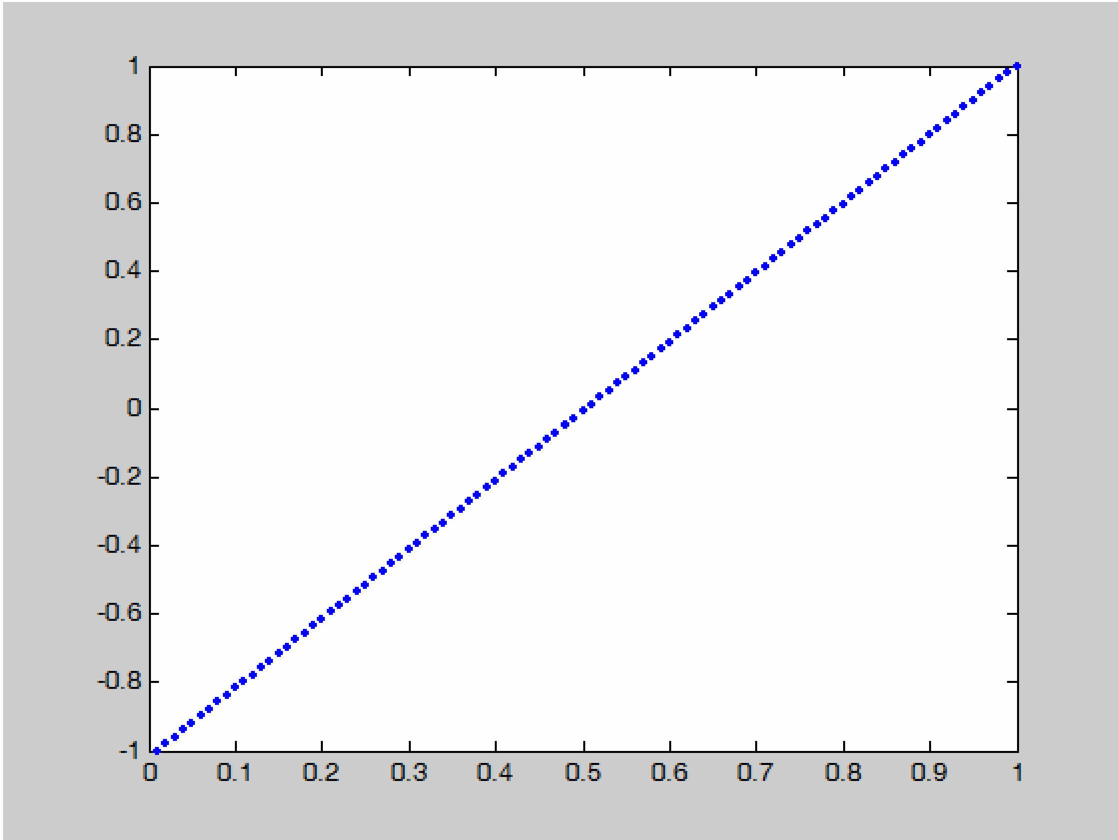
K=2



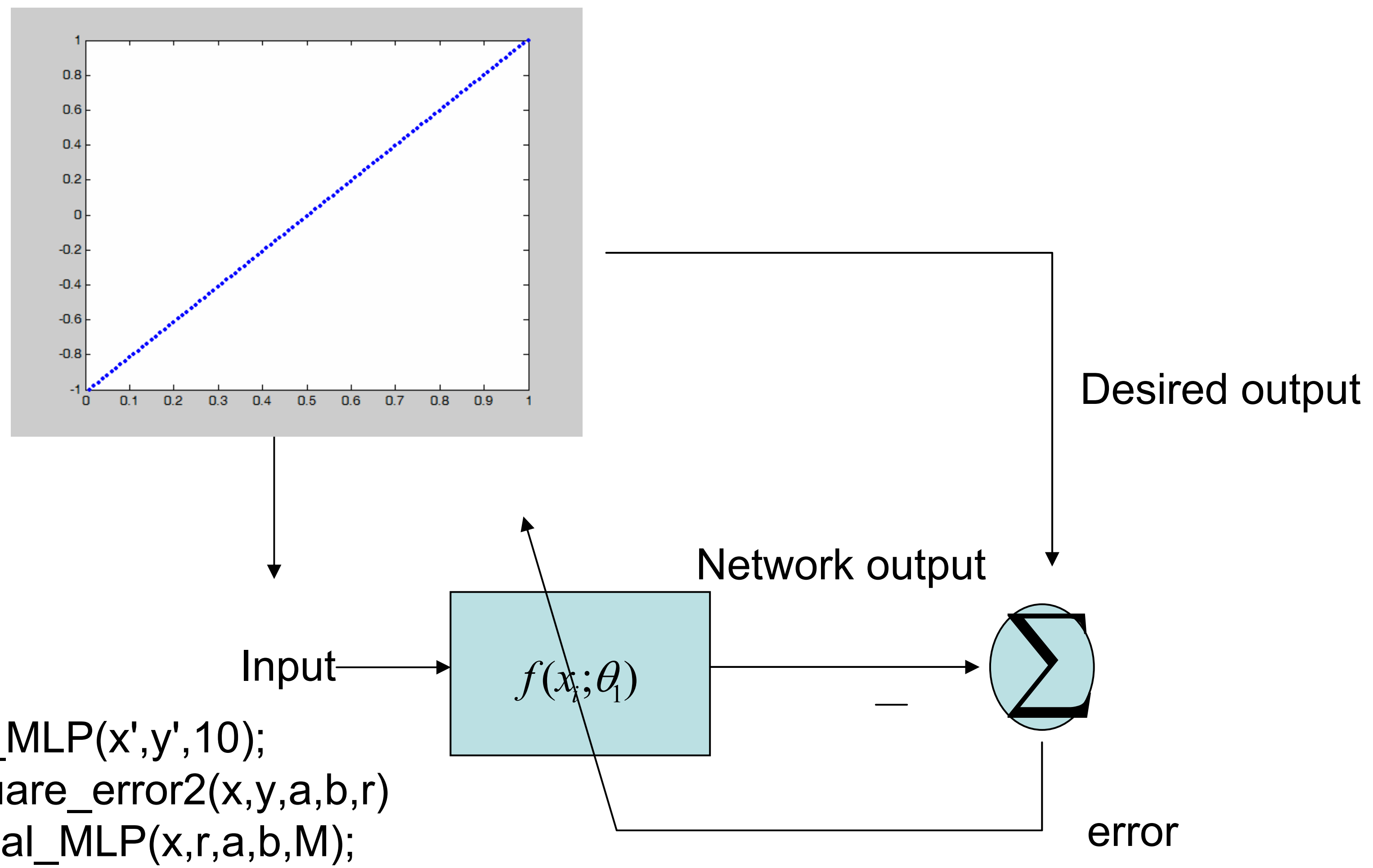
$$z_2 = f(x; \theta_2)$$

Third coordinate

```
z=[y1;y2;ix];k=3;
y=z(k,:)/max(z(k,:));
x=(1:1:length(y))/length(y);
figure
plot(x,y,'.')
```

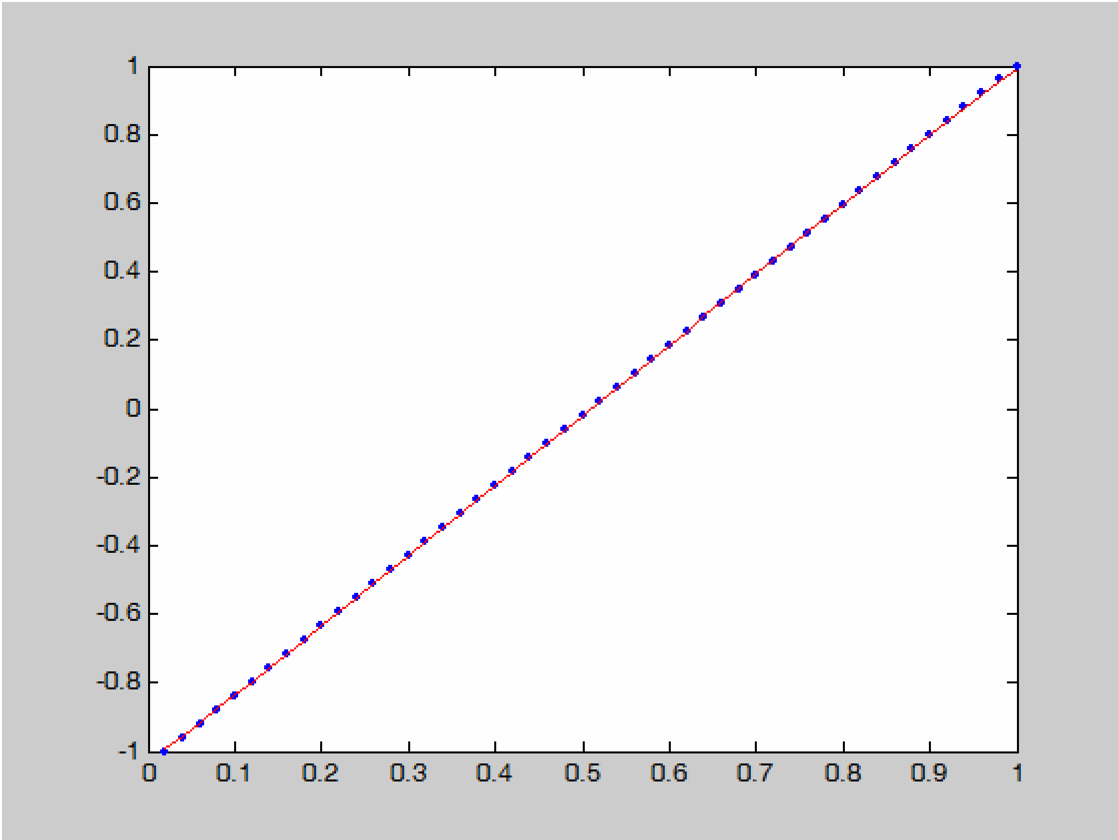


MLP learning



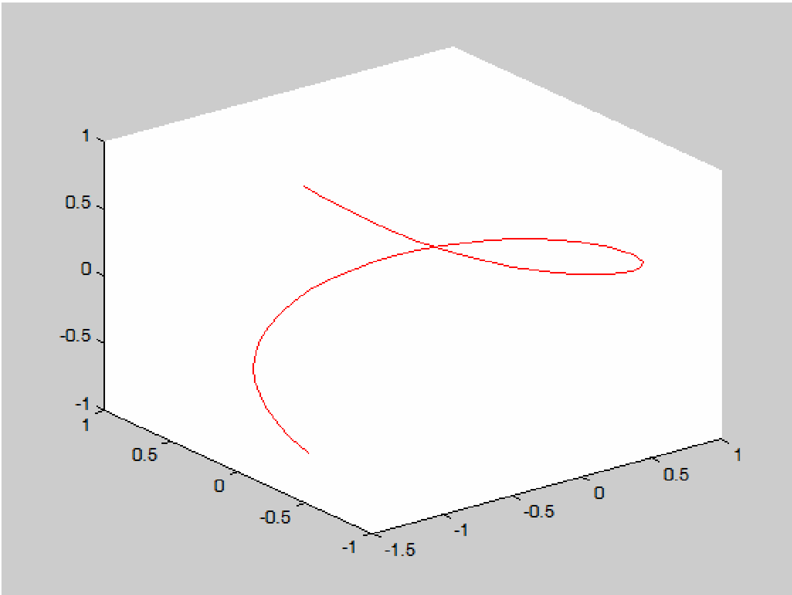

```
[a,b,r]=learn_MLP(x',y',10);
E=mean_square_error2(x,y,a,b,r)
y_hat(k,:)=eval_MLP(x,r,a,b,M);
hold on;
plot(x,y_hat,'r')
```

K=3



$$z_3 = f(x; \theta_3)$$

Learning three polar functions

- 

- fa1d_inv2.m

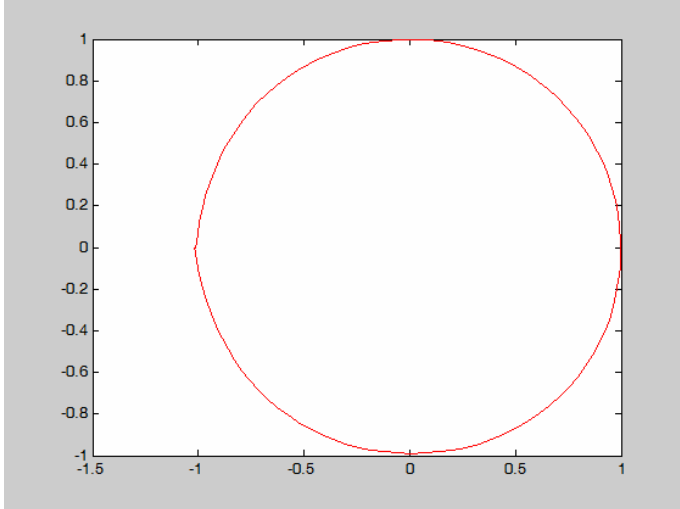
$$z_1 = f(x;\theta_1),$$

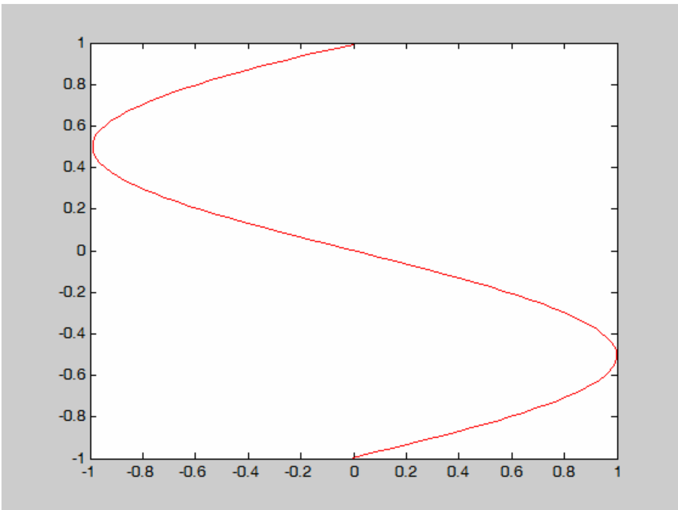
$$z_2 = f(x;\theta_2),$$

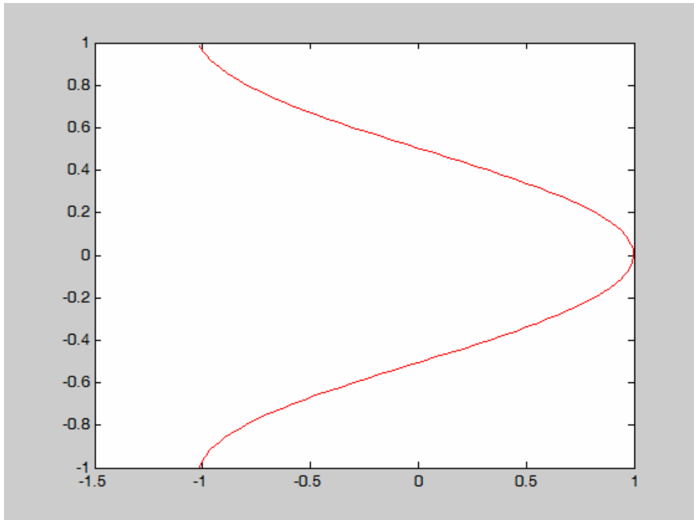
$$z_3 = f(x;\theta_3),$$

Reverse kinematics

```
plot(y_hat(1,:),y_hat(2,:),'r')
plot(y_hat(1,:),y_hat(3,:),'r')
plot(y_hat(2,:),y_hat(3,:),'r')
```







Inverse kinematics of two-link robot

inverse_kin.m

$$x_t, y_t$$

$$B^2 = x_t^2 + y_t^2$$

$$q_1 = \mathrm{atan2}\left(\frac{x_t}{y_t}\right)$$

$$q_2 = \mathrm{acos}\left(\frac{l_1^2 + B^2 - l_2^2}{2 l_1 B}\right)$$

$$P_1 = q_1 + q_2$$

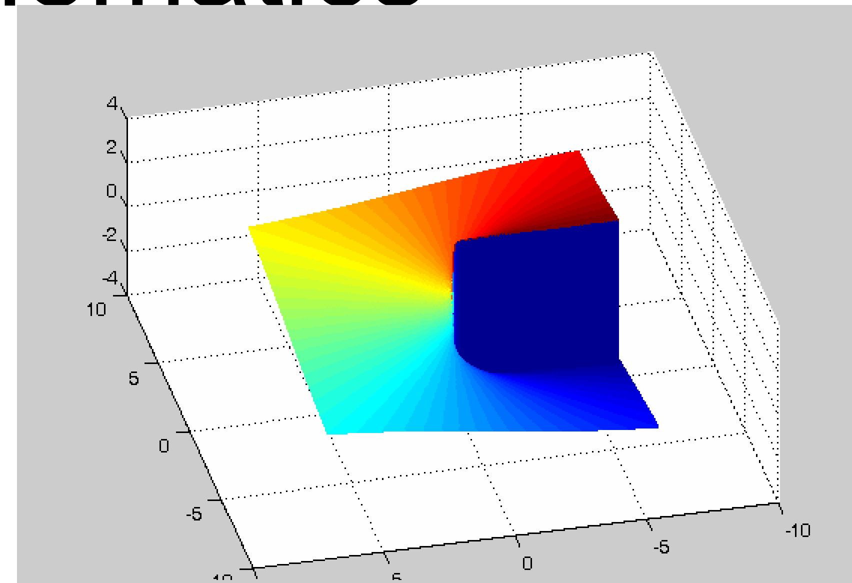
$$\theta = \mathrm{acos}\left(\frac{l_1^2 + l_2^2 - B^2}{2 l_1 l_2}\right)$$

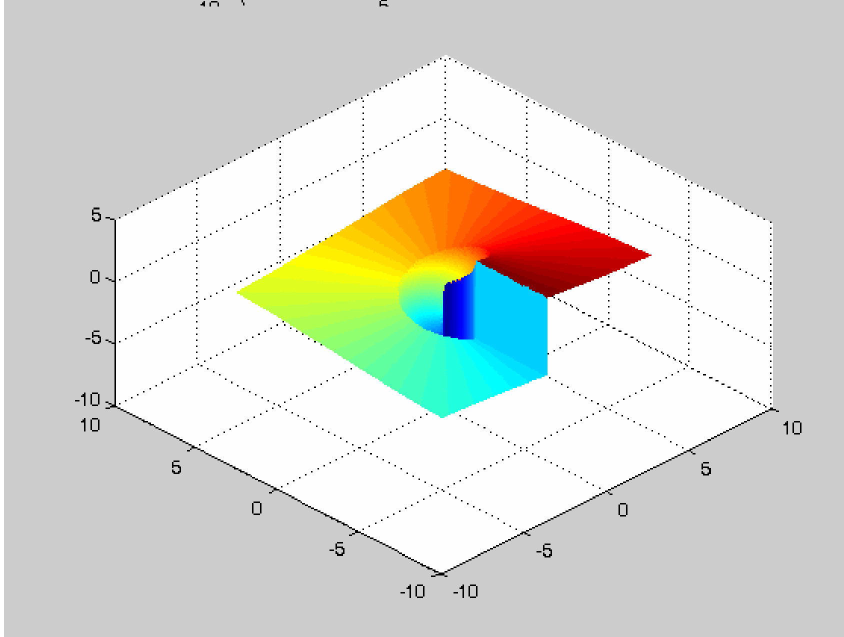
$$P_2 = P_1 - (\pi - \theta)$$

$$P_1, P_2$$

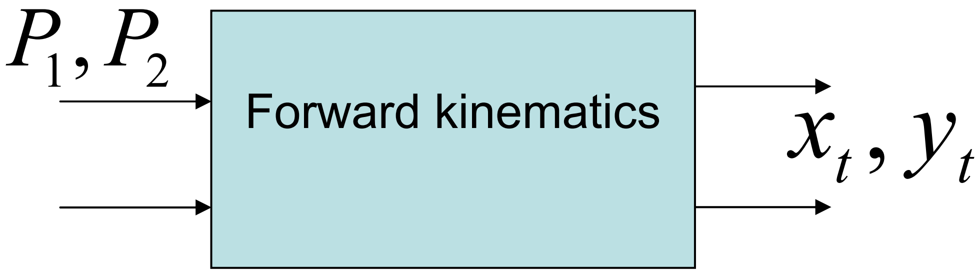
Inverse kinematics

```
range=2*pi;
x1=-range:0.02:range;
x2=x1;
for i=1:length(x1)
        [p1,p2]=inverse_kin(x1(i),x2,1,1);
        C1(i,:)=p1;
        C2(i,:)=p2;
end
mesh(x1,x2,C1);
figure;
mesh(x1,x2,C2);
```



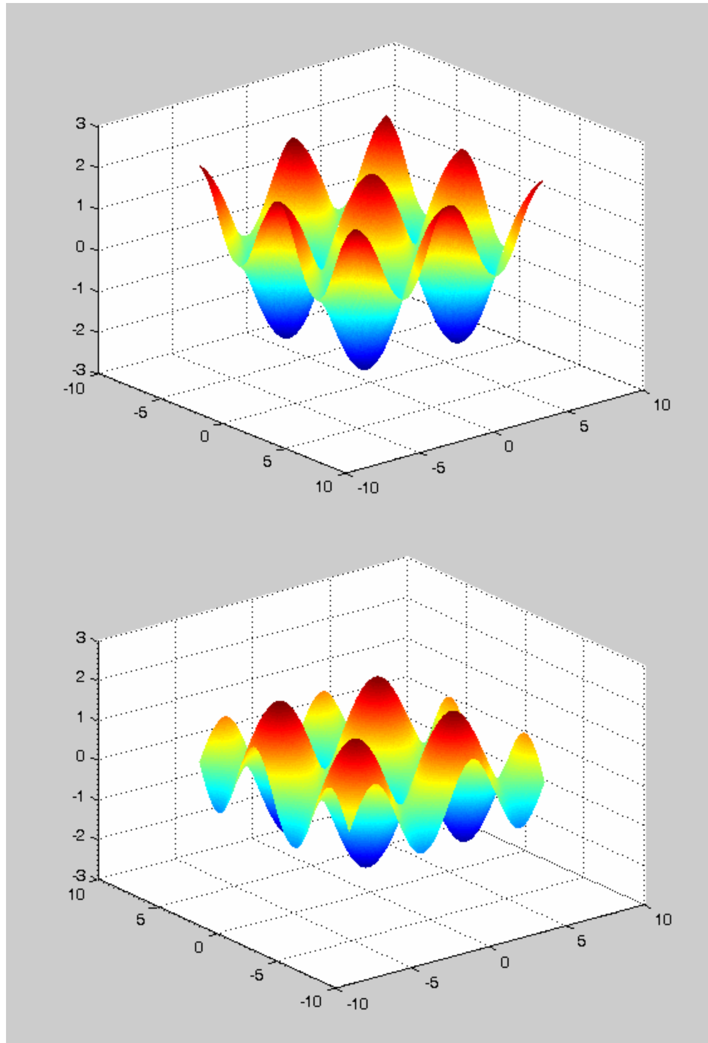


Data preparation

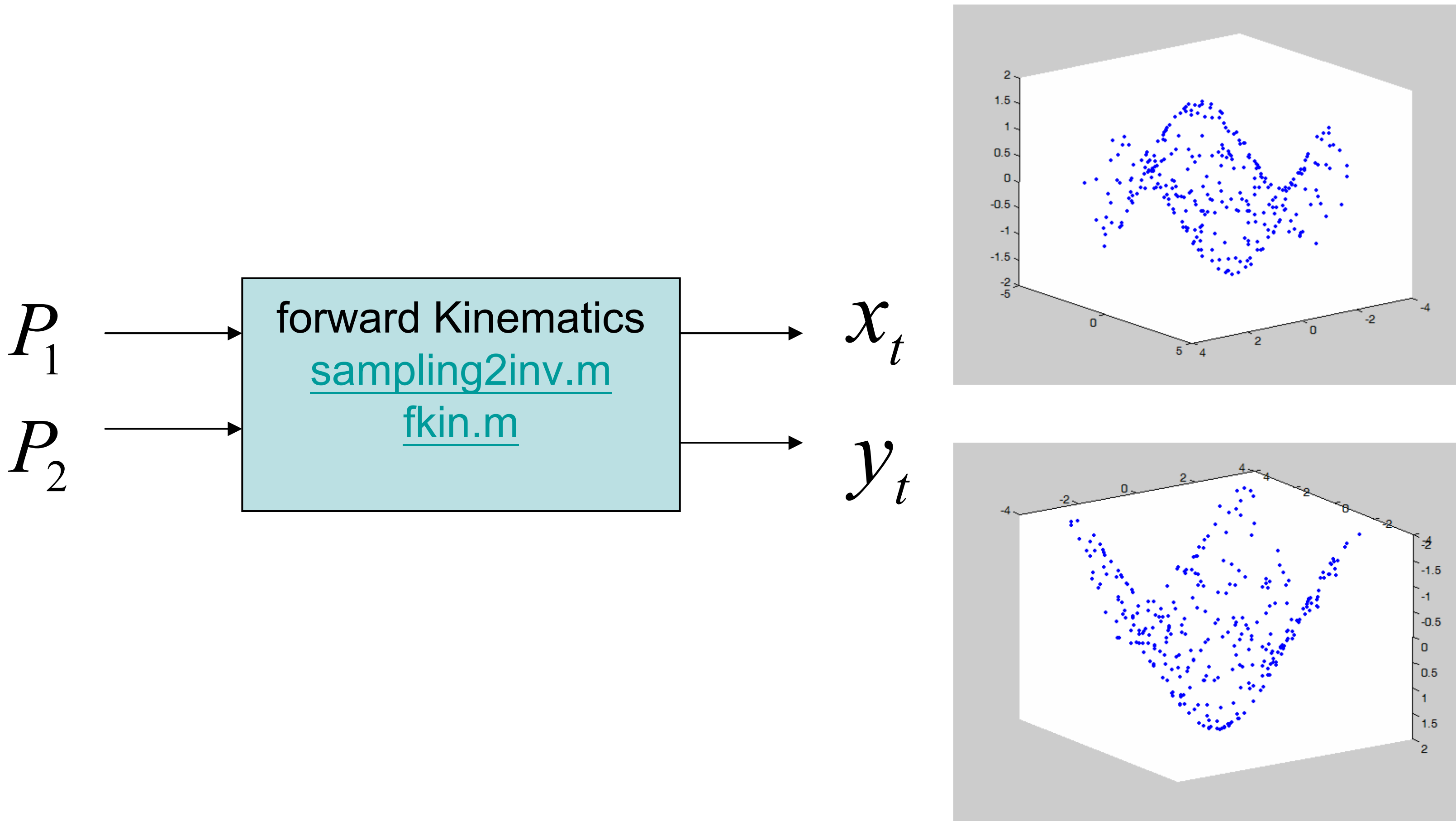


$$x_t = l_1 \cos(P_1) + l_2 \cos(P_2)$$

$$y_t = l_1 \sin(P_1) + l_2 \sin(P_2)$$

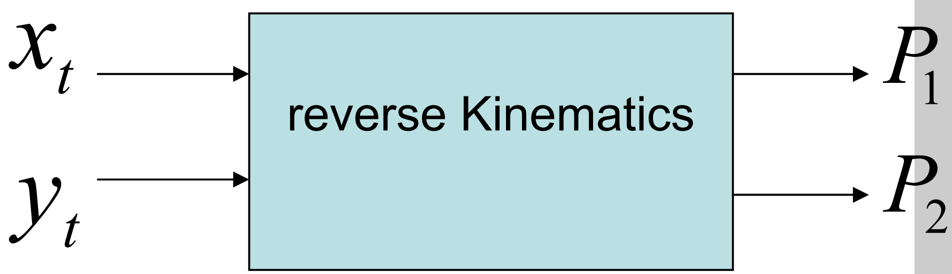


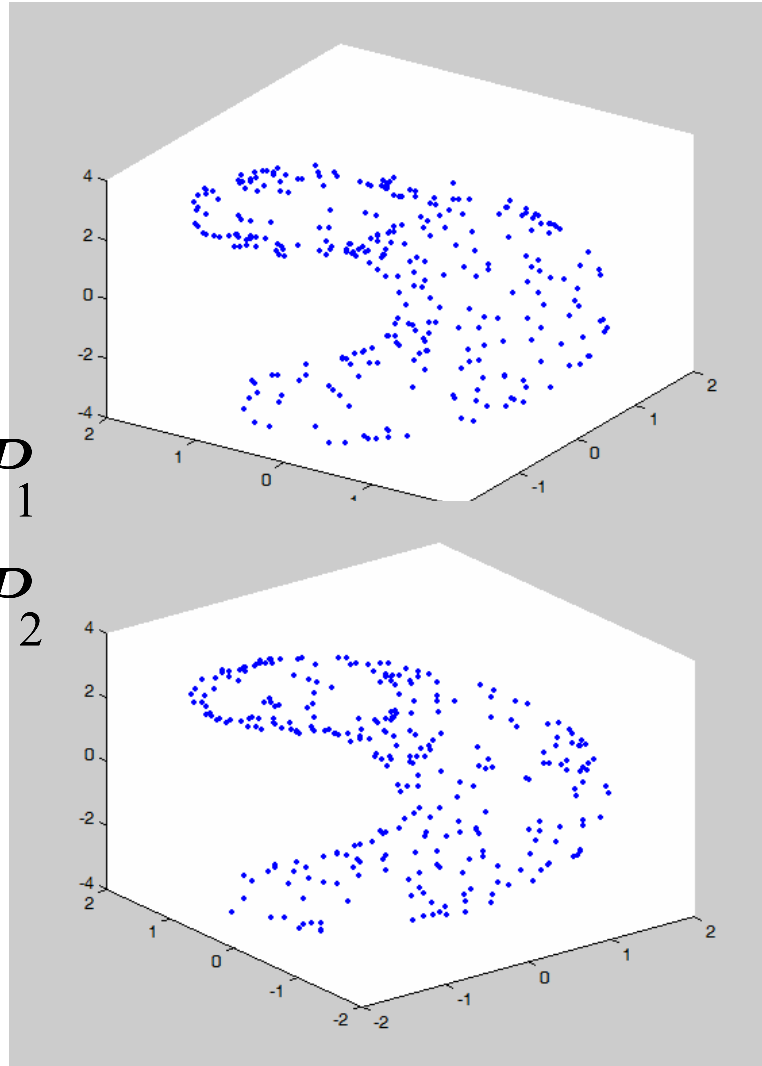
Sampling



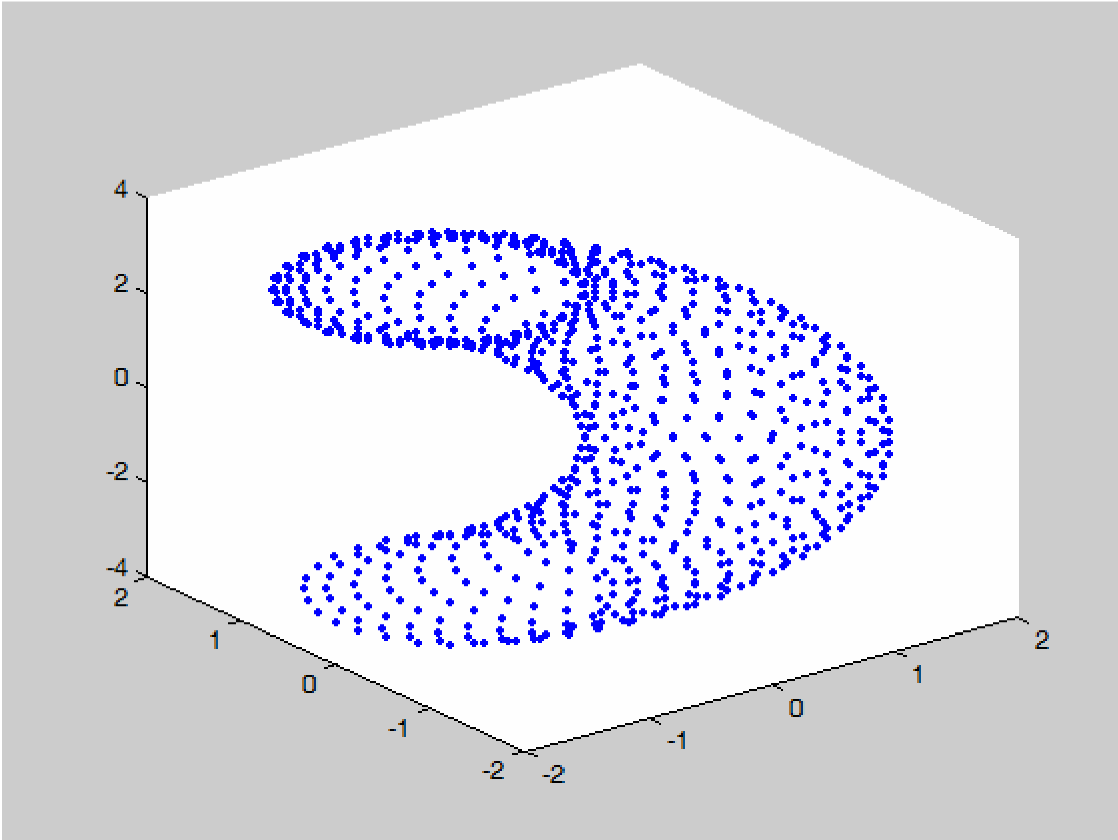
Swapping

```
p=x;
x=[tx;ty];
plot3(x(1,:),x(2,:),p(1,:),'.')
plot3(x(1,:),x(2,:),p(2,:),'.')
```



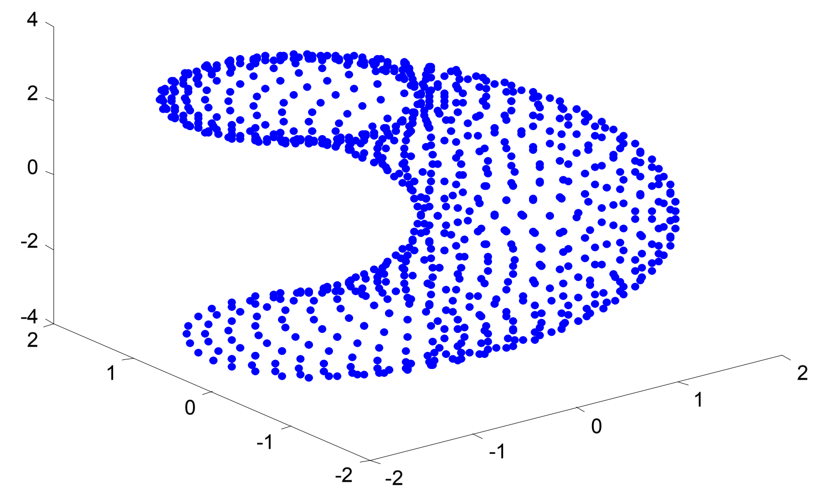


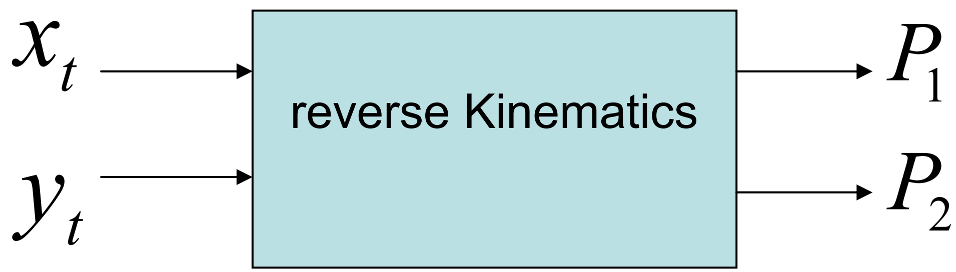
A rolled Plane



3d data points

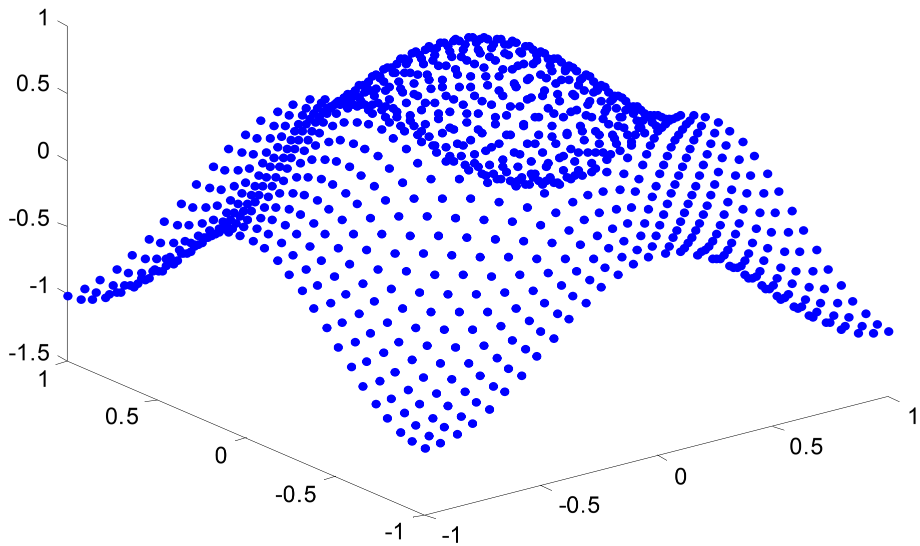
```
sampling2inv
  p=x;
  x=[tx;ty];
  plot3(x(1,:),x(2,:),p(1,:),'.')
  figure
  plot3(x(1,:),x(2,:),p(2,:),'.')
 z=[x;p(1,:)];
```



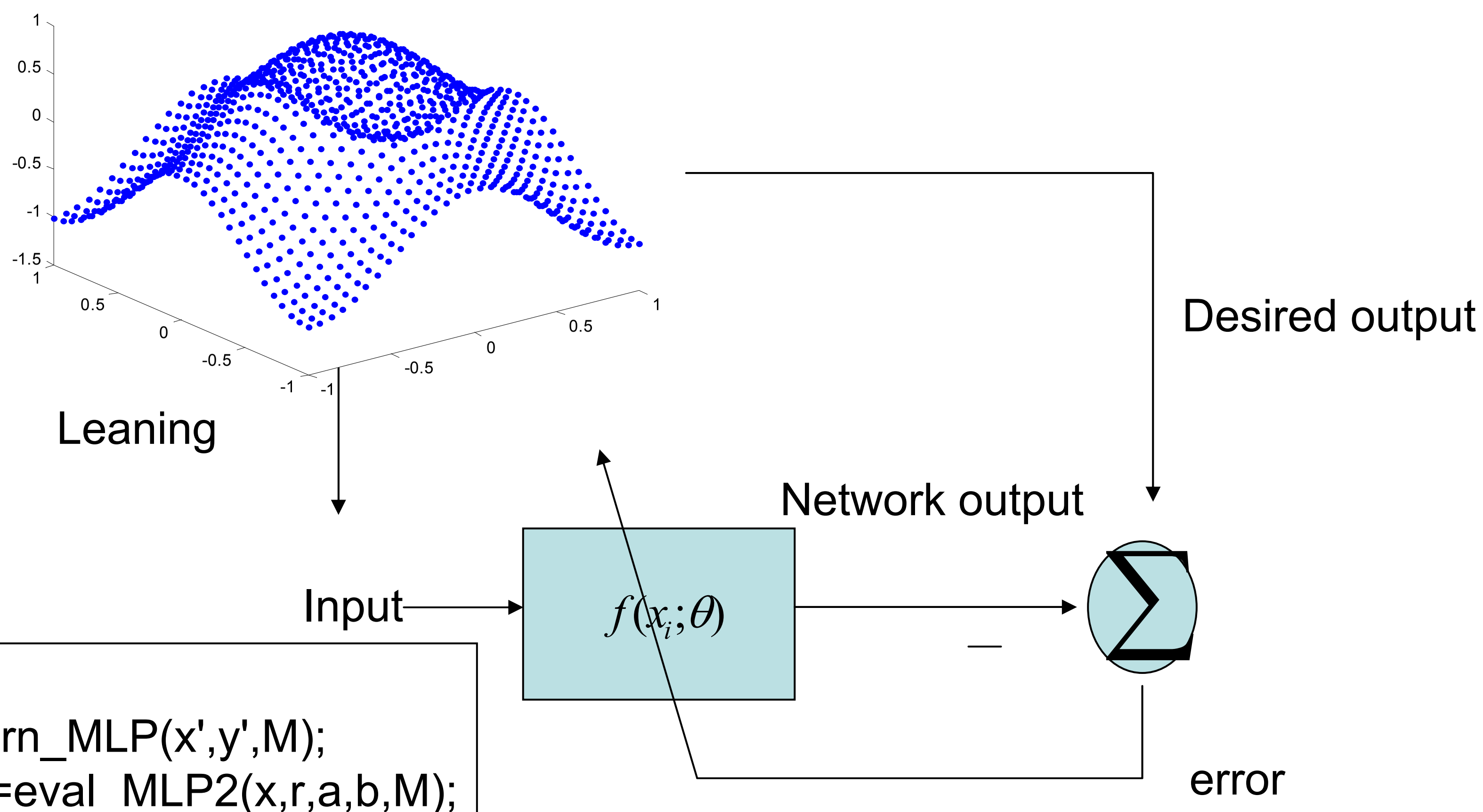


Paired data for the first polar function

```
k=1;
y=z(k,:)/max(z(k,:));
x=[x1;x2];
x=x/max(max(x));
figure;
plot3(x(1,:),x(2,:),y,'.')
```

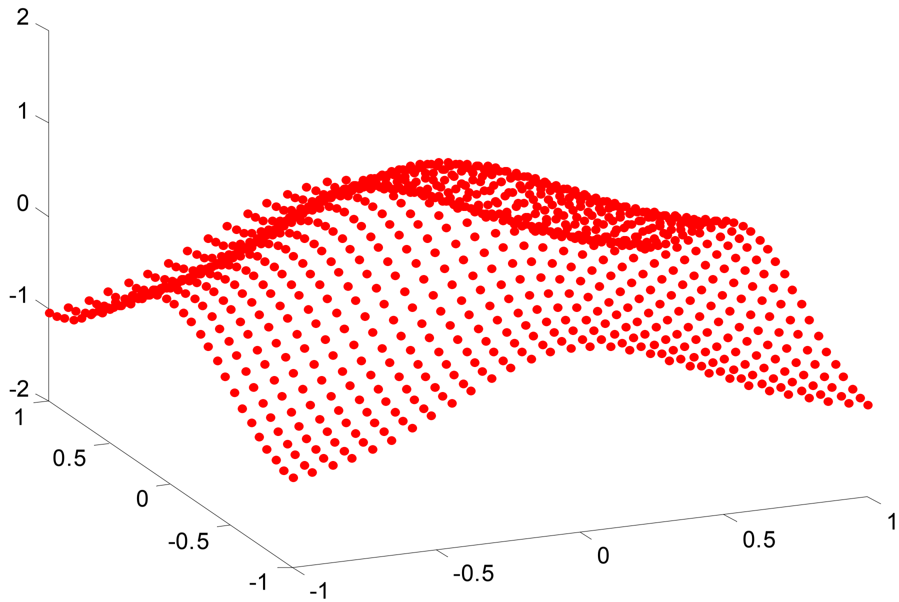


Learning the first polar function



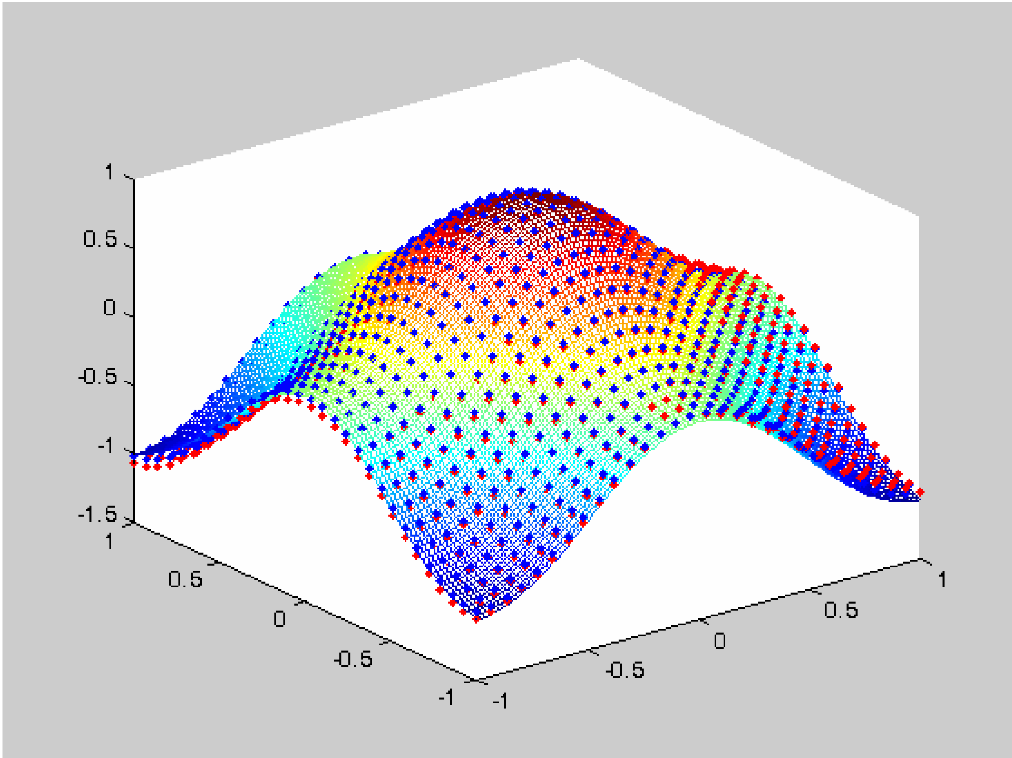

```
M=20;
[a,b,r]=learn_MLP(x',y',M);
y_hat(k,:)=eval_MLP2(x,r,a,b,M);
hold on;
plot3(x(1,:),x(2,:),y_hat(k,:),'r.')
```

Network outputs



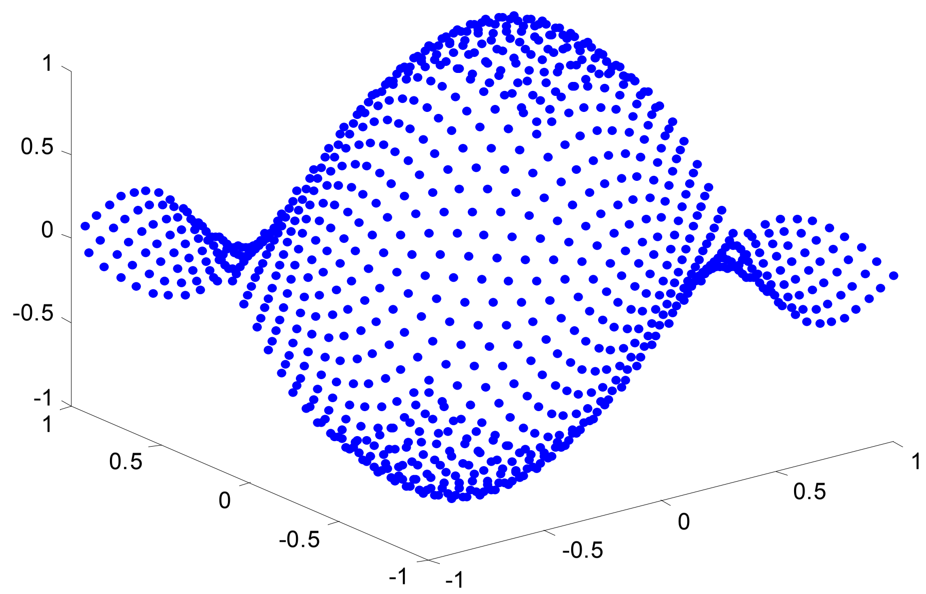
Mesh

```
a1=-1:0.02:1;
a2=a1;
for i=1:length(a1)
   C(i,:)=eval_MLP2([a1(i)*ones(size(a2));a2],r,a,b,M);;
end
mesh(a1,a2,C);
```

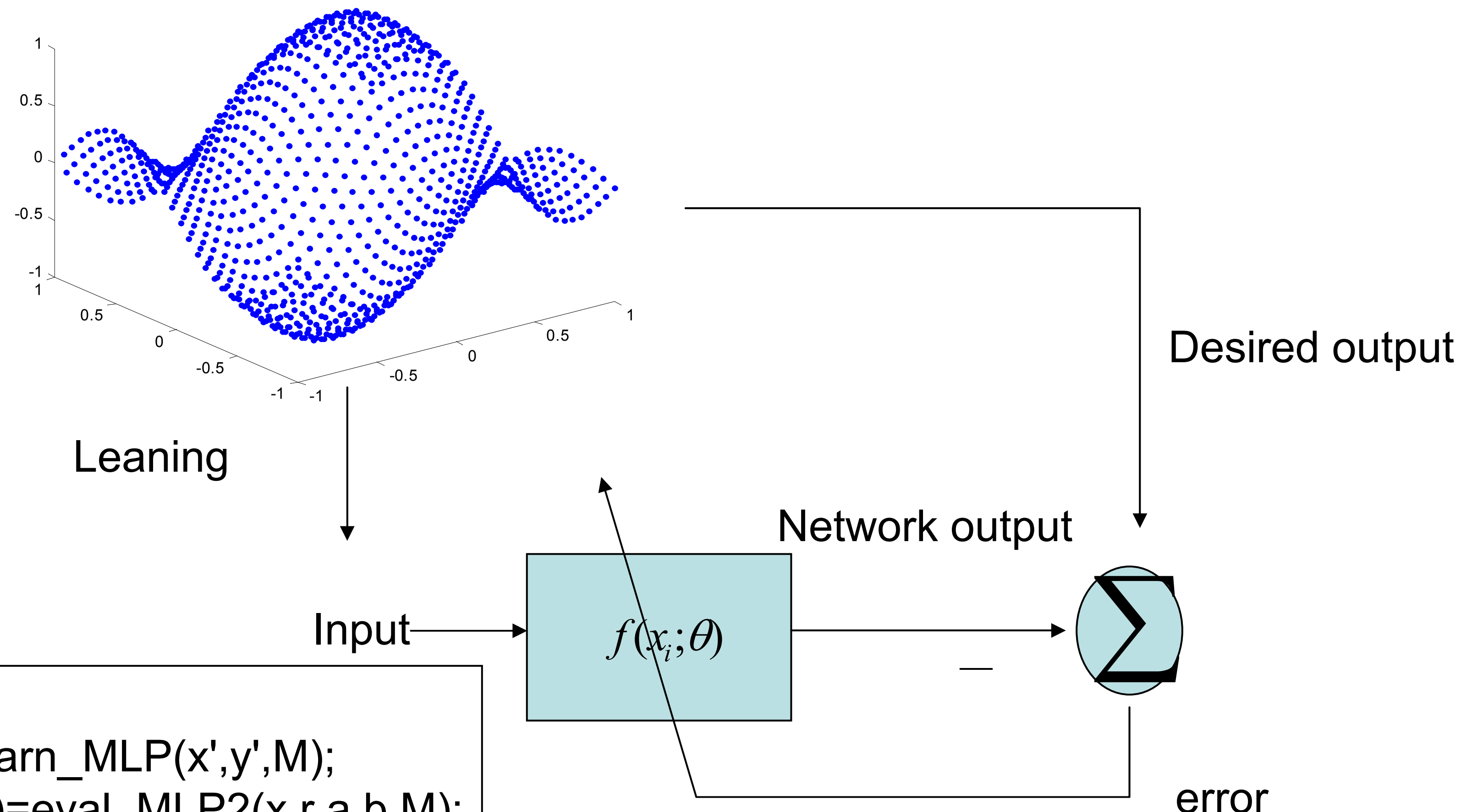


Paired data for the second polar function

```
k=2;
y=z(k,:)/max(z(k,:));
x=[x1;x2];
x=x/max(max(x));
figure;
plot3(x(1,:),x(2,:),y,'.')
```

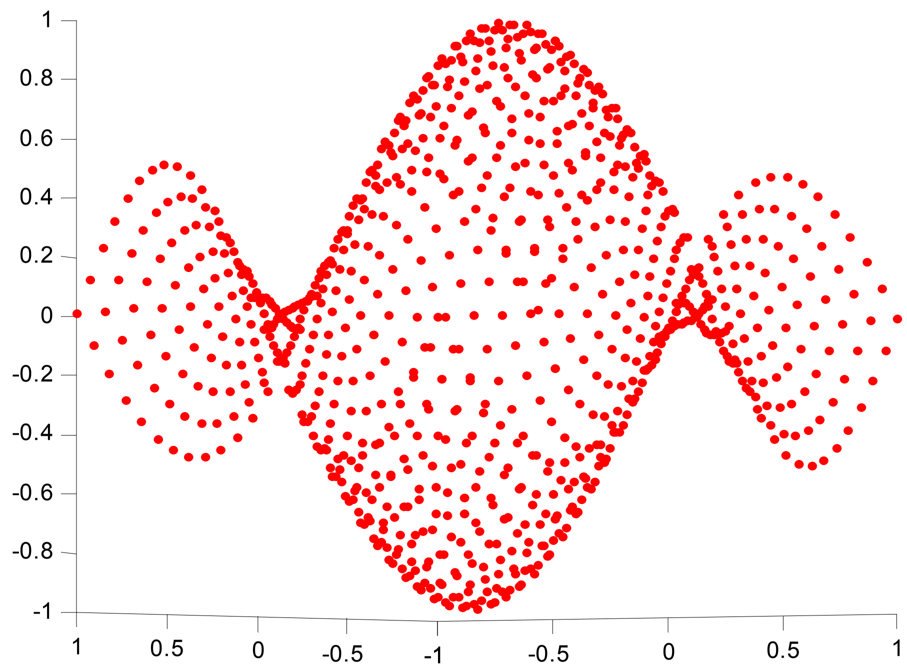


Learning the second polar function



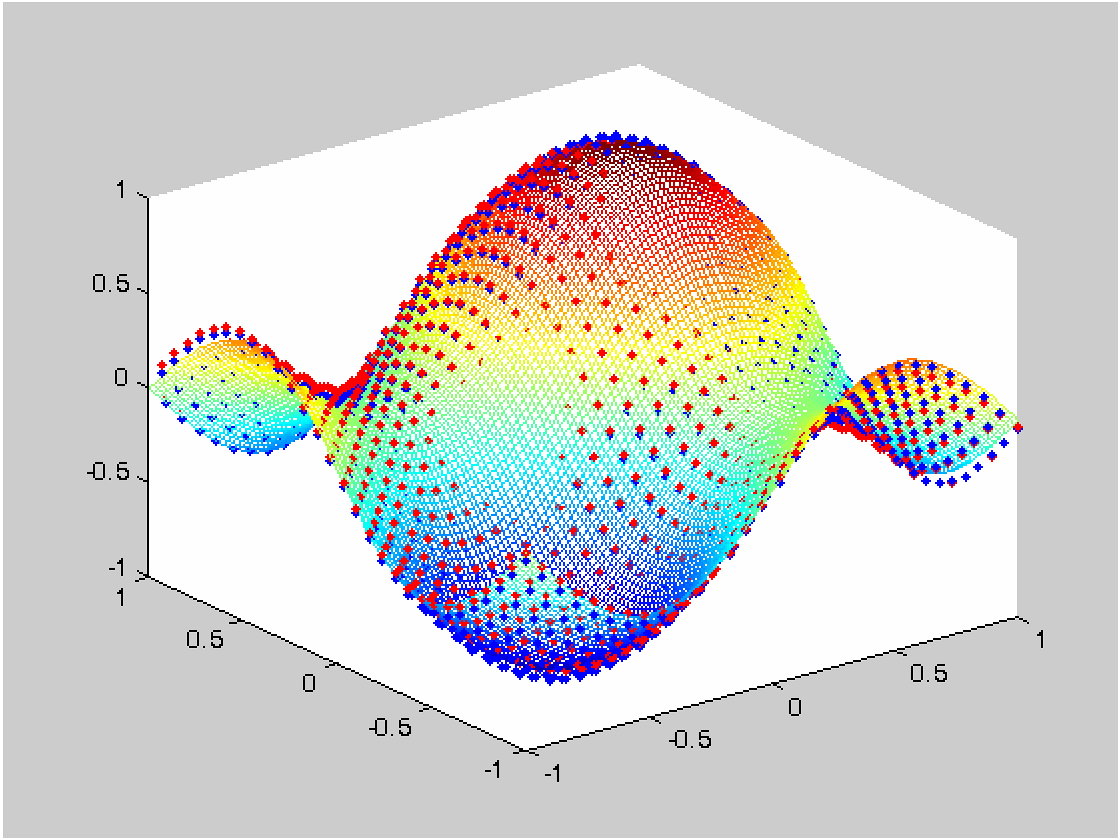

```
M=20;
[a,b,r]=learn_MLP(x',y',M);
y_hat(k,:)=eval_MLP2(x,r,a,b,M);
hold on;
plot3(x(1,:),x(2,:),y_hat(k,:),'r.')
```

Network outputs



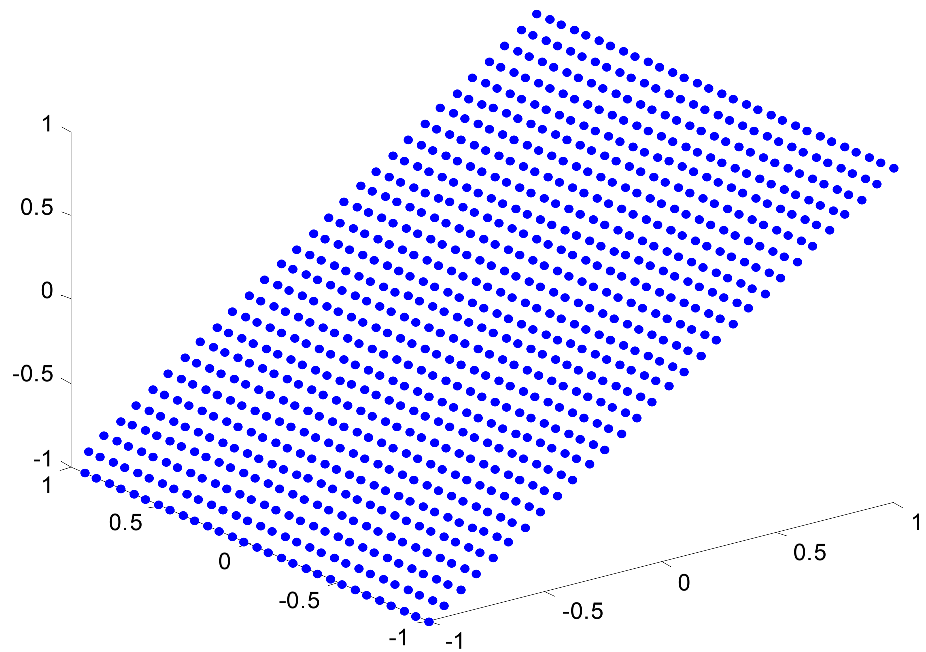
Mesh

```
a1=-1:0.02:1;
a2=a1;
for i=1:length(a1)
    C(i,:)=eval_MLP2([a1(i)*ones(size(a2));a2],r,a,b,M);;
end
mesh(a1,a2,C);
```

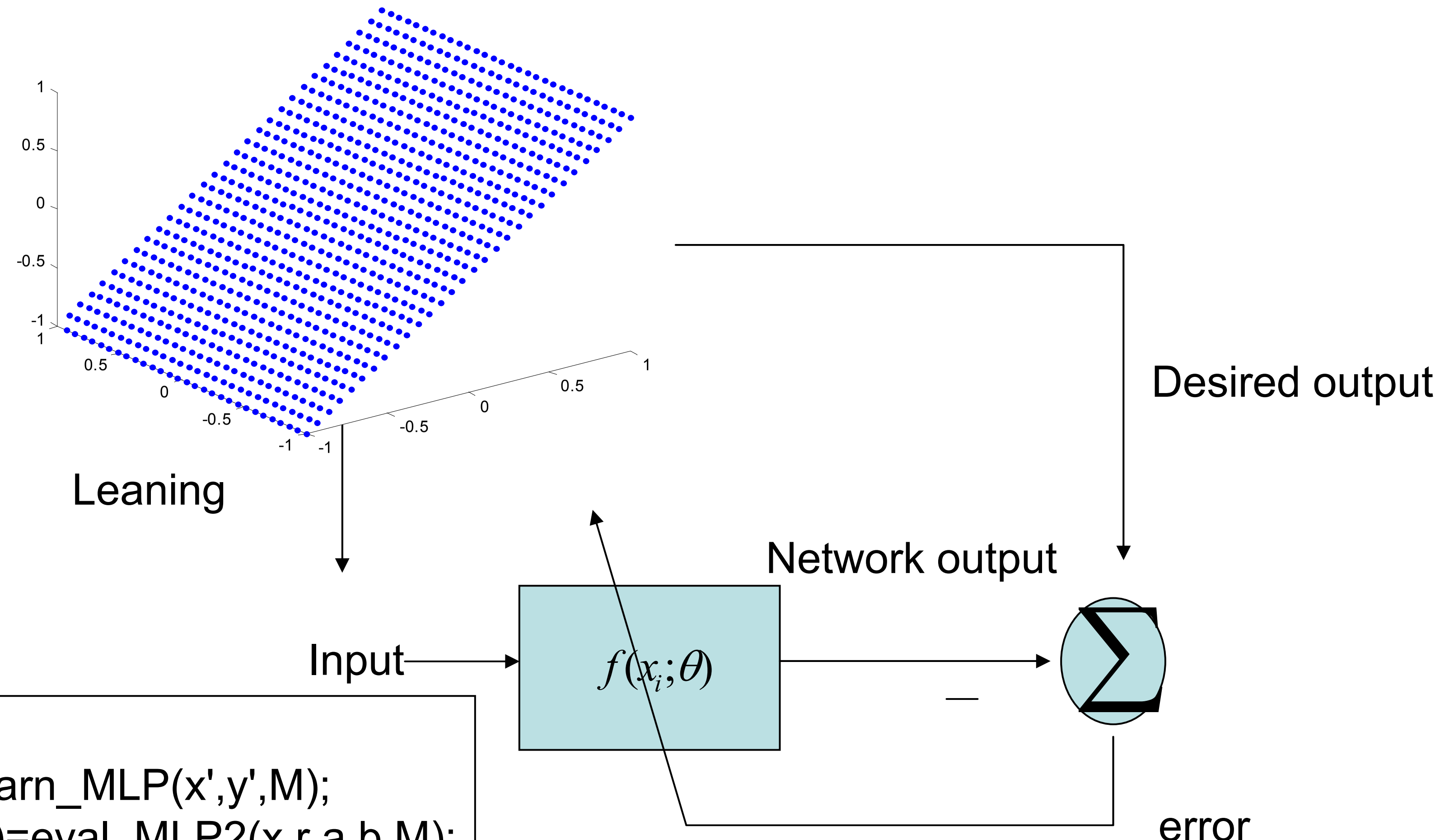


Paired data for the third polar function

```
k=3;
y=z(k,:)/max(z(k,:));
x=[x1;x2];
x=x/max(max(x));
figure;
plot3(x(1,:),x(2,:),y,'.')
```

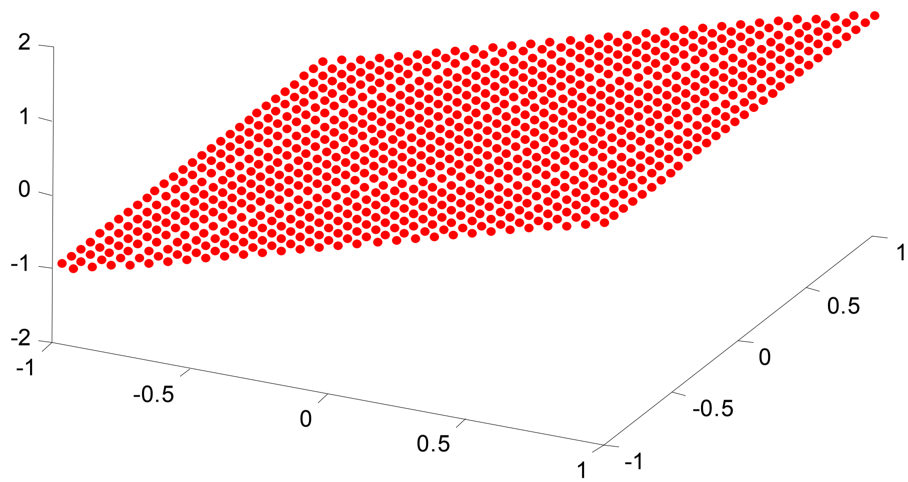


Learning the second polar function



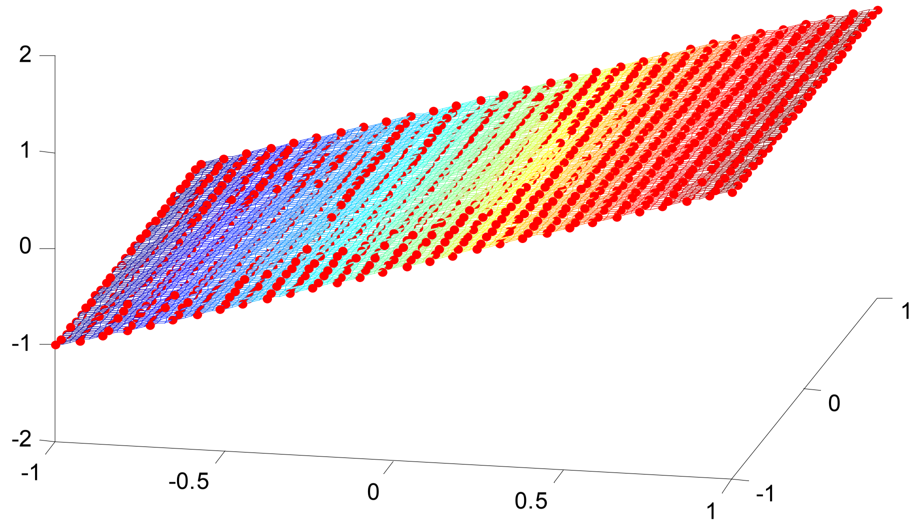

```
M=20;
[a,b,r]=learn_MLP(x',y',M);
y_hat(k,:)=eval_MLP2(x,r,a,b,M);
hold on;
plot3(x(1,:),x(2,:),y_hat(k,:),'r.')
```

Network outputs



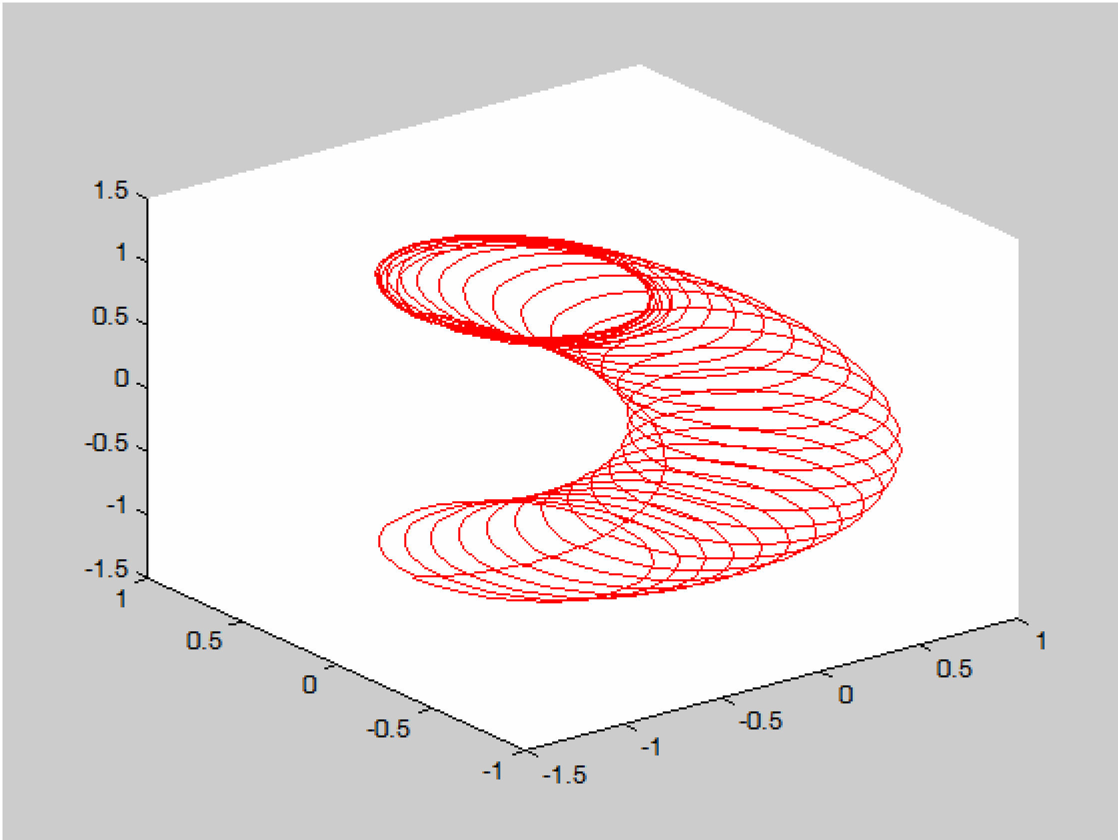
Mesh

```
a1=-1:0.02:1;
a2=a1;
for i=1:length(a1)
    C(i,:)=eval_MLP2([a1(i)*ones(size(a2));a2],r,a,b,M);;
end
mesh(a1,a2,C')
```



Demo_inv

demo_inv.m



Learning inverse kinematics for planar robot

$$x_t = l_1 \cos(P_1) + l_2 \cos(P_2) + l_3 \cos(P_3)$$

$$y_t = l_1 \sin(P_1) + l_2 \sin(P_2) + l_3 \sin(P_3)$$