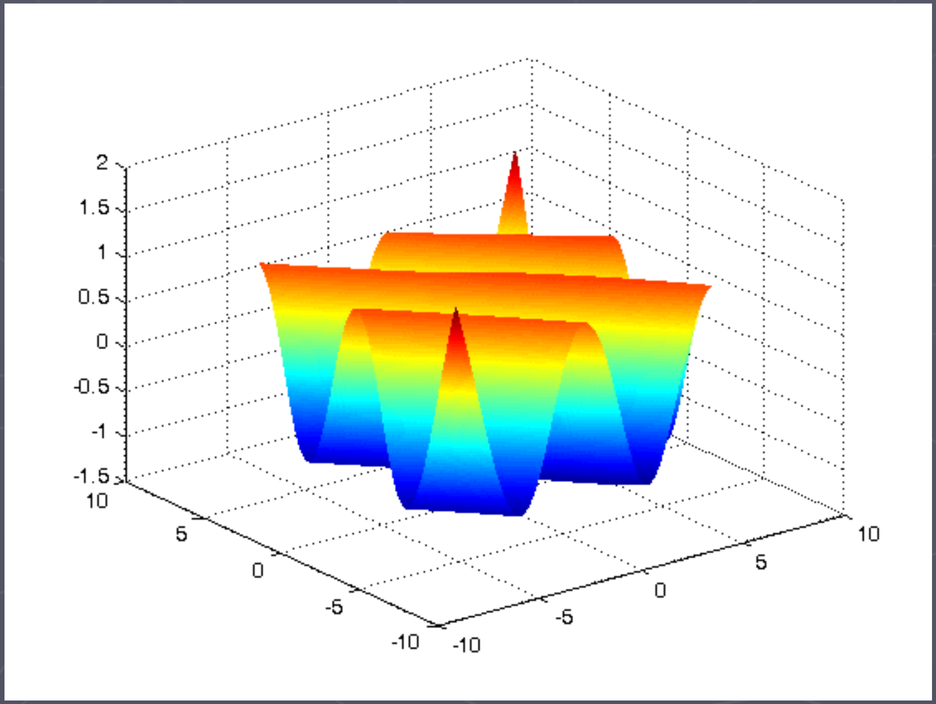
Function Approximation by NNSYSID

- Data driven function approximation
  - MLP network
  - Unconstrained optimization



Objective function

$$E_{\mathrm{MLP}}(a,b,r) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i;a,b,r))^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left(\sum_{m=1}^{M} r_m \tanh(a_m x + b_m) + r_0\right)\right)^2$$

Unconstrained Optimization

- Non-polynomial,
- High dimensional function
- Unknowns: 3M+1

Unconstrained Optimization

- Gradient method
- Gauss-Newton method
- Levenberg-Marquardt method

Find a, b, r to minimize

$$E_{\mathrm{MLP}}(a,b,r)$$

Unconstrained Optimization

$$\text{Find}\,\theta\ \text{to minimize}\ E_{\text{MLP}}(\theta)$$

$$\theta = \{a, b, r\}$$

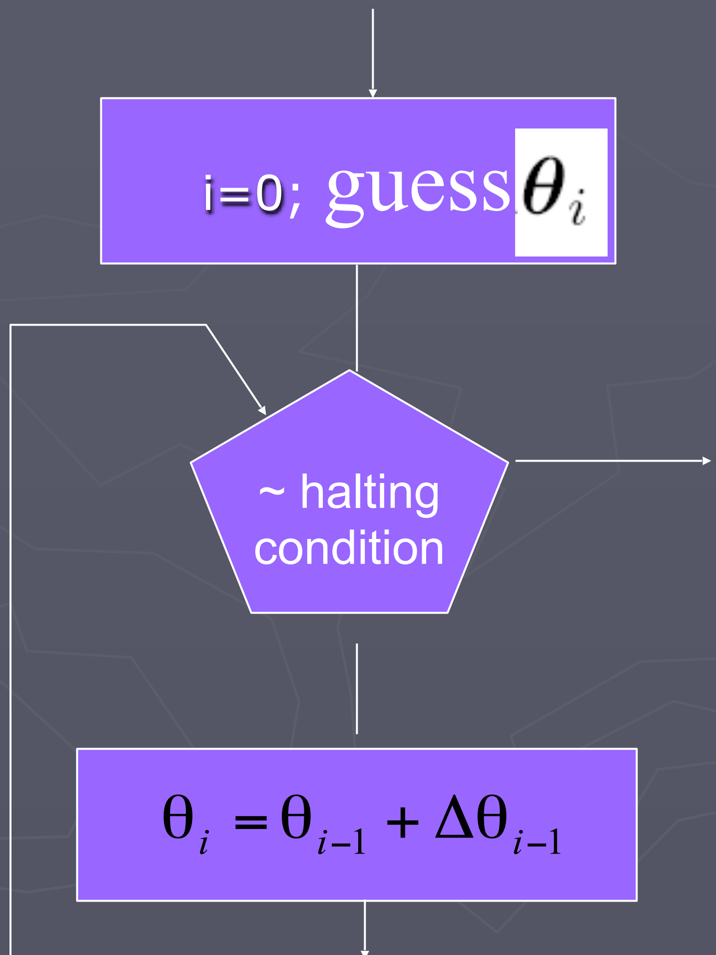
Iterative approach

1. Initialize $\boldsymbol{\theta}_i$ with i=0
2. Determine $\Delta\theta_i$
3. Update network parameters

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \boldsymbol{\Delta\theta}_i$$

4. If halting condition holds, exit
   otherwise i=i+1, go to step 2

Flow Chart



i=0; guess θ_i
~ halting condition
$\theta_i = \theta_{i-1} + \Delta\theta_{i-1}$

Gradient method

$$\Delta\boldsymbol{\theta}_i \propto -\frac{dE_S(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_i}$$

- Derive the derivative of E_S with respect to θ
- Substitute θ_i to $\frac{dE_S(\theta)}{d\theta}$

Install NNSYSID

1. Install

   [The NNSYSID Toolbox](http://www.iau.dtu.dk/research/control/nnsysid.html)

   http://www.iau.dtu.dk/research/control/nnsysid.html

2. Download

   learn_MLP

   By Levenberg Marquardt method

   http://134.208.26.59/AdvancedNA/LMlearningNNSYSID/learn_MLP.m

3. Set path to recruit the directory where NNSYSID.zip is extracted

Data creation

```
fstr=input('input a function: x.^2+cos(x) :','s');
fx=inline(fstr);
```

Get a function

```
range=2*pi;
x=linspace(-range,range);
y=fx(x);max_y=max(y);
plot(x,y/max_y);
```

Function plotting

Sampling

```
N=input('keyin sample size:');
x=rand(1,N)*2*range-range;
n=rand(1,N)*0.1-0.05;
y=fx(x)/max_y+n;
plot(x,y,'.');
```

MLP learning

```
M=5;
[a,b,r]=learn_MLP(x',y',M);
```

MLP learning

```
x=linspace(-range,range);
y=eval_MLP(x,r,a,b,M);
plot(x,y,'r')
```

MLP evaluation

Data Driven Function Approximation

source code

http://134.208.26.59/AdvancedNA/LMlearningNNSYSID/fa1d.m

mean_square_error2.m

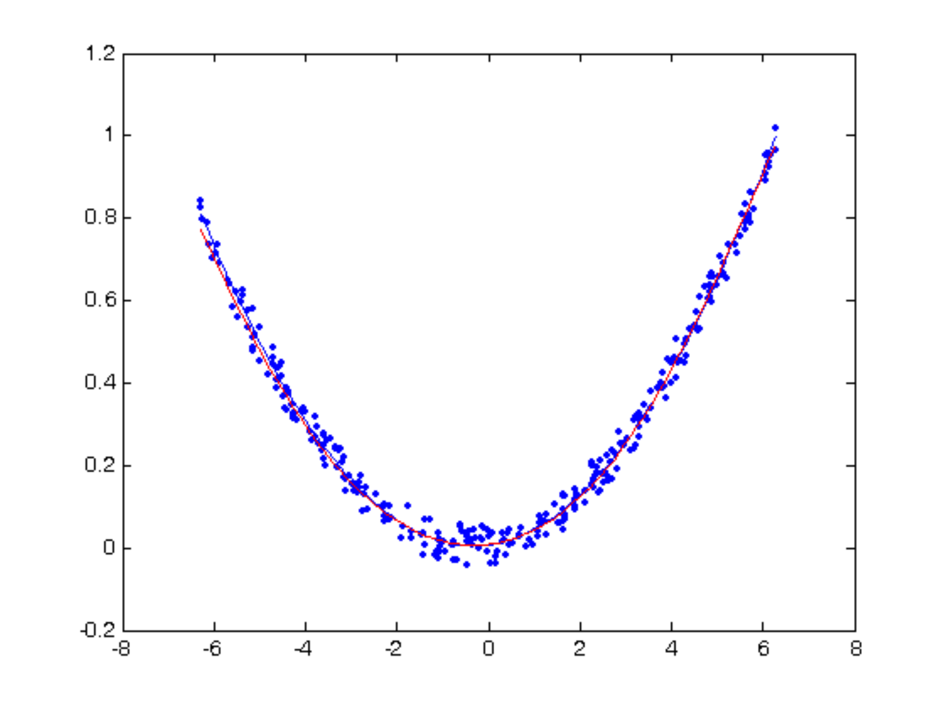
http://134.208.26.59/AdvancedNA/LMlearningNNSYSID/mean_square_error2.m

eval_MLP.m

http://134.208.26.59/AdvancedNA/LMlearningNNSYSID/eval_MLP.m

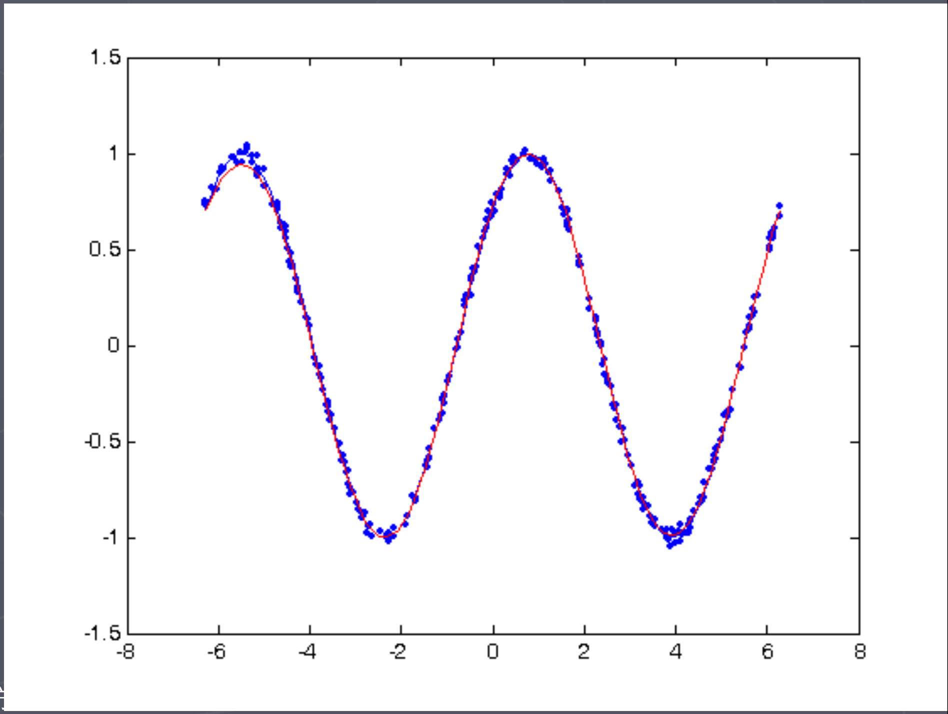
Example

>> fa1d
input a function: x.^2+cos(x) :3*x.^2+2*x+1
keyin sample size:300
keyin the number of hidden units:5



Example

>> fa1d
input a function: x.^2+cos(x) :cos(x)+sin(x)
keyin sample size:300
keyin the number of hidden units:



High-dimensional function approximation

- Given $(\mathbf{x}_i, y_i), i = 1,\ldots,n,$

  $\mathbf{x}_i \in R^d$

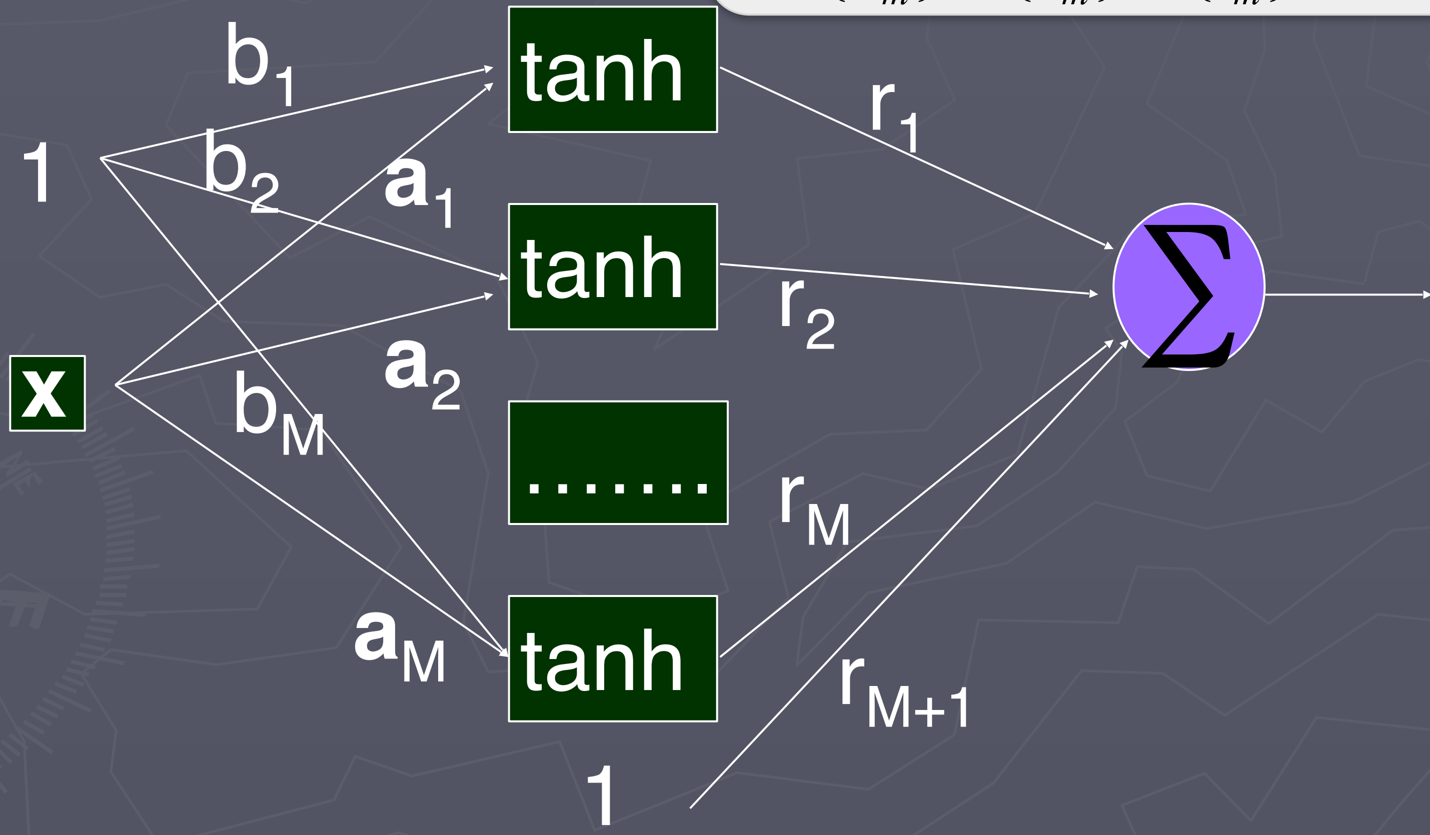
- Find a parametric function to minimize

$$E(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i;\theta))^2$$

Network

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

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



Example

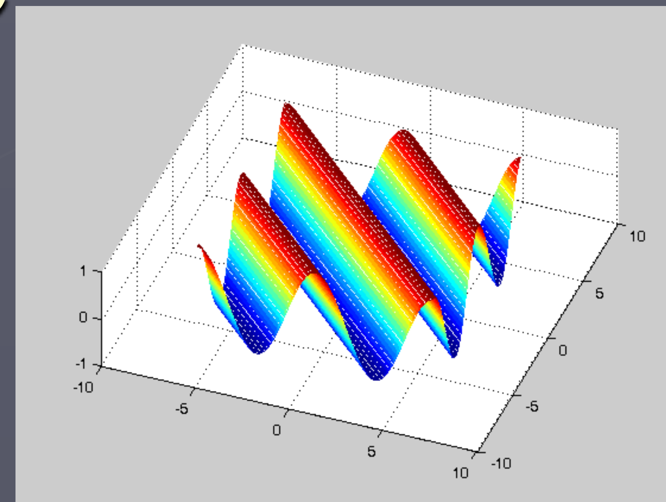
x

plot2d.m

y

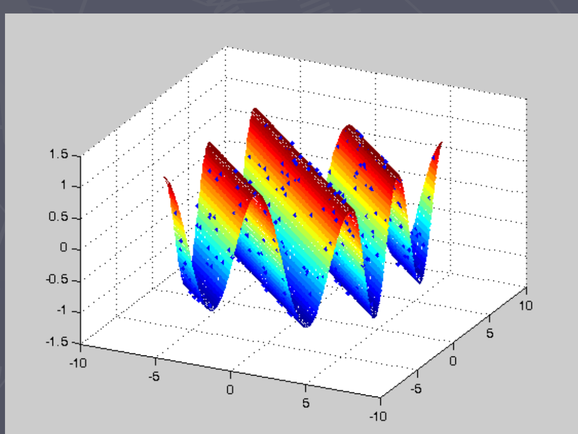
http://134.208.26.59/AdvancedNA/LMlearningNNSYSID/plot2d.m

observations



y=cos(x1+x2)

Create paired data
sampling2.m



DDFA

M=input('M:');

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

D-dimensional function approximation

source codes

http://134.208.26.59/AdvancedNA/LMlearningNNSYSID/fa2d.m

eval_MLP2.m

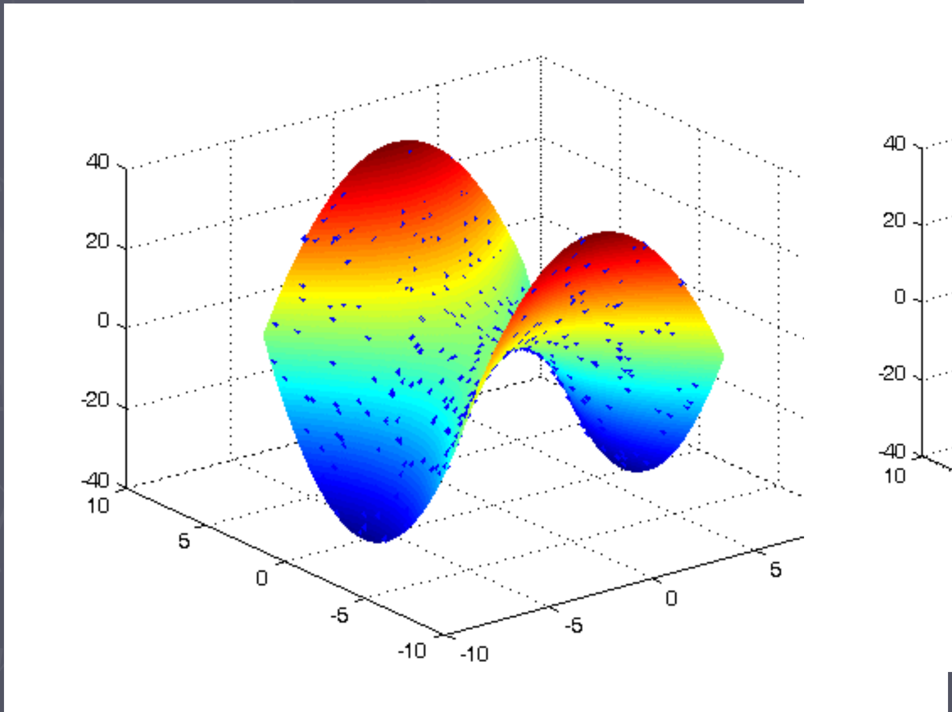
http://134.208.26.59/AdvancedNA/LMlearningNNSYSID/eval_MLP2.m

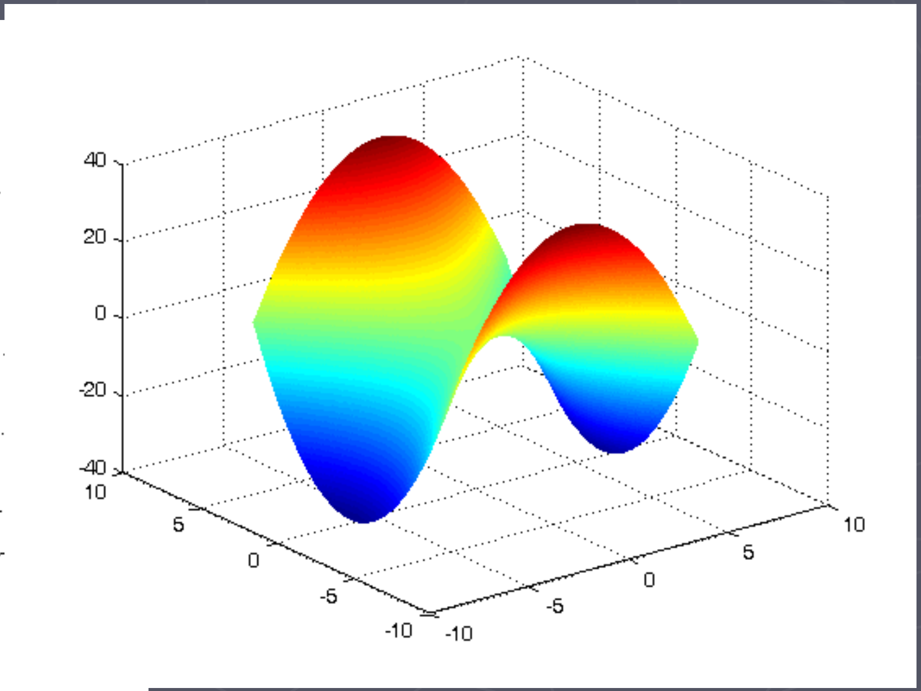
Example

```
>> fa2d
input a 2D function: x1.^2+x2.^2+cos(x1) :x1.^2-x2.^2
keyin sample size:300
keyin the number of hidden units:20
```

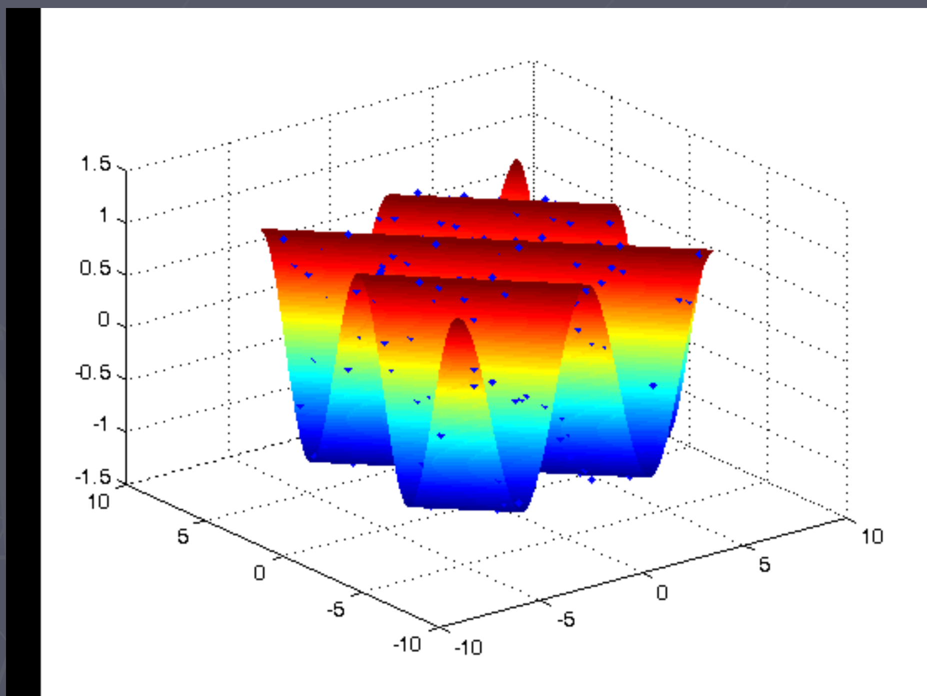
Numerical results

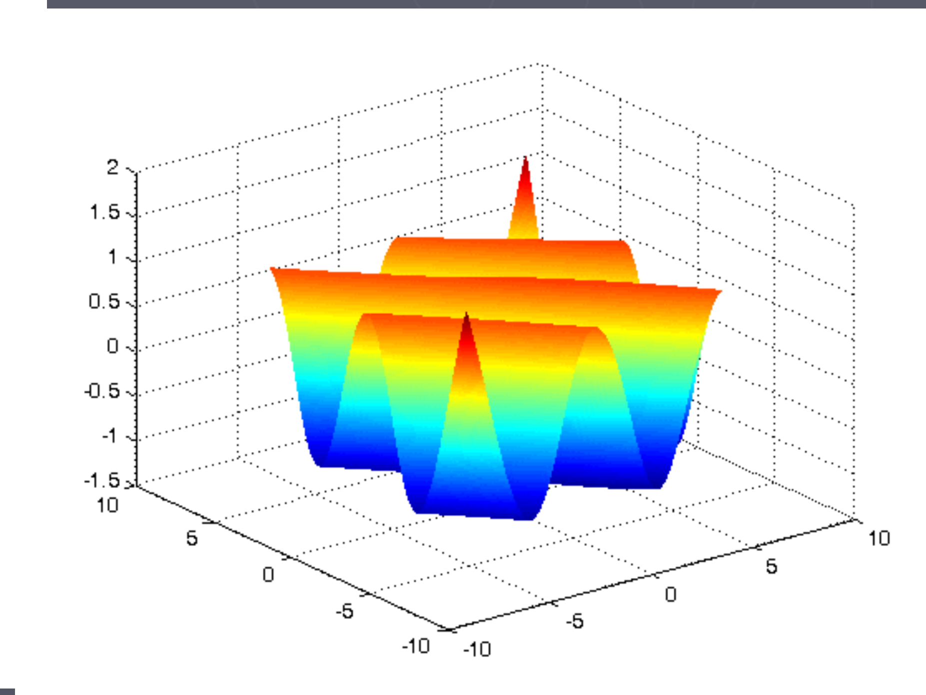
MSE for training data 0.008656
ME for training data 0.065931>>





Example


1.5
1
0.5
0
-0.5
-1
-1.5
10
5
0
-5
-10
-10
-5
0
5
10


2
1.5
1
0.5
0
-0.5
-1
-1.5
10
5
0
-5
-10
-10
-5
0
5
10