Numerical Analysis

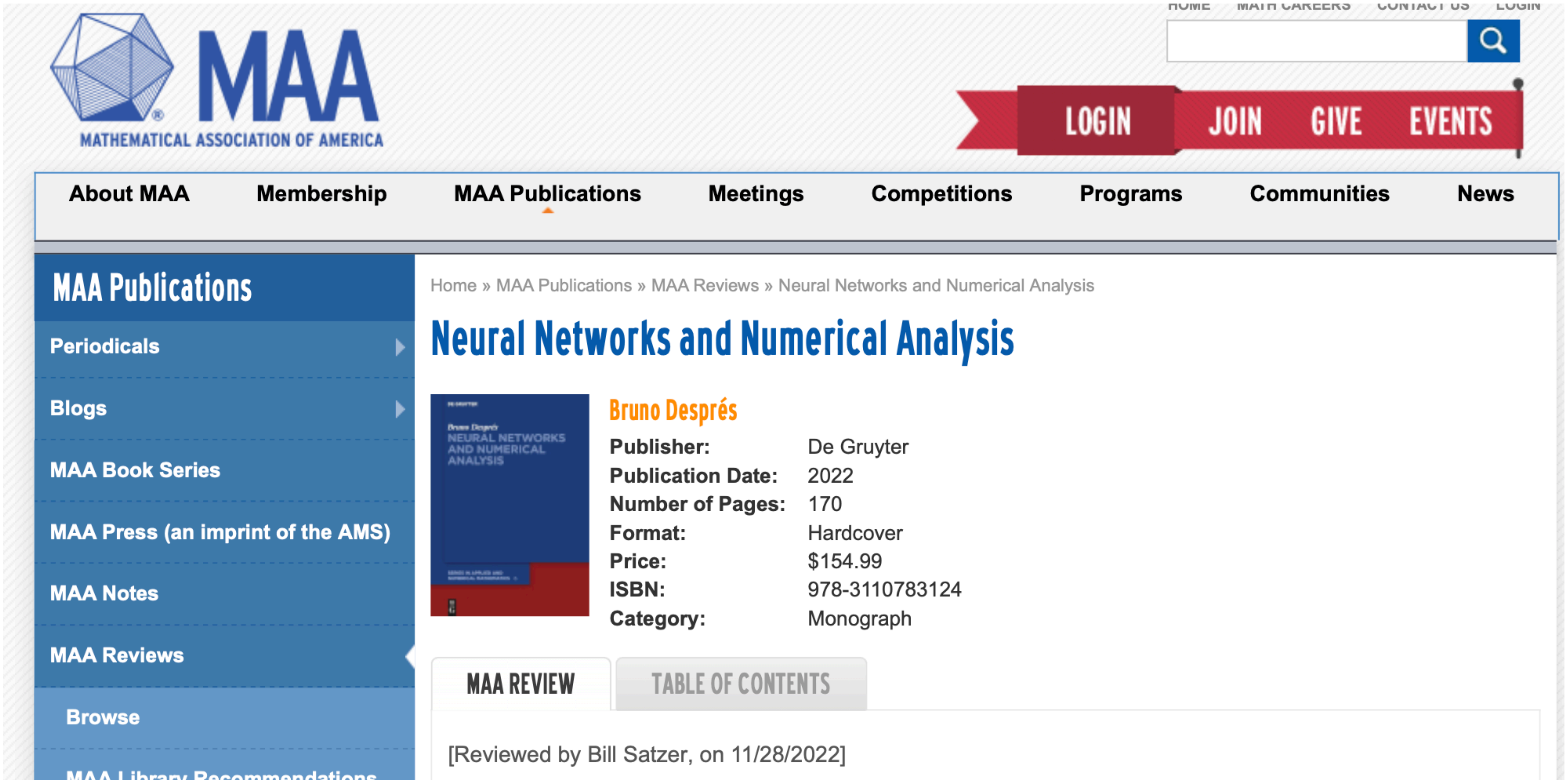
AM NDHU

Prof. Jiann-Ming Wu

Neural Networks and Numerical Analysis

Bruno Després

Neural Networks and Numerical Analysis



HOME MATH CAREERS CONTACT US LOGIN

MATHEMATICAL ASSOCIATION OF AMERICA

LOGIN JOIN GIVE EVENTS

About MAA | Membership | MAA Publications | Meetings | Competitions | Programs | Communities | News

MAA Publications

- Periodicals
- Blogs
- MAA Book Series
- MAA Press (an imprint of the AMS)
- MAA Notes
- MAA Reviews
  - Browse
- MAA Library Recommendations

Home » MAA Publications » MAA Reviews » Neural Networks and Numerical Analysis

Neural Networks and Numerical Analysis

Bruno Després

Publisher:	De Gruyter
Publication Date:	2022
Number of Pages:	170
Format:	Hardcover
Price:	$154.99
ISBN:	978-3110783124
Category:	Monograph

MAA REVIEW | TABLE OF CONTENTS

[Reviewed by Bill Satzer, on 11/28/2022]

[Reviewed by Bill Satzer, on 11/28/2022]

The emerging technologies that are known collectively as neural networks, machine learning and deep learning can offer an efficient means of modeling nonlinear functions. This can be done with very good accuracy using an approach with just input and output data even in high-dimensional spaces. Yet, from the view of applied mathematics, these nonlinear functions are effectively unknowns, arising as the do purely from data. But results using these new methods have sometimes been compelling, and have included, for example, new insights in computational fluid dynamics and analysis of turbulence.

At the same time the relationship between practitioners of these new technologies and more traditional numerical analysts has been distant, often with little communication between the two. Part of this is likely due to their distinctly different terminologies, and perhaps concerns about lack of rigor from the more traditional numerical analysts.

The aim of the author in this book is to provide a unified framework for the foundations of numerical analysis of neural networks using some basic mathematical and algorithmic principles. Another goal is to introduce some related modern software and explain a connection with numerical discretization of simple partial differential equations.

	linear	Nonlinear
Univariate	line	Calculus
Multivariate	Linear algebra	???
Discrete multivariate	Integer programming	???
Mixed discrete and continuous multivariate	?	???

Real life applications

scientific reports

OPEN

Intelligent computing through neural networks for numerical treatment of non-Newtonian wire coating analysis model

Jawaher Lafi Aljohani[1]✉, Eman Salem Alaidarous[1], Muhammad Asif Zahoor Raja[2], Muhammad Shoaib[3] & Muhammed Shabab Alhothuali[1]

In the current study, a modern implementation of intelligent numerical computational solver introduced using the Levenberg Marquardt algorithm based trained neural networks (LMA-TNN) to analyze the wire coating system (WCS) for the elastic-viscous non-Newtonian Eyring–Powell fluid (EPF) with the impacts of Joule heating, magnetic parameter and heat transfer scenarios in the permeable medium. The nonlinear PDEs describing the WCS-EPF are converted into dimensionless nonlinear ODEs containing the heat and viscosity parameters. The reference data for the designed LMA-TNN is produced for various scenarios of WCS-EPF representing with porosity parameter, non-Newtonian parameter, heat transfer parameter and magnetic parameter for the proposed analysis using the state of the art explicit Runge–Kutta technique. The training, validation, and testing operations of LMA-TNN are carried out to obtain the numerical solution of WCS-EPF for various cases and their comparison with the approximate outcomes certifying the reasonable accuracy and precision of LMA-TNN approach. The outcomes of LMA-TNN solver in terms of state transition (ST) index, error-histograms (EH) illustration, mean square error, and regression (R) studies further established the worth for stochastic numerical solution of the WCS-EPF. The strong correlation between the suggested and the reference outcomes indicates the structure's validity, for all four cases of WCS-EPF, fitting of the precision 10^{-5} to 10^{-9} is also accomplished.

Numerical Analysis of Artificial Neural Networks

Σ *mathematics*

Keywords

- Numerical linear algebra
- Numerical methods for dynamical systems
- Numerical optimization
- Geometric numerical integration
- Iterative methods
- Convergence
- Machine learning
- Neural networks
- Classification
- Time series forecasting
- Dimensionality reduction

Special Issue Information

Numerical analysis is one of the pillars, computer algebra being the other, of all computational algorithms. Accurate results of machine learning algorithms for $\boxed{\text{classification,}}$ $\boxed{\text{regression, and prediction}}$ are supported by theoretical features of numerical methods.

The list of examples is overwhelming: $\boxed{\text{principal component}}$ $\boxed{\text{analysis}}$ based upon numerical linear algebra; optimization with $\boxed{\text{Hopfield networks}}$ stemming from concepts rooted in dynamical systems; $\boxed{\text{backpropagation}}$ that requires numerical optimizers; etc. On the other hand, research on computational intelligence techniques has led to advances in many numerical methods, with $\boxed{\text{stochastic gradient descent}}$ being primus inter pares.

In this Special Issue, we aim at fostering the synergy between these two fields, by encouraging the analysis and design of numerical methods for, in, and from machine learning algorithms. We welcome contributions that highlight satisfactory learning results as soundly based on numerical foundations, as well as ground-breaking numerical methods that provide the basis for efficient practical algorithms, at least at the proof-of-concept stage.

The scope of the issue is deliberately broad, including but not limited to numerical techniques from linear algebra, dynamical systems, kernel methods, optimization, spectral methods, and stochastic formulations, as well as algorithms within neural networks, support vector machines, recurrent networks and clustering methods

Prof. Dr. Miguel Atencia
Guest Editor

mathematics

Article

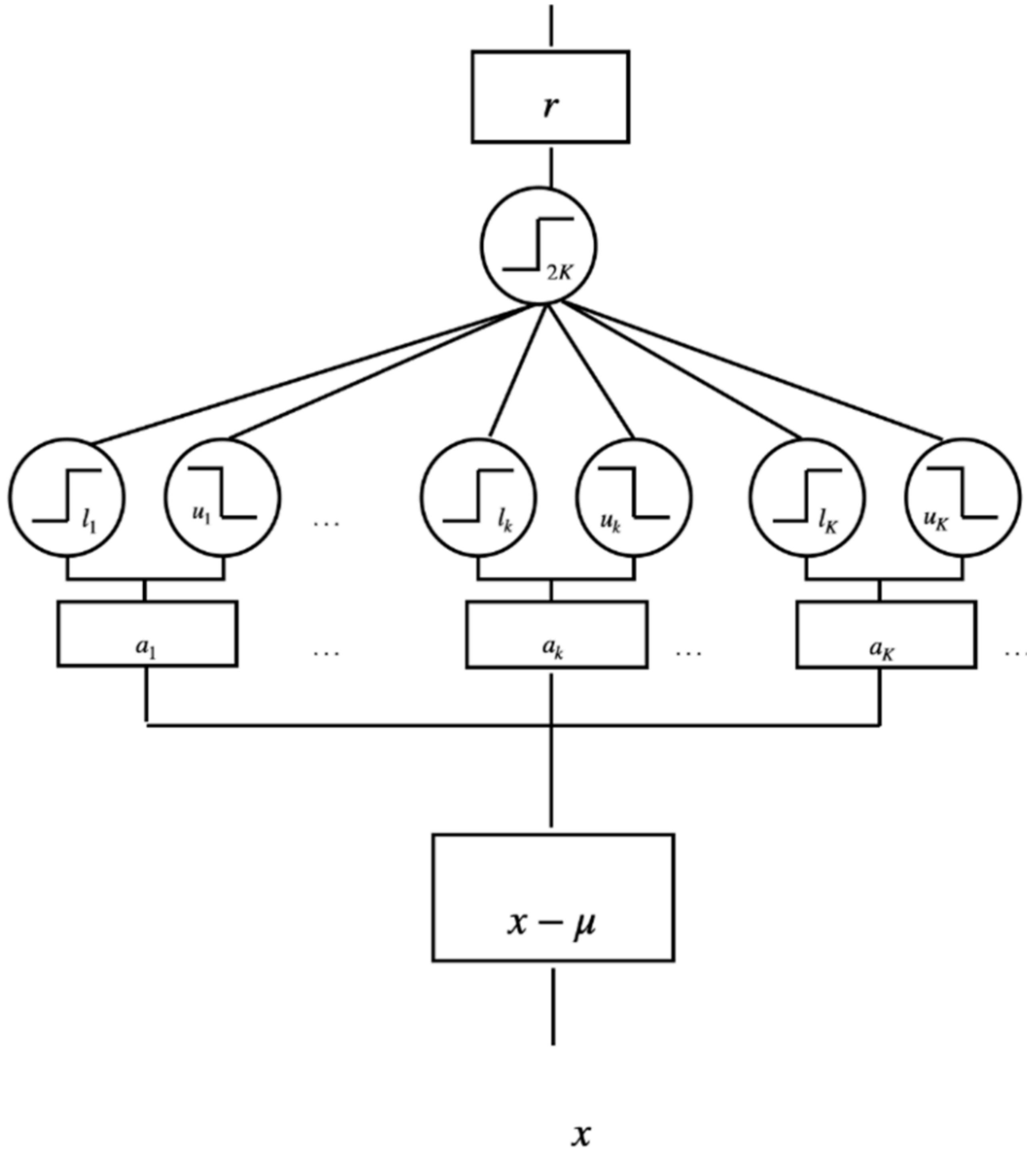
Learning Neural Representations and Local Embedding for Nonlinear Dimensionality Reduction Mapping

Nonlinear Dimensionality Reduction NDR Mapping

This work explores neural approximation for nonlinear dimensionality reduction mapping based on internal representations of graph-organized regular data supports. Given training observations are assumed as a sample from a high-dimensional space with an embedding low-dimensional manifold. An approximating function consisting of adaptable built-in parameters is optimized subject to given training observations by the proposed learning process, and verified for transformation of novel testing observations to images in the low-dimensional output space.

Optimized internal representations sketch graph-organized supports of distributed data clusters and their representative images in the output space. On the basis, the approximating function is able to operate for testing without reserving original massive training observations. The neural approximating model contains multiple modules. Each activates a non-zero output for mapping in response to an input inside its correspondent local support.

Graph-organized data supports have lateral interconnections for representing neighboring relations, inferring the minimal path between centroids of any two data supports, and proposing distance constraints for mapping all centroids to images in the output space. Following the distance-preserving principle, this work proposes Levenberg-Marquardt learning for optimizing images of centroids in the output space subject to given distance constraints, and further develops local embedding constraints for mapping during execution phase. Numerical simulations show the proposed neural approximation effective and reliable for nonlinear dimensionality reduction mapping.



$$\theta_l(h) = \begin{cases} 1, & \text{if } h \geq l \\ 0, & \text{otherwise} \end{cases}$$

$$\theta^u(h) = \begin{cases} 1, & \text{if } h \leq u \\ 0, & \text{otherwise} \end{cases}$$

Figure 1. A feedforward neural module for translation of a high-dimensional observation to an image in the output space.

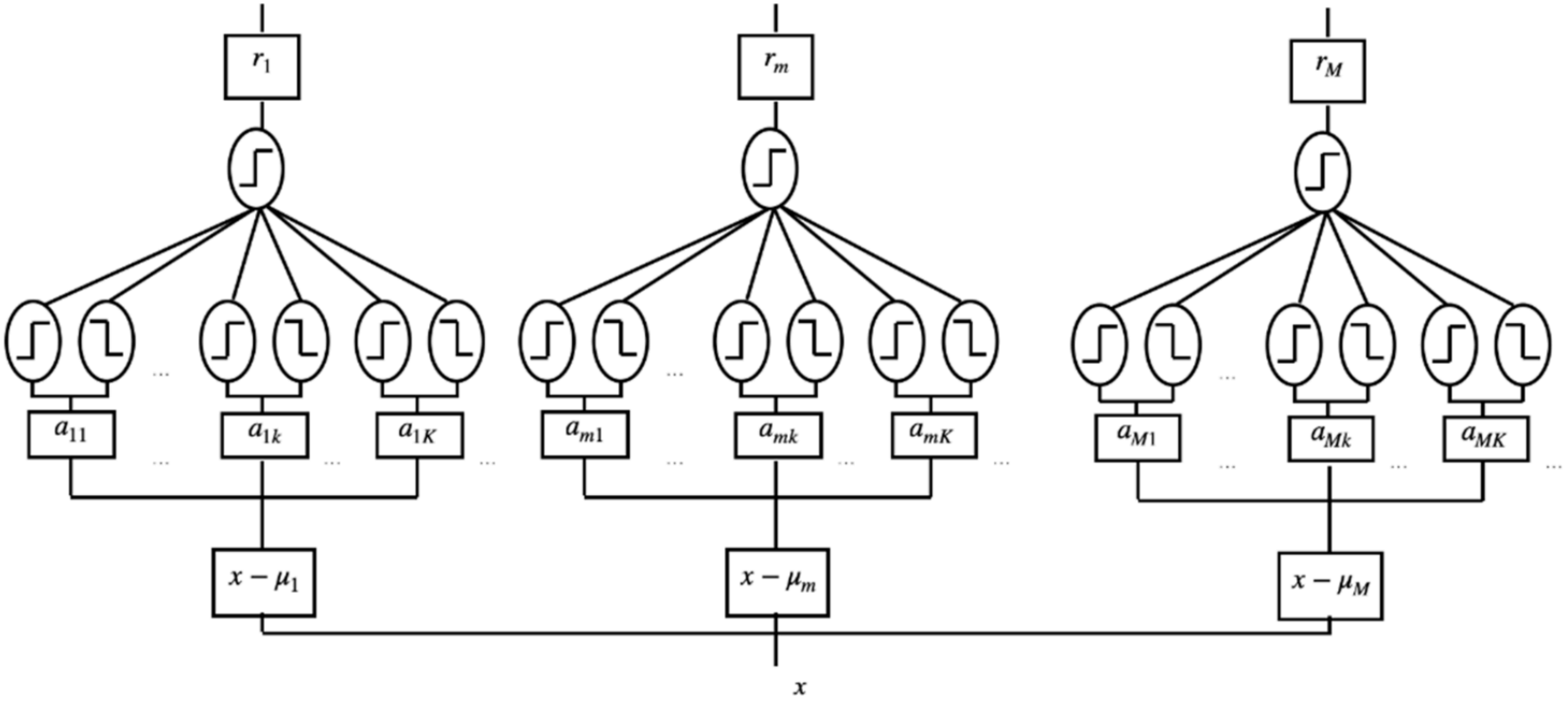


Figure 2. A deep neural network consisting of multiple neural modules for dimensionality reduction mapping.

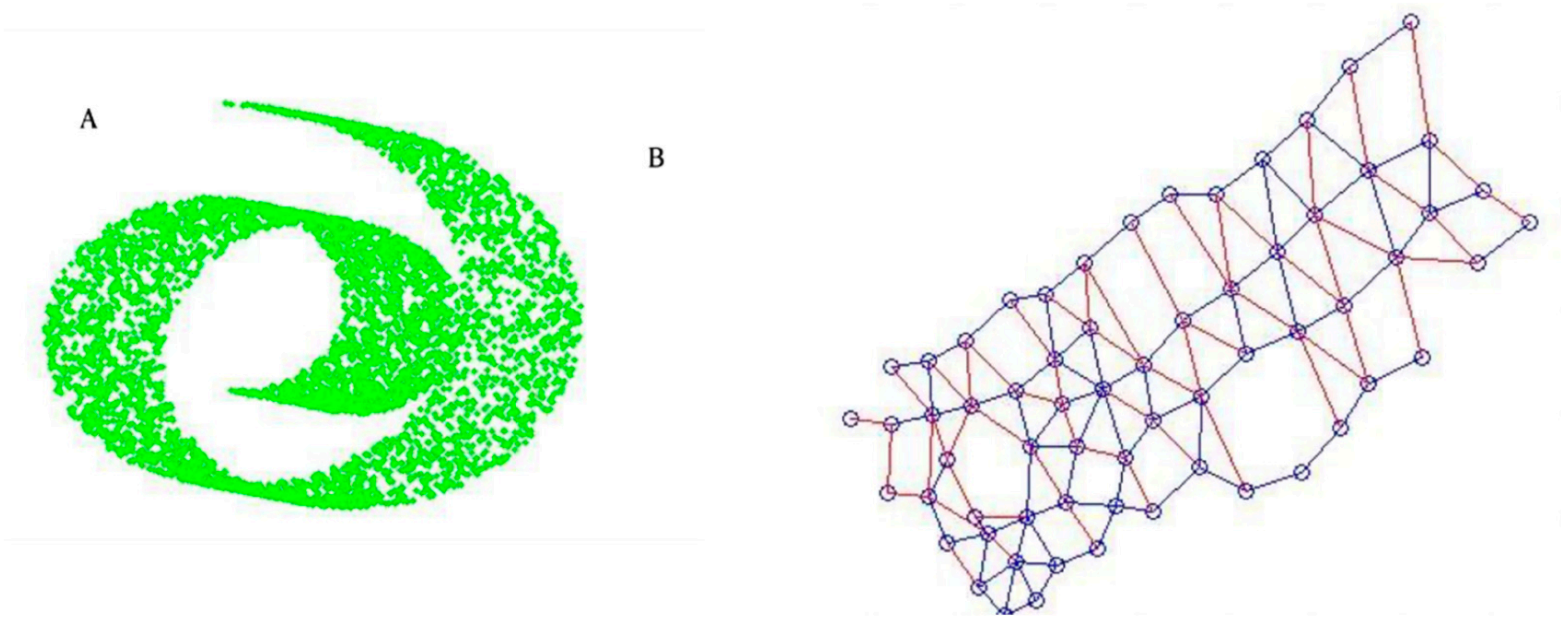


Figure 3. (**A**) Swiss-roll data. (**B**) A graph with derived edges for organizing neural modules.

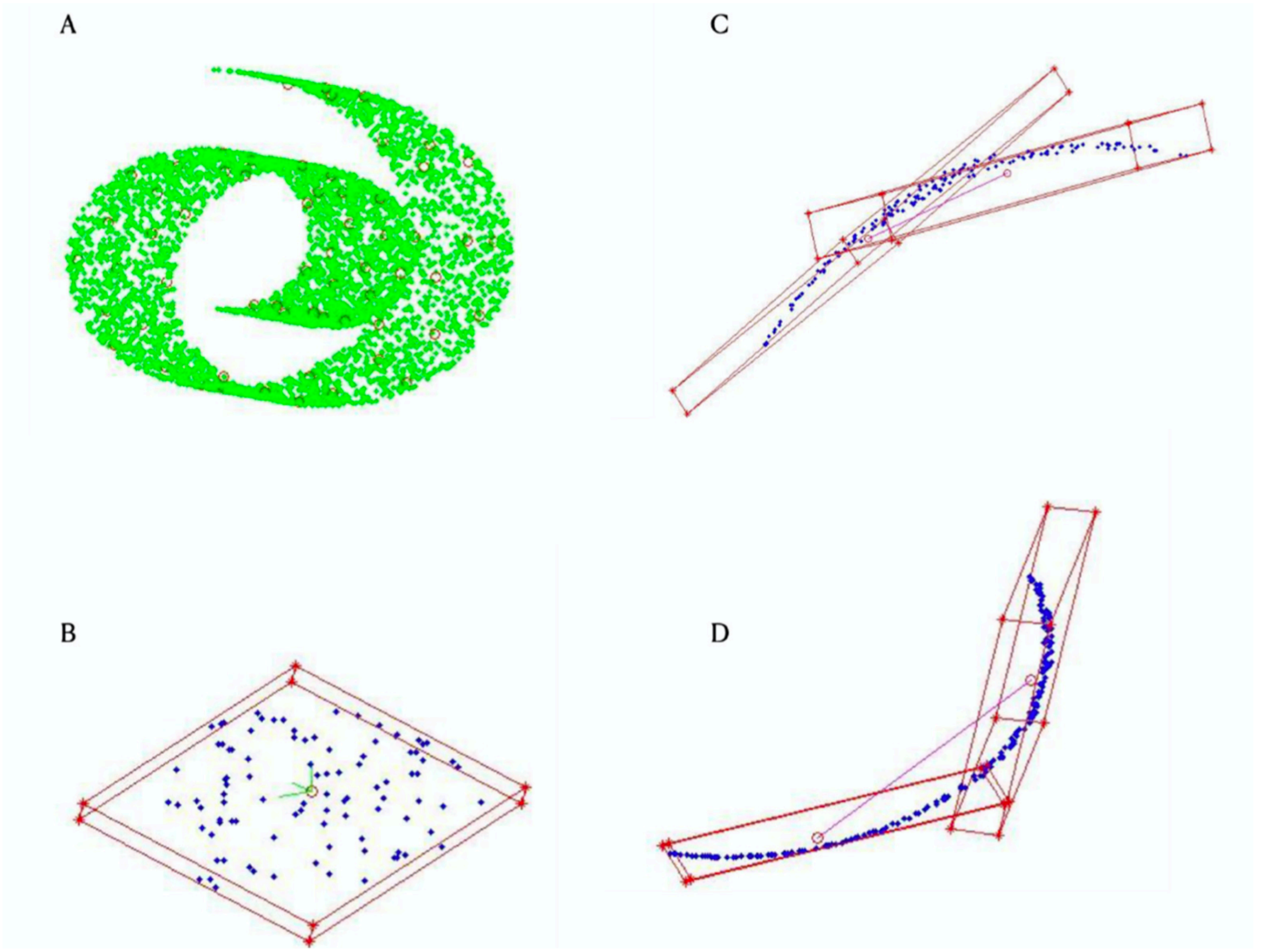


Figure 4. (**A**) Centroids of partitioned subsets. (**B**) A regular box for representing a local support. (**C**) The line segment that connects two centers totally belongs to union of two local supports. (**D**) The line segment that connects two centers partially belonged to union of two local supports.

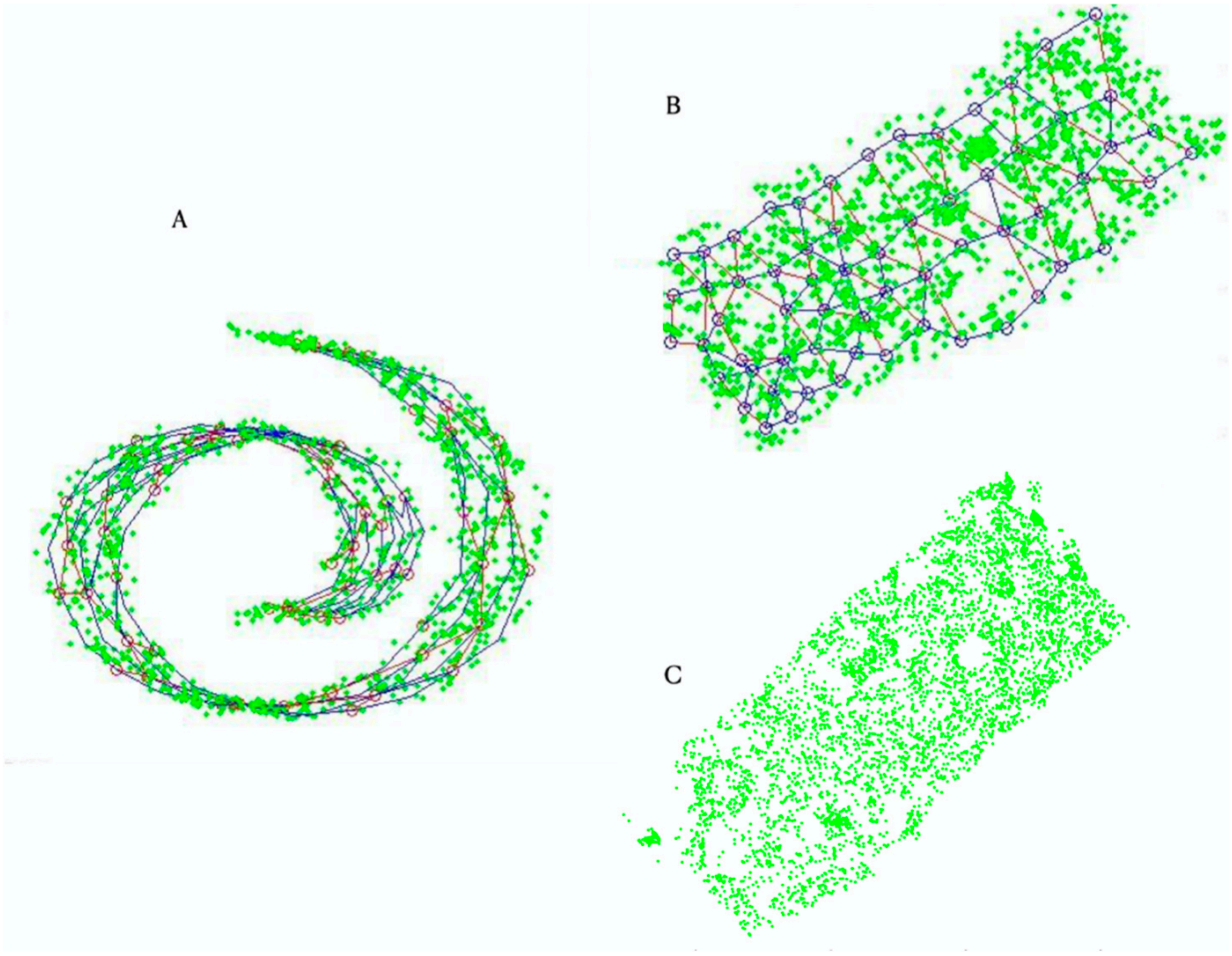


Figure 6. (**A**) Edges on a graph and neighboring relations of cluster supports in the input space. (**B**) Images of centroids and training observations. (**C**) Images of all observations in the output space.

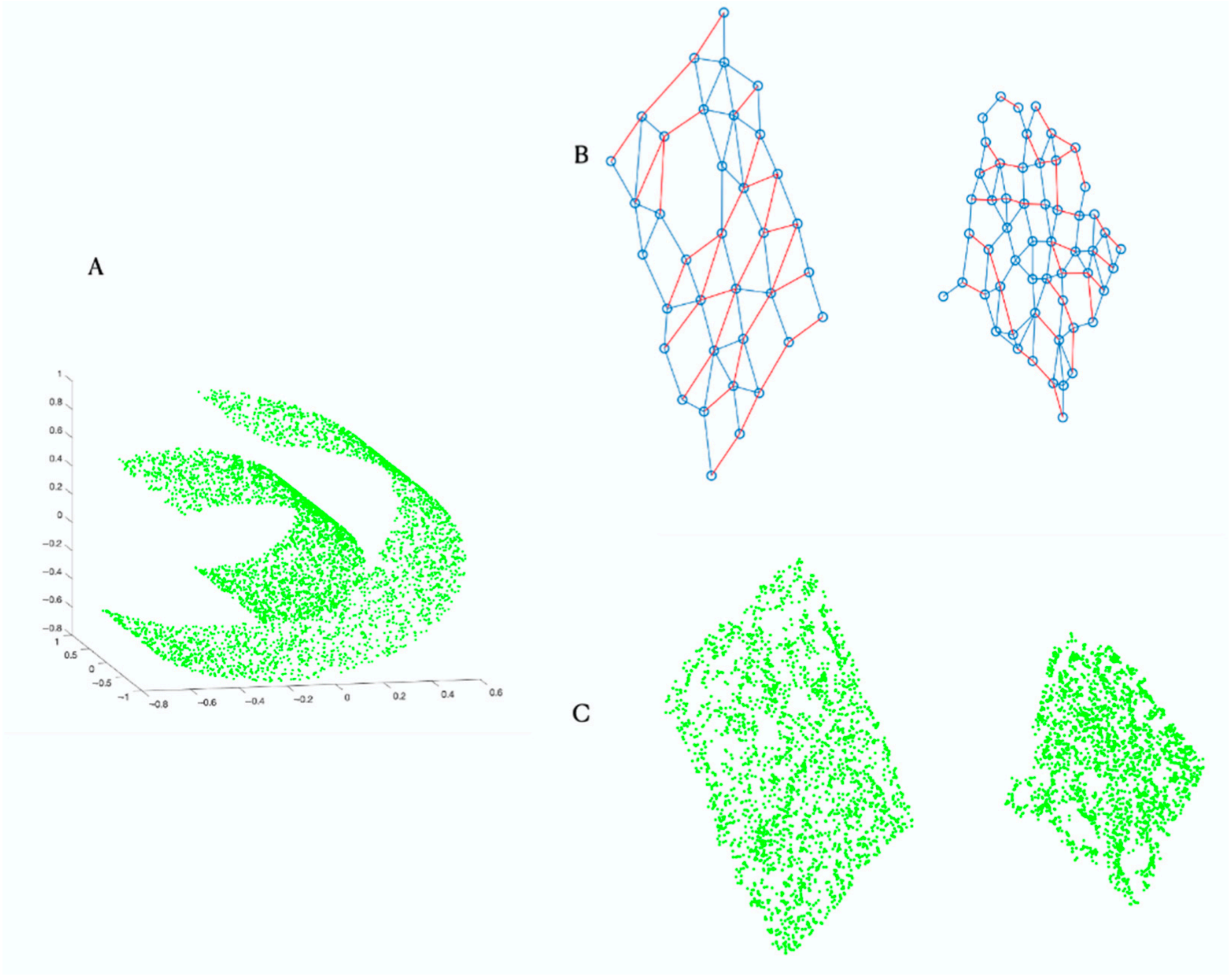


Figure 7. (**A**) A broken Swiss dataset for testing. (**B**) Isolated images of centroids and their edges. (**C**) Images of testing observations.

Reduction 非線性高維度降維

真實世界影像

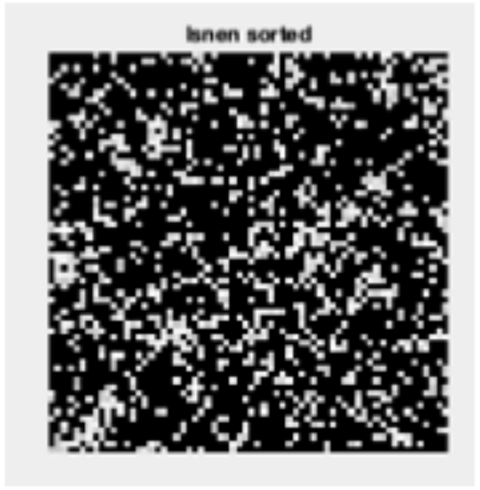
$224 \times 224 \times 3$

feature embedding 特徵內砍

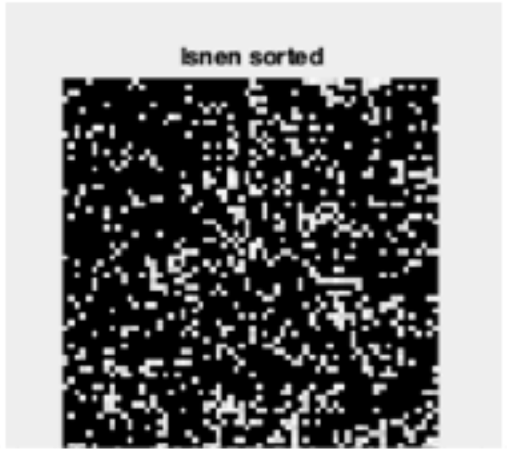
Wu, 2024

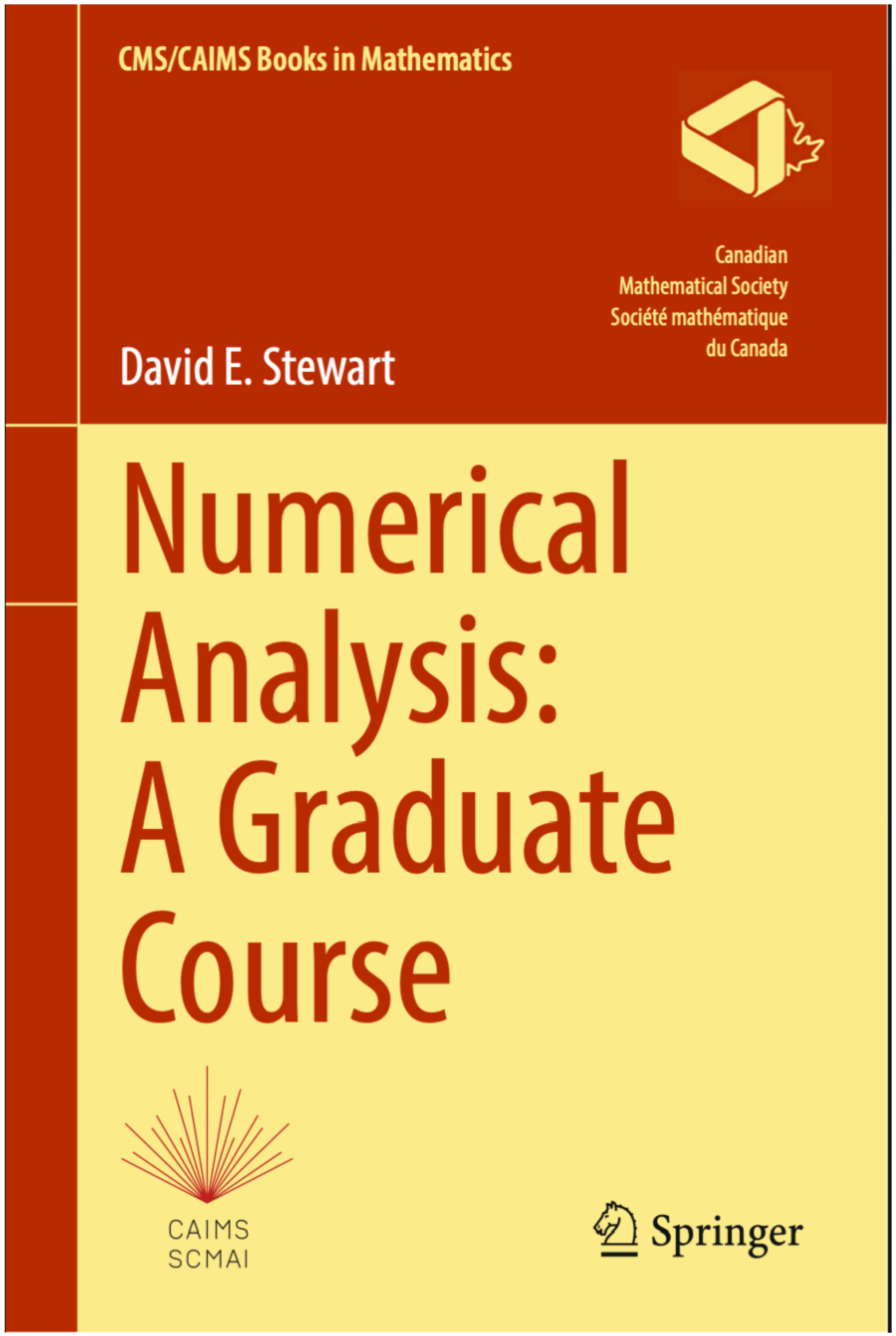
$64 \times 64 \times 1$

$\longrightarrow$



$\longrightarrow$





CMS/CAIMS Books in Mathematics

Canadian Mathematical Society
Société mathématique du Canada

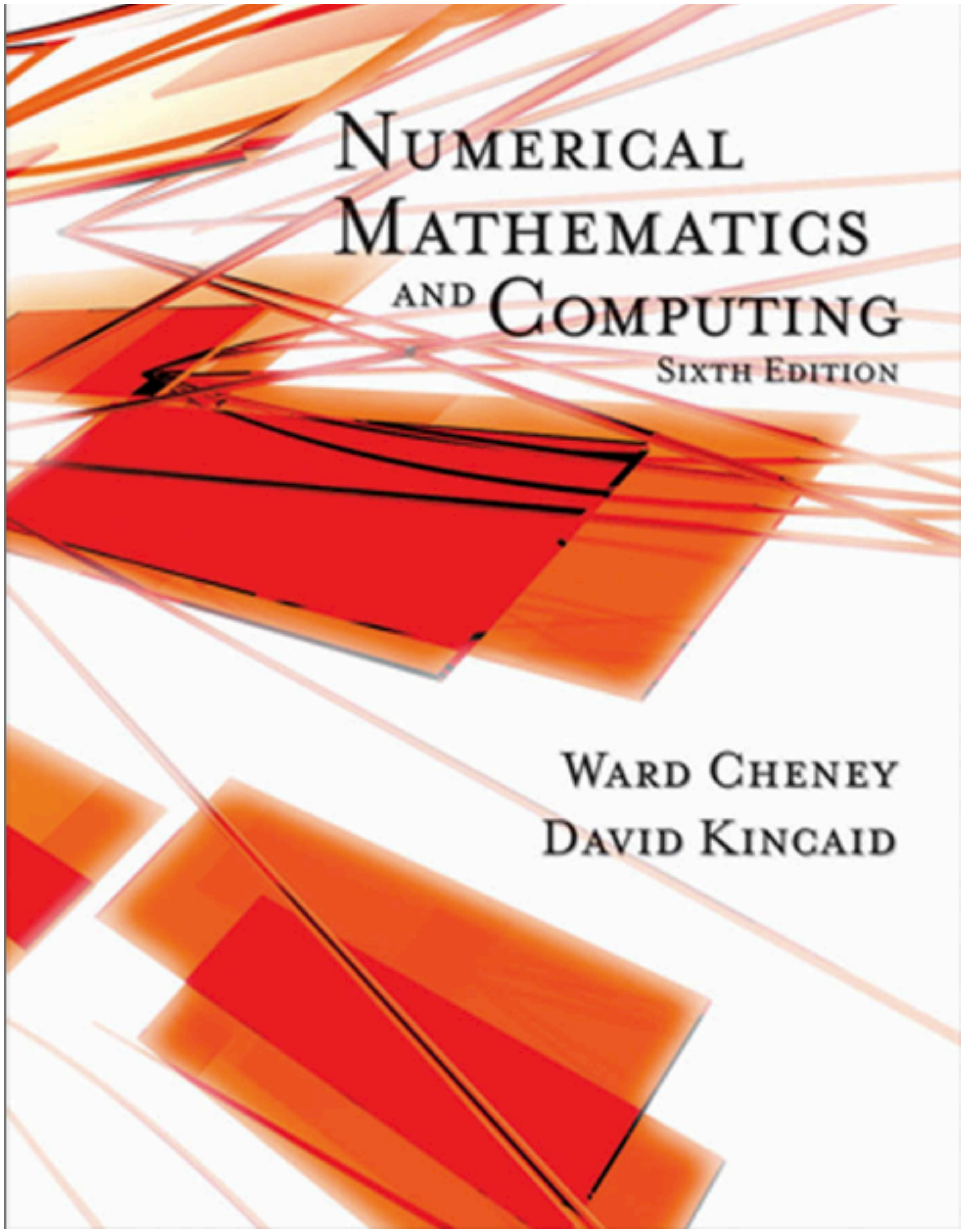
David E. Stewart

Numerical Analysis: A Graduate Course

CAIMS
SCMAI

Springer

NDHU Library EBook



NUMERICAL MATHEMATICS AND COMPUTING

SIXTH EDITION

WARD CHENEY

DAVID KINCAID

S.R. Otto and J.P. Denier

An Introduction to Programming and Numerical Methods in MATLAB

Numerical Analysis and Scientific Computing

David Ung

ELSEVIER

Neurocomputing 34 (2000) 55–77

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Potts models with two sets of interactive dynamics

Jiann-Ming Wu*

Department of Applied Mathematics, National Donghwa University, Hualien, Taiwan, ROC

Received 27 August 1997; accepted 12 April 2000

$$\boldsymbol{u}_i = \sum_j R_{ij}\boldsymbol{v}_j + \langle \boldsymbol{h}_i^{\sigma} \rangle, \tag{41}$$

$$\boldsymbol{v}_i = \sum_{\alpha} \frac{\exp(\beta_1 u_{i\alpha})}{\sum_{\gamma}\exp(\beta_1 w_{i\gamma})}\boldsymbol{e}_{\alpha}, \tag{42}$$

$$\boldsymbol{w}_{\alpha} = \sum_j R_{ij}\boldsymbol{p}_j + \langle \boldsymbol{h}_{\alpha}^{q} \rangle, \tag{43}$$

$$\boldsymbol{p}_{\alpha} = \sum_i \frac{\exp(\beta_2 w_{\alpha i})}{\sum_j \exp(\beta_2 w_{\alpha j})}\boldsymbol{e}_i, \tag{44}$$

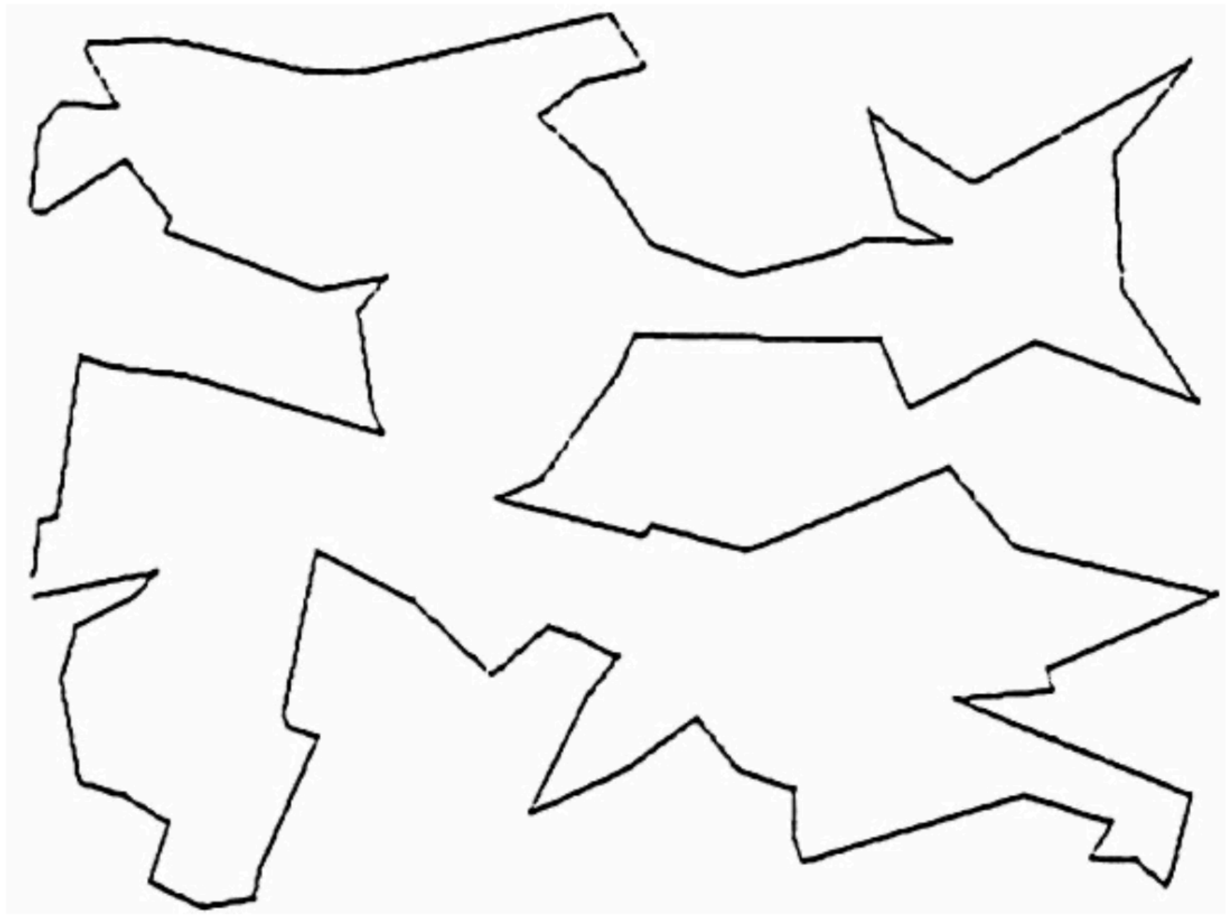


Fig. 1. The best solution obtained by the HAPER network for the 100-city TSP.

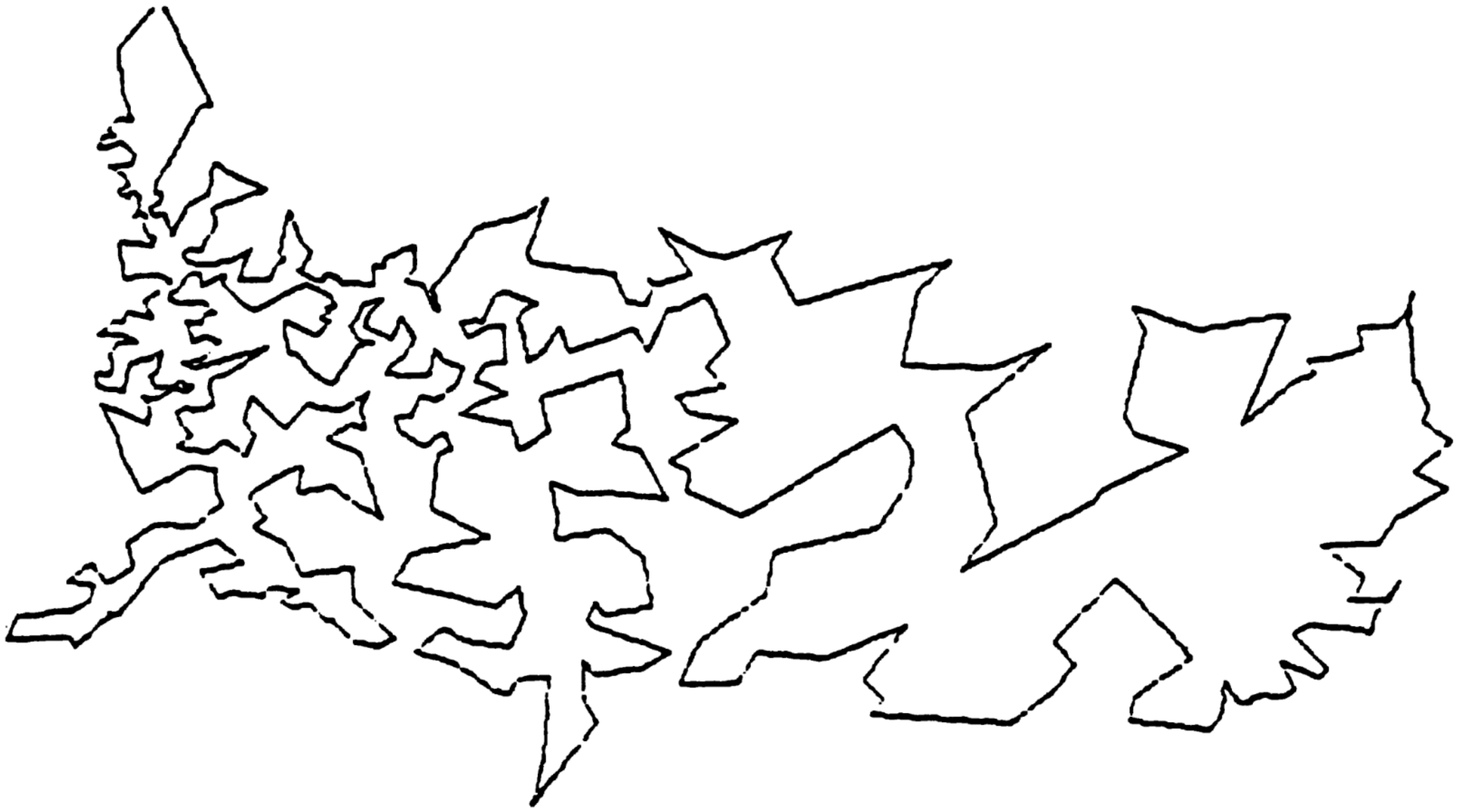
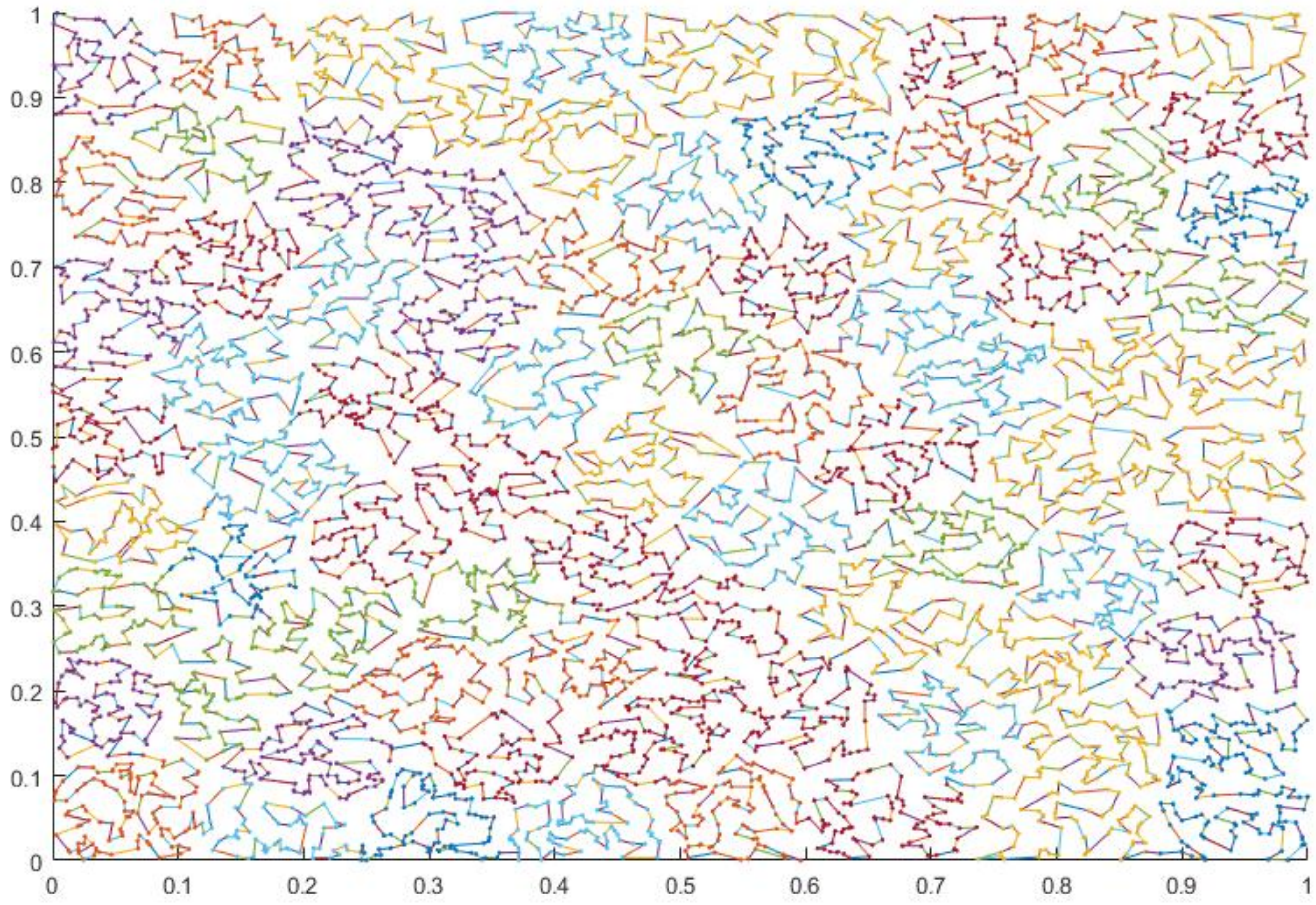


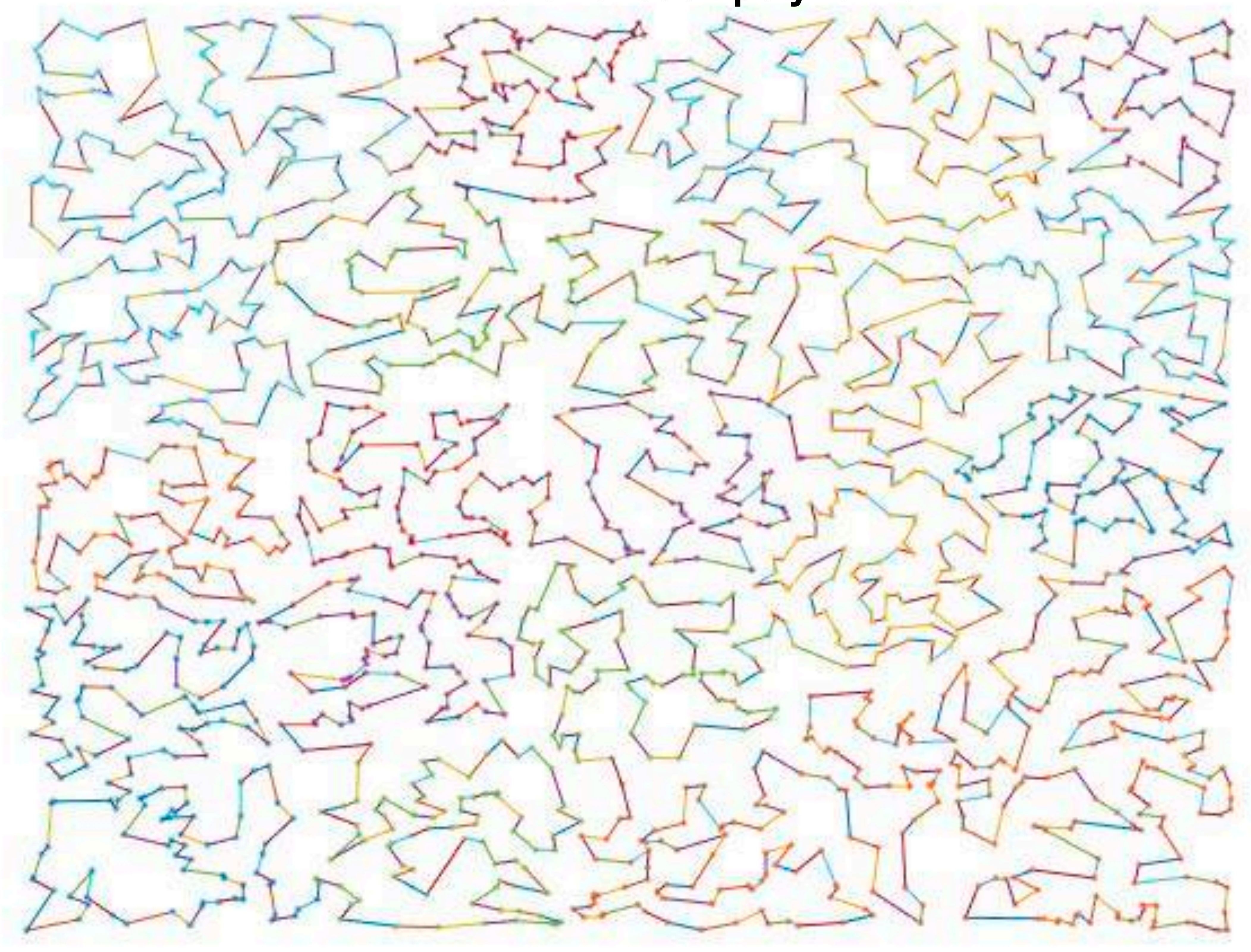
Fig. 2. A solution obtained by the HAPER network for the 532-city TSP.

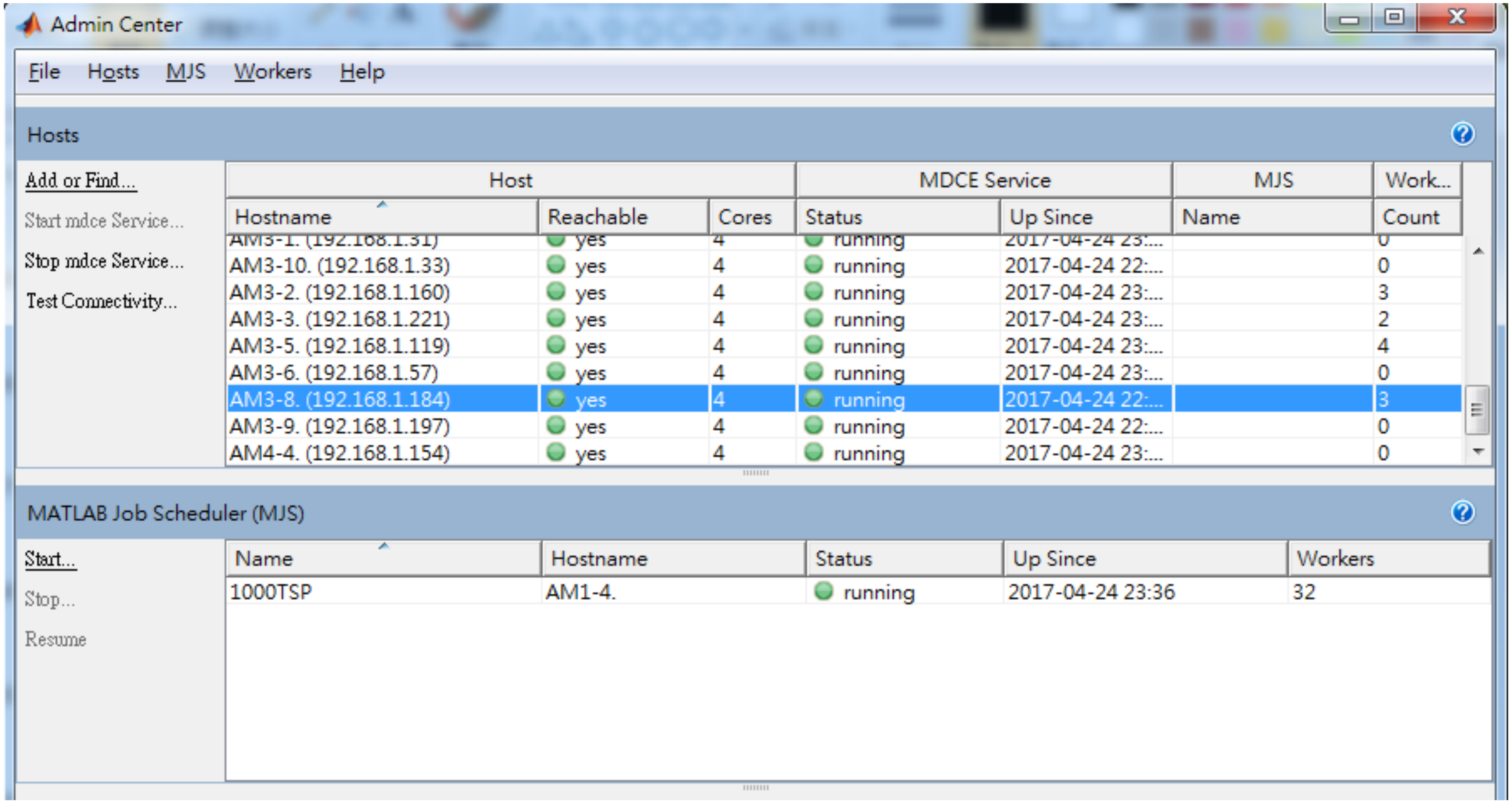


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

T(N=2000) = polynomial of N
Not exist such polynomial

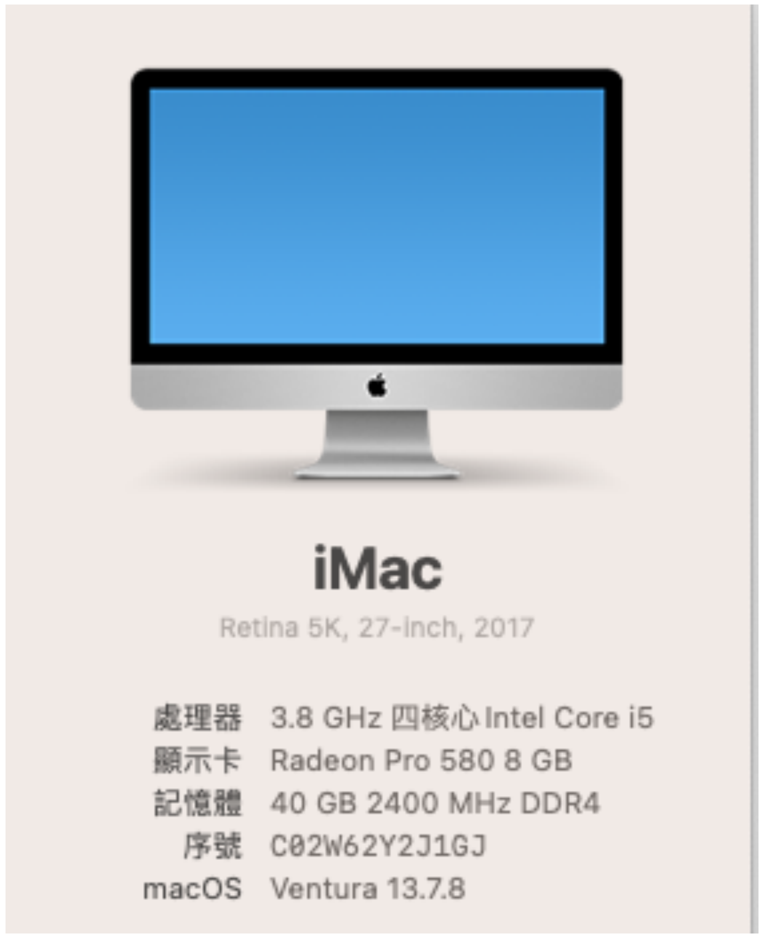
TSP
Bipartite





Host			MDCE Service		MJS	Work...
Hostname	Reachable	Cores	Status	Up Since	Name	Count
AM3-1. (192.168.1.31)	● yes	4	● running	2017-04-24 23:...		0
AM3-10. (192.168.1.33)	● yes	4	● running	2017-04-24 22:...		0
AM3-2. (192.168.1.160)	● yes	4	● running	2017-04-24 23:...		3
AM3-3. (192.168.1.221)	● yes	4	● running	2017-04-24 23:...		2
AM3-5. (192.168.1.119)	● yes	4	● running	2017-04-24 23:...		4
AM3-6. (192.168.1.57)	● yes	4	● running	2017-04-24 23:...		0
AM3-8. (192.168.1.184)	● yes	4	● running	2017-04-24 22:...		3
AM3-9. (192.168.1.197)	● yes	4	● running	2017-04-24 22:...		0
AM4-4. (192.168.1.154)	● yes	4	● running	2017-04-24 23:...		0

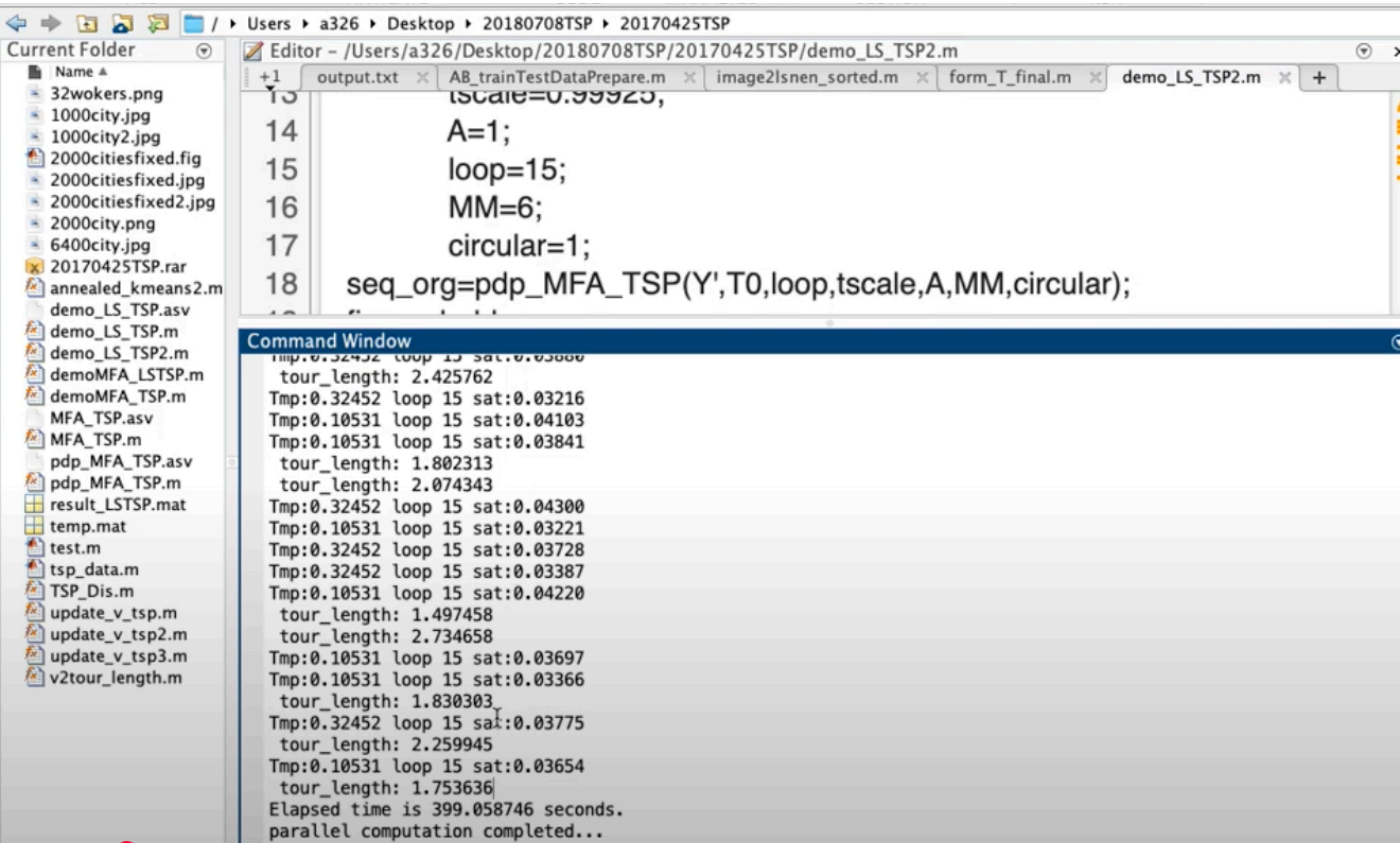
Name	Hostname	Status	Up Since	Workers
1000TSP	AM1-4.	● running	2017-04-24 23:36	32



iMac

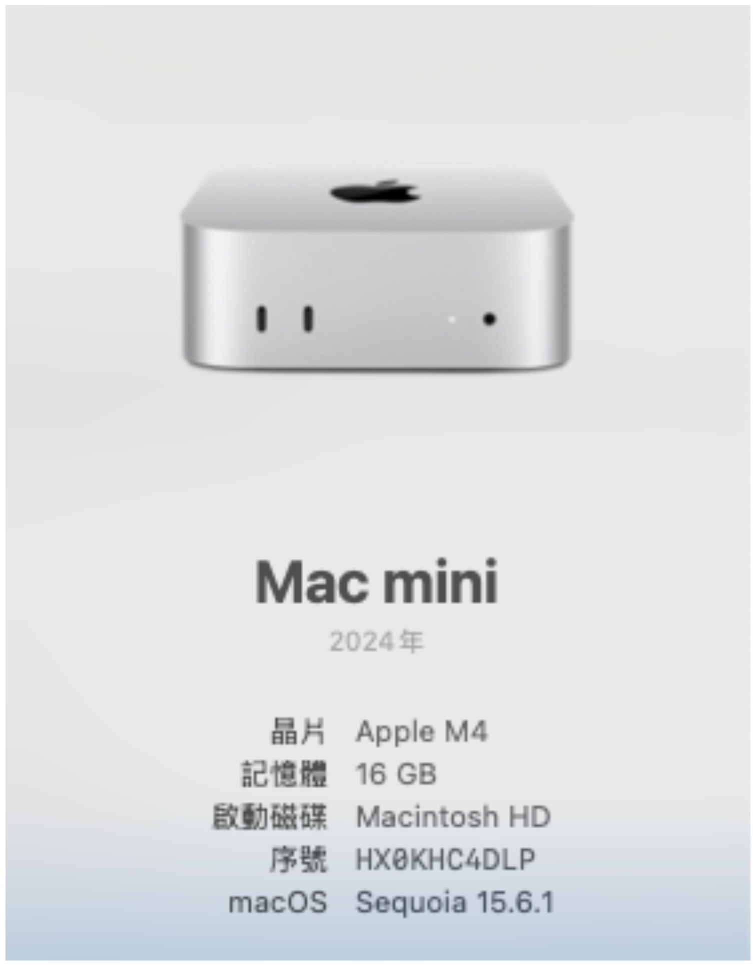
Retina 5K, 27-inch, 2017

處理器	3.8 GHz 四核心 Intel Core i5
顯示卡	Radeon Pro 580 8 GB
記憶體	40 GB 2400 MHz DDR4
序號	C02W62Y2J1GJ
macOS	Ventura 13.7.8



```
          A=1;
          loop=15;
          MM=6;
          circular=1;
seq_org=pdp_MFA_TSP(Y',T0,loop,tscale,A,MM,circular);
```

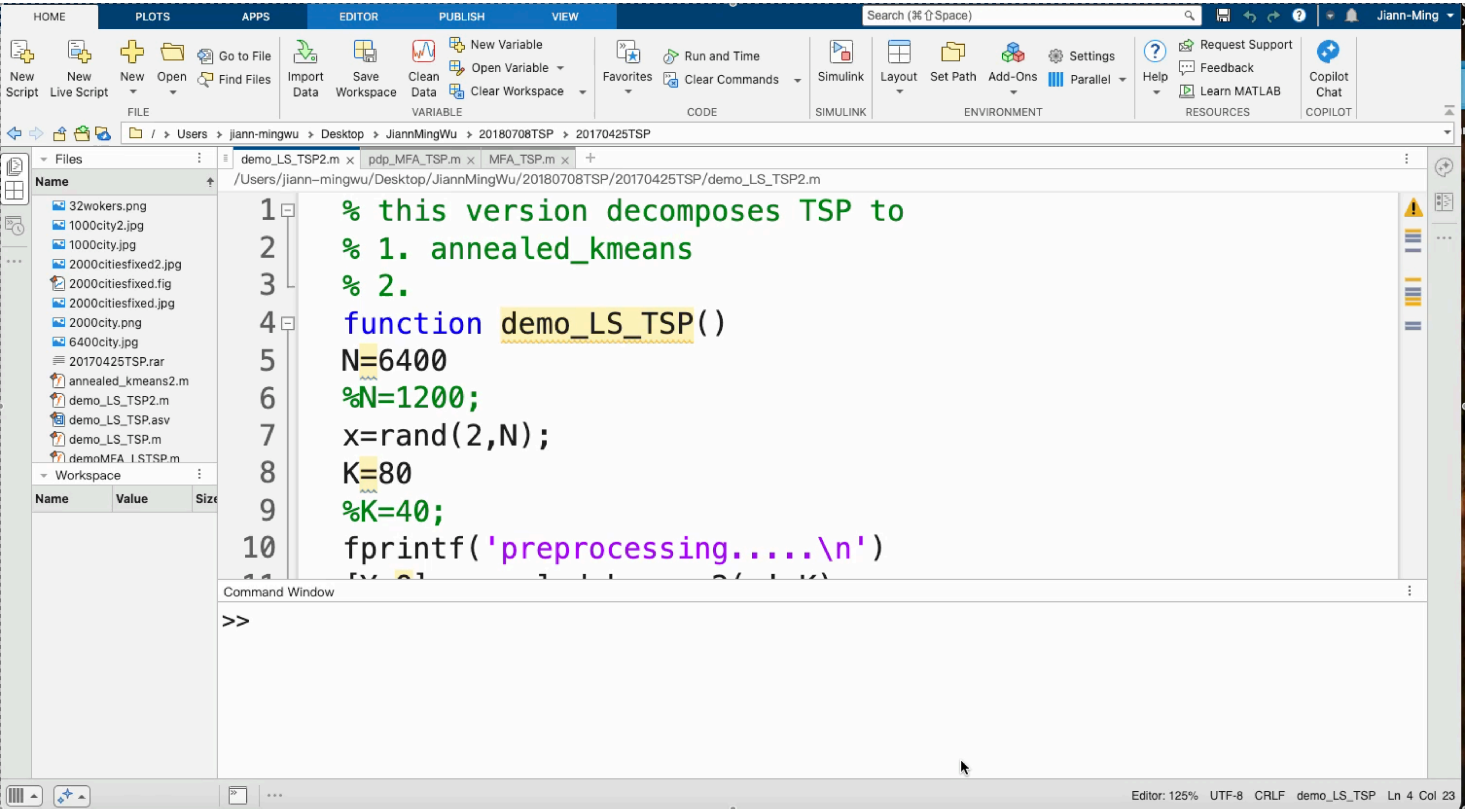
```
 tour_length: 2.425762
Tmp:0.32452 loop 15 sat:0.03216
Tmp:0.10531 loop 15 sat:0.04103
Tmp:0.10531 loop 15 sat:0.03841
 tour_length: 1.802313
 tour_length: 2.074343
Tmp:0.32452 loop 15 sat:0.04300
Tmp:0.10531 loop 15 sat:0.03221
Tmp:0.32452 loop 15 sat:0.03728
Tmp:0.32452 loop 15 sat:0.03387
Tmp:0.10531 loop 15 sat:0.04220
 tour_length: 1.497458
 tour_length: 2.734658
Tmp:0.10531 loop 15 sat:0.03697
Tmp:0.10531 loop 15 sat:0.03366
 tour_length: 1.830303
Tmp:0.32452 loop 15 sat:0.03775
 tour_length: 2.259945
Tmp:0.10531 loop 15 sat:0.03654
 tour_length: 1.753636
Elapsed time is 399.058746 seconds.
parallel computation completed...
```



Mac mini

2024年

晶片	Apple M4
記憶體	16 GB
啟動磁碟	Macintosh HD
序號	HX0KHC4DLP
macOS	Sequoia 15.6.1



/Users/jiann-mingwu/Desktop/JiannMingWu/20180708TSP/20170425TSP/demo_LS_TSP2.m

```
% this version decomposes TSP to
% 1. annealed_kmeans
% 2.
function demo_LS_TSP()
N=6400
%N=1200;
x=rand(2,N);
K=80
%K=40;
fprintf('preprocessing.....\n')
```

探索全新科技的未來，透過我們的GPF技術，提升學習與交流的無障礙體驗，讓每個人都能輕鬆掌握知識，無論年齡與背景。

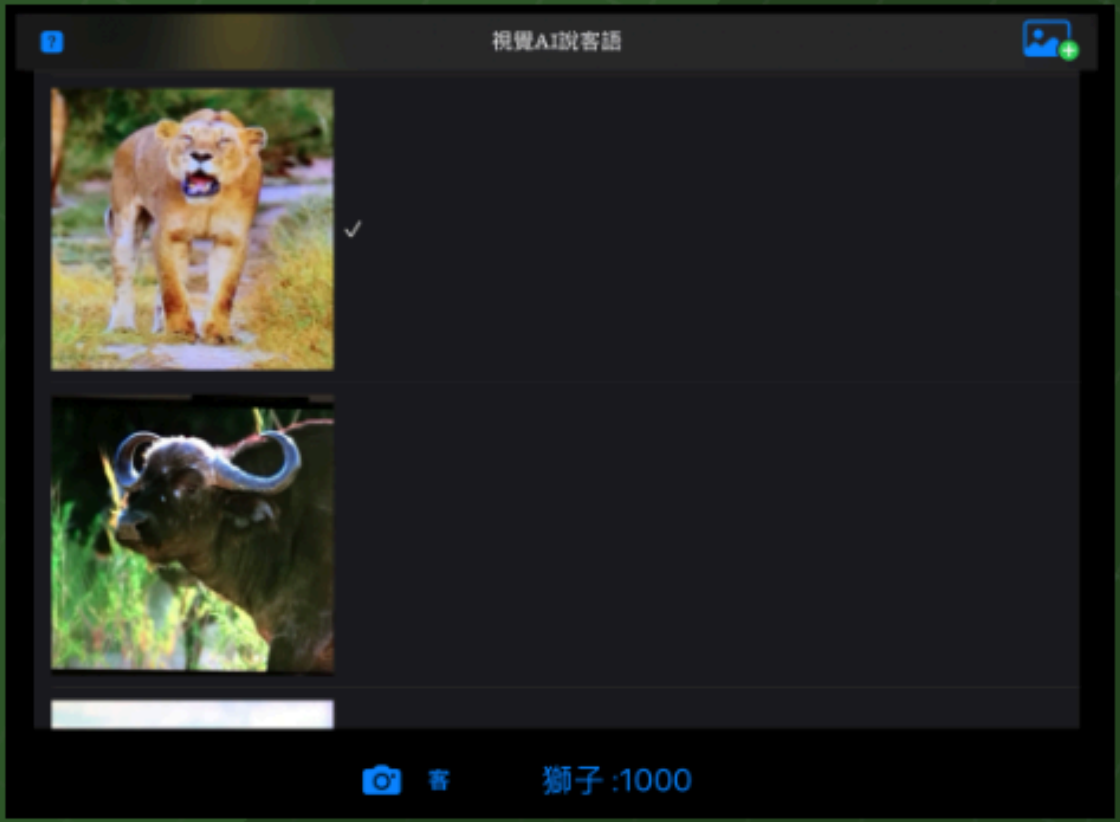
GPF
視覺AI說台語

Home About Us Contact Us

EXPLORE FEATURES OF GPF-VISION-AI

探索新一代的生成預訓練濾波器技術

這款應用程式利用最新的GPF卷積神經網絡技術，幫助用戶即時識別物件影像，並提供精準的台語和客語語音翻譯，讓學習台語客語變得更輕鬆。



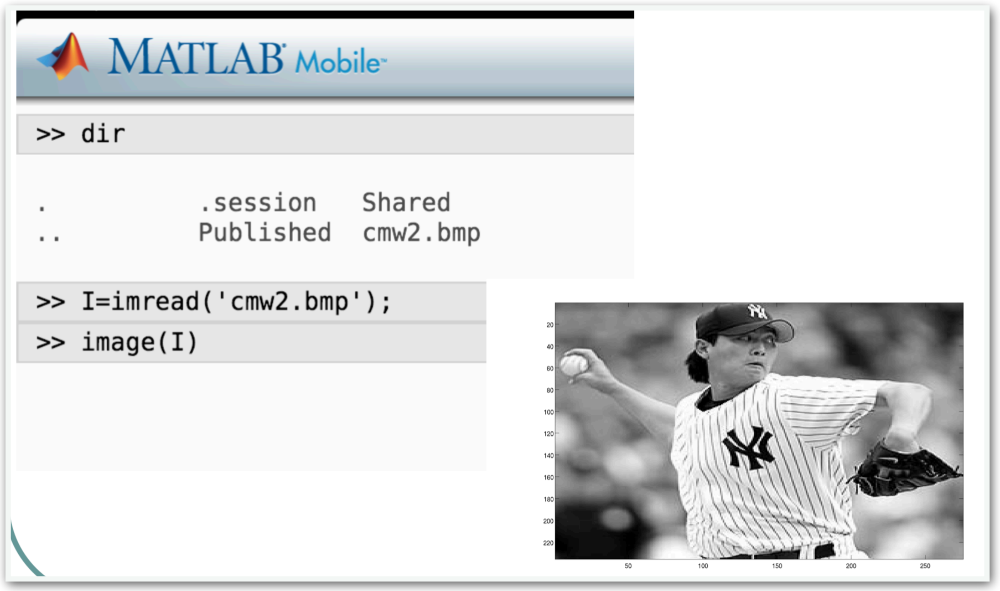
視覺AI說台語客語測試報名

體驗最先進的技術，提升您的學習效率，讓語言溝通無障礙。

數值方法（本系大三）

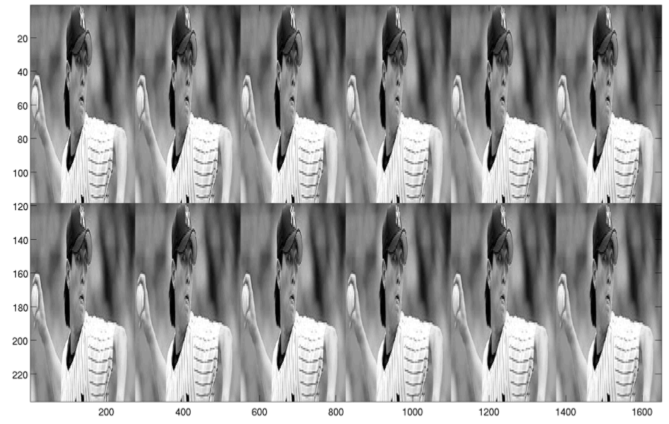
- matrix
- Operator-Data-TYpes-and-Conditional-Statements
- Binary Search & Looping Matlab matrix 2
- Taylor Series Expansion & Newton Method Matlab control
- Newton Method fsolve
- 以fsolve解非線性方程式，並應用在三度空間不共面四點的球心計算
- Numerical Integration Matlab Integration
- 矩陣的列運算與REF建構
- 二次曲面匹配 Matlab–control–II
- 以Richardson外差求數值微分 Taylor展開
- 數值微分在牛頓法求根與Jacogian計算應用 youtube上課影片1 youtube上課影片2 youtube上課影片3
- 追蹤渾沌微分方程: Euler and RK4 youtube上課影片1 youtube上課影片2
- Sudoku Solving and Integer Programming youtube上課影片1 youtube上課影片2

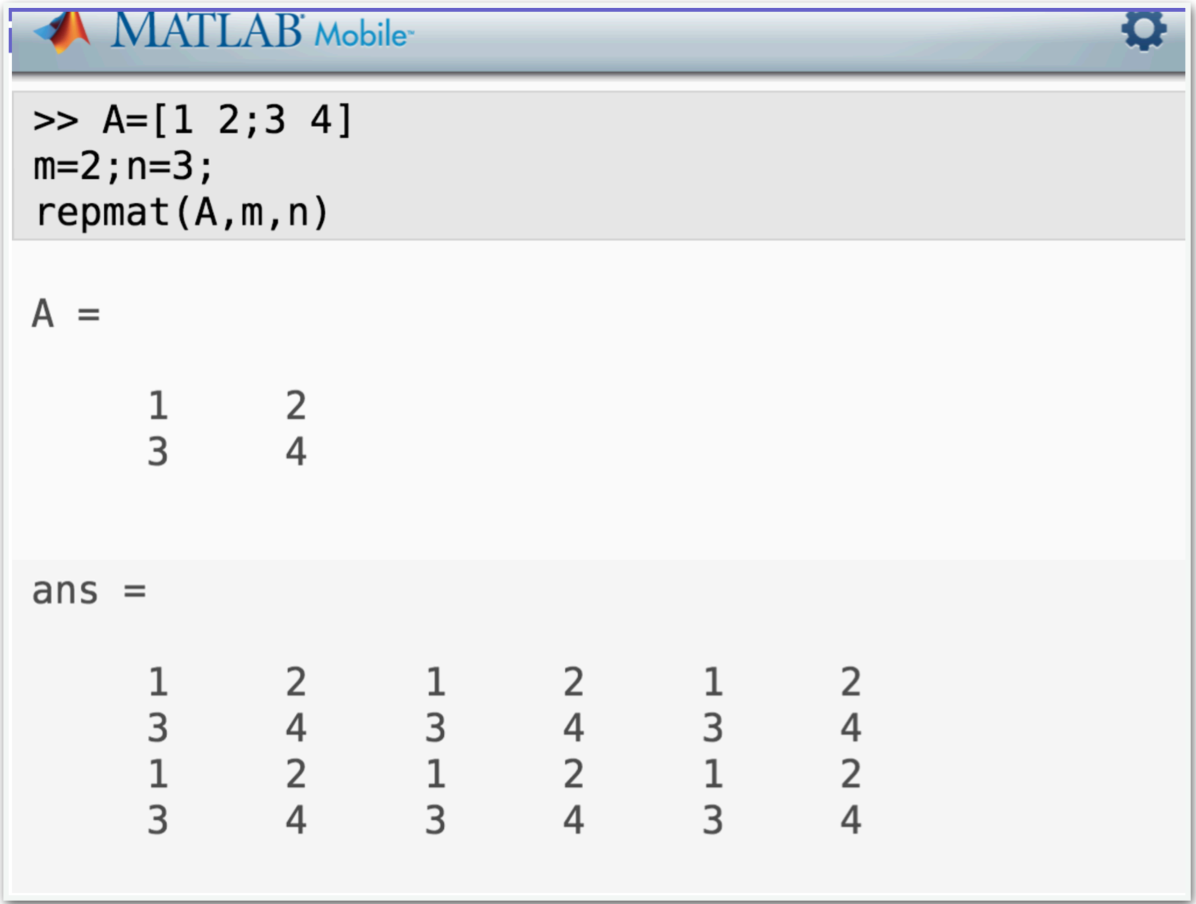
Topic
Problem statement
Material and Data generation
Method and procedure
numerical results
summary



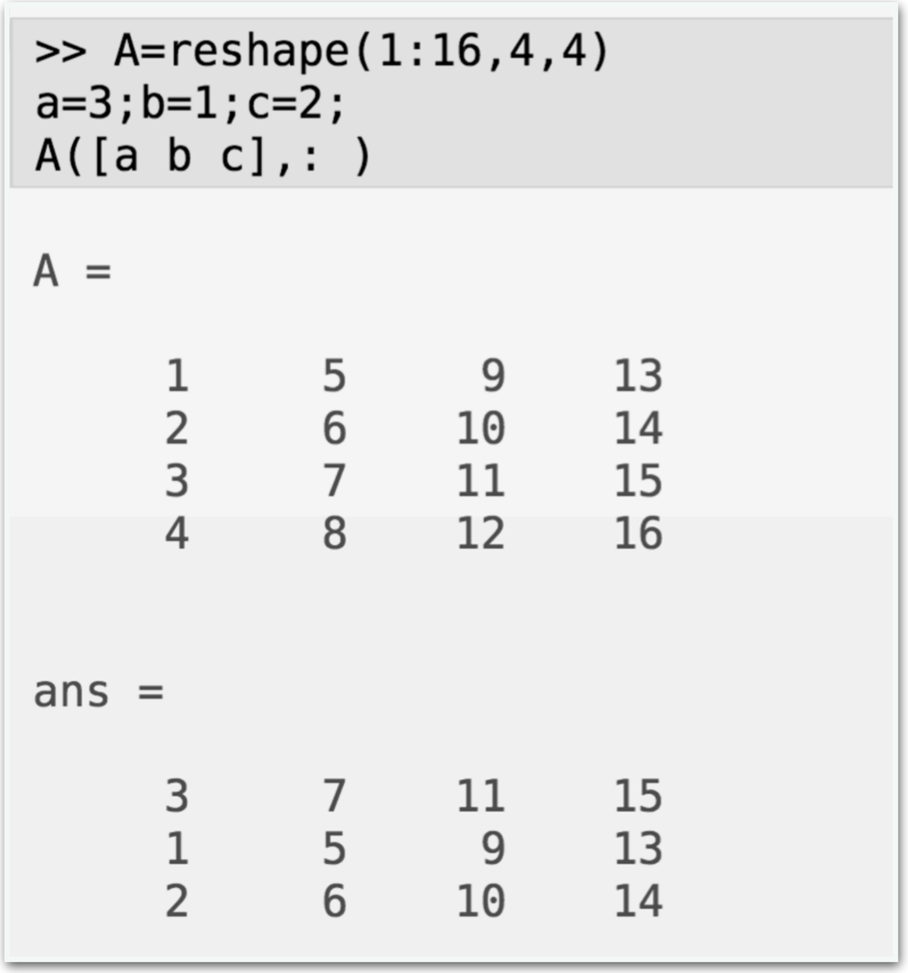
```
>> dir

.           .session   Shared
..          Published  cmw2.bmp

>> I=imread('cmw2.bmp');
>> image(I)
```

```
I2=repmat(Is,2,3);
imagesc(I2);
colormap(gray)
```

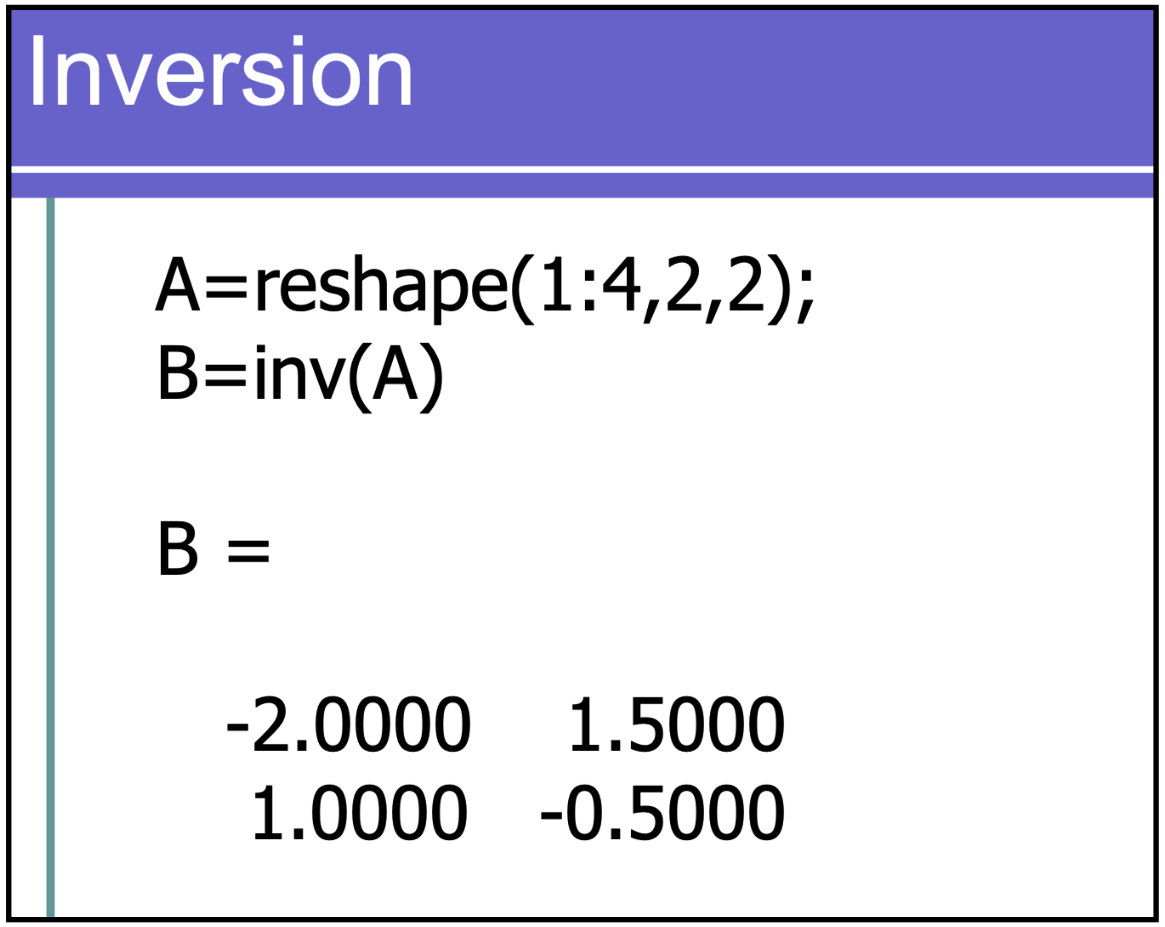




```
>> A=[1 2;3 4]
m=2;n=3;
repmat(A,m,n)

A =

     1     2
     3     4


ans =

     1     2     1     2     1     2
     3     4     3     4     3     4
     1     2     1     2     1     2
     3     4     3     4     3     4
```



```
>> A=reshape(1:16,4,4)
a=3;b=1;c=2;
A([a b c],: )

A =

     1     5     9    13
     2     6    10    14
     3     7    11    15
     4     8    12    16


ans =

     3     7    11    15
     1     5     9    13
     2     6    10    14
```



Inversion

```
A=reshape(1:4,2,2);
B=inv(A)

B =

   -2.0000    1.5000
    1.0000   -0.5000
```

```
A=[2 1 -1;-3 -2 5;1 1 1];
b=[1 0 5]'
```

```
inv(A)*b

ans =

   -1.6000
    5.4000
    1.2000
```

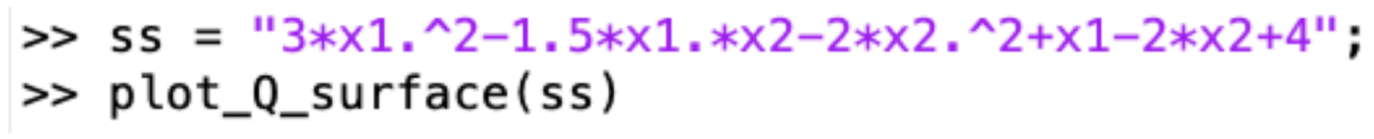
```
A=[2 1 -1;-3 -2 5;1 1 1];
b=[1 0 5]'
```

- A\b
- % Left devision
- % Ax=b could be solved by left devision

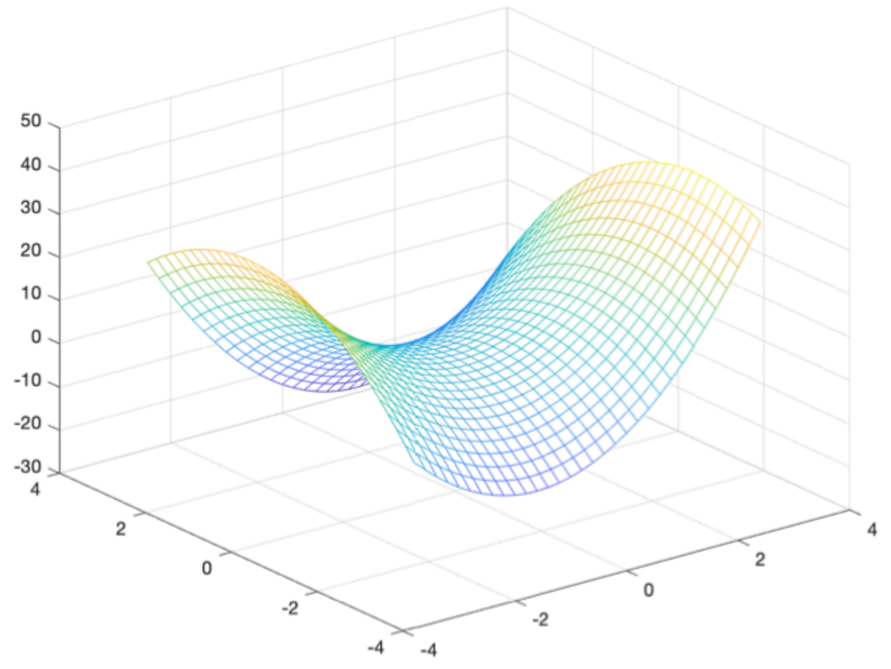
```
A=reshape(1:4,2,2);
>> det(A)

ans =

    -2
```

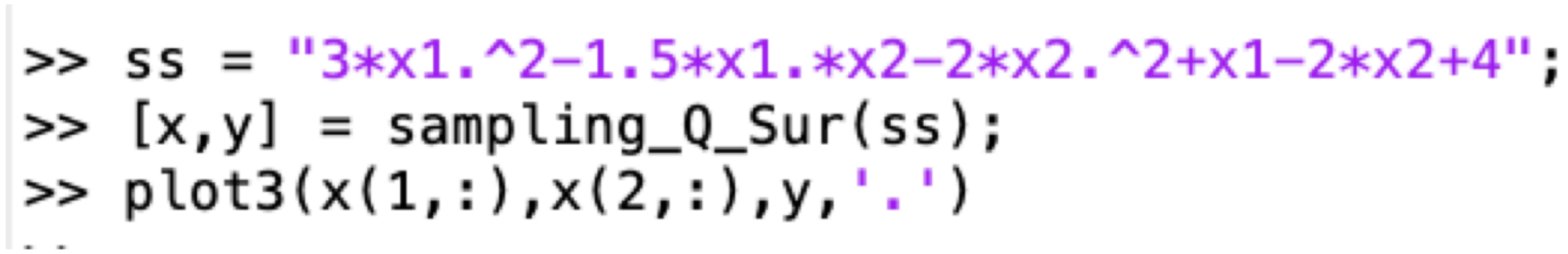
Q surface :

$$3x_1^2 - 1.5x_1x_2 - 2x_2^2 + x_1 - 2x_2 + 4$$

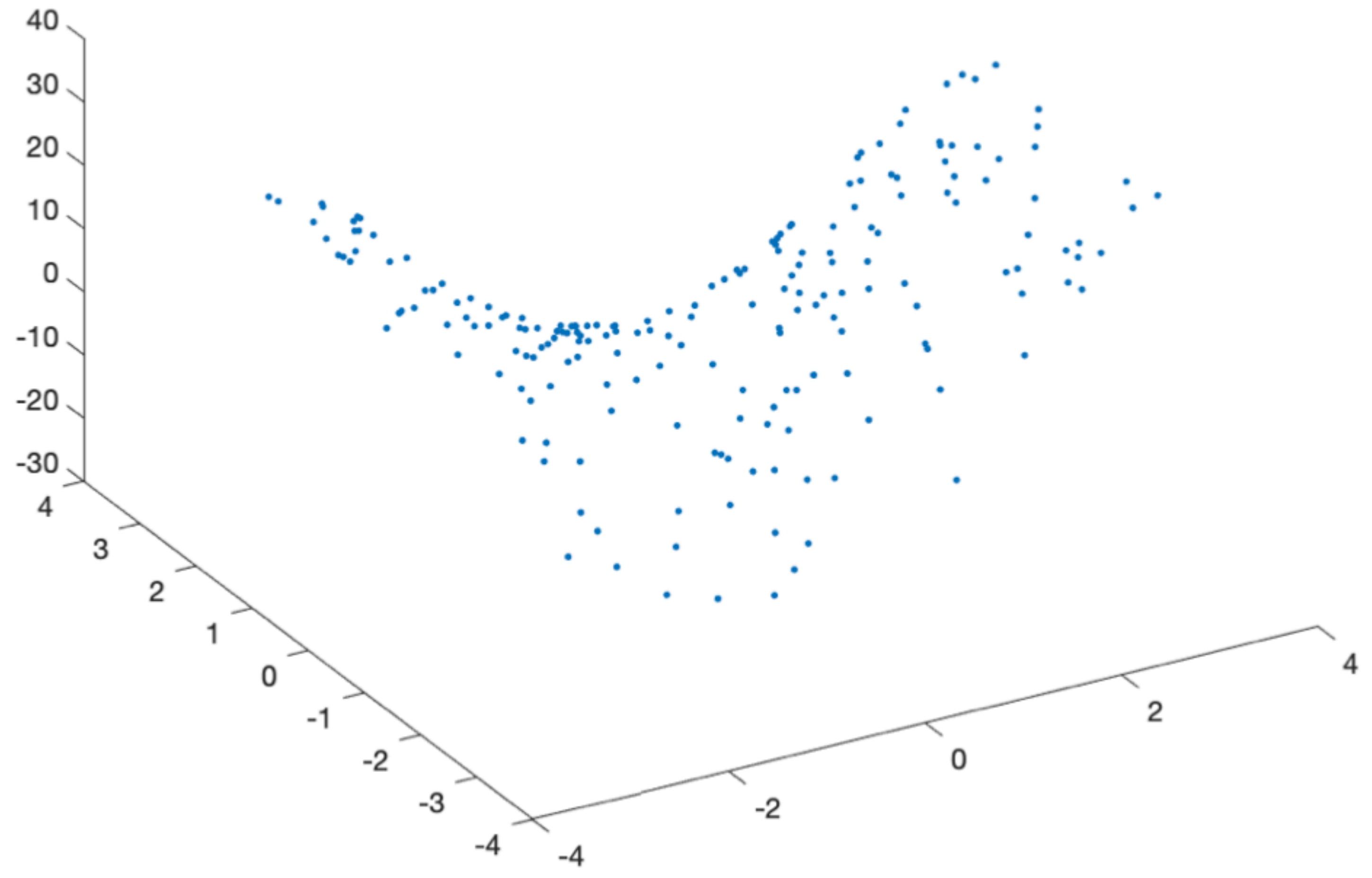


```
>> ss = "3*x1.^2-1.5*x1.*x2-2*x2.^2+x1-2*x2+4";
>> plot_Q_surface(ss)
```





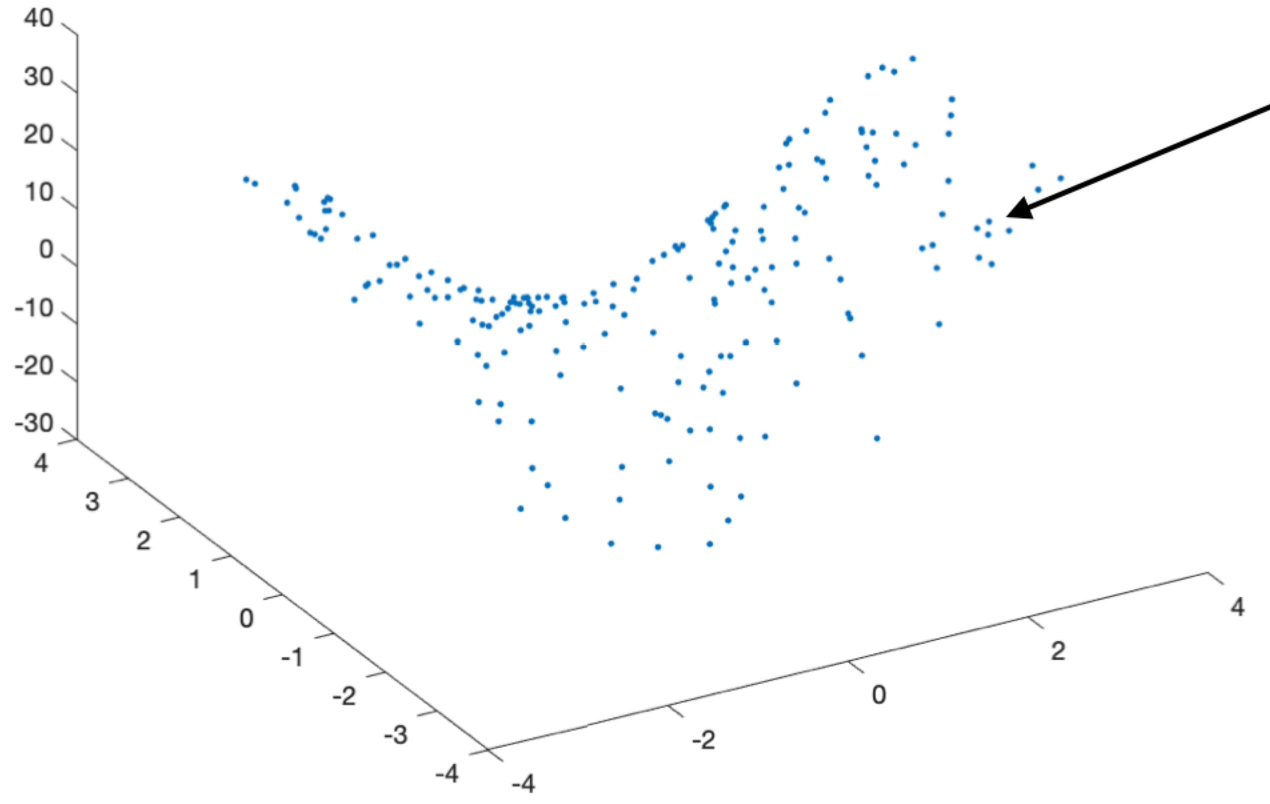
```
>> ss = "3*x1.^2-1.5*x1.*x2-2*x2.^2+x1-2*x2+4";
>> [x,y] = sampling_Q_Sur(ss);
>> plot3(x(1,:),x(2,:),y,'.')
```



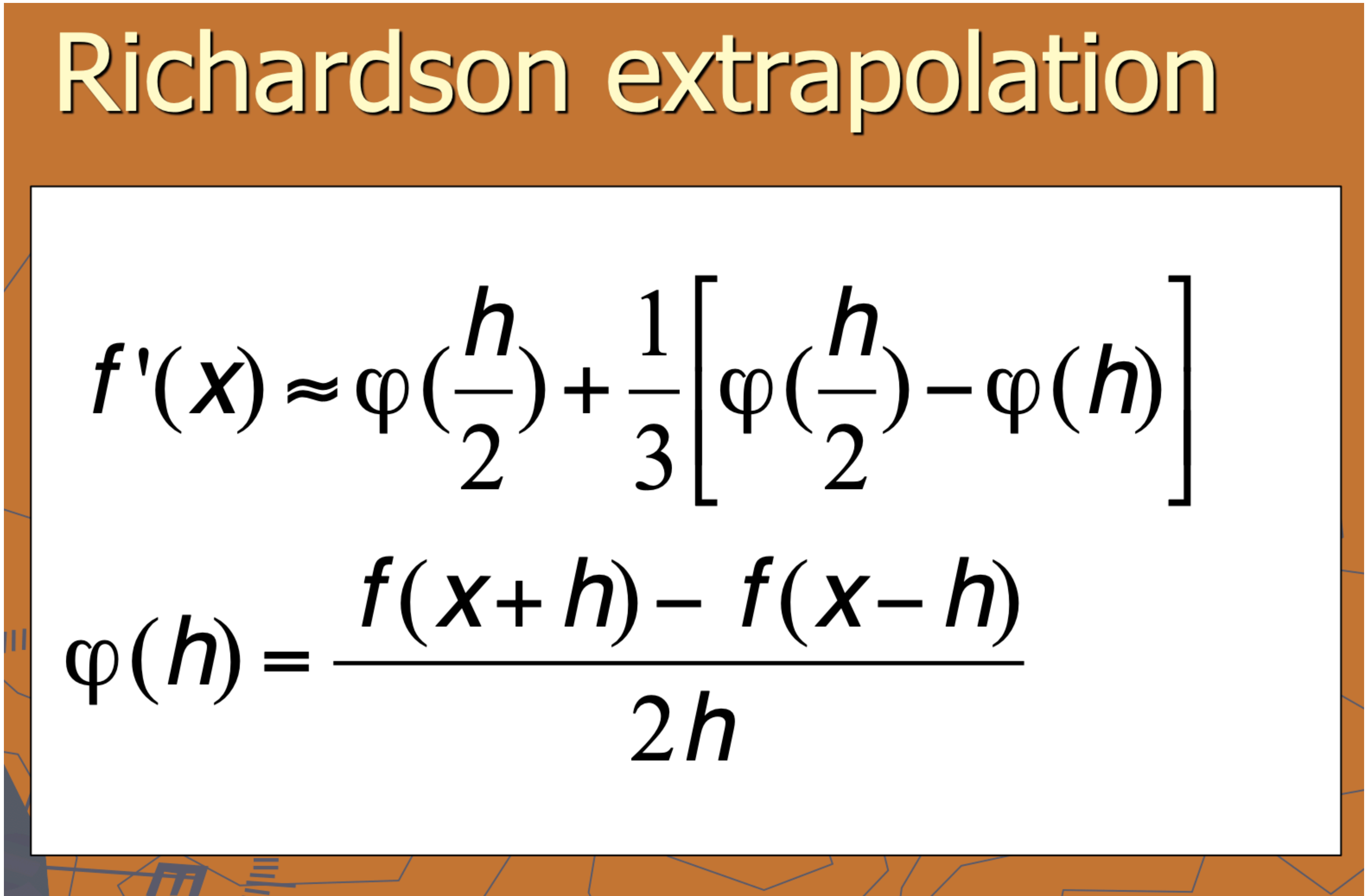
Q surface coefficients :

$$c_1x_1^2 + c_2x_1x_2 + c_3x_2^2 + c_4x_1 + c_5x_2 + c_6 = y$$

A 3D point: $(x_1[i], x_2[i], y[i])$



數值微分在牛頓法求根與Jacobian計算應用

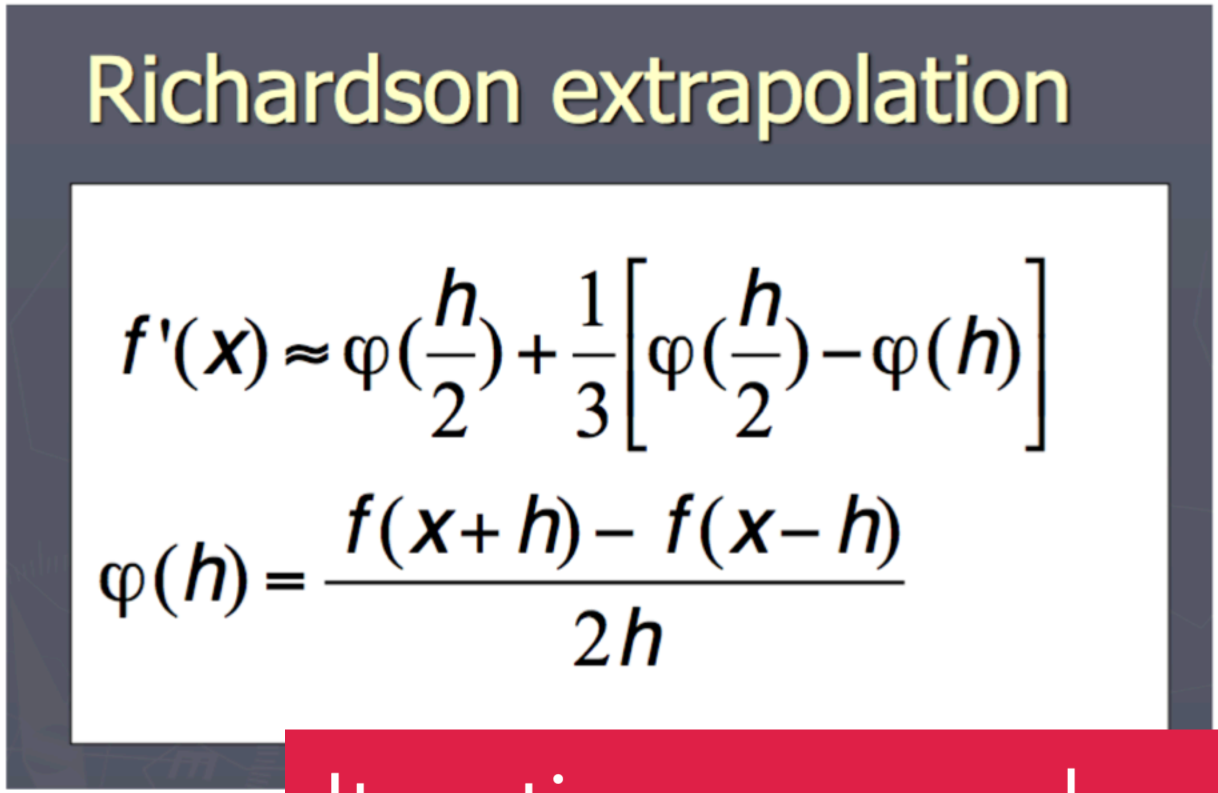


Richardson extrapolation

$$f'(x) \approx \varphi\left(\frac{h}{2}\right) + \frac{1}{3}\left[\varphi\left(\frac{h}{2}\right) - \varphi(h)\right]$$

$$\varphi(h) = \frac{f(x+h) - f(x-h)}{2h}$$

```
function demo_RE()
s = 'x.^2-5*x+6';
f = inline(s);
ss = ['diff(' s ')']
syms x
s1 = eval(ss);
f1 = inline(s1);
z = linspace(-pi,pi);
d = f1(z)-RE(f,z);
mean(abs(d))

function  f_plum = RE(f,x)
% f: inline,
h=0.01;
phy_h=(f(x+h)-f(x-h))/(2*h);
phy_h2=(f(x+h/2)-f(x-h/2))/(2*h/2);
f_plum=phy_h2+1/3*(phy_h2-phy_h);
```

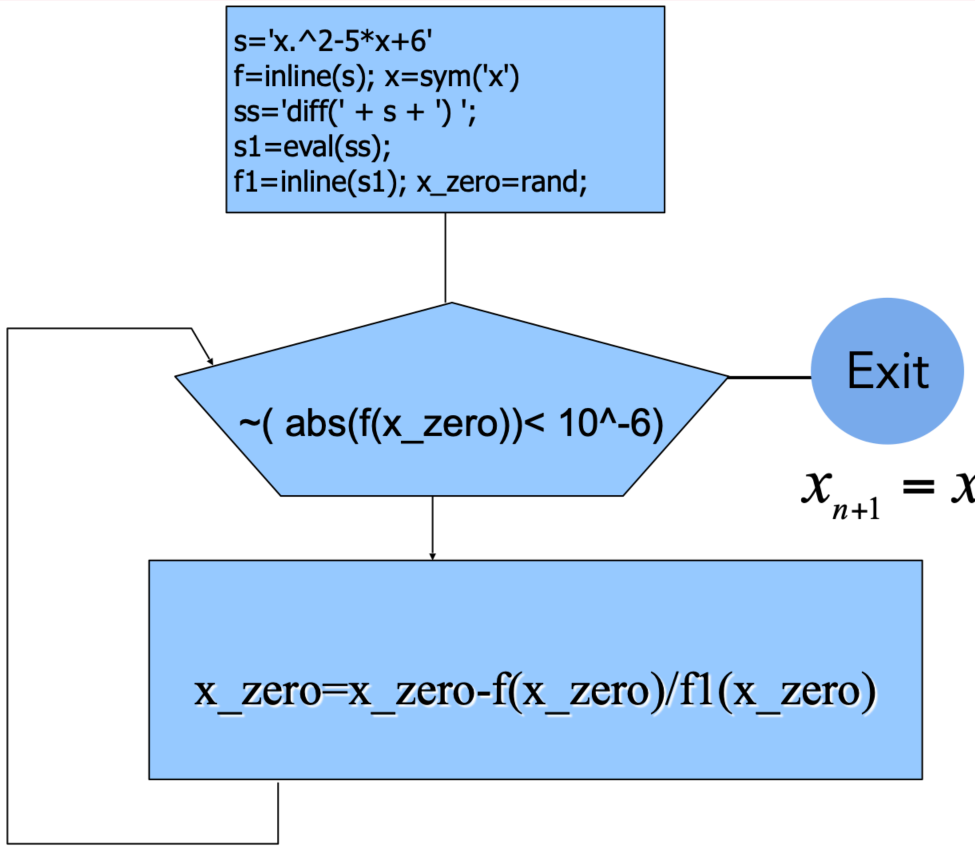


Richardson extrapolation

$$f'(x) \approx \varphi(\frac{h}{2}) + \frac{1}{3}\left[\varphi(\frac{h}{2}) - \varphi(h)\right]$$

$$\varphi(h) = \frac{f(x+h) - f(x-h)}{2h}$$

Iterative approach

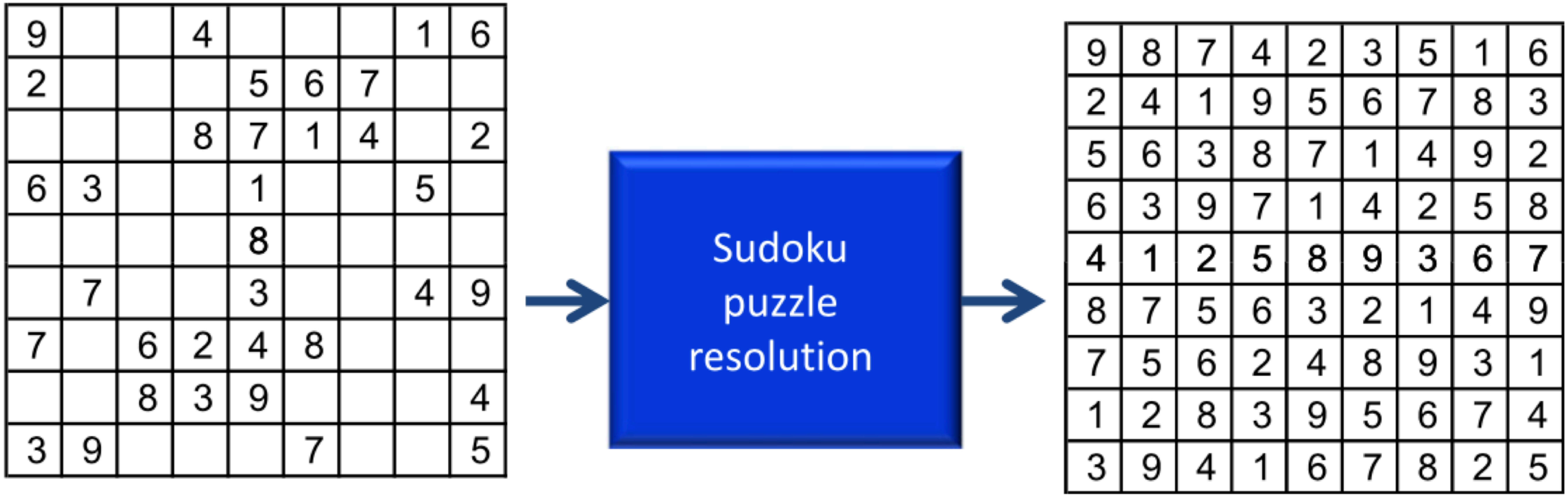


```
s='x.^2-5*x+6'
f=inline(s); x=sym('x')
ss='diff(' + s + ') ';
s1=eval(ss);
f1=inline(s1); x_zero=rand;
```

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

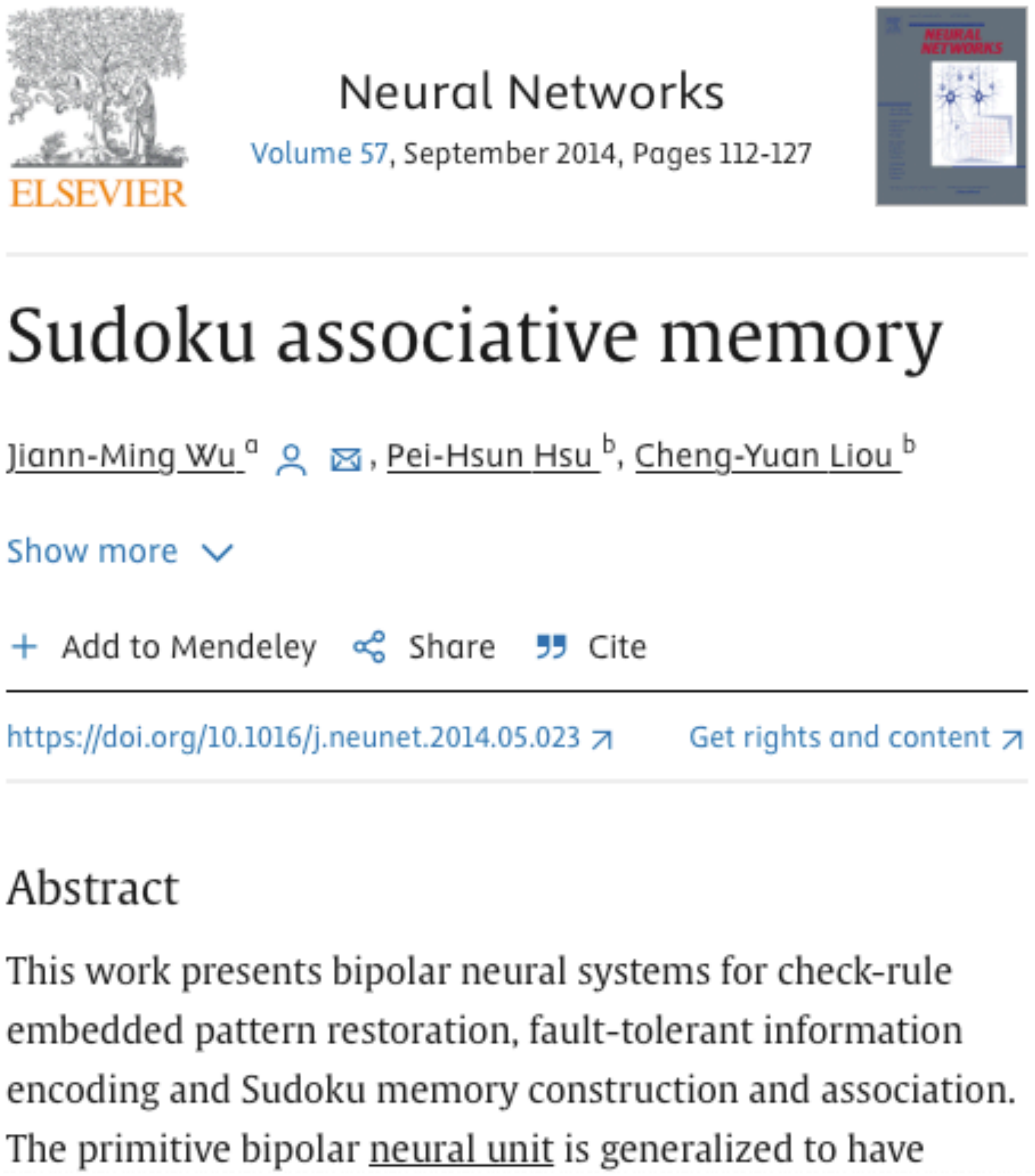
Integer programming and Sudoku Solving

Matlab intlinprog



9			4				1	6
2				5	6	7		
			8	7	1	4		2
6	3			1			5	
				8				
	7			3			4	9
7		6	2	4	8			
		8	3	9				4
3	9				7			5

9	8	7	4	2	3	5	1	6
2	4	1	9	5	6	7	8	3
5	6	3	8	7	1	4	9	2
6	3	9	7	1	4	2	5	8
4	1	2	5	8	9	3	6	7
8	7	5	6	3	2	1	4	9
7	5	6	2	4	8	9	3	1
1	2	8	3	9	5	6	7	4
3	9	4	1	6	7	8	2	5



Neural Networks
Volume 57, September 2014, Pages 112-127

Sudoku associative memory

Jiann-Ming Wu [a], Pei-Hsun Hsu [b], Cheng-Yuan Liou [b]

Show more

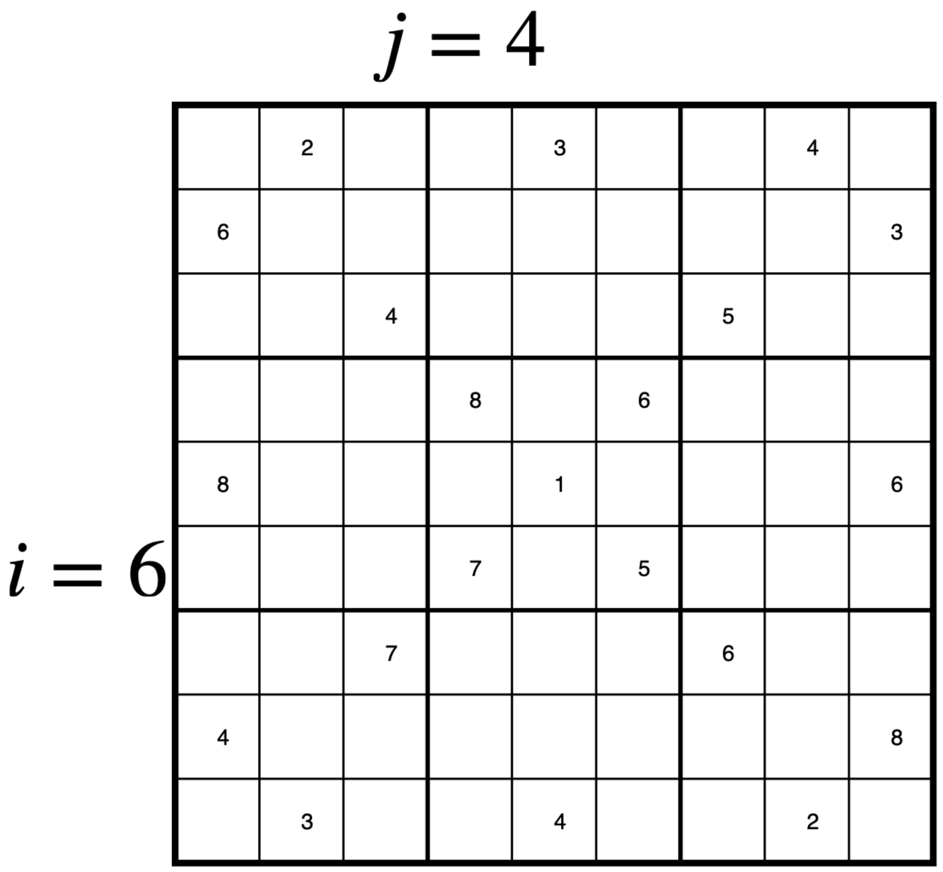
Add to Mendeley Share Cite

https://doi.org/10.1016/j.neunet.2014.05.023 Get rights and content

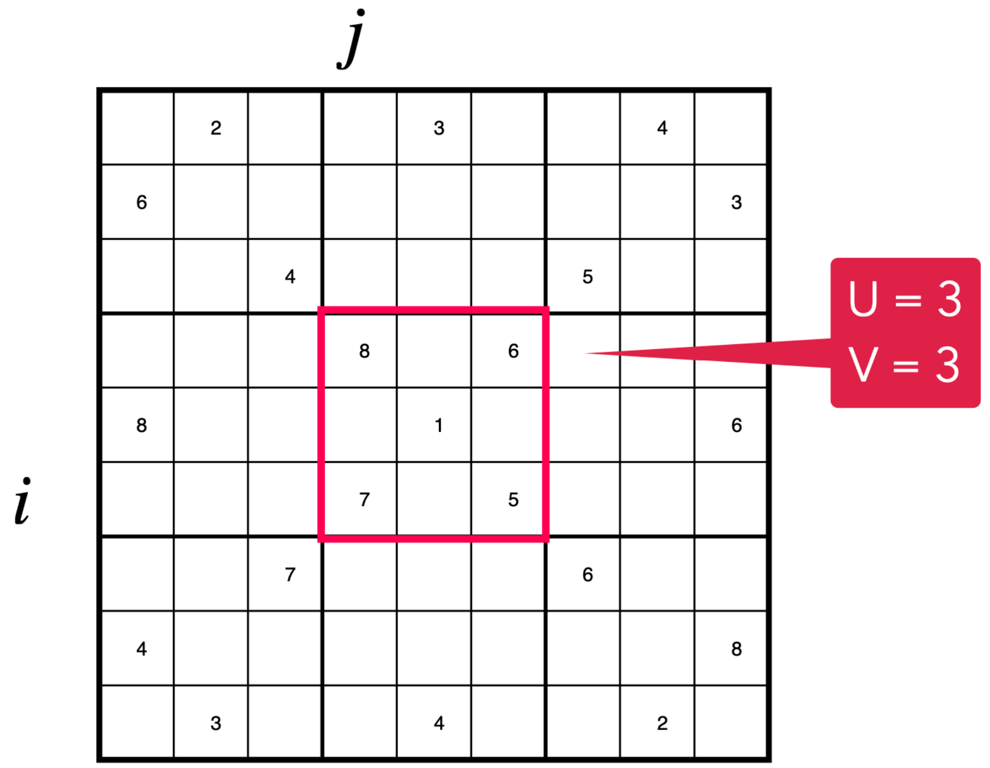
Abstract

This work presents bipolar neural systems for check-rule embedded pattern restoration, fault-tolerant information encoding and Sudoku memory construction and association. The primitive bipolar neural unit is generalized to have

```
B = [1,2,2;
    1,5,3;
    1,8,4;
    2,1,6;
    2,9,3;
    3,3,4;
    3,7,5;
    4,4,8;
    4,6,6;
    5,1,8;
    5,5,1;
    5,9,6;
    6,4,7;
    6,6,5;
    7,3,7;
    7,7,6;
    8,1,4;
    8,9,8;
    9,2,3;
    9,5,4;
    9,8,2];
drawSudoku(B)
```

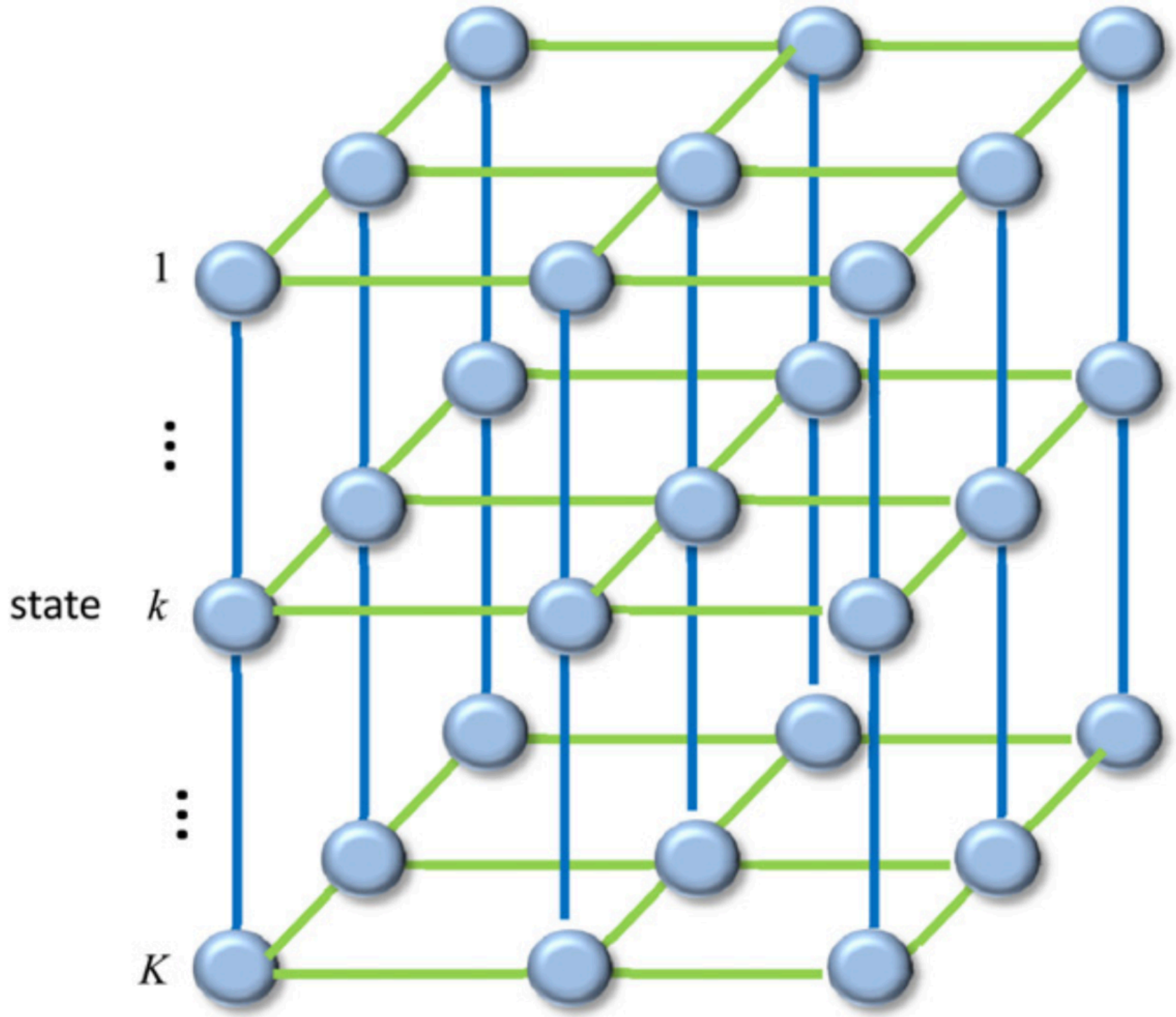


$$x(6,4,7) = 1$$



$$\sum_{i=1}^{3}\sum_{j=1}^{3} x(i+U, j+V, k) = 1, \text{ where } U, V \in \{0,3,6\}$$

$$\sum_{i=1}^{3}\sum_{j=1}^{3} x(i+3, j+3, k) = 1 \quad \text{For k = 1,...,9}$$



$$\sum_{k} x(i,j,k) = 1$$

$$\sum_{j} x(i,j,k) = 1$$

$$\sum_{i} x(i,j,k) = 1$$

	2			3			4	
6								3
		4				5		
			8		6			
8				1				6
			7		5			
		7				6		
4								8
	3			4			2	

Neural networks for regression, function approximation and prediction

APPROXIMATION

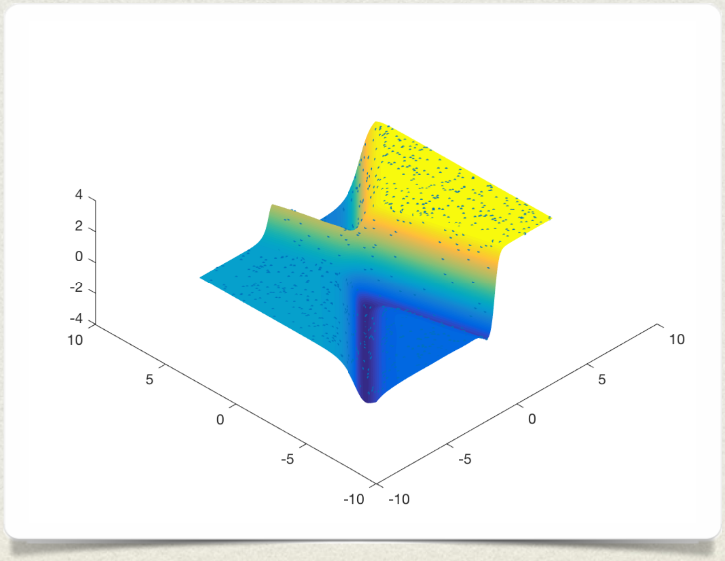
$$v_1 = tanh(\boxed{2x_1 + 0.5x_2 - 1})$$
$$v_2 = tanh(\boxed{x_1 - x_2 + 1})$$
$$y = 2cos(\boxed{2v_1 - v_2 - 1}) + sin(\boxed{v_1 + v_2 - 1})$$

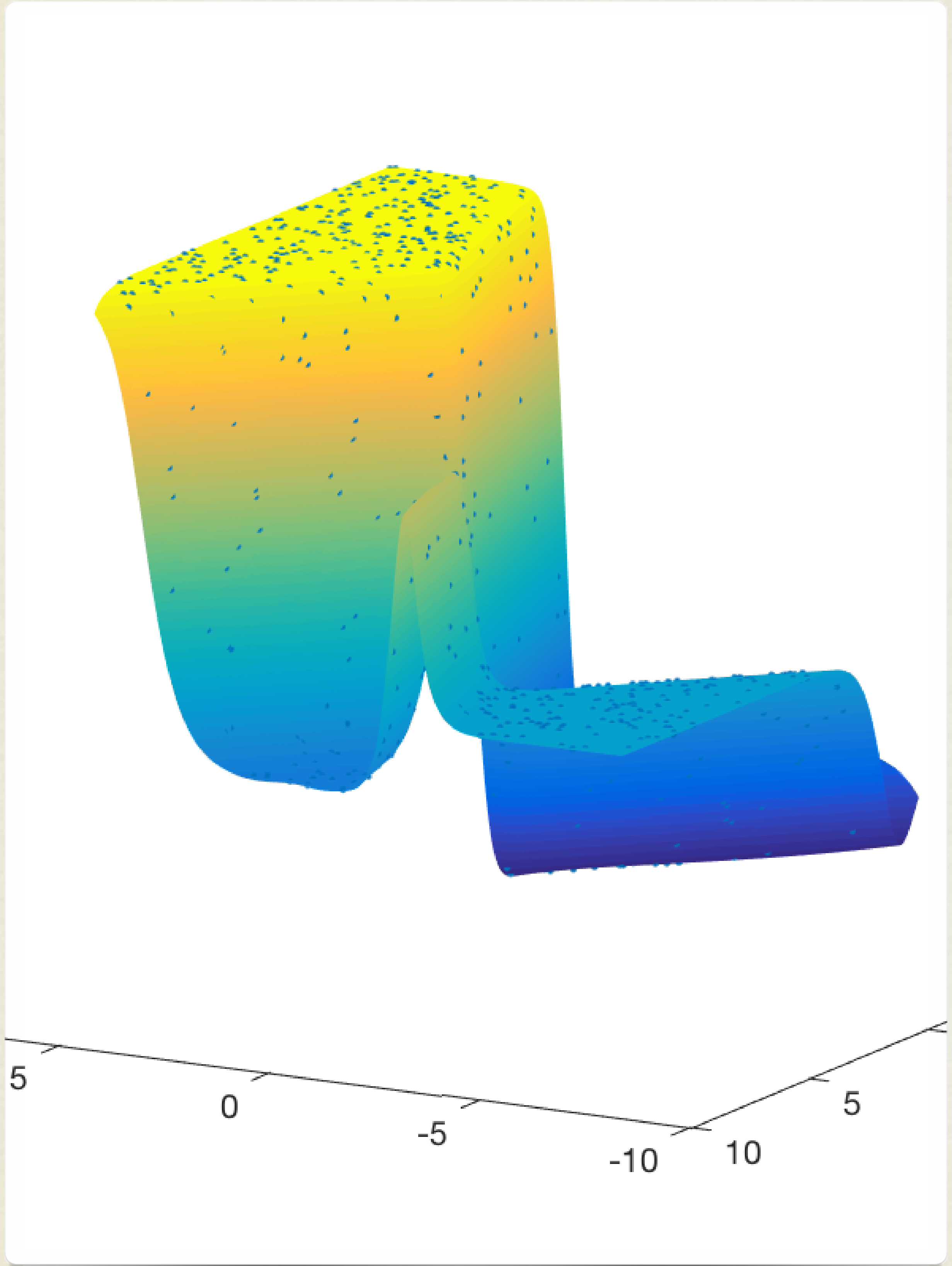
input
output

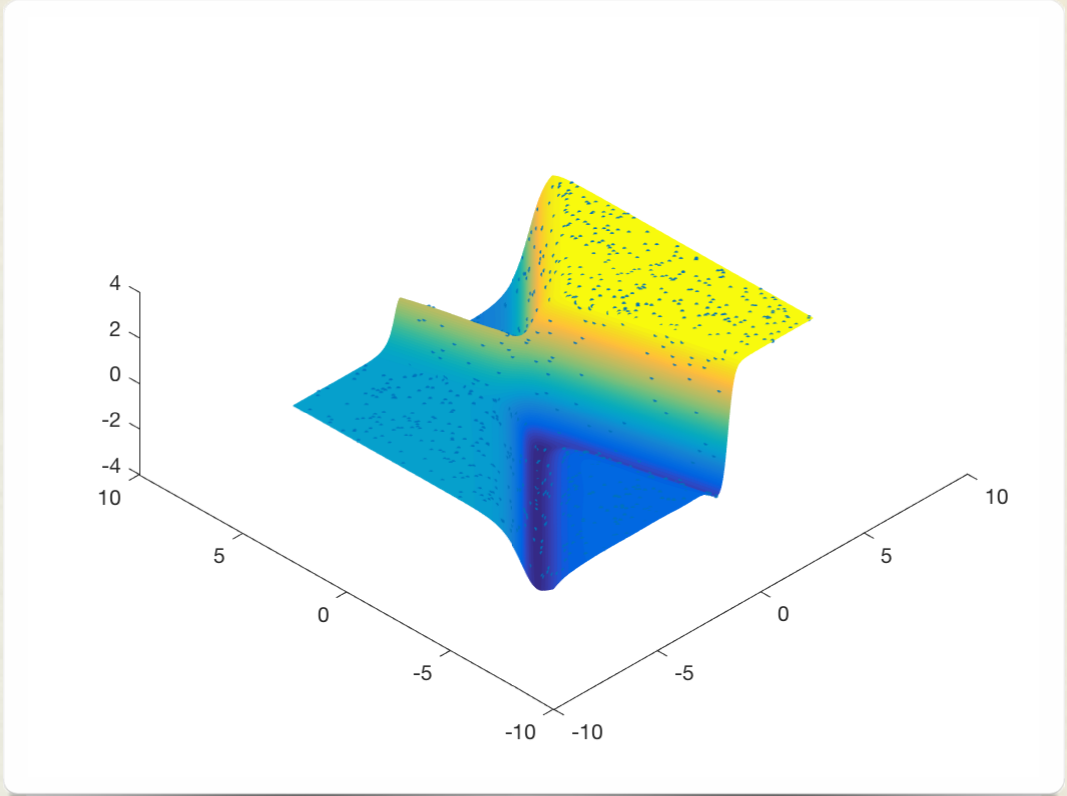
Learning a deep neural network

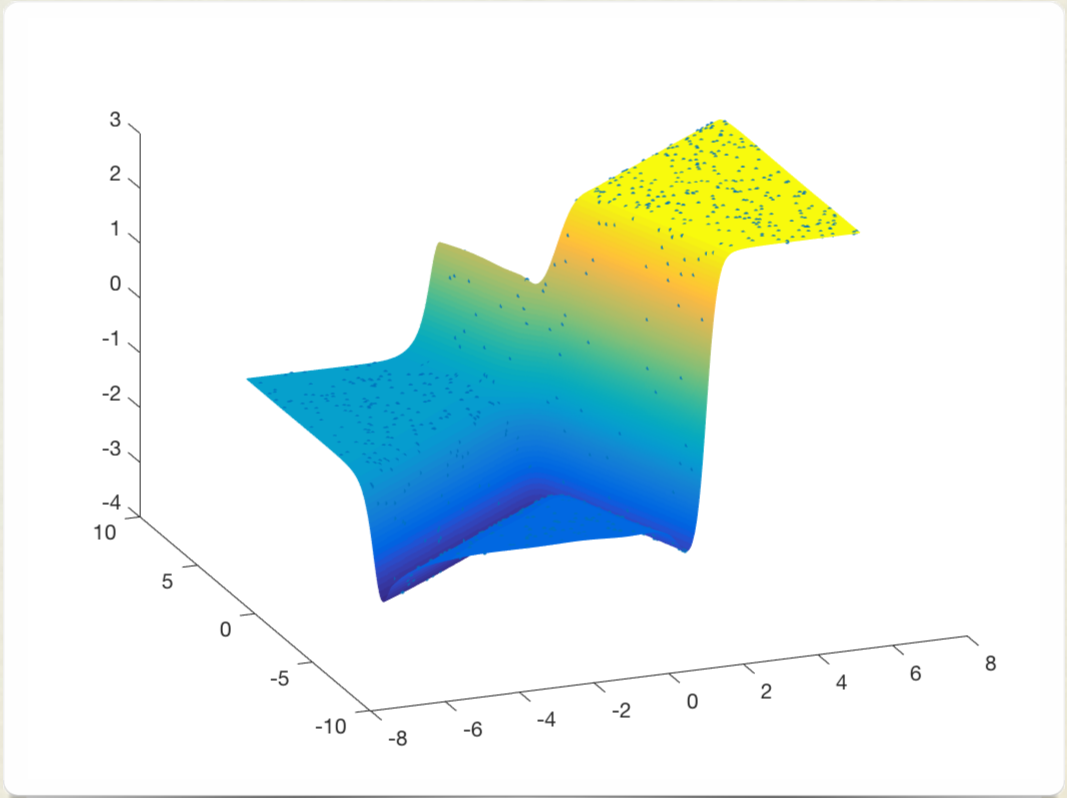
optimal interconnections

An approximating network







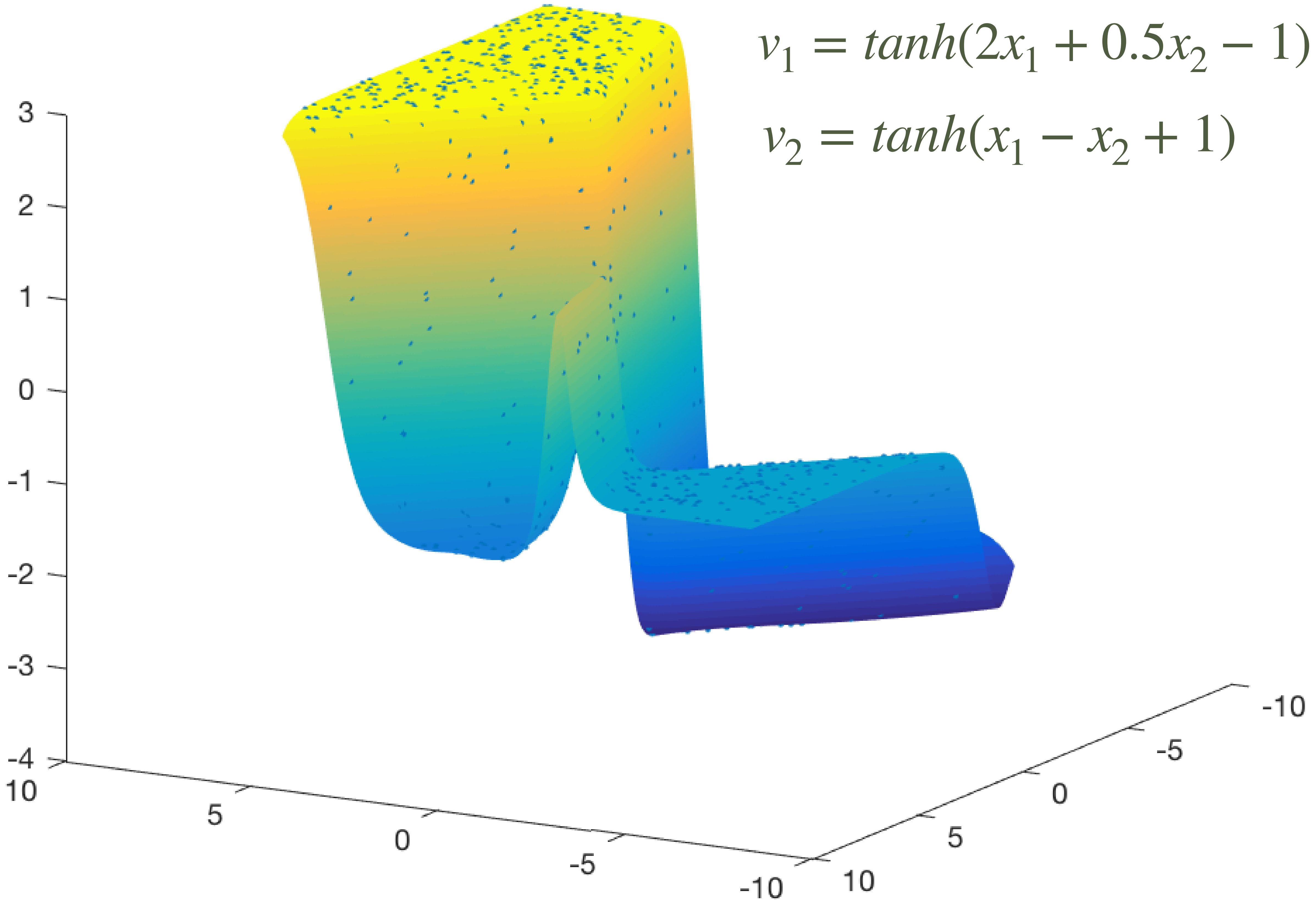


```
v = [tanh(2*x(:,1)+0.5*x(:,2)−1) tanh(x(:,1)−x(:,2)+1)];
y= 2*cos(2*v(:,1)−v(:,2)−1)+sin(v(:,1)+v(:,2)−1);
```

$$y = 2cos(2v_1 - v_2 - 1) + sin(v_1 + v_2 - 1)$$

$$v_1 = tanh(2x_1 + 0.5x_2 - 1)$$

$$v_2 = tanh(x_1 - x_2 + 1)$$



APPROXIMATION

$$v_1 = sign(2x_1 + 0.5x_2 - 1)$$

$$v_2 = sign(x_1 - x_2 + 1)$$

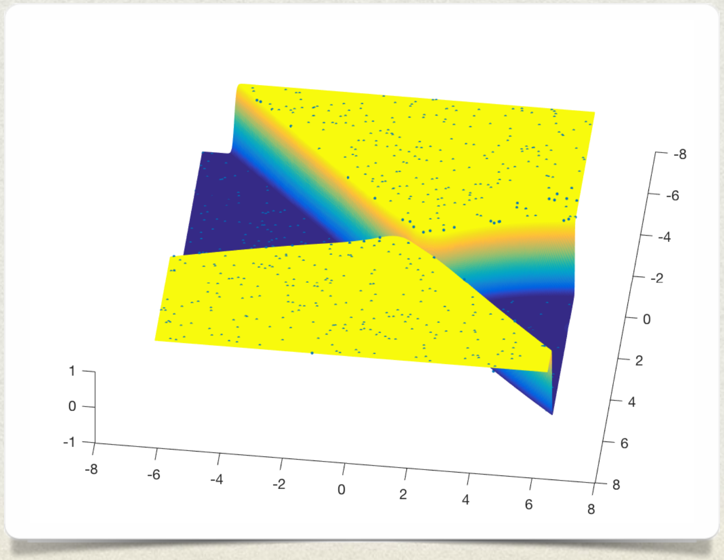
$$y = sign(v_1 v_2)$$

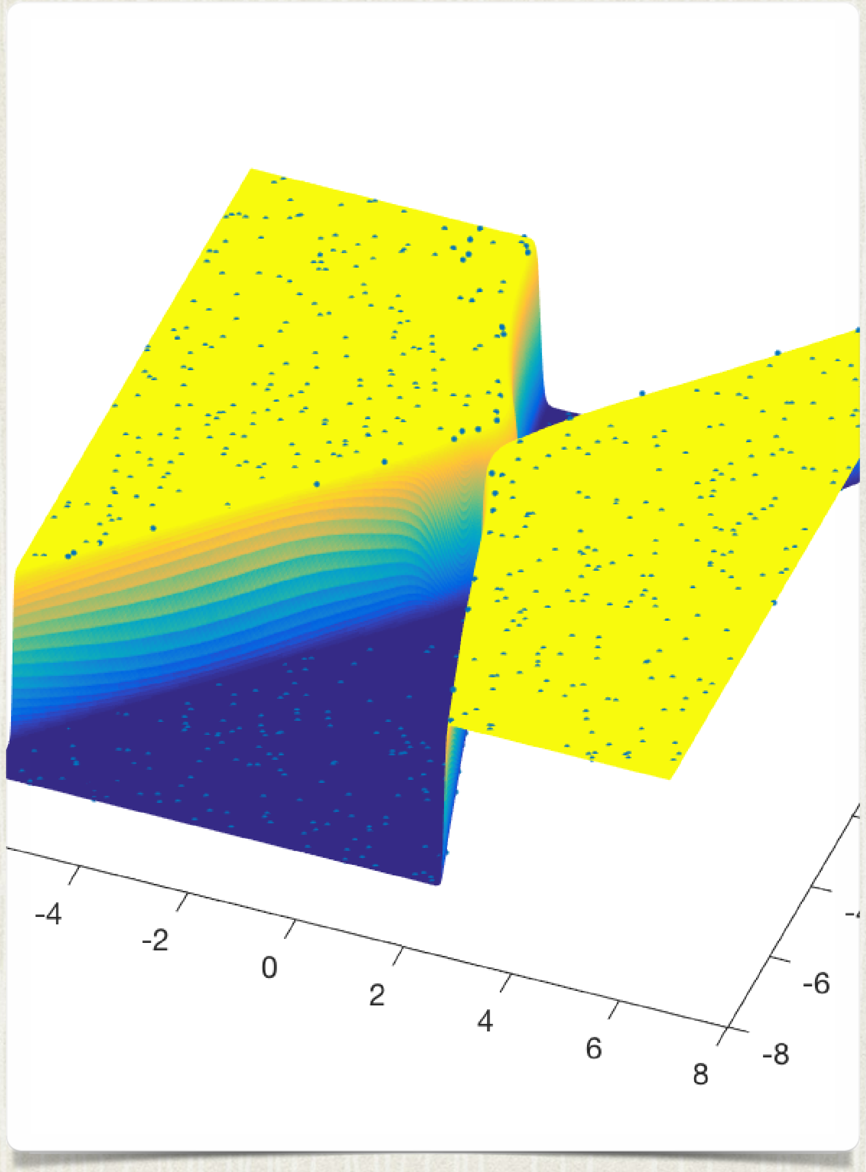
input
output

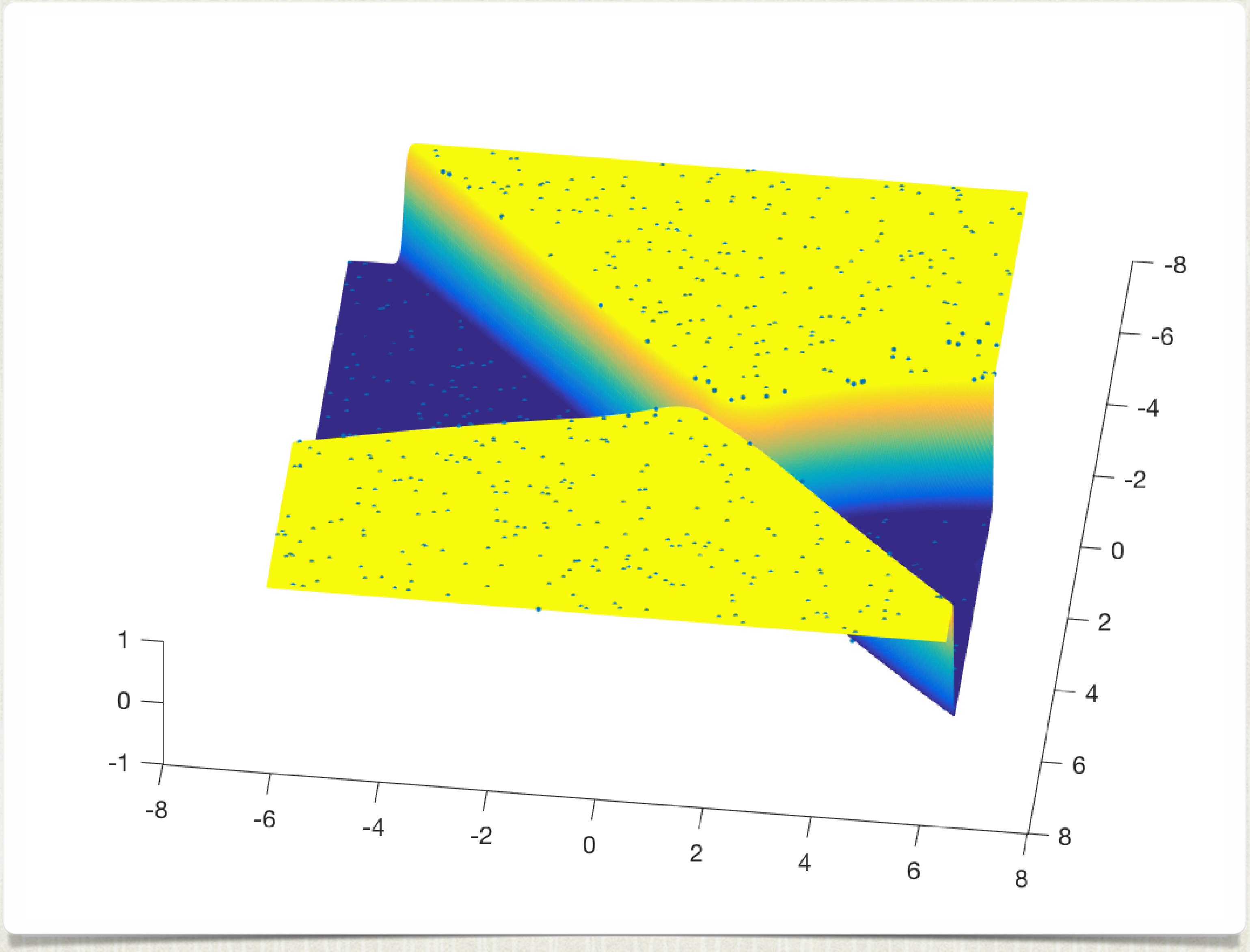
Learning a deep neural network

optimal interconnections

An approximating network

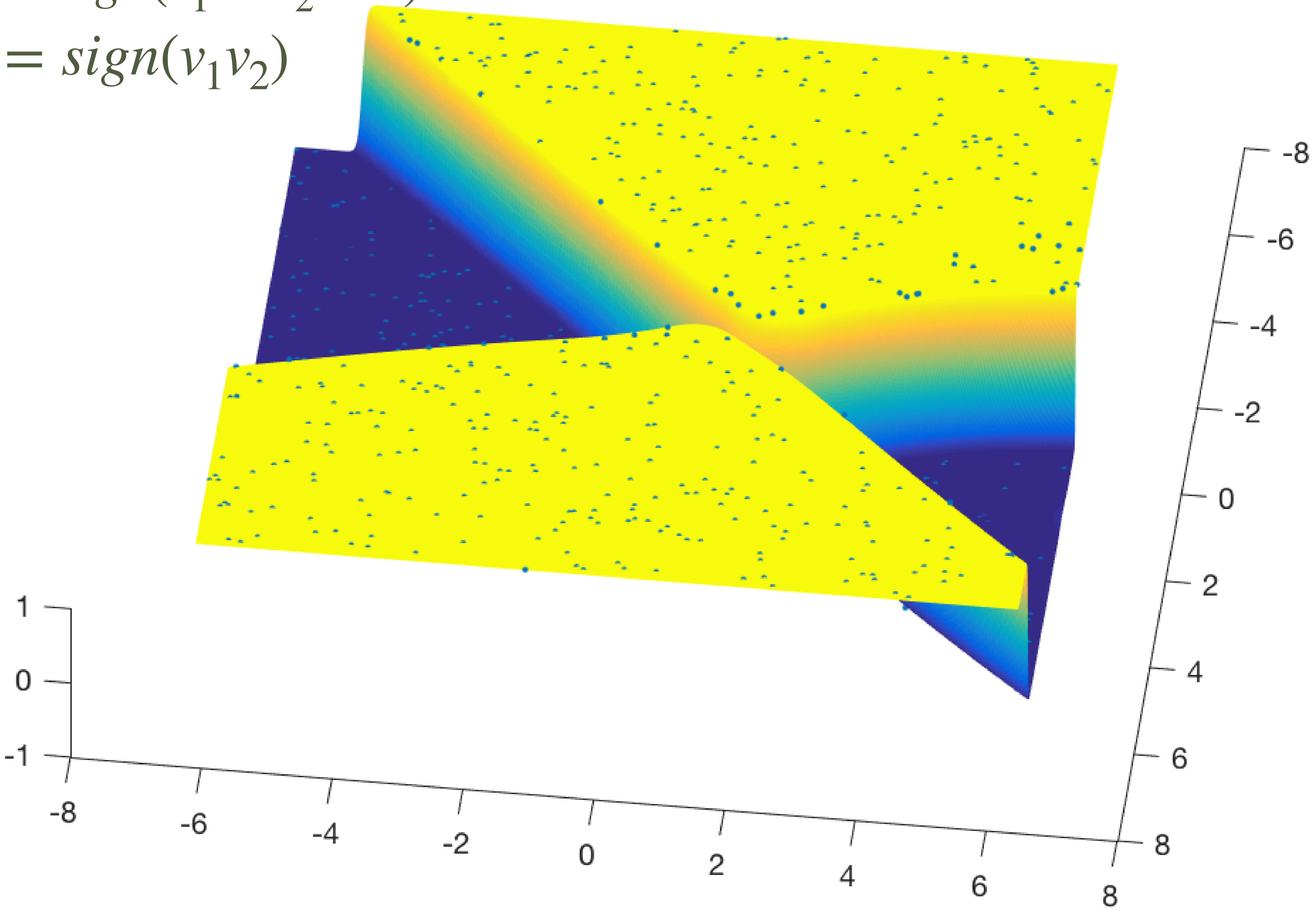


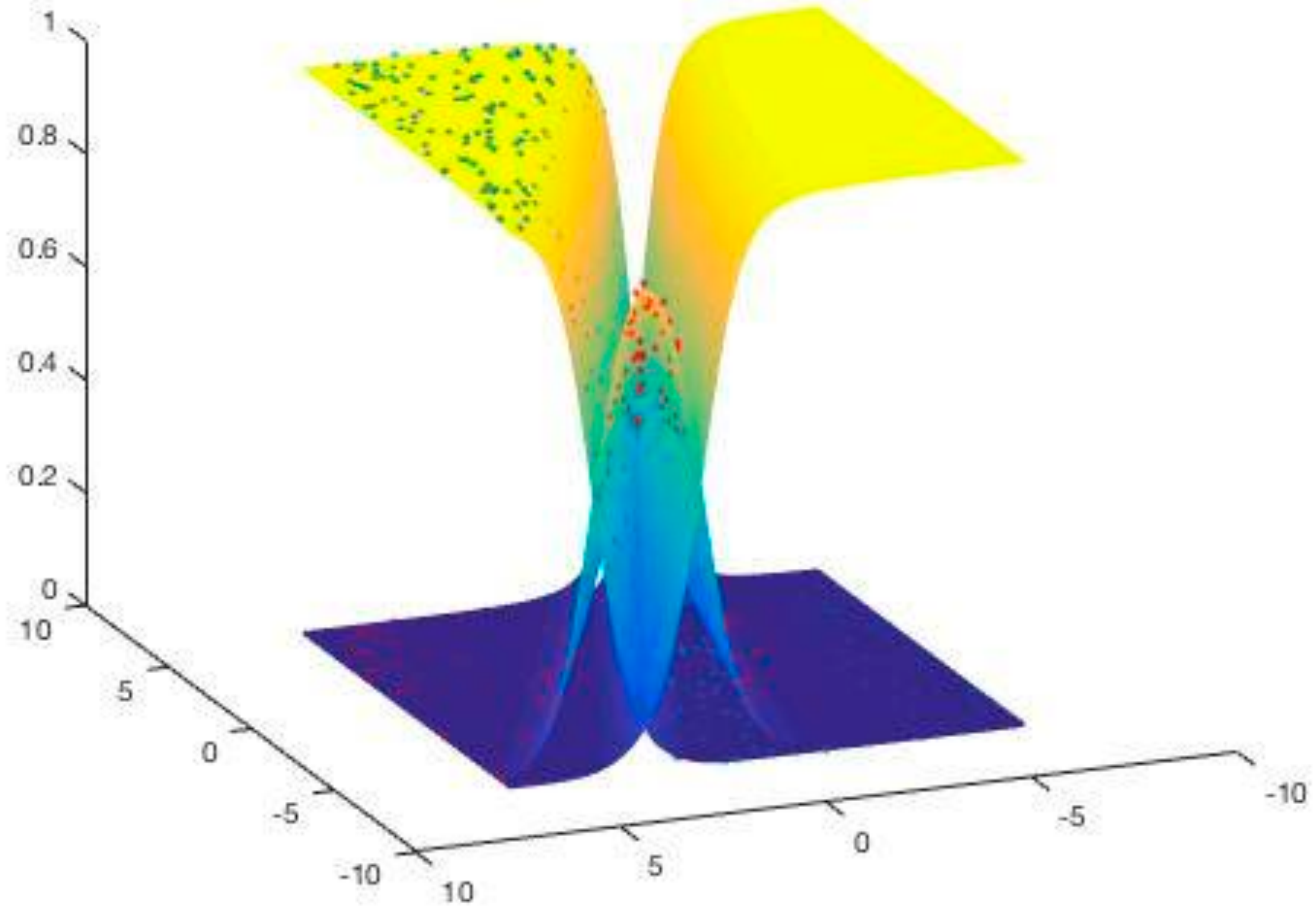




```
v = [sign(2*x(:,1)+0.5*x(:,2)-1) sign(x(:,1)-x(:,2)+1)];
y= sign(v(:,1).*v(:,2));
```

$$v_1 = sign(2x_1 + 0.5x_2 - 1)$$
$$v_2 = sign(x_1 - x_2 + 1)$$
$$y = sign(v_1 v_2)$$




1
0.8
0.6
0.4
0.2
0
10
5
0
-5
-10
10
5
0
-5
-10

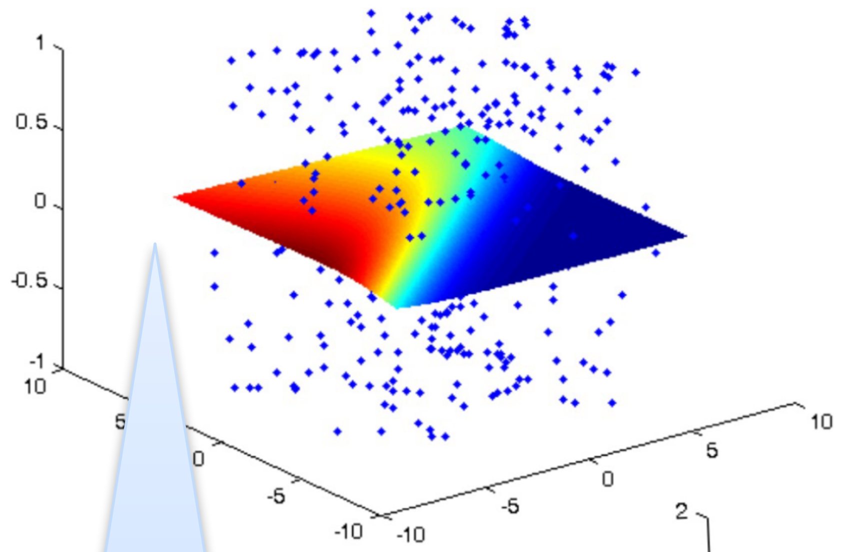
Data oriented function approximation

- Adaptable function

$$y = F(x \mid \theta_{opt})$$

Optimal network parameters

An adaptable network mapping



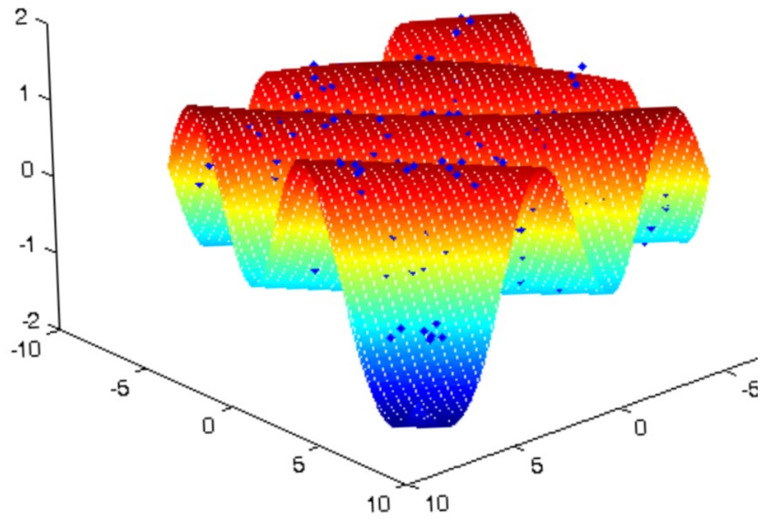


Table 1
Target functions.

$f_1(\mathbf{x}) = \sin(x_1 + x_2)$
$f_2(\mathbf{x}) = x_1^2 + x_2^2$
$f_3(\mathbf{x}) = 0.5x_1^2 - 0.9x_2^2$
$f_4(\mathbf{x}) = \exp(-0.05x_1^2 - 0.09x_2^2)$
$f_5(\mathbf{x}) = \sin([1, -1]^T x) + \exp(-x^T A x)$
$f_6(\mathbf{x}) = \tanh(0.8x_1 + 0.2x_2) + \sin(0.3x_1 - 0.9x_2)$
$f_7(\mathbf{x}) = 0.5\sin(x_1 + x_2) + 0.2x_1 - 0.2x_2$
$f_8(\mathbf{x}) = \exp(-(\mathbf{x} - \mathbf{w}_1)^T A(\mathbf{x} - \mathbf{w}_1)) + \exp(-(\mathbf{x} - \mathbf{w}_2)^T B(\mathbf{x} - \mathbf{w}_1))$
$f_9(\mathbf{x}) = f_8(\mathbf{x}) + 0.5\ \sin(x_1 + 0.3x_2) + 0.5\ \sin(0.2x_1 - 0.8x_2)$
$f_{10}(x) = \sin(x_1 + x_2 + x_3) + \cos(x_1 + x_2 + x_3)$
$f_{11}(x) = \tanh(x_1 + x_2 + x_3 + x_4)$

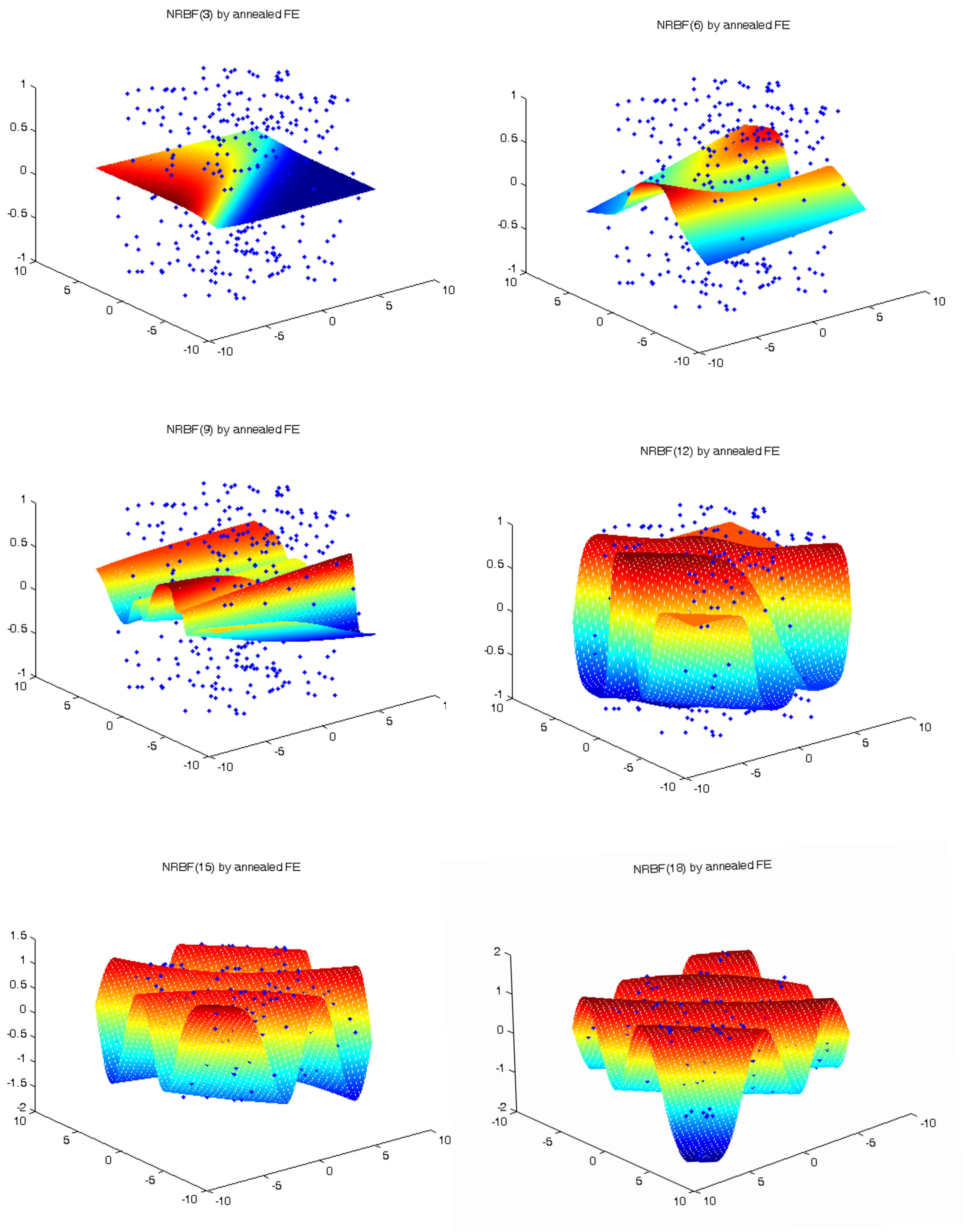

NRBF(3) by annealed FE
NRBF(6) by annealed FE
NRBF(9) by annealed FE
NRBF(12) by annealed FE
NRBF(15) by annealed FE
NRBF(18) by annealed FE

Figure 4

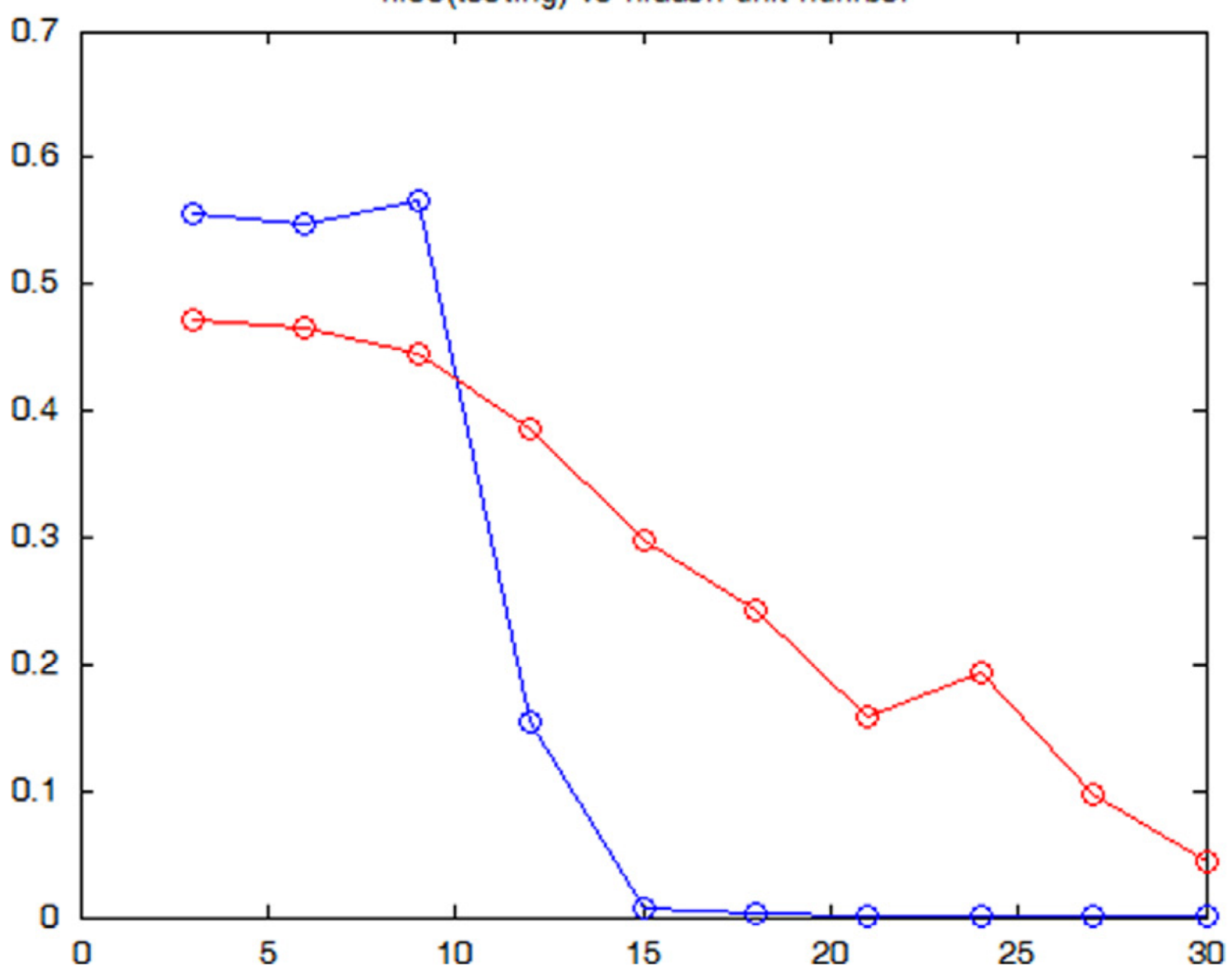
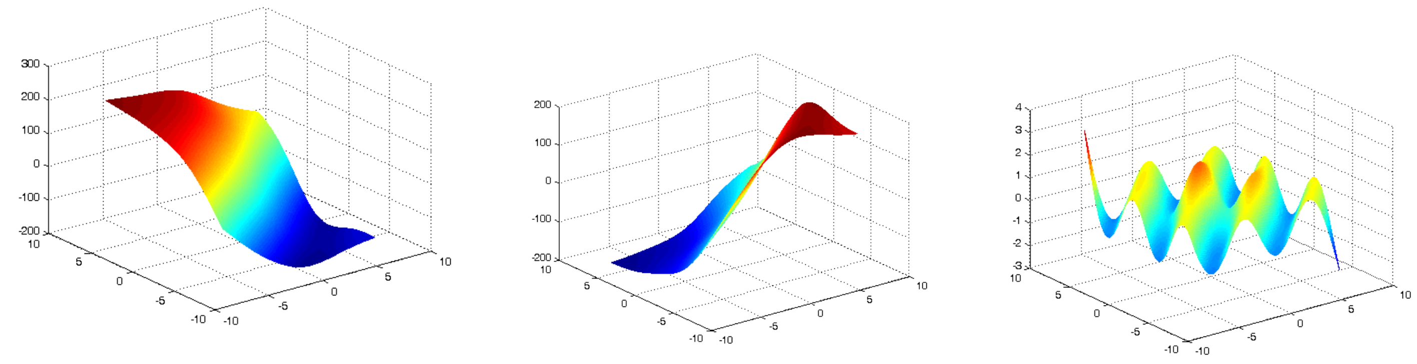
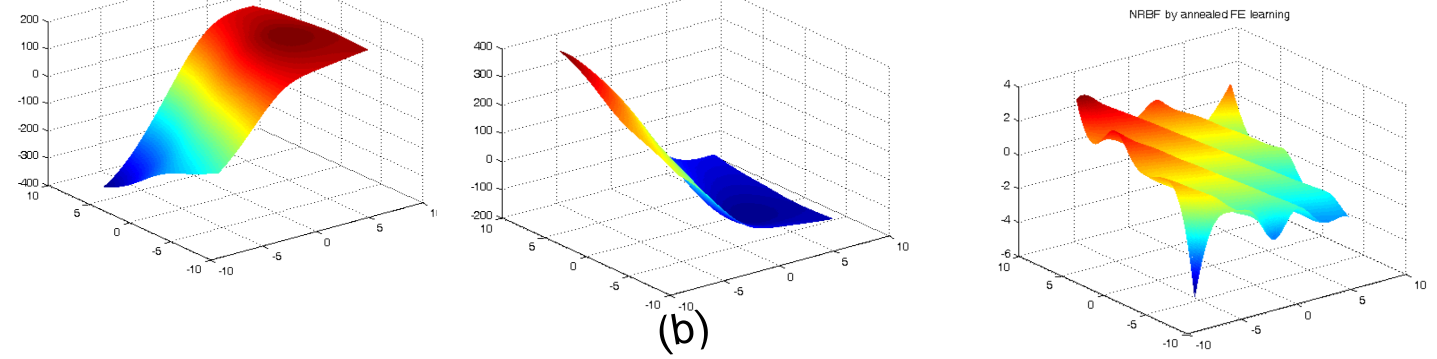


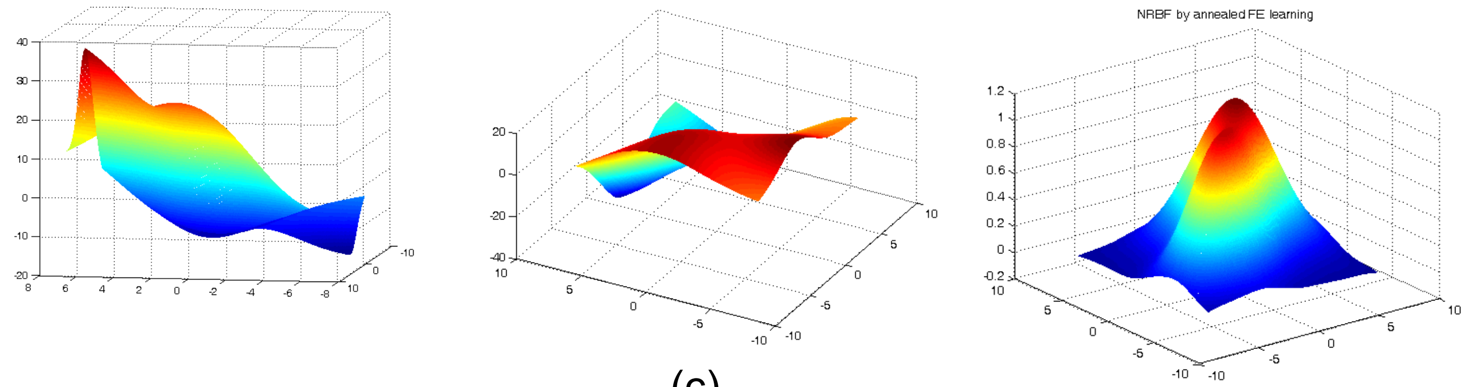
Fig. 5. Mean square testing errors of annealed competitive learning (blue curve) and the Rätsch method (red curve) in approximating f_1 versus the numbers of hidden units. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)



(a)



(b)



(c)

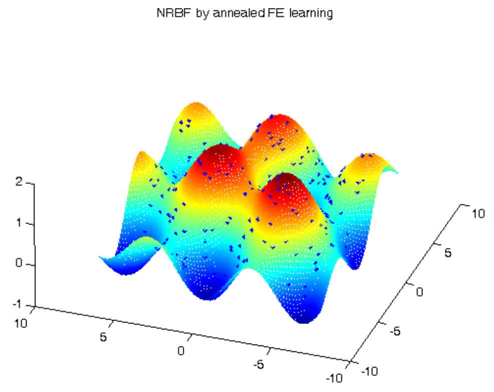
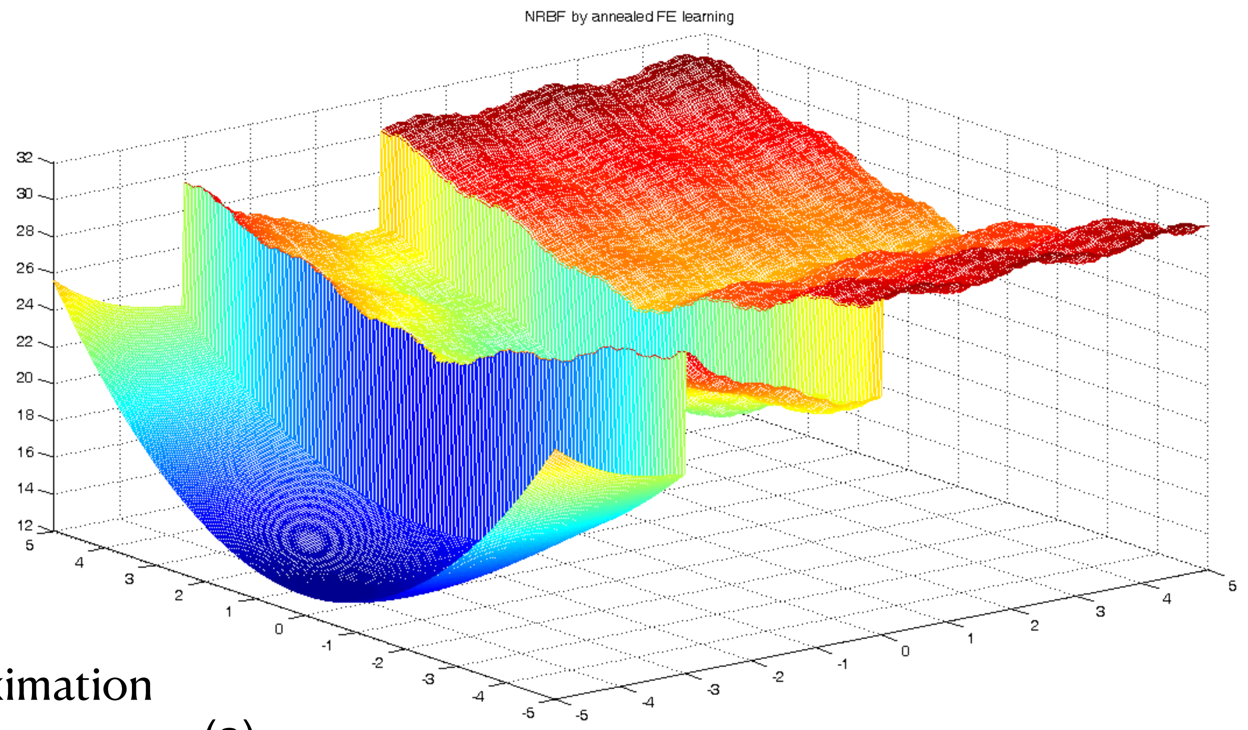


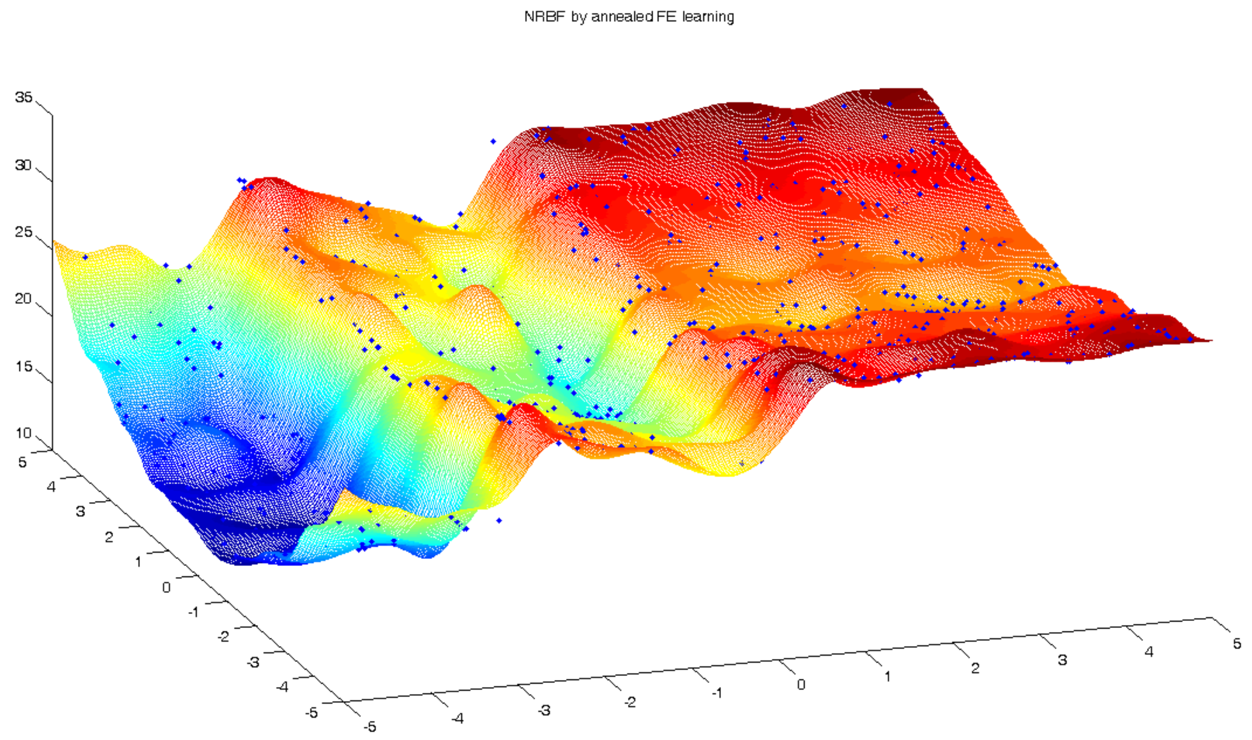
Figure 6

(d)



Data driven function approximation

(a)



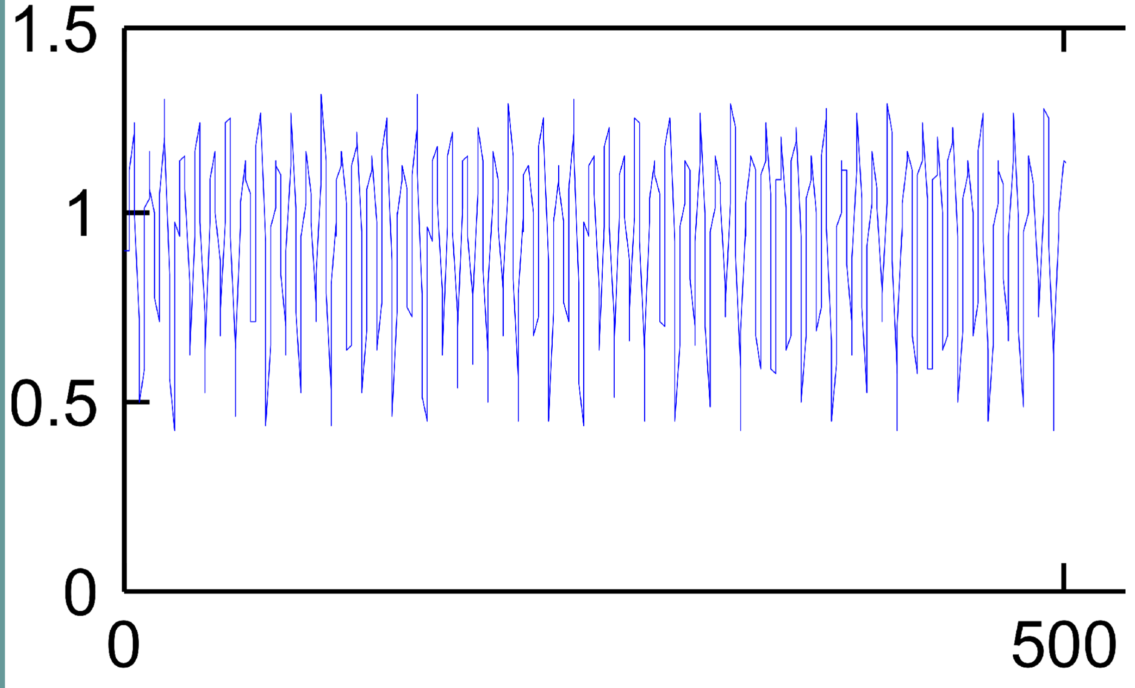
(b)

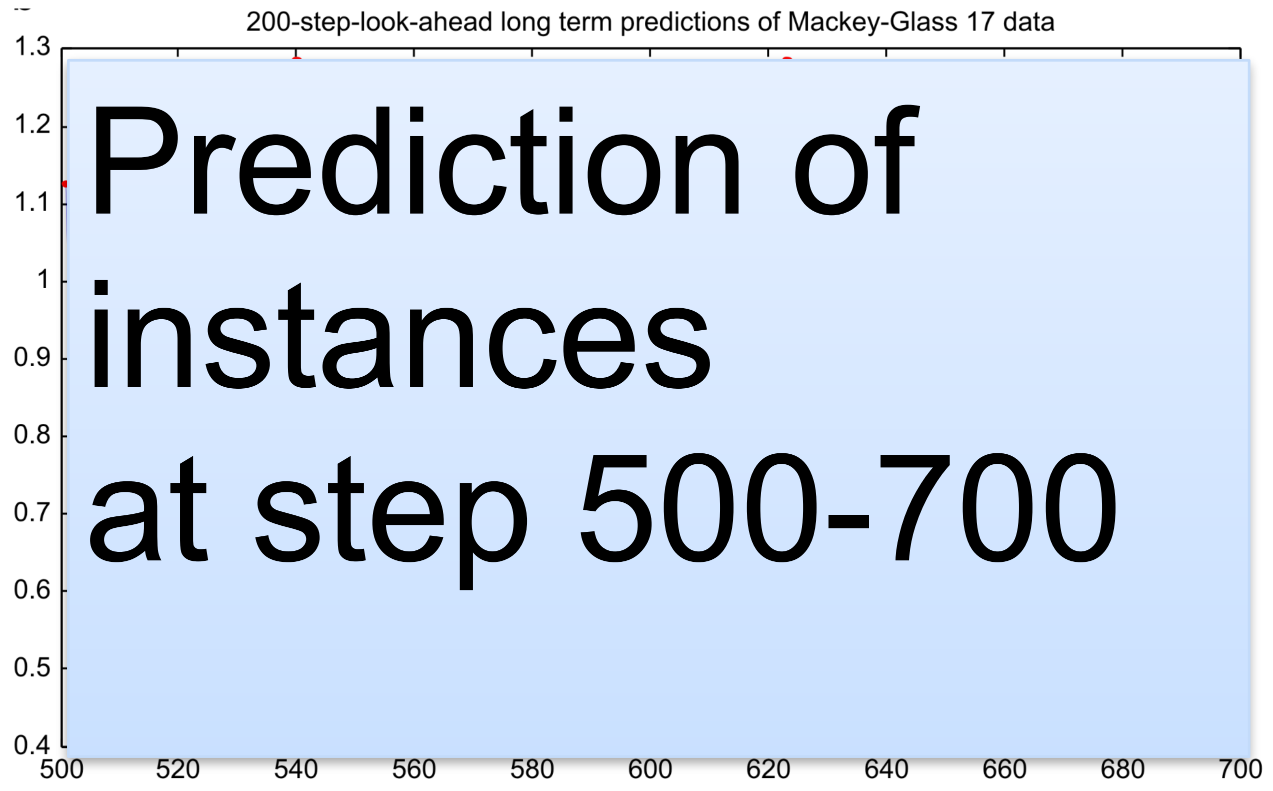
Figure 7

Chaotic differential function approximation

Data driven long-term prediction

- MG(Mackey–Glass) 17 generated by RK(Runge-Kutta) 4



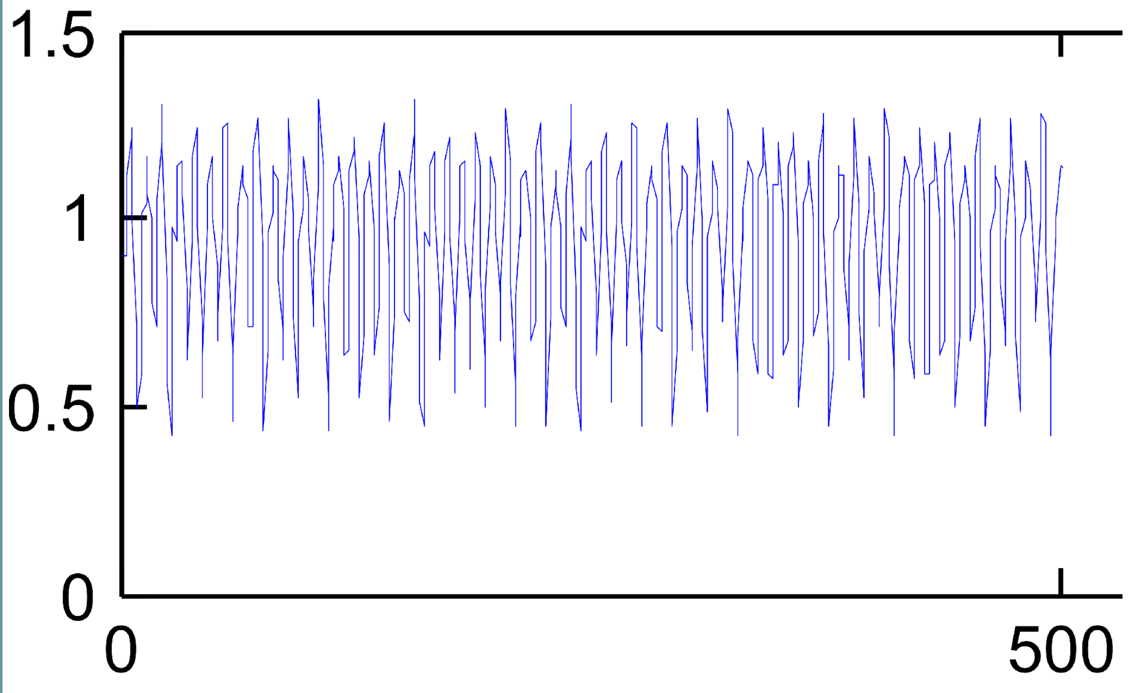


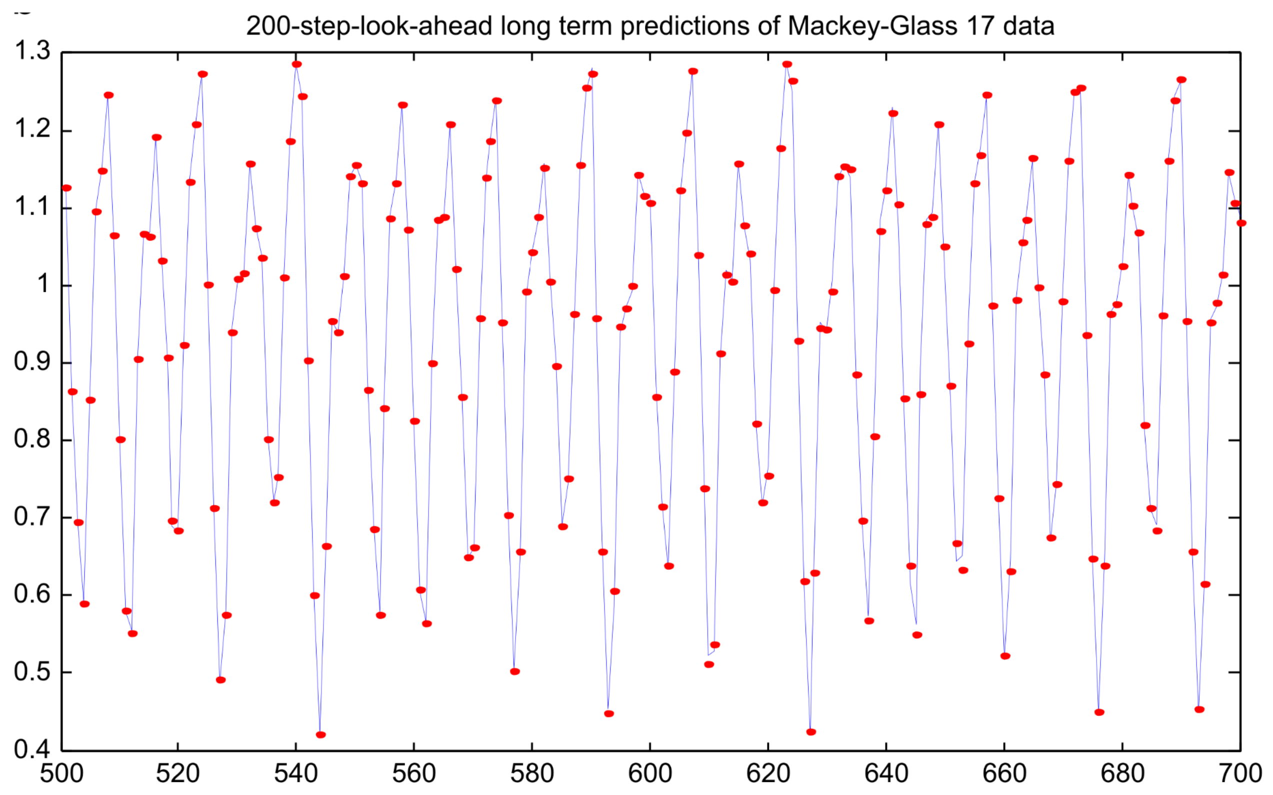
Prediction of instances at step 500-700

200-step-look-ahead prediction.

Data driven long-term prediction

- MG(Mackey–Glass) 17 generated by RK(Runge-Kutta) 4

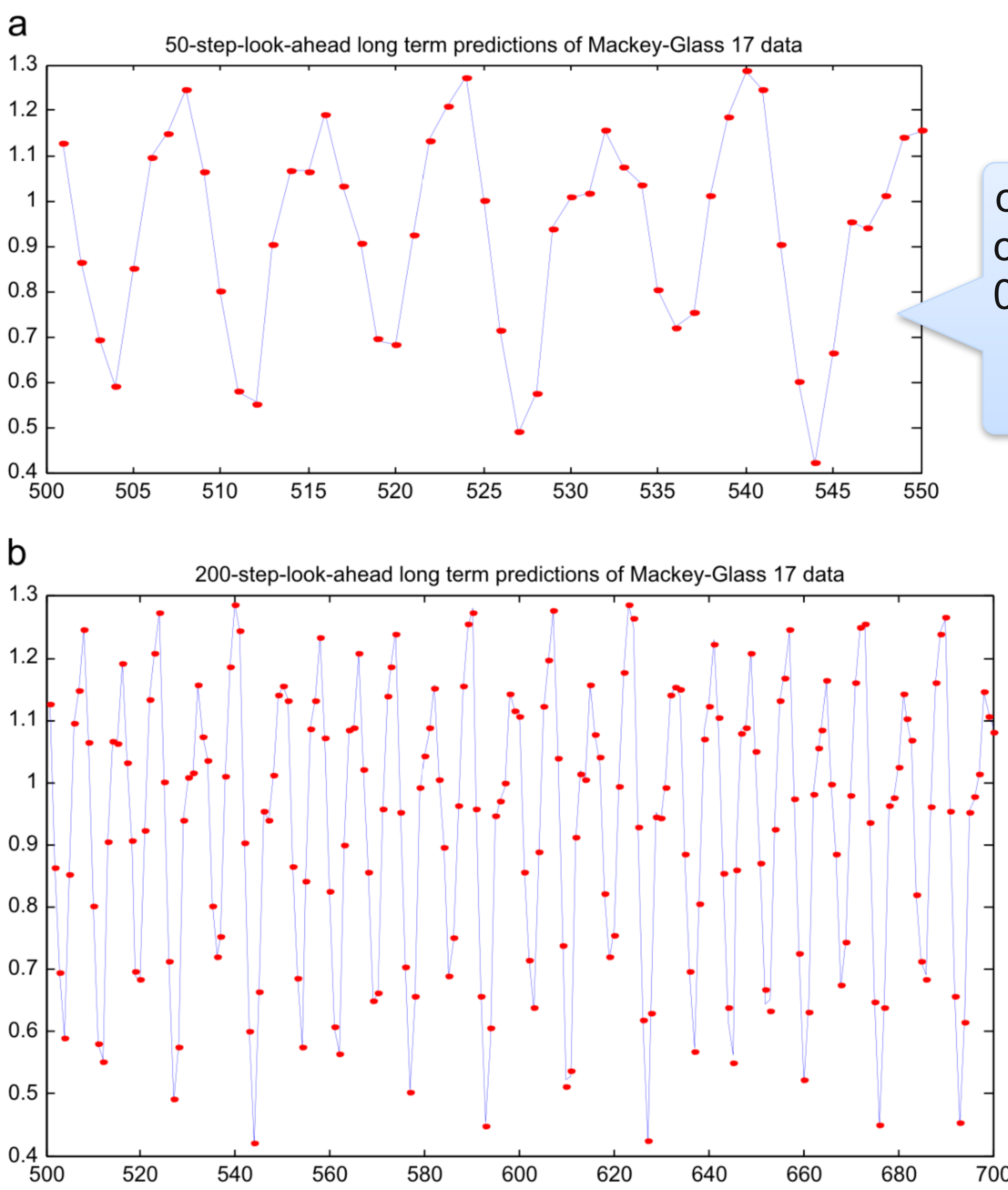




200-step-look-ahead prediction.

$$\frac{\partial x}{\partial t} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t),$$

$\tau = 17$, $a = 0.2$, $c = 10$ and $b = 0.1$



correlation coefficient 0.9999

correlation coefficient 0.9993

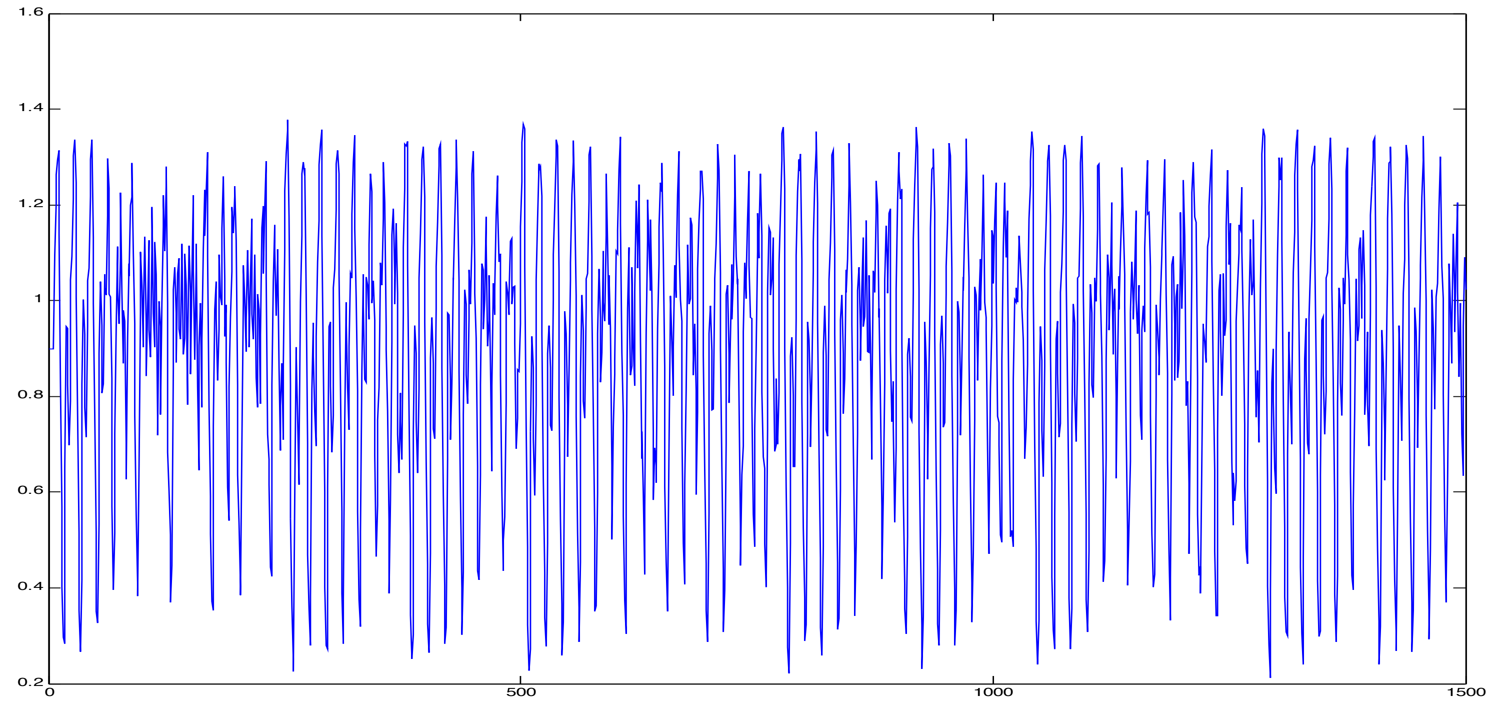
Fig. 10. η-step-look-ahead predictions of MG17 time series with η = 50 and η = 200 by annealed cooperative–competitive learning with K=2. (For interpretation of the

Mackey-Glass 30

$$\frac{\partial x}{\partial t} = \frac{a x(t-\tau)}{1 + x^c(t-\tau)} - b x(t),$$

$\tau = 30 \quad a = 0.2,\ c = 10$ and $b = 0.1$

Mackey-Glass 30 data



50-step-look-ahead long term predictions of Mackey-Glass 30 data

correlation coefficient 0.999

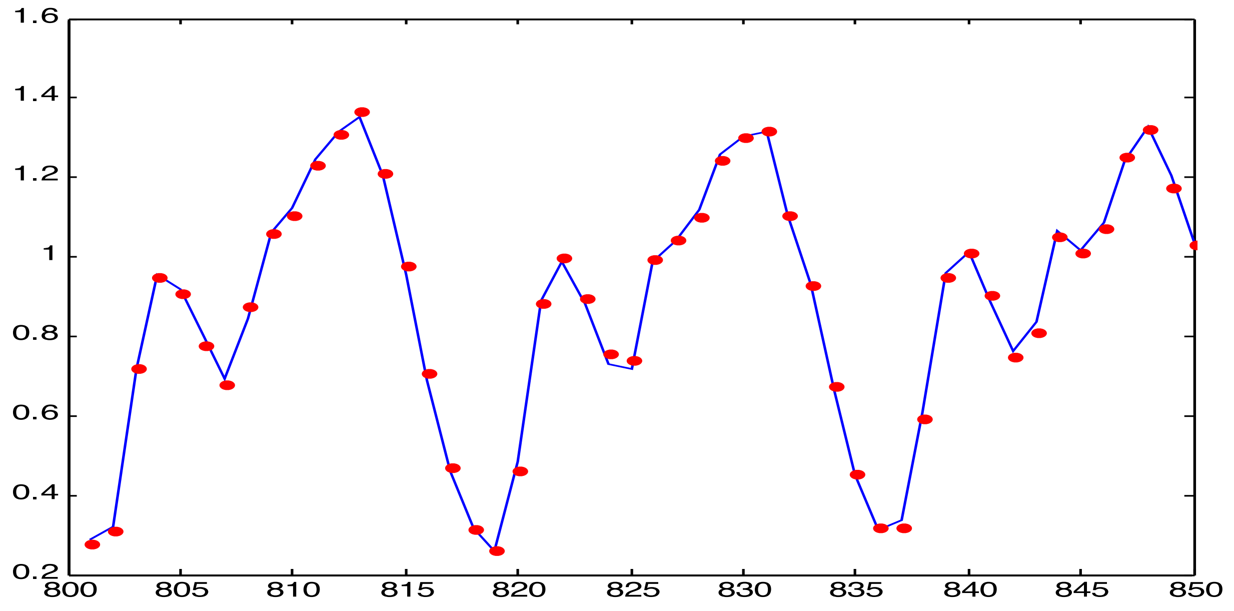


Figure 11

CDFA: Nonlinear delay differential equations

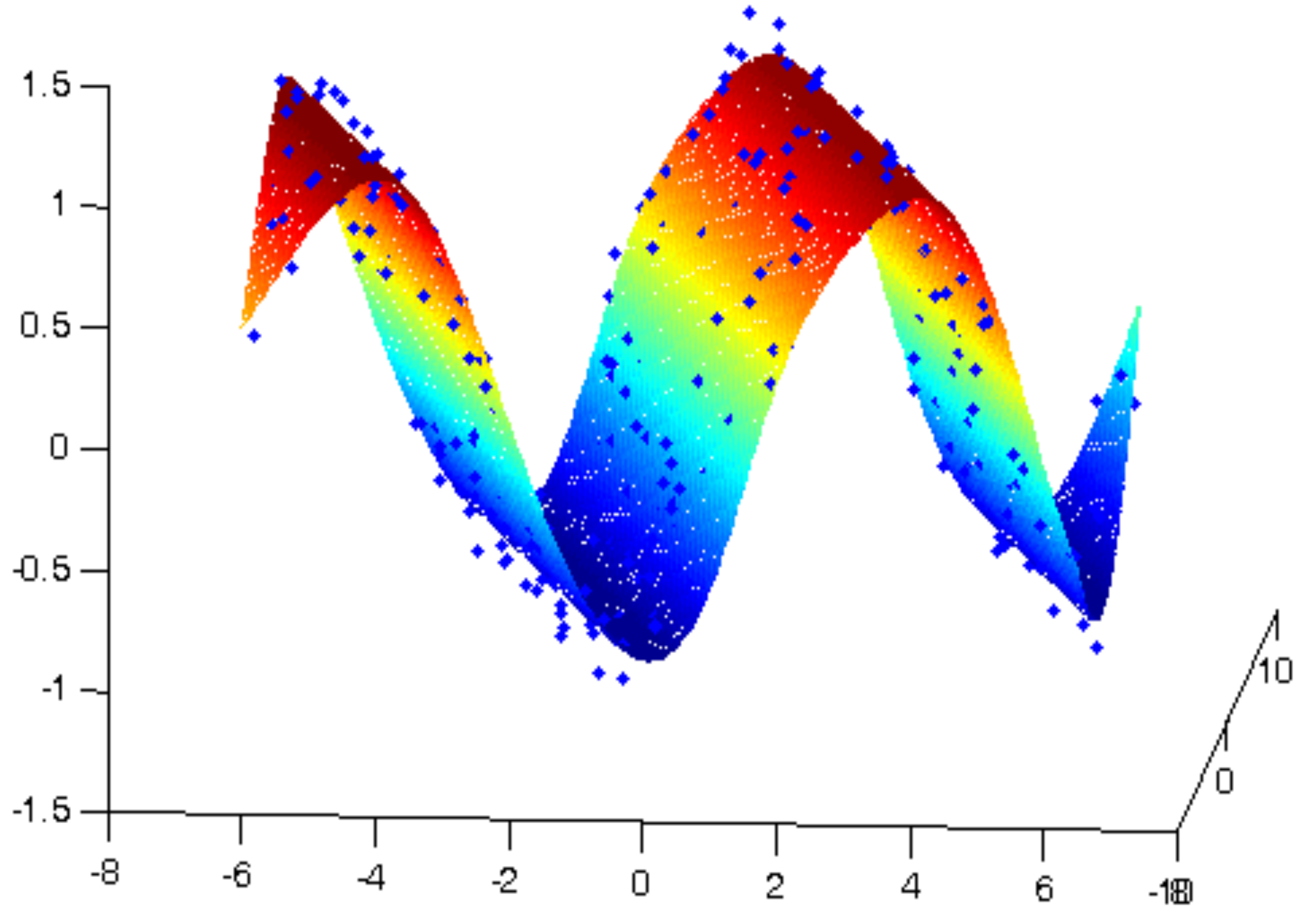
$$\frac{\partial x}{\partial t} = x(t-\tau) - x^3(1-\tau),$$

where the delav τ is set to 1.6.

J.C. Sprott, A simple chaotic delay differential equation, Phys. Lett. A 366 (2007) 397–402.

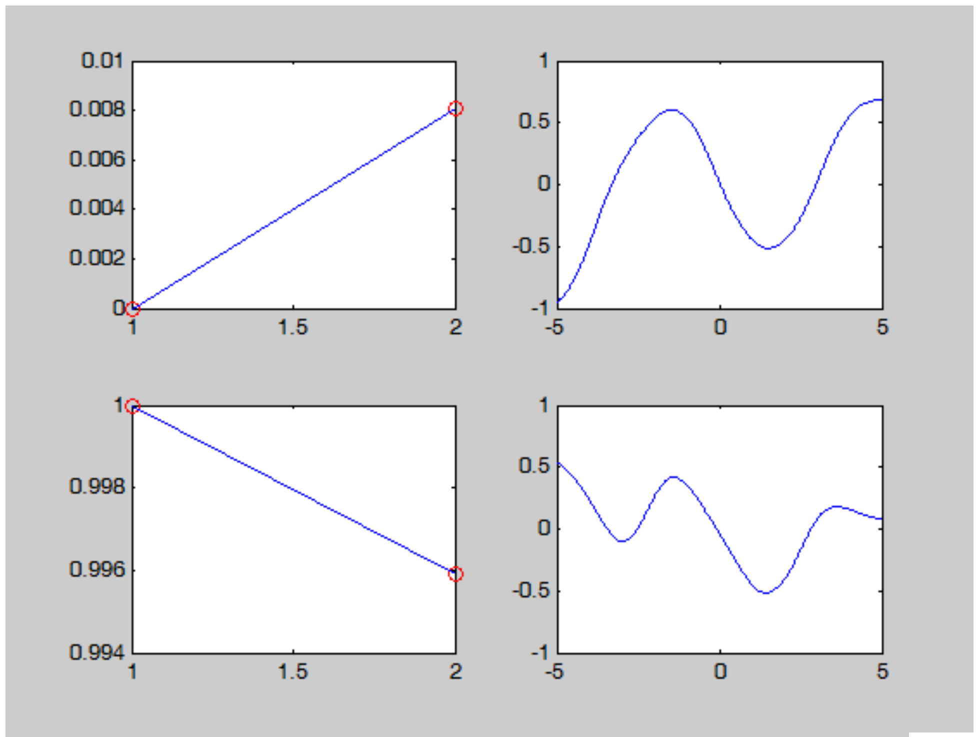
LM learning for MLP

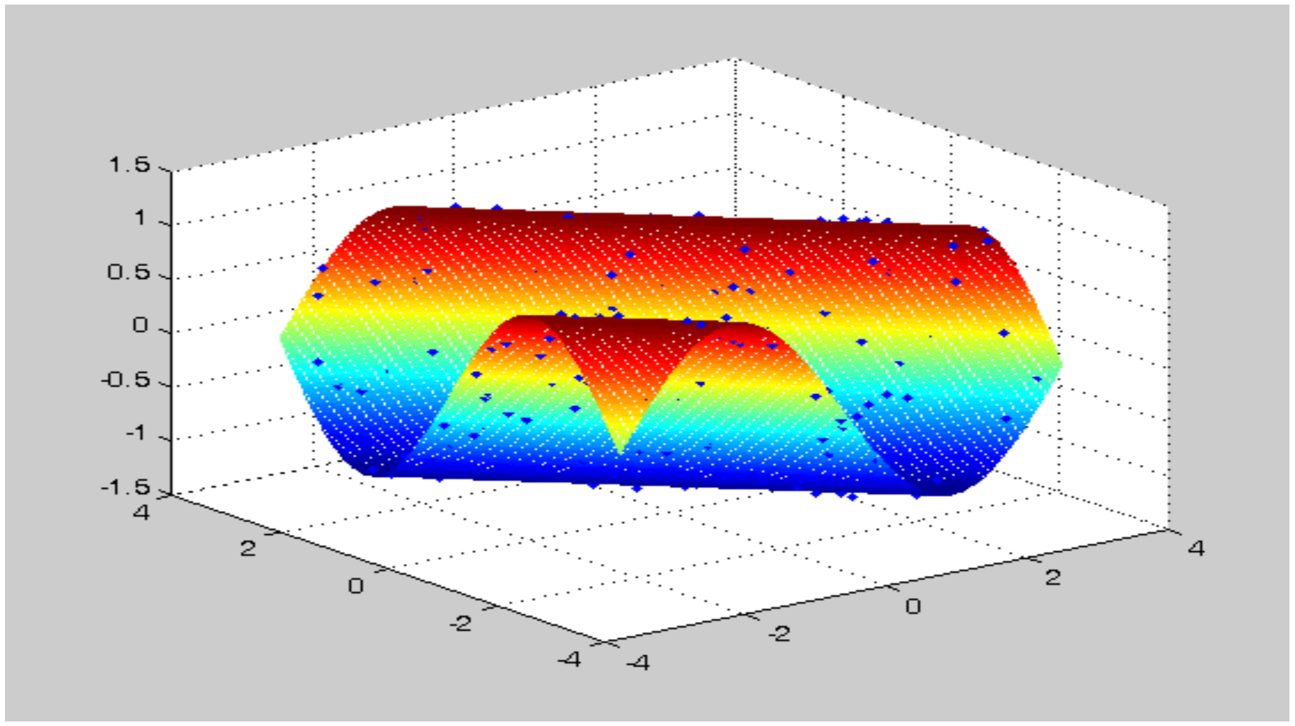
Two-dimensional Function Approximation

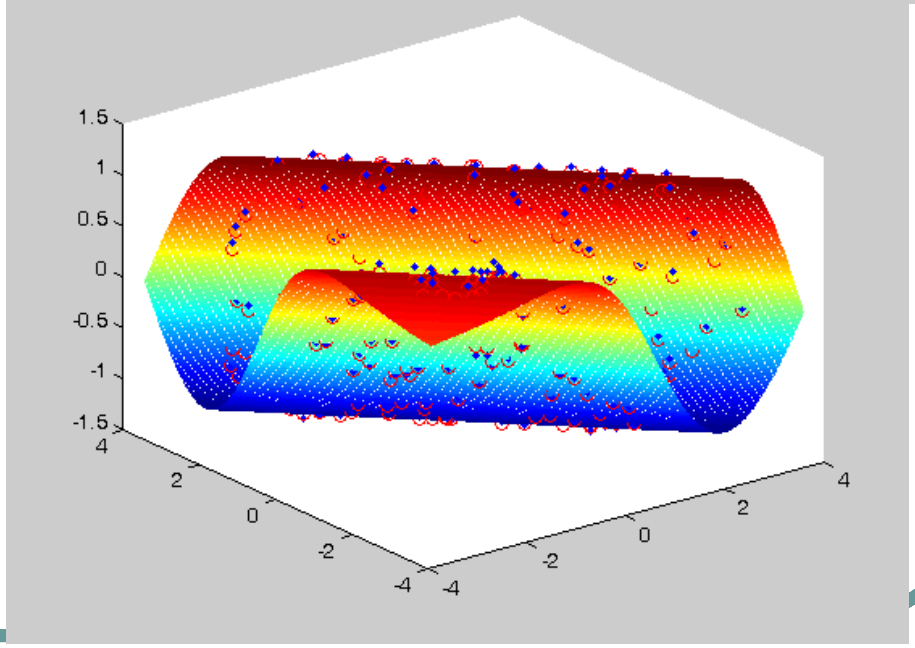


Learning MLPotts networks

Two-dimensional function approximation by learning MLPotts networks (Wu 2008)

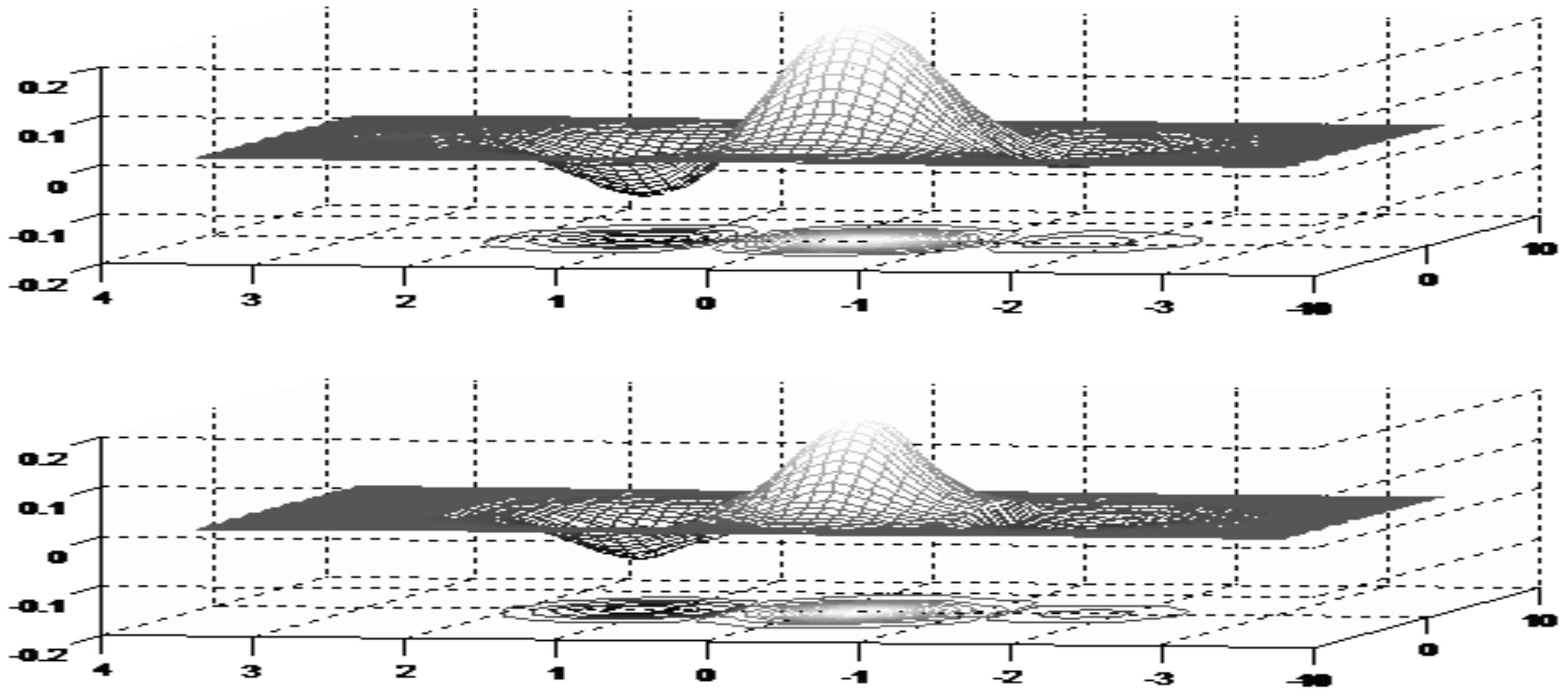






Approximating Gabor function

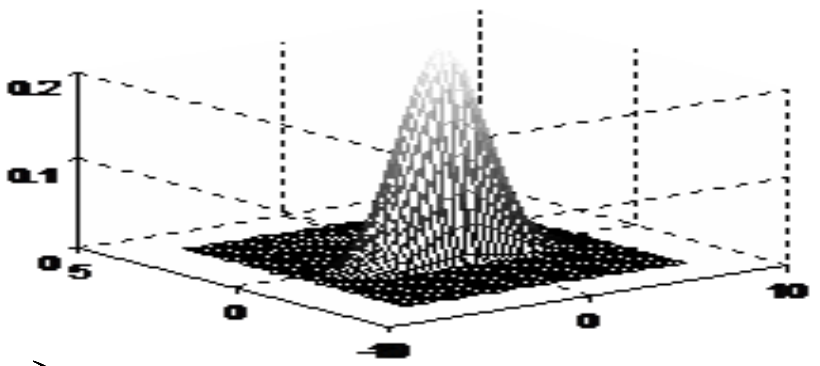
Learning generalized adalines (Wu et al 2006)



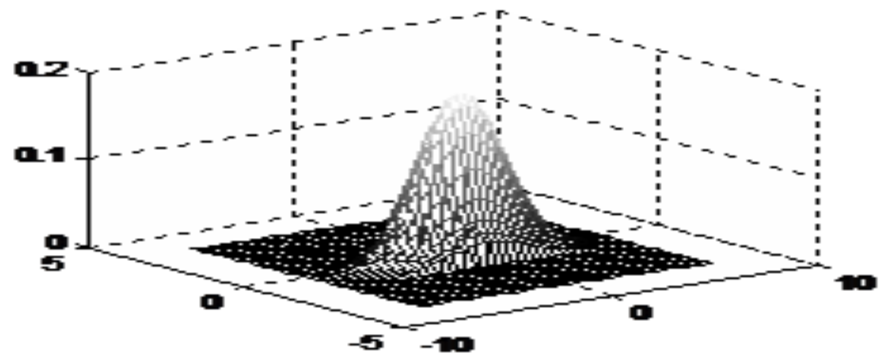
Approximating Gabor function

Learning gadaline networks

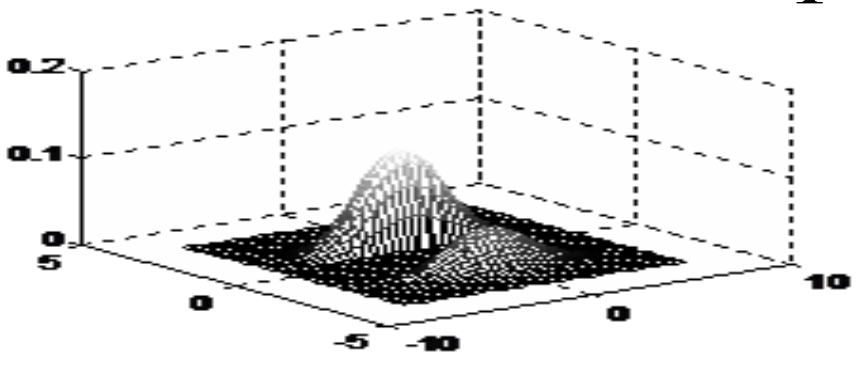
$y_7^+(\mathbf{z})$



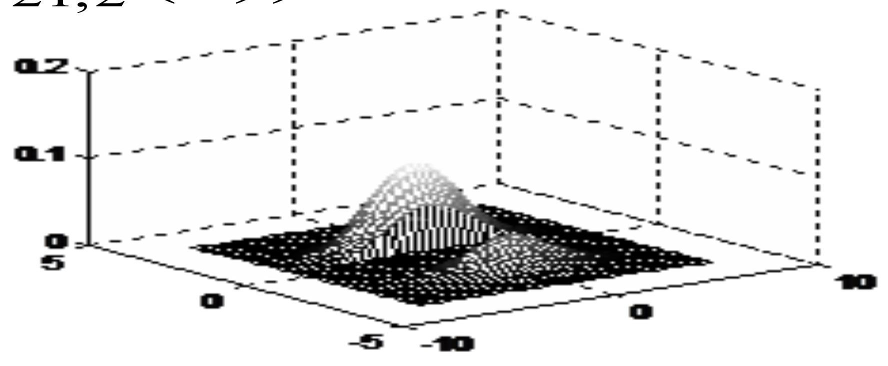
$\exp(G_{21,2}^+(\mathbf{z}))$



$y_7^-(\mathbf{z})$



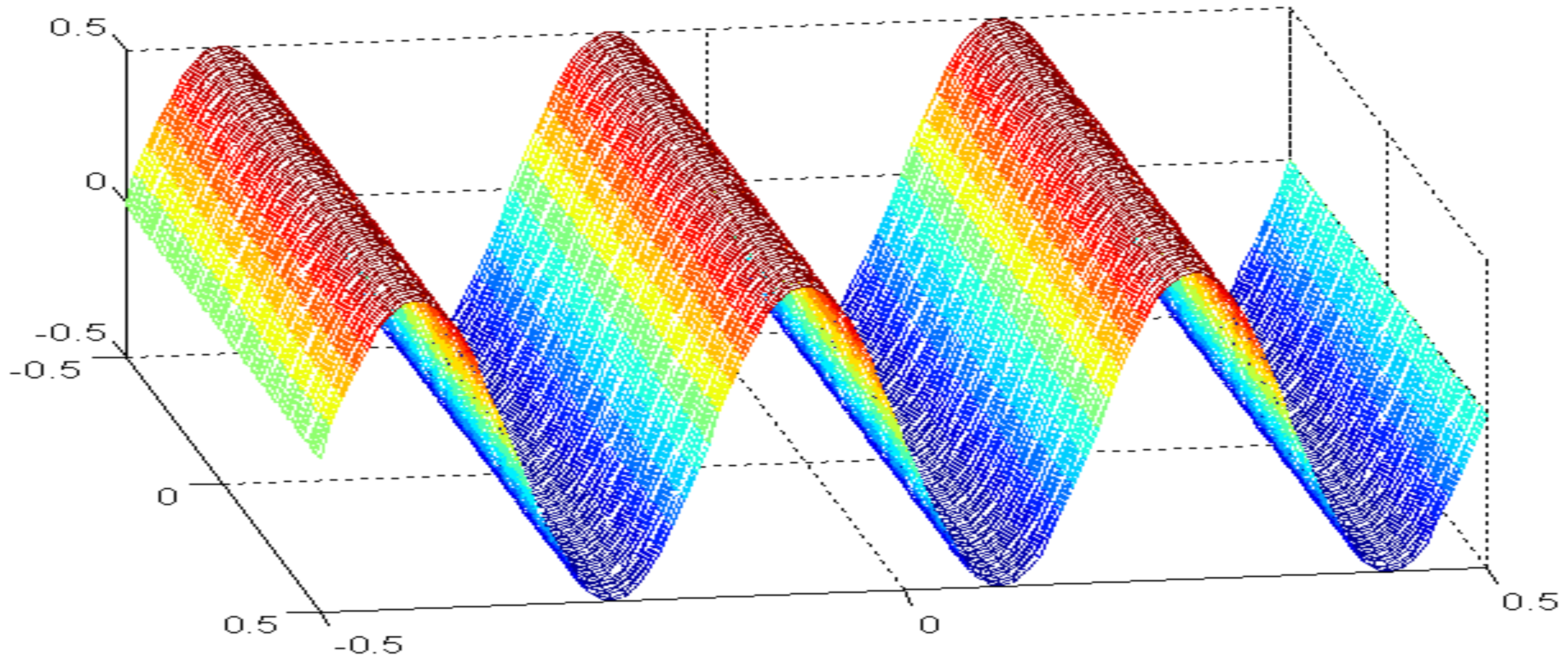
$\exp(G_{21,2}^-(\mathbf{z}))$



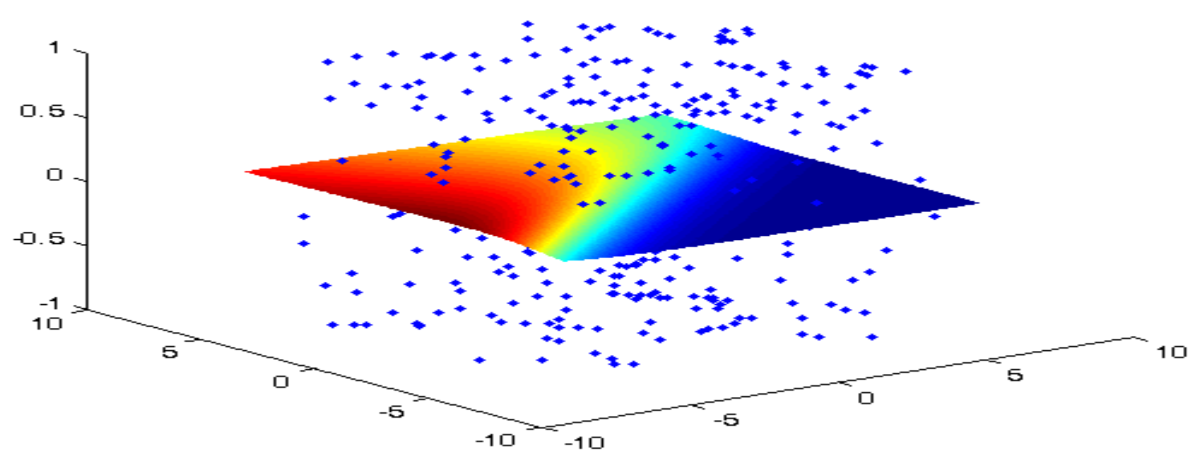
Sinusoidal function approximation

Learning gadaline networks (Wu et al 2006)

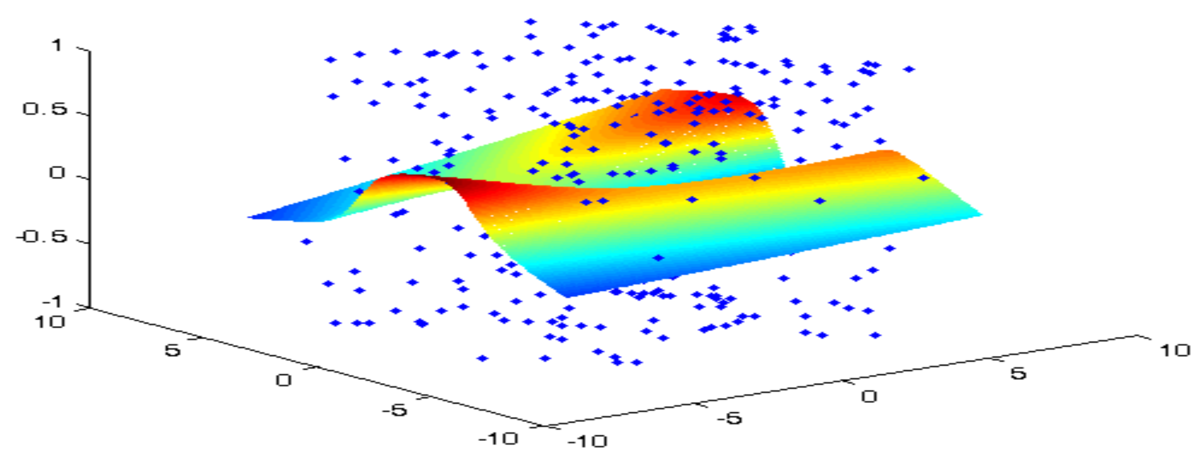
$y(\mathbf{z})$



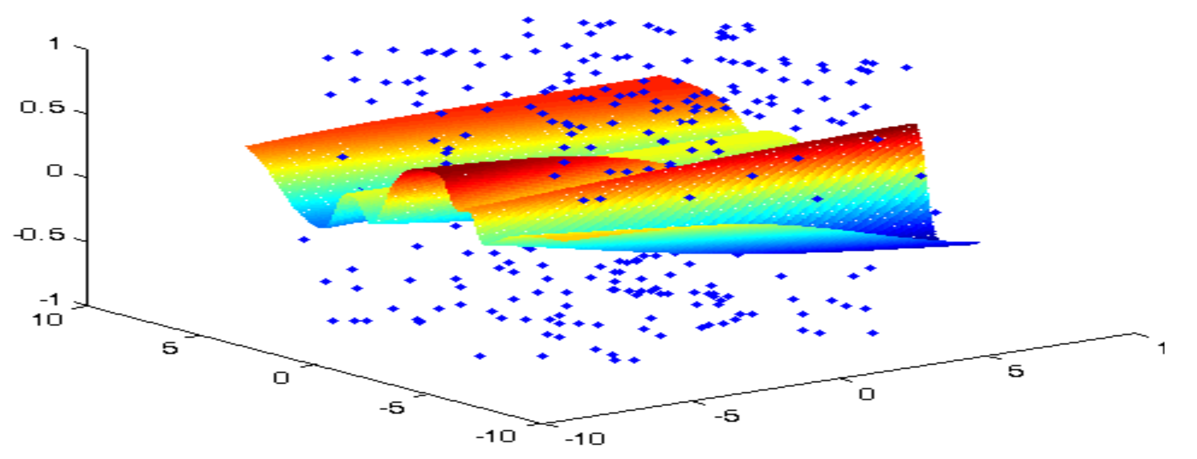
NRBF(3) by annealed FE



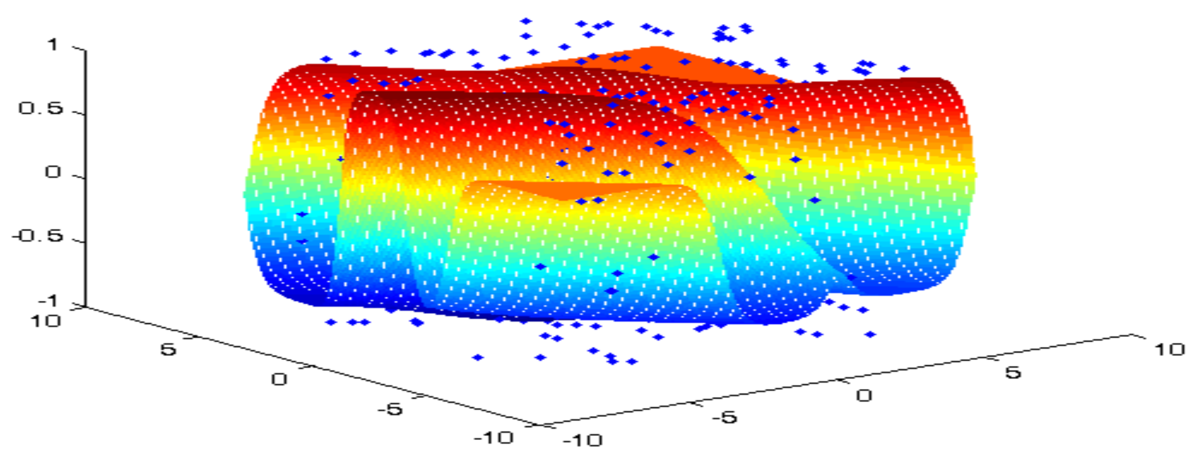
NRBF(6) by annealed FE



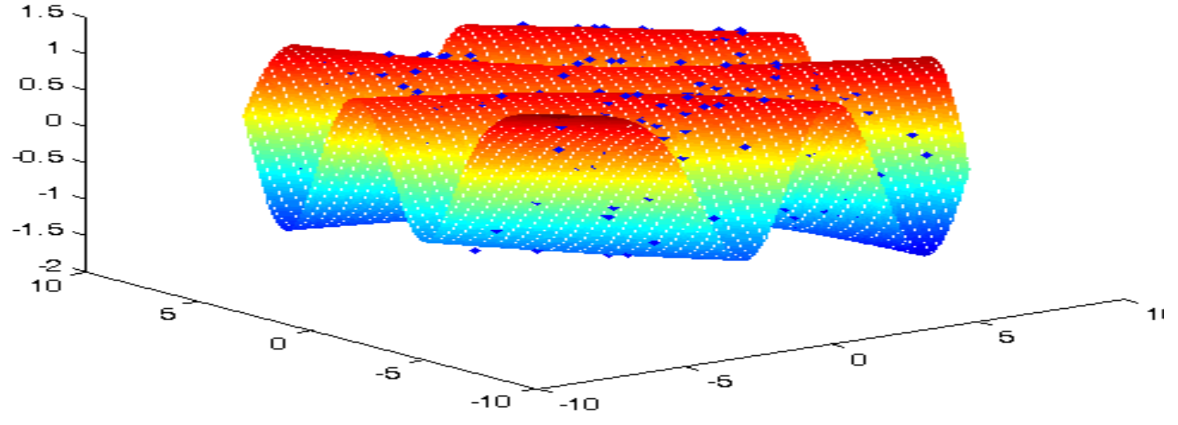
NRBF(9) by annealed FE



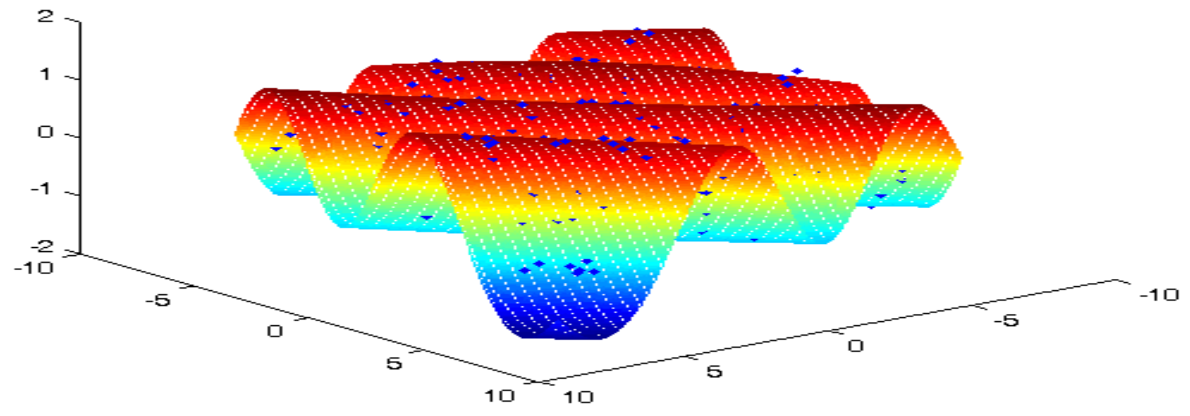
NRBF(12) by annealed FE

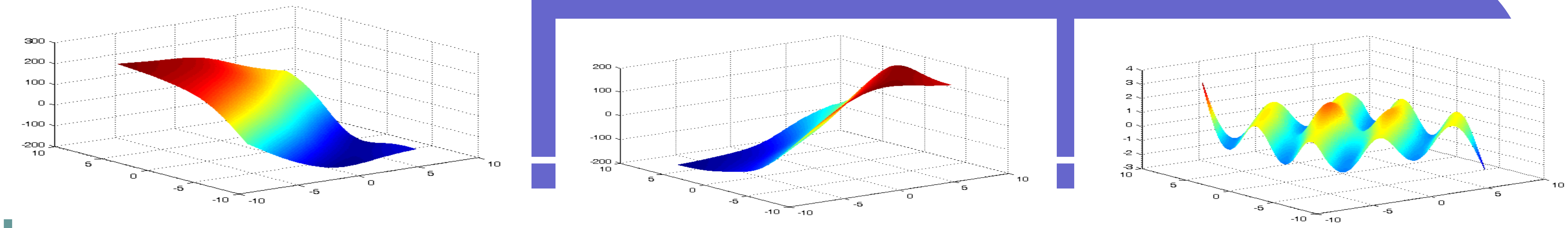


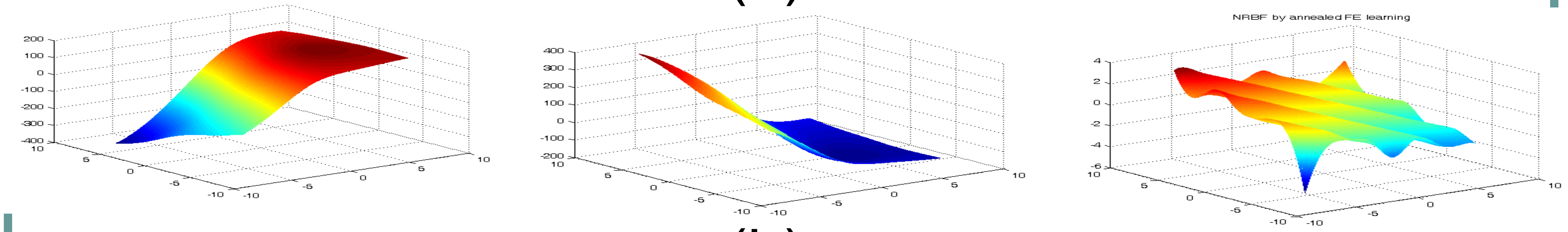
NRBF(15) by annealed FE



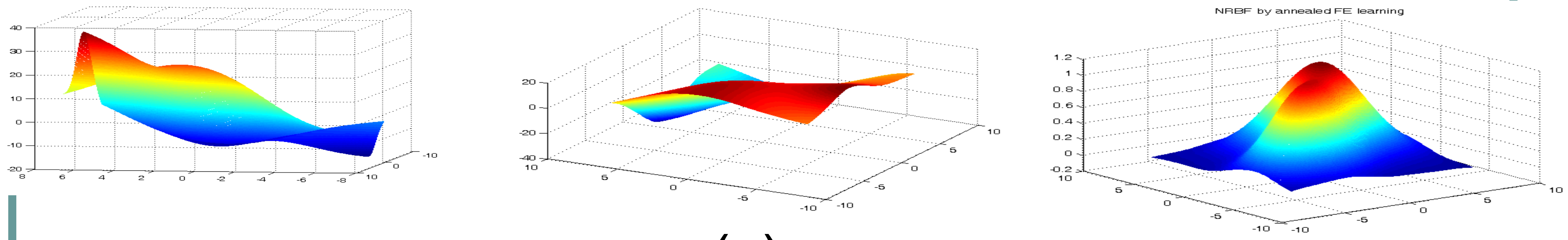
NRBF(18) by annealed FE




(a)



(b)



(c)

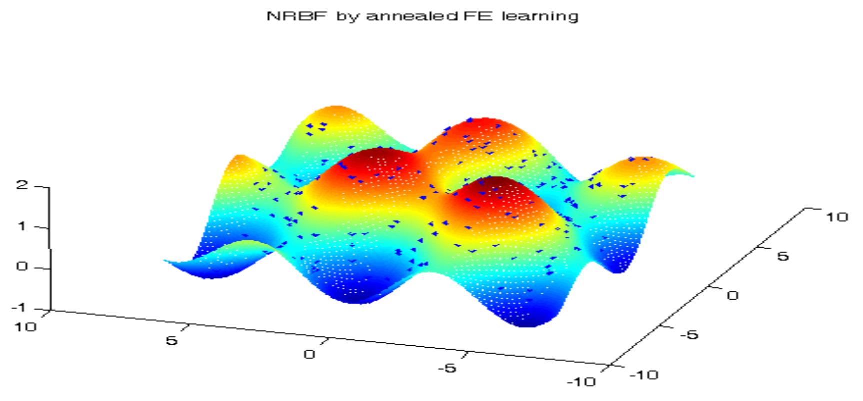
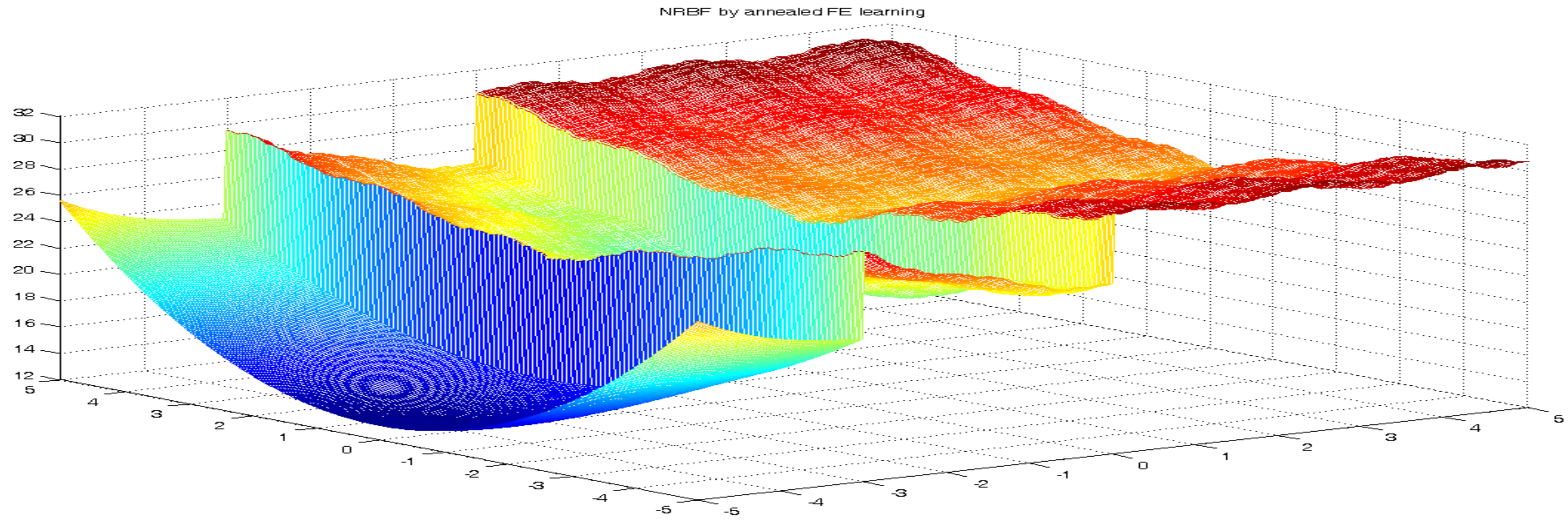
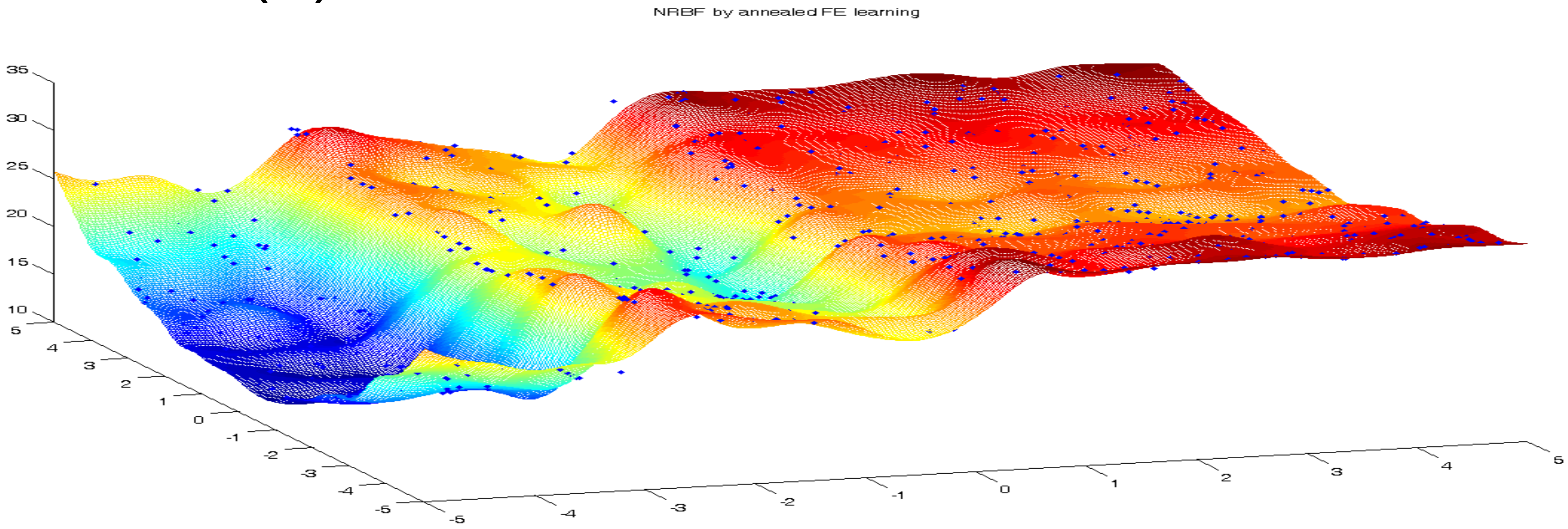


Figure 6

(d)


NRBF by annealed FE learning
32
30
28
26
24
22
20
18
16
14
12
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

(a)


NRBF by annealed FE learning
35
30
25
20
15
10
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

(b)

Figure 8

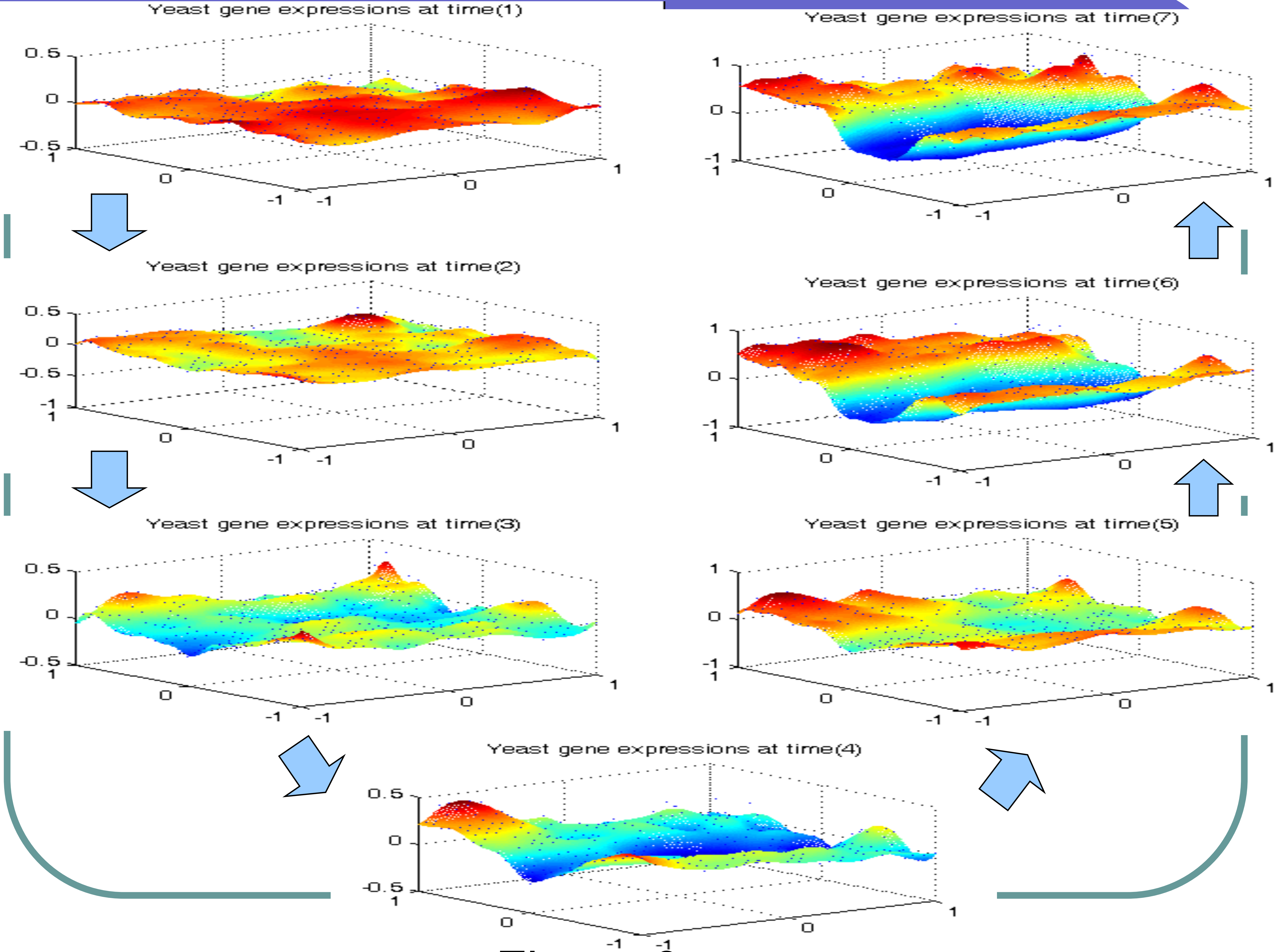

Yeast gene expressions at time(1)
Yeast gene expressions at time(2)
Yeast gene expressions at time(3)
Yeast gene expressions at time(4)
Yeast gene expressions at time(5)
Yeast gene expressions at time(6)
Yeast gene expressions at time(7)
0.5
0
-0.5
-1
1
0
-1

Figure 9

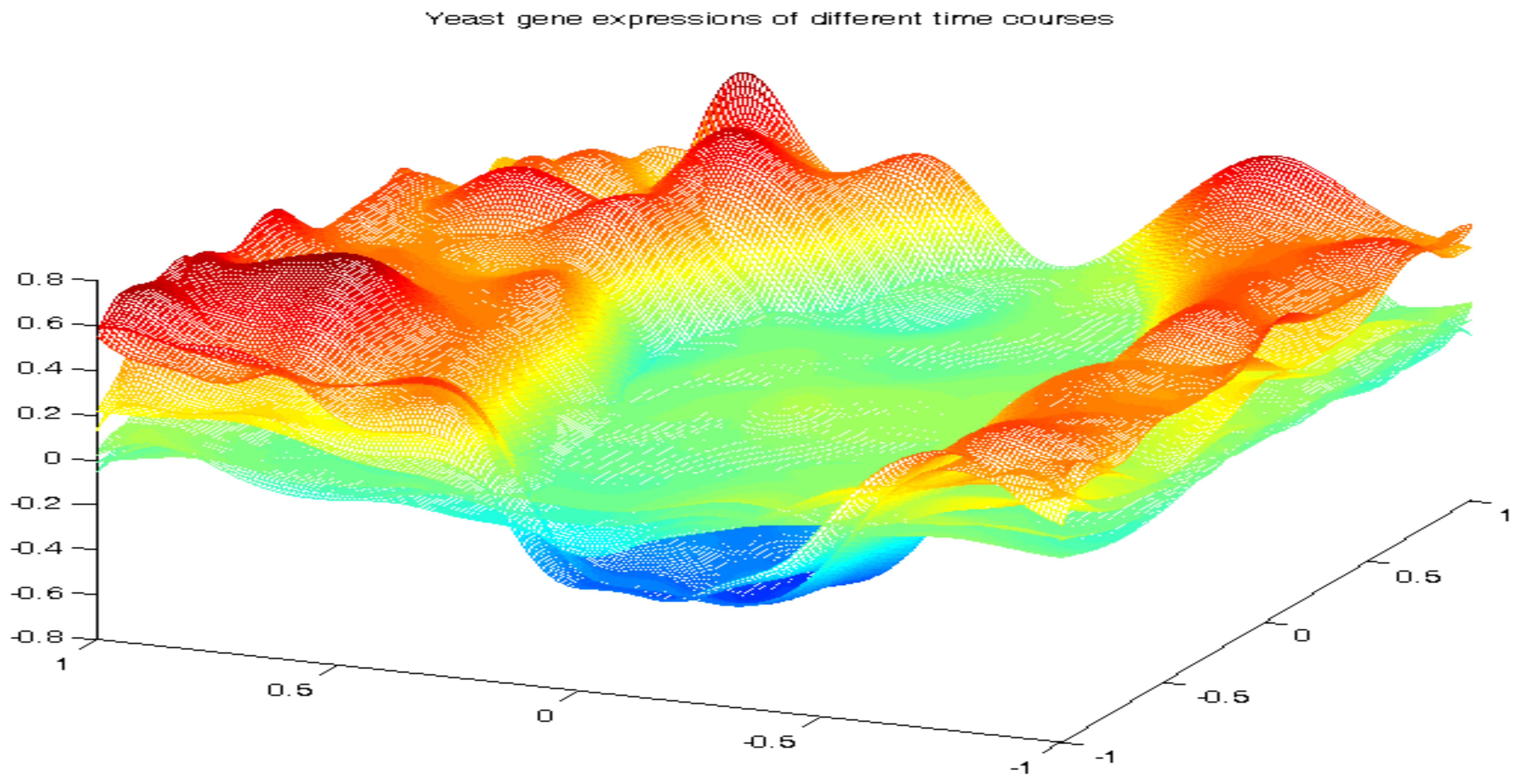
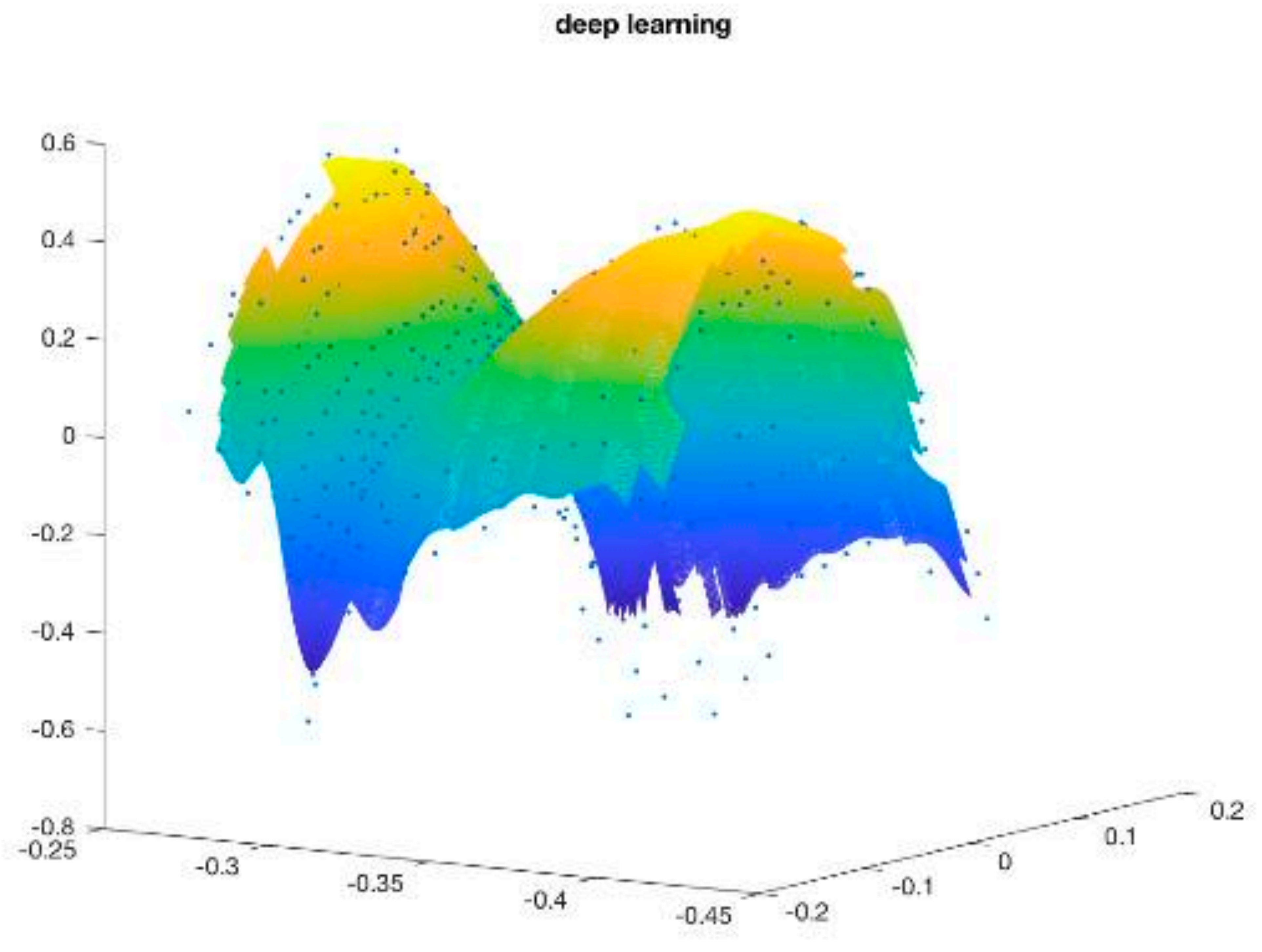

Yeast gene expressions of different time courses
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
1
0.5
0
-0.5
-1
1
0.5
0
-0.5
-1

Figure 10



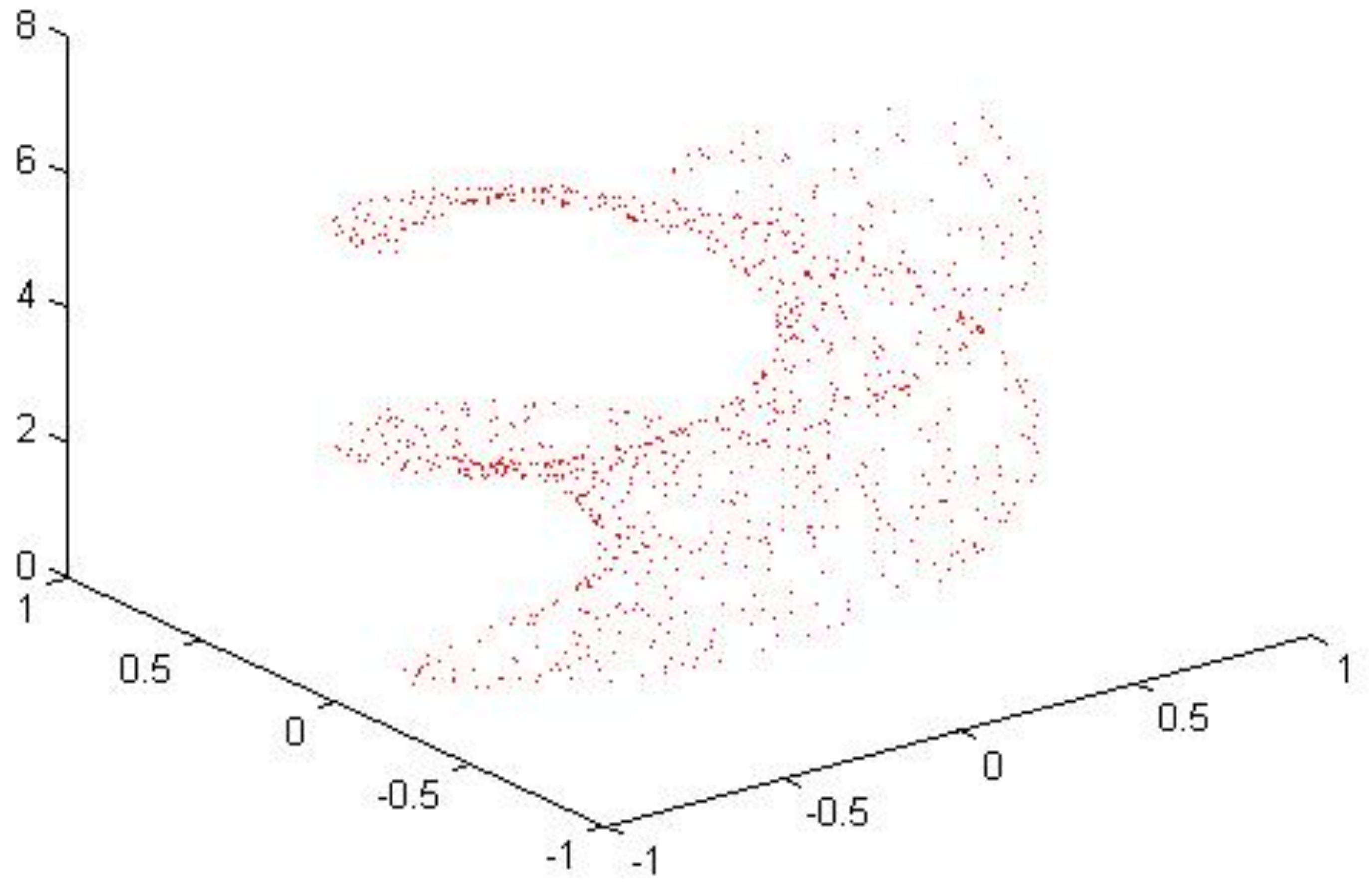
deep learning
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-0.25
-0.3
-0.35
-0.4
-0.45
-0.2
-0.1
0
0.1
0.2

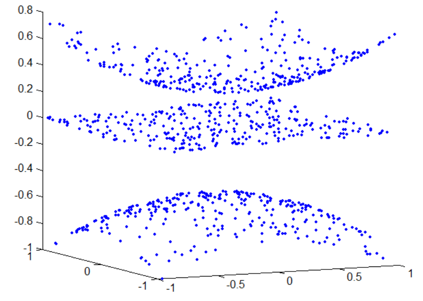
One-to-many function approximation to g_1

- P=5;k=M=30;N=100
  0

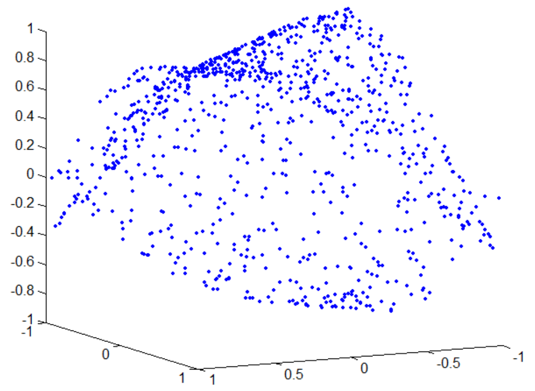
$$a_1 = g_1(q_1, q_2)$$

$$a_1 \in (0, 2\pi)$$

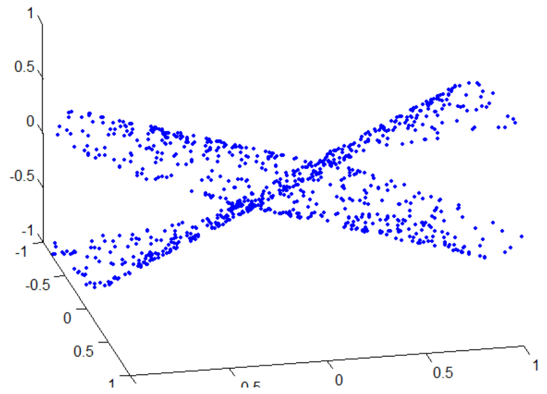




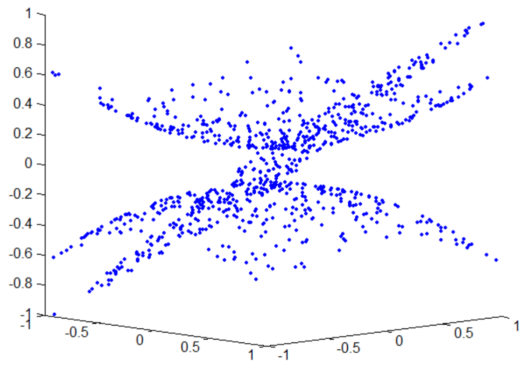
$$f_1(x_1,x_2) = x_1^2 + x_2^2 + 1$$
$$f_2(x_1,x_2) = -x_1^2 - x_2^2 - 2$$
$$f_3(x_1,x_2) = 0.1x_1 + 0.1x_2$$



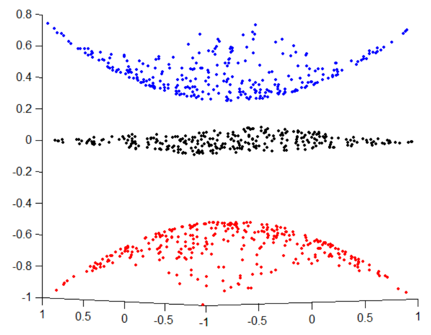
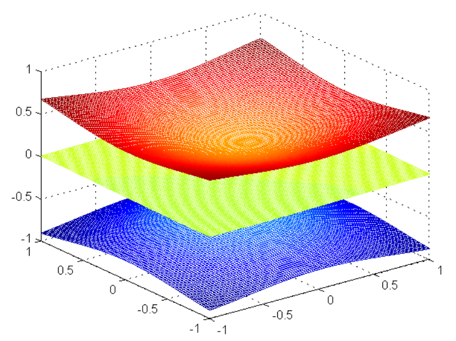
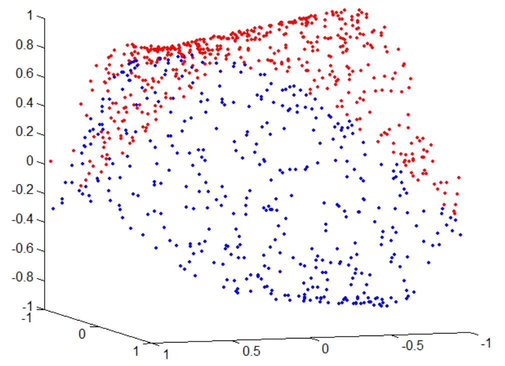
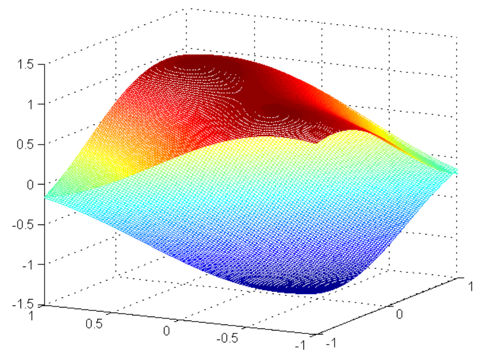
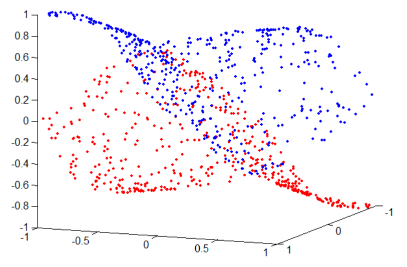
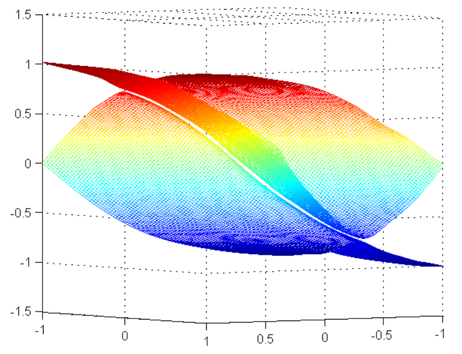
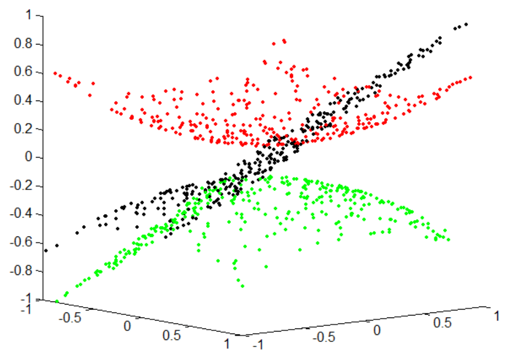
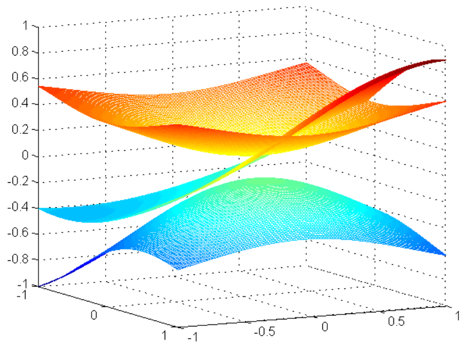
$$f_1(x_1,x_2) = \sin(x_1 + x_2)$$
$$f_2(x_1,x_2) = \cos(x_1 - x_2)$$

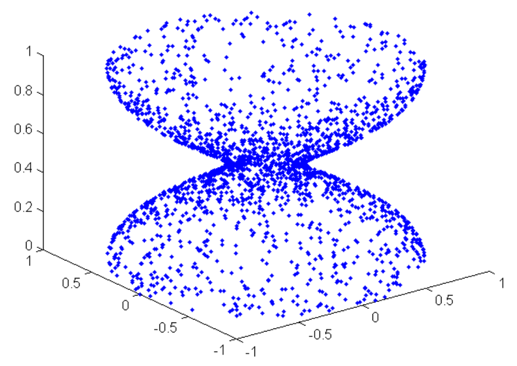
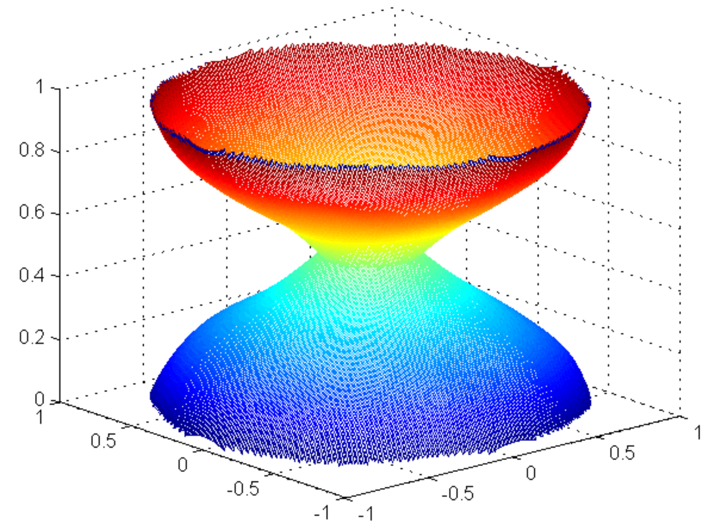
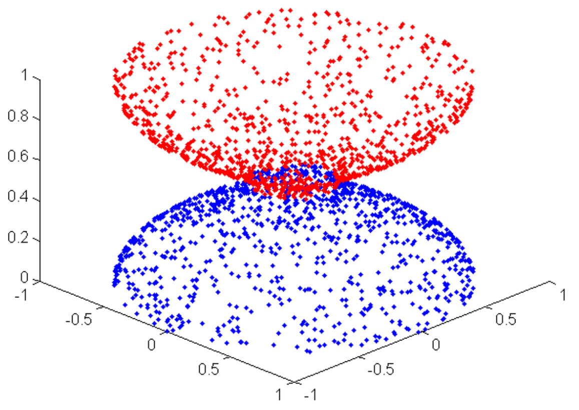
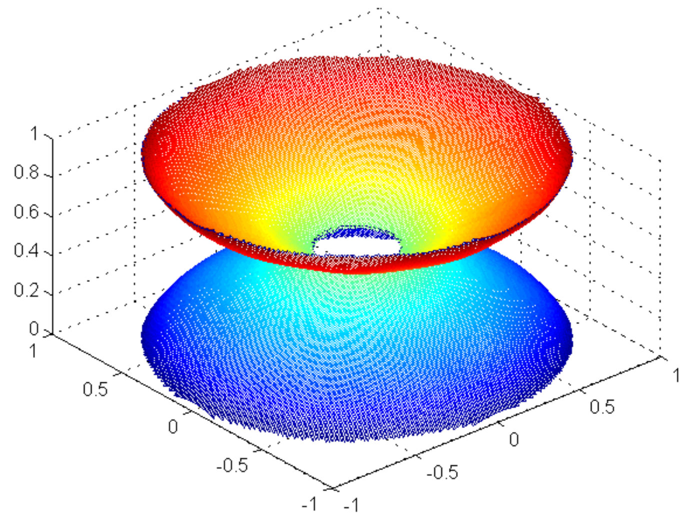


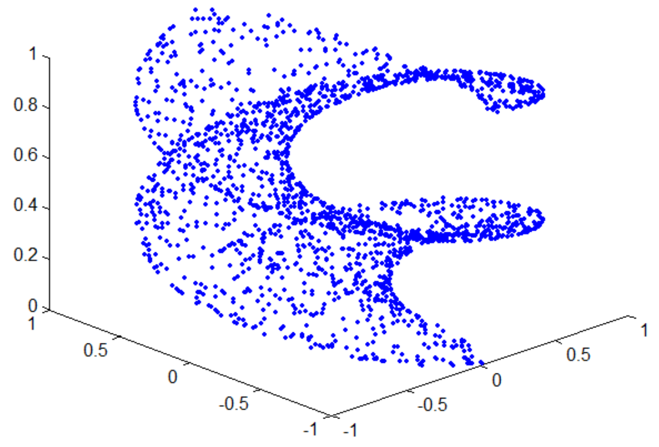
$$f_1(x_1,x_2) = \tanh(x_1 + x_2)$$
$$f_2(x_1,x_2) = \tanh(x_1 - x_2)$$

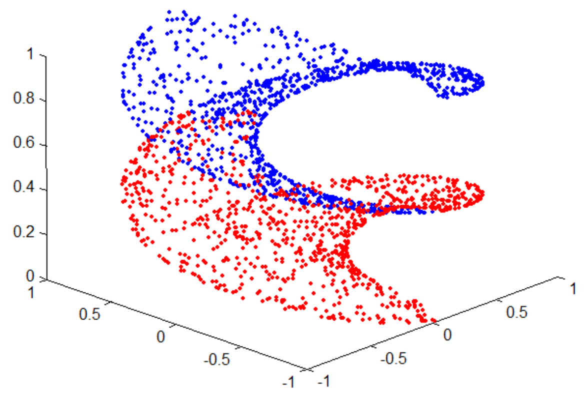


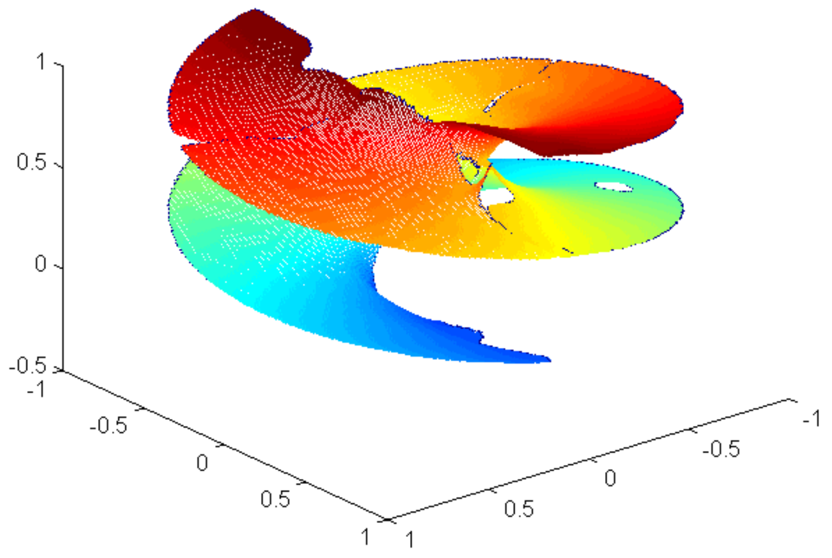
$$f_1(x_1,x_2) = x_1^2 + x_2^2 + 0.5$$
$$f_2(x_1,x_2) = -x_1^2 - x_2^2 - 0.5$$
$$f_3(x_1,x_2) = 2x_1 + 2x_2$$

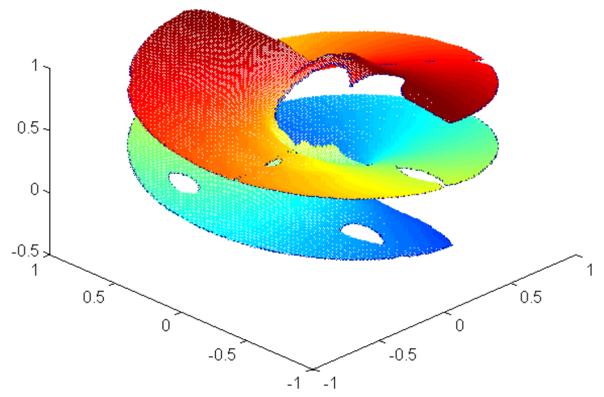







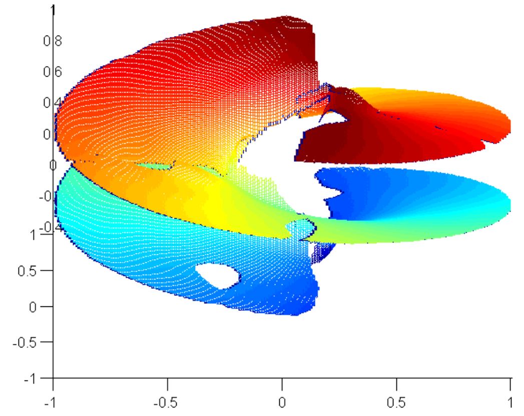




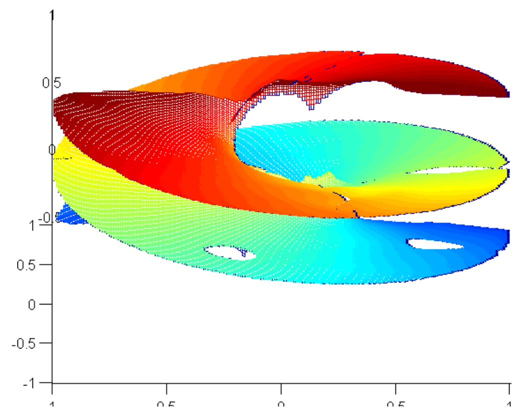
0°



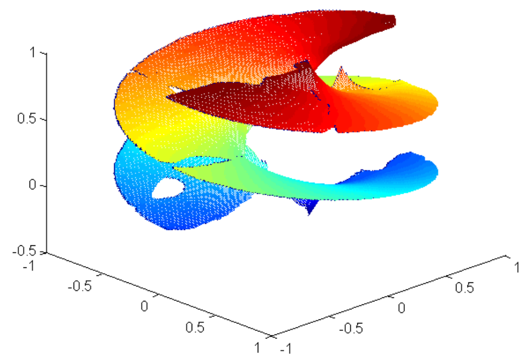
315°



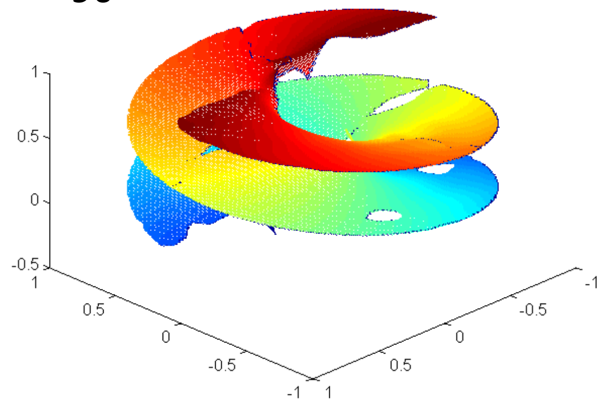
45°



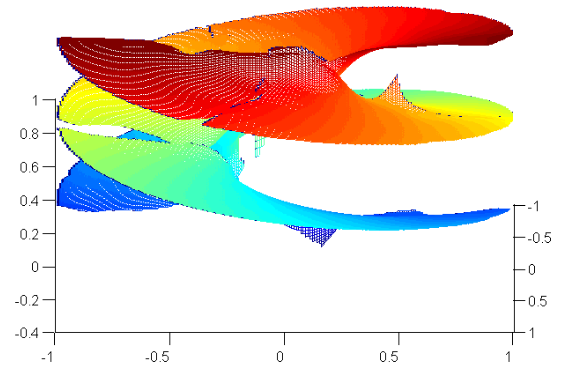
270°



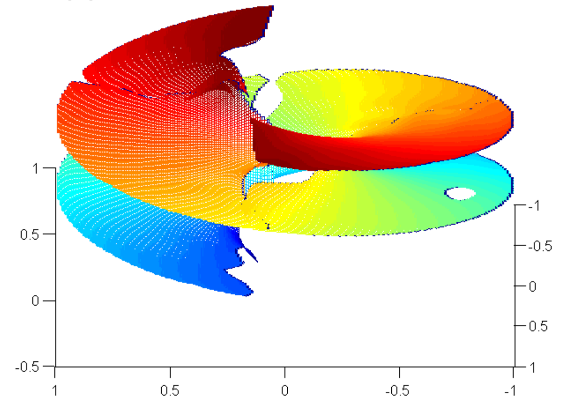
90°



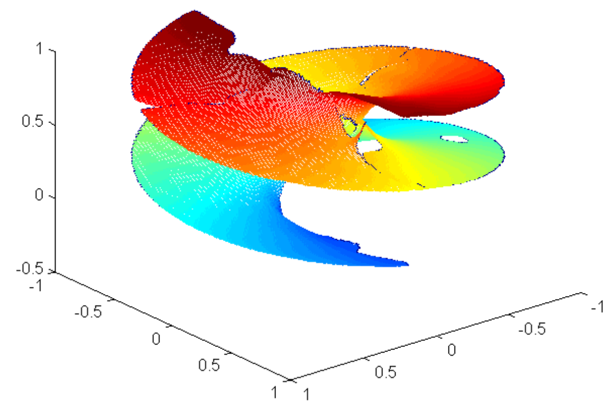
225°

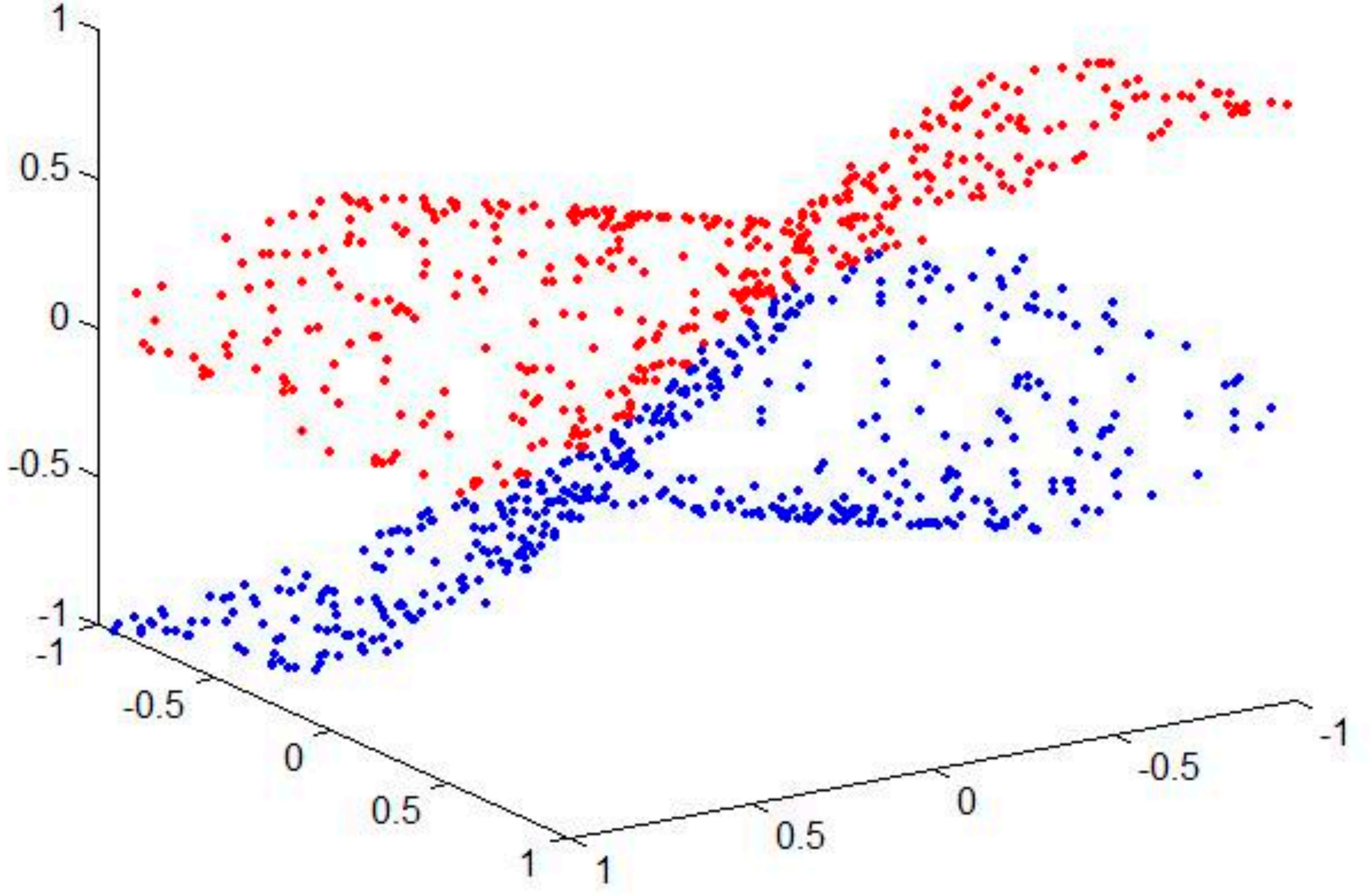


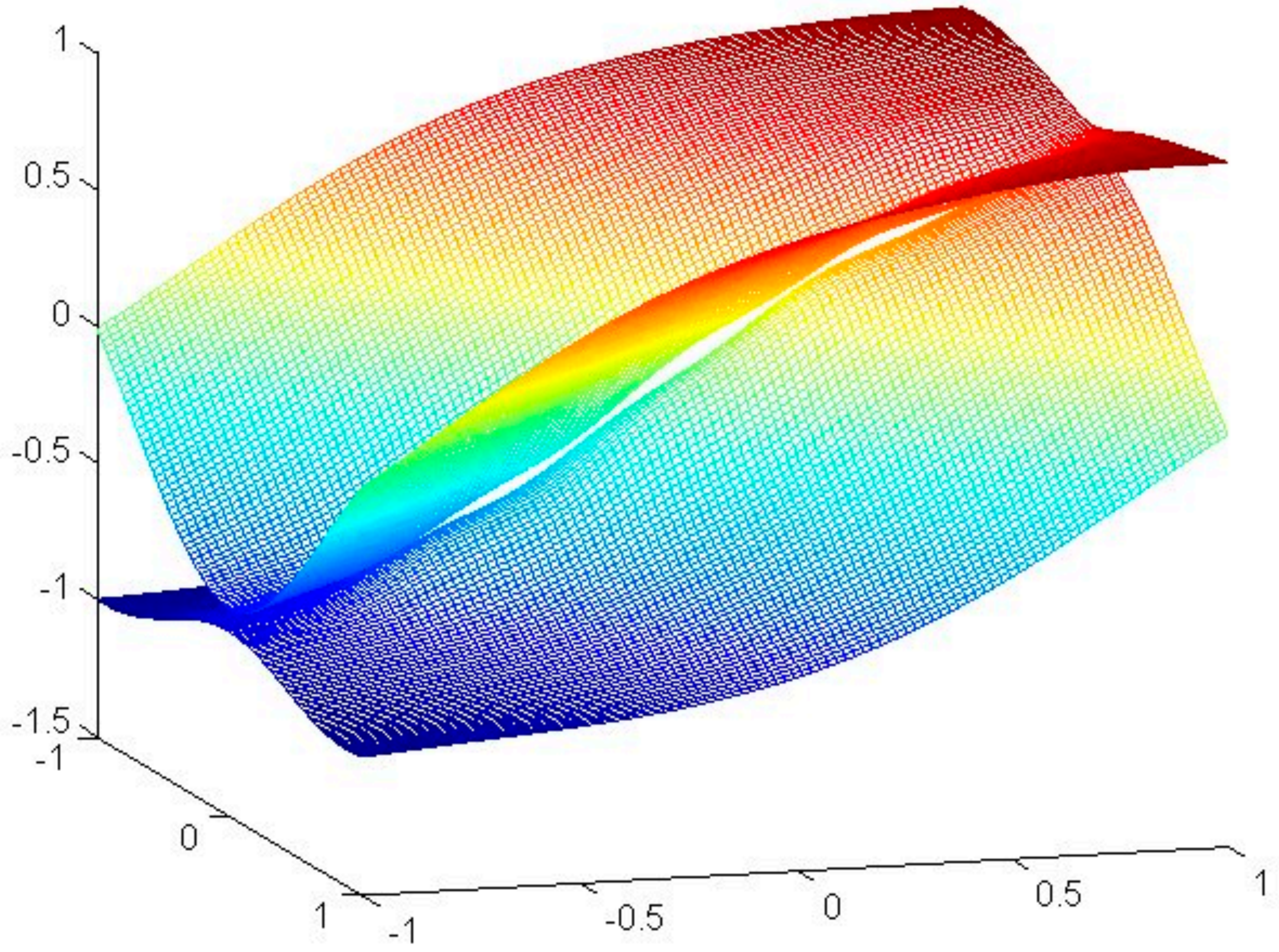
135°



180°



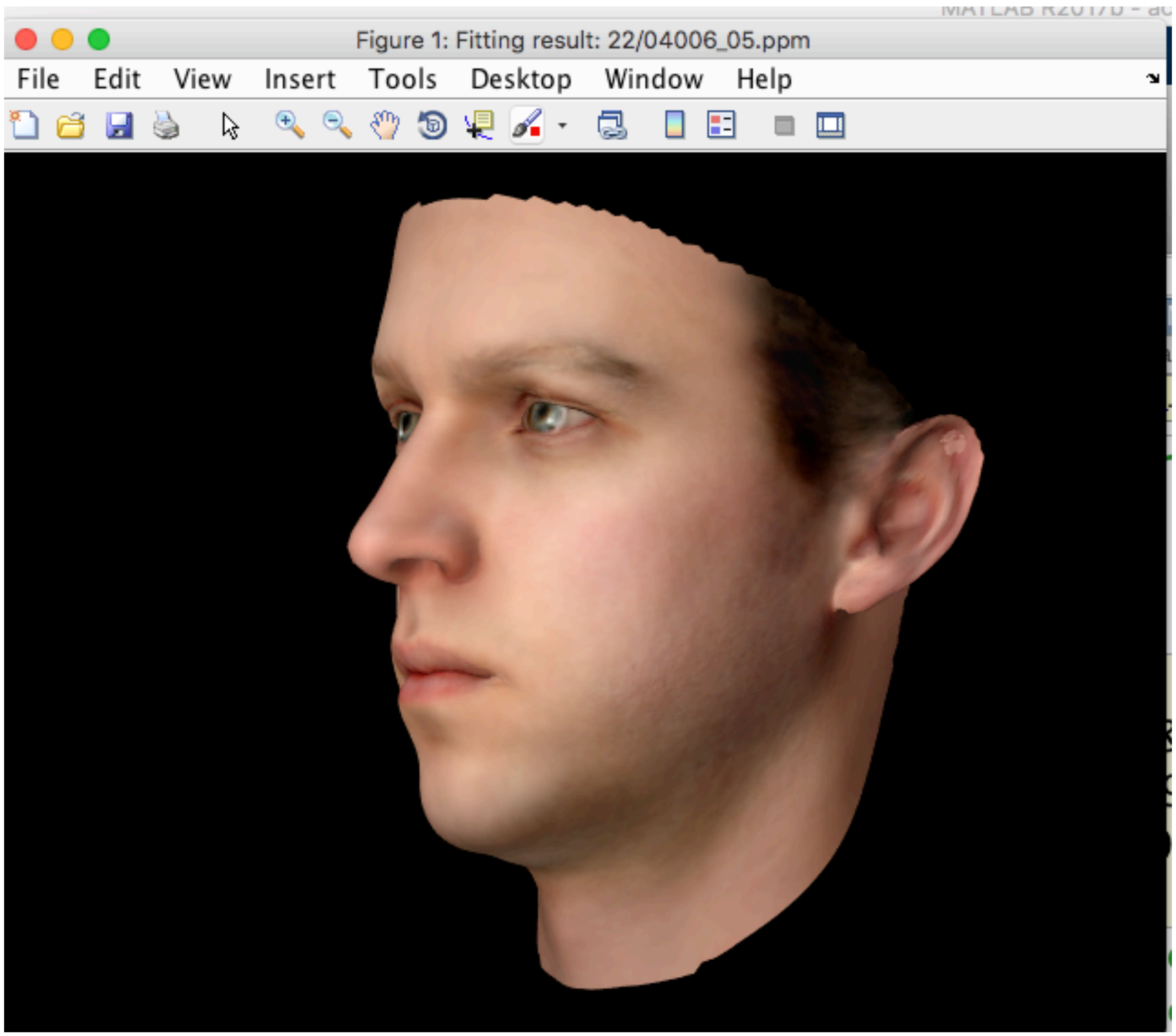

1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1
0.5
0
-0.5
-1


1
0.5
0
-0.5
-1
-1.5
-1
0
1
-1
-0.5
0
0.5
1

3D image recognition & iPhone AI Apps

Mathematical computation for 3D face reconstruction

Matlab Software


MATLAB R2017b - ac
Figure 1: Fitting result: 22/04006_05.ppm
File
Edit
View
Insert
Tools
Desktop
Window
Help

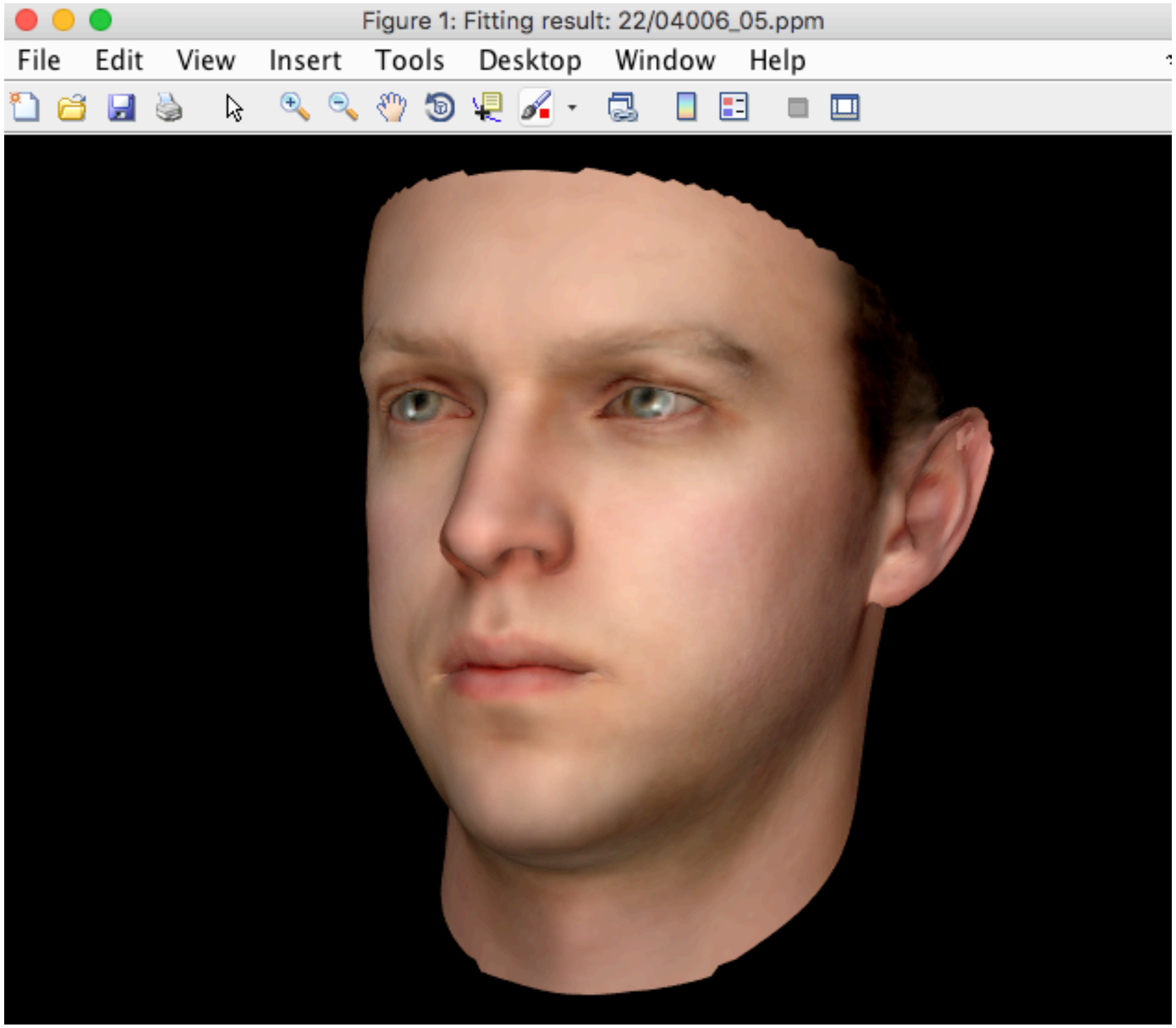


Figure 1: Fitting result: 22/04006_05.ppm
File
Edit
View
Insert
Tools
Desktop
Window
Help

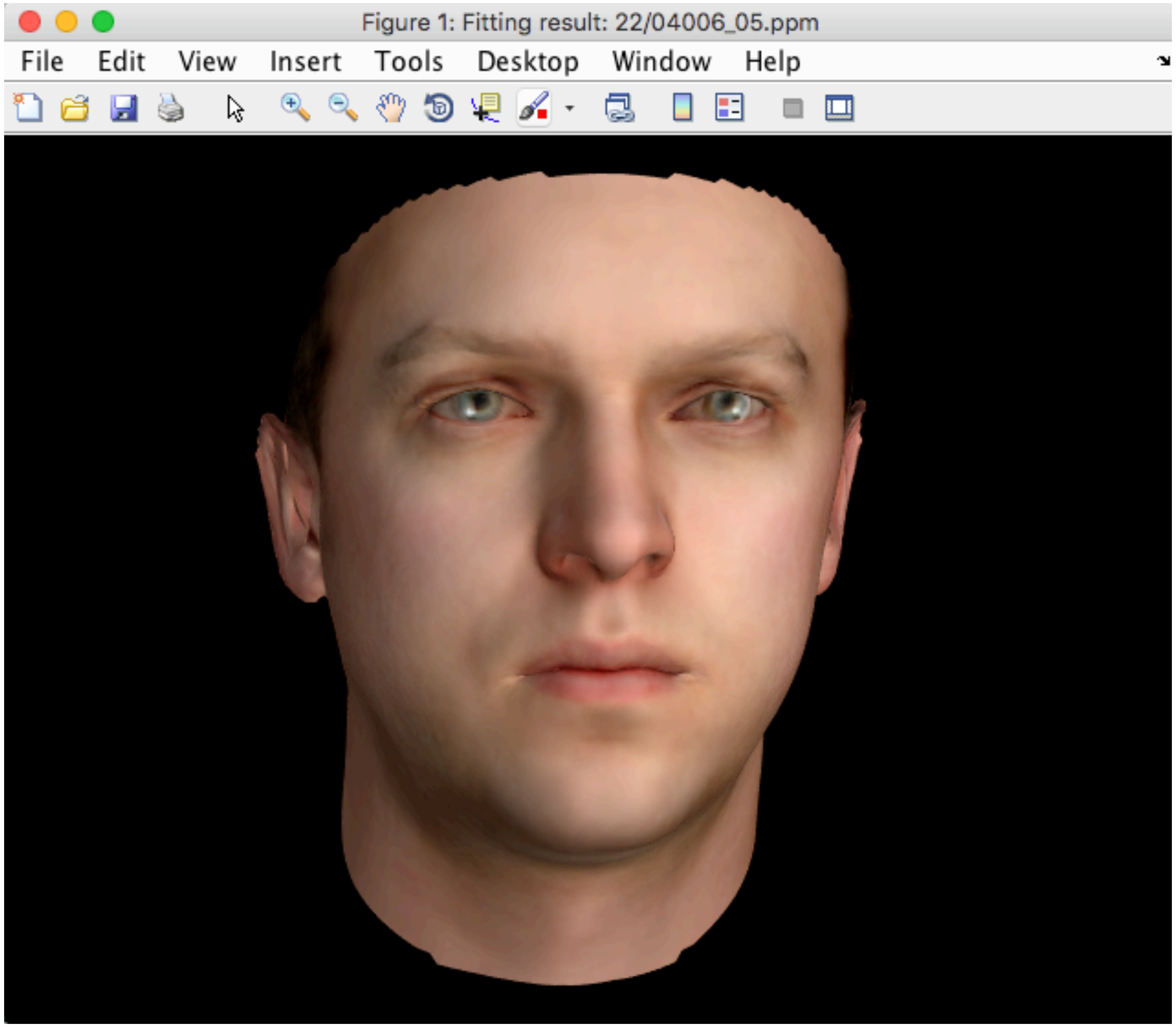


Figure 1: Fitting result: 22/04006_05.ppm
File
Edit
View
Insert
Tools
Desktop
Window
Help

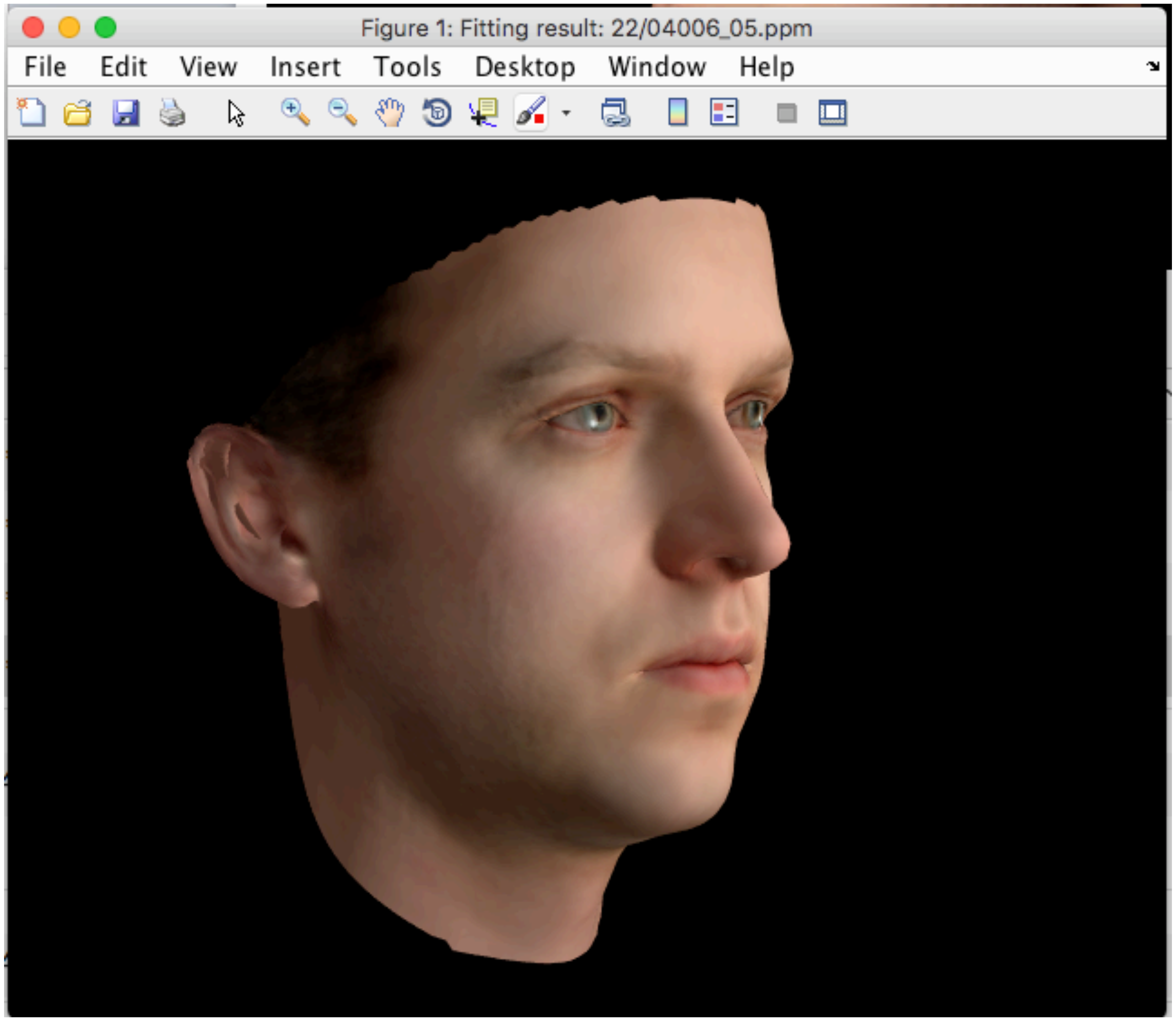


Figure 1: Fitting result: 22/04006_05.ppm
File
Edit
View
Insert
Tools
Desktop
Window
Help

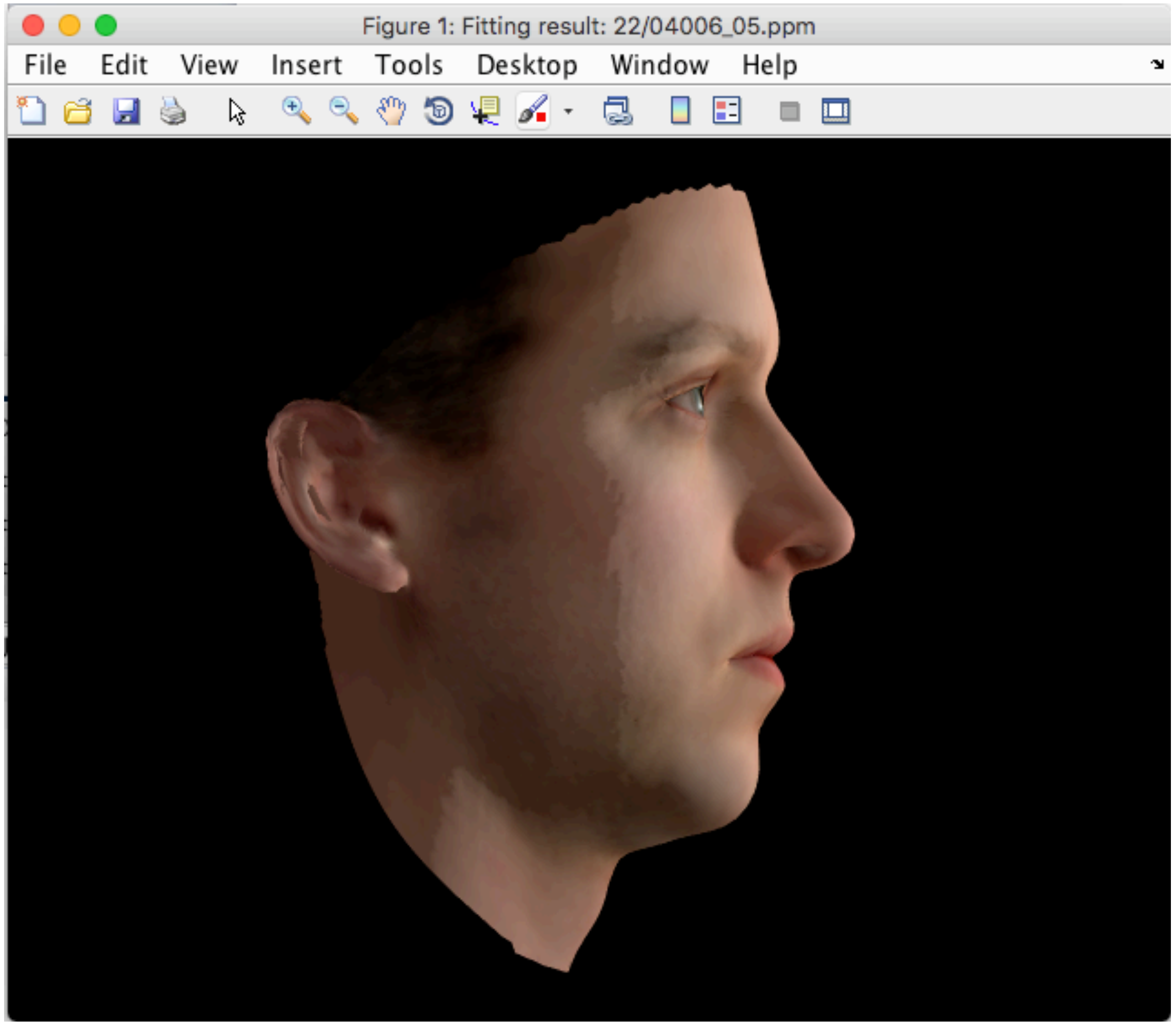
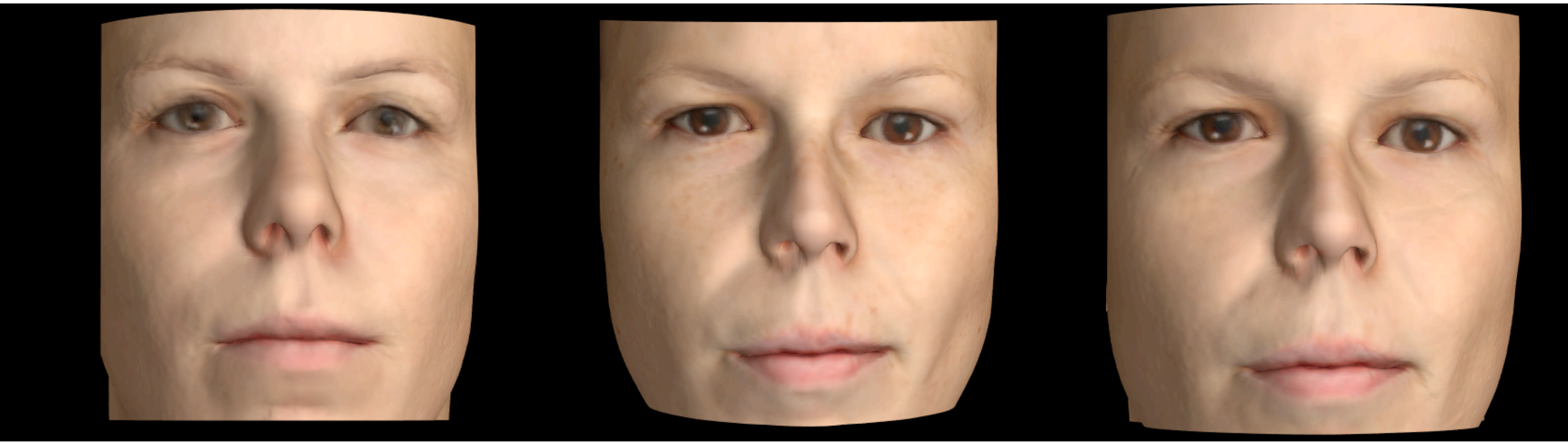
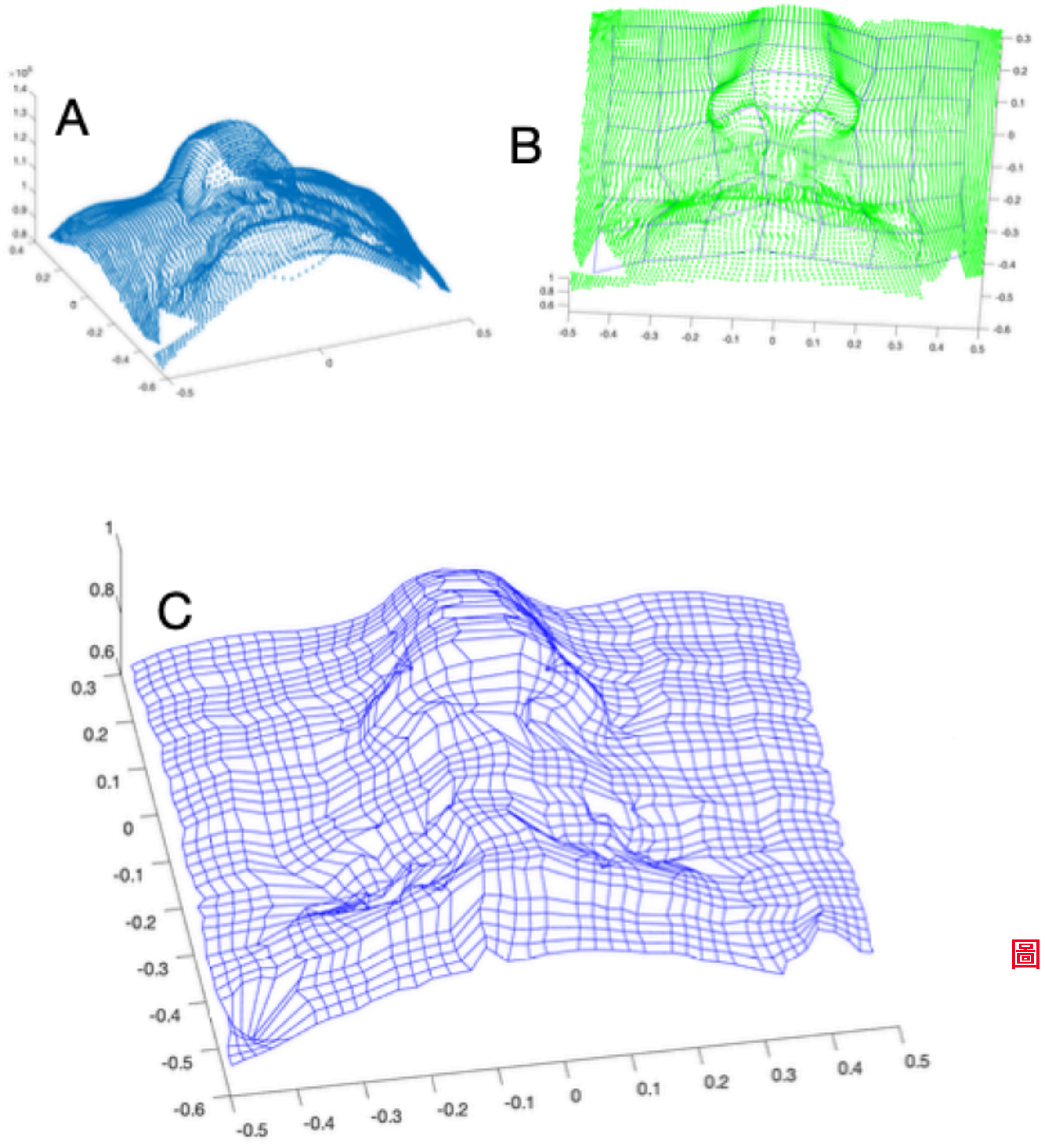


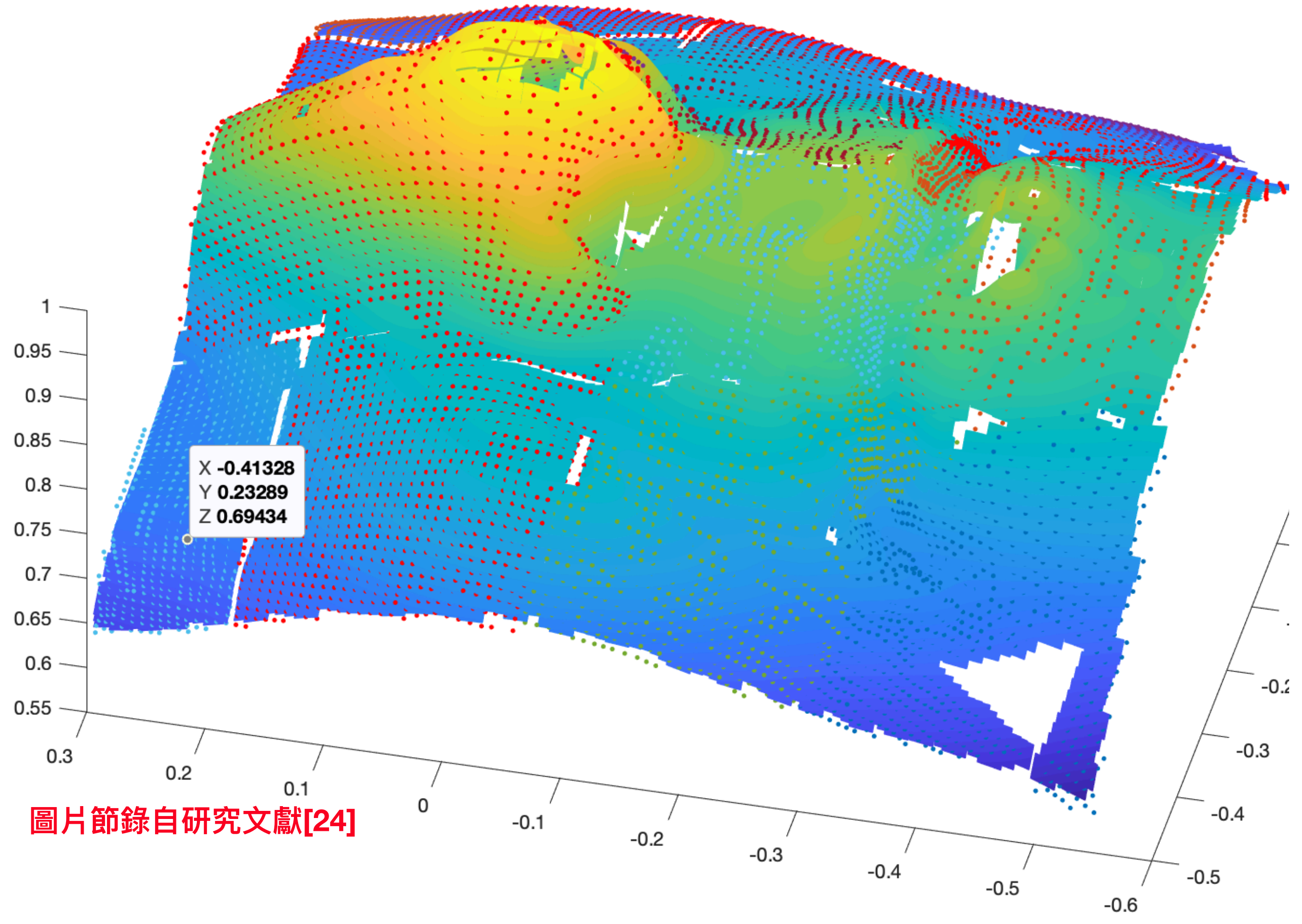
Figure 1: Fitting result: 22/04006_05.ppm
File
Edit
View
Insert
Tools
Desktop
Window
Help



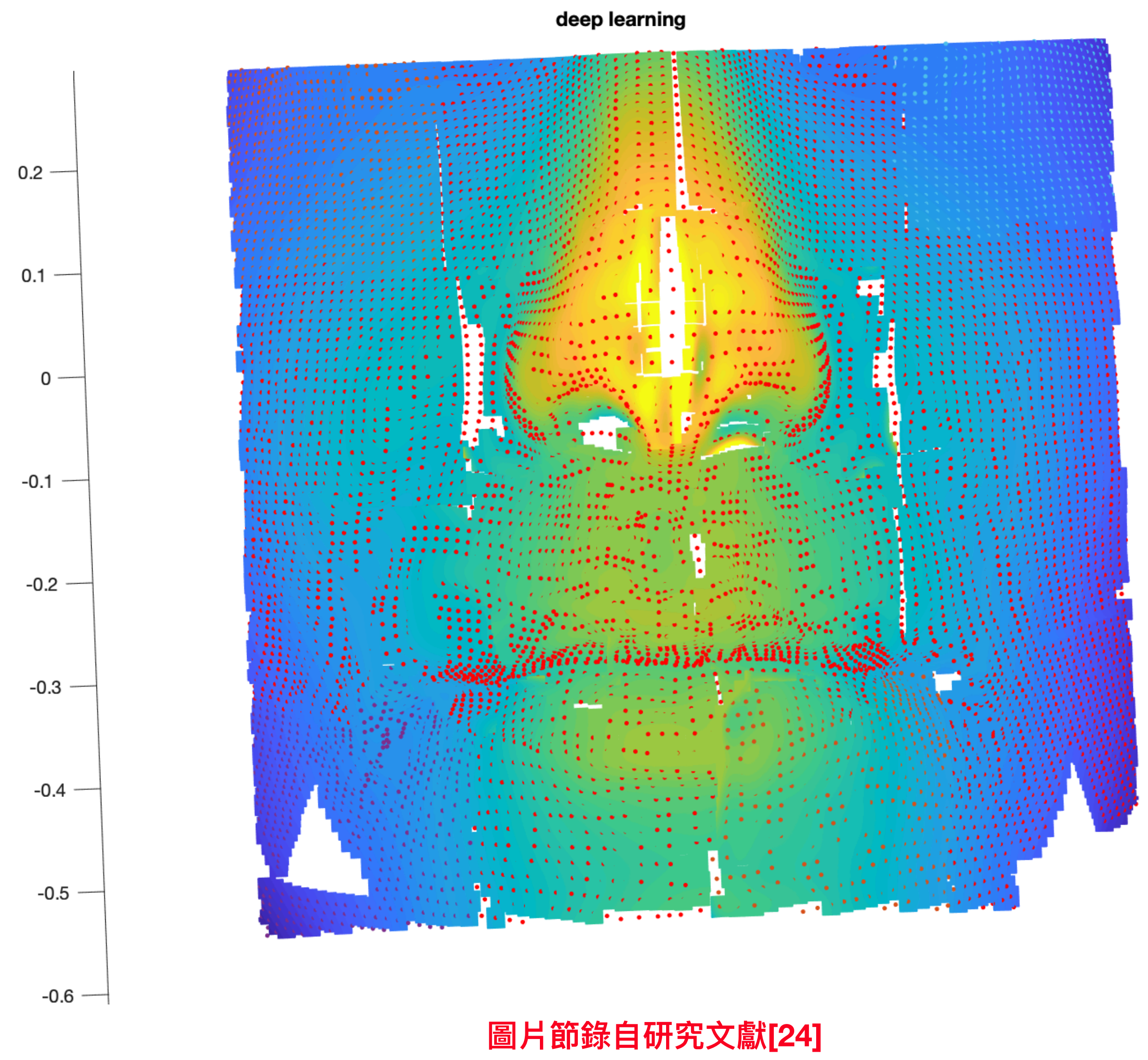


圖片節錄自研究文獻[13]

圖四、A. 部分3D人臉的原始資料點．B. 3D人臉的主彈性網．C. 3D人臉的巨型彈性網模式．



圖六、B、完整的臉頰、鼻子、嘴、下巴3D 人臉幾何重構與近似

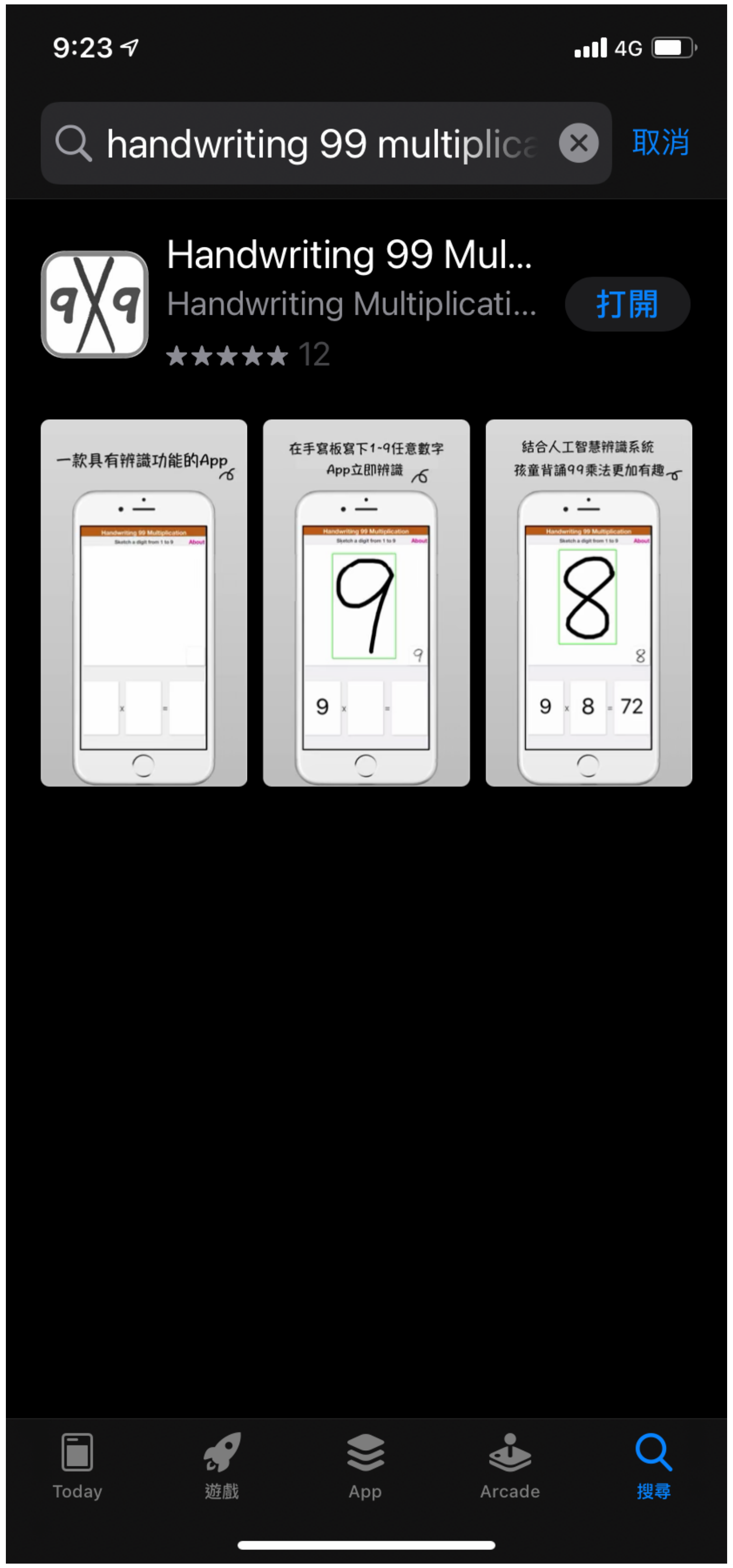


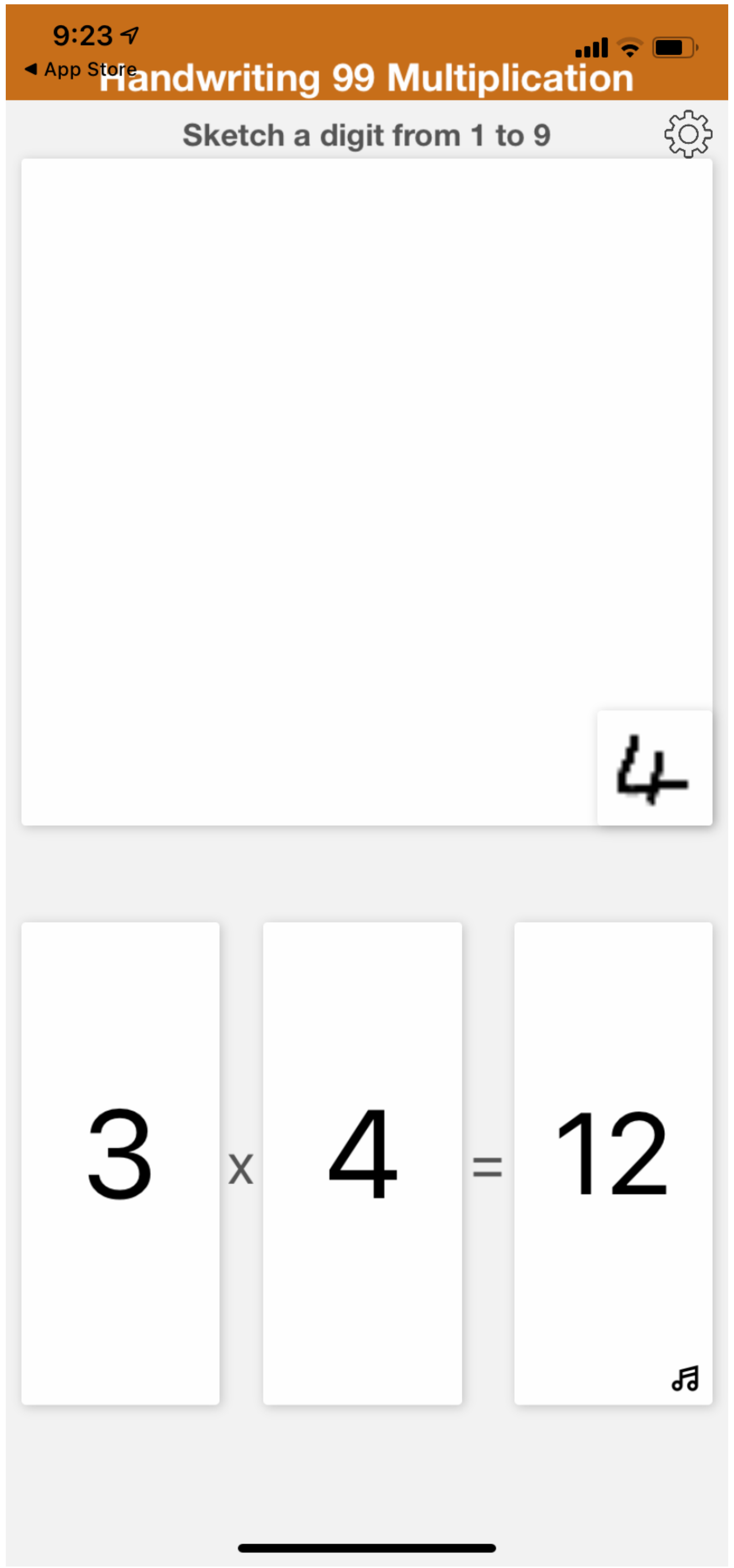
圖六、A完整的臉頰、鼻子、嘴、下巴3D 人臉幾何重構與近似

AI App

Handwriting 99 multiplication

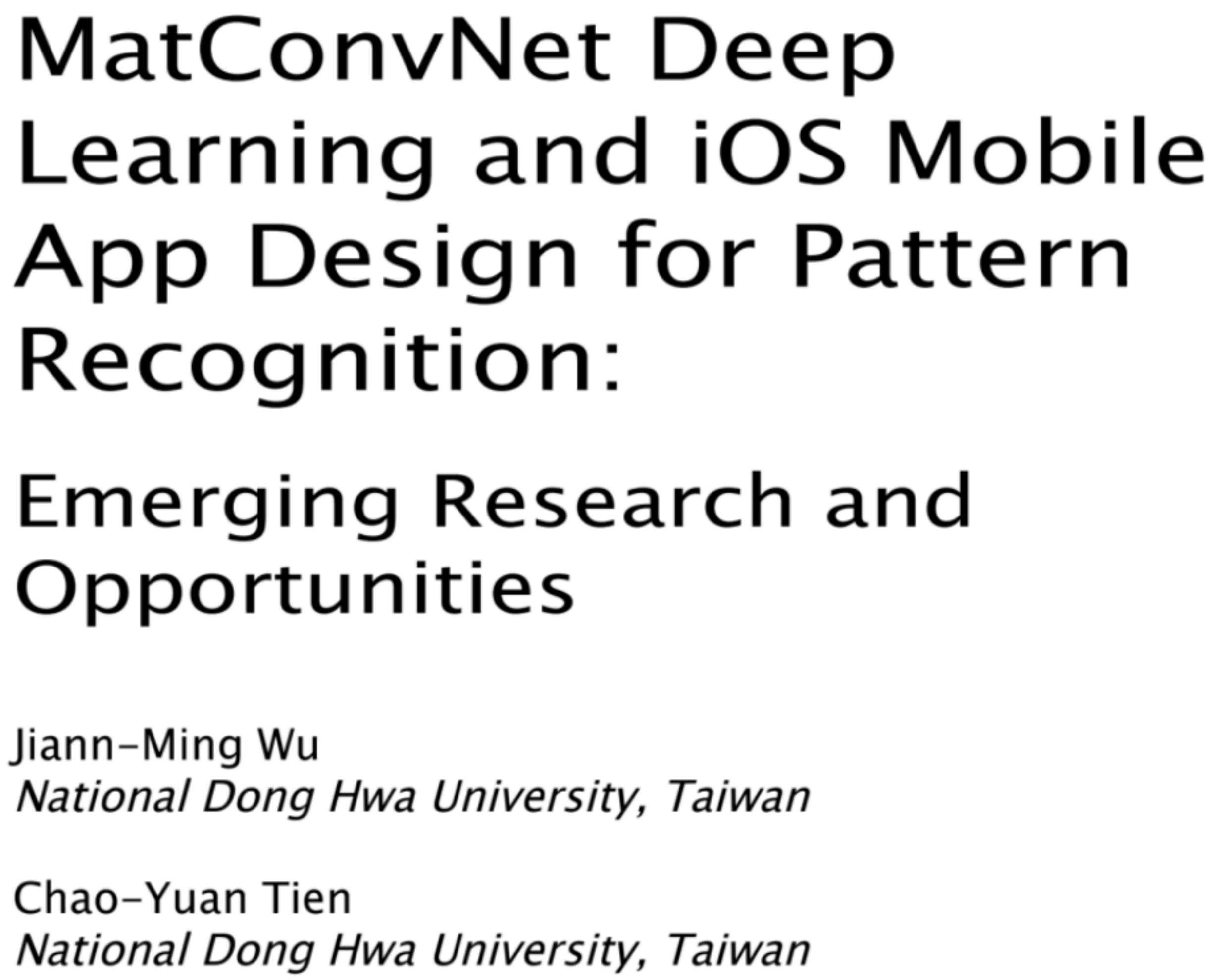
- An App published
- Integrate AI CNN models with touch screen of iPhone
- Author is graduated from AM NDHU
- 2018


9:23
handwriting 99 multiplica
取消
Handwriting 99 Mul...
Handwriting Multiplicati...
打開
★★★★★ 12
一款具有辨識功能的App
在手寫板寫下1-9任意數字 App立即辨識
結合人工智慧辨識系統 孩童背誦99乘法更加有趣
Today
遊戲
App
Arcade
搜尋



A Book

Deep learning has become a trending area of research due to its adaptive characteristics and high levels of applicability. In recent years, researchers have begun applying deep learning strategies to image analysis and pattern recognition for solving technical issues within image classification. As these technologies continue to advance, professionals have begun translating this intelligent programming language into mobile applications for devices. Programmers and web developers are in need of significant research on how to successfully develop pattern recognition applications using intelligent programming.

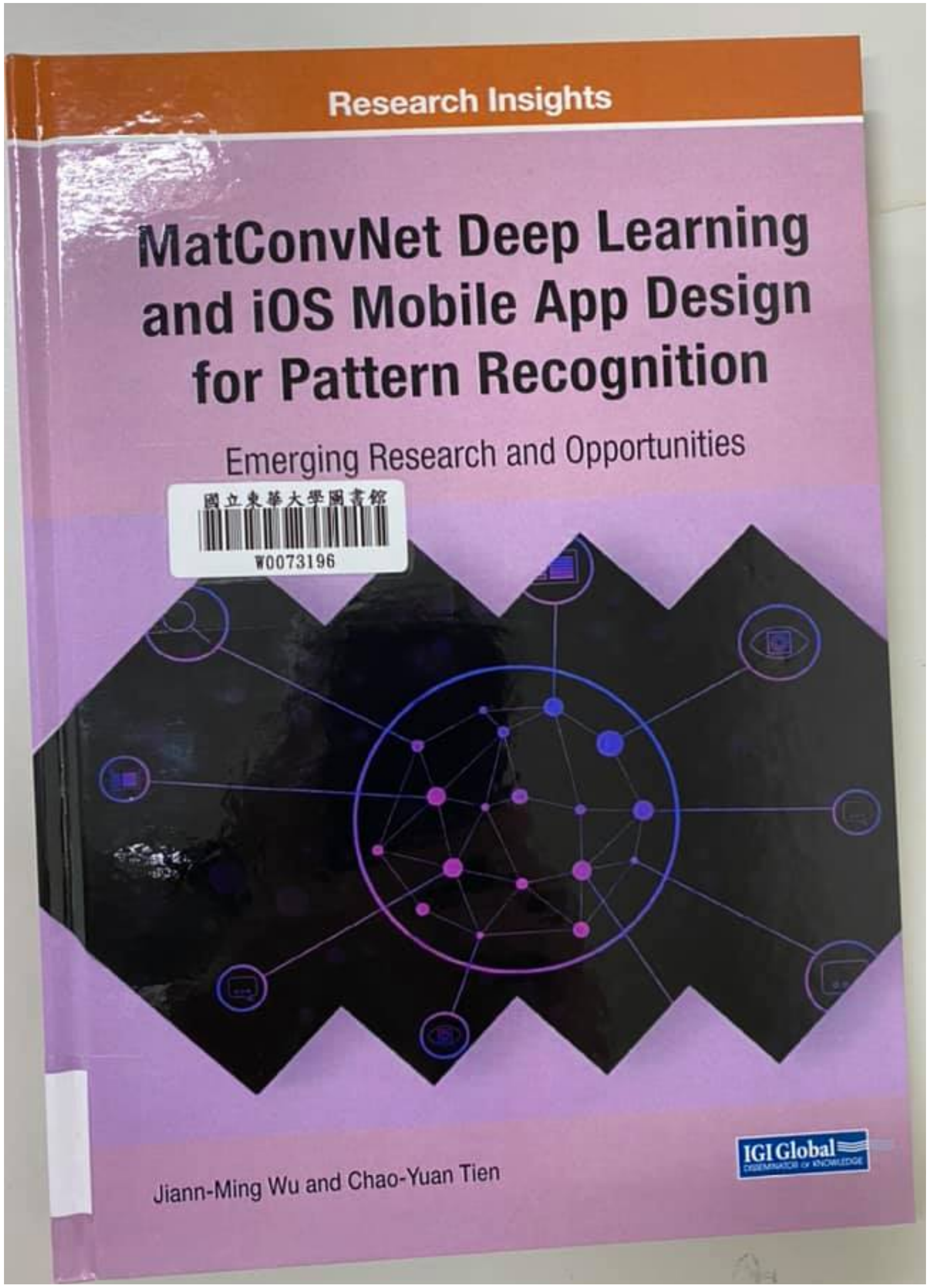


MatConvNet Deep Learning and iOS Mobile App Design for Pattern Recognition:

Emerging Research and Opportunities

Jiann-Ming Wu
National Dong Hwa University, Taiwan

Chao-Yuan Tien
National Dong Hwa University, Taiwan



Research Insights

MatConvNet Deep Learning and iOS Mobile App Design for Pattern Recognition

Emerging Research and Opportunities

Jiann-Ming Wu and Chao-Yuan Tien

IGI Global
DISSEMINATOR OF KNOWLEDGE

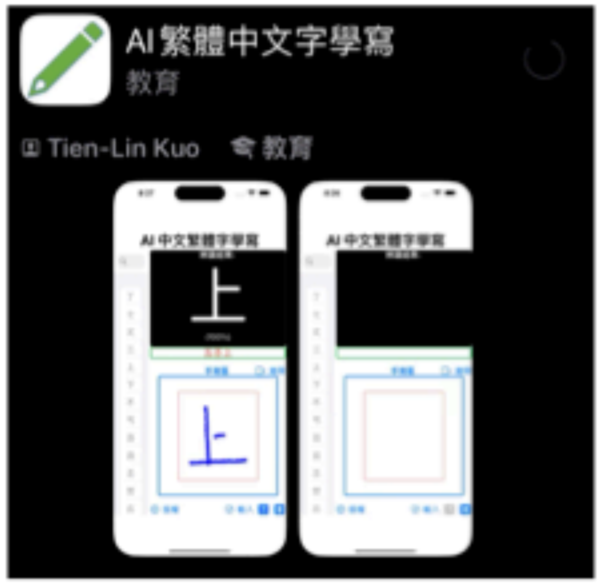
MatConvNet Deep Learning and iOS Mobile App Design for Pattern Recognition: Emerging Research and Opportunities is an essential reference source that presents a solution to developing intelligent pattern recognition Apps on iOS devices based on MatConvNet deep learning. Featuring research on topics such as medical image diagnosis, convolutional neural networks, and character classification, this book is ideally designed for programmers, developers, researchers, practitioners, engineers, academicians, students, scientists, and educators seeking coverage on the specific development of iOS mobile applications using pattern recognition strategies.

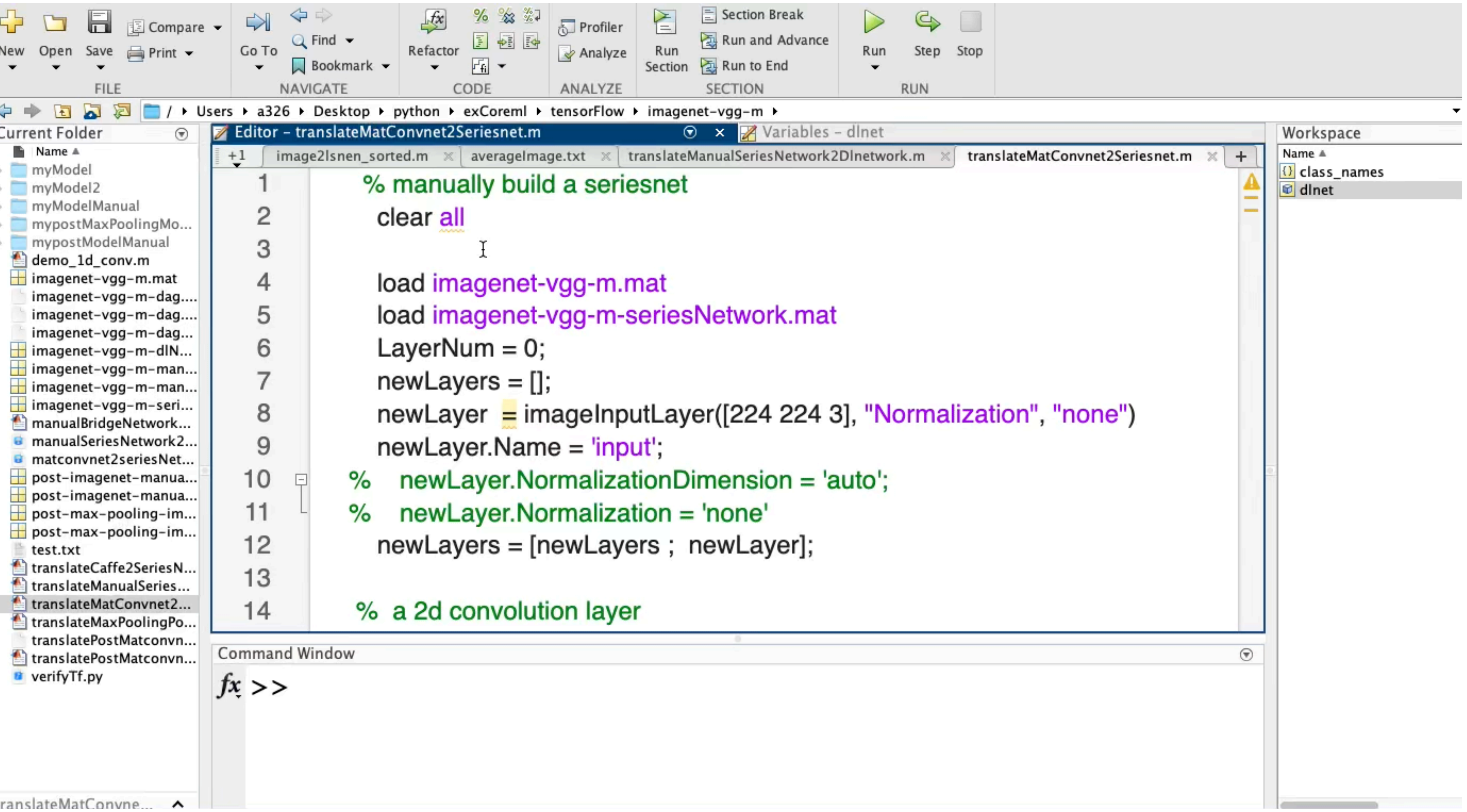
- Energy saving, fault tolerance and collective decisions 省能源，高錯誤容忍，集體決策
- Cars, robots, robot arms, medicine equipments, mobile phones 汽車、機器人、機器手、醫學器材、智慧手機
- 東華應數研究生上架的Apple App
- 研究生進行AI配置

AI景點路徑最佳化

AI看物辨識講台語

花蓮民宿民宿網
災民資料庫查詢

AI 繁體中文字學寫




```
% manually build a seriesnet
 clear all

 load imagenet-vgg-m.mat
 load imagenet-vgg-m-seriesNetwork.mat
 LayerNum = 0;
 newLayers = [];
 newLayer = imageInputLayer([224 224 3], "Normalization", "none")
 newLayer.Name = 'input';
%   newLayer.NormalizationDimension = 'auto';
%   newLayer.Normalization = 'none'
 newLayers = [newLayers ; newLayer];

% a 2d convolution layer
```

```
>>
```

myModel
myModel2
myModelManual
mypostMaxPoolingMo...
mypostModelManual
demo_1d_conv.m
imagenet-vgg-m.mat
imagenet-vgg-m-dag....
imagenet-vgg-m-dag....
imagenet-vgg-m-dag...
imagenet-vgg-m-dlN...
imagenet-vgg-m-man...
imagenet-vgg-m-man...
imagenet-vgg-m-seri...
manualBridgeNetwork...
manualSeriesNetwork2...
matconvnet2seriesNet...
post-imagenet-manua...
post-imagenet-manua...
post-max-pooling-im...
post-max-pooling-im...
test.txt
translateCaffe2SeriesN...
translateManualSeries...
translateMatConvnet2...
translateMaxPoolingPo...
translatePostMatconvn...
translatePostMatconvn...
verifyTf.py

MatConvNet

MatConvNet: CNNs for MATLAB

Obtaining MatConvNet

Documentation

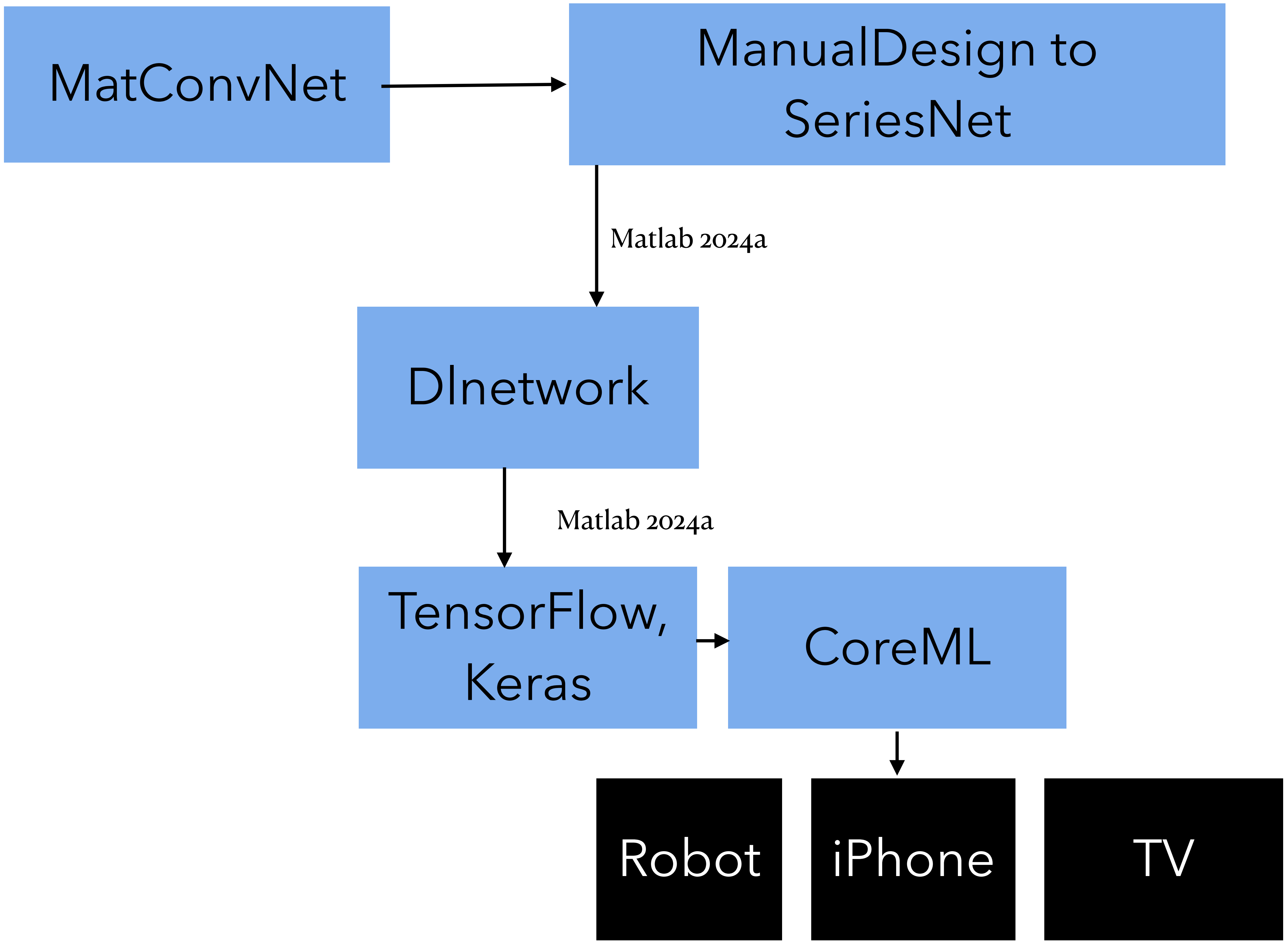
Extensions

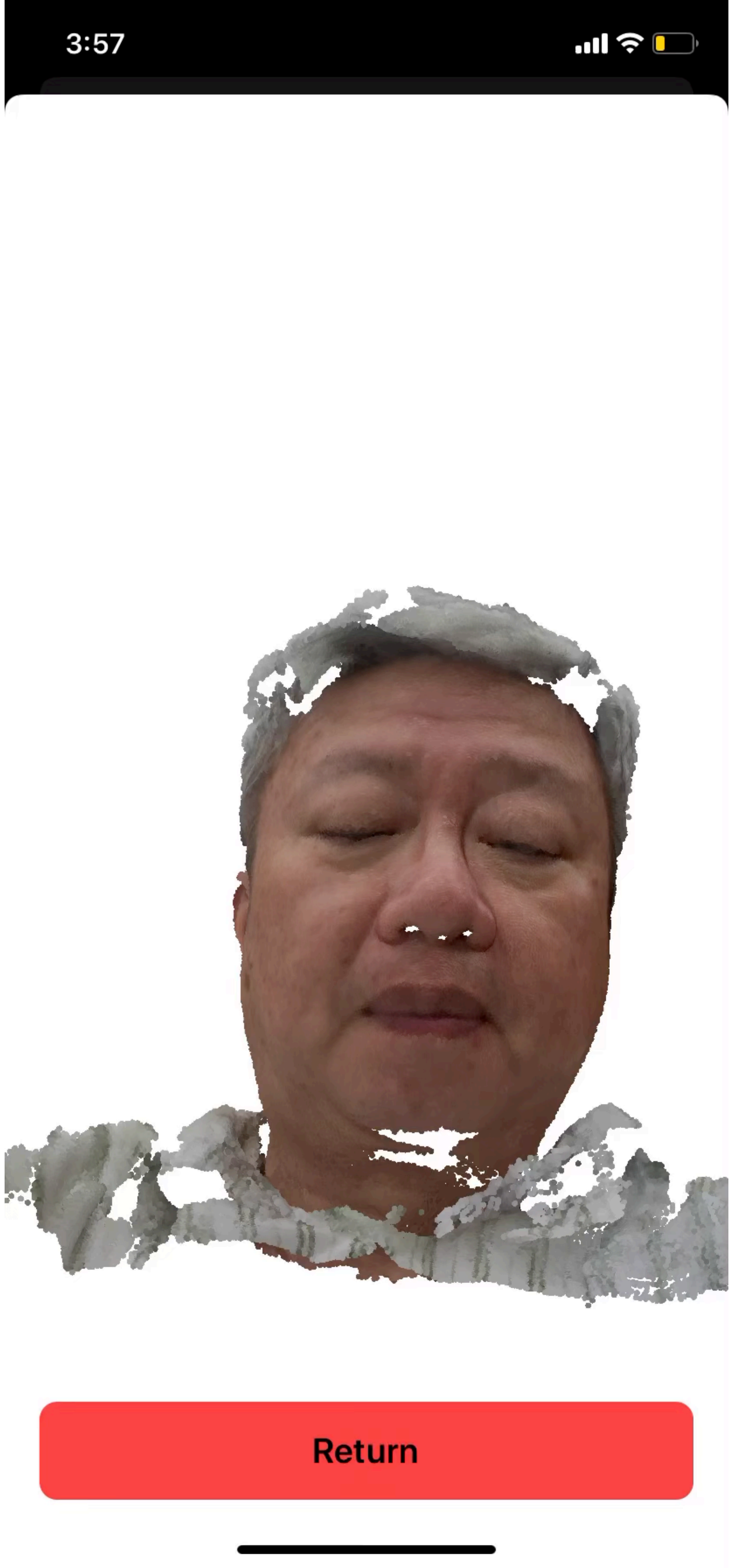
Getting started

Use cases

Other information

MatConvNet: CNNs for MATLAB


MatConvNet
ManualDesign to SeriesNet
Matlab 2024a
DInetwork
Matlab 2024a
TensorFlow, Keras
CoreML
Robot
iPhone
TV


3:57
Return

RadSee Launches Automotive 4D Imaging Radar

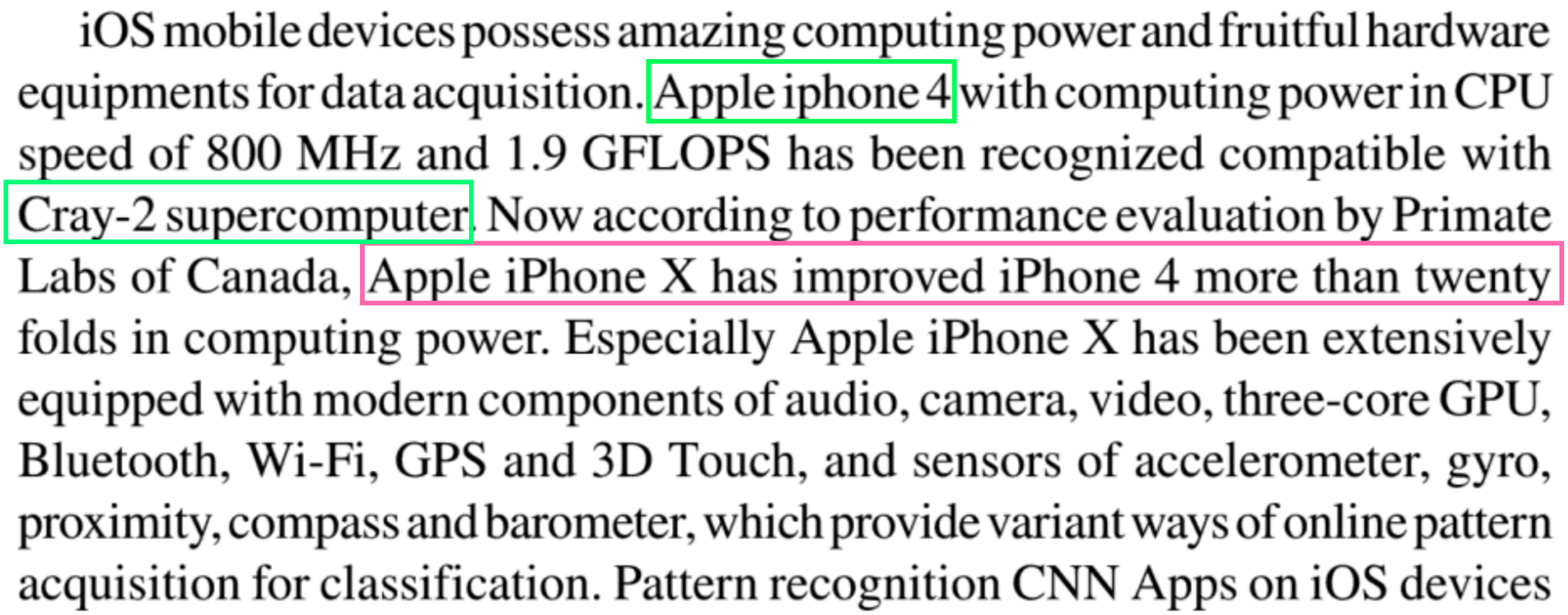
By Tiera Oliver

February 16, 2021

NEWS

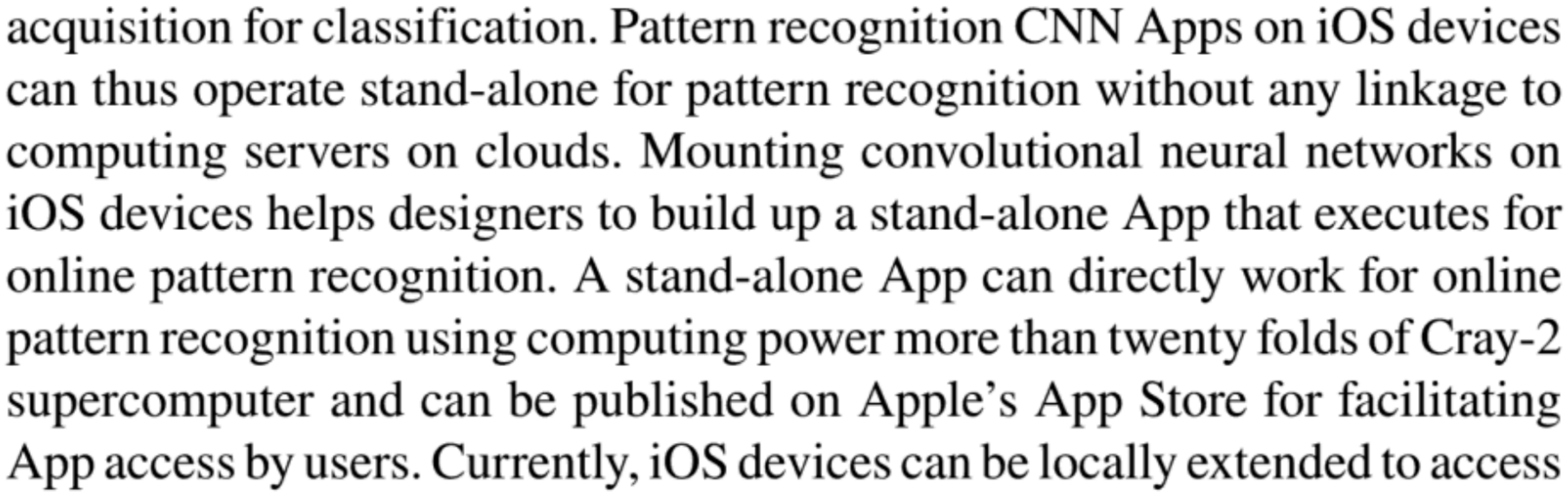
Per the company, RadSee Technologies announced the availability of the automotive industry's first 4D imaging radar for ADAS and autonomous vehicles able to deliver high performance and scalability to OEMs and Tier 1 suppliers at up to one-third the cost of previous solutions.

Iphone X and Cray II



iOS mobile devices possess amazing computing power and fruitful hardware equipments for data acquisition. Apple iphone 4 with computing power in CPU speed of 800 MHz and 1.9 GFLOPS has been recognized compatible with Cray-2 supercomputer. Now according to performance evaluation by Primate Labs of Canada, Apple iPhone X has improved iPhone 4 more than twenty folds in computing power. Especially Apple iPhone X has been extensively equipped with modern components of audio, camera, video, three-core GPU, Bluetooth, Wi-Fi, GPS and 3D Touch, and sensors of accelerometer, gyro, proximity, compass and barometer, which provide variant ways of online pattern acquisition for classification. Pattern recognition CNN Apps on iOS devices

Standalone AI App on Iphones



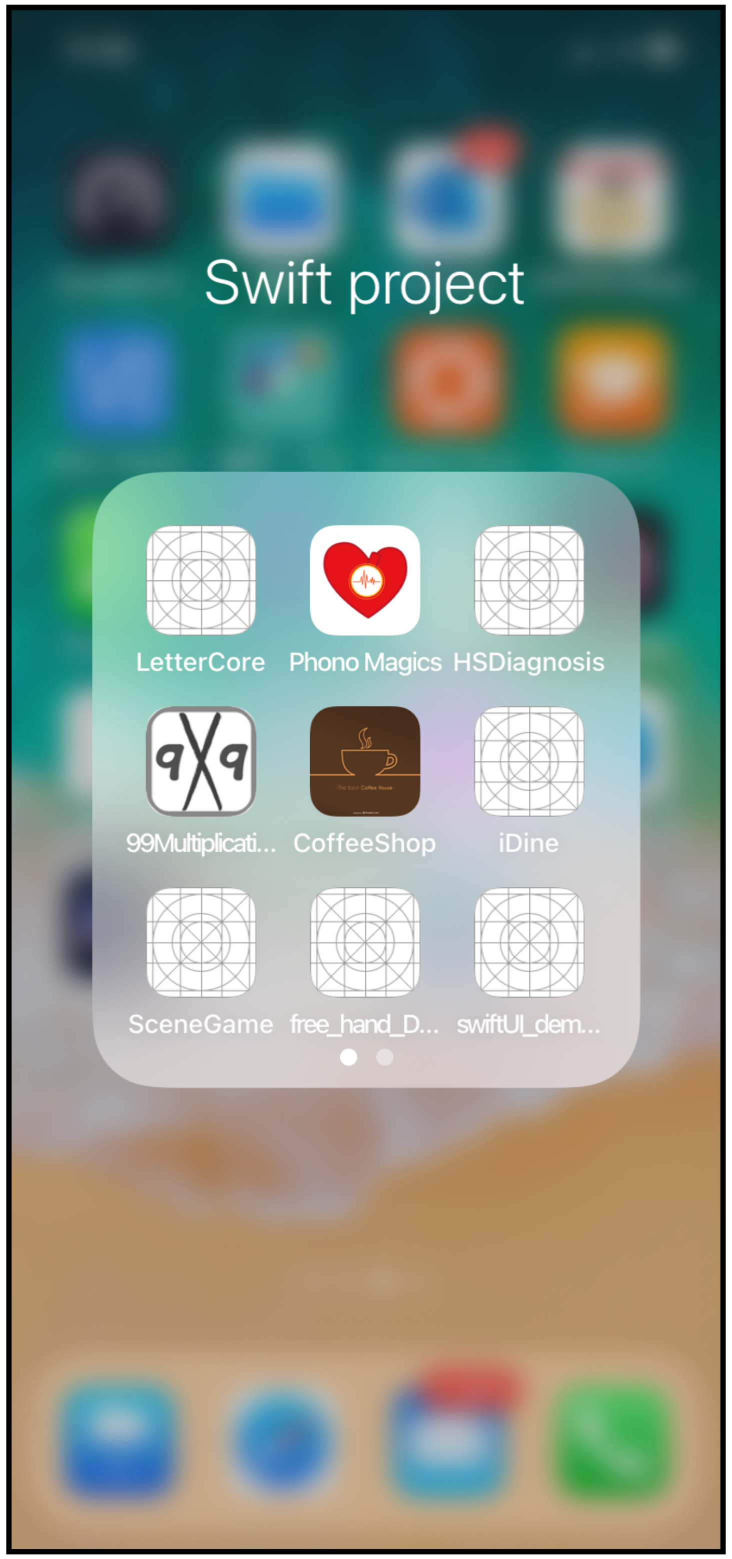
acquisition for classification. Pattern recognition CNN Apps on iOS devices can thus operate stand-alone for pattern recognition without any linkage to computing servers on clouds. Mounting convolutional neural networks on iOS devices helps designers to build up a stand-alone App that executes for online pattern recognition. A stand-alone App can directly work for online pattern recognition using computing power more than twenty folds of Cray-2 supercomputer and can be published on Apple's App Store for facilitating App access by users. Currently, iOS devices can be locally extended to access

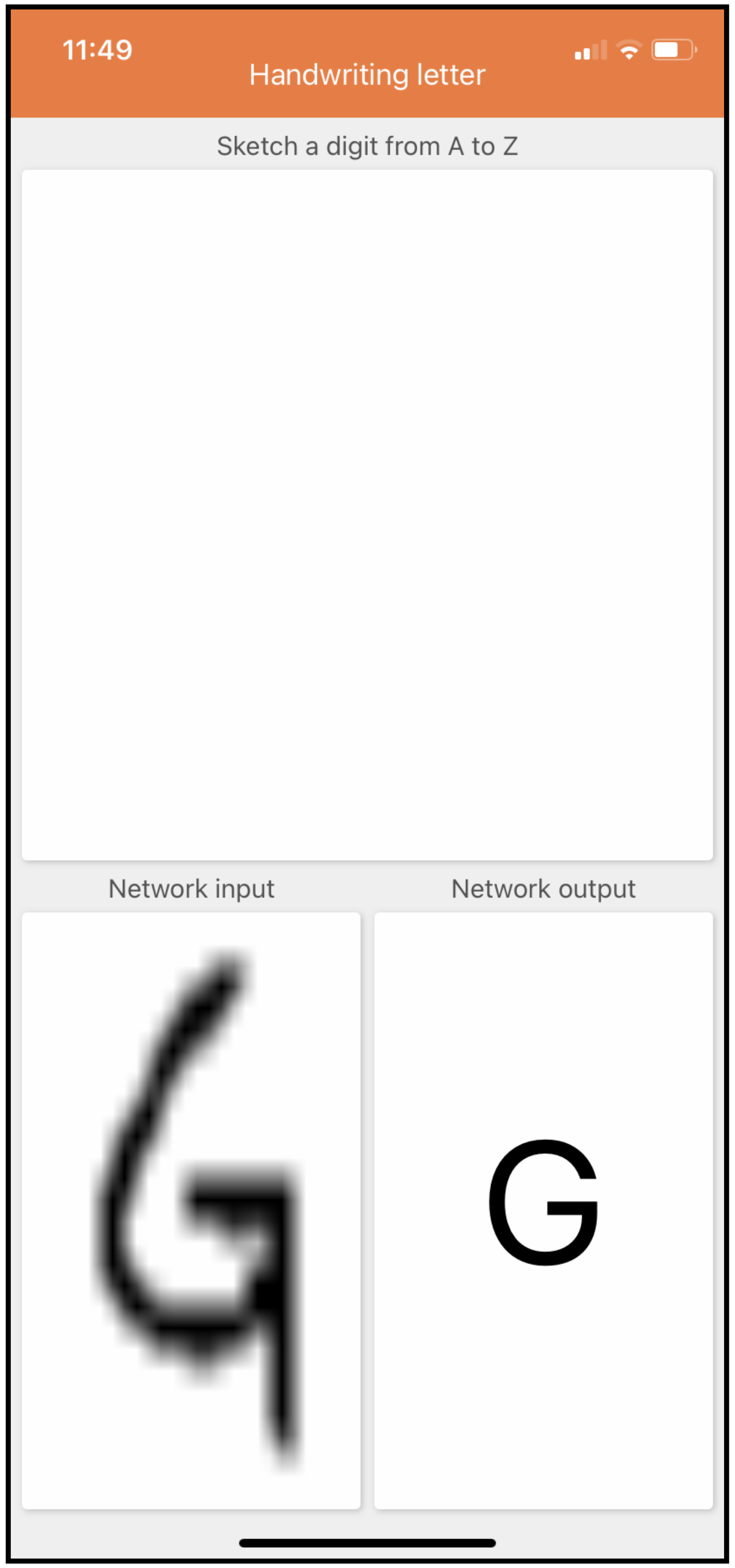
RadSee

3dScannerApp.com

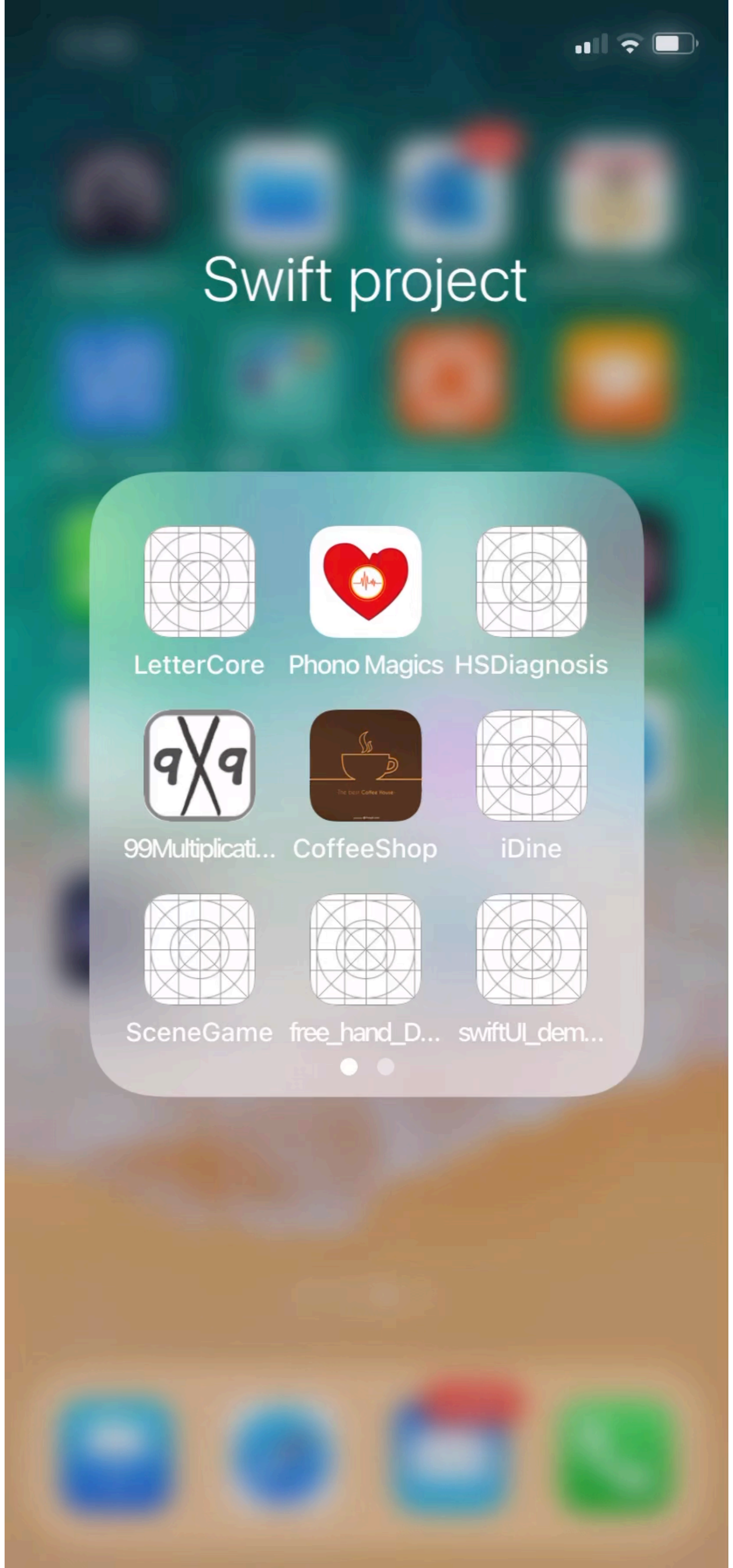
Hand Writing Character Recognition

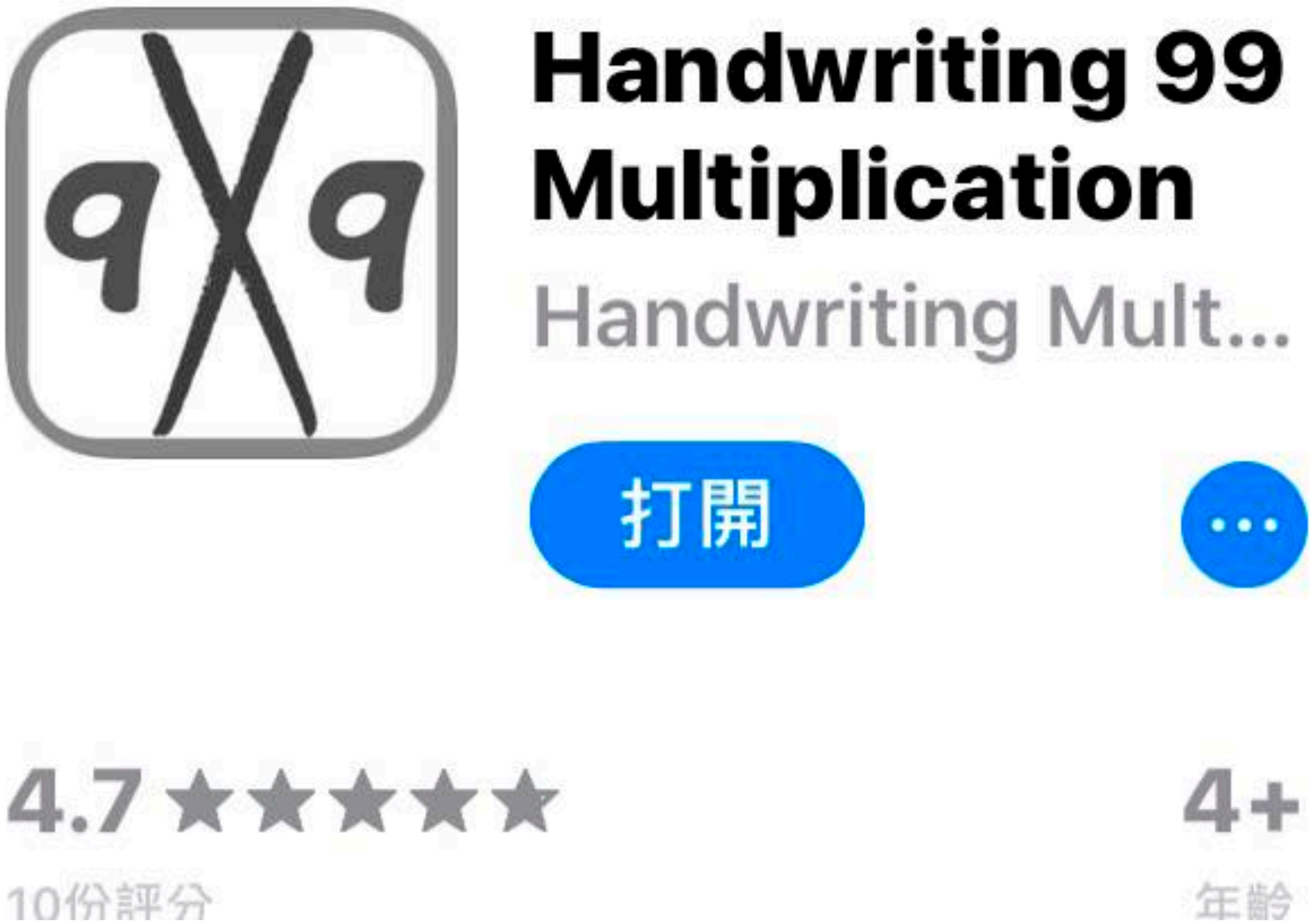
- Hand writing
- Deep learning
- Deep CNN
- AI Pattern Recognition





Hand Writing Character Recognition





Handwriting 99 Multiplication

Handwriting Mult...

4.7 ★★★★★

10份評分

4+

年齡

MathWorks Introduces the Release 2020a of MATLAB and Simulink

By Tiera Oliver

May 14, 2020

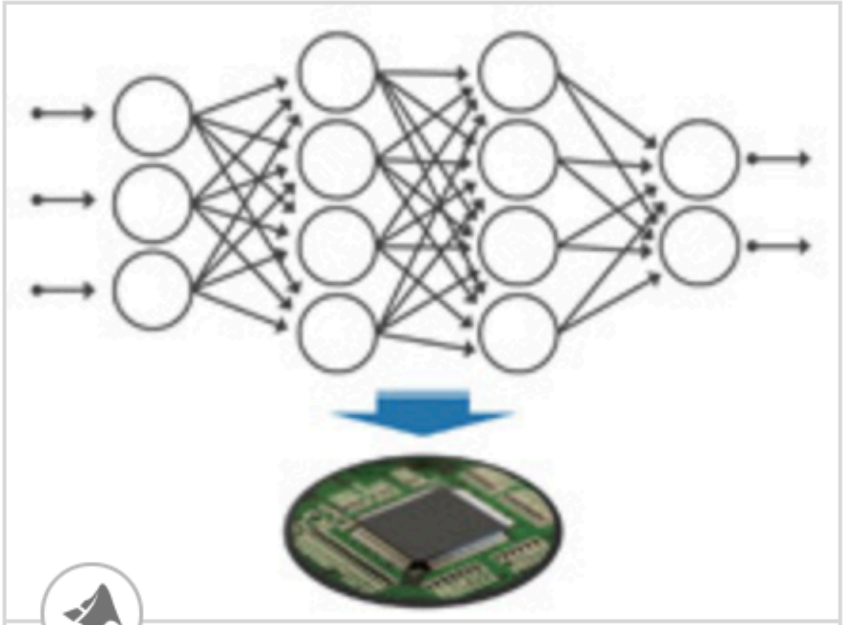
NEWS

LinkedIn Twitter Facebook Email More Print

R2020a introduces expanded AI capabilities for deep learning as well as new capabilities for automotive and wireless engineers.

Lidar Toolbox

Design, analyze, and test lidar processing systems



Deep Learning HDL Toolbox

Prototype and deploy deep learning networks on FPGAs and SoCs

Installed

Statistics and Machine Learning Toolbox

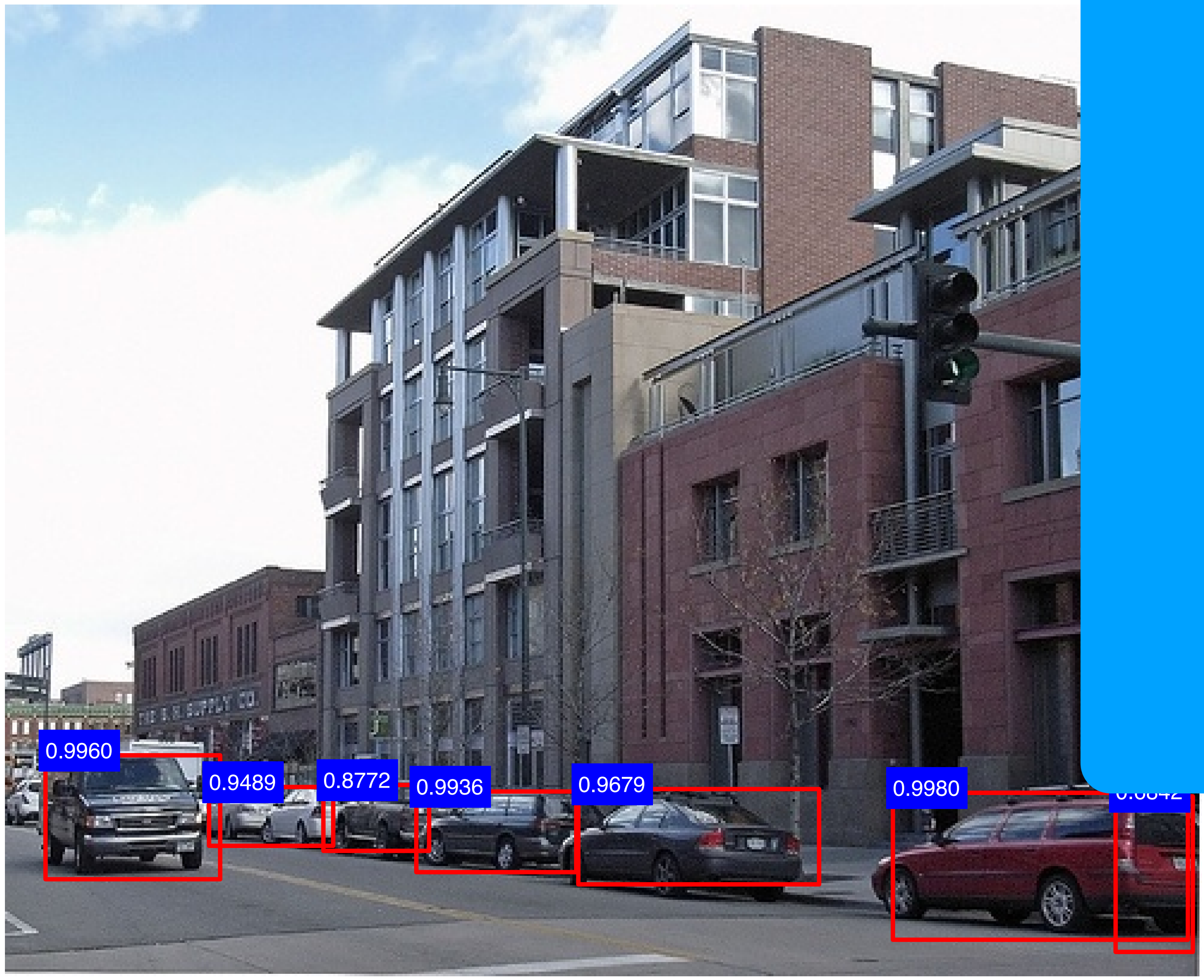
Analyze and model data using statistics and machine learning

R2020b now available

MatconvNet/examples/fast_rcnn

fast_rcnn_demo

Detections for class 'car'

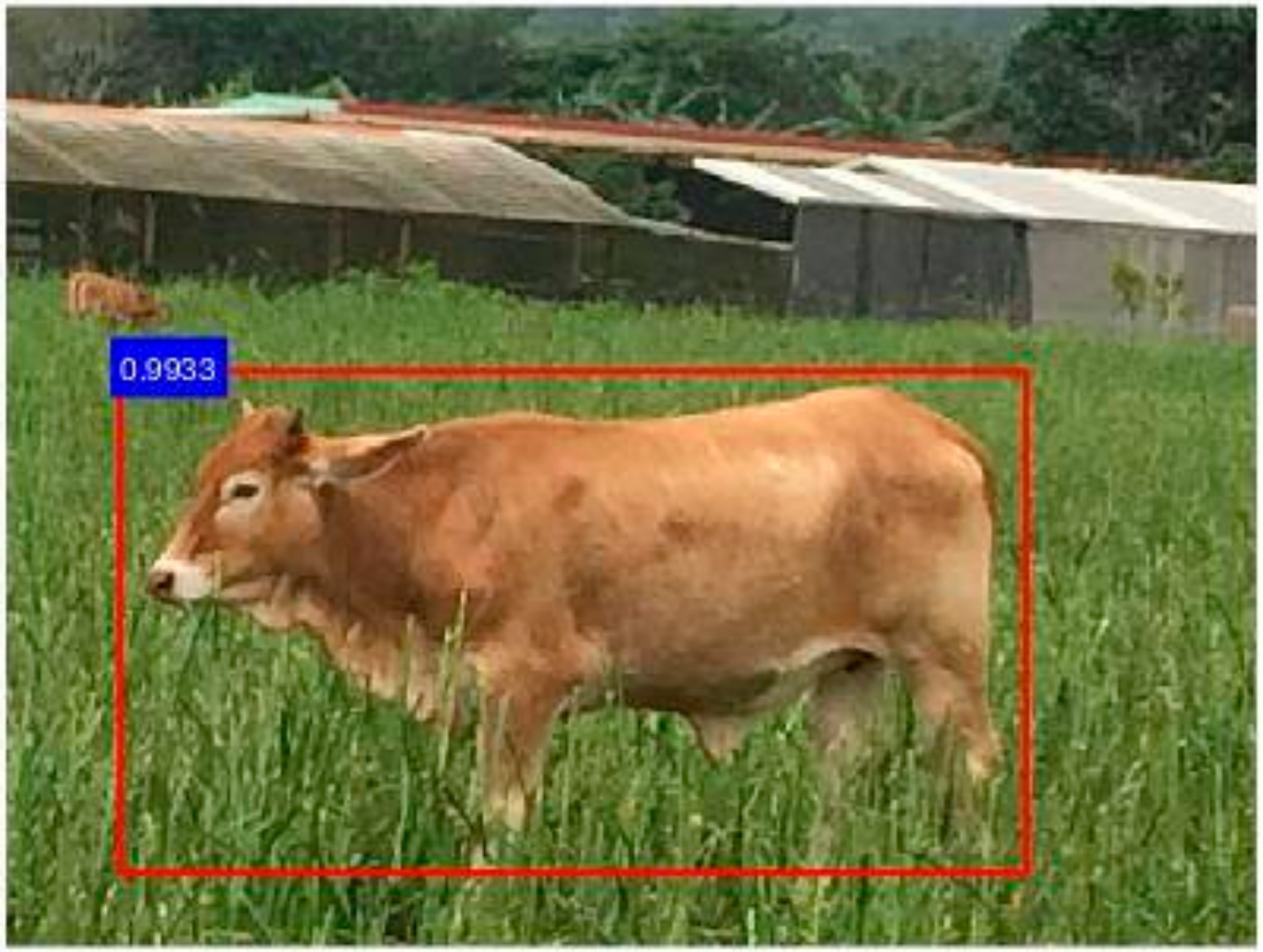


Problem
Architecture
Learning
Data
Execution
Results
your comments
.
.
.
Reference

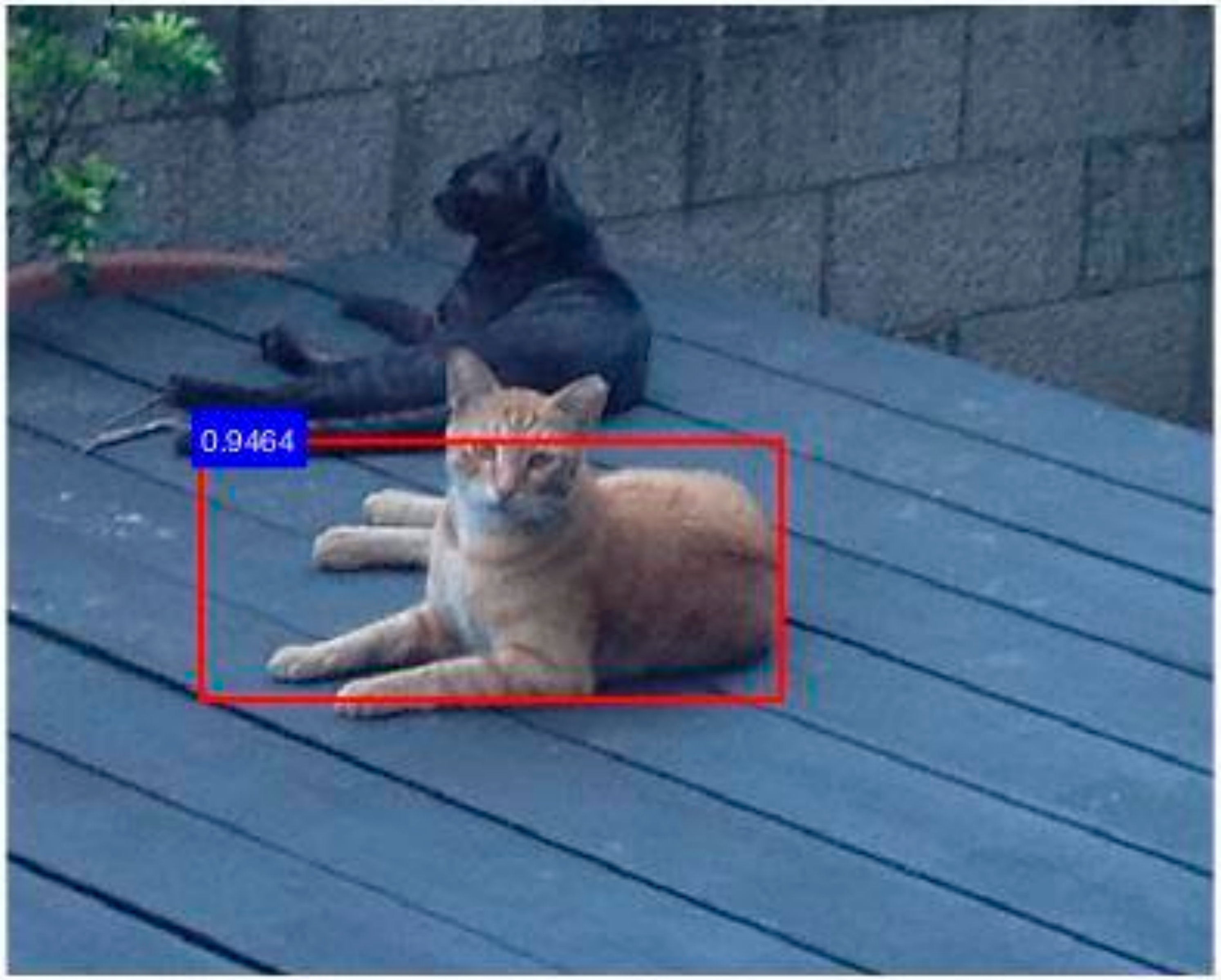
Please download again fast-rcnn-vgg16-pascal07-dagnn

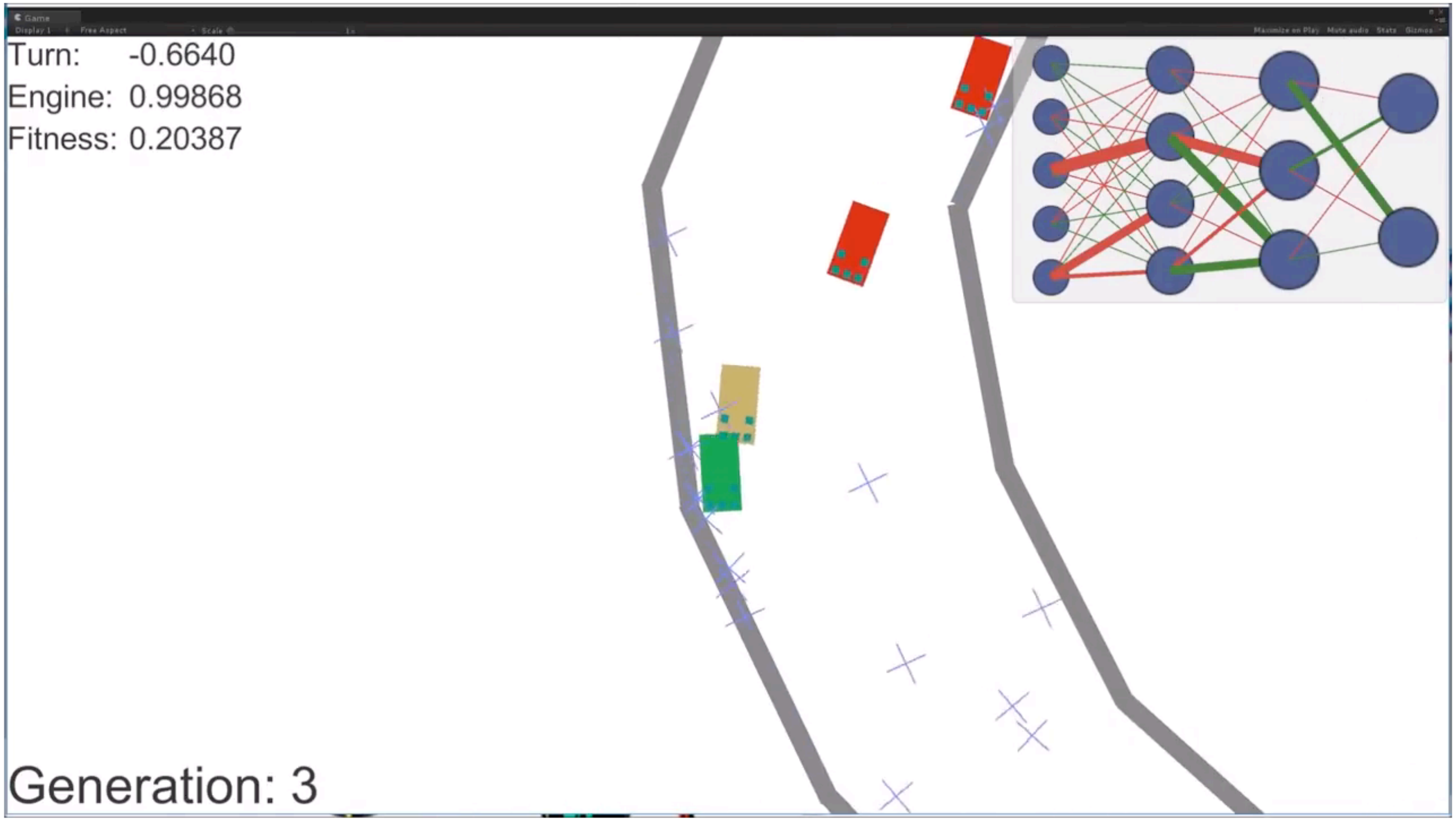
http://www.vlfeat.org/matconvnet/pretrained/

Detections for class 'cow'

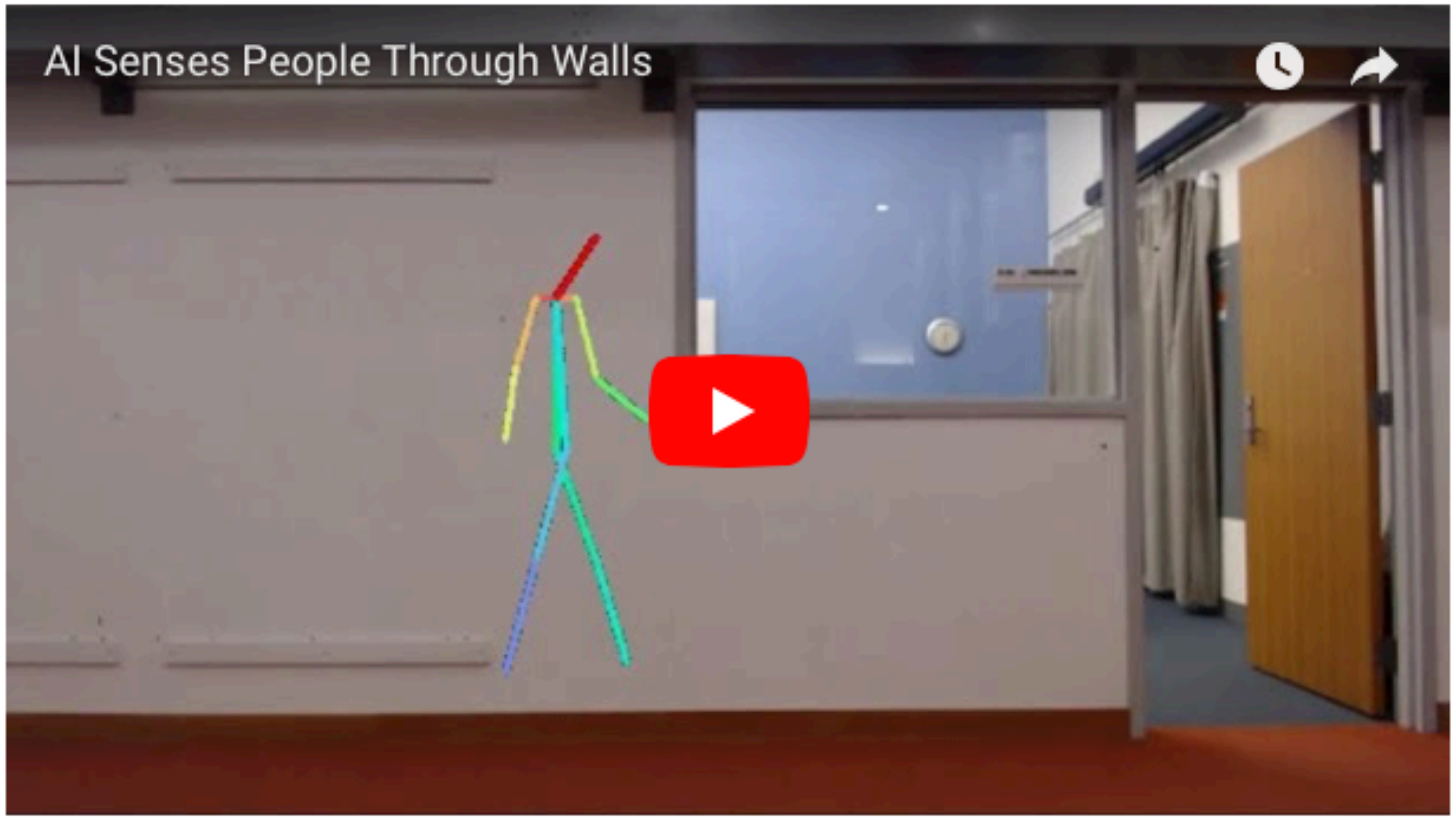


Detections for class 'cat'

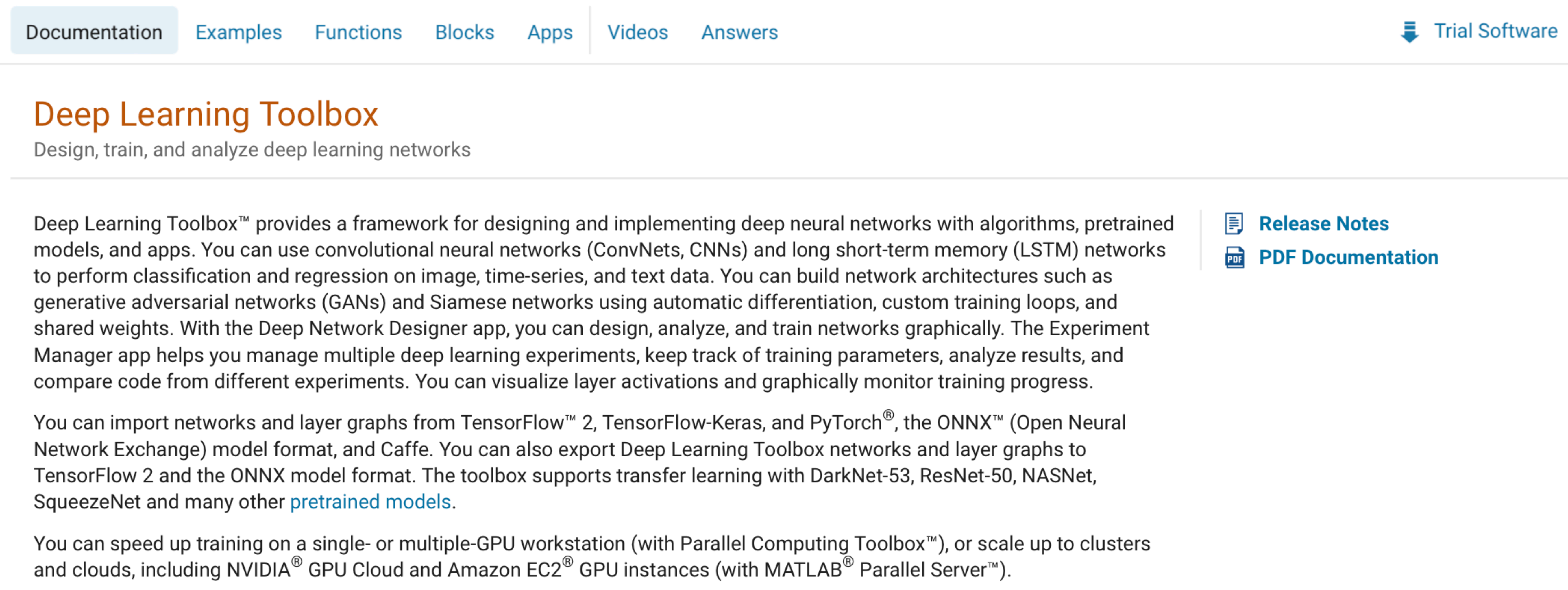




Deep Learning Cars


AI Senses People Through Walls

Neural Networks : Application and algorithm



Deep Learning Toolbox

Design, train, and analyze deep learning networks

Deep Learning Toolbox™ provides a framework for designing and implementing deep neural networks with algorithms, pretrained models, and apps. You can use convolutional neural networks (ConvNets, CNNs) and long short-term memory (LSTM) networks to perform classification and regression on image, time-series, and text data. You can build network architectures such as generative adversarial networks (GANs) and Siamese networks using automatic differentiation, custom training loops, and shared weights. With the Deep Network Designer app, you can design, analyze, and train networks graphically. The Experiment Manager app helps you manage multiple deep learning experiments, keep track of training parameters, analyze results, and compare code from different experiments. You can visualize layer activations and graphically monitor training progress.

You can import networks and layer graphs from TensorFlow™ 2, TensorFlow-Keras, and PyTorch®, the ONNX™ (Open Neural Network Exchange) model format, and Caffe. You can also export Deep Learning Toolbox networks and layer graphs to TensorFlow 2 and the ONNX model format. The toolbox supports transfer learning with DarkNet-53, ResNet-50, NASNet, SqueezeNet and many other pretrained models.

You can speed up training on a single- or multiple-GPU workstation (with Parallel Computing Toolbox™), or scale up to clusters and clouds, including NVIDIA® GPU Cloud and Amazon EC2® GPU instances (with MATLAB® Parallel Server™).

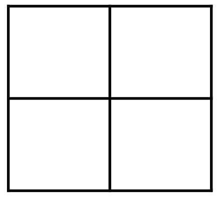
Computation

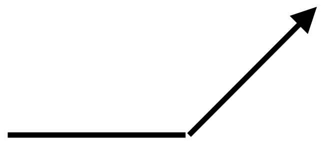
$$h_1, h_2, \ldots, h_n$$

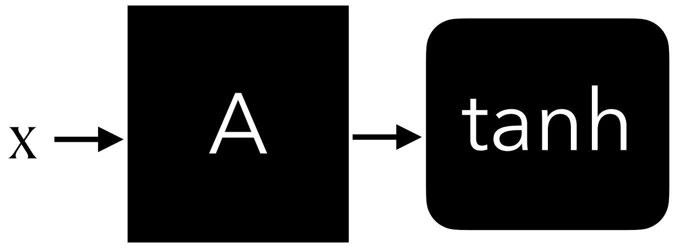
$$z_i = \frac{e^{h_i}}{\sum_j e^{h_j}}$$

- Linear projection, Projective fields
- Radial basis function
- Pooling, Stacking, Tensor
- Lattice-structured data organized following minimal wiring maximal fitting principle
- Post-nonlinear transformation
- Relu
- Sigmoid function
- Built-in parameters optimized by minimization of the mean square error
- Softmax
- Softmaxloss
- 2d convolution, 3d convolution

$$sigmoid(x) = \frac{tanh(x) + 1}{2}$$







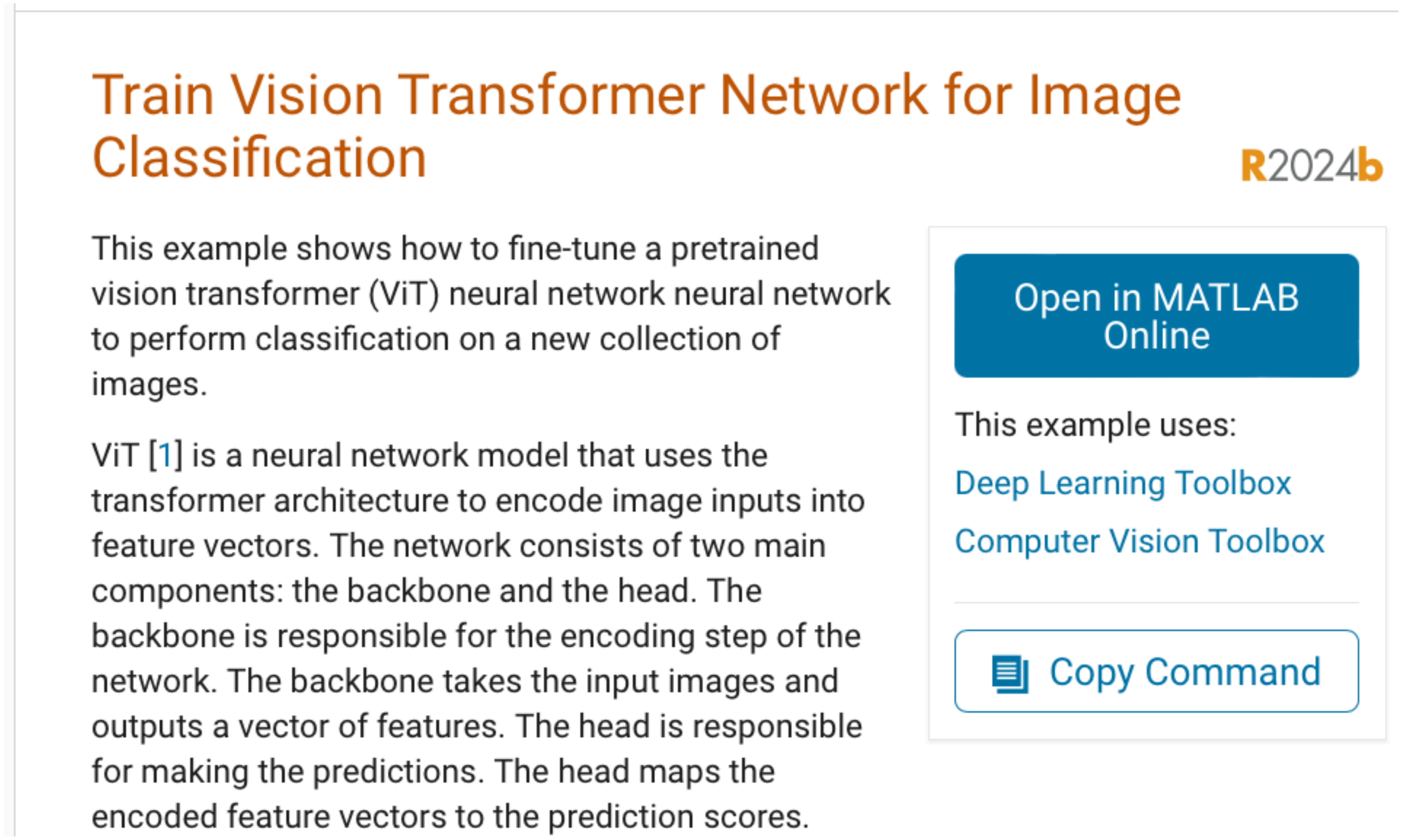
$$\text{Output:} A_1x + A_2x = (A_1 + A_2)x = Ax$$

$$\tanh(A_1x) + \tanh(A_2x) \neq \tanh(Ax)$$

Numerical algorithm

- Backpropagation
- Gradient descent method
- Stochastic gradient descent algorithm
- Newton-Gauss method
- Levenberg-Marquardt method
- Expectation maximization method
- Simulated annealing method
- Mean field annealing method
- Annealed expectation maximization method
- A hybrid of the gradient descent and mean field annealing methods

$$h_1 = tanh(A_1 x)$$
$$h_2 = tanh(A_2 h_1)$$
$$h_{i+1} = tanh(A_{i+1} h_i)$$
$$z = h_n = tanh(A_n h_{n-1})$$
$$\frac{dz}{dx} = ?$$



Train Vision Transformer Network for Image Classification

R2024b

This example shows how to fine-tune a pretrained vision transformer (ViT) neural network neural network to perform classification on a new collection of images.

ViT [1] is a neural network model that uses the transformer architecture to encode image inputs into feature vectors. The network consists of two main components: the backbone and the head. The backbone is responsible for the encoding step of the network. The backbone takes the input images and outputs a vector of features. The head is responsible for making the predictions. The head maps the encoded feature vectors to the prediction scores.

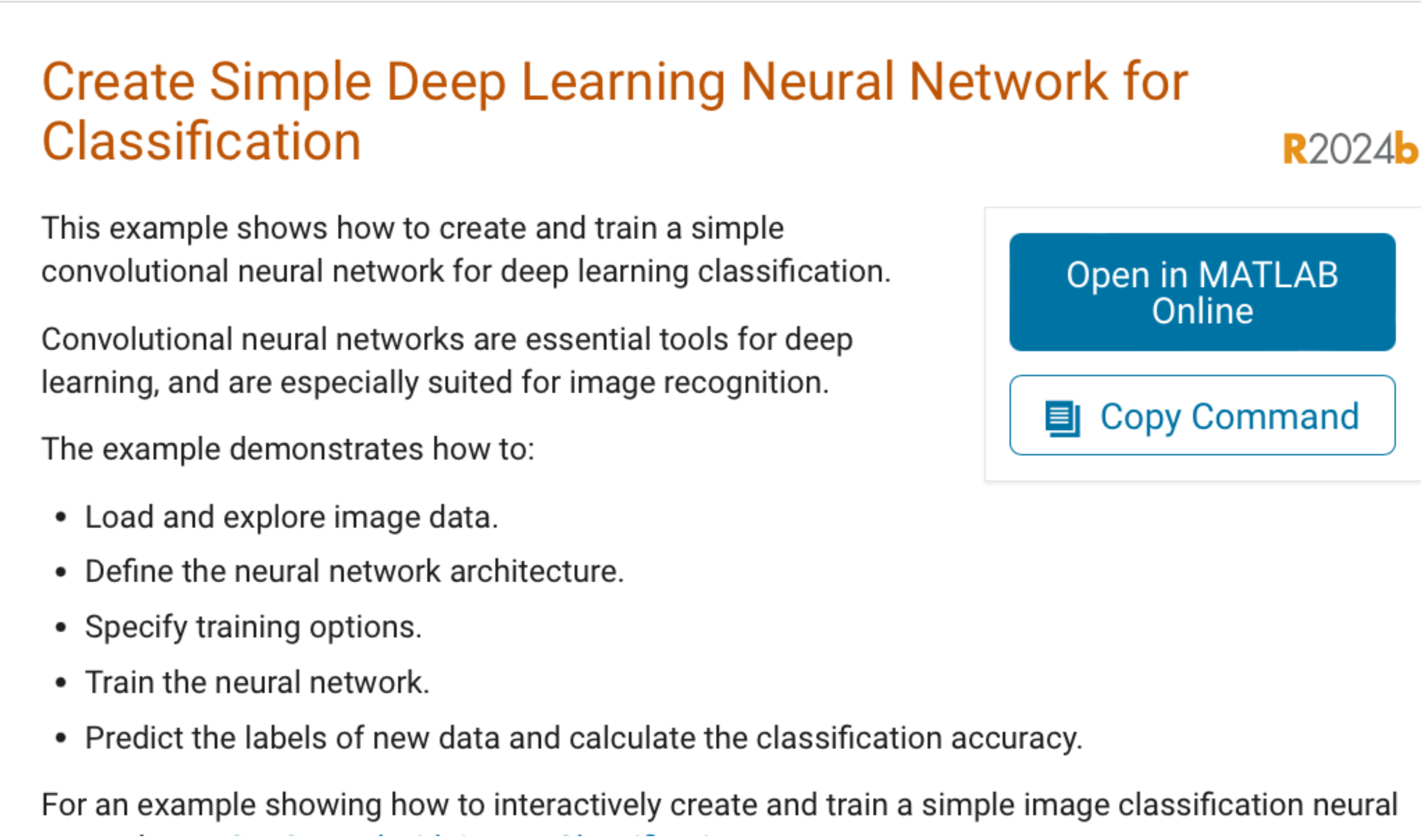
Open in MATLAB Online

This example uses:

Deep Learning Toolbox

Computer Vision Toolbox

Copy Command



Create Simple Deep Learning Neural Network for Classification

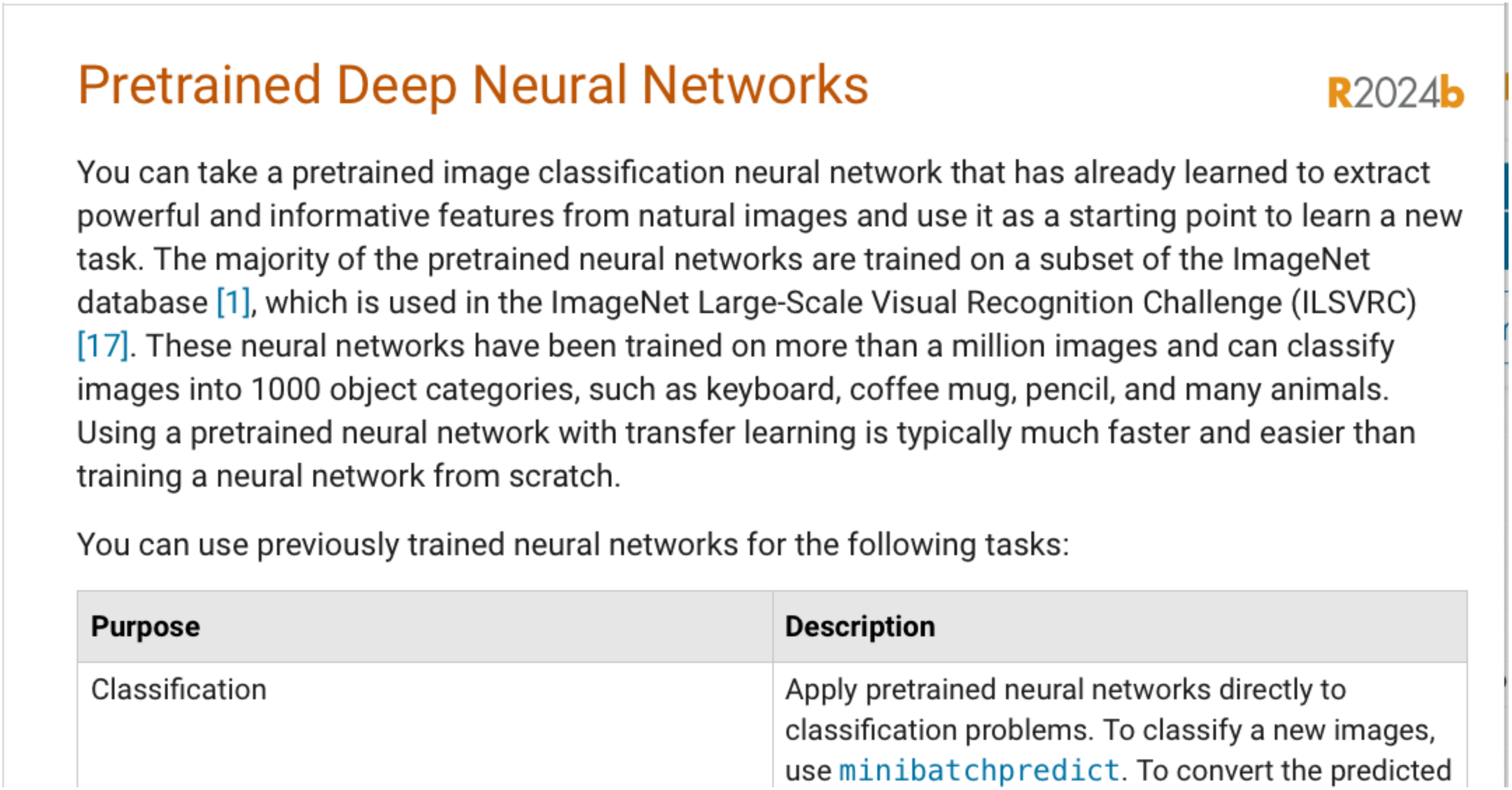
This example shows how to create and train a simple convolutional neural network for deep learning classification.

Convolutional neural networks are essential tools for deep learning, and are especially suited for image recognition.

The example demonstrates how to:

- Load and explore image data.
- Define the neural network architecture.
- Specify training options.
- Train the neural network.
- Predict the labels of new data and calculate the classification accuracy.

For an example showing how to interactively create and train a simple image classification neural



Pretrained Deep Neural Networks

R2024b

You can take a pretrained image classification neural network that has already learned to extract powerful and informative features from natural images and use it as a starting point to learn a new task. The majority of the pretrained neural networks are trained on a subset of the ImageNet database [1], which is used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [17]. These neural networks have been trained on more than a million images and can classify images into 1000 object categories, such as keyboard, coffee mug, pencil, and many animals. Using a pretrained neural network with transfer learning is typically much faster and easier than training a neural network from scratch.

You can use previously trained neural networks for the following tasks:

Purpose	**Description**
Classification	Apply pretrained neural networks directly to classification problems. To classify a new images, use `minibatchpredict`. To convert the predicted