Fruit Fly Optimization Algorithm

繼基因演算法、蟻群演算法與粒子群演算法之後，一種全新的群體智能演算法技術。本書不僅適合教授及老師撰寫學術期刊使用，亦適合老師教授演化式計算課程使用。

果蠅最佳化演算法

最新演化式計算技術

(Second Edition)

Fruit Fly Optimization Algorithm

(Using MATLAB)

Wen-Tsao Pan

- Introduction Fruit Fly Optimization Algorithm
- Fruit Fly Optimization Algorithm for solving maxima and minima
- Z-SCORE model coefficients optimized - for example in financial warning
- FOA Optimized general regression neural network - Web auction logistics service as example
- FOA Optimized Grey Model Neural Network - Fund trading decisions as an example
- FOA Optimized Support vector regression – Research mainland students studying in Taiwan for exampl
- Modified Fruit Fly Optimization Algorithm
- Adaptive Chaos Fruit Fly Optimization Algorithm

Personal Information

Wen-Tsao Pan

PhD in management

(SCI, SSCI) International Journal Referee

E_mail: teacherp0162@yahoo.com.tw

Phone: (00886) 955330842,QQ:324827824

Major

Evolutionary Computation
Data Mining
Neural Network
Marketing Research
Economic Modeling

Publication

Wen-Tsao Pan, “A new fruit fly optimization algorithm: Taking the financial distress model as an example”, Knowledge-Based Systems, Vol.26, pp.69-74, 2012, (SCI).
Chien-Jen Huang, Peng-Wen Chen, Wen-Tsao Pan, Multi-stage Data Mining technique to build the forecast model for Taiwan stocks, Neural Comput & Applic, Online First™ 2011, (SCI)
Wen-Tsao Pan, “Combining PSO Cluster and Nonlinear Mapping algorithm to perform clustering performance analysis: take the enterprise financial alarming as example”, Quality & Quantity, Online First™ 2011 (will be published in Volume 45 Issue 6 it is October 2011), (SSCI, SCI).
Wen-Tsao Pan, “The use of genetic programming for the construction of a financial management model in an enterprise”, International Journal of Applied Intelligence, Online First™, 22 October 2010, (SCI).
Wen-Tsao Pan, “Combining Fuzzy Sammon Mapping and Fuzzy Clustering Approach to perform clustering effect analysis: take the banking service satisfaction as an example”, Expert Systems With Applications, Vol. 37, pp. 4139–4145, 2009,(SCI).
Etc…

In addition, readers use this book published, please refer to authors published in Knowledge-Based Systems The journal articles.
If your school (Hong Kong, Macao, Taiwan and the mainland) require authors "FOA" keynote speech, please contact the author.
If your school has a teacher job openings, do not forget to recommend authors, very grateful!!
Finally, please help recommend authors to participate in the "**Academic Research Award**" selection of your country, the authors thank you!

WeChat: teacherp0162
QQ: 324827824
Email: teacherp0162@yahoo.com.tw
TEL: +886-955330842

潘文超

Wen-Tsao Pan 2014. 02. 01 TAIWAN R.O.C

Table of Contents

Chapter1

Introduction Fruit Fly Optimization Algorithm

1.1 Evolutionary Computation and Swarm Intelligence

Evolutionary Computation is a shared noun, referring to "the survival of the fittest and the elimination of the unfit" of the Darwinian Theory, in this concept the evolutionary process of nature is practically simulated to establish computing modes, such as the Genetic Algorithms of Prof. Holland in the early stages [6]. However, more recently, the core of evolution began to be diverted to animal foraging behavior and group behavior, such as the Particle Swarm Optimization (PSO) of Prof. Eberhart and the Artificial Fish Swarm Algorithm (AFSA) proposed by Prof. Li of Chinese Mainland [5,9]. The two algorithms are developed from the foraging behaviors of animal populations, thus, they are called swarm intelligence algorithms by some scholars.。This author (Wen-Tsao Pan, 2011) this author inspired by the foraging behavior of Drosophila and proposed Fruit Fly Optimization Algorithm, Hope to contribute to the academic community, so that we can carry out this new algorithm practical research, increase the chance that we submit to the International SCI, SSCI journals accepted.

1.2 The basic concept of Fruit Fly Optimization Algorithm

The Fruit Fly Optimization Algorithm was invented by Prof. Pan, a scholar of Taiwan [10]. It is a new method for deducing global optimization based on the foraging behavior of the fruit fly. The sensory perception of the fruit fly is better than that of other species, especially the sense of smell and vision. The olfactory organ of a fruit fly can gather various smells from the air, and even a food source 40km away. Afterwards, the fruit fly flies to the food, uses its acute vision to find the food and where its fellows gather, and then it flies in that direction, as shown in Figure 1.2.1 and Figure 1.2.2.

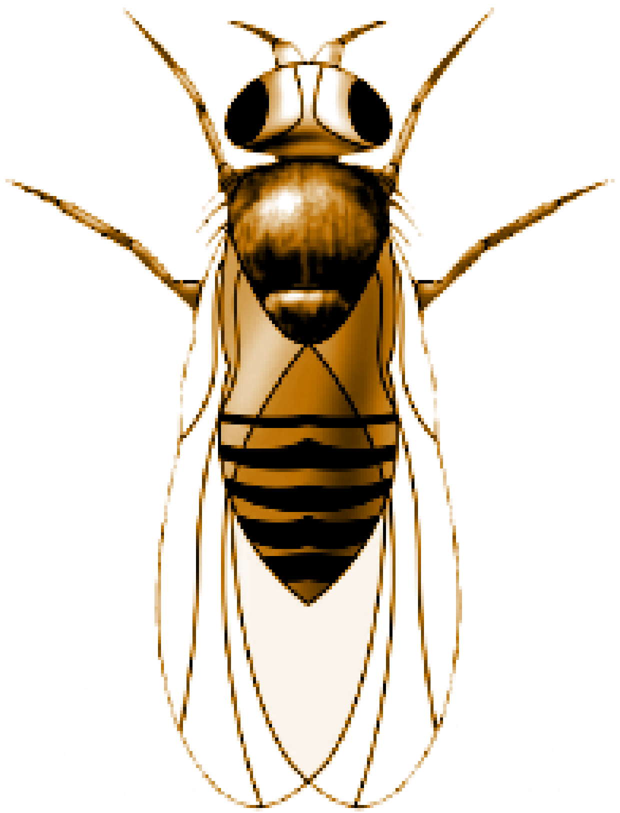


Figure 1.2.1 Drosophila body structure diagram

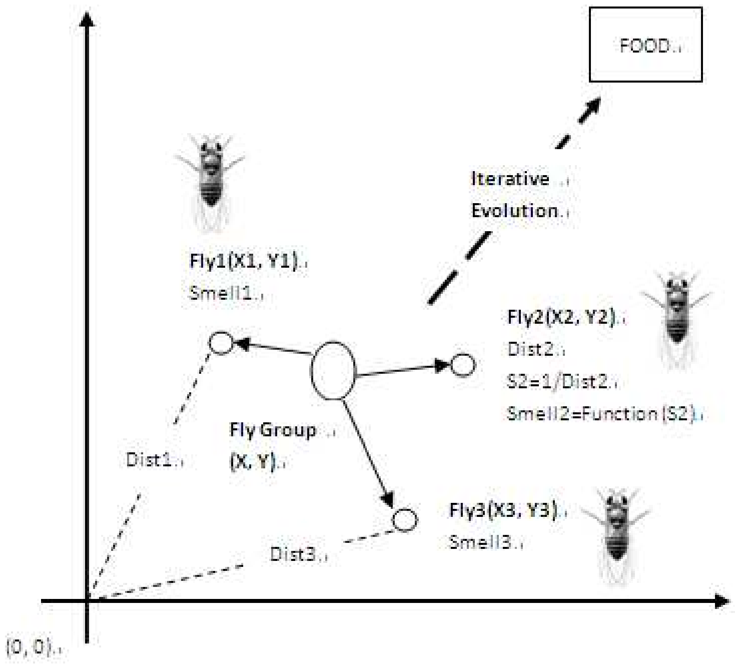


Figure 1.2.2 Schematic diagram of iterative search for food of fruit fly swarm

Fruit fly characteristics of searching for food are reduced to several necessary steps and procedure examples, as reference for readers. The steps are described, as below:

- The random initial position of a fruit fly swarm is as shown in the right of Figure 1.

  Init X_axis; Init Y_axis

- Random direction and distance of searching for food using the sense of smell of a fruit fly individual.

  X_i= X_axis + Random Value

  Y_i= Y_axis + Random Value

- As the location of food cannot be known, the distance (Dist) to the origin is estimated before the decision value of smell concentration (S) is calculated; this value is the reciprocal of distance.

$\text{Dist}_i = \sqrt{X_i^2 + Y_i^2}$; $S_i = \frac{1}{\text{Disti}}$

- The smell concentration decision value (S) is substituted in the smell concentration decision function (also known as the Fitness function) to work out the smell concentration (Smelli) in the position of the fruit fly individual.

Smell_i = Function(S_i)

- Determine the fruit fly with the maximum smell concentration among the fruit fly swarm (seek for the maximum value)

[bestSmell bestIndex] = max(Smell)

- Retain the best smell concentration value and x, y coordinates, here the fruit fly swarm flies toward the position by vision.

Smellbest = bestSmell

X_axis = X(bestIndex)

Y_axis = Y(bestIndex)

■ Enter into iterative optimization, repeat execution steps 2-5, and judge whether the smell concentration is better than the previous iterative smell concentration, if yes, execute Step 6.

1.3 Applications Fruit Fly Optimization Algorithm

Fruit Fly Optimization Algorithm (FOA) Belongs to the category of evolutionary computing, also belongs to the field of artificial intelligence.In the application is part of a research method, without any restrictions on the field, covered the various fields of military, engineering, medicine, management, finance and so on can be to use. In addition, they can be combined with other data mining techniques are used together, such as decision trees, Bayesian theorem, fuzzy math, gray system and neural network, and so on. Therefore, the use of highly flexible elastic, readers may need to be applied in accordance with their different areas, can be mixed with different algorithms. The book will be in a later chapter eleven demonstration FOA used to solve mathematical function pole planting, trimming Z-SCORE model coefficients and hybrid neural network used in various fields example, takes the reader understand the power of FOA, so that readers can easily use FOA study various problems. About the Fruit Fly Optimization Algorithm of resources,Readers can always check the latest information on the following website:

http://www.iLoveMatlab.cn

Readers can access at any time to see if there are advanced applications specialists update release and return an error message, the author will be trying to mix different data mining technology to provide reference. However, the author does not provide a large number of new applications of FOA to provide our readers can participate in, try mixing different data mining techniques on their own, and provide relevant code and the results published on the website so that the author makes everyone Jieneng get benefits. FOA has currently adopted by many scholars, under the brief cited several references to random search FOA journal for readers!!

- Hazim Iscan, Mesut Gunduz. (2014), Parameter analysis on fruit fly optimization algorithm, Journal of Computer and Communications.
- Ramachandran, B., Thomas Bellarmine, G. (2014), Improving observability using optimal placement of phasor measurement units, International Journal of Electrical Power and Energy Systems.
- Xie, A.-S., Yu, Y.-D., Huang, S.-M. (2014), Optimization algorithm based on benchmarking, Journal of Software.
- Pan, Q.-K., Sang, H.-Y., Duan, J.-H., Gao, L. (2014), An improved fruit fly optimization algorithm for continuous function optimization problems, Knowledge-Based Systems.
- Sandeep Kumar, Vineeta Bassi. (2014), Modified fruit fly min-max single depot vehicle routing algorithm (mff-vrp), Proceedings of 6th IRF International Conference, Bangalore, India, 01st June, 2014.
- Show, H. N., SKenneth S orensen, (2014), Metaheuristics - the Metaphor Exposed, University of Antwerp Operations Research Group ANT/OR.
- Anurag Rana and Ankur Sharma, (2014), Optimization of radial basis neural network by mean of amended fruit fly optimization algorithm, Journal of Computer and Mathematical Sciences.
- Yuan, X., Dai, X., Zhao, J., He, Q. (2014), On a novel multi-swarm fruit fly optimization algorithm and its application, Applied Mathematics and Computation.
- Xiao, C.-C., Hao, K.-R., Ding, Y.-S. (2014), An improved shuffled frog leaping algorithm for solving controlled optimization problems of water bath stretching slot, Journal of East China University of Science and Technology.
- Das, K.N., Singh, T.K. (2014), Drosophila food-search optimization, Applied Mathematics and Computation.

- Wei, L.-S., Wu, X., Niu, M.-Q., Chen, Z.-Y, (2014), FOA based PID controller for human balance keeping, Applied Mechanics and Materials.
- Dai, H., Zhao, G., Lu, J., Dai, S. (2014), Comment and improvement on "a new Fruit Fly Optimization Algorithm: Taking the financial distress model as an example", Knowledge-Based Systems.
- Mirjalili, S., Mirjalili, S.M., Lewis, A. (2014), Grey Wolf Optimizer, Advances in Engineering Software.
- Zheng, X.-L., Wang, L., Wang, S.-Y. (2014), A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem, Knowledge-Based Systems.
- Hou, J.-Y., Wang, B. (2014), A kind of diminishing step fruit fly optimization algorithm, Applied Mechanics and Materials.
- Liu, Z.-X., Wang, Y.-F., Zhang, Y. (2014), Multiple population fruit fly optimization algorithm for automatic warehouse order picking operation scheduling problem, Journal of Wuhan University of Technology.
- Li, Q., Cao, G.-H., Shan, D. (2014), A hybrid intelligent algorithm for optimal birandom portfolio selection problems, Mathematical Problems in Engineering.
- Zheng, X.-L., Wang, L., Wang, S.-Y. (2014), A hybrid discrete fruit fly optimization algorithm for solving permutation flow-shop scheduling problem, Control Theory and Applications.
- Niu, D.X., Chen, T.T., Wang, P., Chen, Y.C. (2014), Forecasting residential electricity based on FOAGMNN, Advanced Materials Research.
- Zhang, P., Wang, L. (2014), A grouped fruit-fly optimization algorithm for the no-wait lot streaming flow shop scheduling, Lecture Notes in Computer Science.
- Meng, A.-B., Chen, Y.-C., Yin, H., Chen, S.-Z. (2014), Crisscross optimization algorithm and its application, Knowledge-Based Systems.
- Hong, W.P., Liao, M.J. (2014), Application of fruit fly optimization algorithm-least square support vector machine in fault diagnosis of fans, Advanced Materials Research.
- Jiang, T., Wang, J.Z.(2014), Study on path planning method for mobile robot based on fruit fly optimization algorithm, Applied Mechanics and Materials.

- ◆ Ning, J., Wang, B., Li, H., Xu, B. (2014), Research on and application of diminishing step fruit fly optimization algorithm, Journal of Shenzhen University Science and Engineering
- ◆ Anurag Rana, Ankur Sharma, (2014), Resolving set-streaming stream-shop scheduling in distributed system by mean of an aFOA, International Journal of Computer Science & Engineering Technology.
- ◆ Peng Zhang, Ling Wang, (2014), Grouped fruit-fly optimization algorithm for the no-wait lot streaming flow shop scheduling, Intelligent Computing Methodologies.
- ◆ Xiaofang Yuana, Xiangshan Daia, Jingyi Zhaoa, Qian Heb, (2014), On a novel multi-swarm fruit fly optimization algorithm and its application, Applied Mathematics and Computation.
- ◆ LIU Chengzhong, HUANG Gaobao, ZHANG Renzhi, CHAI Qiang, (2014), Shuffled fruit fly optimization algorithm with local deep search, Journal of Computer Applications.
- ◆ He Zhen-Zong, Qi Hong, Yao Yu-Chen, Ruan Li-Ming, (2014), Inverse estimation of the particle size distribution using the Fruit Fly Optimization Algorithm, Applied Thermal Engineering.
- ◆ Wang Sheng, Yan Bao, (2013), Fruit fly optimization algorithm based fractional order fuzzy-PID controller for electronic throttle, Nonlinear Dynamics.
- ◆ Lin, S.-J., Chang, C., Hsu, M.-F, (2013), Multiple extreme learning machines for a two-class imbalance corporate life cycle prediction, Knowledge-Based Systems.
- ◆ Shi, Z., Miao, Y. (2013), Prediction research on the failure of steam turbine based on fruit fly optimization algorithm support vector regression, Advanced Materials Research.
- ◆ Lin, S.-M.(2013),Analysis of service satisfaction in web auction logistics service using a combination of Fruit fly optimization algorithm and general regression neural network, Neural Computing and Applications.
- ◆ Li, H.-Z., Guo, S., Li, C.-J., Sun, J.-Q.(2013), A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm, Knowledge-Based Systems.
- ◆ Zhang, Y., Fang, C. (2012), Combinatorial optimization using FOA and GA in futures market technical analysis, International Conference on Control, Automation and Systems , art.

- Li, H., Guo, S., Zhao, H., Su, C., Wang, B.(2012), Annual electric load forecasting by a least squares support vector machine with a fruit fly optimization algorithm, Energies.
- Han, J., Wang, P., Yang, X, (2012), Tuning of PID controller based on fruit fly optimization algorithm, 2012 IEEE International Conference on Mechatronics and Automation, ICMA 2012 , art.
- Yang, S.-C., Lee, C.-S., Lee, H.-S. (2012), Evaluation of logistic flow service satisfaction using the evolutionary computation technique and general regression neural network technique, International Journal of Advancements in Computing Technology.
- Li, C., Liu, X., Xu, S.(2012) [illegible] price prediction based on general regression neural network a[illegible]ptimization algorithm, Journal of Computational Information Systems.
- Li, C., Xu, S., Li, W., Hu, L.(2012), A novel modified fly optimization algorithm for designing the self-tuning proportional integral derivative controller, Journal of Convergence Information Technology.
- Wang, X.-G., Zou, Z.-J. (2012), Identification of ship manoeuvring response model based on fruit fly optimization algorithm, Dalian Haishi Daxue Xuebao/Journal of Dalian Maritime University.
- Abidin, Z.Z., Hamzah, M.S.M., Arshad, M.R., Ngah, U.K.(2012), A calibration framework for swarming ASVs' system design, Indian Journal of Marine Sciences.
- Shing-Chih Yang*, Chan-Shal Lee and Hsuan-Shih Lee, (2012), Construction of the prediction model of business operation performance in the electronic industry, E3 Journal of Business Management and Economics.
- Chang-Shu Tu Ching-Ter Chang, Kee-Kuo Chen, Hua-An Lu (2012), A study on business performance with the combination of Z-score and FOAGRNN hybrid model, African Journal of Business Management.
- Zhang Yuwen (2012), Smith predictor in the DDE application, Control and Decision Conference (CCDC), 2012 24th Chinese, pp.2346-2351.
- Xiao-long Zheng, Ling Wang, Sheng-yao Wang, A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem, Knowledge-Based Systems(KNOSYS-D-12-01212R1，Accept).

Problems and Applications

1. 1 What is swarm intelligence? There are those who belong to the field of algorithms? Made by th 1-8 :holars?
2. Figure 1.2.2 concentration should taste the optimal location of the bird flies, fruit flies that two directions toward the Drosophila fly?
3. flavor concentration in the sample program determines function represents what sense? Origin and flavor concentration determines the distance relationship between the value of what??

Chapter2

Fruit Fly Optimization Algorithm for solving maxima and minima

Note to reader: This chapter tells the reader how to deal with some optimization problems FOA, readers should carefully read the description of the sample program, In order to understand the FOA application skills.

2.1 Fruit Fly Optimization Algorithm for solving maxima

In this section, FOA is used to get maximal value with functions of respectively:

$$Y = 3 - X^2$$

and in the former, the solution of the maximal value is 3. The random initialization fruit fly swarm location zone is [0, 10], the random fly direction and distance zone of iterative fruit fly food searching is [-1, 1]. After 100 times of iterative search of minimal value and maximal value, the program implementation result will gradually approach the solution of the functional extreme value. Fig. 2.1.1 is the curve drawn by the solution of the extreme value of iterative search function. From the following figure, it can be seen that the curve gradually approaches a functional maximal value of 3, and the coordinate of the swarm of the fruit fly is (80.3296, 74.3351).

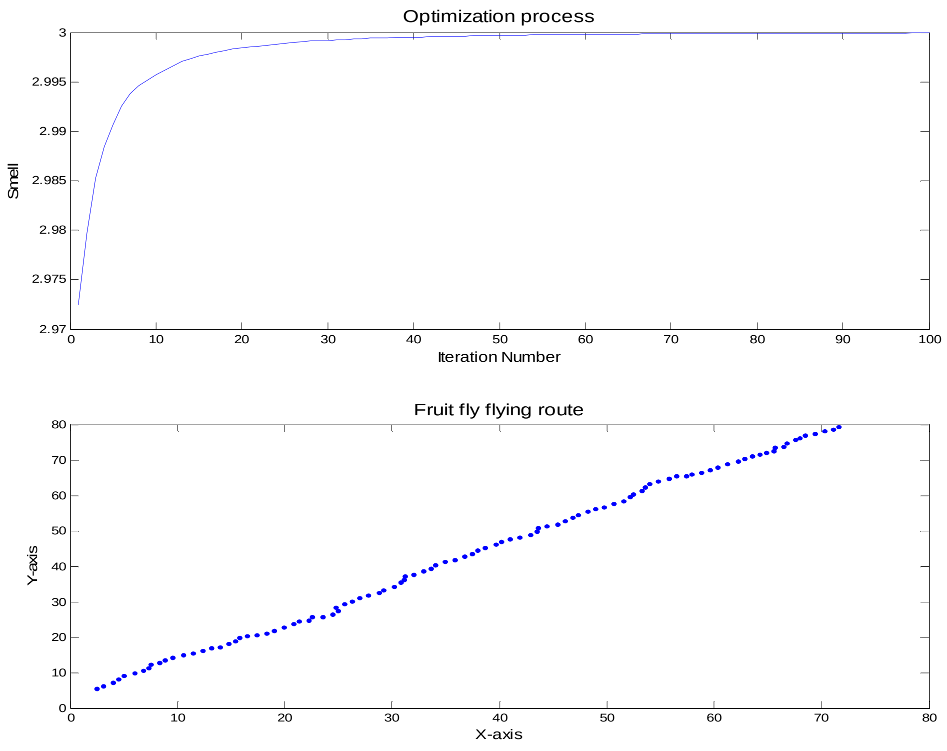


Figure 2.1.1 Curve of maximal value solution of an iterative search

solution

Programs code of the maximal value solution of an iterative search solution, with steps and sample programs, detailed below:

- Random initial fruit fly swarm location

```
X_axis=10*rand();
Y_axis=10*rand();
```

- Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;
```

- Since the food location cannot be known, the distance to the origin is thus estimated first (Dist), then the smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

```
D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);
```

- Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
Smell(i)=3-S(i)^2;
```

- Find out the fruit fly with maximal smell concentration (finding the maximal value) among the fruit fly swarm.

```
[bestSmell bestindex]=max(Smell);
```

- Keep the best smell concentration value and x, y coordinate, and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;
```

- Enter iterative optimization to repeat the implementation of steps 2–5, then judge if the smell concentration is superior to the previous iterative smell concentration, if so, implement step 6.

The program code follows:

```
%*** Empty Memory

clc
clear

%*** Random initial fruit fly swarm location

X_axis=10*rand();
Y_axis=10*rand();

%*** Set parameters

maxgen=100;    % Iterations
sizepop=20;    % Population size

%*** Optimization started, use the sense of smell to find food.

for i=1:sizepop

%*** Give the random direction and distance for the search of food using

osphresis by an individual fruit fly.

X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;

%***     Since the food location cannot be known, the distance to the origin is

thus estimated first (Dist), then the smell concentration judgment value (S) is

calculated, and this value is the reciprocal of distance.
```

```
D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.
Smell(i)=3-S(i)^2;
end

%*** Find out the fruit fly with maximal smell concentration (finding the maximal value) among the fruit fly swarm.
[bestSmell bestindex]=max(Smell);

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;

%*** Iterative optimization start
for g=1:maxgen

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly.
for i=1:sizepop
X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;
```

%*** Since the food location cannot be known, the distance to the origin is thus estimated first (Dist), then the smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

```
D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);
```

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
Smell(i)=3-S(i)^2;
end
```

%*** Find out the fruit fly with maximal smell concentration (finding the maximal value) among the fruit fly swarm.

```
[bestSmell bestindex]=max(Smell);
```

%*** Determine whether the smell concentration better than the previous iteration of the concentration, if yes then keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
if bestSmell>Smellbest
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;
```

```
end
```

%*** Each iteration the Smell optimal value, record to an array yy.

```
yy(g)=Smellbest;
Xbest(g)=X_axis;
Ybest(g)=Y_axis;
end
```

%***Draw smell concentration of each iteration

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('Smell','fontsize',12);
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
```

2.2 Fruit Fly Optimization Algorithm for solving minima

In this section, FOA is used to get minimal value with functions of respectively:

$$Y = -5 + X^2$$

and in the former, the solution of minimal value is -5, The random initialization fruit fly swarm location zone is [0, 10], the random fly direction and distance zone of iterative fruit fly food searching is [-1, 1]. After 100 times of iterative search of minimal value, the program implementation result will gradually approach the solution of the functional extreme value. Fig. 2.2.1 is the curve drawn by the solution of the extreme value of iterative search function. From the following figure, it can be seen that the curve gradually approaches a functional minimal value of -5, and the coordinate of the swarm of the fruit fly is (46.9049, 82.8723).

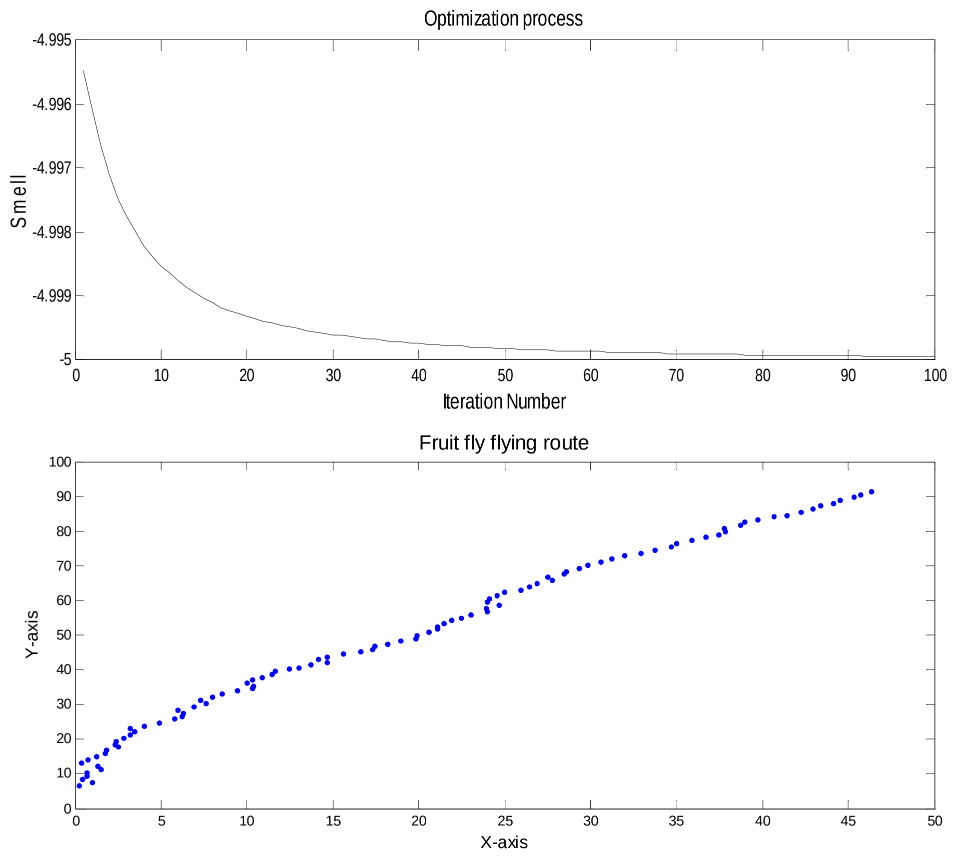


Figure 2.2.1 Curve of minimal value solution of an iterative search solution

Programs code of the minimal value solution of an iterative search solution, with steps and sample programs, detailed below::

- Random initial fruit fly swarm location

```
X_axis=10*rand();
Y_axis=10*rand();
```

- Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
X(i)=X_axis+2*rand()-1;
```

```
Y(i)=Y_axis+2*rand()-1;
```

- Since the food location cannot be known, the distance to the origin is thus estimated first (Dist), then the smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

```
D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);
```

- Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
Smell(i)=-5+S(i)^2;
```

- Find out the fruit fly with minimal smell concentration (finding the minimal value) among the fruit fly swarm.

```
[bestSmell bestindex]=min(Smell);
```

- Keep the best smell concentration value and x, y coordinate, and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;
```

- Enter iterative optimization to repeat the implementation of steps 2–5, then judge if the smell concentration is superior to the previous iterative smell concentration, if so, implement step 6.

The program code follows:

```
%*** Empty Memory

clc
clear
```

```
%*** Random initial fruit fly swarm location
X_axis=10*rand();
Y_axis=10*rand();

%*** Set parameters
maxgen=100;    % Iterations
sizepop=20;    % Population size

%*** Optimization started, use the sense of smell to find food.
for i=1:sizepop

%*** Give the random direction and distance for the search of food using
osphresis by an individual fruit fly.
X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;

%***     Since the food location cannot be known, the distance to the origin is
thus estimated first (Dist), then the smell concentration judgment value (S) is
calculated, and this value is the reciprocal of distance.
D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);

%*** Substitute smell concentration judgment value (S) into smell
concentration judgment function (or called Fitness function) so as to find the
smell concentration (Smelli) of the individual location of the fruit fly.
Smell(i)=3-S(i)^2;
```

```
end

%*** Find out the fruit fly with minimal smell concentration (finding the
minimal value) among the fruit fly swarm.
[bestSmell bestindex]=min(Smell);

%*** Keep the best smell concentration value and x, y coordinate , and at this
moment, the fruit fly swarm will use vision to fly towards that location.
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;

%*** Iterative optimization start
for g=1:maxgen

%*** Give the random direction and distance for the search of food using
osphresis by an individual fruit fly.
for i=1:sizepop
X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;

%***     Since the food location cannot be known, the distance to the origin is
thus estimated first (Dist), then the smell concentration judgment value (S) is
calculated, and this value is the reciprocal of distance.
D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);
```

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
Smell(i)=3-S(i)^2;
end
```

%*** Find out the fruit fly with minimal smell concentration (finding the minimal value) among the fruit fly swarm.

```
[bestSmell bestindex]=min(Smell);
```

%*** Determine whether the smell concentration better than the previous iteration of the concentration, if yes then keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
if bestSmell>Smellbest
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;
end
```

%*** Each iteration the Smell optimal value, record to an array yy.

```
yy(g)=Smellbest;
Xbest(g)=X_axis;
Ybest(g)=Y_axis;
end
```

```
%***Draw smell concentration of each iteration
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('Smell','fontsize',12);
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
```

2.3 FOA for solving global maxima capacity analysis

This section tries to use FOA search global maxima value, the function is:

y=sin(x)./x

Graph of the function shown in Fig. 2.3.1:

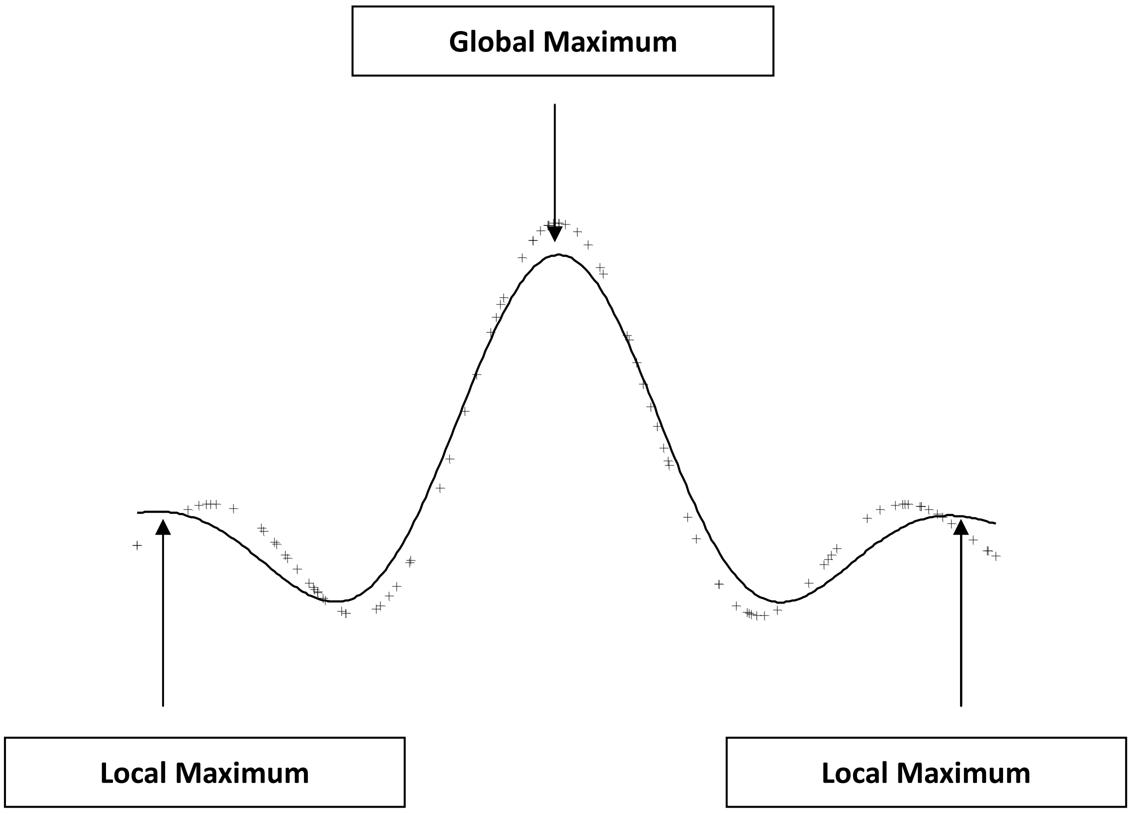
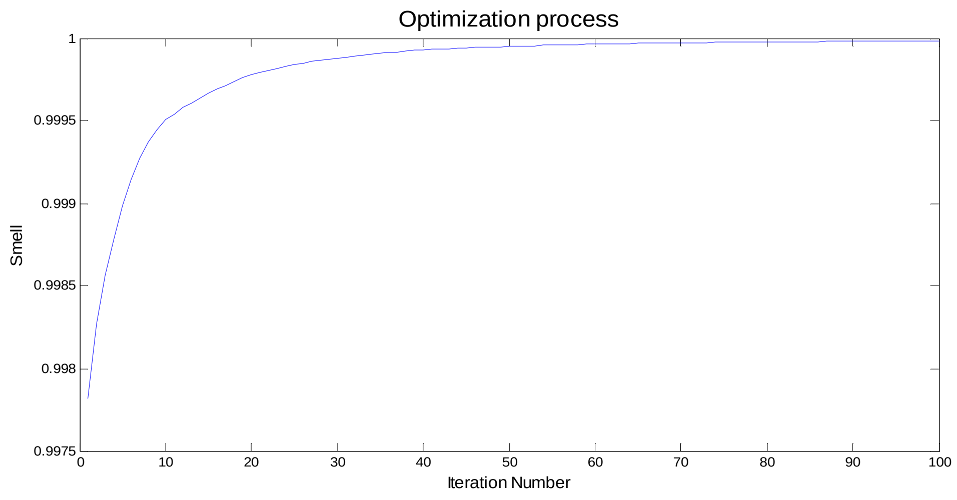


Figure 2.3.1 Graph of the function

Through this graphic found, this function includes about two local maxima and global maxima located in the middle of this graph. The global maximum of the function was 1. This section will attempt to FOA search, understanding FOA is able to escape local maxima, and the search to the global maxima. The random initialization fruit fly swarm location zone is [0, 10], the random fly direction and distance zone of iterative fruit fly food searching is [-1, 1], population size is 20. After 100 times of iterative search of maxima value, the program implementation result will gradually approach the solution of the functional extreme value.。Fig. 2.3.2 is the curve drawn by the solution of the extreme value of iterative search function. From the following figure, it can be seen that the curve gradually approaches a functional maxima value of 1, ，Thus demonstrating FOA can escape local extreme values and find the global extreme values. The figure below shows the path of fruit fly populations in search of food chart, through Figure found that the coordinate of the swarm of the fruit fly is (67.3851, 86.5945)。



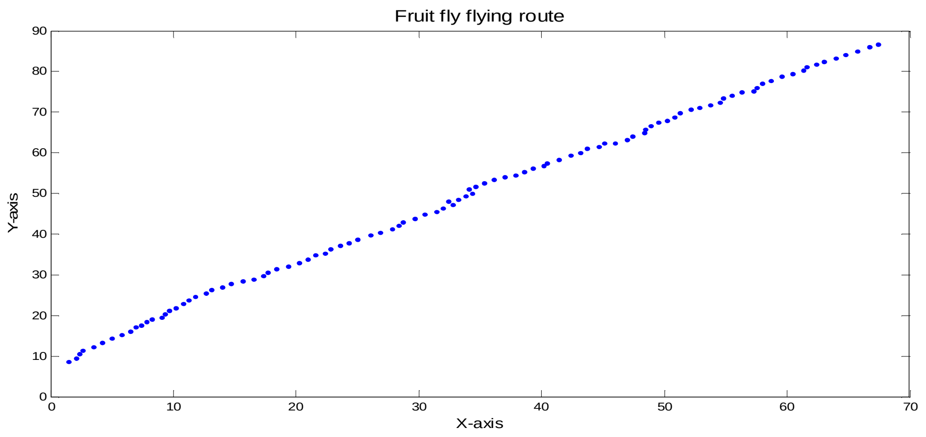


Figure 2.3.2 Flight routes of fruit fly swarm

The program code follows::

%*** Empty Memory

```
clc
clear
```

%*** Random initial fruit fly swarm location

```
X_axis=10*rand();
Y_axis=10*rand();
```

%*** Set parameters

```
maxgen=100;    % Iterations
sizepop=20;    % Population size
```

%*** Optimization started, use the sense of smell to find food.

```
for i=1:sizepop
```

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
X(i)=X_axis+2*rand()-1;
```

Y(i)=Y_axis+2*rand()-1;

%*** Since the food location cannot be known, the distance to the origin is thus estimated first (Dist), then the smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

Smell(i)=sin(S(i))./S(i);
end

%*** Find out the fruit fly with maximal smell concentration (finding the maximal value) among the fruit fly swarm.

[bestSmell bestindex]=max(Smell);

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;

%*** Iterative optimization start

for g=1:maxgen

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
for i=1:sizepop
X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;
```

%***   Since the food location cannot be known, the distance to the origin is thus estimated first (Dist), then the smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

```
D(i)=(X(i)^2+Y(i)^2)^0.5;
S(i)=1/D(i);
```

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
   Smell(i)=sin(S(i))./S(i);
   end
```

%*** Find out the fruit fly with maximal smell concentration (finding the maximal value) among the fruit fly swarm.

```
   [bestSmell bestindex]=max(Smell);
```

%*** Determine whether the smell concentration better than the previous iteration of the concentration, if yes then keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
if bestSmell>Smellbest
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;
end
```

%*** Each iteration the Smell optimal value, record to an array yy.

```
yy(g)=Smellbest;
Xbest(g)=X_axis;
Ybest(g)=Y_axis;
end
```

%***Draw smell concentration of each iteration

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('Smell','fontsize',12);
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
```

Problems and Applications

1. How to distinguish between maxima value and minima value of the smell concentration judgment function?? How to keep the best smell concentration value?
2. Please use FOA try to Coding a program code to search the function $Y = 6 – X^2$, $Y = -7 + X^2$
3. Please explain the significance of Figure 2.3.2

Notes column

Chapter3

Z-SCORE models coefficients optimized - for example in financial warning

3.1 Introduction Financial Z-SCORE model

In order to investigate whether the investors' holdings and the stocks they are going to invest will face any risks, an American scholar, Professor Altman proposed the Z-SCORE financial warning model for people to examine. [6] According to Altman's long-term empirical research, he chose 5 different financial indicators and gave different multipliers. We only need to apply the listed companies' annual financial information to the formula, and we can get the risk profile of a company. Altman's research was called The Z-SCORE models. The Z-SCORE formula and five financial indicators are listed below:

$$Z = 1.2\ X1 + 1.4\ X2 + 3.3\ X3 + 0.6\ X4 + 1.0\ X5$$

Z=Z-SOCRE. The definitions of X1 to X5 are:

X1：Working capital / total assets
X2：Retained Earnings / Total Assets
X3：Pre-tax Net Income / Total Assets
X4：Shareholders Equity / Total liabilities
X5：Sales Amount / Total Assets

According to the aforementioned formulas, Altman concluded that basically a financially sound listed company's Z-SCORE should be higher or equal 2.675. A company with financial crisis would get Z-SCORE lower or equal 2.675. Through this financial risk detecting model, readers can easily detect financial condition of enterprises. In the next section in this chapter, 20 companies' financial warning will be conducted under this model..

3.2 using Z-SCORE models - for example in financial warning

In this section, we are going to see the example of 20 enterprises conducting financial risk warning. The sample data are as follows. The independent variables are X1 to X5, and the dependent variable is Y ("1" represents a normal company; "0" represents a risky company). In addition, 0.5 is corresponding to the Z-SCORE model determining value 2.675. If the predicting outcome of the FOA Z-SCORE is higher than 0.5, (that is to say, higher than 2.675), the company is normal.

Table 3.2.1 Sample data 20 companies

No	X1	X2	X3	X4	X5	Y
1	0.016	0.177	0.4208	1.038	0.022	0
2	1.0957	0.19	0.2224	0.816	0.933	1
3	0.2543	0.206	0.5264	0.4	0.015	1
4	1.2257	0.224	0.3272	1.05	1.049	1
5	0.3872	0.228	0.2256	0.98	0.998	1
6	1.6066	0.231	0.2832	1.054	1.009	1
7	1.1594	0.252	0.3344	0.946	0.987	0
8	0.3424	0.26	0.3408	0.196	0.126	1
9	0.8604	0.261	0.2616	0.994	0.996	0
10	0.5107	0.264	0.2352	0.888	0.004	1
11	0.8981	0.271	0.1536	0.688	0.857	1
12	0.0878	0.275	0.2072	0.732	0.908	1
13	0.2384	0.297	0.2336	0.036	0.099	0
14	0.8591	0.308	0.4208	0.132	1.031	1
15	0.5861	0.309	0.1656	0.836	0.896	1
16	0.6329	0.311	-0.24	-0.134	0.611	0
17	0.6173	0.311	0.1256	0.714	0.848	0
18	0.7811	0.326	0.2896	0.834	0.909	1
19	0.3937	0.329	0.2552	0.024	0.493	0
20	0.1211	0.055	0.3744	1.142	1.077	1

Use the following sample data input ZSCORE models:

$Z = 1.2\ X1 + 1.4\ X2 + 3.3\ X3 + 0.6\ X4 + 1.0\ X5$

The output can be obtained:

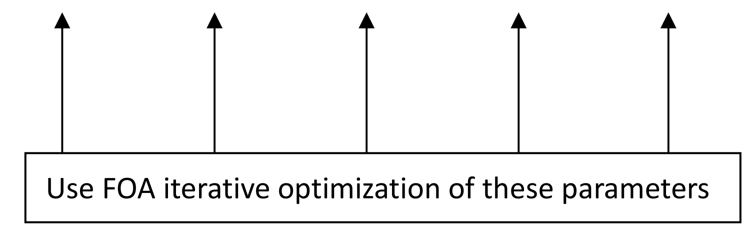
```
2.3004  3.7374  2.5857  4.5432  3.1143
4.8273  4.4022  2.1431  3.8536  2.2954
3.2338  2.5213  1.5934  3.9610  3.0800
0.9335  2.8670  3.7588  2.2826  3.2200
```

Based on Altman’s definition, a financially sound listed company should score higher or equal 2.675. A company with financial crisis would get Z-SCORE lower or equal 2.675. Table 3.2.1 shows that only companies 3,8,9,10,12 and 17 are correct. Hence, the financial crisis warning system of Z-SCORE still has a lot to be improved. In the author’s humble opinion, it was probably due to the different spatio-temporal background that led to the decreasing ability of that model. If we can optimize the Z-SCORE coefficients, the predicting ability could be improved. In the next section, we will replace coefficients of the independent variables in optimized Z-SCORE model with FOA, hoping that the readers can have a better idea of how to apply FOA in case study.

3.3 FOA optimized Z-SCORE and implementation of the financial warning

This section will be optimized Z-SCORE model and using Previous section only 20 companies on a financial crisis warning as a demonstration. Practices are as follows:

$$Z = p1 * X1 + p2 * X2 + p3 * X3 + p4 * X4 + p5 * X5$$



First, initialize the 5 groups of fly populations and were assigned to the parameters (p1, p2, p3, p4 and p5). Each group has 10 fruit flies, the random initialization fruit fly swarm location zone is [0, 1], the random fly direction and distance zone of iterative fruit fly food searching is [-1, 1]. After 100 times of iterative search of p1, p2, p3, p4 and p5, Z-SCORE predictive value gradually approaches the target Y. Figure 3.3.1show that iterative optimization parameters Z-SCORE, RMSE convergence trend between the predicted value and the target value. Figure 3.3.2 show the Five groups of flies swarm optimization flight trajectory. After FOA iterative optimization, the 39th generation of convergence, RMSE value is 0.1248 and the Best p1, p2, p3, p4 and p5 parameter values are 0.1875,0.2705,0.2153,0.263 and 0.1912. Readers can use this five parameters into Z-SCORE model to predict, If the prediction is higher than 0.5 for the normal company and observe whether the predictive ability of Z-SCORE has improved significantly.

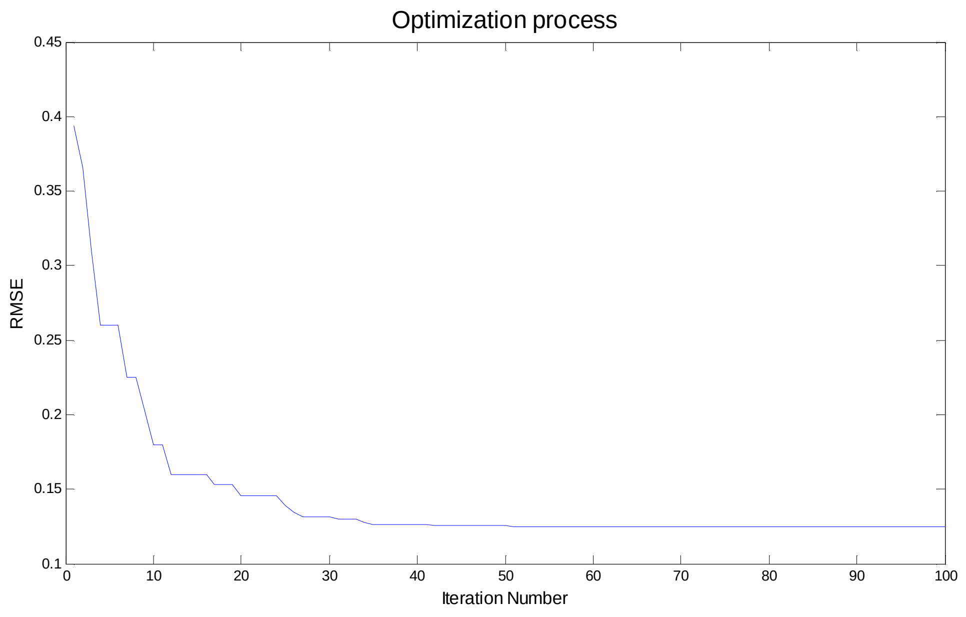


Figure 3.3.1 RMSE convergence trend

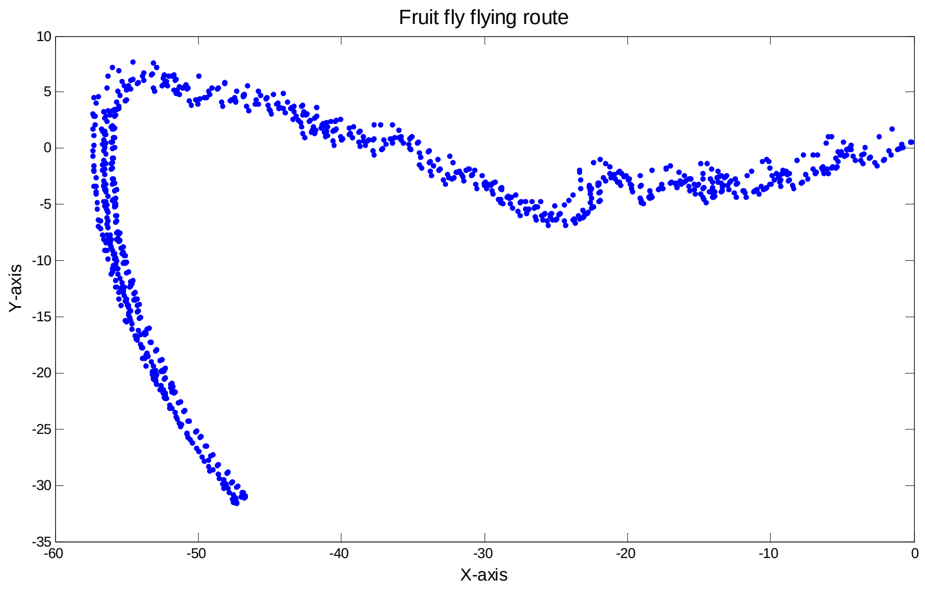


Figure 3.3.2 5 Flight routes of the fruit fly swarm

FOAZSCORE program as follows:

The program code follows:

```
%*** Empty Memory

clc
clear
load c:\TXY.txt;
[row,col]=size(TXY);
set=row/5;
row1=row-set;
tr=TXY(1:row1,1:col-1);
t1=TXY(1:row1,col);
te=TXY(row1+1:row,1:col-1);
t2=TXY(1:row1,col);

%*** Random initial fruit fly swarm location

X_axis=rands(1,5);
Y_axis=rands(1,5);
g=0;
maxgen=100;    % Iterations
sizepop=5;    % Population size

%*** Optimization started, use the sense of smell to find food.

for p=1:sizepop

%*** Give the random direction and distance for the search of food using

osphresis by an individual fruit fly.

X(p,:)=X_axis+2*rand()-1;
Y(p,:)=Y_axis+2*rand()-1;
```

%*** Calculating the distance from the origin

```
D(p,1)=(X(p,1)^2+Y(p,1)^2)^0.5;
D(p,2)=(X(p,2)^2+Y(p,2)^2)^0.5;
D(p,3)=(X(p,3)^2+Y(p,3)^2)^0.5;
D(p,4)=(X(p,4)^2+Y(p,4)^2)^0.5;
D(p,5)=(X(p,5)^2+Y(p,5)^2)^0.5;
```

%** The smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.。

```
S(p,1)=1/D(p,1);
S(p,2)=1/D(p,2);
S(p,3)=1/D(p,3);
S(p,4)=1/D(p,4);
S(p,5)=1/D(p,5);
```

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
a1=S(p,1);
a2=S(p,2);
a3=S(p,3);
a4=S(p,4);
a5=S(p,5);
```

%*** Calculate RMSE, like fitness function.

```
yc=a1*tr(:,1)+a2*tr(:,2)+a3*tr(:,3)+a4*tr(:,4)+a5*tr(:,5);
y=yc-t1;% Calculated RMSE between ZSCORE output value and the target value
for ii=1:row1
g=g+y(ii)^2;
end
```

```
Smell(p)=(g/row1) ^0.5;
end
Smell
[bestSmell bestindex]=min(Smell)
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
X_axis=X(bestindex,:);
Y_axis=Y(bestindex,:);
bestS=S(bestindex,:);
Smellbest=bestSmell
```

%*** Iterative optimization start

```
for gen=1:maxgen
g=0;
gen
```

%*** Use smell to find food

```
for p=1:sizepop
```

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
X(p,:)=X_axis+2*rand()-1;
Y(p,:)=Y_axis+2*rand()-1;
```

%***   Since the food location cannot be known, the distance to the origin is thus estimated first (Dist).

```
D(p,1)=(X(p,1)^2+Y(p,1)^2)^0.5;
D(p,2)=(X(p,2)^2+Y(p,2)^2)^0.5;
```

```
D(p,3)=(X(p,3)^2+Y(p,3)^2)^0.5;
D(p,4)=(X(p,4)^2+Y(p,4)^2)^0.5;
D(p,5)=(X(p,5)^2+Y(p,5)^2)^0.5;

%*** Then the smell concentration judgment value (S) is calculated, and this

value is the reciprocal of distance.
S(p,1)=1/D(p,1);
S(p,2)=1/D(p,2);
S(p,3)=1/D(p,3);
S(p,4)=1/D(p,4);
S(p,5)=1/D(p,5);

%*** Substitute smell concentration judgment value (S) into smell

concentration judgment function (or called Fitness function) so as to find the

smell concentration (Smelli) of the individual location of the fruit fly.
a1=S(p,1);
a2=S(p,2);
a3=S(p,3);
a4=S(p,4);
a5=S(p,5);

%*** Calculate RMSE, like fitness function.
yc=a1*tr(:,1)+a2*tr(:,2)+a3*tr(:,3)+a4*tr(:,4)+a5*tr(:,5);
y=yc-t1;% Calculated RMSE between ZSCORE output value and the target value
for ii=1:row1
g=g+y(ii)^2;
end
Smell(p)= (g/row1) ^0.5;
end
Smell
[bestSmell bestindex]=min(Smell)
```

```
%*** Keep the best smell concentration value and x, y coordinate , and at this

moment, the fruit fly swarm will use vision to fly towards that location.
if bestSmell<Smellbest
X_axis=X(bestindex,:);
Y_axis=Y(bestindex,:);
bestS=S(bestindex,:);
besta1=a1;
besta2=a2;
besta3=a3;
besta4=a4;
besta5=a5;
Smellbest=bestSmell
end
%*** Each generation best value record to the variable yy
    yy(gen)=Smellbest;
    Xbest(gen,:)=X_axis;
    Ybest(gen,:)=Y_axis;
end


%*** Draw the optimization process of trends
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('RMSE','fontsize',12);
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
besta1
besta2
besta3
besta4
besta5
```

Problems and Applications

1. In this section of code, how many groups of fly populations initially? Z-SCORE model coefficients which can be adjusted? Use FOAZSCORE search result 5 parameters into the Z-SCORE model to predict, and Observe the predicted results。
2. Please describe Figure 3.3.1 and Figure 3.3.2 What does it mean?
3. Fixed original coefficient, sorting out the financial and accounting related topics。Use FOA find the best combination of financial variables (ie. Find the Z-SCORE model X1,X2,X3,X4 and X5)。

Chapter4

FOA Optimized general regression neural network - Web auction logistics service as example

Note to reader: This chapter describes the mixed use of FOA and Neural Network, a useful tool for writing SCI and SSCI journals. Because FOA is a brand new research method, many editors and reviewers find it interesting. Therefore, make good use of FOA and turn it into the selling points of your paper! Please read the following chapters closely, it would be very useful. (p.s. If your paper is accepted, don’t forget to treat me a dinner!)

4.1 Introduction of general regression neural network

Specht(1990) proposed Probabilistic Neural Networks structure, which is a monitoring type network structure; its theory is built on Bayes decision and nonparametric technique to predict Probability Density Function(PDF) and this Probability Density Function is of the form of Gaussian Distribution. Yeh (2001) pointed out that this function is as shown in formula (1).

$$f_k(X) = \left(\frac{1}{N_k}\right)\left(\frac{1}{(2\pi)^{m/2}}\right)\left(\frac{1}{\sigma^m}\right)\sum_{j=1}^{N_k}\exp\left(-\frac{\|X - X_{kj}\|}{2\sigma^2}\right) \quad \text{..............................(1)}$$

Since Probabilistic Neural Networks is applicable to general classification problem with the assumption that the eigenvector to be classified must belong to one of the known classifications, then the magnitude of the absolute probabilistic value of each classification is not important, only the relative magnitude needs to be considered, then

$$\left(\frac{1}{(2\pi)^{m/2}}\right)\left(\frac{1}{\sigma^m}\right)$$

of formula (1) can be neglected and formula (1) can be simplified to be

$$f_k(X) = \left(\frac{1}{N_k}\right)\sum_{j=1}^{N_k}\exp\left(-\frac{\|X - X_{kj}\|}{2\sigma^2}\right) \quad \text{....................................(2)}$$

In formula (2), σ is the Smoothing Parameter of Probabilistic Neural Networks; after the completion of network training, the prediction accuracy can be enhanced through the adjustment of Smoothing Parameter σ; the larger this value is, it will approaches the function more smoothly. If Smoothing Parameter σ is not appropriately selected, the neuron number in the network design will be too much or too few and over-fitness or insufficient fitness will be resulted in the function approaching process, and the prediction power will then get reduced.

Let $d_{kj}^{2} = \|X - X_{kj}\|$

be the Euclid distance square between X and Xkj in the sample space, then formula (2) can be re-written as

$$f_k(X) = \left(\frac{1}{N_k}\right)\sum_{j=1}^{N_k}\exp\left(-\frac{1}{2}\left(\frac{d_{kj}}{\sigma}\right)^2\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots .(3)$$

The generalization capability of Probabilistic Neural Networks replies on the adjustment of Smoothing Parameter σ. In formula (1), when Smoothing Parameter σ approaches 0

$$f_k(X) = \frac{1}{N_k}$$

If X=Xkj , otherwise

$$f_k(X) = 0$$

At this moment, Probabilistic Neural Networks has its classification dependent on how close the unclassified sample is to the classified sample. When Smoothing Parameter σ approaches infinity

$$f_k(X) = 1$$

At this moment, Probabilistic Neural Networks gets close to blind classification. However, since Probabilistic Neural

Networks can only perform classification problem study, Specht (1991) uses Probabilistic Neural Networks to evolve General Regression Neural Network to solve the continuous variable problem. General Regression Neural Network not only can be used in the study of classification problem, but has very good prediction power on the construction of prediction model or control model and linear or nonlinear problem.

General Regression Neural Network is similar to Probabilistic Neural Networks and is a four layers neural network; the first layer is input layer, neuron number is the number of independent variable and receives input data; the medium second layer is hidden layer named Pattern Layer, which stores all the training data; here the output data of Pattern Layer will pass through the neuron of third layer of Summation Layer and get corresponded to each possible classification; meanwhile, this layer will perform the calculation of formula (3). The fourth layer is different than Probabilistic Neural Networks and this layer is Linear Layer, this layer performs the output weighted average calculation of Summation Layer and generates the output value.

4.2 Applications of general regression neural network

GRNN has two important functions command::

1. net = newgrnn(P,T,SPREAD), Used to train GRNN, Defined parameters:

P	- RxQ1 matrix of Q1 input vectors
T	- SxQ2 matrix of Q2 target class vectors
SPREAD	- Spread of radial basis functions, default = 1.0.

and returns a new generalized regression neural network.

net	- Training generate network architecture

2. Y = sim(net, P) Analog GRNN Model test output, Defined parameters:

net	- Training generate network architecture
P	- Test inputs
Y	- Network test output

This section tries to use the feature of XOR problem to train GRNN and test its predictive ability. In terms of XOR problem, 0 stands for sameness, and 1 stands for difference. In order to meet the requirements of GRNN, 1 represents sameness and -1 represents difference, as we change it. Sample data is as follows:

X1	X2	Y
0	0	1
0	1	-1
1	0	-1
1	1	1

Input XOR 4 rows data

```
>> X=[0 0; 0 1;1 0;1 1]
X =
0       0
0       1
1       0
1       1

>> Y=[1;-1;-1;1]
Y =
1
-1
-1
1

>> X=X'
X =
        0       0       1       1
        0       1       0       1

>> Y=Y'
Y =
        1      -1      -1       1

>> net=newgrnn(X,Y)
net =
Neural Network object:
architecture:
numInputs: 1
numLayers: 2
biasConnect: [1; 0]
inputConnect: [1; 0]
layerConnect: [0 0; 1 0]
…
…
```

```
…
>> X2=[1.1 0.1;0.2 0.1]
X2 =
1.1000      0.1000
0.2000      0.1000

>> X2=X2'
X2 =
1.1000      0.2000
0.1000      0.1000

>> yc=sim(net,X2)
yc =
-0.1064      0.0554
```

According XOR feature, we could anticipate that the result should be -1 and 1, but the actual test results are -0.1064 and 0.0554. Despite the great inaccuracy, the direction of the approach is correct. If we adjust the Spread parameter values of GRNN, the predictive ability of GRNN could be improved apparently. In the next section, we will use the satisfaction survey data of online shopping logistics service as the sample and adjust the Spread parameter values of GRNN by FOA to observe the optimization capabilities of FOA and the predictive ability of GRNN.

4.3 Optimizing GRNN parameters via FOA

In this section, we use FOA to obtain optimal GRNN parameter values as an example to illustrate the way how FOA dynamically adjust the Spread parameter values of GRNN and the construction of the sellers' satisfaction detection mode of online shopping logistics service. We also compare FOA with GRNN to see their difference in terms of their capacity of category detection. In order to respect the teacher's ownership of the data, this book captures only 40 question items, and use principal component analysis to generate 9 factors (X1-X9). The numerical values are normalized between 0 and 1 as the network input variables. Target (Y) is the overall logistics satisfaction question items. 0 represents satisfied (including three levels: very satisfied, satisfied, and neutral), and 1 represents unsatisfied (including two levels: dissatisfied and very dissatisfied). Because there are 1000 rows of data in total, we show only 20 rows of them in Table 4.3.1. The readers can simulate more data to test the predictive ability of FOAGRNN. However, please note that the GRNN is not suitable for small sample data. The authors also tested a small sample of the actual data, test results indeed larger inaccuracy. After increasing the sample data, the inaccuracy has been gradually improved, this feature is worthy for the reader's reference. The authors show 20 rows of sample data below, readers can enter data into Notepad (not including the serial number and title), save to C: \ drive root directory and name the file as TXY.txt. Parameter settings on FOA, the random initialization fruit fly swarm location zone is [0, 1], the random flying direction and distance zone of iterative fruit fly food searching is [-10,10], populations size is 10, and the iterative number is 100. Figure 4.3.1 is Flight routes of the fruit fly swarm.

Table 4.3.1 The 20 rows sample data of logistics seller satisfaction

No	X1	X2	X3	X4	X5	X6	X7	X8	X9	Y
1	0.3985	0.3984	0.3984	0.4752	0.4677	0.4263	0.3828	0.3745	0.3796	0
2	0.3988	0.3984	0.3984	0.4367	0.4200	0.4279	0.3949	0.3695	0.3738	0
3	0.3993	0.3983	0.3984	0.4675	0.4071	0.4448	0.4005	0.3886	0.3852	0
4	0.3994	0.3984	0.3984	0.8252	0.6736	0.4033	0.4144	0.3966	0.3973	1
5	0.3988	0.3984	0.3984	0.8280	0.6967	0.4047	0.4120	0.3823	0.3893	1
6	0.3988	0.3984	0.3984	0.9070	0.8585	0.4056	0.4130	0.4020	0.4006	1
7	0.4007	0.3984	0.3984	0.5262	0.5087	0.4057	0.4047	0.4017	0.4068	1
8	0.3987	0.3984	0.3984	0.7345	0.6975	0.4061	0.4174	0.4032	0.4024	1
9	0.3993	0.3984	0.3984	0.7745	0.7115	0.4072	0.4294	0.4055	0.4068	1
10	0.3991	0.3984	0.3984	0.5934	0.5738	0.4082	0.4037	0.3990	0.4000	1
…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…	…	…
991	0.3987	0.3984	0.3984	0.5763	0.5078	0.4106	0.4105	0.4003	0.4014	1
992	0.3988	0.3984	0.3984	0.5804	0.5053	0.4118	0.4026	0.3871	0.3919	1
993	0.3987	0.3984	0.3984	0.5211	0.4645	0.4138	0.4043	0.3851	0.3961	0
994	0.3989	0.3984	0.3984	0.5335	0.5172	0.4146	0.4024	0.3935	0.3992	1
996	0.3992	0.3984	0.3984	0.6425	0.6174	0.4151	0.4185	0.4096	0.4082	1
996	0.3994	0.3984	0.3984	0.5090	0.4678	0.4160	0.3768	0.3725	0.3787	0
997	0.3987	0.3984	0.3984	0.5406	0.5140	0.4178	0.4009	0.3950	0.3931	1
998	0.3989	0.3984	0.3984	0.5187	0.5005	0.4194	0.4268	0.4180	0.4178	1
999	0.3986	0.3984	0.3984	0.4970	0.4547	0.4206	0.3845	0.3740	0.3737	0
1000	0.4010	0.3984	0.3984	0.4616	0.4331	0.4256	0.4080	0.3955	0.3945	0

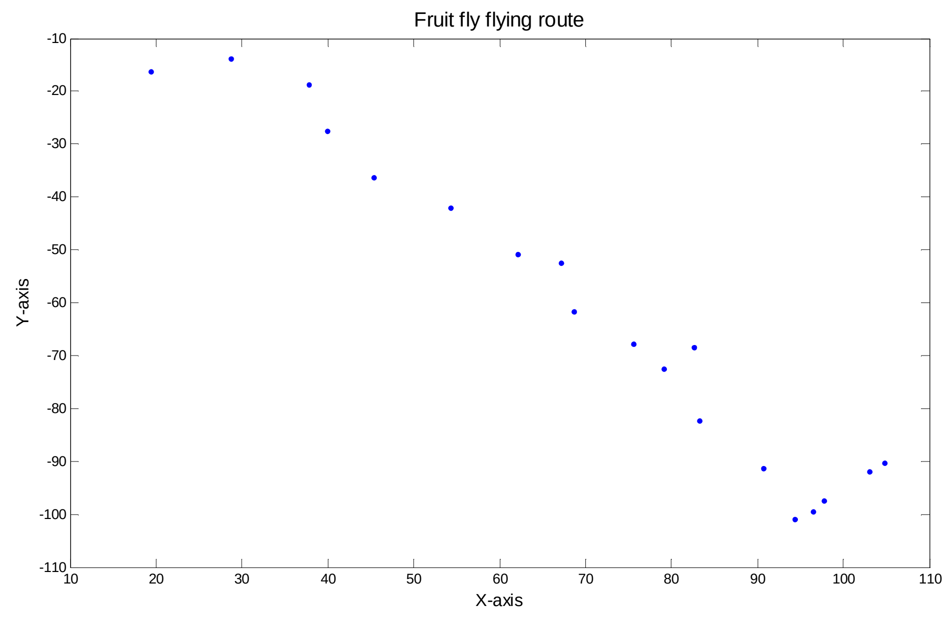


Figure 4.3.1 Flight routes of the fruit fly swarm

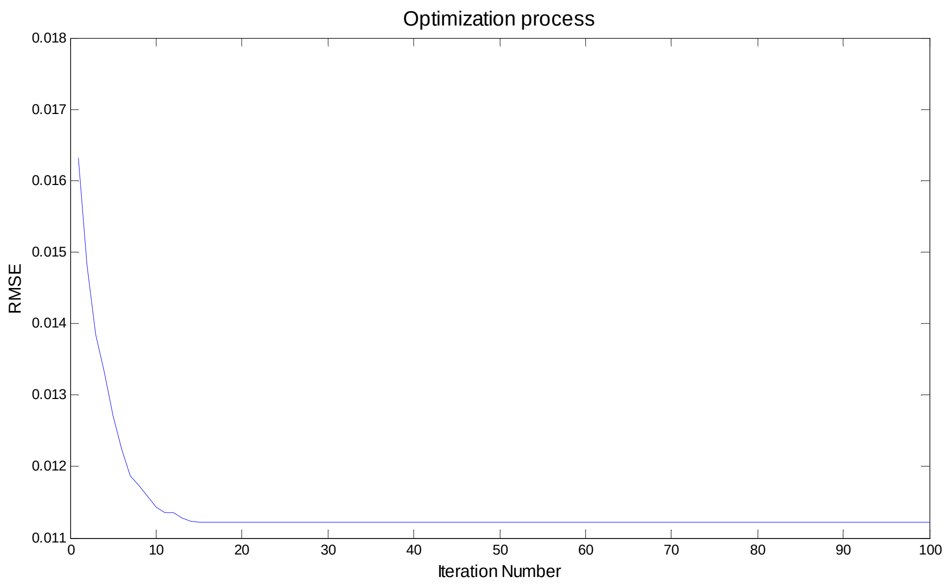


Figure 4.3.2 RMSE convergence trend

Figure 4.3.2 is the RMSE convergence trend of GRNN after 100 times of iterations dynamic adjustment by the FOA. The results found that during the evolution, the RMSE converges in the 14th generation, the RMSE value is 0.0112, the best Spread value is 0.0072, and the position of the fly swarm is (90.6389, -91.2353). Readers can enter Spread value (bestS) into GRNN training network architecture, and enter the test data into optimized GRNN model. The predicted results of the last 40 rows of data are shown in Table 4.3.2:

Table 4.3.2 Predicted results of the last 40 rows of data of FOAGRNN and GRNN

Target	0	0	0	1	1	1	1	1	1	1
FOAGRNN	0.0097	0.2729	0	0.9938	1	1	0	0.9999	1	1
GRNN	0.4341	0.437	0.408	0.4468	0.5468	0.4925	0.7115	0.4697	0.5868	0.5742
Target	1	1	1	1	0	1	1	0	0	0
FOAGRNN	1	1	1	0.9989	0.0005	0.2367	0.9708	0.2486	0.0581	0
GRNN	0.5649	0.6394	0.5888	0.4834	0.434	0.4294	0.4394	0.4614	0.4373	0.4264
Target	0	0	0	0	0	1	1	1	1	1
FOAGRNN	0.0001	0	0	0.0007	0	1	1	1	1	1
GRNN	0.4031	0.4151	0.4203	0.4191	0.4044	0.5603	0.6198	0.7404	0.5978	0.5946
Target	1	1	0	1	1	0	1	1	0	0
FOAGRNN	0.9961	0.8973	0.1478	0.9779	1	0	0.9851	1	0	0.0049
GRNN	0.4825	0.4823	0.4509	0.4696	0.5416	0.4458	0.4708	0.4618	0.4385	0.4239

The predicted results are defined as below: FOAGRNN and GRNN output greater than or equal to 0.5, then classified as 1; output less than 0.5 are classified as 0. According to the table, the FOAGRNN has 2 errors; GRNN has 11 errors. Thus, the GRNN which Spread parameter values are adjusted by FOAhas good predictive ability. Below is the description of FOAGRNN code:

Note: (GRNN network overtraining issue)

Our training program divides sample data into two groups to avoid the overtraining of the network. Training sample data are suggested to be evenly divided into several groups for interactive testing, or set an stop condition for iterative search (for example, when the RMSE is less than 0.01, searching is stopped) to avoid network overtraining.

The program code follows:

```
%*** Empty Memory

clc
clear

%*** Read data and divided into two groups for cross-training to avoid

overtraining.

load c:\TXY.txt;
[row,col]=size(TXY);
set=row/5;
row=row-set;
row1=row/2;
traindata1=TXY(1:row1,1:col-1);
traindata2=TXY(row1+1:row,1:col-1);
t1=TXY(1:row1,col);
t2=TXY(row1+1:row,col);
t1=t1';
t2=t2';
tr1=traindata1';
tr2=traindata2';
la=1;

%*** Random initial fruit fly swarm location

X_axis=rand();
Y_axis=rand();
maxgen=100;    % Iterations number
sizepop=10;    % Populations size

%*** Fly optimization start

fprintf('Fruit fly searching ....\n')
fprintf('_______________________________\n')
```

```
%*** Use the sense of smell to find food.
for i=1:sizepop

%*** Give the random direction and distance for the search of food using
osphresis by an individual fruit fly.
X(i)=X_axis+20*rand()-10;
Y(i)=Y_axis+20*rand()-10;

%*** Calculating the distance from the origin
D(i)=(X(i)^2+Y(i)^2)^0.5;

%*** The smell concentration judgment value (S) is calculated, and this value
is the reciprocal of distance.
S(i)=1/D(i);

%*** Substitute smell concentration judgment value (S) into smell
concentration judgment function (or called Fitness function) so as to find the
smell concentration (Smelli) of the individual location of the fruit fly.
g=0;
p=S(i);
if 0.001>p
p=1;
end
  if la == 1
net=newgrnn(tr1,t1,p);
yc=sim(net,tr2);
y=yc-t2;% Output minus target
for ii=1:row1
```

```
g=g+y(ii)^2;
end
Smell(i)= (g/row1) ^0.5;
la=2;
else
net=newgrnn(tr2,t2,p);
yc=sim(net,tr1);
y=yc-t1;%Output minus target
for ii=1:row1
g=g+y(ii)^2;
end
Smell(i)= (g/row1) ^0.5;
la=1;
end
end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
[bestSmell bestindex]=min(Smell);
X_axis=X(bestindex);
Y_axis=Y(bestindex);
bestS=S(bestindex);
Smellbest=bestSmell;
```

%*** Iterative optimization start

```
fprintf('Enter iterative fruit fly searching ....\n')
fprintf('_______________________________\n')
for gen=1:maxgen
gen
for i=1:sizepop
```

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
    g=0;
    X(i)=X_axis+20*rand()-10;
    Y(i)=Y_axis+20*rand()-10;
```

%*** Since the food location cannot be known, the distance to the origin is thus estimated first (Dist).

```
    D(i)=(X(i)^2+Y(i)^2)^0.5;
```

%*** Then the smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

```
    S(i)=1/D(i);
```

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
    p=S(i);
    if 0.001>p
    p=1;
    end
  if la == 1% Two sets of training data cross training and testing
    net=newgrnn(tr1,t1,p);
    yc=sim(net,tr2);
    y=yc-t2;% Network output minus target value
    for ii=1:row1
    g=g+y(ii)^2;
    end
    Smell(i)= (g/row1) ^0.5;
    la=2;
  else
    net=newgrnn(tr2,t2,p);
```

```
   yc=sim(net,tr1);
   y=yc-t1;% Output minus target
   for ii=1:row1
   g=g+y(ii)^2;
   end
   Smell(i)= (g/row1) ^0.5;
   la=1;
  end
   end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
   [bestSmell bestindex]=min(Smell);
     if bestSmell<Smellbest
              X_axis=X(bestindex);
              Y_axis=Y(bestindex);
              bestS=S(bestindex);
              Smellbest=bestSmell;
     end
```

%*** Each generation best value record to the variable yy

```
     yy(gen)=Smellbest;
     Xbest(gen)=X_axis;
     Ybest(gen)=Y_axis;
     if bestSmell<0.01 % Set iterative search stop condition
     break;
     end
end
```

%*** Draw the optimization process of trends

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('RMSE','fontsize',12);
```

```
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
bestS
```

Problems and Applications

1. In the code, how many groups of fruit fly swarms are initiated? In GRNN, how many coefficients are available for adjustment?
2. Try to write an FOAGRNN program, and substitute X1-X6 and Y from table 4.3.1 into the model, then test the predictive ability of FOAGRNN.

Chapter5

FOA Optimized Grey Model Neural Network —Fund trading decisions as an example

5.1 Introduction Grey Model Neural Network

Grey problem means the forecasting problem regarding the development change of the behavioral feature value of grey uncertain system. The original series of the feature value of the uncertain system, that is, $X_t^{(0)}$ (t=0,1,2,….,N-1), after one time AGO (accumulated generating operation), we can obtain new series $X_t^{(1)}$, which shows exponential growth pattern, hence, a continuous function or differential equation can be used to perform data simulation and forecast. For the convenience of expression, the symbol is re-defined, and the original series $X_t^{(0)}$ is represented as X(t), and after one time AGO (accumulated generating operation), the obtained series $X_t^{(1)}$is represented as Y(t), and the forecast result $X_t^{*(1)}$is represented as Z(t).

The differential equation of Grey Model Neural Network model of n parameters is expressed as:

$$\frac{\mathrm{dy1}}{\mathrm{dt}} + ay_1 = b_1y_2 + b_2y_3 + \cdots + b_{n-1}y_n \tag{1}$$

In the equation, $y_2, \ldots, y_n$ is system input parameter; $y_{1,}$ is system output parameter; a, b_1, b_2,…, b_{n-1} are differential equation coefficients.

The reaction time of equation (1) is:

$$\mathrm{z(t)} = (y_1(0) - \frac{b_1}{a}y_2(t) - \frac{b_2}{a}y_3(t) - \cdots - \frac{b_{n-1}}{a}y_n(t))\mathrm{e}^{-\mathrm{at}} + \frac{b_1}{a}y_2(t) + \frac{b_2}{a}y_3(t) + \cdots + \frac{b_{n-1}}{a}y_n(t) \tag{2}$$

Let $\mathrm{d} = \frac{b_1}{a}y_2(t) + \frac{b_2}{a}y_3(t) + \cdots + \frac{b_{n-1}}{a}y_n(t)$

Equation (2) can be transformed to equation (3)

$$z(t) = ((y_1(0) - d) \cdot \frac{e^{-at}}{1 + e^{-at}} + d \cdot \frac{1}{1 + e^{-at}}\Big) \cdot (1 + e^{-at}) =$$

$$\left((y_1(0) - d)\left(1 - \frac{1}{1 + e^{-at}}\right) + d \cdot \frac{1}{1 + e^{-at}}\right) \cdot (1 + e^{-at}) =$$

$$\left((y_1(0) - d) - y_1(0) \cdot \frac{1}{1+e^{-at}} + 2d \cdot \frac{1}{1+e^{-at}}\right) \cdot (1 + e^{-at})$$

(3)

When the transformed equation (3) is mapped to an expanded BP neural network, we can then obtain Grey Model Neural Network of n input parameters and 1 output parameter. It is as shown in figure 5.5.1:

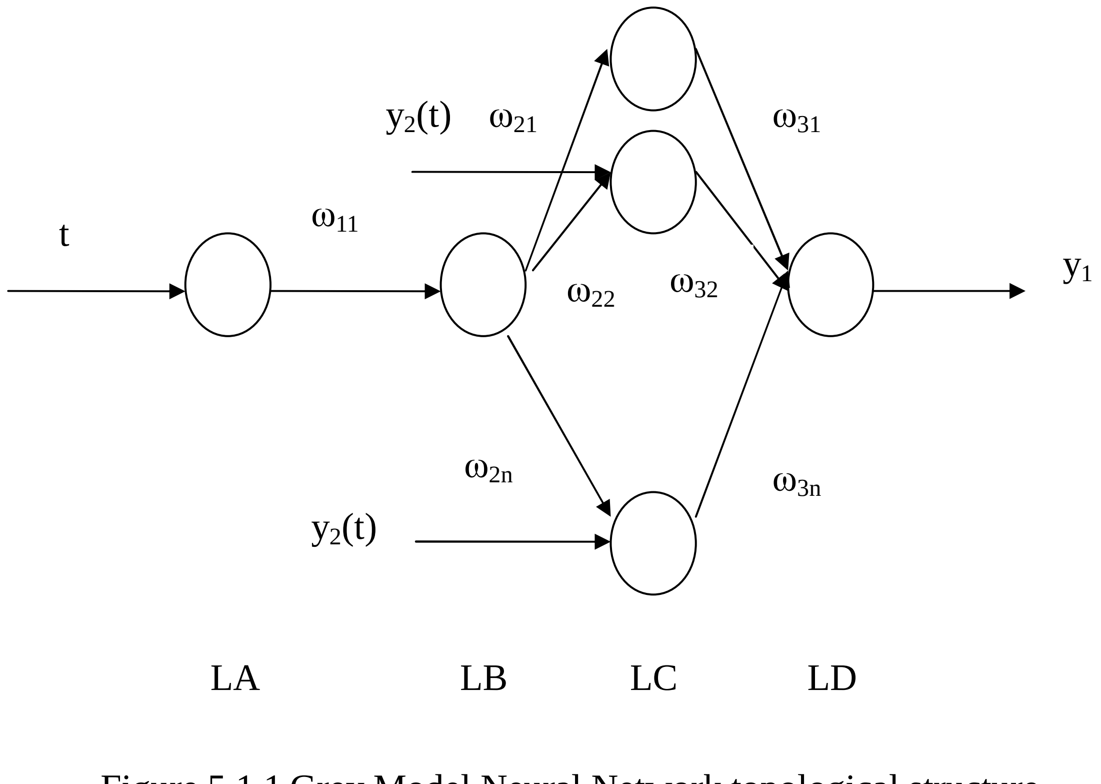


Figure 5.1.1 Grey Model Neural Network topological structure

Here, t is input parameter serial number; $y_2(t), \ldots y_n(t)$ are network input parameters; $\omega_{21}, \omega_{22}, \ldots, \omega_{2n}, \omega_{31}, \omega_{32,} \ldots \omega_{3n}$ are network weighting values; y_1 is network forecast value; LA,LB,LC,LD are used to represent respectively four layers of Grey Model Neural Network.

Let $\frac{2b_1}{a} = u_1$, $\frac{2b_2}{a} = u_2$, …, $\frac{2b_{n-1}}{a} = u_{n-1}$, then the network initial weighting value can be represented as:

$$\omega_{11} = a \text{ ，} \omega_{21} = -y_1(0) \text{ ，} \omega_{22} = u_1 \text{ ，} \omega_{23} = u_2 \text{ ，} \ldots \text{ ，} \omega_{2n} = u_{n-1}$$

$$\omega_{31} = \omega_{32} = \cdots = \omega_{3n} = 1 + \mathrm{e}^{-\mathrm{at}}$$

In LD layer, the threshold value of the output node is:

$$\theta = (1 - \mathrm{e}^{-\mathrm{at}})\big(d - y_1(0)\big)$$

The learning process of Grey Model Neural Network is as in the following:

Step 1: Follow the training data feature to initialize network structure and to initialize parameters a,b, meanwhile, a,b values are used to calculate u.

Step 2: Follow network weighting definition to calculate $\omega_{11}, \omega_{21}, \omega_{22}, \ldots, \omega_{2n}, \omega_{31}, \omega_{32,} \ldots \omega_{3n}$

Step 3: For each input series $\big(t, \mathrm{y}(t)\big), t = 1,2,3, \ldots, N,$ calculate the output of each layer

LA layer:$\mathrm{a} = \omega_{11} t$

LB layer:$\mathrm{b} = \mathrm{f}(\omega_{11} t) = \frac{1}{1+\mathrm{e}^{-\omega_{11}\mathrm{t}}}$

LC layer: $c_1 = \mathrm{b}\omega_{21}$, $c_2 = y_2(t) b \omega_{22}$ ，$c_3 = y_3(t) b \omega_{23}$ ， … ，$c_n = y_n(t) b \omega_{2n}$

LD layer:$\mathrm{d} = \omega_{31} c_1 + \omega_{32} c_2 + \cdots + \omega_{3n} c_n - \theta_{y1}$

Step 4: Calculate the error between network forecast output and expectation output, and follow the error to adjust weighting value and threshold value.

LD layer error:$\delta = \mathrm{d} - y_1(t)$

LC layer error: $\delta_1 = \delta(1+\mathrm{e}^{-\omega_{11}\mathrm{t}})$, $\delta_2 = \delta(1+\mathrm{e}^{-\omega_{11}\mathrm{t}})$, … , $\delta_n = \delta(1+\mathrm{e}^{-\omega_{11}\mathrm{t}})$

LB layer error: $\delta_{n+1} = \frac{1}{1+\mathrm{e}^{-\omega_{11}\mathrm{t}}}\left(1-\frac{1}{1+\mathrm{e}^{-\omega_{11}\mathrm{t}}}\right)(\omega_{21}\delta_1 + \omega_{22}\delta_2 + \cdots + \omega_{2n}\delta_n)$

Follow forecast error to adjust the weighting value.

Adjust the connection weighting value from LB to LC.

$$\omega_{21} = -y_1(0) \text{ , } \omega_{22} = \omega_{22} - \mu_1\delta_2 b \text{ , … , } \omega_{2n} = \omega_{2n} - \mu_{n-1}\delta_n b$$

Adjust the connection weighting value from LA to LB:$\omega_{11} = \omega_{11} + at\delta_{n+1}$

Adjust threshold value: $\theta = (1+\mathrm{e}^{-\omega_{11}\mathrm{t}})\left(\frac{\omega_{22}}{2}y_2(t) + \frac{\omega_{23}}{2}y_3(t) + \cdots + \frac{\omega_{2n}}{2}y_n(t) - y_1(0)\right)$

Step 5: Judge whether the training is ended or not, if not, go back to step 3.

5.2 Grey model neural network applications

The book reference Shi Feng et al. write Program and Introduced its Grey model neural network program code. In Matlab toolbox, no GMNN tool box can be used, must be written by the user themselves. Therefore, the application is not common in Taiwan, Readers are worth study. Its program code is as follows:

```
%*** Empty Memory
clc
clear

%***Read data
load c:\TXY.txt;
[row,col]=size(TXY);
set=row/5;
row1=row-set;
X=TXY;

%***Raw data accumulated generating operation
for i=1:row
        y(i,1)=sum(X(1:i,1));
        y(i,2)=sum(X(1:i,2));
        y(i,3)=sum(X(1:i,3));
        y(i,4)=sum(X(1:i,4));
        y(i,5)=sum(X(1:i,5));
        y(i,6)=sum(X(1:i,6));
end
```

```
%*** Network parameters random initialization (Note: 5×1)

a=0.3+rand(1)/4;
b1=0.3+rand(1)/4;
b2=0.3+rand(1)/4;
b3=0.3+rand(1)/4;
b4=0.3+rand(1)/4;
b5=0.3+rand(1)/4;

%*** Initialize the learning rate

u1=0.0015;
u2=0.0015;
u3=0.0015;
u4=0.0015;
u5=0.0015;

%*** Initialize the weights

t=1;
w11=a;
w21=-y(1,1);
w22=2*b1/a;
w23=2*b2/a;
w24=2*b3/a;
w25=2*b4/a;
w26=2*b5/a;
w31=1+exp(-a*t);
w32=1+exp(-a*t);
w33=1+exp(-a*t);
w34=1+exp(-a*t);
w35=1+exp(-a*t);
w36=1+exp(-a*t);
theta=(1+exp(-a*t))*(b1*y(1,2)/a+b2*y(1,3)/a+b3*y(1,4)/a+b4*y(1,5)/a+b5*y(1,
6)/a-y(1,1));
```

%*** Record error value

```
for j=1:100
E(j)=0;
for i=1:row1 %data rows number
t=i;
```

%*** Network output calculation

```
    LB_b=1/(1+exp(-w11*t));       %LB layer output
    LC_c1=LB_b*w21;
    LC_c2=y(i,2)*LB_b*w22;     %LC layer output
    LC_c3=y(i,3)*LB_b*w23;     %LC layer output
    LC_c4=y(i,4)*LB_b*w24;     %LC layer output
    LC_c5=y(i,5)*LB_b*w25;     %LC layer output
    LC_c6=y(i,6)*LB_b*w26;     %LC layer output
LD_d=w31*LC_c1+w32*LC_c2+w33*LC_c3+w34*LC_c4+w35*LC_c5+w36*
LC_c6;     %LD layer output
theta=(1+exp(-w11*t))*(w22*y(i,2)/2+w23*y(i,3)/2+w24*y(i,4)/2+w25*y(i,5)/2
+w26*y(i,6)/2-y(1,1));               %Threshold output
ym=LD_d-theta;                       % Network predictive value
yc(i)=ym;
```

%*** Correction weights

```
error=ym-y(i,1)                      % calculation error value
E(j)=E(j)+abs(error);                % Plus the total error
error1=error*(1+exp(-w11*t));        %LC layer error
error2=error*(1+exp(-w11*t));
error3=error*(1+exp(-w11*t));
error4=error*(1+exp(-w11*t));
error5=error*(1+exp(-w11*t));
error6=error*(1+exp(-w11*t));
error7=(1/(1+exp(-w11*t)))*(1-1/(1+exp(-w11*t)))*(w21*error1+w22*error2+w
23*error3+w24*error4+w25*error5+w26*error6);     %LB layer error
```

%***update weights

```
w22=w22-u1*error2*LB_b;
w23=w23-u2*error3*LB_b;
w24=w24-u3*error4*LB_b;
w25=w25-u4*error5*LB_b;
w26=w26-u5*error6*LB_b;
w11=w11+a*t*error7;
end
end
```

%*** Begin to predict

```
for i=row1+1:row
    t=i;
    LB_b=1/(1+exp(-w11*t));     %LB
    LC_c1=LB_b*w21;
    LC_c2=y(i,2)*LB_b*w22;     %LC
    LC_c3=y(i,3)*LB_b*w23;     %LC
    LC_c4=y(i,4)*LB_b*w24;     %LC
    LC_c5=y(i,5)*LB_b*w25;     %LC
    LC_c6=y(i,6)*LB_b*w26;     %LC
LD_d=w31*LC_c1+w32*LC_c2+w33*LC_c3+w34*LC_c4+w35*LC_c5+w36*
LC_c6;     %LD layer output
theta=(1+exp(-w11*t))*(w22*y(i,2)/2+w23*y(i,3)/2+w24*y(i,4)/2+w25*y(i,5)/2
+w26*y(i,6)/2-y(1,1));    % Threshold output
ym=LD_d-theta;                         % Network predictive value
yc(i)=ym;
end
```

%*** Each obtained a predicted value is decremented by 1

```
for j=row:-1:row1+1
    ys(j)=(yc(j)-yc(j-1));
end
ys
```

This chapter try using XOR to train GMNN and test the predictive ability, the predicted output of the XOR is the same as 0, 1 are not the same. In order to meet GMNN characteristic, the output is changed to the same as 1; not the same as -1, as follows (Note: GMNN output Y, placed on the left!!), Keep data save to “C: \ “ root directory.

TXY.txt

```
Y    X1   X2
1    0    0
-1   0    1
-1   1    0
1    1    1
1    0.99 1
-1   1.1  -0.08
```

Test results are as follows:

```
0    -0.0047    0.0847    0.1406    0.1413    0.0989
```

Although there is an error by the test results, such as the fifth column of the actual value is 1, and the test is 0.1413；The sixth column the actual value of -1, and the test is 0.00989, But the observed test value 0.1413 is indeed in the direction of 1 approach; test value 0.00989, is indeed the direction of approach to -1, but poor accuracy. If we can increase the data rows and adjustment GMNN parameters can improve this result.

Because the test data is only two input variables, so we will change the GMNN program code as follows:

```
%***Read data
load c:\TXY.txt;
[row,col]=size(TXY);
X=TXY;
for i=1:row
      y(i,1)=sum(X(1:i,1));
      y(i,2)=sum(X(1:i,2));
      y(i,3)=sum(X(1:i,3));
end

%*** Initialize the parameters and learning rate
a=0.3+rand(1)/4;
b1=0.3+rand(1)/4;
b2=0.3+rand(1)/4;
u1=0.0015;
u2=0.0015;

%*** Initialize the weights
t=1;
w11=a;
w21=-y(1,1);
w22=2*b1/a;
w23=2*b2/a;
w31=1+exp(-a*t);
w32=1+exp(-a*t);
w33=1+exp(-a*t);
theta=(1+exp(-a*t))*(b1*y(1,2)/a+b2*y(1,3)/a-y(1,1));
for j=1:100
j
E(j)=0;
for i=1:4
```

```
        t=i;
```

%*** Network output calculation

```
        LB_b=1/(1+exp(-w11*t));
        LC_c1=LB_b*w21;
        LC_c2=y(i,2)*LB_b*w22;
        LC_c3=y(i,3)*LB_b*w23;
        LD_d=w31*LC_c1+w32*LC_c2+w33*LC_c3;
theta=(1+exp(-w11*t))*(w22*y(i,2)/2+w23*y(i,3)/2-y(1,1));
ym=LD_d-theta;
yc(i)=ym
```

%*** Correction weights

```
error=ym-y(i,1);
E(j)=E(j)+abs(error);
error1=error*(1+exp(-w11*t));
error2=error*(1+exp(-w11*t));
error3=error*(1+exp(-w11*t));
error4=(1/(1+exp(-w11*t)))*(1-1/(1+exp(-w11*t)))*(w21*error1+w22*error2+w
23*error3);
w22=w22-u1*error2*LB_b;
w23=w23-u2*error3*LB_b;
w11=w11+a*t*error4;
end
end
```

```
%*** Begin to predict
for i=5:6
        t=i;
        LB_b=1/(1+exp(-w11*t));
        LC_c1=LB_b*w21;
        LC_c2=y(i,2)*LB_b*w22;
        LC_c3=y(i,3)*LB_b*w23;
        LD_d=w31*LC_c1+w32*LC_c2+w33*LC_c3;
theta=(1+exp(-w11*t))*(w22*y(i,2)/2+w23*y(i,3)/2-y(1,1));
ym=LD_d-theta;
yc(i)=ym;
end
for j=6:-1:2
        ys(j)=(yc(j)-yc(j-1))/10;
end
```

5.3 Optimization of Gray Model Neural Network Parameters by FOA—Taking fund trading decision-making as an example

In this section, fund trading decision-making sample data are used to construct the model of trading decision-making. To respect the data provider's ownership of the data, only three variables are listed in this book: monthly salary income (X1), risk preferences (X2) and fund trading decisions (Y). The indications of the three variables are listed below:
Monthly salary income: 1) Below NT$ 10,000 2) NT$10,000 ~ NT$20,000 3) NT$20,000~ NT$30,000 4) .... 9) Over NT$80,000.
Risk appetite: 1) very low 2) low 3) minor low 4) ordinary low 5) Normal 6) Ordinary high 7) minor high 8) high 9) very high
Fund trading decisions (Y): 0) Buy 1)Not to buy.

25 rows of sample data in total are shown in Table 5.3.1. Divide X1 and X2 by 10 to make the value between 0.1 to 0.9, and Y is binary data. Please enter the data (not including serial numbers and title) into Notepad and save to "C:\" root directory as "TXY.txt".

In this section, FOA is used to optimize the three parameters, a, b1, and b2, of gray model neural network to improve its detection capabilities. To set the parameters of FOA, the interval of the random initial fruit fly swarm location should be [0,1]; the interval of the flight direction and distant of iterative fruit fly food searching should be [-10,10]; the population size should be 20, and the iterative number should be 100.

Table 5.3.1 Fund trading wishes sample data

No	X1	X2	Y	No	X1	X2	Y
1.	0.9	0.1	1	14.	0.5	0.7	0
2.	0.8	0.2	1	15.	0.9	0.6	0
3.	0.7	0.3	1	16.	0.1	0.2	1
4.	0.3	0.7	1	17.	0.4	0.3	1
5.	0.2	0.8	1	18.	0.3	0.3	1
6.	0.1	0.9	1	19.	0.5	0.7	0
7.	0.3	0.3	1	20.	0.9	0.6	0
8.	0.2	0.2	1	21.	0.3	0.2	1
9.	0.1	0.1	1	22.	0.4	0.2	1
10.	0.7	0.7	0	23.	0.1	0.3	1
11.	0.8	0.8	0	24.	0.8	0.7	0
12.	0.9	0.9	0	25.	0.8	0.9	0
13.	0.9	0.8	0				

Figure 5.3.1 is through the FOA 100 iterations dynamic adjustment GMNN, RMSE convergence trend. Flight routes of the fruit fly swarm show in Figure 5.3.2.

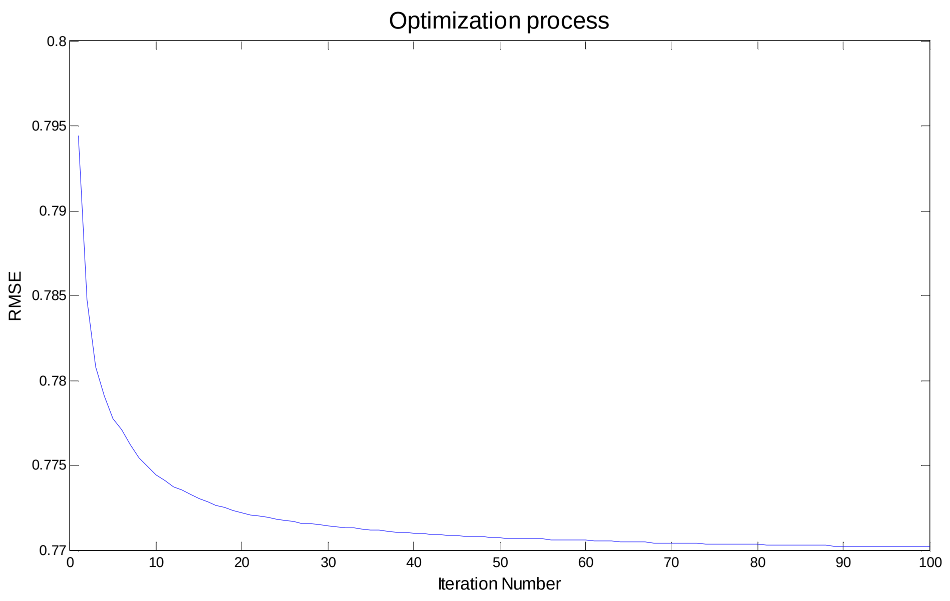


Figure 5.3.1 RMSE convergence trend

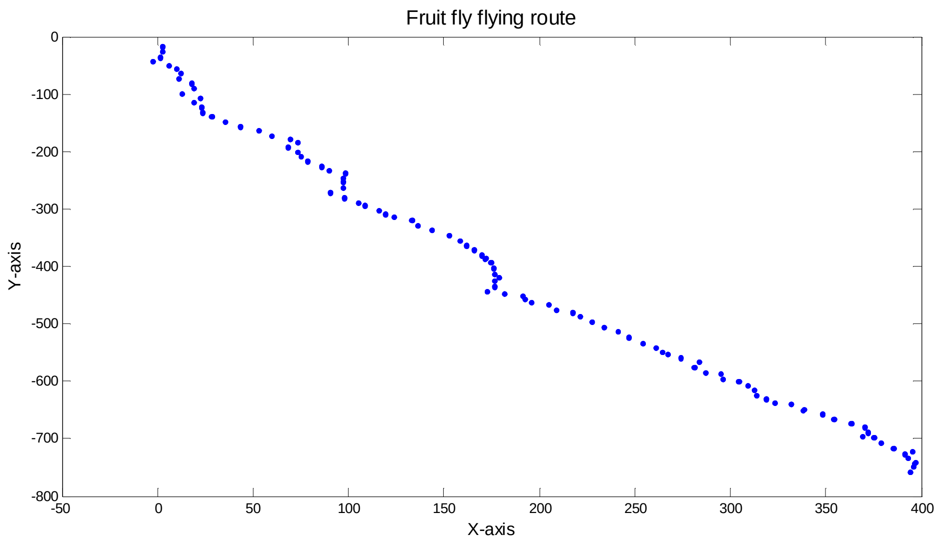


Figure 5.3.2 flight routes of the fruit fly swarm

We input test data model FOA GRNN, if greater than 0.5 are classified as 1, last 5 rows predicted result are 0.6001, 0.5868, 0.6357, 0.3423, 0.4401, predicted values all correctly. Prediction error is smaller than using only GMNN and the results can be seen by the forecast GMNN suitable for small sample forecasting. FOA optimized grey model neural network (FOAGMNN) code below:

```
%***Set parameters of three fields Y X1-X2

% Empty Memory

clc
clear
load c:\TXY.txt;
[row,col]=size(TXY);
set=row/5;
row1=row-set;
```

% Random initial fruit fly swarm location

```
X_axis=rands(1,3);
Y_axis=rands(1,3);
maxgen=100;    % Iterations number
sizepop=20;    % Populations size
NN=10;           %Network learning times
```

%*** Fly optimization start

```
% Use the sense of smell to find food
for p=1:sizepop
```

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
X(p,:)=X_axis+20*rand()-10;
Y(p,:)=Y_axis+20*rand()-10;
```

%*** Calculating the distance from the origin

```
D(p,1)=(X(p,1)^2+Y(p,1)^2)^0.5;
D(p,2)=(X(p,2)^2+Y(p,2)^2)^0.5;
D(p,3)=(X(p,3)^2+Y(p,3)^2)^0.5;
```

%*** The smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

```
S(p,1)=1/D(p,1);
S(p,2)=1/D(p,2);
S(p,3)=1/D(p,3);
```

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
a=0.3+3*S(p,1);
b1=0.3+3*S(p,2);
b2=0.3+3*S(p,3);
%GREYANN
for i=1:row
        y(i,1)=sum(TXY(1:i,1));
        y(i,2)=sum(TXY(1:i,2));
        y(i,3)=sum(TXY(1:i,3));
end

u1=0.0015;
u2=0.0015;
u3=0.0015;

t=1;
w11=a;
w21=-y(1,1);
w22=2*b1/a;
w23=2*b2/a;
w31=1+exp(-a*t);
w32=1+exp(-a*t);
w33=1+exp(-a*t);
theta=(1+exp(-a*t))*(b1*y(1,2)/a+b2*y(1,3)/a-y(1,1));
for j=1:NN
E(j)=0;
for i=1:row1
        t=i;
        LB_b=1/(1+exp(-w11*t));
        LC_c1=LB_b*w21;
        LC_c2=y(i,2)*LB_b*w22;
        LC_c3=y(i,3)*LB_b*w23;
        LD_d=w31*LC_c1+w32*LC_c2+w33*LC_c3;
theta=(1+exp(-w11*t))*(w22*y(i,2)/2+w23*y(i,3)/2-y(1,1));
ym=LD_d-theta;
yc(i)=ym;
error=ym-y(i,1);
E(j)=E(j)+abs(error);
error1=error*(1+exp(-w11*t));
error2=error*(1+exp(-w11*t));
error3=error*(1+exp(-w11*t));
error4=(1/(1+exp(-w11*t)))*(1-1/(1+exp(-w11*t)))*(w21*error1+w22*error2+w
23*error3);
w22=w22-u1*error2*LB_b;
w23=w23-u2*error3*LB_b;
w11=w11+a*t*error4;
end
E(j)=E(j)/row1;
end
Smell(p)=min(E);
End
```

%*** Keep the best smell concentration value and x, y coordinate , and at this

moment, the fruit fly swarm will use vision to fly towards that location.

```
[bestSmell bestindex]=min(Smell);
X_axis=X(bestindex,:);
Y_axis=Y(bestindex,:);
bestS=S(bestindex,:);
Smellbest=bestSmell;
```

%*** Iterative optimization start

```
for gen=1:maxgen
gen
for p=1:sizepop
```

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly.

```
X(p,:)=X_axis+20*rand()-10;
Y(p,:)=Y_axis+20*rand()-10;
```

%*** Since the food location cannot be known, the distance to the origin is thus estimated first (Dist).

```
D(p,1)=(X(p,1)^2+Y(p,1)^2)^0.5;
D(p,2)=(X(p,2)^2+Y(p,2)^2)^0.5;
D(p,3)=(X(p,3)^2+Y(p,3)^2)^0.5;
```

%*** Then the smell concentration judgment value (S) is calculated, and this value is the reciprocal of distance.

```
S(p,1)=1/D(p,1);
S(p,2)=1/D(p,2);
S(p,3)=1/D(p,3);
```

%*** Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly.

```
a=0.3+3*S(p,1);
b1=0.3+3*S(p,2);
b2=0.3+3*S(p,3);
%GREYANN
t=1;
w11=a;
w21=-y(1,1);
w22=2*b1/a;
```

```
w23=2*b2/a;
w31=1+exp(-a*t);
w32=1+exp(-a*t);
w33=1+exp(-a*t);
theta=(1+exp(-a*t))*(b1*y(1,2)/a+b2*y(1,3)/a-y(1,1));
for j=1:NN
E(j)=0;
for i=1:row1
    t=i;
    LB_b=1/(1+exp(-w11*t));
    LC_c1=LB_b*w21;
    LC_c2=y(i,2)*LB_b*w22;
    LC_c3=y(i,3)*LB_b*w23;
    LD_d=w31*LC_c1+w32*LC_c2+w33*LC_c3;
theta=(1+exp(-w11*t))*(w22*y(i,2)/2+w23*y(i,3)/2-y(1,1));
ym=LD_d-theta;
yc(i)=ym;
error=ym-y(i,1);
E(j)=E(j)+abs(error);
error1=error*(1+exp(-w11*t));
error2=error*(1+exp(-w11*t));
error3=error*(1+exp(-w11*t));
error4=(1/(1+exp(-w11*t)))*(1-1/(1+exp(-w11*t)))*(w21*error1+w22*error2+w
23*error3);
w22=w22-u1*error2*LB_b;
w23=w23-u2*error3*LB_b;
w11=w11+a*t*error4;
end
E(j)=E(j)/row1;
end
Smell(p)=min(E);
end
[bestSmell bestindex]=min(Smell);
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
if bestSmell<Smellbest
X_axis=X(bestindex,:);
Y_axis=Y(bestindex,:);
bestS=S(bestindex,:);
besta=a;
bestb1=b1;
bestb2=b2;
Smellbest=bestSmell;
End
```

%*** Each generation best value record to the variable yy

```
    yy(gen)=Smellbest;
    Xbest(gen,:)=X_axis;
    Ybest(gen,:)=Y_axis;
end
```

%*** Draw the optimization process of trends

```
%%%% Predicted start
for i=row1+1:row
        t=i;
        LB_b=1/(1+exp(-w11*t));
        LC_c1=LB_b*w21;
        LC_c2=y(i,2)*LB_b*w22;
        LC_c3=y(i,3)*LB_b*w23;
        LD_d=w31*LC_c1+w32*LC_c2+w33*LC_c3;
theta=(1+exp(-w11*t))*(w22*y(i,2)/2+w23*y(i,3)/2-y(1,1));
ym=LD_d-theta;
yc(i)=ym;
end
ys=0;
for j=row:-1:row1+1
        ys(j)=(yc(j)-yc(j-1));
end
ys
%%%%%%%%
besta;
bestb1;
bestb2;
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('RMSE','fontsize',12);
bestS;
Xbest;
Ybest;
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
```

Problems and Applications

1. In FOAGMNN code, How many groups of fly populations initially? and GMNN have number of parameters can be adjusted?
2. Try to write a FOAGMNN program code, moreover, Table 5.3.1 addition X3 and given their data values into the model, and test the FOAGMNN predictive ability.

Chapter6

FOA Optimized Support vector regression —Research mainland students studying in Taiwan for example

6.1 Introduction Support Vector Regression

Support Vector Machines (SVM) is a machine learning system developed by Dr. Vapnik(1995) in Bell Lab based on statistical theory as basis. Its basic concept is to construct the best super plane among samples or in the feature space so that super plane will have maximal distance to sample set of different type. However, support vector machine is mainly used in the sample data classification, can not be predicted for continuous data. Therefore, in order to meet predicted needs, in accordance with the SVM develop a Support Vector Regression (SVR) model. Due to the current application support vector regression has been popular, detailed theory of SVR, the reader can referred to the relevant papers and books, the focus of this book is how to apply support vector regression and optimization of its parameters. However, the chapter1 to the chapter5 to use the Matlab neural network toolbox, most schools have purchased and built-in Matlab toolbox. This chapter describes support vector regression, the reader must download and install their own use, as it relates to copyright issues, the reader seek the consent of the author before downloading (Note: The author is professor of English Steve Gunn, Email: srg@ecs.soton.ac.uk). Please download and unzip the folder was renamed SVR, and then copy this folder to the Matlab Toolbox root directory, which is described in detail in the next section by installation.

6.2 Installation and configuration support vector regression

Copy SVR folder to the Matlab To directory, as shown in Figure

6.2.1.

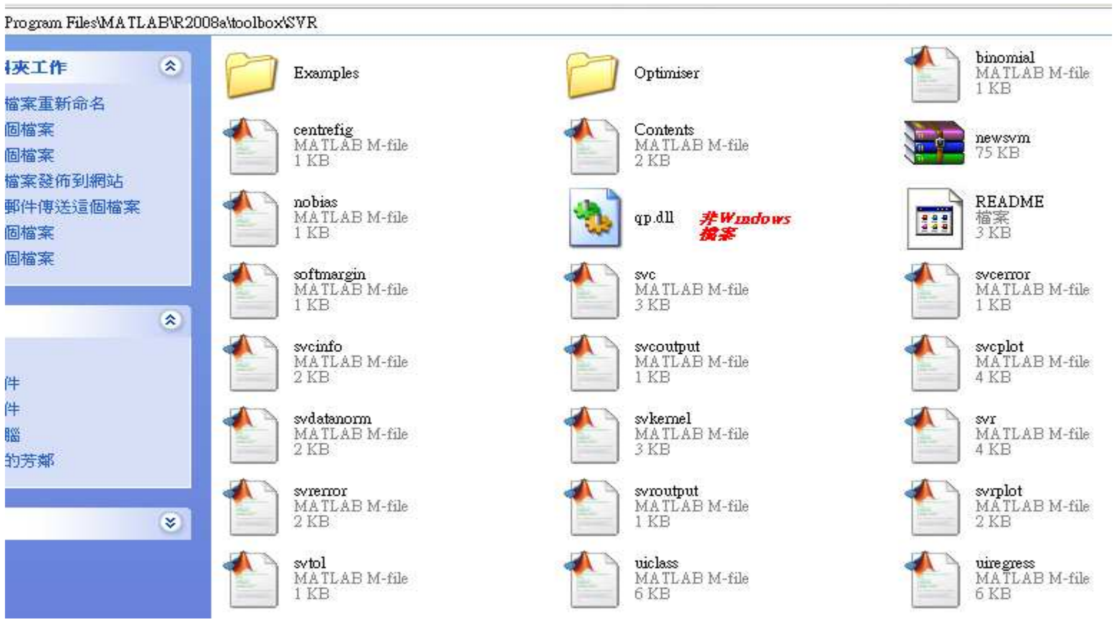
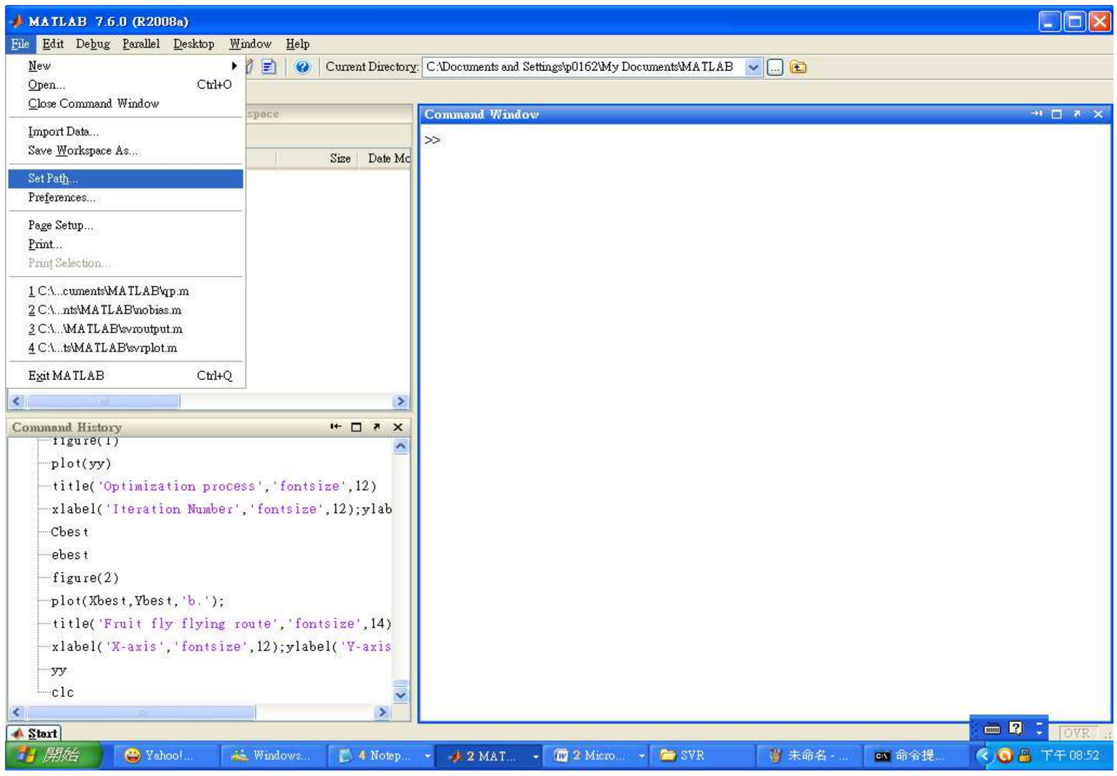

Program Files\MATLAB\R2008a\toolbox\SVR
Examples
Optimiser
binomial MATLAB M-file 1 KB
centrefig MATLAB M-file 1 KB
Contents MATLAB M-file 2 KB
newsvm 75 KB
nobias MATLAB M-file 1 KB
qp.dll
README 檔案 3 KB
softmargin MATLAB M-file 1 KB
svc MATLAB M-file 3 KB
svcerror MATLAB M-file 1 KB
svcinfo MATLAB M-file 2 KB
svcoutput MATLAB M-file 1 KB
svcplot MATLAB M-file 4 KB
svdatanorm MATLAB M-file 2 KB
svkernel MATLAB M-file 3 KB
svr MATLAB M-file 4 KB
svrerror MATLAB M-file 2 KB
svroutput MATLAB M-file 1 KB
svrplot MATLAB M-file 2 KB
svtol MATLAB M-file 1 KB
uiclass MATLAB M-file 6 KB
uiregress MATLAB M-file 6 KB

Figure 6.2.1 SVR folder

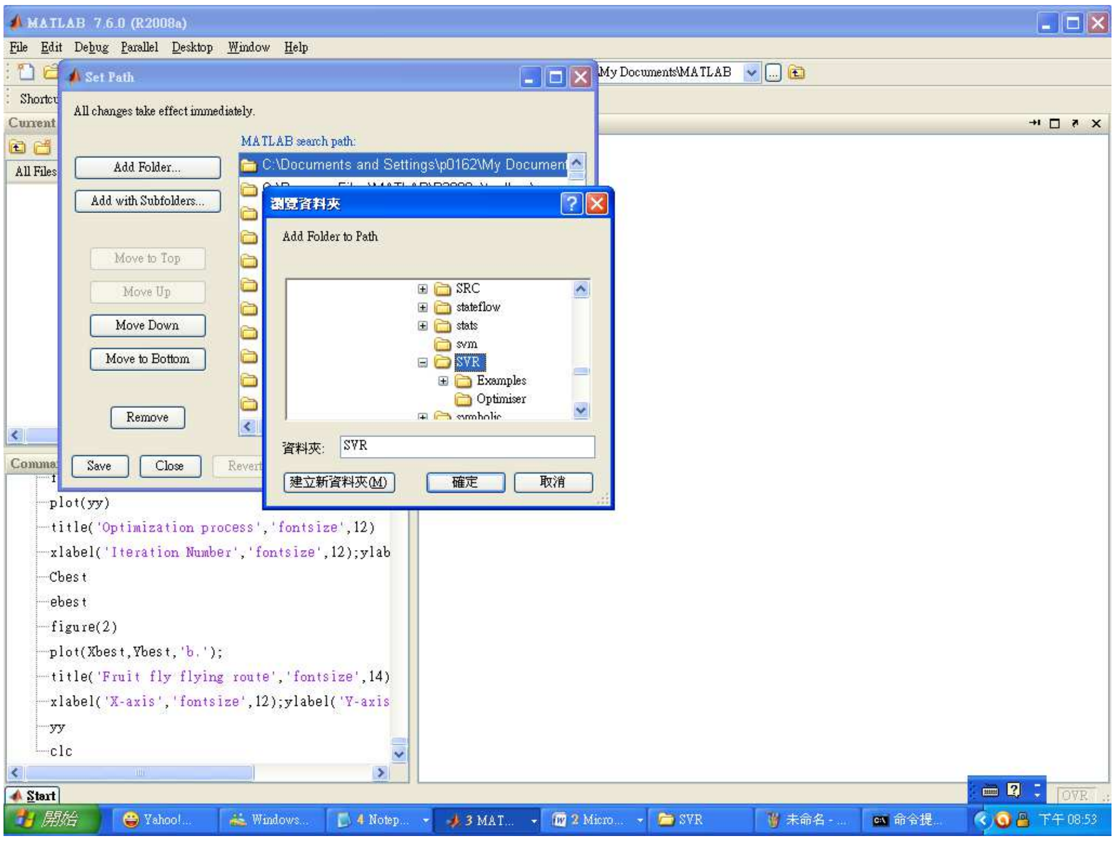

MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Current Directory: C:\Documents and Settings\p0162\My Documents\MATLAB
New
Open... Ctrl+O
Close Command Window
Import Data...
Save Workspace As...
Set Path...
Preferences...
Page Setup...
Print...
Print Selection...
1 C:\...cuments\MATLAB\qp.m
2 C:\...nts\MATLAB\nobias.m
3 C:\...\MATLAB\svroutput.m
4 C:\...ts\MATLAB\svrplot.m
Exit MATLAB Ctrl+Q
Command Window
Command History
Start
開始
下午 08:52

```
>>
```

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylab
Cbest
ebest
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis
yy
clc
```

Figure 6.2.2 Matlab Set Path option

Then start Matlab, from the toolbar, File -> Set Path show in Figure 6.2.2. open the Set Path window, Add and browse to Matlab-Toolbox-SVR folder, show in Figure 6.2.3, Press” OK” -> Save -> Close.



All changes take effect immediately.

MATLAB search path:

C:\Documents and Settings\p0162\My Document

Add Folder to Path

資料夾: SVR

```
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylab
Cbest
ebest
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis
yy
clc
```

Figure 6.2.3 Matlab Set Path Menu

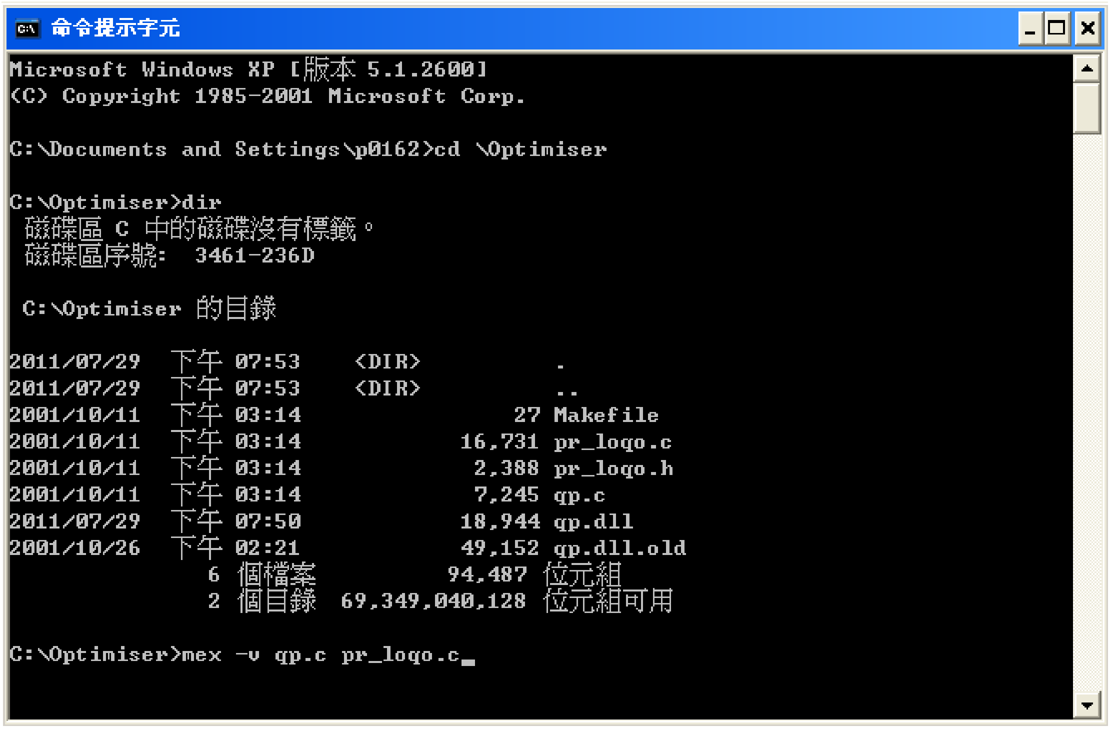
In the folder have two folders (Examples and Optimiser、19 program file and a qp.dll file). After the test authors found, in the SVR folders "qp.dll" not belong to the Windows file, judged by the authors, the file may be in Unix-Like OS generation. Figure 6.2.1 have special labeling，the following error message appears when you run the program:

```
Support Vector Regressing ....
______________________________

Constructing ...
Optimising ...
??? Invalid MEX-file 'C:\Program Files\MATLAB\R2008a\toolbox\svr\qp.dll': C:\Program
Files\MATLAB\R2008a\toolbox\svr\qp.dl     valid Win32 application.
```

```
.
Error in ==> svr at 85
        [alpha lambda how] = qp(Hb, c, A, b, vlb, vub, x0, neqcstr);
```

Therefore, the reader must generate qp.dll files in their own windows system. The method is simple to produce, first SVR folder Optimiser folder copy to the C: \, and then opened a DOS command prompt, type the command as shown in 6.2.4 to generate qp.dll file.



```
Microsoft Windows XP [版本 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\p0162>cd \Optimiser

C:\Optimiser>dir
 磁碟區 C 中的磁碟沒有標籤。
 磁碟區序號:  3461-236D

 C:\Optimiser 的目錄

2011/07/29  下午 07:53    <DIR>          .
2011/07/29  下午 07:53    <DIR>          ..
2001/10/11  下午 03:14                27 Makefile
2001/10/11  下午 03:14            16,731 pr_loqo.c
2001/10/11  下午 03:14             2,388 pr_loqo.h
2001/10/11  下午 03:14             7,245 qp.c
2011/07/29  下午 07:50            18,944 qp.dll
2001/10/26  下午 02:21            49,152 qp.dll.old
               6 個檔案          94,487 位元組
               2 個目錄  69,349,040,128 位元組可用

C:\Optimiser>mex -v qp.c pr_loqo.c_
```

Figure 6.2.4 generate qp.dll file command and produce

Readers should note that before executing the command, the file should be renamed qp.dll, After generating new qp.dll file, copy it to a Matlab SVR data folder, Re-execute the program again, can run correctly. Run message is below:

```
Support Vector Regressing ....
_______________________________
Constructing ...
```

```
Optimising ...
Execution time :    0.3 seconds
Status : OPTIMAL_SOLUTION
|w0|^2       : 26.101602
Sum beta : -0.023430
Support Vectors : 19 (95.0%)
Support Vector Regressing ....
```

Next section, the author will tests SVR command, and briefly describes SVR commonly used functions and commands。

6.3 Support vector regression applications

SVR commonly used functions and ıand includes:

1. [nsv beta bias] = svr(X,Y,ker,C,loss,e), used to construct support vector regression, Its parameters defined:

X	- Training inputs
Y	- Training targets
ker	- kernel function, default is 'linear'
C	- upper bound (non-separable case), default is Inf(Infinity)
loss	- loss function , default is 'eInsensitive'
e	- insensitivity, default is 0.0
nsv	- number of support vectors
beta	- Difference of Lagrange Multipliers
bias	- bias term

2. tstY = svroutput(trnX,tstX,ker,beta,bias), used to calculate the SVR test output, Its parameters defined:

trnX	- Training inputs
tstX	- Test inputs
ker	- kernel function
beta	- Difference of Lagrange Multipliers
bias	- bias
comp	- sparse components

3. svrplot(X,Y,ker,alpha,bias,e,mag,xaxis,yaxis,input), used to draw graphics output, Its parameters defined:

X	- Training inputs
Y	- Training targets
ker	- kernel function
beta	- Difference of Lagrange Multipliers
bias	- Bias term
e	- e insensitive value
mag	- display magnification
xaxis	- xaxis input (default: 1)
input	- vector of input values (default: zeros(no_of_inputs))

This chapter attempts to use "XOR" training "SVR" and test its ability to predict, XOR feature is the "same is 0, not the same is 1", in order to meet the requirements of SVR, change to "same is 1, not the same is -1", Sample data is as follows:

TXY.txt

```
X1  X2  Y
0   0   1
0   1   -1
1   0   -1
1   1   1
```

First, we input XOR 4 rows

```
>> X=[0 0; 0 1; 1 0; 1 1]

X =

     0     0
     0     1
     1     0
     1     1
```

```
>> Y=[1; -1; -1; 1]

Y =

     1
    -1
    -1
```

```
     1
```

The data input to the support vector regression calculation and construction

```
>> [nsv,beta,bias]=svr(X,Y,'rbf')
Support Vector Regressing ....
______________________________
Constructing ...
Optimising ...
Execution time :    0.0 seconds
Status : DUAL_INFEASIBLE
|w0|^2       : 0.046185
Sum beta : 0.000000
Support Vectors : 4 (100.0%)

nsv =

     4

beta =

    1.9884
   -1.9884
   -1.9884
    1.9884

bias =

     0
```

Then enter test data

```
>> X2=[0.9 0.1;0.1 0.1]

X2 =

    0.9000    0.1000
    0.1000    0.1000
```

The data input to the model and test, Predicted results of the first row should be close to -1, the second row should be close to 1.

```
>> yc=svroutput(X,X2,'rbf',beta,bias)

yc =

    -0.0038
    0.0038

>>
```

Test results are generally correct, but also need to adjust the C and e two kinds of parameters, the accuracy can be improved. The next section, we show how to use the "FOA" dynamically adjust SVR's two kinds of parameters.

6.4 FOA Optimized Support vector regression

In this section, the topic of the study is the survey of mainland Chinese students' willingness to study in Taiwan. From the survey, we got the survey data as the sample data, and then we use FOA to optimize support vector regression to construct the detection model for mainland Chinese students' willingness to study in Taiwan. From this model we can understand the study conditions in Taiwan that affect mainland Chinese students' willingness to study in Taiwan. The result of the study could be the reference for Taiwanese universities to improve. The question items in the survey include basic information (e.g. genders, education level, household income……) and ten key question items. To respect the data provider's ownership of the data, only two independent variables, the importance of the ranking of universities (X1) and tuition cost (X2), are listed in this section. Dependent variables are mainland Chinese students' willingness to study in Taiwan (Y). 1 represents willingness, and -1 represents unwillingness. For the same reason, only 50 rows of sample data are selected and shown below in this section for readers' testing. X1 and X2 are between 0.1~0.9. The higher the value is, the higher the correlation is. Y is binary data. Please enter the data (not including serial numbers and title) into Notepad and save to "C:\" root directory as "TXY.txt".

Table 6.4.1 Mainland China students to study in Taiwan willingness survey sample data

No	X1	X2	Y	No	X1	X2	Y

1	0.9	0.3	1	26	0.7	0.3	1
2	0.2	0.8	-1	27	0.2	0.8	-1
3	0.1	0.8	-1	28	0.1	0.6	-1
4	0.7	0.3	1	29	0.2	0.7	-1
5	0.9	0.4	1	30	0.8	0.2	1
6	0.1	0.7	1	31	0.8	0.4	1
7	0.2	0.8	-1	32	0.4	0.8	-1
8	0.4	0.8	-1	33	0.2	0.7	-1
9	0.2	0.7	-1	34	0.1	0.6	-1
10	0.8	0.2	1	35	0.2	0.7	-1
11	0.8	0.3	1	36	0.9	0.3	1
12	0.2	0.9	-1	37	0.6	0.1	1
13	0.1	0.7	-1	38	0.9	0.1	1
14	0.4	0.9	-1	39	0.1	0.6	-1
15	0.2	0.7	-1	40	0.9	0.4	1
16	0.9	0.3	1	41	0.9	0.2	1
17	0.6	0.1	1	42	0.1	0.6	-1
18	0.9	0.1	1	43	0.1	0.8	-1
19	0.2	0.7	-1	44	0.9	0.3	1
20	0.9	0.4	1	45	0.6	0.1	1
21	0.9	0.3	1	46	0.7	0.3	1
22	0.9	0.1	1	47	0.1	0.8	-1
23	0.1	0.8	-1	48	0.2	0.8	-1
24	0.8	0.1	1	49	0.2	0.9	-1
25	0.9	0.2	1	50	0.8	0.1	1

This section uses FOA iterative dvnamic adjustment of support vector regression two parameters C and ε, make the model to improve detection capabilities. Parameter settings on FOA, the random initialization fruit fly swarm location zone is [0, 1], the random fly direction and distance zone of iterative fruit fly food searching is [-10,10], populations size is 10, iterative number is 100. Figure 6.4.1 is RMSE convergence trend.

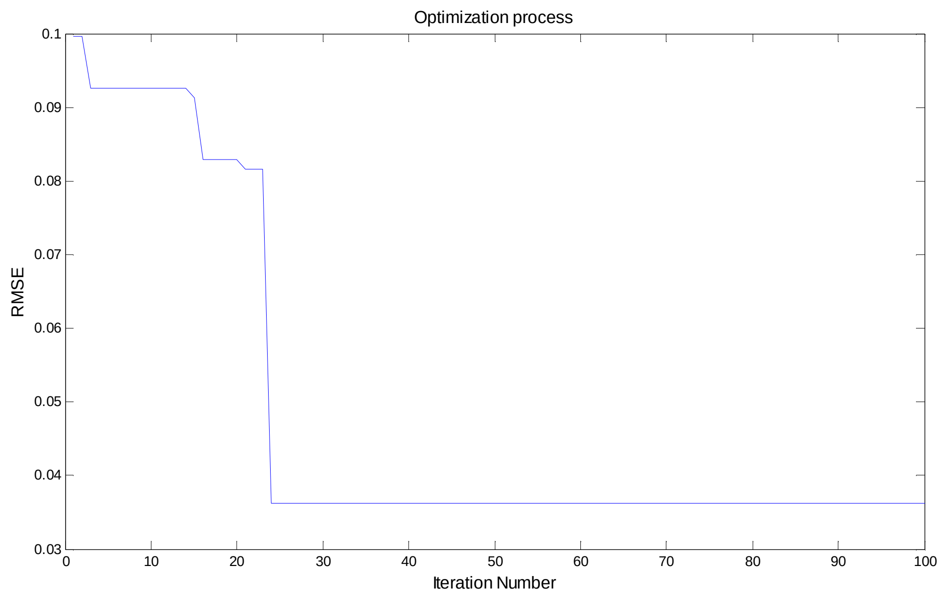


Figure 6.4.1 RMSE convergence trend

Figure 6.4.2 show the Flight routes of the fruit fly swarm, Observe the fly the route, the route we found unstable, it seems relevant rows number of data, may also be on the "FOA" parameter settings related, readers deserve further discussion.

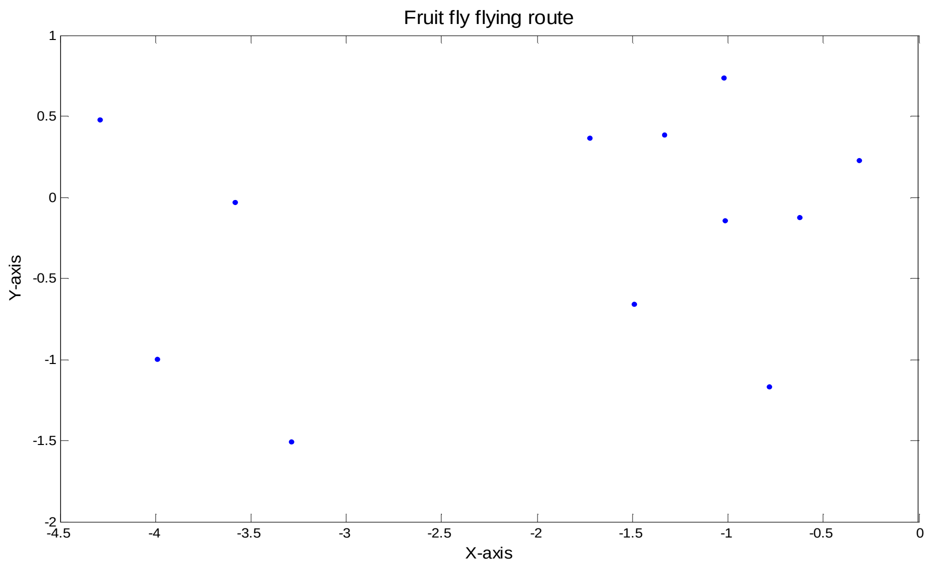


Figure 6.4.2 Flight routes of the fruit fly swarm

FOA results found in the first 24 iterations converge, RMAE is 0.0363, the optimal value of C is 25.6914, the best e value is 0.7924, we put two parameter values into the model to tests, followed by 10 rows prediction results are as follows:

0.2702, -0.2201, -0.2201, 0.2702, 0.2702, 0.2702, -0.2201, -0.2201, -0.2201, 0.2702

and the RMSEis 0.239 。

If we use the default C = Inf (infinity), e = 0.0 into the model to predict the results are as follows:

0.1394, -0.1638, -0.1638, 0.1394, 0.1394, 0.1394, -0.1638, -0.1638, -0.1638, 0.1394

And the RMSE is 0.268 。

Because during the "FOA" search parameters, sample data prior to minus 10 test cases, then the remaining 40 rows data were divided into two groups cross-testing, avoid of neural network over learning. Thus, 20 rows sample data too few resulted in gap between the two is only about 0.03.

Readers can try this sample data, repeated 10 times to copy that data to increase to 500 rows data, if increased data items, I believe there will be a better test results. Under the FOA to optimize support vector regression (FOASVR) program code:

FOASVR Program code

%* Empty Memory**

```
clc
clear
```

%*Sample data minus the 1/5 as the test data**

```
load c:\TXY.txt;
[row,col]=size(TXY);
set=row/5;
row=row-set;
```

%* Data were divided into two groups cross-validation avoid over learning**

```
row1=row/2;
traindata1=TXY(1:row1,1:col-1);
traindata2=TXY(row1+1:row,1:col-1);
t1=TXY(1:row1,col);
t2=TXY(row1+1:row,col);
t1=t1;
t2=t2;
tr1=traindata1;
tr2=traindata2;
```

%*** Random initial fruit fly swarm location

```
X_axis=rands(1,2);
Y_axis=rands(1,2);
maxgen=100;
sizepop=20;
la=1;
```

%* Fly optimization start**

```
for i=1:sizepop
X(i,:)=X_axis+20*rand()-10;
Y(i,:)=Y_axis+20*rand()-10;
D(i,1)=(X(i,1)^2+Y(i,1)^2)^0.5;
D(i,2)=(X(i,2)^2+Y(i,2)^2)^0.5;
S(i,1)=1/D(i,1);
S(i,2)=1/D(i,2);
```

%*Set SVR parameters**

```
g=0;
ker='rbf';
C=20*S(i,1);% Use FOA adjust the parameters C
loss='einsensitive';
e=S(i,2);% Use FOA adjust the parameters e
```

%* Two sets of data to cross-test iterations**

```
 if la == 1
   [nsv,beta,bias]=svr(tr1,t1,ker,C,loss,e);
   yc=svroutput(tr1,tr2,ker,beta,bias);
   y=yc-t2;% Network output minus target
   for ii=1:row1
   g=g+y(ii)^2;
   end
```

%***Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly

```
   Smell(i)=g^0.5/row1;
   la=2;
 else
   [nsv,beta,bias]=svr(tr2,t2,ker,C,loss,e);
   yc=svroutput(tr2,tr1,ker,beta,bias);
   y=yc-t1; Network output minus target
   for ii=1:row1
   g=g+y(ii)^2;
   end
   Smell(i)= (g/row1) ^0.5;
   la=1;
 end
end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
[bestSmell bestindex]=min(Smell);
X_axis=X(bestindex,:);
Y_axis=Y(bestindex,:);
bestS=S(bestindex,:);
Smellbest=bestSmell;
```

%*** Iterative optimization start

```
for gen=1:maxgen
gen
   for i=1:sizepop
   g=0;
   X(i,:)=X_axis+20*rand()-10;
   Y(i,:)=Y_axis+20*rand()-10;
   D(i,1)=(X(i,1)^2+Y(i,1)^2)^0.5;
   D(i,2)=(X(i,2)^2+Y(i,2)^2)^0.5;
   S(i,1)=1/D(i,1);
   S(i,2)=1/D(i,2);
```

%* Set SVR parameters**

```
  ker='rbf';
  C=20*S(i,1)
  loss='einsensitive';
  e=S(i,2)
  if la == 1
    [nsv,beta,bias]=svr(tr1,t1,ker,C,loss,e);
    yc=svroutput(tr1,tr2,ker,beta,bias)
    y=yc-t2;
   for ii=1:row1
    g=g+y(ii)^2;
    end
```

%***Substitute smell concentration judgment value (S) into smell concentration judgment function (or called Fitness function) so as to find the smell concentration (Smelli) of the individual location of the fruit fly

```
    Smell(i)=g^0.5/row1;
    la=2;
  else
    [nsv,beta,bias]=svr(tr2,t2,ker,C,loss,e);
    yc=svroutput(tr2,tr1,ker,beta,bias)
    y=yc-t1;
   for ii=1:row1
    g=g+y(ii)^2;
    end
    Smell(i)= (g/row1) ^0.5;
    la=1;
  end
end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location.

```
    [bestSmell bestindex]=min(Smell);
      if bestSmell<Smellbest
              X_axis=X(bestindex,:);
              Y_axis=Y(bestindex,:);
              bestS=S(bestindex,:);
              Smellbest=bestSmell;
              Cbest=10*S(bestindex,1);
              ebest=S(bestindex,2);
     end
     % Each generation of optimal value to record array yy
     yy(gen)=Smellbest;
     Xbest(gen,:)=X_axis;
     Ybest(gen,:)=Y_axis;
     if bestSmell<0.01 % Set termination conditions, avoid network overtraining.
     break;
     end
 end
```

<u>**%***** Draw the optimization process of trends</u>

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('RMSE','fontsize',12);
Cbest
ebest
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
```

Problems and Applications

1. In FOASVR code, How many groups of fly populations initially? and SVR have number of parameters can be adjusted?
2. Try to write a FOASVR program code, moreover, Table 6.4.1 addition X3 and given their data values into the model, and test the FOASVR predictive ability.

Chapter7

Modified Fruit Fly Optimization Algorithm

7.1 Introduction of Modified Fruit Fly Optimization Algorithm

Some scholars pointed out that FOA is easily trapped in local extremum and unable to find the global extremum in a Chinese Matlab Internet forum (www.ilovematlab.cn). Take function “f (x) = sin (x) / x” for example, this is caused by the appropriate Fitness function of FOA. To leave the confine of local extremum, Fitness function must be corrected. In other words, Si must be corrected. In addition, Disti (the distance) is positive, and Si (the reciprocal of the distance) is also positive. Therefore, the Fitness function of FOA cannot be negative. A large number of scholars have pointed out this deficiency. We refer to the papers of Li (2012) and proposed the correction of FOA—Modified Fruit Fly Optimization Algorithm (MFOA). By adding Δ, a trip parameter, to Si, the local extremum can be avoided, the global extremum can be found, and the Fitness function can be negative as well. Modified "Si" is as follows:

$Si=1/Disti+\Delta$; $\Delta=Disti \times (0.5-\delta)$; $0\leqq \delta \leqq 1$

Moreover, fruit flies fly in three-dimensional space, but the original FOA searches the global extremum in two-dimensional space. As a result, the best value in three-dimensional space might not be found. We use the concept of three-dimensional space searching to modify FOA programming examples for the first three steps:

- Random initial fruit fly swarm location

  Init X_axis; Init Y_axis; Init Z_axis

- Random direction and distance of a fruit fly individual searching for food by the sense of smell.

X_i= X_axis + Random Value

Y_i= Y_axis + Random Value

Z_i= Z_axis + Random Value

- Since the location of the food is unknown, the distance (Dist) to the origin should be estimated first. Its reciprocal is the decision value of the smell concentration (S).

$$\mathrm{Dist}_i=\sqrt{X_i^2+Y_i^2+Z_i^2}\ ;\ S_i=\frac{1}{\mathrm{Disti}}+\Delta$$

In the next section of this book, MFOA will be used to compute the extreme value of "f (x) = sin (x) / x".

7.2 Solving Sin(x)/x by Modified Fruit Fly Optimization Algorithm

In this section, Modified Fruit Fly Optimization Algorithm (MFOA) is used to solve the function "f (x) = sin (x) / x", which maximum value is 1. FOA can solve the function "f (x) = sin (x) / x"; however, if the function is shifted 7 units on the X-axis to either the left or the right, FOA is not workable anymore. Therefore, in this section, the function is changed into "Sin (x-7) / (x-7)". Both FOA and MFOA are used to compute the extreme value of the function and compare the differences between these two algorithms. Firstly, we use FOA to compute the extreme value of "Sin (x-7) / (x-7)". The result of the computing is as Figure 7.2.1 shows:

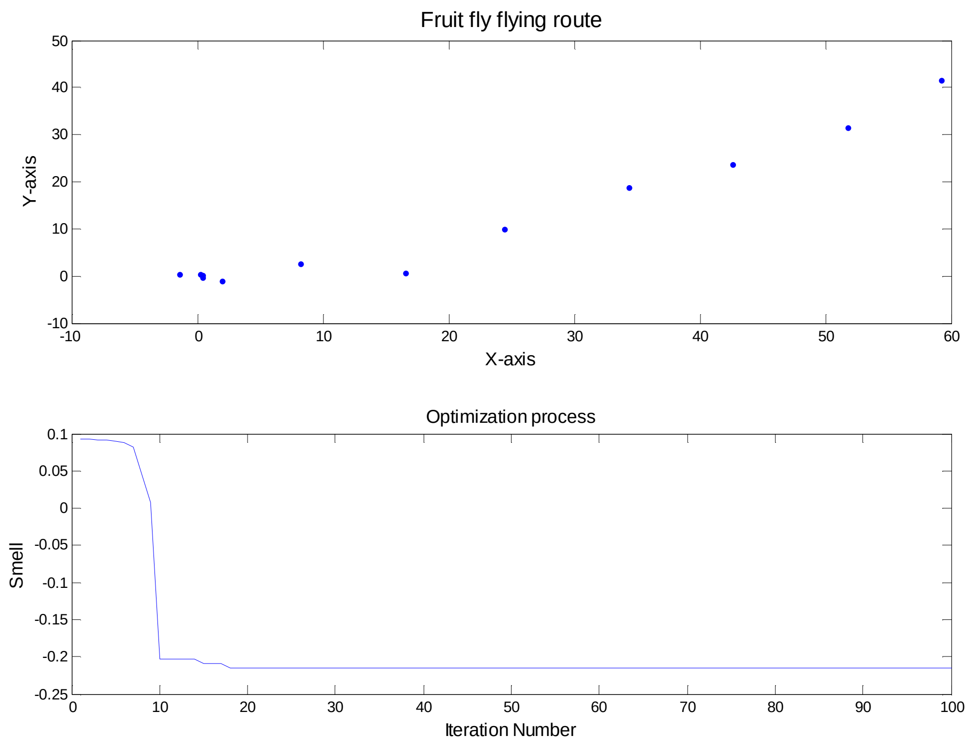


Figure 7.2.1 FOA solving "Sin(x-7)/(x-7)" results graph

From the figure we found the fruit fly swarm follows a fixed direction of the flying route to search for the extreme value. The figure of the optimization process shows the extreme value curve of the fruit fly's random function searching—after 100 generations, the extreme value of the function (1) still remains unknown.

Now, we use MFOA to compute the extreme value of "Sin (x-7) / (x-7)". The result of the computing is as Figure 7.2.2 shows:

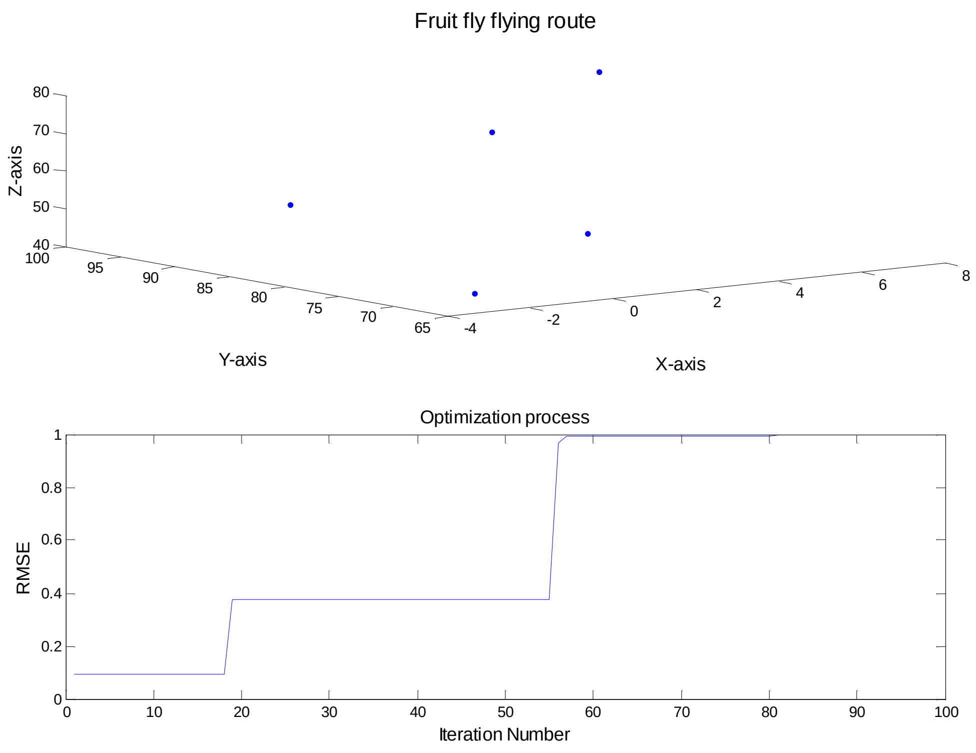


Figure 7.2.2 MFOA computing function "Sin (x-7) / (x-7)" extremes graphic result

From the figure we found the fruit fly swarm does not follow a fixed direction of the flying route to search the extreme value; moreover, the path points are fewer and random. The figure of the optimization process shows the extreme value curve of the fruit fly's random function searching—after 100 generations, the correct extreme value of the function (1) is found..

The program code follows:

Program code search sin(x-7)/(x-7) extreme value

%*** Empty Memory

```
clc
clear
```

%*** Random initial fruit fly swarm location

```
X_axis=100*rand(1,2);
Y_axis=100*rand(1,2);
Z_axis=100*rand(1,2);
maxgen=100;    %
sizepop=20;
B=0;
```

%*** Give the random direction and distance for the search of food using

osphresis by an individual fruit fly

```
for i=1:sizepop
X(i,:)=X_axis+20*rand()-10;
Y(i,:)=Y_axis+20*rand()-10;
Z(i,:)=Z_axis+20*rand()-10;
D(i,1)=(X(i,1)^2+Y(i,1)^2+Z(i,1)^2)^0.5;
D(i,2)=(X(i,2)^2+Y(i,2)^2+Z(i,2)^2)^0.5;
S(i,1)=1/D(i,1)+B;
S(i,2)=1/D(i,2)+B;
```

%***Like fitness function

```
x=S(i,1);
y=S(i,2);
Smell(i)=sin(x-7)/(x-7);
end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location

```
[bestSmell bestindex]=max(Smell);
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Z_axis=Z(bestindex);
Smellbest=bestSmell;
Dist=max(D(:,1))
```

%*** Iterative optimization start

```
for g=1:maxgen
B=Dist*(0.5-rand());
        for i=1:sizepop
   X(i,:)=X_axis+20*rand()-10;
   Y(i,:)=Y_axis+20*rand()-10;
   Z(i,:)=Z_axis+20*rand()-10;
   D(i,1)=(X(i,1)^2+Y(i,1)^2+Z(i,1)^2)^0.5;
   D(i,2)=(X(i,2)^2+Y(i,2)^2+Z(i,2)^2)^0.5;
   S(i,1)=1/D(i,1)+B;
   S(i,2)=1/D(i,2)+B;
   x=S(i,1);
   y=S(i,2);
```

% Like fitness function

```
   Smell(i)= sin(x-7)/(x-7);
   end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location

```
   [bestSmell bestindex]=max(Smell);
```

```
    if bestSmell>Smellbest
            X_axis=X(bestindex);
            Y_axis=Y(bestindex);
            Z_axis=Z(bestindex);
            Smellbest=bestSmell;
    end
     % Each generation of optimal value to record array yy
    yy(g)=Smellbest;
    Xbest(g)=X_axis;
    Ybest(g)=Y_axis;
    Zbest(g)=Z_axis;
end
```

%*** Draw the optimization process of trends

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('RMSE','fontsize',12);
figure(2)
plot3(Xbest,Ybest,Zbest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);zlabel('Z-axis','fontsize',12);
```

7.3 Solving Schaffer's by Modified Fruit Fly Optimization Algorithm

This section then use the well-known Schaffer's function as an example, the use of "MFOA" computing extremes of this function. Schaffer's. Extreme value of this function is 1, MFOA Each group has 20 fruit flies, the random initialization fruit fly swarm location zone is [0,100], the random fly direction and distance zone of iterative fruit fly food searching is [-10,10]. After 1000 number of generations evolved, find the maximum value is 1. In Figure 8.3.1, The above graphic is fly populations search extremes path, through figure found, rarely travel path and faster. Then observe the extreme value curve random search function in fly, after 100 generations, you can find the correct function extreme 1.

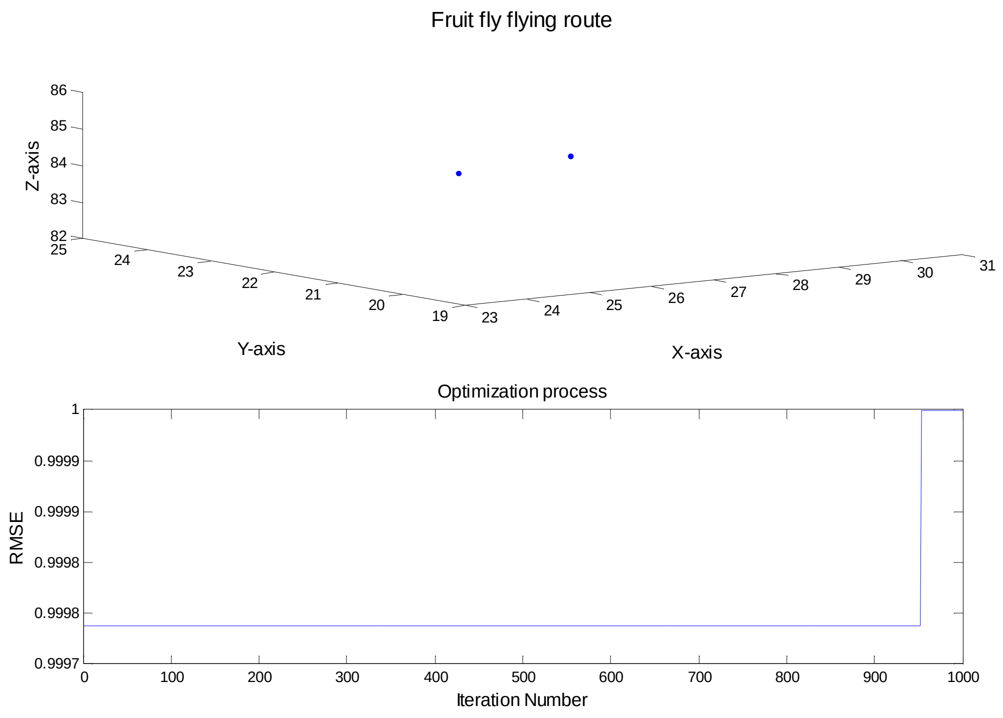


Figure 7.3.1 MFOA computing function" Schaffer's" extremes graphic result

The program code follows:

Program code search Schaffer function extremes value

%*** Empty Memory

```
clc
clear
```

%*** Random initial fruit fly swarm location

```
X_axis=100*rand(1,2);
Y_axis=100*rand(1,2);
Z_axis=100*rand(1,2);
maxgen=1000;
sizepop=20;
B=0;
```

%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly

```
for i=1:sizepop
X(i,:)=X_axis+20*rand()-10;
Y(i,:)=Y_axis+20*rand()-10;
Z(i,:)=Z_axis+20*rand()-10;
D(i,1)=(X(i,1)^2+Y(i,1)^2+Z(i,1)^2)^0.5;
D(i,2)=(X(i,2)^2+Y(i,2)^2+Z(i,2)^2)^0.5;
S(i,1)=1/D(i,1)+B;
S(i,2)=1/D(i,2)+B;
```

%*** Like fitness function

```
x=S(i,1);
y=S(i,2);
Smell(i)= 0.5-(((sin(x^2+y^2)^2)^0.5-0.5)/(1+0.001*(x^2+y^2))^2);
end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location

```
[bestSmell bestindex]=max(Smell);
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Z_axis=Z(bestindex);
Smellbest=bestSmell;
Dist=max(D(:,1))
```

%*** Iterative optimization start

```
for g=1:maxgen
B=Dist*(0.5-rand());
        for i=1:sizepop
   X(i,:)=X_axis+20*rand()-10;
   Y(i,:)=Y_axis+20*rand()-10;
   Z(i,:)=Z_axis+20*rand()-10;
   D(i,1)=(X(i,1)^2+Y(i,1)^2+Z(i,1)^2)^0.5;
   D(i,2)=(X(i,2)^2+Y(i,2)^2+Z(i,2)^2)^0.5;
   S(i,1)=1/D(i,1)+B;
   S(i,2)=1/D(i,2)+B;
   x=S(i,1);
   y=S(i,2);
```

% Like fitness function

```
   Smell(i)= 0.5-(((sin(x^2+y^2)^2)^0.5-0.5)/(1+0.001*(x^2+y^2))^2);
   end
```

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location

```
   [bestSmell bestindex]=max(Smell);
```

```
        if bestSmell>Smellbest
              X_axis=X(bestindex);
              Y_axis=Y(bestindex);
              Z_axis=Z(bestindex);
              Smellbest=bestSmell;
     end
      % Each generation of optimal value to record array yy
     yy(g)=Smellbest;
     Xbest(g)=X_axis;
     Ybest(g)=Y_axis;
     Zbest(g)=Z_axis;
end
```

<u>**%***** Draw the optimization process of trends</u>

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('RMSE','fontsize',12);
figure(2)
plot3(Xbest,Ybest,Zbest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);zlabel('Z-axis','fontsize',12);
```

Problems and Applications

1. Try writing a program to test MFOA computing Rastrigin's function extreme, and observe the degree of convergence fruit fly search path and whether there are differences with the FOA?
2. Use Chapter 4 “Web auction logistics service” 20 row sample data, to verify the effectiveness of MFOA.

Notes column

Chapter 8

Adaptive Chaos Fruit Fly Optimization Algorithm

8.1 Introduction Chaos Theory

Chaos Theory for unpredictable phenomenon and process analysis. A chaotic process is a deterministic process, but it looks chaotic and random. Studies chaotic behavior arises from the mathematics and pure sciences, after being cited economics and finance. In these areas, because people want to know in the back of the existence of certain natural phenomena has not yet been recognized laws, which inspired the people for chaos research. Scientists have noticed that certain phenomena, such as planetary motion, there is a stable law, but others, like the weather, it is capricious. Therefore, the key question is whether the weather phenomenon is random. Once thought to be random, and later proved to be chaotic, this issue has stimulated people to explore the truth of enthusiasm. If a variable or a process of evolution, or the time path of seemingly random, but in fact is determined, then the variable or time path to exhibit chaotic behavior. This time the path is generated by a deterministic nonlinear equation.

Chaos theory has the following features:

Randomness: the system is in a chaotic state is generated by the internal system dynamics randomness irregular behavior, often referred to as the randomness.

Sensitivity: chaotic motion system, whether discrete or continuous, low-dimensional or higher dimensional, conservative or dissipation, have an essential feature, i.e., moving the track system to the initial value sensitivity.

Fractal dimension: the chaos of having a fractal dimension, orbital motion of the system in phase space geometry can be used to describe the fractal dimension.

Scaling law: Chaos phenomenon is a non-cyclical ordered state, with infinite levels of self-similar structure, the existence of scale-free zone.

8.2 Chaos theory combined with FOA

Currently, the Chinese mainland scholars (Han Junying, 2013) application of chaos theory to improve the FOA missing, references in this section provided by mainland China Xiao-Fang Yuan (2013) Professor of papers and procedures, chaos theory outlined mixed fruit fly optimization algorithm (CFOA) of the computing process, there are the following steps:

step1: Initialized. Sets the maximum number of iterations Kmax, chaos search technology end K1, bureau search starting point K2.

Let K= 1, initialization chaotic sequence of fly group position M(·)。

$X\_axisi=Xmin,i+M(\cdot)(Xmax,i-Xmin,i);$

$Y\_axisi=Ymin,i+M(\cdot)(Ymax,i-Ymin,i);$

Step 2: if $K \leqq K1$, ，perform a global chaos search technology:

$X(i)=X\_axis+ M(\cdot) \cdot max\ \{(Xmax-X\_axis),(X\_axis-Xmin)\}$;

$Y(i)=Y\_axis+ M(\cdot) \cdot max\ \{(Ymax-Y\_axis),(Y\_axis-Ymin)\}$;

Otherwise, given a random direction and distance of individual fly, use smell in search of food.

$X(i)=X\_axis+R(K)* M(\cdot);$

$Y(i)=Y\_axis+R(K)* M(\cdot);$

Wherein, $R(K)=(Xmax-Xmin)/2*((Kmax-k)/ Kmax)^2$

Small R (K) of the number of iterations in the final, may increase local solution to fine-tune vector. Large R (K) in front of the number of iterations, and may increase the solution of the global solution diversity.

Step 3: Estimate of the distance from the origin (Dist), and then calculate the smell concentration judgment value (S)

D(i)=X(i)^2-Y(i)^2;

S(i)=D(i);

Step 4: Smell concentration judgment value (S) is substituted into the smell concentration judgment function (or called Fitness function), then find the smell concentration (Smelli) fly individual locations.

Smell(i)=Function(S(i));

Step 5: Find out the fruit fly with maximal smell concentration (finding the maximal value) among the fruit fly swarm.

[bestSmell bestindex]=max(Smell);

Step 6: if bestSmell>Smellbest or K<K2, skip to step 7, otherwise, perform local search methods are as follows:

X_L=0.618*X_axis+0.382*X(bestIndex);

Y_L=0.618*Y_axis+0.382*Y(bestIndex);

Computing $Dist_L$, S_L and $Smell_L$。

If $Smell_L$>Smellbest, updated fitness function value (Fitness Value) and optimized variable values (Optimization Variables Value), skip to step 7; otherwise:

X_C = X(bestIndex)+1.618*(X_axis−X(bestIndex));

Y_C = Y(bestIndex)+1.618*(Y_axis−Y(bestIndex));

Computing $Dist_C$, S_C and $Smell_C$. Then, if you find a better fitness function value is updated search results.

Step 7: Judgment fitness are better than previous generations, if so, then retaining the best smell concentration and x, y coordinates. Meanwhile, the fly populations using visual flew toward the position.

X_axis=X(bestindex);

Y_axis=Y(bestindex);

Smellbest=bestSmell;

Step 8: if K≥Kmax, stop CFOA search; otherwise, skip to step 2.

After testing, CFOA solution can solve global optimal solution as well as local optimal solution of the problem, worth to try, the flow chart shown in Figure

8.2.1:

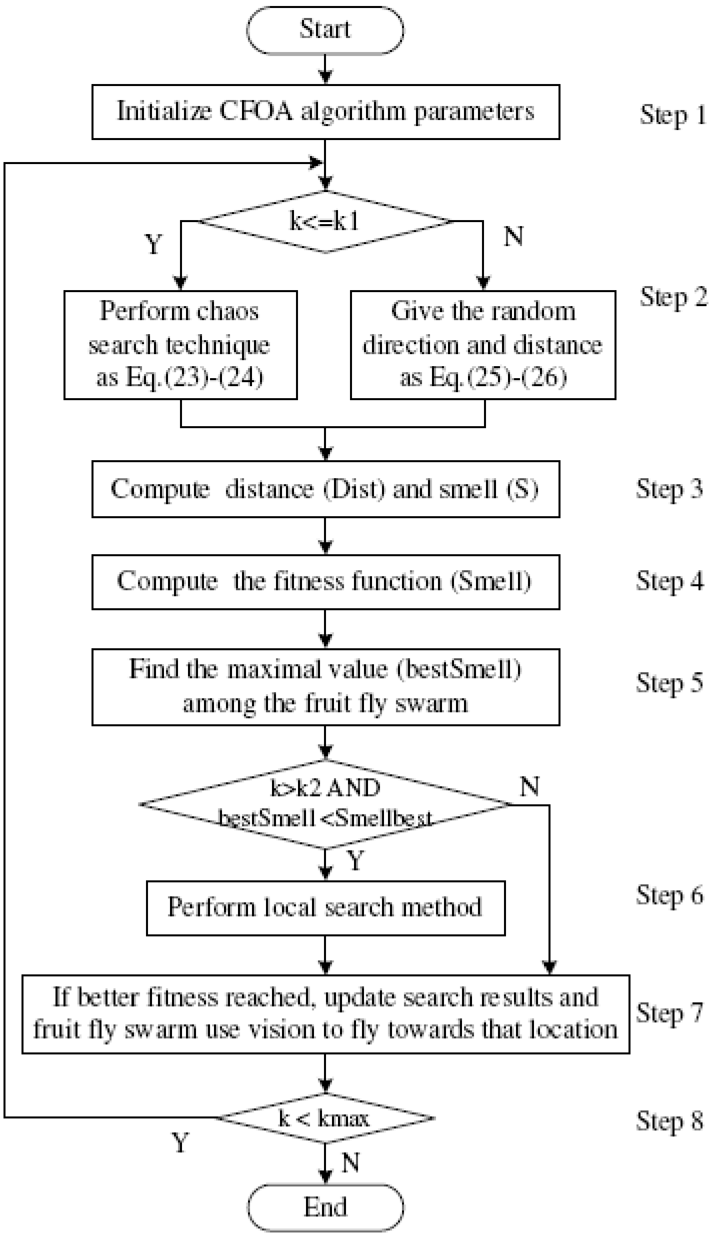

Start
Initialize CFOA algorithm parameters
Step 1
k<=k1
Y
N
Perform chaos search technique as Eq.(23)-(24)
Give the random direction and distance as Eq.(25)-(26)
Step 2
Compute distance (Dist) and smell (S)
Step 3
Compute the fitness function (Smell)
Step 4
Find the maximal value (bestSmell) among the fruit fly swarm
Step 5
k>k2 AND bestSmell<Smellbest
N
Y
Perform local search method
Step 6
If better fitness reached, update search results and fruit fly swarm use vision to fly towards that location
Step 7
k < kmax
Step 8
Y
N
End

Figure 8.2.1 CFOA operational processes and procedures (Source: Professor Yuan paper)

8.3 Solving Sin(x)/x by Adaptive Chaos FOA

This section uses "Sin (x) / x" and try again to move left or right to position 7, and the same as the previous section, we solve the "Sin (x-7) / (x-7)" function, using the chaos of fruit fly optimization algorithm (CFOA) to solve the extreme value of the function. Maximum value of the function is 1, CFOA the evolution of the number of iterations is 400, the number of 30 fly populations. After 400 generations of evolution, the search results maximum value is 1. In figure 8.3.1, The figure shows that the top graphic is the fly populations search path, we found the fly faster, and less flight path. Then, bottom graphic is the fly random search function extreme curve, after 400 generations, you can find the correct function extreme 1.

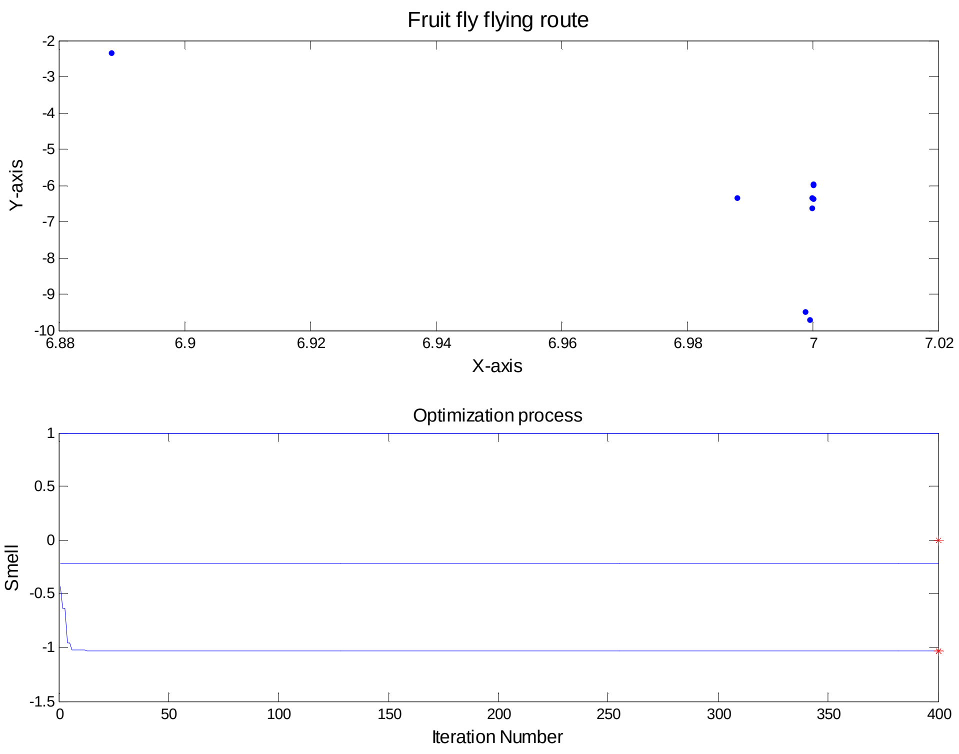


Figure 8.3.1 CFOA solve ” Sin(x-7)/(x-7)” extremes graphic result

The program code follows:

Program code search sin(x-7)/(x-7) extreme value

```
%*** Empty Memory
clc
clear
ss=10;
ssc=0.5;
pp=0;
pp2=0;
%*** Random initial fruit fly swarm location
X1_axis=ss*(rand()-0.5);
X2_axis=ss*(rand()-0.5);
X1_axisc=ss*(rand()-0.5);
X2_axisc=ss*(rand()-0.5);
maxgen=400;
sizepop=30;
%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly
for i=1:sizepop
X1(i)=X1_axis+(2*rand()-1)*ssc;
X2(i)=X2_axis+(2*rand()-1)*ssc;
%*** Like fitness function
Smell(i)=sin(X1(i)-7)/(X1(i)-7);
end

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location
[bestSmell bestindex]=max(Smell);
X1_axis=X1(bestindex);
X2_axis=X2(bestindex);
X1_axisc=X1_axis;
X2_axisc=X2_axis;
Smellbest=bestSmell;
%*** Iterative optimization start
for k=1:maxgen
    if mod(k,15)==0
```

```
for i=1: sizepop
X1_axisc=ss*(rand()-0.5);
X2_axisc=ss*(rand()-0.5);
X1(i)=X1_axis+(2*rand()-1)*ssc;
X2(i)=X2_axis+(2*rand()-1)*ssc;
Smell(i)=sin(X1(i)-7)/(X1(i)-7);
end
%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the 
fruit fly swarm will use vision to fly towards that location
  [bestSmell bestindex]=max(Smell);
X1_axisc=X1(bestindex);
X2_axisc=X2(bestindex);
  if bestSmell>Smellbest
        X1_axis=X1(bestindex);
        X2_axis=X2(bestindex);
        Smellbest=bestSmell;
    pp=pp+1;
  end
end
        mac=((maxgen-k-50)/(maxgen-50))^2;
for i=1:sizepop/2
 X1(i)=X1_axis+(2*rand()-1)*ss*ssc*mac;
 X2(i)=X2_axis+(2*rand()-1)*ss*ssc*mac;
 %***Like fitness function
Smell(i)=sin(X1(i)-7)/(X1(i)-7);
 end
 for i=sizepop/2+1: sizepop
 X1(i)=X1_axisc+(2*rand()-1)*ss*ssc*mac;
 X2(i)=X2_axisc+(2*rand()-1)*ss*ssc*mac;
%***Like fitness function
Smell(i)=sin(X1(i)-7)/(X1(i)-7);
  end
 %*** Keep the best smell concentration value and x, y coordinate , and at this moment, the
fruit fly swarm will use vision to fly towards that location
 [bestSmell bestindex]=max(Smell);
        X1_axisc=X1(bestindex);
        X2_axisc=X2(bestindex);
  if bestSmell>Smellbest
        X1_axis=X1(bestindex);
        X2_axis=X2(bestindex);
        Smellbest=bestSmell;
     end
  % Each generation of optimal value to record array yy
 for i=1:sizepop
 X1(i)=X1_axisc+(2*rand()-1)*(X1_axis-X1_axisc);
 X2(i)=X2_axisc+(2*rand()-1)*(X2_axis-X2_axisc);
Smell(i)=sin(X1(i)-7)/(X1(i)-7);
end
[bestSmell bestindex]=max(Smell);
        X1_axisc=X1(bestindex);
        X2_axisc=X2(bestindex);
    if bestSmell>Smellbest
        X1_axis=X1(bestindex);
        X2_axis=X2(bestindex);
        Smellbest=bestSmell;
     pp2=pp2+1;
  end
  yy(k)=Smellbest;
  X1best(k)=X1_axis;
  X2best(k)=X2_axis;
 End
```

%* Draw the optimization process of trends**

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('Smell','fontsize',12);
figure(2)
plot(X1best,X2best,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
```

8.4 Solving Schaffer's by Adaptive Chaos FOA

This section we uses "Schaffer's function" and using the chaos of fruit fly optimization algorithm (CFOA) to solve the extreme value of the function. Maximum value of the function is 1, CFOA the evolution of the number of iterations is 400, the number of 30 fly populations. After 400 generations of evolution, the search results maximum value is 1. In figure 8.4.1, The figure shows that the top graphic is the fly populations search path, we found the fly faster, and less flight path. Then, bottom graphic is the fly random search function extreme curve, after 400 generations, you can find the correct function extreme 1.

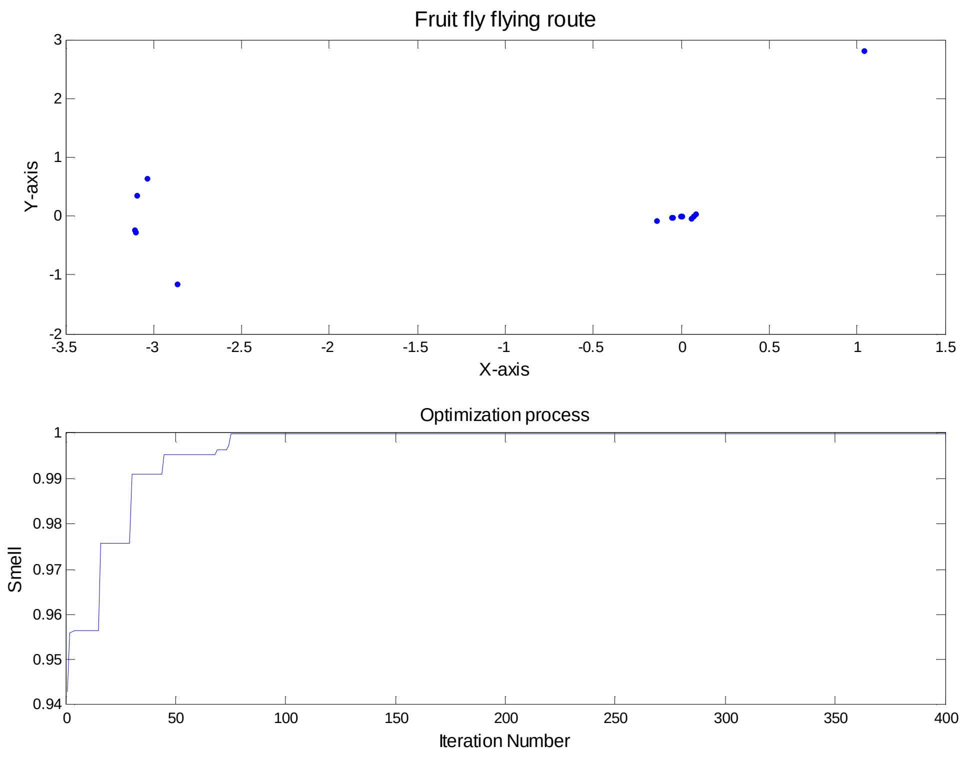


Figure 8.4.1 CFOA solve ” Schaffer's” extremes graphic result

The program code follows:

Program code search Schaffer extreme value

```
%*** Empty Memory
clc
clear
ss=10;
ssc=0.5;
pp=0;
pp2=0;
%*** Random initial fruit fly swarm location
X1_axis=ss*(rand()-0.5);
X2_axis=ss*(rand()-0.5);
X1_axisc=ss*(rand()-0.5);
X2_axisc=ss*(rand()-0.5);
maxgen=400;
sizepop=30;
%*** Give the random direction and distance for the search of food using osphresis by an individual fruit fly
for i=1:sizepop
X1(i)=X1_axis+(2*rand()-1)*ssc;
X2(i)=X2_axis+(2*rand()-1)*ssc;

%*** Like fitness function

Smell(i)=0.5-(sin((X1(i).^2+X2(i).^2).^0.5).^2-0.5)/(1+0.001*(X1(i).^2+X2(i).^2).^2);
end

%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the fruit fly swarm will use vision to fly towards that location
[bestSmell bestindex]=max(Smell);
X1_axis=X1(bestindex);
X2_axis=X2(bestindex);
X1_axisc=X1_axis;
X2_axisc=X2_axis;
Smellbest=bestSmell;
%*** Iterative optimization start
for k=1:maxgen
    if mod(k,15)==0
for i=1: sizepop
```

```
X1_axisc=ss*(rand()-0.5);
X2_axisc=ss*(rand()-0.5);
X1(i)=X1_axis+(2*rand()-1)*ssc;
X2(i)=X2_axis+(2*rand()-1)*ssc;
Smell(i)=0.5-(sin((X1(i).^2+X2(i).^2).^0.5).^2-0.5)/(1+0.001*(X1(i).^2+X2(i).^2).^2);
end
%*** Keep the best smell concentration value and x, y coordinate , and at this moment, the
fruit fly swarm will use vision to fly towards that location
  [bestSmell bestindex]=max(Smell);
X1_axisc=X1(bestindex);
X2_axisc=X2(bestindex);
  if bestSmell>Smellbest
      X1_axis=X1(bestindex);
      X2_axis=X2(bestindex);
      Smellbest=bestSmell;
    pp=pp+1;
  end
end
      mac=((maxgen-k-50)/(maxgen-50))^2;
for i=1:sizepop/2
 X1(i)=X1_axis+(2*rand()-1)*ss*ssc*mac;
 X2(i)=X2_axis+(2*rand()-1)*ss*ssc*mac;
 %***Like fitness function
Smell(i)=0.5-(sin((X1(i).^2+X2(i).^2).^0.5).^2-0.5)/(1+0.001*(X1(i).^2+X2(i).^2).^2);
 end
 for i=sizepop/2+1: sizepop
 X1(i)=X1_axisc+(2*rand()-1)*ss*ssc*mac;
 X2(i)=X2_axisc+(2*rand()-1)*ss*ssc*mac;
 %***Like fitness function
Smell(i)=0.5-(sin((X1(i).^2+X2(i).^2).^0.5).^2-0.5)/(1+0.001*(X1(i).^2+X2(i).^2).^2);
  end
 %*** Keep the best smell concentration value and x, y coordinate , and at this moment, the
fruit fly swarm will use vision to fly towards that location
 [bestSmell bestindex]=max(Smell);
      X1_axisc=X1(bestindex);
      X2_axisc=X2(bestindex);
```

```
   if bestSmell>Smellbest
         X1_axis=X1(bestindex);
         X2_axis=X2(bestindex);
         Smellbest=bestSmell;
      end
   % Each generation of optimal value to record array yy
  for i=1:sizepop
  X1(i)=X1_axisc+(2*rand()-1)*(X1_axis-X1_axisc);
  X2(i)=X2_axisc+(2*rand()-1)*(X2_axis-X2_axisc);
Smell(i)=0.5-(sin((X1(i).^2+X2(i).^2).^0.5).^2-0.5)/(1+0.001*(X1(i).^2+X2(i).^2).^2);
end
[bestSmell bestindex]=max(Smell);
         X1_axisc=X1(bestindex);
         X2_axisc=X2(bestindex);
     if bestSmell>Smellbest
         X1_axis=X1(bestindex);
         X2_axis=X2(bestindex);
         Smellbest=bestSmell;
      pp2=pp2+1;
   end
   yy(k)=Smellbest;
   X1best(k)=X1_axis;
   X2best(k)=X2_axis;
  End
```

<u>%* Draw the optimization process of trends</u>**

```
figure(1)
plot(yy)
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('Smell','fontsize',12);
figure(2)
plot(X1best,X2best,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
```

Problems and Applications

1. Try writing a program to test CFOA computing Rastrigin's function extreme, and observe the degree of convergence fruit fly search path and whether there are differences with the FOA?
2. Use Chapter 3 “ZSCORE Model”, to verify the effectiveness of CFOA.

Reference

[1] Holland, J.(1975), Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI.

[2] Dorigo, M. and Gambardella, L. M. (1997), Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem, IEEE Transactions on Evolutionary Computation, Vol.1, No. 1, pp.53-66.

[3] Eberhart, R.C. and Kennedy, J. (1995), new optimizer using particle swarm theory, Proc, Sixth International Symposium on Nagoya, Japan, pp.39-43.

[4] Pan, W.T., 2011. A new evolutionary computation approach: Fruit Fly Optimization Algorithm, 2011 Conference of Digital Technology and Innovation Management, Taipei. Program code on the website http://www.oitecshop.byethost16.com/FOA.html.

[5] Pan, W.T., 2011. A new fruit fly optimization algorithm: Taking the financial distress model as an example, Knowledge-Based Systems, Vol.26, pp.69-74, 2012.

[6] Altman, E. I., 1968, Financial Ratios, Discriminant Analysis and the

[7] Prediction of Corporate Bankruptcy, Journal of Finance, 23, 4, 589-609.

[8] Specht, D.F.(1990), Probabilistic neural networks and the polynomial adaline as complementary techniques for classification. IEEE Trans. on Neural Networks, Vol. 1, No. 1, pp.111-121.

[9] Specht, D.F.(1991), A general regression neural network. IEEE Tras. Neural Networks, 2(6), 568-576.

[10] Deng J. (1982), The control problems of grey system, System & Control Letters, No.5, pp. 288-294.

[11] Shi Feng et al.,(2010), MATLAB Neural Networks 30 case studies, Beijing University of Aeronautics and Astronautics Press.

[12] Wang, F. and Xia, H. (2008), Network traffic prediction based on grey neural network integrated model, International Conference on Computer Science and Software Engineering, pp.915-918.

[13] Vapnik, V.(1995), “Support-vector networks,” MachineLearning, Vol. 20, No. 3, pp. 273-297.

[14] Li, C., Xu, S., Li, W., Hu, L.(2012), A novel modified fly optimization algorithm for designing the self-tuning proportional integral derivative controller, Journal of Convergence Information Technology, Vol.7, No.16, pp.69-77.

[15] J. Y. Han and C.Z. Liu, (2013)，Adaptive chaos fruit fly optimization algorithm, Journal of Computer Applications，Vol.33, No.5, pp.1313-1333.

[16] Xiaofang Yuan∗ , Qian He, YaonanWang, (2013), Chaotic fruit fly optimization algorithm and application, Unpublished.

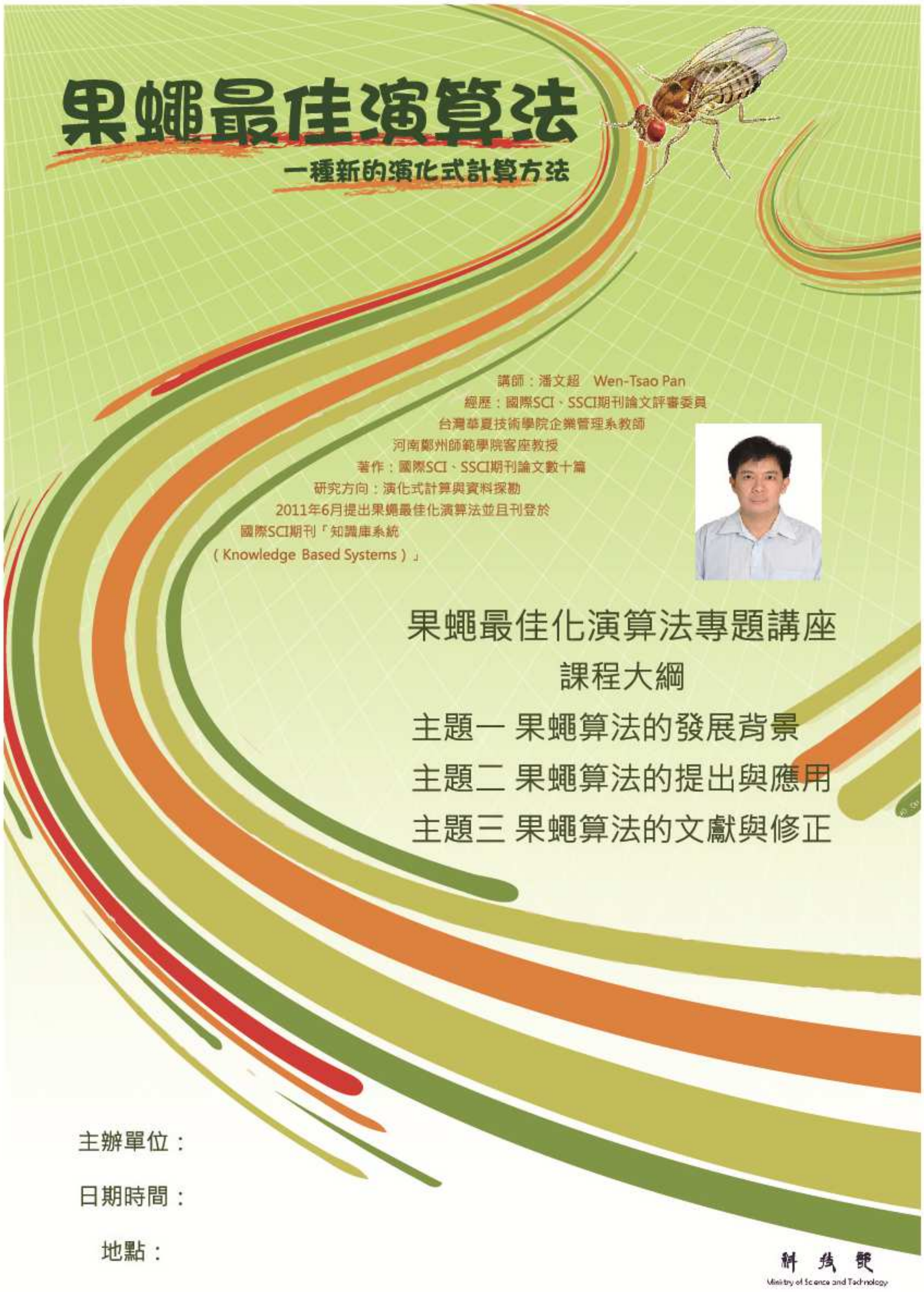
Appendix 1:

TSP_FOA Code

```
function [ tourGbest,L_best ] = TSP_FOA( cityCoor )
global N;
cityDist = GetCityDist(cityCoor);
nMaxGen=200;
sizepop = 200;
FF=zeros(sizepop,N);
for i=1:sizepop
    FF(i,:)=randperm(N);
end
D=CalPathLength(cityDist,FF);
Smell=1./D;
[bestSmell bestindex]=max(Smell);
bestAxis=FF(bestindex,:);
Smellbest=bestSmell;
xnew1=FF;
for g=1:nMaxGen
    for i=1:sizepop
        c1=round(rand*(N-2))+1;
        c2=round(rand*(N-2))+1;
        while c1==c2
            c1=round(rand*(N-2))+1;
            c2=round(rand*(N-2))+1;
        end
        chb1=min(c1,c2);
        chb2=max(c1,c2);
        cros=bestAxis(chb1:chb2);
        ncros=size(cros,2);
        for j=1:ncros
            for k=1:N
                if xnew1(i,k)==cros(j)
                    xnew1(i,k)=0;
                    for t=1:N-k
                        temp=xnew1(i,k+t-1);
                        xnew1(i,k+t-1)=xnew1(i,k+t);
                        xnew1(i,k+t)=temp;
                    end
                end
            end
        end
        xnew1(i,N-ncros+1:N)=cros;
        dist=0;
        for j=1:N-1
            dist=dist+cityDist(xnew1(i,j),xnew1(i,j+1));
        end
        dist=dist+cityDist(xnew1(i,1),xnew1(i,N));
        if D(i)>dist
            FF(i,:)=xnew1(i,:);
        end
        c1=round(rand*(N-1))+1;
        c2=round(rand*(N-1))+1;
        while c1==c2
            c1=round(rand*(N-2))+1;
            c2=round(rand*(N-2))+1;
        end
        temp=xnew1(i,c1);
        xnew1(i,c1)=xnew1(i,c2);
        xnew1(i,c2)=temp;
         dist=0;
        for j=1:N-1
            dist=dist+cityDist(xnew1(i,j),xnew1(i,j+1));
        end
        dist=dist+cityDist(xnew1(i,1),xnew1(i,N));
        if D(i)>dist
            FF(i,:)=xnew1(i,:);
        end
    %      FF(i,:)=TwoOpt(cityDist,FF(i,:));
    end
    rnd=round(rand*(sizepop-1))+1;
    FF(rnd,:) =TwoOpt(cityDist,FF(rnd,:));
    D=CalPathLength(cityDist,FF);
    Smell=1./D;
```

```
    [bestSmell bestindex]=max(Smell);
    if bestSmell>Smellbest
        bestAxis=FF(bestindex,:);
        Smellbest=bestSmell;
    end
    L_best(g)=1/Smellbest;
    tourGbest = bestAxis;
end
```

Appendix 2:



果蠅最佳演算法

一種新的演化式計算方法

講師：潘文超　Wen-Tsao Pan

經歷：國際SCI、SSCI期刊論文評審委員

台灣華夏技術學院企業管理系教師

河南鄭州師範學院客座教授

著作：國際SCI、SSCI期刊論文數十篇

研究方向：演化式計算與資料探勘

2011年6月提出果蠅最佳化演算法並且刊登於

國際SCI期刊「知識庫系統

（Knowledge Based Systems）」

果蠅最佳化演算法專題講座

課程大綱

主題一 果蠅算法的發展背景

主題二 果蠅算法的提出與應用

主題三 果蠅算法的文獻與修正

主辦單位：

日期時間：

地點：

科技部

Ministry of Science and Technology