

Auto2Fit User Manual

7D Soft High Technology Inc.

May/1st/2009

Chapter 1 Introduction

1.1 General Information about Auto2Fit

Auto2Fit is a software package for the analysis of mathematical optimization problem. It is developed by 7D-Soft High Technology Inc independently with complete intellectual property rights. It is in a world-leading place in the area of non-linear regression, curve fitting, non-linear complex model parameters estimation and solving, linear / nonlinear programming compared with other similar commercial software. Besides easy-to-use user interface, its computation core is based on the innovative research results, Universal Global Optimization (UGO) algorithm, from more than a decade of scientific research work done by experts in 7D-Soft. This algorithm solved the problem of the initial value must be given for numerical optimization calculation, namely, the initial value is given by Auto2Fit randomly instead of user, then find the optimal solution through its Universal Global Optimization algorithm. Take nonlinear regression as an example, the most famous software packages in this area, such as Matlab, OriginPro, SAS, SPSS, DataFit, GraphPad, etc., all need the appropriate initial values of parameters provided or guessed by users for finding the optimal solution. If the initial value of the parameter is set incorrectly, then result is difficult to converge and it is impossible to get the correct answers finally. In practice, it is quite difficult to provide (guess) the appropriate initial values of parameters for most users. Especially when the amount of parameters is large, guess of initial start values would be a nightmare for users. With the superiors of optimization and fault tolerance, Auto2Fit can achieve the correct result with random initial start value in most of cases (>90%).

1.2 State-of-the-art of similar software national and international

In the area of comprehensive data analysis, there is no doubt that foreign software dominates markets. In the area of nonlinear curve fit, parameter optimization, there are several well-known software products, such as OriginPro, Matlab, SAS, SPSS, DataFit, GraphPad, TableCurve2D, TableCurve3D, etc.. No matter what the user interface is and how the history and reputation is, the most used algorithms, i.e., Levenberg-marquardt or Simplex Method, can all categorized to local optimization algorithm. Thus how to define appropriate initial parameters is always a bottleneck that is difficult to overcome and some practical problems may never get correct solutions. Nationally though there is few data analysis software in the market, they cannot compete with foreign similar products due to poor functionality and weakness in theory and methodology, not mention to make some noises in the world. With its innovative algorithm theory, Auto2Fit is advanced than any known software package in the world in the area of nonlinear curve fit and parameter estimation. The English version of Auto2Fit is on sale in many countries, like USA, Germany, France, UK, Finland, Sweden, Netherlands, South Africa, Australia, New Zealand,

Turkey, etc.

1.3 The optimization algorithms adopted in Auto2Fit

The optimization algorithms Auto2Fit adopt include:

- 1) Universal Global Optimization (UGO)
- 2) Simplex Method (SM) + Universal Global Optimization (UGO)
- 3) Differential Evolution (DE)
- 4) Max Inherit Optimization (MIO)
- 5) Genetic Algorithms (GA)
- 6) Simulated Annealing (SA)
- 7) Particle Swarm Optimization (PSO)
- 8) Self-Organizing Migrating Algorithms (SOMA)
- 9) Tabu Search (TS)
- 10) Simplex Linear Program

1.4 The application areas of Auto2Fit

The application areas of Auto2Fit include:

- 1) Auto-calibration
- 2) Parameter estimation
- 3) Linear or nonlinear curve fit and regression
- 4) Nonlinear simultaneous system equations solving
- 5) Ordinary differential equation (ODE) and simultaneous equation, initial value problems (IVP) and boundary value problems (BVP)
- 6) Function optimization with any dimensions, including implicitly function
- 7) Function chart, chart of implicit function
- 8) Linear, nonlinear or integer programming
- 9) Combination optimization
- 10) Advanced calculator

1.5 Features of Auto2Fit

- 1) Powerful: it is the only software packages that can find the optimal solution of nonlinear regression test dataset of National Institute of Standards and Technology (NIST) using random initial start-values in current world market.
- 2) It can be widely used in hydrology, water resources and other engineering optimization problems. As embedded with VB and Pascal, it can help describing and dealing with complex optimization model.
- 3) Link the objective function of external dynamic link library (dll) or command-line executive file (exe) which are compiled from any other programming language (such as, C++, FORTRAN, Basic, Pascal ...).

- 4) Nonlinear curve fit can deal with any formation of user-defined equations, any number of parameters and variables, batch processing of data fitting, weight fitting, fitting with constraints, fitting with missing variables.
 - 5) Process multiple data files simultaneously when model does auto-calibration.
 - 6) Easily handle multiple outputs.
 - 7) Real-time calculation results display.
 - 8) Read and save different format of files, such as Excel, CSV, etc.
 - 9) User-friendly interface, easy to use.
 - 10) Come with more than one hundred examples, covering almost all optimization issues.
- Users can easily understand the usage of Auto2Fit through different type of examples.

1.6 Key words of Auto2Fit

Main key words:

Name of key words	Meaning and example
Parameter	<p>Parameter definition</p> <p>E.g.: Define a, b, c, d, four parameters: <i>Parameter a, b, c, d;</i></p> <p>E.g.: Define a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, ten parameters:</p> <p style="padding-left: 40px;"><i>Parameter a1, a2, a3, a4, a5, a6, a7, a8, a9, a10;</i></p> <p>Can be abbreviated: <i>Parameter a(1:10);</i></p> <p>or: <i>Parameter a(10);</i></p> <p>E.g.: Define a, ranged in [-1,1], whose initial value is 0.5</p> <p style="padding-left: 40px;"><i>Parameter a = 0.5[-1,1];</i></p> <p>E.g.: Define a as an integer, ranged in [-100,100]</p> <p style="padding-left: 40px;"><i>Parameter a[-100,100,0];</i> or <i>IntParameter a=[-100,100];</i></p>
ParameterDomain	<p>Define parameters' range in a batch</p> <p>E.g.: Define parameter a, ranged in [-1,1], other parameters ranged in [0,10];</p> <p style="padding-left: 40px;"><i>Parameter a = 0.5[-1,1], b, c, d, e, f, g;</i></p> <p style="padding-left: 40px;"><i>ParameterDomain = [0,10];</i></p>
BinParameter	<p>Define 0-1 parameter</p> <p>E.g.: Define a as a parameter whose value is either 0 or 1</p> <p style="padding-left: 40px;"><i>BinParameter a;</i></p>
IntParameter	<p>Define positive interger parameter</p> <p>E.g.: Define a as positive integer</p> <p style="padding-left: 40px;"><i>Parameter a=[0,,0];</i> or <i>IntParameter a;</i></p>
StartRange	<p>Define the range of initial value of parameter</p> <p>E.g.: Define the range of initial value of parameter, a, as [-100,100]</p> <p style="padding-left: 40px;"><i>StartRange a = [-100,100];</i></p>
Variable	<p>Define variable</p> <p>E.g.: Define three variables, x, y, and z: <i>Variable x, y, z;</i></p>
QuickReg	<p>Set fast fitting function</p>
Function	<p>Define function</p> <p>E.g.: Curve fit with two variables: <i>Function y = a + b*exp(c - x);</i></p>

	E.g.: Function optimization with two variables: <i>Function</i> $(x+((2-x)*(2+y))^2)*\sin(x*y);$
Constant	Define constant E.g.: <i>Constant</i> $a=1, b=2;$ E.g.: <i>Constant</i> $p(3)=[1,2,3];$
ConstStr	Define constant string E.g.: Curve fit with two variables: <i>Function</i> $y = a * (c-x)^2 + b * \exp((c-x)^4);$ can be expressed as: <i>ConstStr</i> $B = (c-x)^2$ <i>Function</i> $y = a*B + b * \exp(B^2);$
VarConstant	Define varconstant
VarParameter	Define varparameter
Data	Define start of data
RowData	Define start of data, with row formation
DataFile	Define data file
NewDivision	Define new code section
SubDivision	Define sub code section
StartProgram	Define start of program
EndProgram	Define end of program
Maximum	Find the maximum
Minimum	Find the minimum
PlotFunction	Plot function
Algorithms	Define optimization algorithm
Exclusive	Define exclusive problem, such as TSP problem
MutliRun	
HotRun	
SharedModel	Define share parameter
DataSet EndDataSet	Define dataset Define end of dataset
RowDataSet EndRowDataSet	
MinFunction	Find the solution of getting the maximum value of a function
MaxFunction	Find the solution of getting the minimum value of a function
PlotParaFunction	Plot graph of a function
Title	Define a title of a code section
RegType	Set the least absolute value method fitting
MDataSet EndMDataSet	Define network node data format, equivalent to matrix format
ConstrainedResult	Value of constrained function in programming mode
ObjectiveResult	Value of objective function in programming mode
BatchFileModel	
FullLoopModel	Full loop calculation mode
MinMax	Solution of minmax optimization problem

SharedModel	Shared model of multiple functions fitting
WeightedReg	Weighted fitting
ExeParameterFile	Define parameter output file when invoke external command line executive file
ExeObjectiveFile	Define objective function output file when invoke external command line executive file
MaxIteration	

Auto2Fit has another three special delimiters:

■ **Summation: *Sum***

E.g., $\sum_{i=1}^n (x_i \cdot \sin(x_i + 1))$, expressed in Auto2Fit as: *Sum(i=1:n)(x[i]*sin(x[i]+1))*

If subscripts are all i, it can be expressed for short as:

*Sum(i=1:n,x)(x*sin(x+1))*

If the start value of subscript is 1, it can be expressed for short as:

*Sum(i=n,x)(x*sin(x+1))*

■ **The product of multiplication: *Prod***

E.g., $\prod_{i=1}^n (x_i \cdot \sin(x_i + 1))$, expressed in Auto2Fit as: *Prod(i=1:n)(x[i]*sin(x[i]+1))*

If subscripts are all i, it can be expressed for short as:

*Prod(i=1:n,x)(x*sin(x+1))*

If the start value of subscript is 1, it can be expressed for short as:

*Prod(i=n,x)(x*sin(x+1))*

■ **Loop delimiter: *For***

E.g., system equations:
$$\begin{cases} x_1 \sin(x_1) - 1 + x_1 = 0 \\ x_2 \sin(x_2) - 2 + x_2 = 0 \\ x_3 \sin(x_3) - 3 + x_3 = 0 \\ x_4 \sin(x_4) - 4 + x_4 = 0 \\ x_5 \sin(x_5) - 5 + x_5 = 0 \end{cases}$$

Code of Auto2Fit:

```
Function  x1*sin(x1)-1+x1 = 0;
          x2*sin(x2)-2+x2 = 0;
          x3*sin(x3)-3+x3 = 0;
          x4*sin(x4)-4+x4 = 0;
          x5*sin(x5)-5+x5 = 0;
```

Use For expression, expressed as follows:

Function For(i=1:5)(x[i]*sin(x[i])-i+x[i]=0);

or Function For(i=1:5,x)(x*sin(x)-i+x=0);

or Function For(i=5,x)(x*sin(x)-i+x=0);

1.7 Mathematical functions supported in Auto2Fit

No.	Function	Note	Example
1	Abs(X: Real): Real;	Absolute value function	Abs(-0.25) = 0.25
2	Arccos(X: Real): Real;	Inverse cosine function	Arccos(-0.25) = 1.823476582
3	Arccosh(X: Real): Real;	Inverse cosine hyperbolic function	Arccosh(-0.25) = 0
4	Arcsin(X: Real): Real;	Inverse sine function	Arcsin(-0.25) = -0.2526802551
5	Arcsinh(X: Real): Real;	Inverse sine hyperbolic function	Arcsinh(-0.25) = -0.247466461
6	Arctan(X: Real): Real;	Inverse tangent function	Arctan(-0.25) = -0.2449786631
7	Arctanh(X: Real): Real;	Inverse tangent hyperbolic function	Arctanh(-0.25) = -0.255412811
8	ATan(X: Real): Real;		ATan(-0.25) = -0.2449786631
9	ATand(X: Real): Real;		ATand(-0.25) = -0.0043632954
10	BessI(N: Integer, X: Real): Real;	Bessel function I	BessI(2, 0.25) = 0.0078532695
11	BessI0(X: Real): Real;	Bessel function I0	BessI0(0.25) = 1.015686133
12	BessI1(X: Real): Real;	Bessel function I1	BessI1(0.25) = 0.1259791086
13	BessJ(N: Integer, X: Real): Real;	Bessel function J	BessJ(2, 0.25) = 0.0077718892
14	BessJ0(X: Real): Real;	Bessel function J0	BessJ0(0.25) = 0.9844359314
15	BessJ1(X: Real): Real;	Bessel function J1	BessJ1(0.25) = 0.1240259773
16	BessK(N: Integer, X: Real): Real;	Bessel function K	BessK(2, 0.25) = 31.51771458
17	BessK0(X: Real): Real;	Bessel function K0	BessK0(0.25) = 1.541506736
18	BessK1(X: Real): Real;	Bessel function K1	BessK1(0.25) = 3.747025981
19	BessY(N: Integer, X: Real): Real;	Bessel function Y	BessY(2, 0.25) = -20.70126879
20	BessY0(X: Real): Real;	Bessel function Y0	BessY0(0.25) = -0.9315730315
21	BessY1(X: Real): Real;	Bessel function Y1	BessY1(0.25) = -2.704105228
22	Beta(X1: Real; X2: Real): Real;	Beta function	Beta(0.3, 0.4) = 5.112091244
23	BetaCDF(X: Real; A[>0]: Real; B[>0]: Real): Real;	Beta cumulative distribution function	Betacdf(0.6, 0.3, 0.7) = 0.7773849481
24	BetaPDF(X: Real; A[>0]: Real; B[>0]: Real): Real;	Beta probability density function	BetaPdf (0.2, 0.6, 0.2) = 0.3875354323
25	Bin2Real(X1: String; X2: Integer): Real;		Bin2Real(23, 3) = 9
26	BinomialCDF(n1: Integer; n2[<n1]: Integer; X[0,1]: Real): Real;		BinomialCdf (0.3, 2, 0.5)= 0.25
27	BinomialPDF(n1: Integer; n2[<n1]: Integer; X[0,1]: Real): Real;		BinomialPdf(0.4,6,0.3)= 0.117649

28	ChiSquareCDF(X: Real; n: Integer): Real;		ChisquareCdf(0.3,2) = 0.13929
29	ChiSquarePDF(X: Real; n: Integer): Real;		ChisquarePdf(0.4,3) = 0.20657
30	Cos(X: Real): Real;	Cosine function	Cos(2.3) = -0.6662760213
31	Cosd(X: Real): Real;		Cosd(3.5) = 0.9981347984
32	Cosh(X: Real): Real;	Cosine hyperbolic function	Cosh(0.6) = 1.185465218
33	Cot(X: Real): Real;		Cot(5.4) = -0.8213276958
34	Coth(X: Real): Real;		Coth(1.2) = 1.199537544
35	Erf(X: Real): Real;	Error function	Erf(0.6) = 0.6038561848
36	Erfc(X: Real): Real;		Erfc(3.2) = 6.031483983e-6
37	Exp(X: Real): Real;	Exponential function	Exp(1.8) = 6.049647464
38	ExponentialCDF(X: Real; A: Real): Real;	Exponential cumulative distribution function	ExponentialCDF(0.5,0.9) = 0.4262465793
39	ExponentialCDFInv(P[0,1]: Real; A: Real): Real;	Inverse of exponential cumulative distribution function	ExponentialCDFInv(0.3,0.4)= 0.1426699776
40	ExponentialPDF(X: Real; A: Real): Real;	Exponential probability density function	ExponentialPDF(0.12,0.54)= 1.482847042
41	FCDF(X: Real; n1: Integer; n2: Integer): Real;	F cumulative distribution function	FCDF(0.6, 3, 1) = 0.2871897411
42	FPDF(X: Real; n1: Integer; n2: Integer): Real;	F probability density function	FPDF(0.3, 3, 2) = 0.5961681487
43	Gamma(X: Real): Real;	Gamma function	Gamma(0.6) = 1.489192249
44	GammaCDF(X: Real; A[>0]: Real; B[>0]: Real): Real;	Gamma cumulative distribution function	GammaCdf(0.3, 0.1, 0.2) = 0.988655
45	GammaCDFInv(P[0,1]: Real; A[>0]: Real; B[>0]: Real): Real;	Inverse of Gamma cumulative distribution function	GammaCDFInv(0.1,2,3) = 1.595434825
46	GammaPDF(X: Real; A[>0]: Real; B[>0]: Real): Real;	Gamma probability density function	GammaPDF(0.3, 0.4, 0.2) = 0.3943490019
47	Ln(X: Real): Real;	Natural logarithm function	Ln(3.2) = 1.16315081
48	Log(X: Real): Real;	Logarithm base 10	Log(3.2) = 0.5051499783
49	Logn(Base: Real; X: Real): Real;	Logriahm base as set value	Logn(10, 3.2) = 0.5051499783
50	Max(X1: Real; X2: Real): Real;	Maximum of two numbers	Max(2.3,3.2) = 3.2
51	Min(X1: Real; X2: Real): Real;	Minimum of two numbers	Min(2.3,3.2) = 2.3
52	NormalCDF(X: Real): Real;	Normal cumulative distribution function	NormalCDF(0.3) = 0.6179114222
53	NormalPDF(X: Real): Real;	Normal probability density function	NormalPDF(0.9) = 0.2660852499
54	PoissonCDF(n: integer; X: Real): Real;	Poisson cumulative distribution function	PoissonCDF(2,0.5) = 0.985612322
55	PoissonPDF(n: Integer; X: Real): Real;	Poisson probability density function	PoissonPDF(2,0.5) = 0.07581633246

56	Power(Base: Real; Exponent: Real): Real;	Exponential function base as set value	Power(2.2, 3.1) = 11.52153413
57	Real2Bin(X: Real; n: Integer): String;		Real2Bin(0.5, 2) = 0.101
58	Round(X: Real): Real;	Rounding function	Round(0.364) = 0
59	Sign(X: Real): Real;	Signum function	Sign(2.3) = 1
60	Sin(X: Real): Real;	Sine function	Sin(3.2) = -0.05837414343
61	Sind(X: Real): Real;		Sind(0.6) = 0.01047178412
62	Sinh(X: Real): Real;	Sine hyperbolic function	Sinh(5.6) = 135.2113548
63	Sqr(X: Real): Real;	Square function	Sqr(4.2) = 17.64
64	Sqrt(X: Real): Real;	Square root function	Sqrt(3.5) = 1.870828693
65	StudentCDF(X: Real; n: Integer): Real;	Student cumulative distribution function	StudentCDF(0.5, 2) = 0.6666666667
66	StudentPDF(X: Real; n: Integer): Real;	Student probability density function	StudentPDF(0.5, 2) = 0.2962962963
67	Tan(X: Real): Real;	Tangent function	Tan(3.2) = 0.05847385446
68	Tand(X: Real): Real;		Tand(6.5) = 0.1139356083
69	Tanh(X: Real): Real;	Tangent hyperbolic function	Tanh(0.9) = 0.7162978702
70	Trunc(X: Real): Real;	Integral function	Trunc(3.2) = 3

1.8 User interface

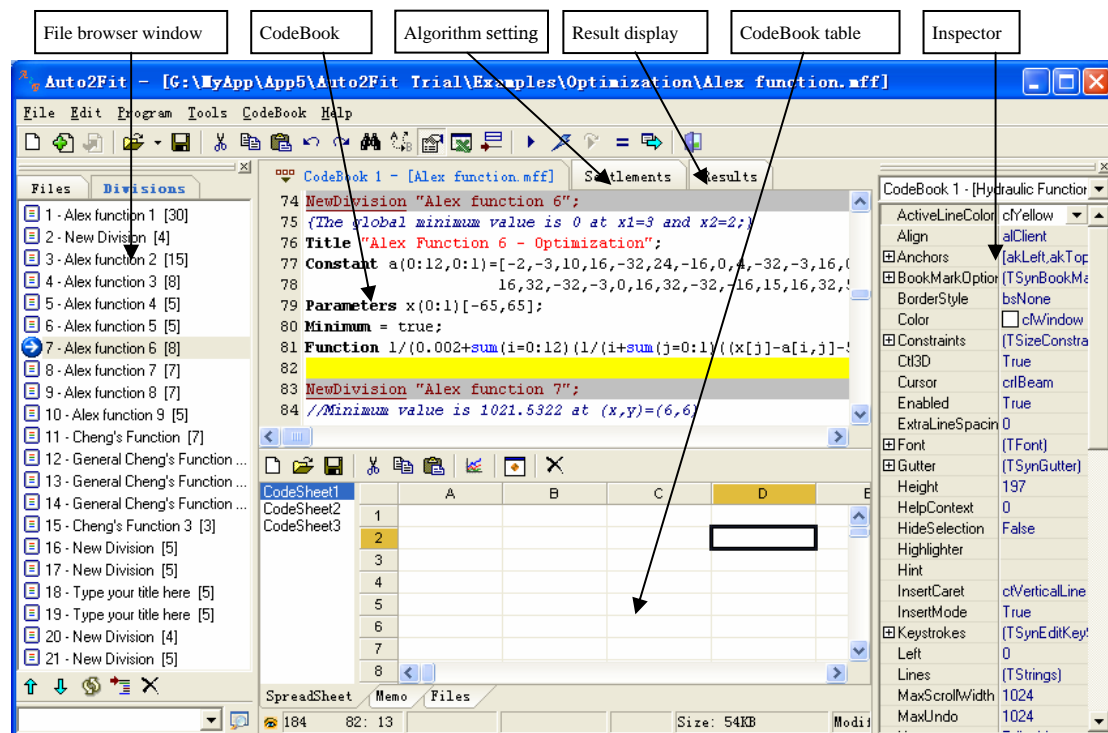


图 1-1 Auto2Fit main interface

Toolbar setting: right click the toolbar, choose “Customize Toolbar”, a toolbar icon setting dialog box will popup like follows,

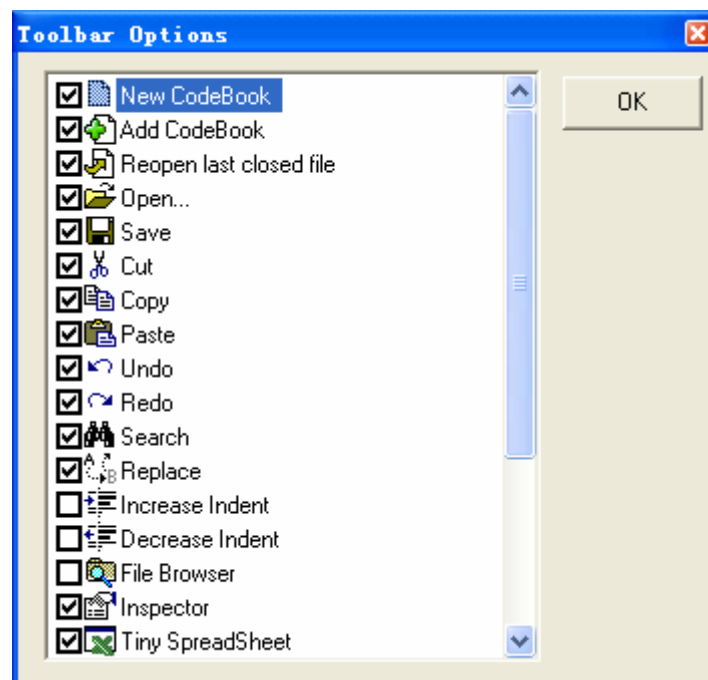
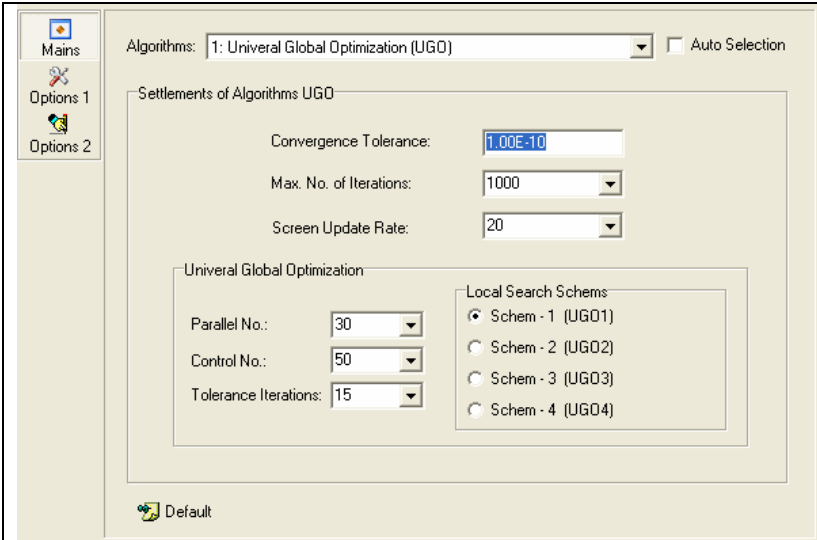
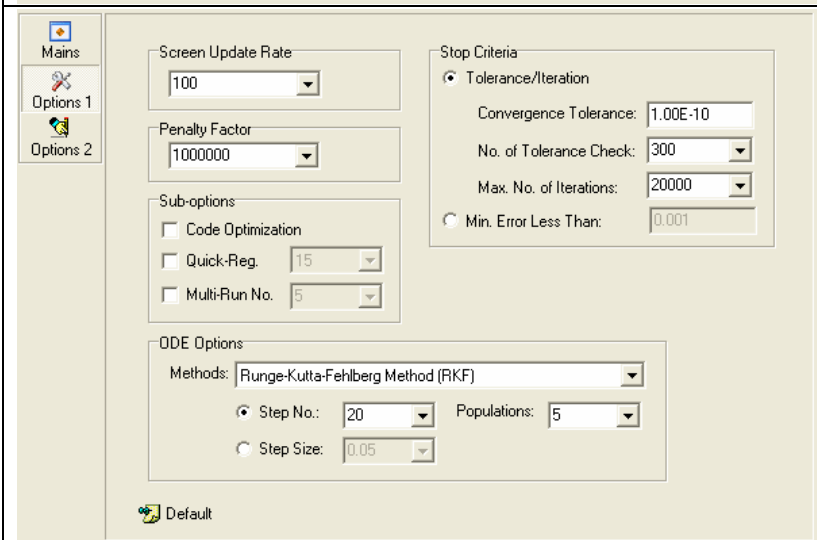
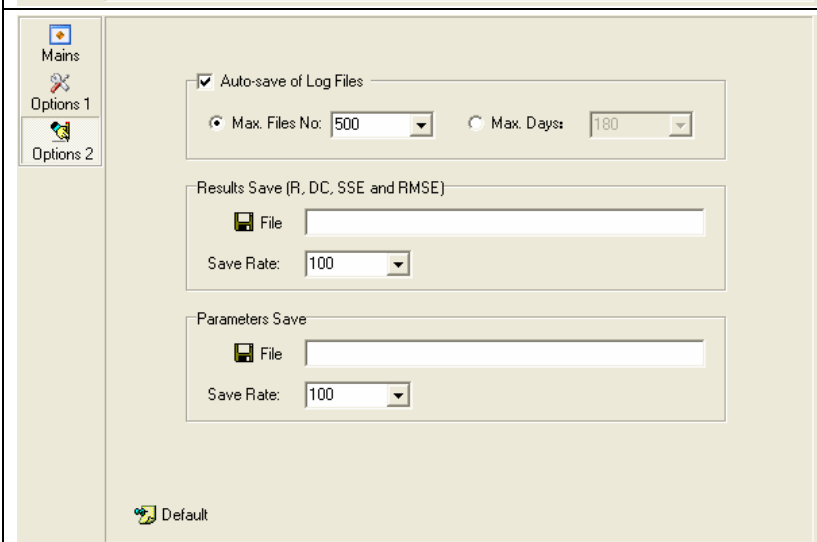


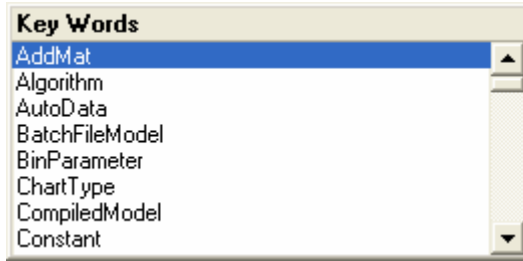
Figure 1-2 Toolbar icon setting

Algorithm setting

	<p>Main setting:</p> <ul style="list-style-type: none"> Set algorithm and its controlling parameter
	<p>Option setting 1:</p> <ul style="list-style-type: none"> Set algorithm controlling parameter Set algorithm and related parameters for solving ODE (ordinary differential equation)
	<p>Option setting 2:</p> <ul style="list-style-type: none"> Set log file option Set save option

1.9 Shortcuts and combinations in Auto2Fit

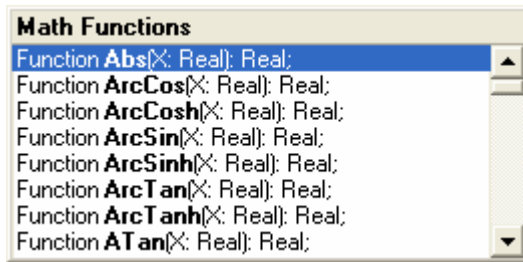
- Key combination for key word fast input window: press “Ctrl+K” in CodeBook



When window popup, input key words in order. User can search and input key word needed quickly.

Figure 1-3 Key combination for key word fast input

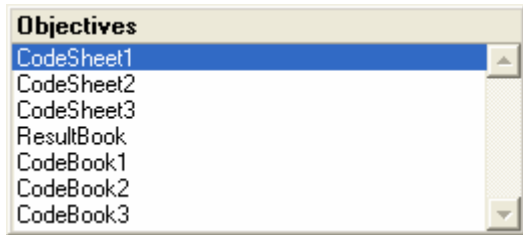
- Key combination for mathematical function fast input: press “Ctrl+M” in CodeBook



When window popup, input letters of mathematical function in order. User can search and input mathematical function needed quickly.

Figure 1-4 Key combination for mathematical function fast input

- Key combination for CodeBook and CodeBook spreadsheets: press “Ctrl+J” in CodeBook



When window popup, choose CodeBook or CodeBook spreadsheets you want. When the name of list in CodeBook differs from the name of sheet (the name in the bracket is the name of list), choose the corresponding name of sheet.

Figure1-5 Key combination for CodeBook and CodeBook spreadsheets

- Recover the source code executed previous time: Press “Ctrl+Shift+T” in codebook

Due to some wrong code input or other unknown reasons sometimes, Auto2Fit will shut down when user finishes input code and presses calculation command. If the code were not saved before, how to recover the code input before? Re-launch Auto2Fit, open a new CodeBook, press “Ctrl+Shift+T”, then the code executed previous time can be recovered.

User can write codes for different problems within one CodeBook, which are differentiated by key word “NewDivision”. User can open multiple CodeBooks to edit codes. Rich text, such as graph, table, equation, can be added in one code file. Different formatted files can also be added.

1.10 CodeBook spreadsheet

The function of CodeBook spreadsheet is similar to spreadsheet in previous section. The data in it can be invoked directly in CodeBook. When save file, data is also saved in the same file.

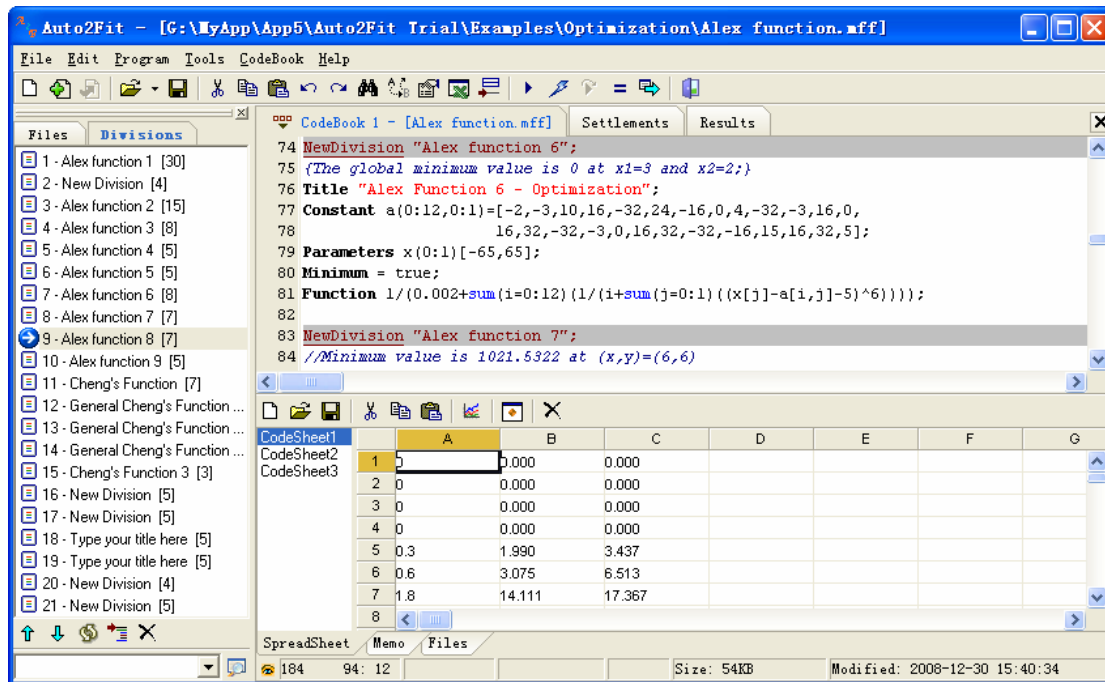


Figure 1-7 Auto2Fit codebook spreadsheets

1.11 Data processing spreadsheet

Auto2Fit comes with spreadsheet similar to Excel. It has functions of multi-form, equation support, data input/output directly to Excel and text file (.txt, .csv), tree-style form management, easy to read and classification, easy to data process in front-end and back-end.

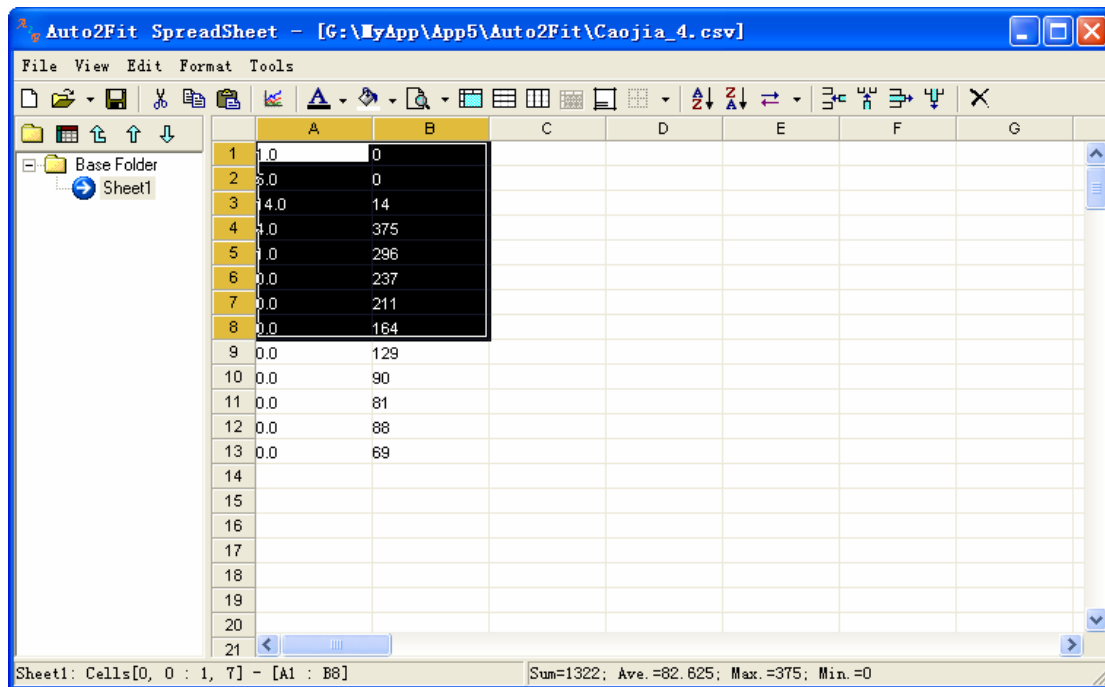


Table 1-8 Auto2Fit spreadsheets

1.12 Optimization algorithm setting

There are 10 optimization algorithms in total in Auto2Fit. Which algorithm to choose for different problem? General speaking:

- ✧ Nonlinear regression, curve fit, equation and equations solving, unconstrained function optimization:
 - ◆ Universal Global Optimization – UGO
 - ◆ Simplex Method – SM + Universal Global Optimization – UGO
 - ◆ Differential evolution algorithm
 - ◆ Max inherit optimization
- ✧ Constrained function optimization:
 - ◆ Simplex Method –SM + Universal Global Optimization –UGO
 - ◆ Differential evolution algorithm
 - ◆ Universal Global Optimization – UGO
 - ◆ Max inherit optimization
- ✧ Linear programming:
 - ◆ Simplex Linear Program – SLP
 - ◆ Standard Simplex Method – SM + Universal Global Optimization – UGO
- ✧ Combination optimization:
 - 1) Max inherit algorithm
 - 2) Tabu search algorithm
 - 3) Simulated annealing
 - 4) Genetic algorithm

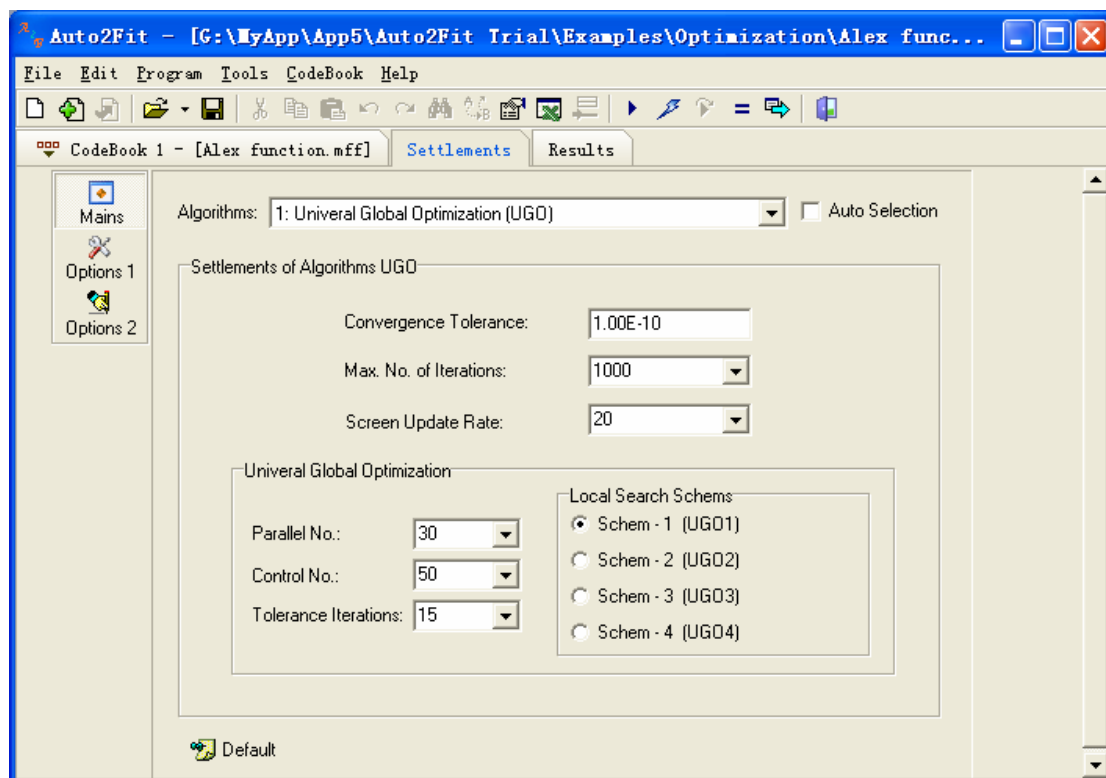


Figure 1-9 Optimization algorithm setting

In most of cases (>90%), the default optimization algorithm settings can meet the

requirements. For some more complex problem, parameters can be changed and the way to do it is shown in the following table:

Table: Parameter modification for algorithm settlement

Algorithm	Parameter modification
Universal Global Optimization	Increase “parallel No.”, e.g., change default value from “30” to “60”. The large the “parallel No.”, the higher the success possibility, but it needs more computation time.
Max inherit optimization	Increase the “population size” and “local step length”
Differential evolution algorithm	Start “local search”, change the option of differential evolution algorithm
Simplex method	Change “mode” option, increase “multi calculation number”
Self-Organizing migrating algorithm	Increase “number of group”, decrease “step length”
Simulated annealing	Increase “internal loop number”
Genetic algorithm	Increase “number of group”
Tabu search	
Particle swarm optimization	Increase “number of group”

1.13 Basic grammar

Every line of code ends with ‘;’, e.g.,

Parameter a, b, c, d ;

Constant $p1 = 1, p2 = 4, p3 = 5$;

Every CodeBook can be used to describe multiple problems, which are separated by key word “NewDivision”.

Regarding curve fit, the default names of independent and dependent variable are x and y respectively, in the case of two-dimension variables; the default names of independent variables are x1, x2, x3, ... and the default of dependent variable is y for three or more than three-dimension variables. For example, the following two blocks of codes have the same functionalities. User doesn’t need to re-define variables and parameters which will be recognized by Auto2Fit automatically.

Code 1	Code 2
Variables x, y; Parameters a, b, c, d; Function $y=a-b*\exp(-c*x^d)$; Data ; 0.05 0.13 0.15 0.13 0.25 0.19 0.35 0.34	Function $y=a-b*\exp(-c*x^d)$; Data ; 0.05 0.13 0.15 0.13 0.25 0.19 0.35 0.34

As for function optimization, if there is no range limit for parameters, the definition of parameter can be omitted. The following two blocks of codes have the same functionalities.

Code 1	Code 2
Parameters x, y; Minimum = True; Function exp(sin(50*x)) + sin(60*exp(y)) + sin(70*sin(x)) + sin(sin(80*y)) - sin(10*(x+y)) + (x^2+y^2)/4;	MinFunction exp(sin(50*x)) + sin(60*exp(y)) sin(70*sin(x)) + sin(sin(80*y)) - sin(10*(x+y)) + (x^2+y^2)/4;

1.14 Execute calculation

Because the initial values in Auto2Fit are usually generated randomly, once failure of calculation doesn't mean it will fail again next time, vice versa. Press shortcut "F9" to execute calculation, press "F10" to stop calculation.

Chapter 2 Application of Auto2Fit

2.1 Find optimal value of any format, any dimensional, constrained or un-constrained function

Auto2Fit can be used to find the optimal value of un-constrained function and can also be used to find the optimal value of constrained function. Constrained function can be equality or inequality.

2.1.1 Find the minimum value of the following one-dimensional function

$$f = x \cdot \sin(x) + \sin(x) \quad (2-1-1)$$

here, $x \in [-3\pi, 3\pi]$

Auto2Fit code:

```
Parameter x = [-3*pi,3*pi];  
Minimum;  
Function x*sin(x)+sin(x);
```

or simpler format:

```
Parameter x = [-3*pi,3*pi];  
MinFunction x*sin(x)+sin(x);
```

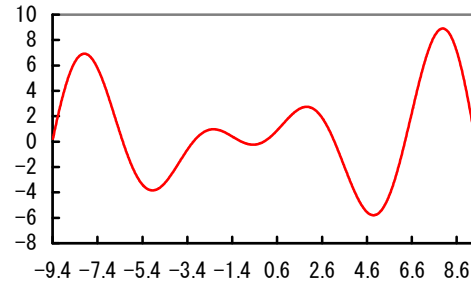


Figure 2-1-1 Graph of one dimensional function

Result: $f = -5.7976, x = 4.8808$

2.1.2 Find the minimum value of the following multi-dimensional function

$$f = \sum_{i=1}^{n-1} \left(3 \cdot (\cos(2 \cdot x_i) + \sin(2 \cdot x_{i+1})) + \sqrt{x_i^2 + x_{i+1}^2} \right) \quad (2-1-2)$$

here, $\bar{X} \in [-30, 30], n = 20$

Auto2Fit code:

```

Constant n = 20;
Parameter x(1:n) = [-30,30];
MinFunction Sum(i = 1:n-1) (3*(Cos(2*x[i]) + Sin(2*x[i+1])) + Sqrt(x[i+1]^2 + x[i]^2));
Result: f = -51.7695

```

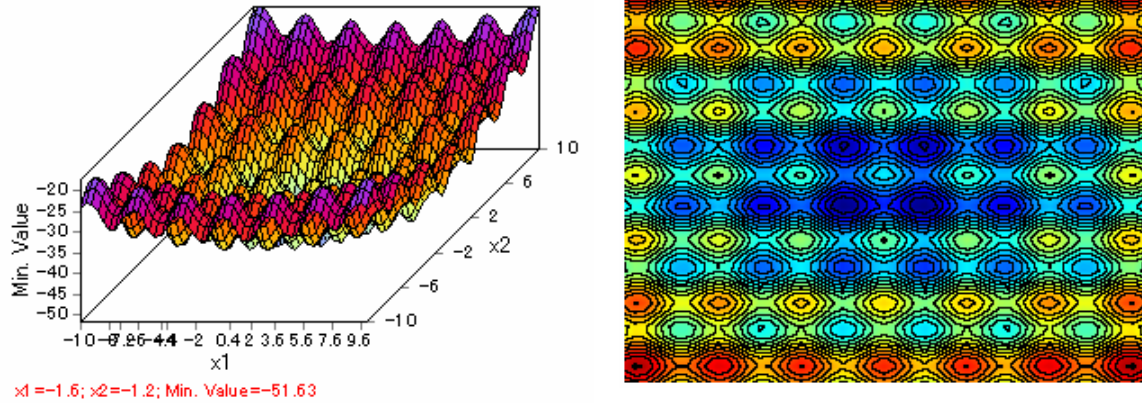


Figure 2-1-2 Three dimensional graph of multi-dimensional function optimization

2.1.3 Find the minimum value of the following implicit function z

$$z = \sin\left((z \cdot x - 0.5)^2 + 2 \cdot x \cdot y^2 - \frac{z}{10}\right) \cdot \exp\left(-\left((x - 0.5 - \exp(-y + z))^2 + y^2 - \frac{z}{5} + 3\right)\right)$$

(2-1-3)

here, $x \in [-1, 7]$, $y \in [-2, 2]$

Auto2Fit code:

```

Parameters x[-1,7], y[-2,2];
Minimum = z;
Function z = sin((z*x-0.5)^2 + x*2*y^2-z/10)*exp(-(x-0.5-exp(-y+z))^2 + y^2-z/5+3));
Result: z = 0.02335 (x = 2.898329,y = -0.8573138)

```

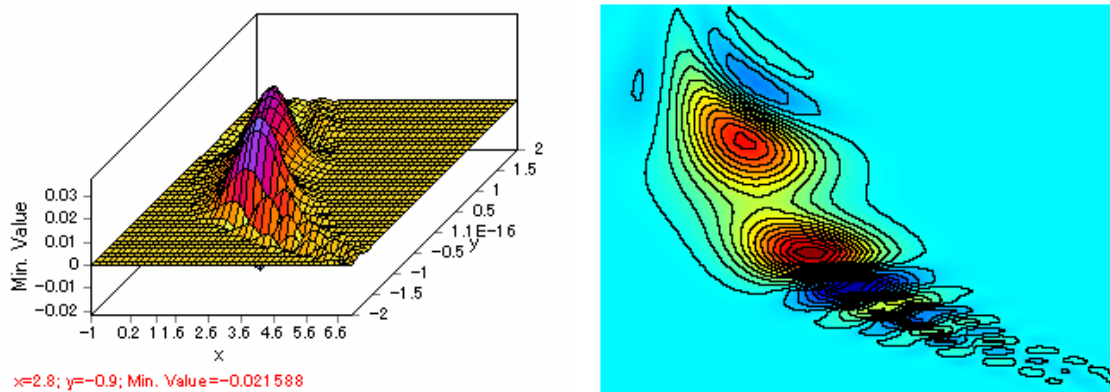


Figure 2-1-3 Three dimensional graph of implicit function optimization

2.1.4 Needle-shaped global optimized function

$$f(r) = \frac{\sin(r)}{r} + 1 \quad (2-1-4)$$

here:

$$r = \sqrt{(x-50)^2 + (y-50)^2} + e$$

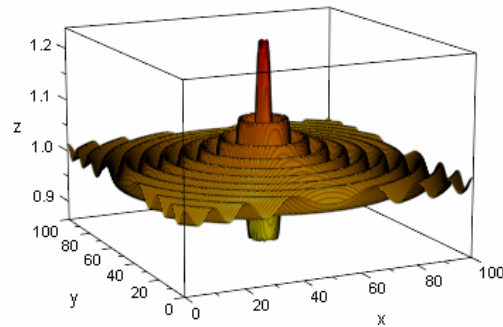


Figure 2-1-4 Three dimensional graph of needle-shaped function

The graph of this function is shown in Figure 2-1-4. The global maximum value of this function, 1.1512, can be achieved at the point of (50, 50). The second largest value is 1.12837. It is impossible to find the global optimal point using traditional optimization method while Auto2Fit can calculate it easily.

Auto2Fit code:

```
Parameter x[0,100], y[0,100];
ConstStr r = sqrt((x-50)^2+(y-50)^2)+exp(1);
MaxFunction Sin(r)/r + 1;
```

2.1.5 Linear programming problem

Auto2Fit has included Simplex algorithm that can solve linear programming problem effectively. Differed from some optimization software package, for example Lingo, the default ranges are positive, the default parameter range of Auto2Fit is free. See the following examples of multi-constrained linear programming problem:

Linear programming example 1

$$\begin{aligned} &\text{Max} \quad 2 \cdot x_1 + 3 \cdot x_2 + x_3 \\ &\text{St.} \quad \begin{cases} x_1 + 3 \cdot x_2 + x_3 \leq 15 \\ 2 \cdot x_1 + 3 \cdot x_2 - x_3 \leq 18 \\ x_1 - x_2 + x_3 \leq 3 \\ x_1, x_2, x_3 \geq 0 \end{cases} \end{aligned} \quad (2-1-5)$$

Auto2Fit code:

```
Parameter x(1:3)[0,];
MaxFunction 2*x1+3*x2+x3;
```

$$\begin{aligned}x_1+3x_2+x_3 &\leq 15; \\2x_1+3x_2-x_3 &\leq 18; \\x_1-x_2+x_3 &\leq 3;\end{aligned}$$

Result: $x_1=5$, $x_2=3$, $x_3=1$, the maximum value is 20

Linear programming example 2

Min: $0.44x_1+0.94x_2+0.88x_3+0.48x_4+4x_5+3.4x_6+2.3x_7+0.12x_8+1.6x_9+19x_{10}+25x_{11}$

$$3230x_1+2640x_2+2500x_3+1730x_4+2900x_5+2230x_6+2500x_7>2750$$

$$8.27x_1+43x_2+40x_3+15.4x_4+62x_5+50x_6+45x_7>15$$

$$8.27x_1+43x_2+40x_3+15.4x_4+62x_5+50x_6+45x_7<16$$

$$0.038x_1+0.32x_2+0.32x_3+0.14x_4+3.91x_5+4.6x_6+33.4x_8+21x_9>2.85$$

$$0.038x_1+0.32x_2+0.32x_3+0.14x_4+3.91x_5+4.6x_6+33.4x_8+21x_9<3$$

$$0.058x_1+0.15x_2+0.14x_3+0.32x_4+2.9x_5+2.15x_6+0.14x_8+18.5x_9>0.5$$

$$0.058x_1+0.15x_2+0.14x_3+0.32x_4+2.9x_5+2.15x_6+0.14x_8+18.5x_9<0.55$$

$$0.26x_1+2.45x_2+2.41x_3+0.54x_4+4.35x_5+3.28x_6+2.6x_7+99x_{11}>0.8$$

$$0.125x_1+0.48x_2+0.51x_3+0.18x_4+1.65x_5+1.31x_6+0.65x_7+99x_{10}>0.31$$

$$0.298x_1+1.08x_2+1.4x_3+0.58x_4+2.21x_5+1.74x_6+0.83x_7+99x_{10}>0.58$$

$$0.298x_1+1.08x_2+1.4x_3+0.58x_4+2.21x_5+1.74x_6+0.83x_7+99x_{10}<0.63$$

$$0.077x_1+0.6x_2+0.6x_3+0.27x_4+0.8x_5+0.64x_6>0.19$$

$$x_1>0.5, x_1<0.66$$

$$x_2+x_3>0.1, x_2+x_3<0.22$$

$$x_4>0.04, x_4<0.2$$

$$x_5+x_6>0.03, x_5+x_6<0.07$$

$$0<x_7<0.035$$

$$x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9+x_{10}+x_{11}=1$$

Auto2Fit code:

Parameter $x_1[0.5,0.66]$, $x_4[0.04,0.2]$, $x_7[0,0.035]$;

MinFunction $0.44x_1+0.94x_2+0.88x_3+0.48x_4+4x_5+3.4x_6+2.3x_7+0.12x_8+1.6x_9+19x_{10}+25x_{11}$;

$$3230x_1+2640x_2+2500x_3+1730x_4+2900x_5+2230x_6+2500x_7>2750;$$

$$8.27x_1+43x_2+40x_3+15.4x_4+62x_5+50x_6+45x_7>15;$$

$$8.27x_1+43x_2+40x_3+15.4x_4+62x_5+50x_6+45x_7<16;$$

$$0.038x_1+0.32x_2+0.32x_3+0.14x_4+3.91x_5+4.6x_6+33.4x_8+21x_9>2.85;$$

$$0.038x_1+0.32x_2+0.32x_3+0.14x_4+3.91x_5+4.6x_6+33.4x_8+21x_9<3;$$

$$0.058x_1+0.15x_2+0.14x_3+0.32x_4+2.9x_5+2.15x_6+0.14x_8+18.5x_9>0.5;$$

$$0.058x_1+0.15x_2+0.14x_3+0.32x_4+2.9x_5+2.15x_6+0.14x_8+18.5x_9<0.55;$$

$$0.26x_1+2.45x_2+2.41x_3+0.54x_4+4.35x_5+3.28x_6+2.6x_7+99x_{11}>0.8;$$

$$0.125x_1+0.48x_2+0.51x_3+0.18x_4+1.65x_5+1.31x_6+0.65x_7+99x_{10}>0.31;$$

$$0.298x_1+1.08x_2+1.4x_3+0.58x_4+2.21x_5+1.74x_6+0.83x_7+99x_{10}>0.58;$$

$$0.298x_1+1.08x_2+1.4x_3+0.58x_4+2.21x_5+1.74x_6+0.83x_7+99x_{10}<0.63;$$

$$0.077x_1+0.6x_2+0.6x_3+0.27x_4+0.8x_5+0.64x_6>0.19;$$

$$x_2+x_3>0.1;$$

$$x_2+x_3<0.22;$$

$$x_5+x_6>0.03;$$

$$x_5+x_6<0.07;$$

$$x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9+x_{10}+x_{11}=1;$$

Result:

Number of iteration: 13
 Computation time (Hour:Minute:Second:Millisecond): 00:00:00:15
 Algorithm: Simplex linear programming algorithm
 The minimum value of this linear programming is: 0.651708124

The optimal parameter solution is:

x1: 0.66
 x4: 0.04
 x7: 0
 x2: -0.028913011
 x3: 0.212176502
 x5: 0.041774768
 x6: -0.011774768
 x8: 0.068574559
 x9: 0.017195715
 x10: 0.000731968
 x11: 0.000234268

2.1.6 Nonlinear programming problem

Nonlinear mixed integer programming problem example-1

$$\begin{aligned} \text{Max} \quad & \frac{(\sin(2 \cdot \pi \cdot x_1))^3 \cdot \sin(2 \cdot \pi \cdot x_2)}{x_1^3 \cdot (x_1 + x_2)} \\ \text{St.} \quad & \begin{cases} x_1^2 - x_2 + 1 \leq 0 \\ 1 - x_1 + (x_2 - 4)^2 \leq 0 \\ 0 \leq x_1 \leq 10 \\ 0 \leq x_2 \leq 10 \end{cases} \end{aligned} \quad (2-1-6)$$

Auto2Fit code:

```
Parameters x1[0,10], x2[0,10];
MaxFunction (sin(2*pi*x1))^3*sin(2*pi*x2)/(x1^3*(x1+x2));
x1^2-x2+1 <= 0;
1-x1+(x2-4)^2 <= 0;
```

Result: $x_1 = 1.22797$, $x_2 = 4.24537$, Maximum value: 0.095825

Nonlinear mixed integer programming problem example-2

$$\begin{aligned} \text{Min} \quad & 1.5 \cdot (x_1 - \sin(x_1 - x_2))^2 + 0.5 \cdot x_2^2 + x_3^2 - x_1 \cdot x_2 - 2 \cdot x_1 + x_2 \cdot x_3 \\ \text{St.} \quad & \begin{cases} -20 < x_1 < 20 \\ -20 < x_2 < 20 \\ -10 < x_3 < 10 \end{cases} \quad x_1, x_2 \text{ are real, } x_3 \text{ is integer} \end{aligned} \quad (2-1-7)$$

Auto2Fit code:

```
Parameters x1[-20,20],x2[-20,20],x3[-10,10,0];
MinFunction 1.5*(x1-sin(x1-x2))^2+0.5*x2^2+x3^2-x1*x2-2*x1+x2*x3;
```

Result: when $x_1 = 4.49712$, $x_2 = 9.147501$, $x_3 = -4$, the minimum value is -10.51832

Nonlinear integer programming problem example -3

$$\text{Min: } -2 \cdot x_1 - x_2 - 4 \cdot x_3 - 3 \cdot x_4 - x_5 \quad (2-1-8)$$

$$\text{St. } \begin{cases} 2 \cdot x_2 + x_3 + 4 \cdot x_4 + 2 \cdot x_5 < 54 \\ 3 \cdot x_1 + 4 \cdot x_2 + 5 \cdot x_3 - x_4 - x_5 < 62 \\ x_1, x_2 \in [0, 100]; x_3 \in [3, 100], x_4 \in [0, 100], x_5 \in [2, 100] \end{cases}$$

here, from x_1 to x_5 are all integer.

Auto2Fit code:

Parameter x(1:2)[0,100,0], x3[3,100,0], x4[0,100,0], x5[2,100,0];

MinFunction -2*x1-x2-4*x3-3*x4-x5;

2*x2+x3+4*x4+2*x5<54;

3*x1+4*x2+5*x3-x4-x5<62;

The results have two groups: $x_1, x_2, x_3, x_4, x_5 = (15, 0, 6, 11, 2)$ or $x_1, x_2, x_3, x_4, x_5 = (19, 0, 4, 10, 5)$, the minimum value is -89

2.1.7 Combination optimization problem

Auto2Fit can also be used to solve combination optimization problem. The self-developed max inherit algorithm performs better than genetic algorithm, simulated annealing and tabu search when solve this category of problem.

TSP problem

TSP problem is famous combination optimization problem: there are N cities, a salesman starts from one city, has to travel every city once and finally return to the start city. The question is to find the short distance route for his traveling. The following example uses 15 cities.

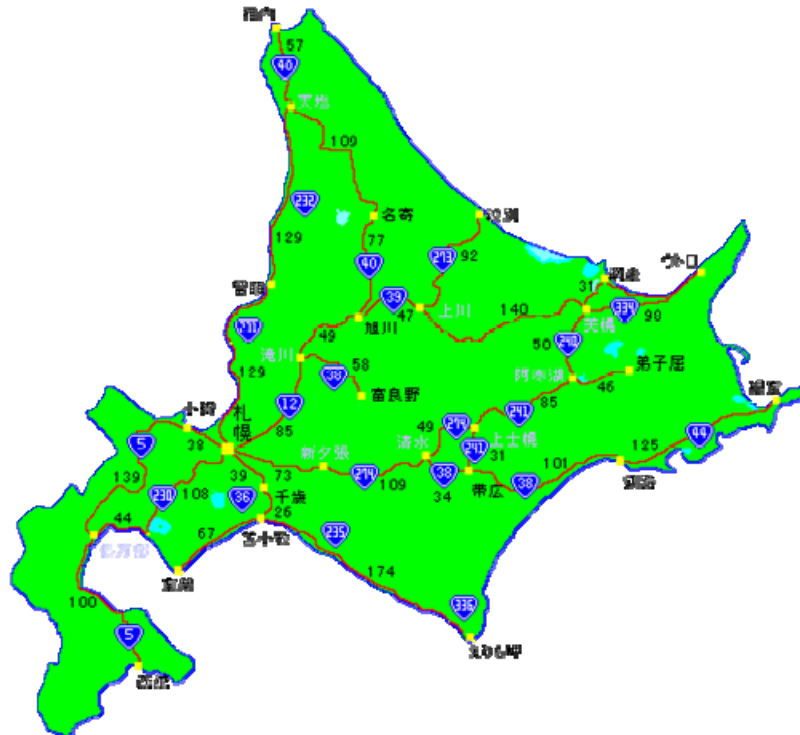


Figure 2-1-5 Map of 15 cities

Table of distances between 15 cities:

	札幌	函館	旭川	弟子屈	苫小牧	千歳	釧路	稚内	帯広	ウトロ	小樽	富良野	根室	網走	留萌
札幌	0	261	134	364	66	39	328	315	164	418	38	143	453	349	129
函館	261	0	395	625	255	251	591	587	425	679	244	420	715	615	389
旭川	134	395	0	242	200	173	289	243	177	281	157	57	386	215	78
弟子屈	364	625	242	0	356	360	72	404	165	107	402	285	116	77	320
苫小牧	66	255	200	356	0	25	304	381	191	463	104	168	429	390	201
千歳	39	251	173	360	25	0	308	365	195	467	77	166	432	394	168
釧路	328	591	289	72	304	308	0	479	114	182	366	233	125	152	367
稚内	315	587	243	404	381	365	479	0	420	410	330	300	520	327	182
帯広	164	425	177	165	191	195	114	420	0	272	202	120	239	199	255
ウトロ	418	679	281	107	463	467	182	410	272	0	438	338	165	83	359
小樽	38	244	157	402	104	77	366	330	202	438	0	214	491	387	148
富良野	143	420	57	285	168	166	233	300	120	338	214	0	357	272	131
根室	453	715	386	116	429	432	125	520	239	165	491	357	0	193	464
網走	349	615	215	77	390	394	152	327	199	83	387	272	193	0	293
留萌	129	389	78	320	201	168	367	182	255	359	148	131	464	293	0

Auto2Fit code:

```

Constant n = 15;
Constant Distance(0:n-1, 0:n-1) = [0,261,134,364,66,39,328,315,164,418,38,143,453,349,129,
261,0,395,625,255,251,591,587,425,679,244,420,715,615,389,
134,395,0,242,200,173,289,243,177,281,157,57,386,215,78,
364,625,242,0,356,360,72,404,165,107,402,285,116,77,320,
66,255,200,356,0,25,304,381,191,463,104,168,429,390,201,
39,251,173,360,25,0,308,365,195,467,77,166,432,394,168,
328,591,289,72,304,308,0,479,114,182,366,233,125,152,367,
315,587,243,404,381,365,479,0,420,410,330,300,520,327,182,
164,425,177,165,191,195,114,420,0,272,202,120,239,199,255,
418,679,281,107,463,467,182,410,272,0,438,338,165,83,359,
38,244,157,402,104,77,366,330,202,438,0,214,491,387,148,
143,420,57,285,168,166,233,300,120,338,214,0,357,272,131,
453,715,386,116,429,432,125,520,239,165,491,357,0,193,464,
349,615,215,77,390,394,152,327,199,83,387,272,193,0,293,
129,389,78,320,201,168,367,182,255,359,148,131,464,293,0];

Parameters Cities(0:n-1)[0,n-1];
Exclusive = True;
Minimum = True;
StartProgram;
Var TemSum : Double;
  i : integer;
Begin
  TemSum := 0;
  for i := 0 to n-2 do
    TemSum := TemSum + Distance[Cities[i], Cities[i+1]];
  FunctionResult := TemSum + Distance[Cities[n-1], Cities[0]];
End;
EndProgram;

```

Result: the shortest distance is 2076 KM. the traveling route is:

札幌 ⇒ 旭川 ⇒ 富良野 ⇒ 帯広 ⇒ 釧路 ⇒ 根室 ⇒ 弟子屈 ⇒ ウトロ ⇒ 網走 ⇒ 稚内 ⇒ 留萌 ⇒ 小樽 ⇒ 函館 ⇒ 苫小牧 ⇒ 千歳 ⇒ 札幌

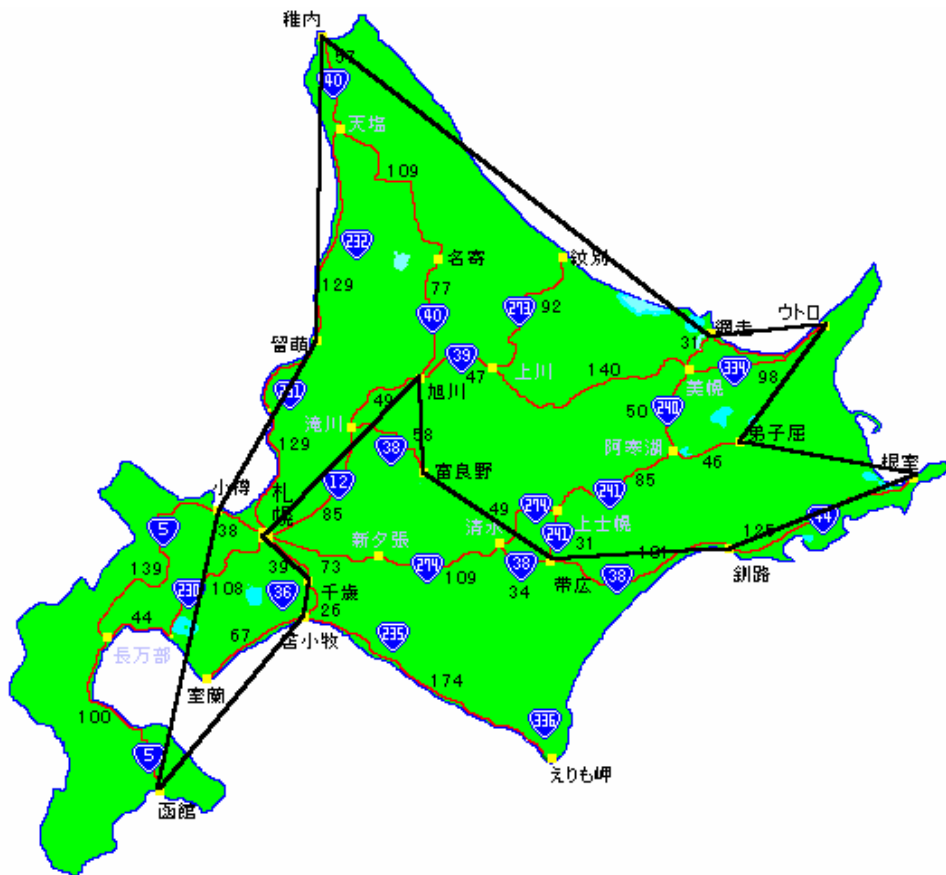


图 2-1-6 The shortest path of traveling route between 15 cities

The shortest path problem

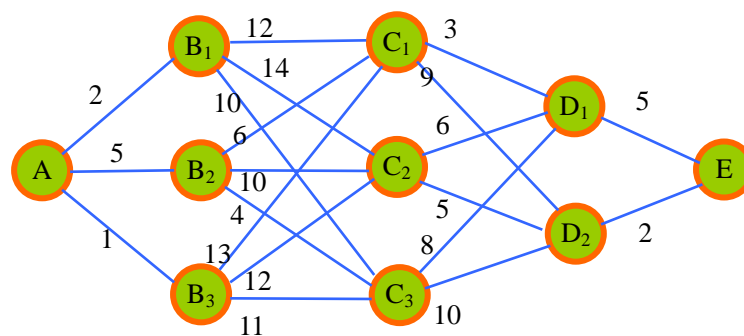


Figure 2-1-7 Illustration of the shortest path

Figure 2-1-7 shows the distances from starting point A to ending point B. Find the shortest path between A and E.

This problem is a typical combination optimization problem. Differed from traditional linear programming algorithm, Auto2Fit uses global optimization to find the solution.

Table of distances between cities (Value of 1000 represents no connection between cities):

	No.	0	1	2	3	4	5	6	7	8	9
No.		A	B1	B2	B3	C1	C2	C3	D1	D2	E
0	A	0	2	5	1	1000	1000	1000	1000	1000	1000
1	B1	2	0	1000	1000	12	14	10	1000	1000	1000
2	B2	5	1000	0	1000	6	10	4	1000	1000	1000
3	B3	1	1000	1000	0	13	12	11	1000	1000	1000
4	C1	1000	12	6	13	0	1000	1000	3	9	1000
5	C2	1000	14	10	12	1000	0	1000	6	5	1000
6	C3	1000	10	4	11	1000	1000	0	8	10	1000
7	D1	1000	1000	1000	1000	3	6	8	0	1000	5
8	D2	1000	1000	1000	1000	9	5	10	1000	0	2
9	E	1000	1000	1000	1000	1000	1000	1000	5	2	0

Auto2Fit code

```

Constant n = 8;
Parameters Cities(1:n)[1,n,0];
Constant Dis(0:n+1,0:n+1) = [0,2,5,1,1000,1000,1000,1000,1000,1000,
                                2,0,1000,1000,12,14,10,1000,1000,1000,
                                5,1000,0,1000,6,10,4,1000,1000,1000,
                                1,1000,1000,0,13,12,11,1000,1000,1000,
                                1000,12,6,13,0,1000,1000,3,9,1000,
                                1000,14,10,12,1000,0,1000,6,5,1000,
                                1000,10,4,11,1000,1000,0,8,10,1000,
                                1000,1000,1000,1000,3,6,8,0,1000,5,
                                1000,1000,1000,1000,9,5,10,1000,0,2,
                                1000,1000,1000,1000,1000,1000,1000,5,2,0];

Minimum = True;
StartProgram;
var i : integer;
    temD : Double;
begin
    temD := 0;
    for i := 1 to n - 1 do
        temD := temD + Dis[Cities[i],Cities[i+1]];
    FunctionResult := temD + Dis[0,Cities[1]] + Dis[Cities[n],n+1];
end;
EndProgram;

```

Result: the length of the shorest path is 19, the route is: 0 -> 2 -> 4 -> 7 -> 9, i.e., A -> B2 -> C1 -> D1 -> E

2.2 Nonlinear curve fit

The nonlinear curve fit of Auto2Fit is powerful than any other similar software package available today, such as SPSS, SAS, Matlab, Origin, Systat, DataFit, etc. The greatest feature is that it is no long to need the end-user to provide or guess the initial start-values for each parameter, but randomly generated by machine, the probability for finding the correct solution is higher than any others. American National Institute of Standards and Technology (NIST) have a set of test dataset which include 27 nonlinear curve fit questions. Almost all of the data analysis software venders make those test data as their performance criteria. Auto2Fit is the current only software package that doesn't need the initial values provided by NIST compared with other software package and it can get all of the optimal solution with any random values (if use the initial values provided by NIST, it is easier to find the optimal solution using Auto2Fit). In practice, it is really difficult to choose appropriate initial values especially when the number of parameters is large. From this point of view, the practical application ability of Auto2Fit is in a leading place in the world.

Nonlinear regression test result of NIST test dataset

No.	Test name	Difficulty	Number of parameter	Initial value	Algorithm adopted by Auto2Fit	Success rate (%)
1	Misra1a	Lower	2	Randomly provided by Auto2Fit	Universal Global Optimization algorithm	100
2	Chwirut2		3			100
3	Chwirut1		3			100
4	Lanczos3		6			100
5	Gauss1		8			> 90
6	Gauss2		8			> 90
7	DanWood		2			100
8	Misra1b		2			100
9	Kirby2	Middle	5			100
10	Hahn1		7			100
11	Nelson		3			100
12	MGH17		5			100
13	Lanczos1		6			100
14	Lanczos2		6			100
15	Gauss3		8			> 90
16	Misra1c		2			100
17	Misra1d		2			100
18	Roszman1		4			100
19	ENSO		9			100
20	MGH09	higher	4			100
21	Thurber		7			100
22	BoxBod		2			100
23	Rat42		3			100

24	MGH10		3			100
25	Eckerle4		3			100
26	Rat43		4			100
27	Bennett5		3			>90

Note: NIST webpage:http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml

The regression function in Auto2Fit is all user-defined function. "Function" and "Data/RowData/DataFile" are two essential key words. Other optional key words include "Variable", "Parameter" and "QuickReg", etc. As for two-variable curve fitting, the default name of independent variable is x , while y for dependent. There are two default settings for three-variable scenarios. One is the names of independent variables are x_1 and x_2 and dependent variable is y , the other is the names of independent variables are x and y and dependent variable is z . For multi-variable (>3) scenario, the default names of independent variables are x_1, x_2, x_3, \dots , and the name of dependent variable is y . The outputs of the following two blocks of codes are the same. There is no need to use "Variable" and "Parameter" to define variable and parameter, which will be recognized by Auto2Fit automatically.

Comparison 1 of curve fit codes

Code 1	Code 2
Variables x, y;	Function y = a-b*exp(-c*x^d);
Parameters a, b, c, d;	Data;
Function y = a-b*exp(-c*x^d);	0.05 0.13
Data;	0.15 0.13
0.05 0.13	0.25 0.19
0.15 0.13	0.35 0.34
0.25 0.19	
0.35 0.34	

When the volume of data is large, it can be considered to represent data in row in order to save the space of CodeBook, i.e., use key word "RowData" to replace "Data". The format of data need to be modified accordingly, every row of data ends with ";".

Comparison 2 of curve fit codes

Code 1	Code 2
Function y = a-b*exp(-c*x^d);	Function y = a-b*exp(-c*x^d);
Data;	RowData;
0.05 0.13	0.05,0.15,0.25,0.35;
0.15 0.13	0.13,0.13,0.19,0.34;
0.25 0.19	
0.35 0.34	

Data can be saved as file and invoked by key word "DataFile". The formats of file include standard ASCII text format and Excel file format. The invoke method is as follows if Excel data (see figure 2-2-1) is saved to "c:\test1.xls". Note that the key word "Variable" cannot be omitted when "DataFile" is used.

```
Variable x,y;
Function y = b1*(x^2+x*b2)/(x^2+x*b3+b4);
DataFile "C:\test1.xls[Sheet1[B4:C14]]";
```

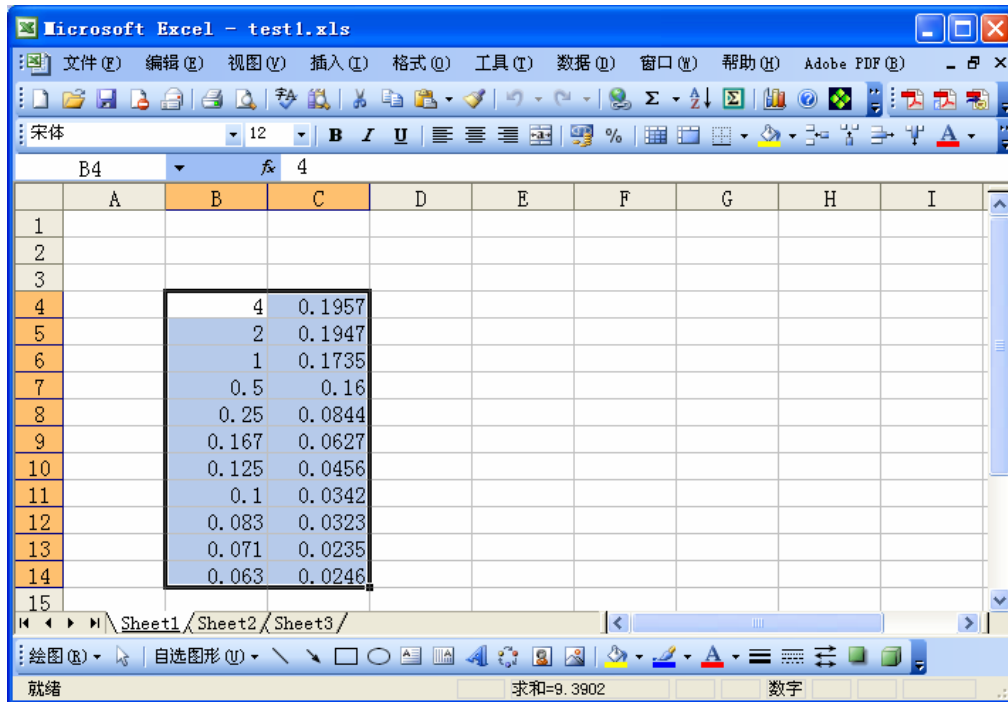


Figure 2-2-1 Read data from Excel

2.2.1 Curve fit with SharedMode function

SharedMode means that different fit equation can have one or more than one shared parameters. The implementation of SharedMode is by using the key word "SharedModel".

Data for curve fit

x	y ₁	y ₂	y ₃	y ₄
0.	0.0	0.0	0.0	0.0
50.	17.03	7.23	3.57	0.0
100.	22.16	13.34	8.14	0.1
150.	30.64	17.76	13.55	4.81
200.	33.57	25.09	10.83	5.23
400.	50.40	38.05	21.93	14.58
600.	58.36	45.01	25.50	13.46
800.	62.68	50.52	31.82	15.87
1000.	63.54	52.57	38.13	20.90
10000.	81.0	74.3	70.9	61.7

Four fit equations

$$\left\{ \begin{array}{l} y_1 = \frac{v \cdot x}{k_1 + x} + p_1 \cdot x + c_1 \\ y_2 = \frac{v \cdot x}{k_2 + x} + p_2 \cdot x + c_2 \\ y_3 = \frac{v \cdot x}{k_3 + x} + p_3 \cdot x + c_3 \\ y_4 = \frac{v \cdot x}{k_4 + x} + p_4 \cdot x + c_3 \end{array} \right. \quad (2-2-1)$$

Here, $k_1 = m_1 \cdot (1 + \frac{1}{m_2})$, $k_1 = m_1 \cdot (1 + \frac{3}{m_2})$, $k_1 = m_1 \cdot (1 + \frac{10}{m_2})$, $k_1 = m_1 \cdot (1 + \frac{30}{m_2})$

One independent variable, four dependent variable, eleven parameters: $m_1, m_2, v, p_1, p_2, p_3, p_4, c_1, c_2, c_3, c_4$, the first three parameters m_1, m_2, v are shared parameters.

Auto2Fit code

```
ConstStr k1=m1*(1+1/m2), k2=m1*(1+3/m2), k3=m1*(1+10/m2), k4=m1*(1+30/m2);
SharedModel;
Variable x,y(4); //y1, y2, y3, y4;
Function y1 = v*x/(k1+x)+p1*x+c1;
        y2 = v*x/(k2+x)+p2*x+c2;
        y3 = v*x/(k3+x)+p3*x+c3;
        y4 = v*x/(k4+x)+p4*x+c4;
Data;
//x, y1, y2, y3, y4
0.  0.0  0.0  0.0  0.0
50. 17.0261  7.227563  3.574113  0.0
100. 22.16059  13.34309  8.142564  0.1
150. 30.64281  17.76278  13.55202  4.805006
200. 33.57431  25.08648  10.82765  5.232621
400. 50.40222  38.048  21.93352  14.57796
600. 58.35754  45.00776  25.49771  13.45863
800. 62.68015  50.51803  31.82192  15.86972
1000.63.53971  52.56842  38.12983  20.90453
10000. 81.0 74.3 70.9 61.7
```

Result:

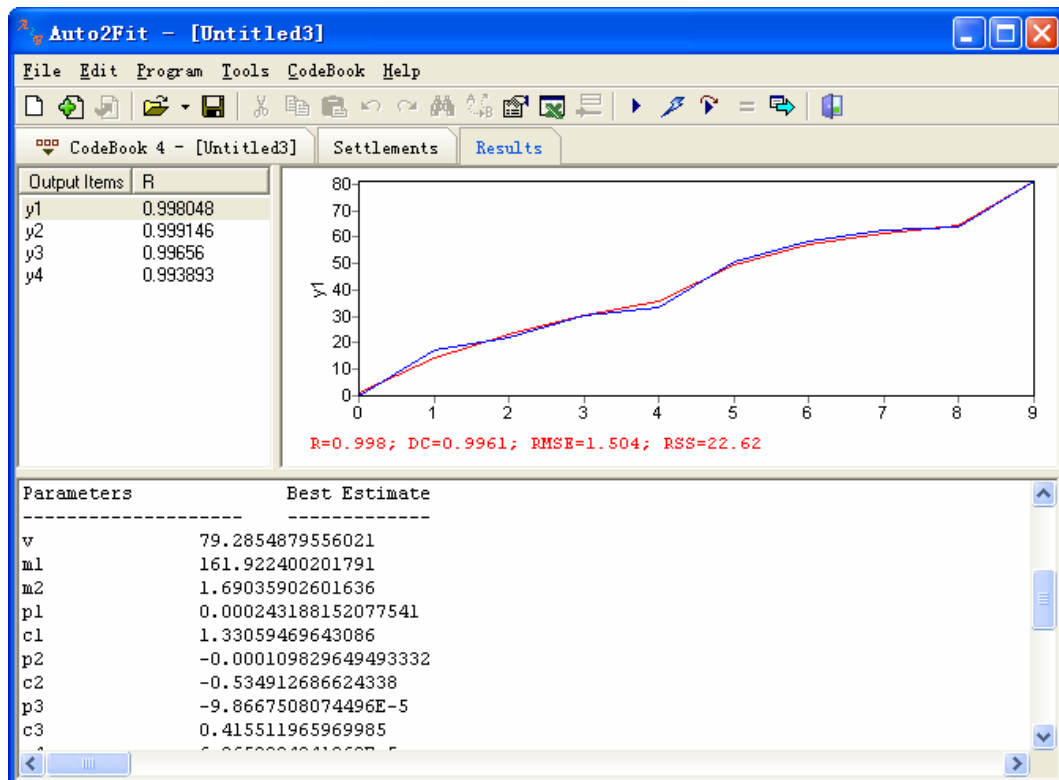


Figure 2-2-2 The results of nonlinear regression in SharedModel

2.2.2 Curve fit with missing variables values

Example 1: the curve fit functions are as follows, x , y data are known

$$\begin{aligned} x &= r \cdot (z - \sin(z + b)) \cdot \cos(b) + r \cdot (1 - \cos(z \cdot b)) \cdot \sin(b) \\ y &= r \cdot (z - \cos(z - b)) \cdot \cos(b) - r \cdot (z - \sin(z \cdot b)) \cdot \sin(b) \end{aligned} \quad (2-2-2)$$

Data for curve fit

x	35	41	46	52	58	63	70	75	80	85	90	95	35
y	32	33	33.5	33	32	30	28	25	21	17.5	13	9	32

Now it is needed to do the curve fit for the above parametric equation, using x and y . This parameter equation is cycloid equation, here r is the cycloid radius of the spheronization, z is angle and its value ranged from 0 to 2π , b is the initial angle of rotation curve, how to find parameter r and b ?

Both two equations in 2-2-2 have variable a . If the values of variable z which corresponds to x and y is known, then the problem can be solved easier using key word “SharedModel”. But here z is unknown and it is impossible to derive a normal two dimension curve fit format like $y = f(x)$ without z . The key word “ParVariable” in Auto2Fit can be used to define this category of unknown variable. The code is as follows:

```
Variable x,y;
ParVariable z[0,2*pi];
SharedModel;
Function x=r*(z-sin(z+b))*cos(b)+r*(1-cos(z*b))*sin(b);
        y=r*(1-cos(z-b))*cos(b)-r*(z-sin(z*b))*sin(b);
RowData;
35,41,46,52,58,63,70,75,80,85,90,95;
32,33,33.5,33,32,30,28,25,21,17.5,13,9;
```

Result:

Number of iteration: 11
Computation time (Hour:Minute:Second:Millisecond): 00:00:05:547
Optimization algorithm: Universal Global Optimization (UGO1)
Reason of computation stops: User stops
Mean square deviation (RMSE): 0.486859909332516
Residual sum of squares (RSS): 5.68878171156637
Correlation coefficient (R): 0.999956231439338
The square of correlation coefficient (R^2): 0.999912464794363
Determination coefficient (DC): 0.999912464794363
F-Statistic: -426.618375101261

Parameter	Optimal estimation
r	43.7937063646739
b	0.771359156985534
z0	1.15327638954225
z1	1.37452196344552
z2	1.55093358150438
z3	1.73198956917058

z4	1.88497841264526
z5	2.00312651744943
z6	2.14185504147438
z7	2.24269773820352
z8	2.34439505606345
z9	2.43538721773244
z10	2.52901659199243
z11	2.61369705866839

Using Auto2Fit, it is very easy to find the solution. Not only parameter r and b are found, but z values corresponding to x and y is found.

Example 2: the parameter equations are as follows:

$$\begin{aligned} x &= r \cdot (\cos(t) + (t - b) \cdot \sin(t)) + x_0 \\ y &= r \cdot (\sin(t) - (t - b) \cdot \cos(t)) + y_0 \end{aligned} \quad (2-2-3)$$

Data for curve fit

x	15.5910	24.6601	33.3732	49.3445	62.7992	68.3962	73.1584	79.9955	83.0531
y	86.2142	83.7114	80.2618	70.7216	58.0891	50.8058	42.9946	26.1718	8.4225

Here, variable t is unknown, parameter x_0 , y_0 , r and b are need to be found.

The method of solving is like example 1, use two key words “SharedModel” and “ParVariable”, the code is as follows:

```
Algorithm = UGO1[100];
Parameter R ,B,x0,y0;
ParVariable t;
Variable x, y;
SharedModel;
Function x = R*(cos(t)+(t-B)*sin(t))+x0;
        y = R*(sin(t)-(t-b)*cos(t))+y0;
Data;
15.5910    86.2142
24.6601    83.7114
33.3732    80.2618
49.3445    70.7216
62.7992    58.0891
68.3962    50.8058
73.1584    42.9946
79.9955    26.1718
83.0531    8.4225
```

It is quite difficulty to find the correct solution to this problem. The above Auto2Fit code can find the correct solution with a possibility of 50%.

Results as follows:

```
Number of iteration: 21
Computation time (Hour:Minute:Second:Millisecond): 00:00:57:32
Optimization algorithm: Universal Global Optimization (UGO1)
Reason of computation stops: meet the criteria of convergence
Mean square deviation (RMSE): 1.34537356121028E-5
Residual sum of squares (RSS): 3.25805403456652E-9
Correlation coefficient (R): 0.999999999999975
The square of correlation coefficient (R^2): 0.999999999999951
Determination coefficient (DC): 0.999999999999951
```

F-Statistic: -2008250717607.4

Parameter	Optimal estimation
r	-3.19952275016598
b	-27.5800928107093
x0	-0.000329656381182569
y0	0.000374670776059415
t0	-0.215438422244315
t1	-0.32315759519847
t2	-0.430878058302956
t3	-0.646316755114794
t4	-0.861756728212156
t5	-0.969475917540198
t6	-1.07719566112363
t7	-1.29263476604028
t8	-1.50807217873645

2.2.3 Curve fit with Batch processing function

The value of x, y can be found in the following table, try to conduct curve fit with B = 1,2,3,4 respectively.

Data for curve fit

No.	x	y			
		B = 1	B = 2	B = 3	B = 4
1	-157	20	25	30	35
2	-142	26	34	37.5	45
3	-112	36	45	50	54
4	-97	30	35	40	47.5
5	-82	35	47.5	55	59
6	-52	49	60	71	81
7	-37	45	55	65	76
8	-22	54	70	85	94
9	8	59	83.5	95	103.5
10	23	55	71	84.5	95.5
11	38	59	75	91	100
12	68	54	73.5	85	92.5
13	83	50	65	79	87.5
14	98	48	58	70	85
15	128	32.5	42	54	58

Curve fit equation:

$$y = y_0 + \frac{A}{W \cdot \sqrt{\frac{\pi}{2}}} \cdot \exp\left(-\frac{(2 \cdot (x - x_0)^2)}{W^2}\right) + B \quad (2-2-4)$$

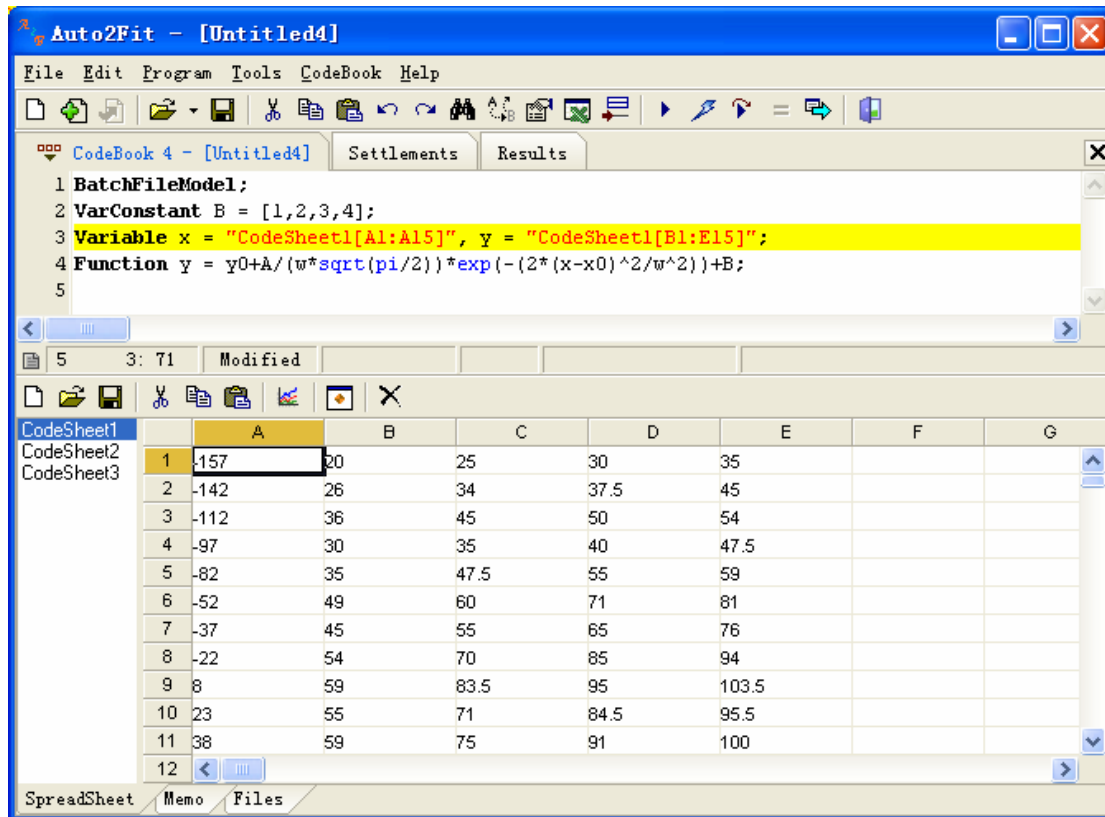


Figure 2-2-3 Curve fit with Batch processing function

Auto2Fit code

```

BatchFileModel;
VarConstant B = [1,2,3,4];
Variable x = "CodeSheet1[A1:A15]", y = "CodeSheet1[B1:E15]";
Function y = y0+A/(w*sqrt(pi/2))*exp(-(2*(x-x0)^2/w^2))+B;

```

Three points needed to be paid attention for the above code:

- Key word "BatchFileModel": find the relationships of different y and x respectively
- Key word "VarConstant": Define value of B when different curve fit in use
- When use "Variable" define variable, read data from codebook spreadsheets directly

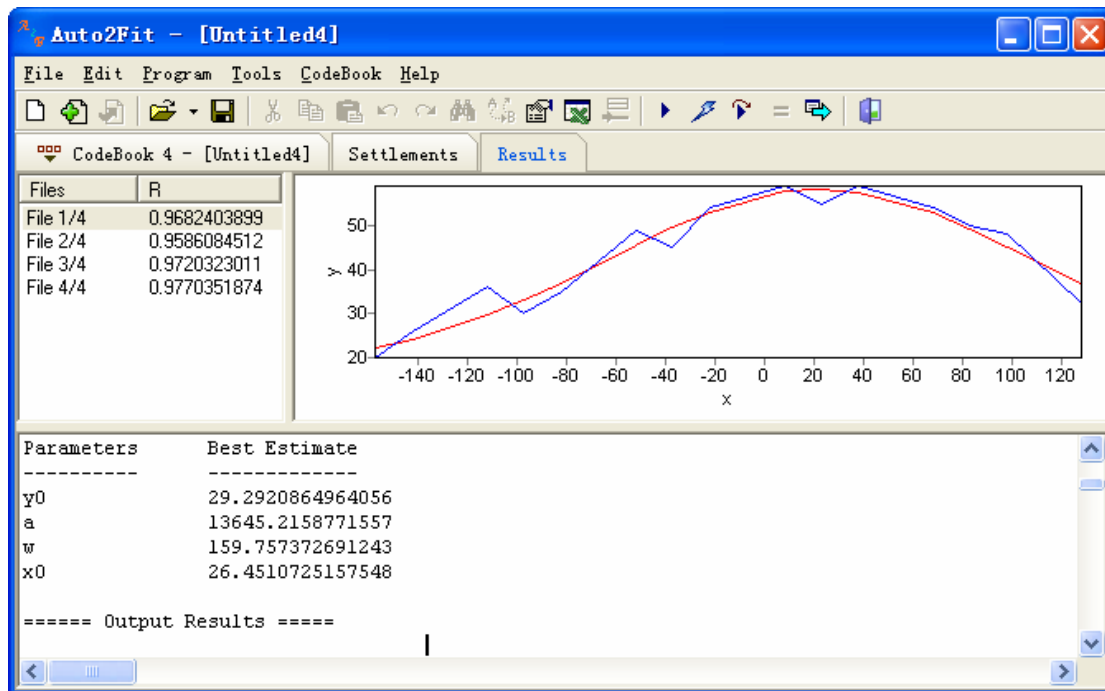


Figure 2-2-4 Result of curve fit with batch processing function

2.2.4 Weighted curve fit

Weighted curve fit can use key word “WeightedReg” to define settings:

Value of WeightedReg	Objective function	Explanation
1	$\sum_{i=1}^n \left(\frac{(y_i - y'_i)^2}{y_i^2} \right)$	y: actual dependent variable y': calculated dependent variable n: length of data for curve fit StdDev: Standard deviation
2	$\sum_{i=1}^n \left(\frac{(y_i - y'_i)^2}{StdDev} \right)$	
3	$\sum_{i=1}^n \left(\frac{(y_i - y'_i)^2}{y_i} \right)$	

Example: Weighted curve fit

Data:

x	10,20,30,40,60,90,120,180,210,240,300,360
Y	0.1,0.55,1.2,2,1.95,1.85,1.6,0.86,0.78,0.6,0.21,0.18

Curve fit equation:

$$y = \frac{a_1 \cdot \left(1 + 4 \cdot a_2^2 \cdot \frac{x}{a_3}\right)}{\left(1 - \left(\frac{x}{a_4}\right)^2\right)^2 + 4 \cdot a_5^2 \cdot \left(\frac{x}{a_2}\right)^2} \quad (2-2-5)$$

Auto2Fit code:

```
Parameters a(1:5);
WeightedReg = 1; //1: w = y^2; 2: w = StdDev^2; 3: w = y;
Variable x, y;
Function Y = a1*(1+4*a2^2*(x/a3)^2)/((1-(x/a4)^2)^2+4*a5^2*(x/a2)^2);
Data;
10  0.1
20  0.55
30  1.2
40  2
60  1.95
90  1.85
120 1.6
180 0.86
210 0.78
240 0.6
300 0.21
360 0.18
```

2.2.5 Constrained curve fit

Example: Constrained nonlinear regression

Curve fit equation and data are as follows:

Equation

$$y = \frac{a + b \cdot x}{1 + c \cdot x + d \cdot x^2} \quad (2-2-6)$$

Data:

X	1	2	8	12	17	21	24
Y	1	2	3	6	6	4	4

Different from general curve fit problem, the curve got in this example must go through point 1 and 3, i.e., $(x_1, y_1)=(1,1)$, $(x_2, y_2)=(8,3)$; the constrain condition is:

$$y_1 = \frac{a + b \cdot x_1}{1 + c \cdot x_1 + d \cdot x_1^2} \quad \text{and} \quad y_2 = \frac{a + b \cdot x_2}{1 + c \cdot x_2 + d \cdot x_2^2} \quad (2-2-7)$$

Auto2Fit code:

```
Constant x1=1, y1=1, x2=8, y2=3;
DataSet;
x, y=
1  1
2  2
8  3
12 6
17 6
21 4
24 4
```

```

EndDataSet;
MinFunction Sum(i=1:7)((y[i]-(a+b*x[i])/(1+c*x[i]+d*x[i]^2))^2);
y1=(a+b*x1)/(1+c*x1+d*x1^2);
y2=(a+b*x2)/(1+c*x2+d*x2^2);

```

Result: a = 0.861917, b = 0.035839, c = -0.105828, d = 0.0035860

If there is no constrain condition, the solution is: a = 1.011742, b = 0.056511, c = -0.103843, d = 0.003781868

Example: Constrained nonlinear regression

Constraints can be expressed in curve fit mode directly, or in standard function optimization with constrained conditions.

Curve fit equation:

$$y = \frac{p_3 + p_2 \cdot x^{p_4}}{p_5 + p_1 \cdot x^{p_4}} \quad (2-2-8)$$

Constrain condition:

$$\begin{cases} p_2 = 3 \cdot p_3 + \sum_{i=1}^3 p_i \\ 20.3 \geq p_1 + p_2 \geq 20 \end{cases}$$

Data for curve fit

x	0.010, 0.020, 0.040, 0.060, 0.080, 0.100, 0.120, 0.140, 0.160, 0.180, 0.200, 0.220, 0.240, 0.260, 0.280, 0.300, 0.320, 0.340, 0.360, 0.380, 0.400
y	3.936, 4.117, 4.775, 5.553, 6.268, 6.935, 7.480, 8.188, 8.361, 8.533, 8.771, 9.120, 9.024, 9.288, 9.462, 9.379, 9.685, 9.482, 9.545, 9.604, 9.546

Auto2Fit code

```

Algorithm = UGO[100];
Variable x, y;
Function y = (p3+p2*p1*x^p4)/(p5+p1*x^p4);
p2=3*p3+sum(i=1:3)(p[i]);
20.3 >=p1+p2>=20;

Data;
0.0103.936
0.0204.117
0.0404.775
0.0605.553
0.0806.268
0.1006.935
0.1207.480
0.1408.188
0.1608.361
0.1808.533
0.2008.771
0.2209.120
0.2409.024

```

```

0.2609.288
0.2809.462
0.3009.379
0.3209.685
0.3409.482
0.3609.545
0.3809.604
0.4009.546

```

The above code can also be expressed in function optimization format

```

Algorithm = UGO[100];
RowDataSet;
    x = 0.010, 0.020, 0.040, 0.060, 0.080, 0.100, 0.120, 0.140, 0.160, 0.180, 0.200, 0.220, 0.240, 0.260, 0.280,
        0.300, 0.320, 0.340, 0.360, 0.380, 0.400;
    y = 3.936, 4.117, 4.775, 5.553, 6.268, 6.935, 7.480, 8.188, 8.361, 8.533, 8.771, 9.120, 9.024, 9.288,
        9.462, 9.379, 9.685, 9.482, 9.545, 9.604, 9.546;
EndRowDataSet;
MinFunction Sum(x,y)(((p3+p2*p1*x^p4)/(p5+p1*x^p4)-y)^2);
    p2=3*p3+sum(i=1:3)(p[i]);
    20.3 >=p1+p2>=20;

```

This question looks easy but it is actually a real difficult optimization problem. There is one local optimal solution, which can be easy to found:

```

Min. = 2.32592892,p1 = 7.65783913843992E-02,p2 = 20.0086306479937,
p3 = -1.91445978460306E-02,p4 = .423355212168873,
p5 = 4.99918501444038E-02

```

But the global optimal solution is:

```

Min. = 2.235333,p1 = -3.83814476105978E-17,p2 = 20.0000612864639,
p3 = -2.41557981892391E-17,p4 = .486170393747984,
p5 = -2.66116003491166E-17

```

Note that “Algorithm = UGO1[100]” in code, this line of code set the algorithm to be used is Universal Global Optimization algorithm, the first local search type, number of parallel lines is 100. If use the number of parallel lines as default value, 30, it is difficult to get the optimal solution.

2.2.6 Curve fit with integration

There is an integration part in the curve fit equation below:

$$y = a - b \cdot \exp(-c_1 \cdot x^d) \cdot \int_{t=0.1}^c (t+x) dt \quad (2-2-9)$$

Auto2Fit code:

```

Parameter a,b,c,d;
Variable x,y;
Function y=a-b*exp(-c*x^d)*int((t+x),t=0.1,c);
RowData;
0.05,0.15,0.25,0.35,0.45,0.55,0.65,0.75,0.85,0.95,1.05,1.15,1.25,1.35,1.45;
0.13,0.13,0.19,0.34,0.53,0.71,1.06,1.6,1.64,1.83,2.09,2.05,2.13,2.12,2.09;

```

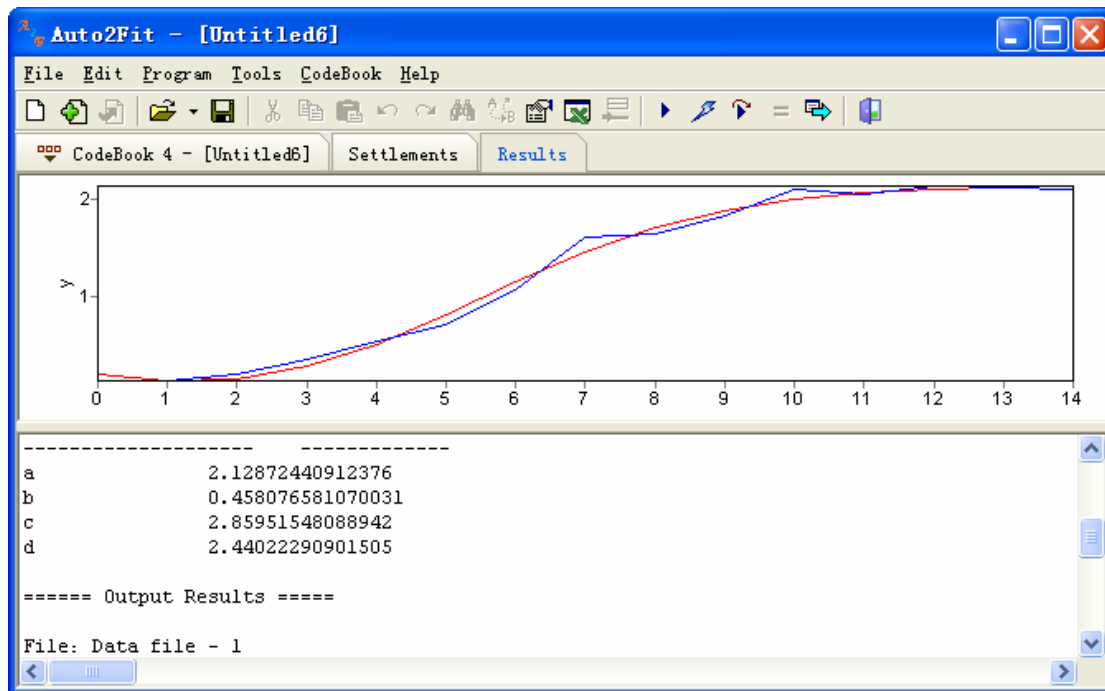


Figure 2-2-5 Result of curve fit with integration

2.2.7 Least-absolute and other special curve fit

Data for curve fit

x	-0.08, -0.065, -0.05, -0.03, -0.015, 0.015, 0.03, 0.05, 0.065, 0.08, 0
y	20.26, 19.73, 19.50, 18.73, 18.59, 18.59, 18.88, 19.55, 19.89, 20.99, 18.12

Curve fit equation:

$$y = a \cdot (x - x_0)^4 + b \cdot (x - x_0)^2 + c \quad (2-2-10)$$

■ General curve fit

General curve fit usually means least squares curve fit, i.e., the criteria of curve fit is:

$$\text{Min. } RSS = \sum_{i=1}^n (Y_i - y_i)^2$$

Here, Y , y are computation and target dependent variable respectively, RSS is Residue of Sum of Square, n is length of data for curve fit.

Parameter a,b,c,x0;

Variable x,y;

Function $y = a \cdot (x - x_0)^4 + b \cdot (x - x_0)^2 + c$;

Data;

-0.08,20.26

-0.065,19.73

-0.05,19.50

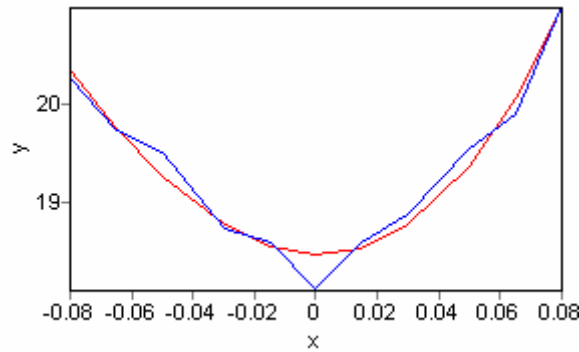
-0.03,18.73

-0.015,18.59

0.015,18.59

0.03,18.88
0.05,19.55
0.065,19.89
0.08,20.99
0,18.12

It is easy to get the following result using Universal Global Optimization algorithm.



Number of iteration: 36
Computation time (Hour:Minute:Second:Millisecond): 00:00:03:120
Optimization algorithm: Universal Global Optimization (UGO2)
Reason of computation stops: meet the criteria of convergence
Mean square deviation (RMSE): 0.154577732398945
Residual sum of squares (RSS): 0.262837028889598
Correlation coefficient (R): 0.981806182300711
The square of correlation coefficient (R^2): 0.963943379603898
Determination coefficient (DC): 0.963943379603897
Chi-Square: 0.00694913498803648
F-Statistic) 64.7130348795956

Parameter	Optimal estimation
a	421.902418131742
b	-170.887335709735
c	35.768453358506
x0	-0.448782169655507


- The curve fit criteria is to find the minimum value of maximum absolute difference of points between fitted curve and actual curve

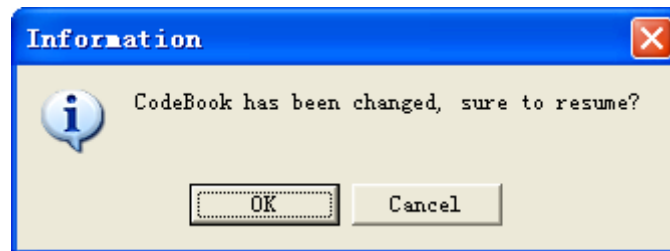
$$\text{Min. } MAD = \text{Max}(|Y_1 - y_1|, |Y_2 - y_2|, \dots, |Y_n - y_n|)$$

MAD(Max. Absolute Difference is the maximum absolute difference of points between fitted curve and actual curve. Key word “RegType” is used to define in this case.

RegType	Method of curve fit	Criteria of curve fit
0	Least square:RSS (Residue of Sum of Square)	Min. $RSS = \sum_{i=1}^n (Y_i - y_i)^2$

1	Least absolute value:SAE (Sum of Absolute Error)	Min. $SAE = \sum_{i=1}^n Y_i - y_i $
2	MAE (Max. Absolute Error)	Min. $MAE = \max(Y_1 - y_1 , Y_2 - y_2 , \dots, Y_n - y_n)$
3	MRAE (Max. Relative Absolute Error)	Min. $MRAE = \max\left(\left \frac{Y_1 - y_1}{y_1}\right , \left \frac{Y_2 - y_2}{y_2}\right , \dots, \left \frac{Y_n - y_n}{y_n}\right \right)$

Thus it is only necessary to add “RegType = 2” in the code. When the value of RegType is not equal to 0, i.e., the method used is not least square method, it is difficult to find a stable optimal parameter when use Auto2Fit with random initial values. At this moment, a hot execution mode of Auto2Fit can be used. Firstly use least square method to do curve fit, then change the value of RegType in the code and press hot execution key  to do the calculation when get a stable optimal solution. When the following information appears, choose “OK” button, good results can be found for other three cases except “RegType = 0”.



Target value	Calculation value			
	RegType = 0	RegType = 1	RegType = 2	RegType = 3
20.26	20.3313	20.2600	20.4821	20.4978
19.73	19.7514	19.7125	19.8319	19.8340
19.5	19.2625	19.2520	19.2779	19.2711
18.73	18.7752	18.7959	18.7173	18.7052
18.59	18.5513	18.5900	18.4528	18.4408
18.59	18.5311	18.5900	18.4040	18.3990
18.88	18.7655	18.8259	18.6579	18.6584
19.55	19.3674	19.4201	19.3302	19.3378
19.89	20.0577	20.0967	20.1121	20.1235
20.99	20.9718	20.9900	21.1562	21.1690
18.12	18.4649	18.5165	18.3421	18.3327

RegType	Parameter				RSS	SAE	MAE	MRAE
	a	b	c	x ₀				
0	421.90073	170.88733	35.76852	0.44878	0.26284	1.30094	0.34493	0.019035
1	417.40185	163.67643	34.56210	0.44254	0.28587	1.11867	0.39645	0.021879
2	580.24642	195.05023	34.73149	0.40759	0.38658	1.93441	0.22212	0.012258

3	525.53627	196.46359	36.69230	0.43030	0.40510	1.99478	0.23784	0.001325
---	-----------	-----------	----------	---------	---------	---------	---------	----------

2.2.8 Curve fit of implicit function

In some situations, the format of nonlinear regression is implicit function, such as:

$$-(x - p_4) \cdot \sin(p_3) + (y - p_5) \cdot \cos(p_3) + p_5 = p_1 + p_2 \cdot \sin(p_3) \cdot \ln((x - p_4) \cdot \cos(p_3) + (y - p_5) \cdot \sin(p_3) + p_4) \quad (2-2-11)$$

Here, x is independent variable, y is dependent variable. As for this category of problem, Auto2Fit can easily handle, the format of expression is the same.

Auto2Fit code:

```
Variable x, y;
Function -(x-p4)*sin(p3)+(y-p5)*cos(p3)+p5 =
          p1+p2*sin(p3)*ln((x-p4)*cos(p3)+(y-p5)*sin(p3)+p4);
Data;
//x   y
0.0000 262.5079
1.0000 262.8300
2.0000 263.1452
3.0000 263.4532
```

Example: Nonlinear regression of involute implicit function

Involute function and data are known as follows :

$$\begin{cases} x = R \cdot [\cos(t) + (t - B) \cdot \sin(t)] + x_0 \\ y = R \cdot [\sin(t) - (t - B) \cdot \cos(t)] + y_0 \end{cases} \quad (2-2-12)$$

Here, t is independent variable, R, B, x_0, y_0 are parameters.

Data

No.	X	Y
1	15.5910	86.2142
2	24.6601	83.7114
3	33.3732	80.2618
4	49.3445	70.7216
5	62.7992	58.0891
6	68.3962	50.8058
7	73.1584	42.9946
8	79.9955	26.1718
9	83.0531	8.4225

The difficulty is that the value of t is unknown corresponding to x and y . There are three methods to solve this problem using Auto2Fit:

- ✧ Method 1: Consider t as a parameter to be solved, and then there are nine parameters with t , and plus other four parameters R, B, x_0, y_0 , so there are 13 parameters in total. Objective

function of optimization is to minimize the following function:

$$\begin{aligned} \min : f &= \sum_{i=1}^9 (y_i - y'_i)^2 + \sum_{i=1}^9 (x_i - x'_i)^2 \\ &= \sum_{i=1}^9 (y_i - (R \cdot [\sin(t) - (t - B) \cdot \cos(t)] + y_0))^2 + \\ &\quad \sum_{i=1}^9 (x_i - (R \cdot [\cos(t) + (t - B) \cdot \sin(t)] + x_0))^2 \end{aligned} \quad (2-2-13)$$

Auto2Fit code:

```
Parameter R,B,x0,y0;
Parameter t(1:9);
DataSet;
X, Y =
15.5910    86.2142
24.6601    83.7114
33.3732    80.2618
49.3445    70.7216
62.7992    58.0891
68.3962    50.8058
73.1584    42.9946
79.9955    26.1718
83.0531     8.4225
EndDataSet;
MinFunction Sum(i=1:9)((-X[i]+R*(COS(t[i])+(t[i]-B)*SIN(t[i]))+X0)^2)+
Sum(i=1:9)((-Y[i]+R*(SIN(t[i])-(t[i]-B)*COS(t[i]))+Y0)^2);
```

Result: R = 3.1995227, B = -24.438500, x0 = -0.000329680, y0 = 0.000374680
t1 = 2.926154, t2 = 2.8184350, t3 = 2.710714, t4 = 2.495275, t5 = 2.2798359
t6 = 2.1721167, t7 = 2.064396, t8 = 1.848957, t9 = 1.633520

The correction solution can be found using this method. But when the volume of data is large, e.g., 500 sets of data, there would be 500 values of t need to be solved as parameter. Such a large number of parameter will increase the difficulty of finding the solution, no mention that the intermediate variable t is not the value we want.

✧ Method 2: use key word “ParVariable” define t , then use “SharedModel” to do the curve fit directly.

Auto2Fit code:

```
Parameter R,B,x0,y0;
ParVariable t;
Variable x, y;
SharedModel;
Function x = R*(cos(t)+(t-B)*sin(t))+x0;
Function y = R*(sin(t)-(t-b)*cos(t))+y0;
Data;
15.5910    86.2142
24.6601    83.7114
33.3732    80.2618
49.3445    70.7216
62.7992    58.0891
68.3962    50.8058
73.1584    42.9946
79.9955    26.1718
83.0531     8.4225
```

✧ Method 3: transform the formula to eliminate the t variable

From equation

$$\begin{cases} x = R \cdot [\cos(t) + (t - B) \cdot \sin(t)] + x_0 \\ y = R \cdot [\sin(t) - (t - B) \cdot \cos(t)] + y_0 \end{cases} \quad (2-2-14)$$

We can get

$$\frac{(x - x_0)^2}{R^2} + \frac{(y - y_0)^2}{R^2} = 1 + (t - B)^2 \quad (2-2-15)$$

it is

$$t = \sqrt{\frac{(x - x_0)^2}{R^2} + \frac{(y - y_0)^2}{R^2} - 1} + B \quad (2-2-16)$$

Objective function like method 1 is

$$f = \sum_{i=1}^9 (y_i - (R \cdot [\sin(t) - (t - B) \cdot \cos(t)] + y_0))^2 + \sum_{i=1}^9 (x_i - (R \cdot [\cos(t) + (t - B) \cdot \sin(t)] + x_0))^2 \quad (2-2-17)$$

Then replace t in objective function using 2-2-16. Note that use key word “ConstStr”.

Auto2Fit code:

```
ConstStr t = B+Sqrt(1/R^2*((X[i]-X0)^2+(Y[i]-Y0)^2)-1);
Parameter R ,B,X0,Y0;
DataSet;
X, Y =
15.5910    86.2142
24.6601    83.7114
33.3732    80.2618
49.3445    70.7216
62.7992    58.0891
68.3962    50.8058
73.1584    42.9946
79.9955    26.1718
83.0531     8.4225
EndDataSet;
MinFunction Sum(i=1:9)((-X[i]+R*(COS(t)+(t-B)*SIN(t))+X0)^2)+
Sum(i=1:9)((-Y[i]+R*(SIN(t)-(t-B)*COS(t))+Y0)^2);
```

Result:R = 3.199541, B = 0.694399, x0 = -0.0003171, y0 = 0.00035955

2.2.9 Auto-search of function for curve fit

In lots of situations, only the data of independent and dependent variables are known while the model function is unknown. In this case, how to find a best-fit model equation? Auto2Fit provides a function to automatic search equation for this kind of problem.

Automatic search equation for curve fit is used to find the best equation fit the data and make a list of equations based on the results when the model equation is unknown. As for two-variable and three-variable data, the function library includes 3700 and 1000 kinds of different types of equations. As for more than three-variable curve fit, the end-user must add equation library

themselves. Whatever two-variable, three-variable or more variables, the following rules must be followed when user adds equation to library:

- Two-variable: x represents independent variable, y represents dependent variable, p1, p2, p3,...represent parameters
- Three-variable: x, y represent independent variables, z represent dependent variable, p1, p2, p3,... represent parameters
- More than three variables: x1, x2, x3,... represent independent variables, y represent dependent variables, p1, p2, p3,... represent parameters
- Parameter is represented by p, subscript starts from 1, which must be continuous, if there are three parameters, they can only defined like p1, p2, p3 instead of p1, p3, p4

Equation acquired through Quick Fit cannot guarantee 100% optimal. Thus it is better to choose equation first and then use general curve fit method to verify the result. If there is some physical meaning constraint, corresponding constrained process must be done accordingly.

Only key words “Data” and “RowData” are needed when use Quick Fit.

Automatic equation search example

Data

x	15,30,45,60,75,90,105,120,135,495
y	0.489,0.427,0.373,0.327,0.285,0.250,0.218,0.191,0.167,0.005

Auto2Fit code

Data;	
15	0.489
30	0.427
45	0.373
60	0.327
75	0.285
90	0.250
105	0.218
120	0.191
135	0.167
495	0.005

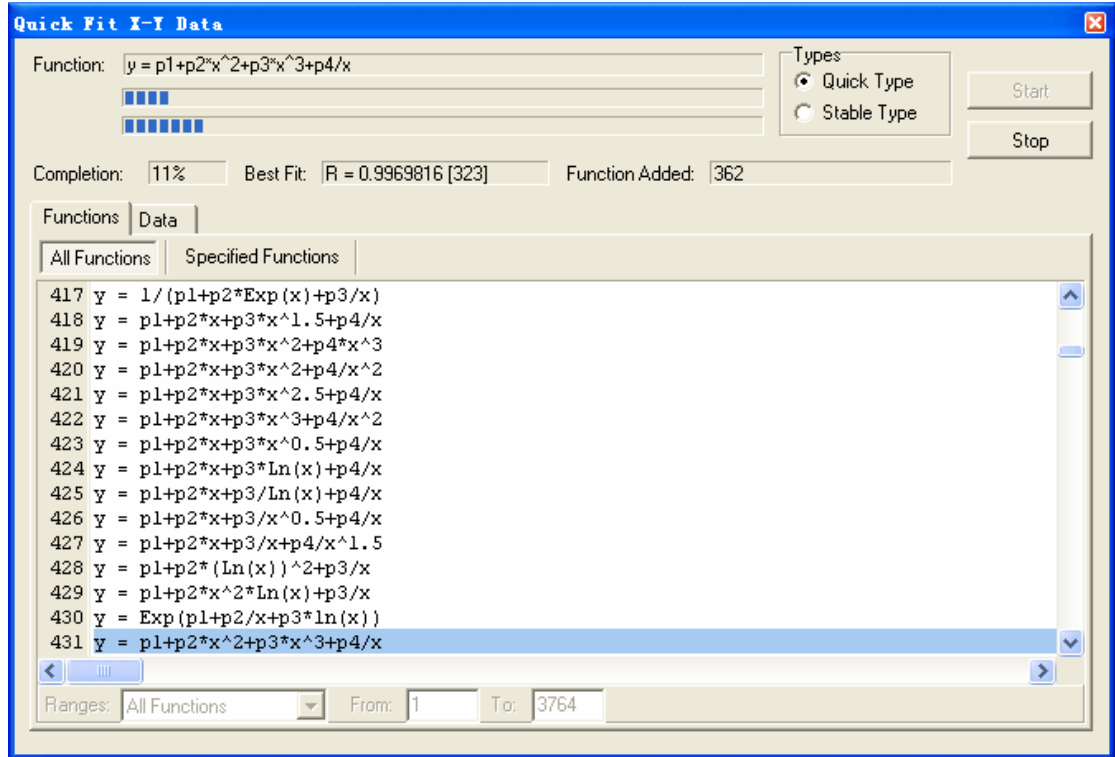


Figure 2-2-6 interface of automatic equation search

2.2.10 Range setting of initial value for curve fit

Curve fit equation:

$$y = 1 - \text{Exp}\left(\frac{-d(b+cx)x}{a+x}\right) \quad (2-2-18)$$

Characteristics:

- The value of x is very large, it is not in the order of magnitude of y;
- From fit equation, c,b,d may not be to the sole, i.e., over fitting of parameters

Solutions:

- In Auto2Fit, the default parameter start range is [0,5] ,obviously, it cannot meet the requirement in this case, thus set the parameter a and c's start range as [0,1E+30] and [-1E-35,1E-10] ;
- Though for curve fit problem, the first choice is Universal Global Optimization (UGO1), Differential Evolution (DE) algorithm is most effective in this case.

Auto2Fit code:

```
Parameters a[0,1e+30,,], b, c[-1e-35,1e-10,,], d;
Algorithm = DE1[100];
Variable x, y;
Function y=1-EXP(-d*(b+c*x)*x/(a+x));
```

```
Data:// x, y
7.50E+21 0.01
1.10E+22 0.018
1.70E+22 0.029
3.40E+22 0.053
1.70E+23 0.09
3.40E+23 0.11
6.70E+23 0.12
8.40E+23 0.14
1.10E+24 0.15
1.70E+24 0.17
2.10E+24 0.24
3.40E+24 0.31
4.20E+24 0.32
6.70E+24 0.49
8.40E+24 0.5
```

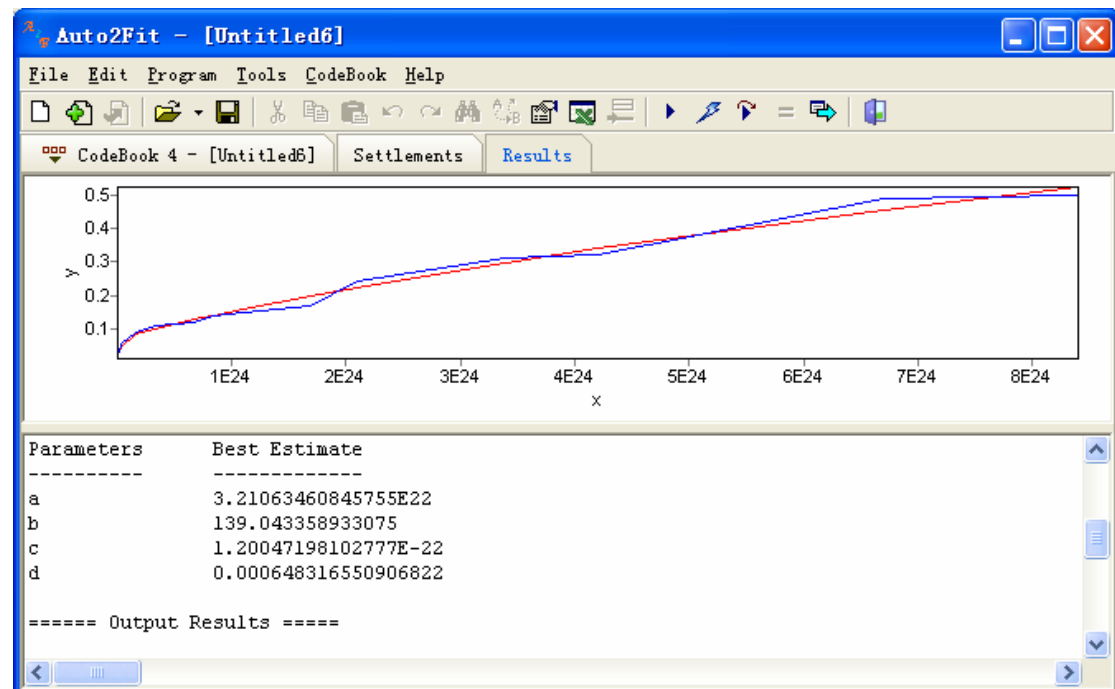


Figure 2-2-7 Result of curve fit

2.2.11 Two/three-dimension analysis and prediction/verification

After optimization computation finishes, two/three dimension analysis and data prediction/verification can be preceded. Two/three dimensional analysis mainly shows the relationships between the value of objective function and one or two parameters. Prediction/verification is used to compute the unknown point, e.g., calculate dependent variable based on independent variable after the computation of curve fit.

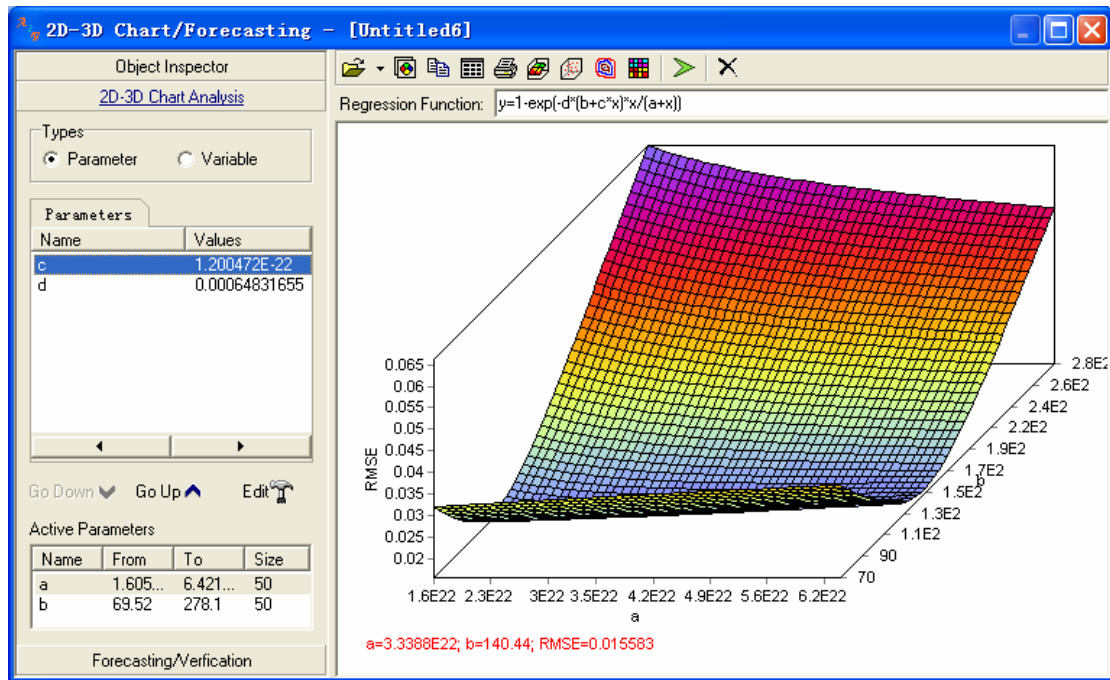


Figure 2-2-8 Two/three dimensional analysis

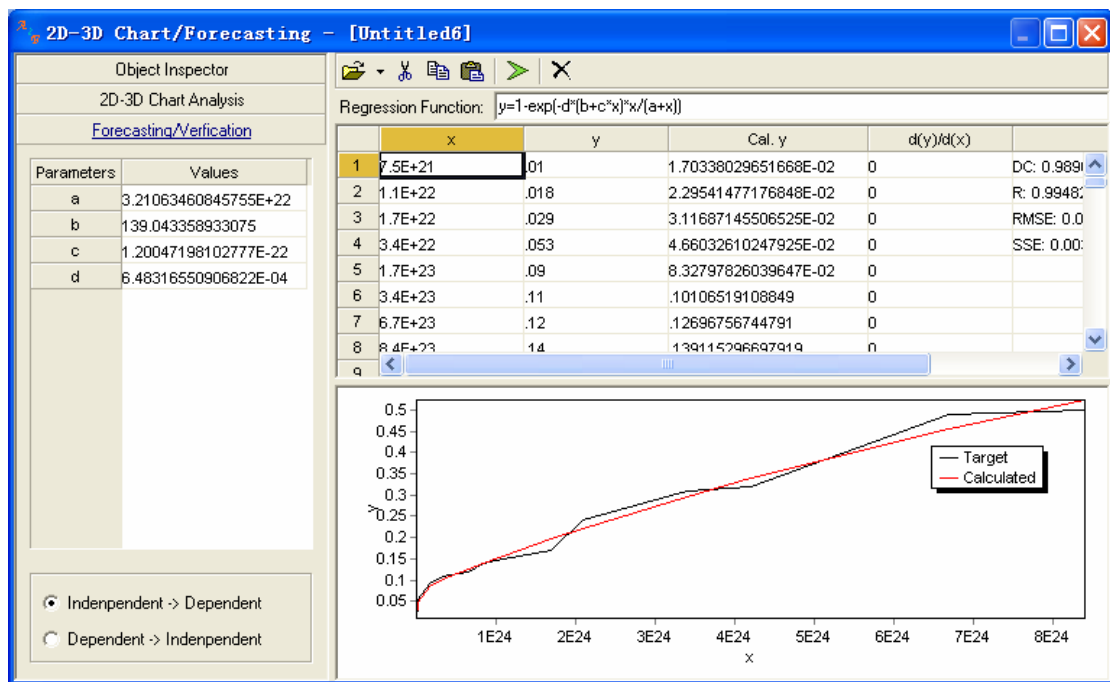


Figure 2-2-9 Prediction and verification

2.3 Solving equation and system equations

Auto2Fit can solve any type of linear, nonlinear equation or system equations. The key word used is "Function".

2.3.1 Solving general system equations

Solving system equations, example-1

$$\begin{cases} (x-0.3)^{y^z} + \frac{x}{y \cdot z} - x \cdot y \cdot \sin(z) + (x+y-z)^{\cos(x-1)} = 1 \\ (y-0.2)^{z \cdot x} + \frac{y}{z \cdot x} - y \cdot z \cdot \sin(x) + (y+z-x)^{\cos(y-2)} = 2 \\ (z-0.1)^{x \cdot y} + \frac{z}{x \cdot y} - z \cdot x \cdot \sin(y) + (z+x-y)^{\cos(z-3)} = 3 \end{cases} \quad (2-3-1)$$

Auto2Fit code:

Parameter	x, y, z;
Function	(x-0.3)^y^z+x/y/z-x*y*sin(z)+(x+y-z)^cos(x-1) = 1; (y-0.2)^z*x+y/z/x-y*z*sin(x)+(y+z-x)^cos(y-2) = 2; (z-0.1)^x^y+z/x/y-z*x*sin(y)+(z+x-y)^cos(z-3) = 3;

Result : x = 0.79390634413219, y = 0.902585377949916, z = 1.21622367662841

Solving system equations, example-2

$$\begin{cases} \exp(0.1 \cdot x_1) - \exp(0.1 \cdot x_2) - x_3 (\exp(-0.1) - \exp(-10 \cdot 0.1)) = 0 \\ \exp(0.2 \cdot x_1) - \exp(0.2 \cdot x_2) - x_3 (\exp(-0.2) - \exp(-10 \cdot 0.2)) = 0 \\ \exp(0.3 \cdot x_1) - \exp(0.3 \cdot x_2) - x_3 (\exp(-0.3) - \exp(-10 \cdot 0.3)) = 0 \end{cases} \quad (2-3-2)$$

here, $x_1 \in [-100, 100]$, $x_2 \in [-100, 100]$, $x_3 \in [0.1, 100]$

Auto2Fit code:

Parameter	x1[-100,100], x2[-100,100], x3[0.1,100];
Function	exp(-0.1*x1)-exp(0.1*x2)-x3*(exp(-0.1)-exp(-10*0.1))=0; exp(-0.2*x1)-exp(0.2*x2)-x3*(exp(-0.2)-exp(-10*0.2))=0; exp(-0.3*x1)-exp(0.3*x2)-x3*(exp(-0.3)-exp(-10*0.3))=0;

Results: $x_1 = 1, x_2 = -10, x_3 = 1$

2.3.2 Solving loop equation

See the following system functions, k is variable and ranged in $[0, 1]$, length of step is 0.05, try to find the values of x and y corresponding to different value of k .

$$\begin{cases} 0.23 + 0.32 \left(1 + 1.5 \left(\frac{x}{0.18} - 1 \right) k - 0.5 \left(\frac{x}{0.18} - 1 \right)^3 \right) - y = 0 \\ x - k \cdot y^{0.5} = 0 \end{cases} \quad (2-3-3)$$

In Auto2Fit, use key word "LoopConstant" to describe the above problem, the code is as follows:

LoopConstant	k=[0:0.05:1];
PlotLoopData	x[x], y;
Function	0.23+0.32*(1+1.5*(x/0.18-1)*k-0.5*(x/0.18-1)^3)-y; x-k*y^0.5;

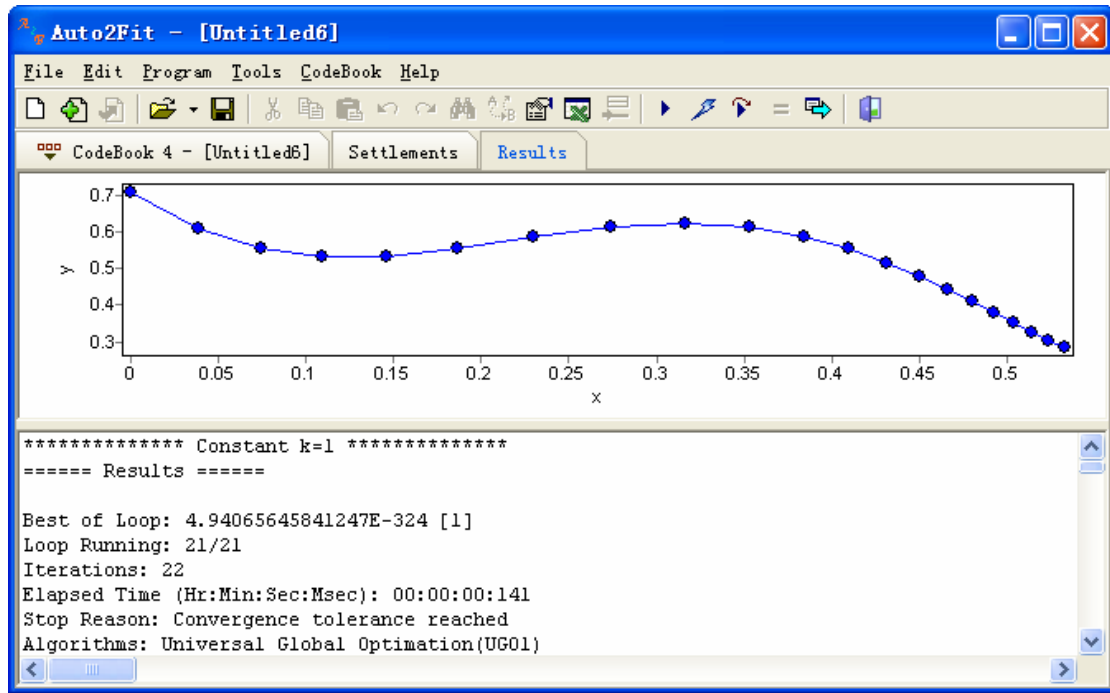


Figure 2-3-1 Result of loop equation solve

2.3.3 Solving loop iteration equation

See the following iteration equation, $n = 50$, $x_n = 200$, $y_n = 0.3$, try to find the values of x_n , x_{n-1} , $x_{n-2} \dots x_1$ and y_n , y_{n-1} , $y_{n-2} \dots y_1$

$$\begin{cases} (x_{i+1} - x_i) \cdot y_{i+1} = \frac{L}{2} \cdot \left(\frac{-4.5 \cdot (\sin(y_{i+1}) + \sin(y_i)) + 0.02 \cdot ((1 + 2x_{i+1})^{0.5} \cdot \cos(y_{i+1})^2 + (1 + 2x_i)^{0.5} \cdot \cos(y_i)^2)}{2 \cdot ((1 + 2x_{i+1})^{0.5} \cdot \sin(y_{i+1})^2 + (1 + 2x_i)^{0.5} \cdot \sin(y_i)^2)} \right) \\ (y_{i+1} - y_i) \cdot x_{i+1} = \frac{L}{x_{i+1} + x_i} \cdot \left(\frac{-4.5 \cdot (\cos(y_{i+1}) + \cos(y_i)) + 2 \cdot ((1 + 2x_{i+1})^{0.5} \cdot \sin(y_{i+1})^2 + (1 + 2x_i)^{0.5} \cdot \sin(y_i)^2)}{2 \cdot ((1 + 2x_{i+1})^{0.5} \cdot \sin(y_{i+1})^2 + (1 + 2x_i)^{0.5} \cdot \sin(y_i)^2)} \right) \end{cases} \quad (2-3-4)$$

This example is to try to find the value of x_i and y_i if the values of x_{i+1} and y_{i+1} are known. After finding the values of x_i and y_i , try to find the values of x_{i-1} and y_{i-1} , ... similarly, find the values of x_1 and y_1 , Auto2Fit code is as follows:

```

Constant n=50, L=100/n;
LoopConstant x2(n)=[200, x1(n-1)], y2(n)=[0.3, y1(n-1)];
PlotLoopData y;
function (x2-x1)*y2=1/2*(-4.5*(sin(y2)+sin(y1))+0.02*(sqrt(1+2*x2)*cos(y2)^2+sqrt(1+2*x1)*cos(y1)^2));
(y2-y1)*x2=1/(x2+x1)*(-4.5*(cos(y2)+cos(y1))+2*(sqrt(1+2*x2)*sin(y2)^2+sqrt(1+2*x1)*sin(y1)^2));

```

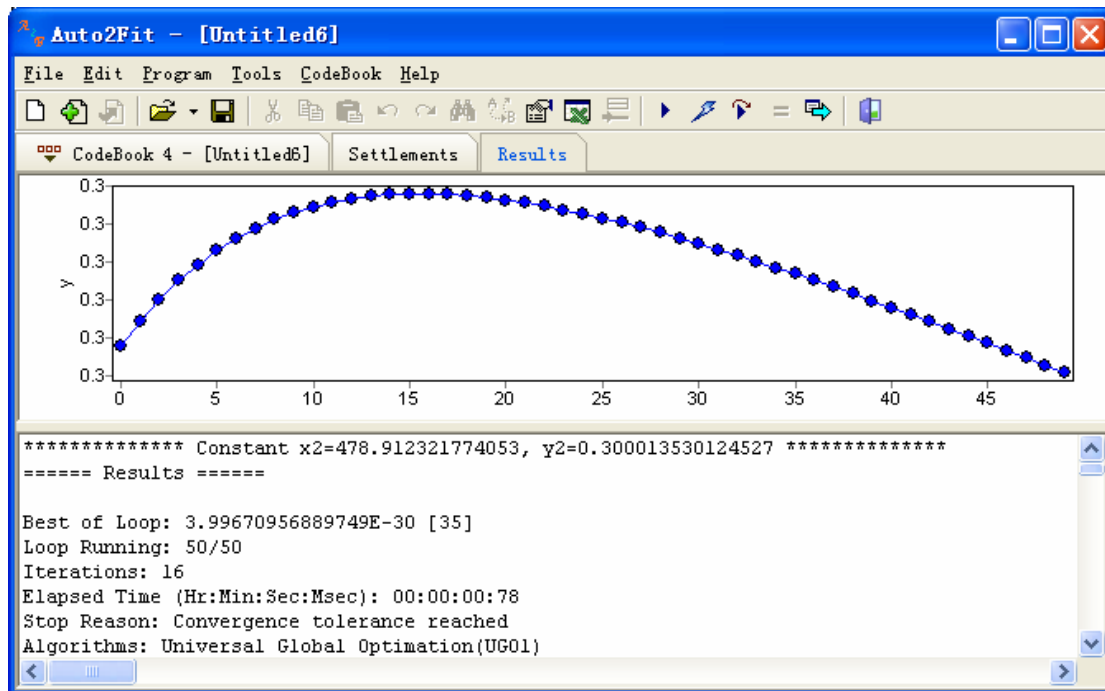


Figure 2-3-2 Result of solving loop iteration equation

Results:

No.	x	y		X	Y
1	200	0.3	26	350.2779362	0.3002137
2	206.4100362	0.3000380	27	355.9099442	0.3002080
3	212.7821026	0.3000712	28	361.5162867	0.3002019
4	219.1169717	0.3001001	29	367.0972725	0.3001955
5	225.4153787	0.3001251	30	372.6532023	0.3001887
6	231.6780239	0.3001467	31	378.1843685	0.3001816
7	237.9055759	0.3001653	32	383.6910563	0.3001742
8	244.0986735	0.3001812	33	389.1735433	0.3001666
9	250.2579283	0.3001947	34	394.6321003	0.3001587
10	256.3839261	0.3002060	35	400.0669914	0.3001506
11	262.4772291	0.3002154	36	405.4784742	0.3001424
12	268.5383772	0.3002230	37	410.8668003	0.3001339
13	274.5678896	0.3002290	38	416.2322152	0.3001253
14	280.5662659	0.3002335	39	421.5749587	0.3001166
15	286.5339874	0.3002368	40	426.8952653	0.3001077
16	292.4715183	0.3002388	41	432.1933640	0.3000986
17	298.3793065	0.3002398	42	437.4694789	0.3000895
18	304.2577846	0.3002398	43	442.7238291	0.3000803
19	310.1073709	0.3002390	44	447.9566290	0.3000710
20	315.9284698	0.3002373	45	453.1680884	0.3000615
21	321.7214731	0.3002349	46	458.3584127	0.3000521
22	327.4867600	0.3002318	47	463.5278031	0.3000425
23	333.2246983	0.3002281	48	468.6764567	0.3000329
24	338.9356446	0.3002238	49	473.8045665	0.3000232
25	344.6199452	0.3002190	50	478.9123218	0.3000135

2.3.4 Solving integer equation

Magic matrix problems

Put the number from 1 to 16 in a table with 4 columns and 4 rows. Every number is used only once. How to arrange the numbers to make the sum of every column, every row and two diagonals equal to 34 respectively? In another word, find the values of x_1 to x_{16} .

	x_1	x_2	x_3	x_4	34
	x_5	x_6	x_7	x_8	34
	x_9	x_{10}	x_{11}	x_{12}	34
	x_{13}	x_{14}	x_{15}	x_{16}	34
34	34	34	34	34	34

This problem can be classified as solving integer equation. Auto2Fit code and results are as follow:

Auto2Fit code:

```

Constant S = 34;
Parameters x(1:16)[1,16,0];
Exclusive = true;
Function x1 + x2 + x3 + x4 = S;
        x5 + x6 + x7 + x8 = S;
        x9 + x10 + x11 + x12 = S;
        x13 + x14 + x15 + x16 = S;
        x1 + x5 + x9 + x13 = S;
        x2 + x6 + x10 + x14 = S;
        x3 + x7 + x11 + x15 = S;
        x4 + x8 + x12 + x16 = S;
        x1 + x6 + x11 + x16 = S;
        x4 + x7 + x10 + x13 = S;

```

Results of magic matrix

	2	14	15	3	34
	7	11	10	6	34
	9	5	8	12	34
	16	4	1	13	34
34	34	34	34	34	34

Results: $x_1 = 2$, $x_2 = 14$, $x_3 = 15$, $x_4 = 3$, $x_5 = 7$, $x_6 = 11$, $x_7 = 10$, $x_8 = 6$, $x_9 = 9$, $x_{10} = 5$, $x_{11} = 8$, $x_{12} = 12$, $x_{13} = 16$, $x_{14} = 4$, $x_{15} = 1$, $x_{16} = 13$

2.4 Numerical solution of ordinary differential equations (ODE)

In many cases, we are unable to get the analytical solution of ordinary differential equations. Then numerical methods can only be used to solve the problem. Solving ODE is an important and difficult job.

2.4.1 Key words for solving differential equations in Auto2Fit

Key words	Meaning
Variable	Define name of variables and their range, essential
ODEFunction	Define differential equation or system equations, essential
Plot,PlotLoopData	Plot command, optional
ChartType	Set the type of curve, 1: dash-line graph, 2: line-line graph, 3: dot-dot graph, optional
ODEOptions	<p>Solving ordinary differential equations option, basic expression is as follows: <code>ODEOptions = [SN=10,A=0,P=5]</code> here:</p> <ul style="list-style-type: none"> • SN: number of step, integer, it can be changed to SS, which means length of step; • A: set the algorithm of solving ordinary differential equation. 0: Runge-Kutta-Fehlberg Method, 1: Euler method, 2: Second order Runge-Kutta Method, 3: Third order Runge-Kutta Method, 4: Forth order Runge-Kutta Method, 5: Fifth order Runge-Kutta Method. • P: number of group, only use to solve boundary value optimization problem, the value is larger, the possibility of computation convergence is higher but the computation time is longer. <p>E.g. <code>ODEOptions = [SN=10,A=0,P=5]</code> Number of step is 10, algorithm is Runge-Kutta-Fehlberg Method, number of groups is 5.</p> <p>E.g.: <code>ODEOptions = [SS=0.1,A=4,P=20]</code> Length of step is 0.1, algorithms fourth order Runge-Kutta method, number of groups is 5.</p>

Ordinary differential equation options can also be set through control panel. Code-level setting takes precedence over control panel settings.

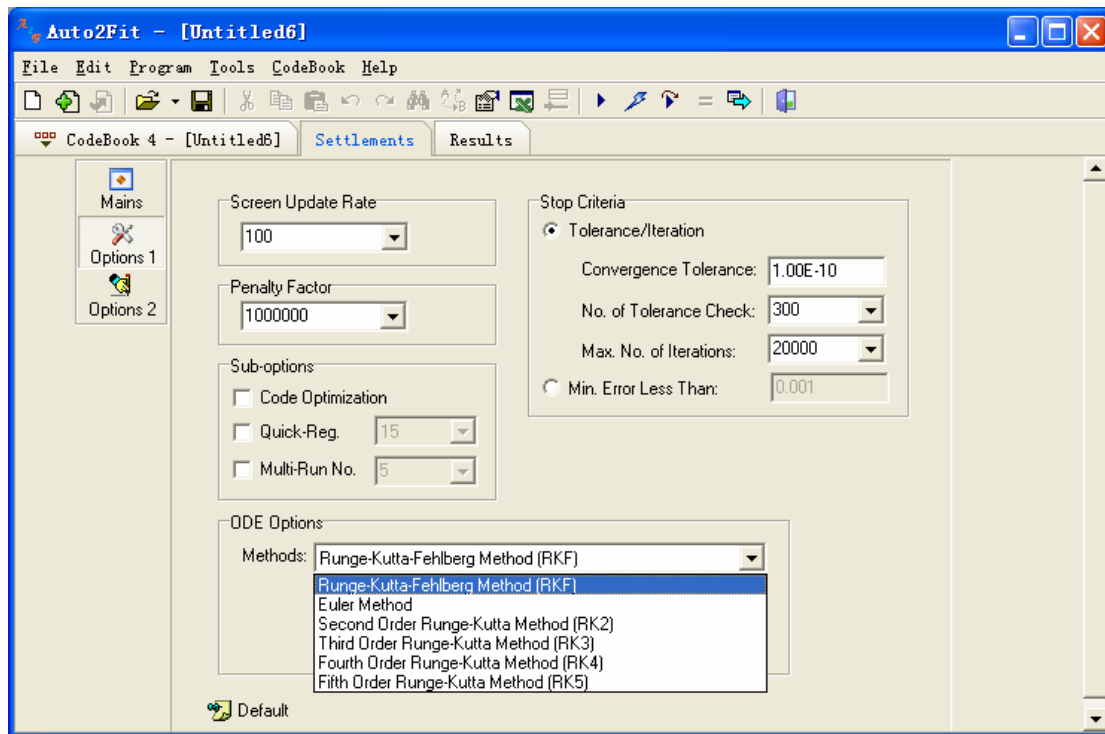


Figure 2-4-1 Settings of ordinary differential equation

2.4.2 Initial Value Problem (IVP) of ODE

Ordinary differential equation, example 1

$$y' = y^{\sin(t-y)} - \ln(yt) \quad (2-4-2)$$

Initial value: when $t = 0.2$, $y' = 1.5$, t is ranged in $[0.2, 4]$

Auto2Fit code

```
Variable t=[0.2 :0.1 :4], y=1.5;
Plot t[x], y, y'[y2];
ODEFunction y'=y^(sin(t-y))-ln(y*t);
```

In the above code, “Plot t[x], y, y'[y2]” means t is as x axis, y is as left vertical axis, y' is as right vertical axis.

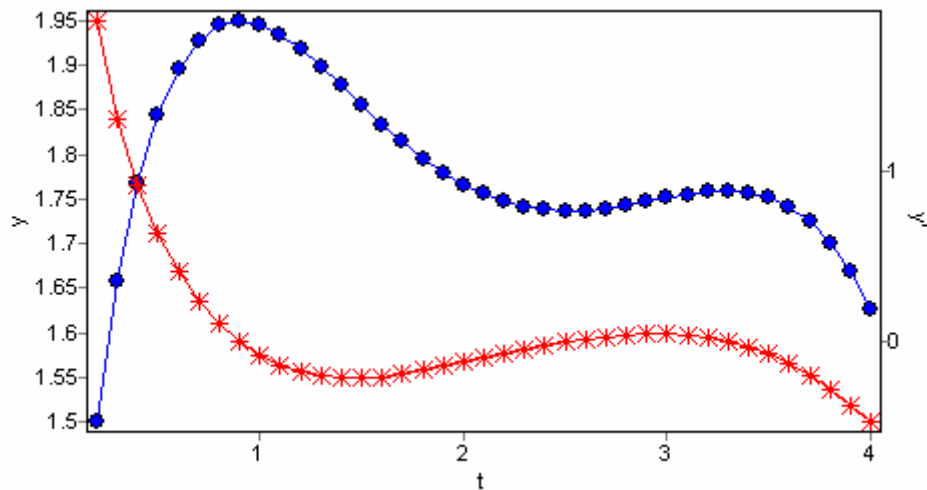


Figure 2-4-2 Result of ordinary differential equation example 1

Ordinary differential equation example 2

$$y'' + 2 \cdot t \cdot y' + y = f(t)$$

(2-4-3)

here $f(t)$ is a piecewise function of t :
$$f(t) = \begin{cases} t & t < 1 \\ 2 \cdot t & 1 \leq t < 2 \\ 4 & 2 \leq t < 5 \\ 0 & t > 5 \end{cases}$$

Initial values: $y(0) = 0, y'(0) = 0$, plot the curve of y, y', y'' while t changes,
Auto2Fit code

```
Variable t=[0:0.1:6], y=0,y'=0;
ConstStr f=if(t<=1, t, if((t>1)and(t<=2),2*t,if((t>2)and(t<=5),4,0)));
Plot t[x],y,y',y'';
ODEFunction y''=f-2*t*y'-y*t;
```

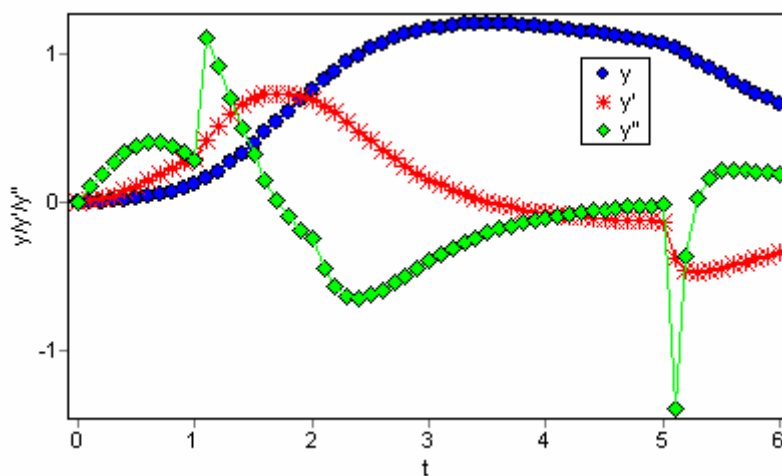


Figure 2-4-3 Result of ordinary differential equation example 2

2.4.3 Implicit ordinary differential equation and system equations

Auto2Fit can be used to solve the first-order implicit ordinary differential equation or system equations; e.g., x is ranged in $[0, 2.5]$, initial values: $y_1(0) = 1$, $y_2(0) = 0.25$.

$$\begin{cases} \frac{dy_1}{dx} = \cos\left(y_1 - \sin(x + y_2) + \frac{dy_2}{dx}\right) - \sin\left(\frac{2 \cdot x}{y_1} + y_2\right) \\ \frac{dy_2}{dx} = -2 \cdot x \cdot y_2 + y_1 + \sin\left(x - \frac{dy_1}{dx}\right) \cdot y_1 \end{cases} \quad (2-4-4)$$

Auto2Fit code

```
Variable y1=1,y2=0.25, x=[0,2.5];
Plot y1[x], y2;
ODEFunction y1'=cos(y1-sin(x+y2)+y2')-sin(2*x/y1+y2);
y2'=-2*x*y2+y1+sin(x-y1')*y1;
```

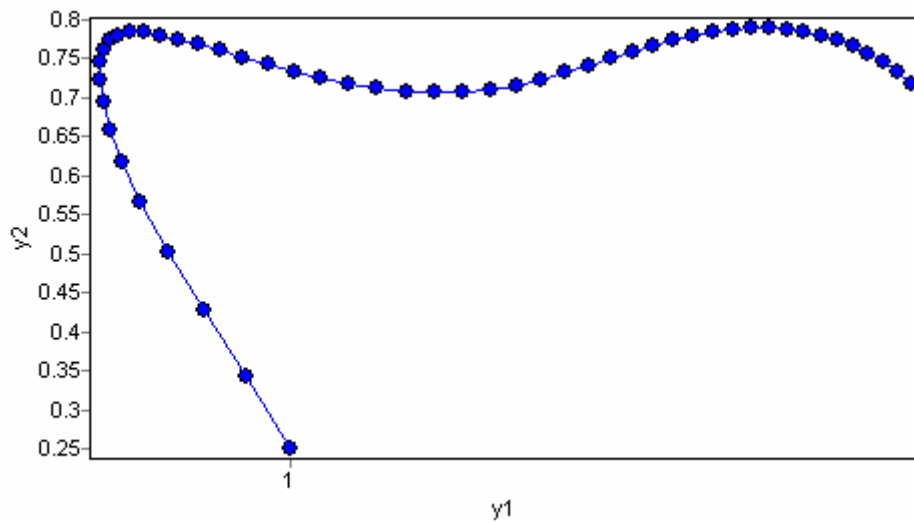


Figure 2-4-4 Results of implicit ordinary differential equation

2.4.4 Ordinary differential equation with variable coefficients

Example 1 : ordinary differential equation with variable coefficients

$$\begin{cases} \frac{dx}{dt} = y - \cos(x + y - a) + y \cdot (\sin(x^2 \cdot y + a)) \\ \frac{dy}{dt} = x - x^3 - 0.1 \cdot y + 0.5 \cdot \cos(0.2 \cdot t \cdot y + x - a) \cdot t \end{cases} \quad (2-4-5)$$

Here : a is variable parameter and is ranged in [0,3], extent of variation is 0.02, t is ranged in [0,2], initial value: x=0, y=0.

Auto2Fit code

```
LoopConstant a=[1:0.02:3];
Variable x=0, y=0, t=[0,2];
Plot x[x],y,y',x';
ODEFunction x'=y-cos(x+y-a)+y*(sin(x^2*y+a));
y'=x-x^3-0.1*y+0.5*cos(0.2*t*y+x-a)*t;
```

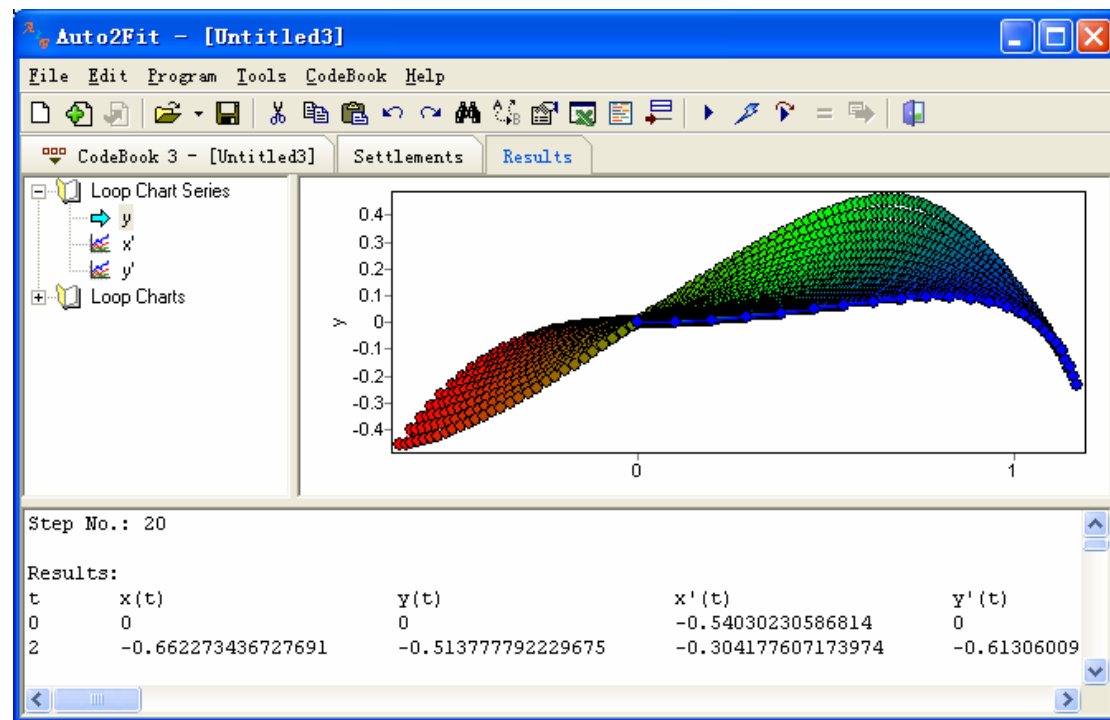


Figure 2-4-5 Result of ordinary differential equation with variable coefficients

Example 2 BVP with changeable parameters

$$y'' = \frac{dy}{dt} = (1 - y^2 + a) \cdot y' - \sin(t \cdot y') \cdot y + \cos(t \cdot y') \quad (2-4-6)$$

Boundary condition: when $t = 0$, $y = a$, $y' = 2 \cdot a$, a is a coefficient of variation and is ranged in [0, 1], extent of variation is 0.01.

Auto2Fit code

```
LoopConstant a=[0:0.01:1];
Plot y,y'[x],y'';
Variable t=[0,2], y=a, y'=1*a*2;
ODEFunction y''=(1-y^2+a)*y'-sin(t*y')*y+cos(t*y');
```

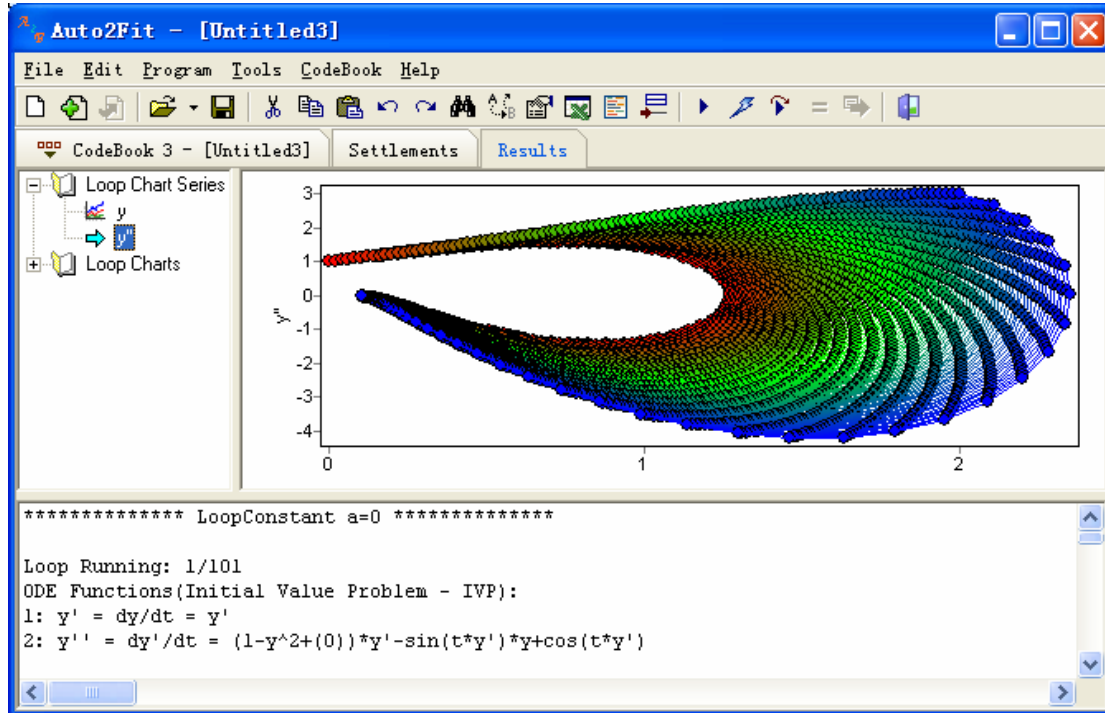



Figure 2-4-6 Results of variable boundary initial value differential equations

Right click and choose “Catch data”:

Auto2Fit Spreadsheet - [Untitled]

File View Edit Format Tools

Base Folder

Data-1

Data-2

Data-3

Data-4

Data-5

Data-6

Data-7

Data-8

Data-9

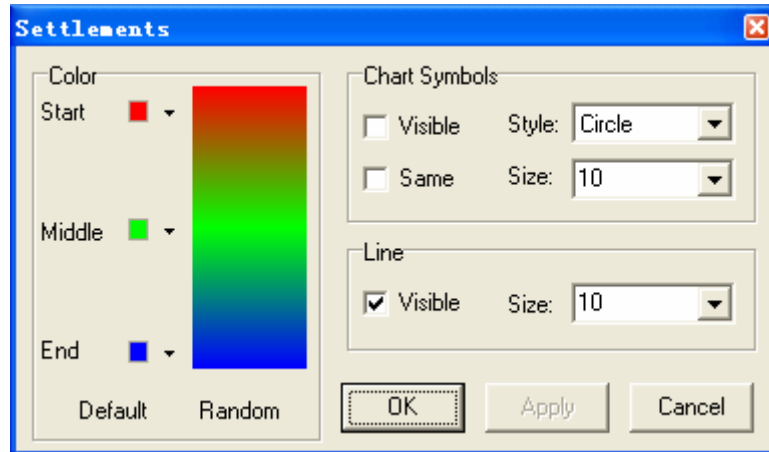
Data-10

Data-11

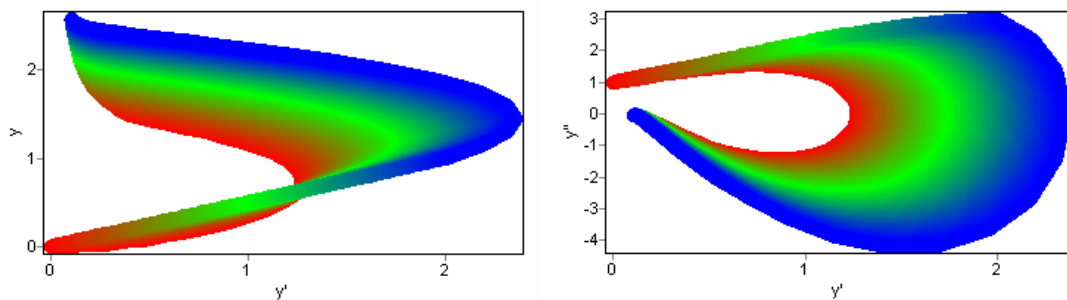
	A	B	C	D	E	F	G
1	循环常数: a=1						a=1.02
2	t	x(t)	y(t)	x'(t)	y'(t)		xi
3	0	0	0	-5403023058680			0
4	.04	-2.12536360967	-2.15752885928	-522510611230	-1.07765434782		.04
5	.08	-4.18112560873	-8.60960906640	-505499355178	-2.14662463389		.08
6	.12	-6.17038017515	-1.93143661127	-489254315325	-3.20350550239		.12
7	.16	-8.09615983604	-3.42181771559	-473758996971	-4.24580565264		.16
8	.2	-.099614276037	-5.32590683569	-458995227730	-5.27182109755		.2
9	.24	-.117690712853	-7.63696366920	-444943652242	-.062805239025		.24
10	.28	-.135218995967	-1.03479555293	-431584142535	-.072714649747		.28
11	.32	-.152226397732	-1.34517715718	-418896137386	-8.24468951336		.32
12	.36	-.168739364135	-.016941405514	-406858922074	-9.20066380542		.36
13	.4	-.184783513412	-2.08101111351	-395451858192	-.101402121348		.4
14	.44	-.200383642982	-2.50515343688	-384654571675	-.110644626723		.44
15	.48	-.215563743194	-2.96598253790	-374447105924	-.119748013099		.48
16	.52	-.230347016591	-3.46297335967	-364810045814	-.128728325037		.52
17	.56	-.244755901640	-3.99566883546	-355724617420	-.137603462801		.56
18	.6	-.258812100027	-4.56368674300	-347172767525	-.146392906523		.6
19	.64	-.272536606798	-5.16672555221	-339137226300	-.155117487706		.64
20	.68	-.285949742700	-.058045694436	-331601555985	-.163799202116		.68
21							

Sheet1: Cell[0, 0] - [A1 = 循环常数: a=1]

Right click and choose “Chart option...” on the graph:



Set the thickness of lines as 10, then the graphs look like follows:



Example 3: Implicit variable coefficient ordinary differential equations:

Equation 2-4-4 changes as:

$$\begin{cases} \frac{dy_1}{dx} = \cos\left(y_1 - \sin(x + y_2) + \frac{dy_2}{dx} \cdot a\right) - \sin\left(\frac{2 \cdot x}{y_1} + y_2 - a\right) \\ \frac{dy_2}{dx} = -2 \cdot x \cdot y_2 + y_1 + \sin\left(x - \frac{dy_1}{dx} + a\right) \cdot y_1 \end{cases} \quad (2-4-7)$$

Here: a is variable parameter and is ranged in [0,1], extent of variation is 01, x is ranged in [0,2.5-a].

Auto2Fit code

```
LoopConstant a=[0:0.1:1];
Variable y1=1,y2=0.25, x=[0,2.5-a];
Plot y1[x], y2;
ODEFunction y1'=cos(y1-sin(x+y2)+y2'*a)-sin(2*x/y1+y2-a);
y2'=-2*x*y2+y1+sin(x-y1'+a)*y1;
```

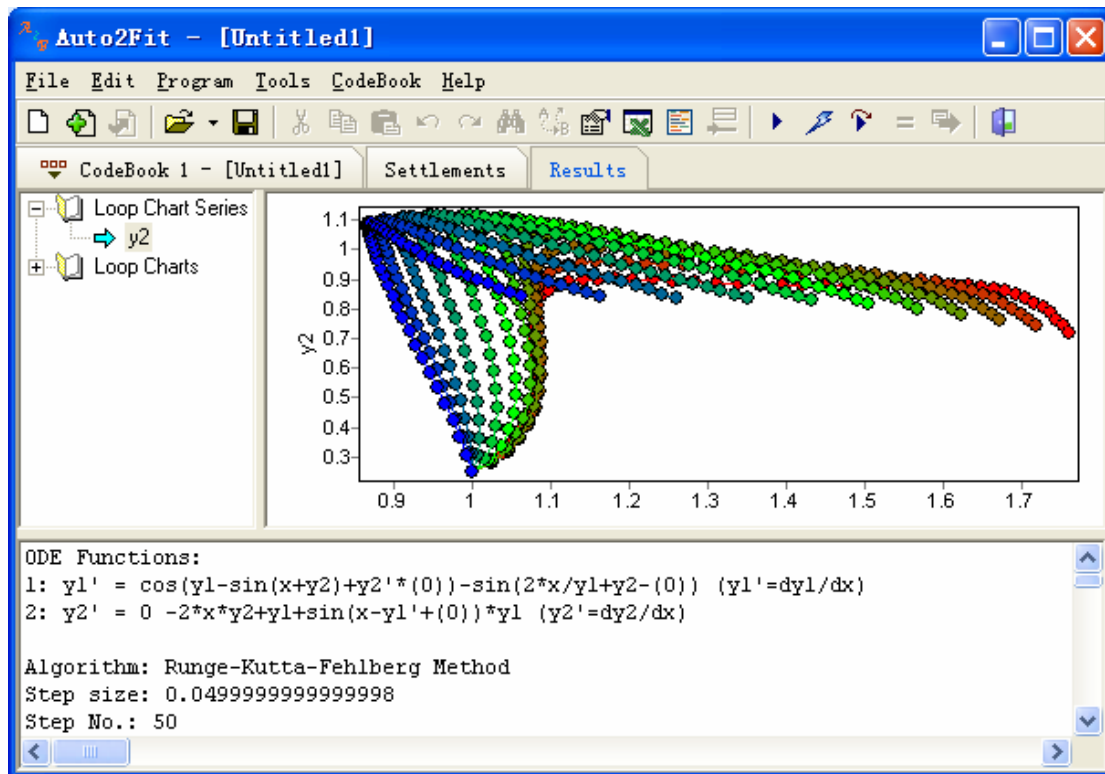


Figure 2-4-7 Result of implicit variable coefficient ODEs

2.4.5 Higher-order ordinary differential equations and system equations

Auto2Fit can handle any higher-order ordinary differential equations or system equations automatically and it doesn't need any man-made reduced-order. The expression must put the most high-end item in the left of equation along.

Second order ordinary differential equation example:

$$y'' - (1 - y^2) \cdot y' + x \cdot y = 0 \quad (2-4-8)$$

Initial value: when $x = 0$, $y' = -0.1$, $y = 0$; $x = [0, 14]$

Auto2Fit code:

```
Variable x = [0,14], y = 0, y' = -0.1;
ODEOptions = [SS=0.01,A=0,P=5];
Plot y[x], y';
ChartType = 3;
ODEFunction y'' = (1-y^2)*y'-y*x;
```

In the above code, "Plot y[x]" means y' as x axis.

Result:

x	y(x)	y'(x)	y''(x)
0	0	-0.1	-0.1
14	0.227406351611116	7.54066705119636	3.96702272139131

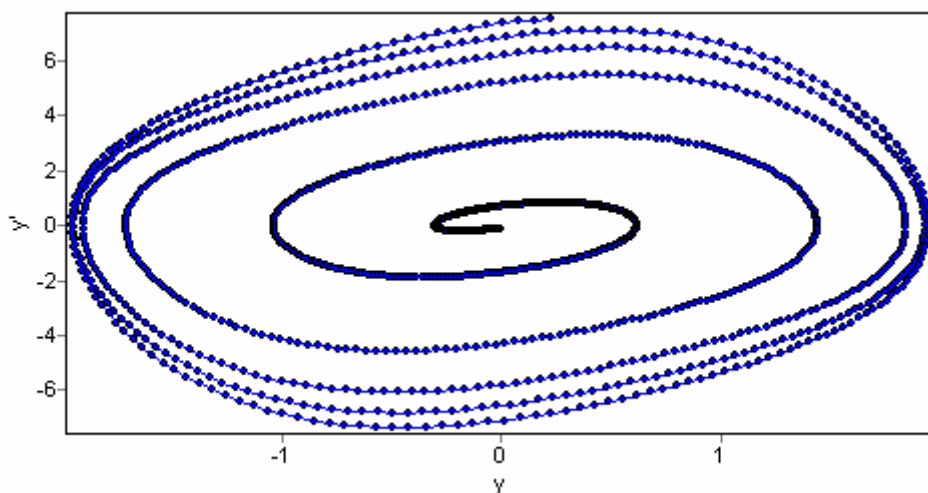


Figure 2-4-8 Result of second order ordinary differential equations

In the above Auto2Fit code, if change “Plot y'[x], y”; to “Plot y'[x], y'';”, a graph can be shown as:

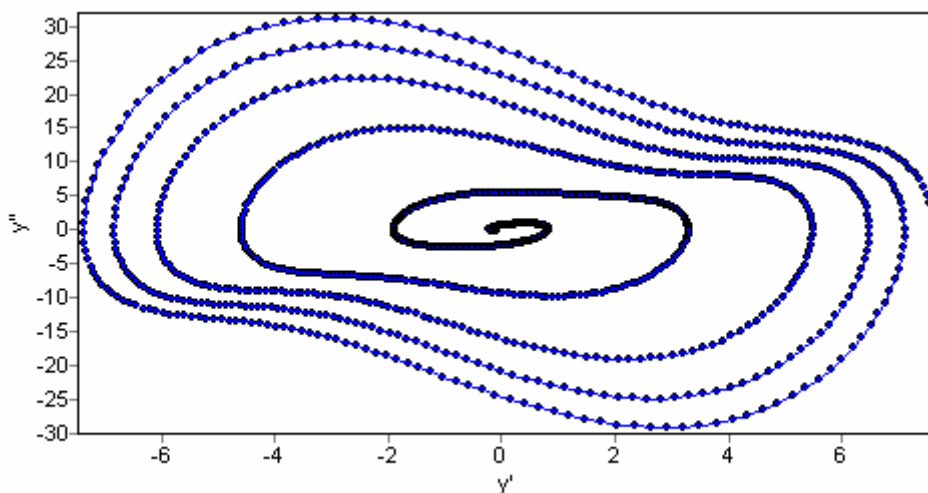


Figure 2-4-9 Result of second order ordinary differential equations

High-order ODE with variable coefficients

$$y'' = \frac{dy}{dt} = (1 - y^2 + a) \cdot y' - y \quad (2-4-9)$$

Variable range: $t = [0, 5]$, initial value: $t = 0$ 时 $y = a$, $y' = 1 + 4 \cdot a$, here a is variable coefficient and is ranged in $[0, 2]$, length of step is 0.01.

Auto2Fit code:

```
LoopConstant a=[0:0.01:1];
Plot y[x],y',y'';
```

Variable t=[0,5], y=a, y'=1+a*4;
 ODEFunction y'=(1-y^2+a)*y'-sin(t)*y;

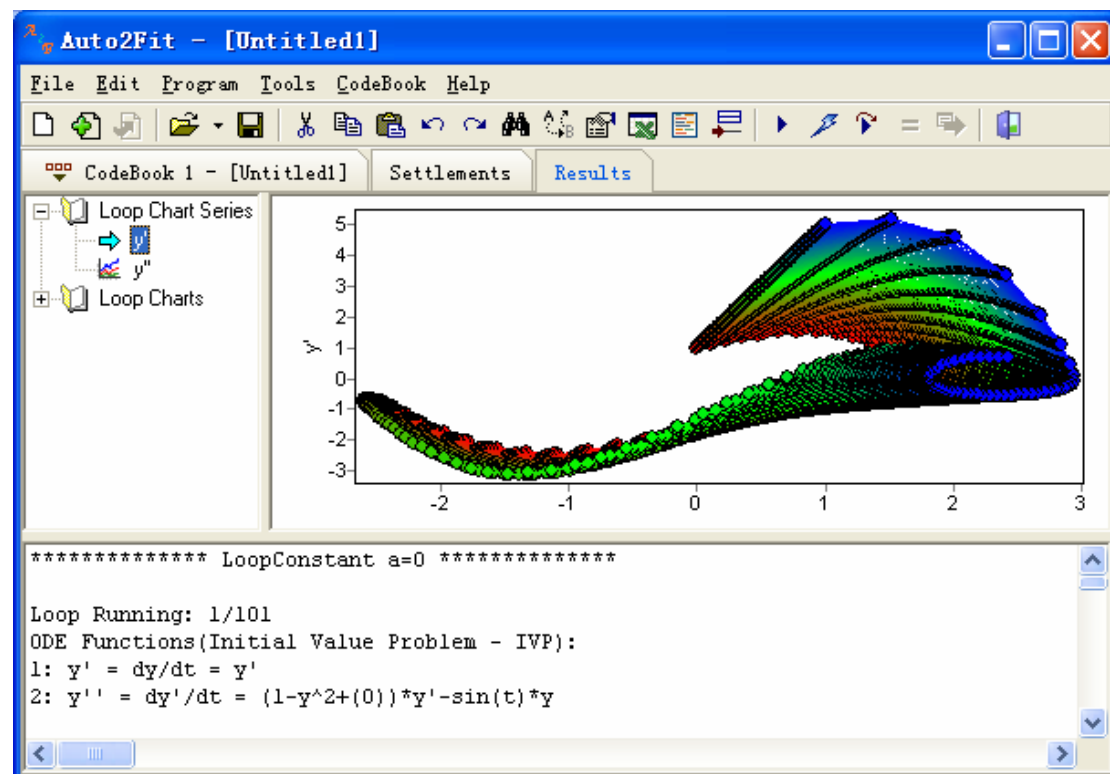
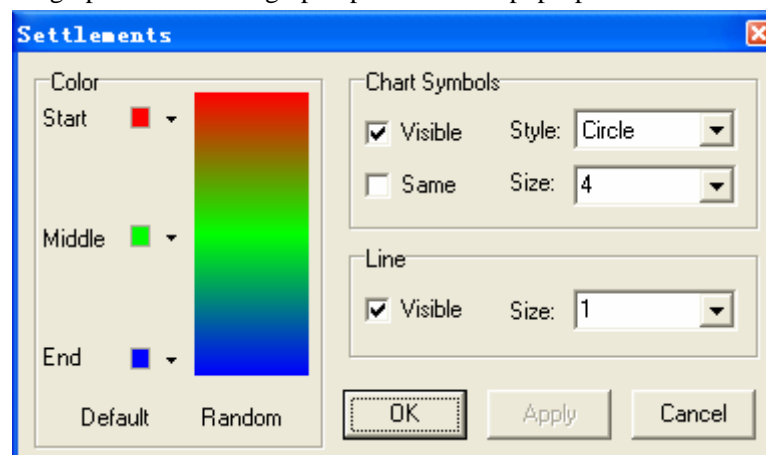
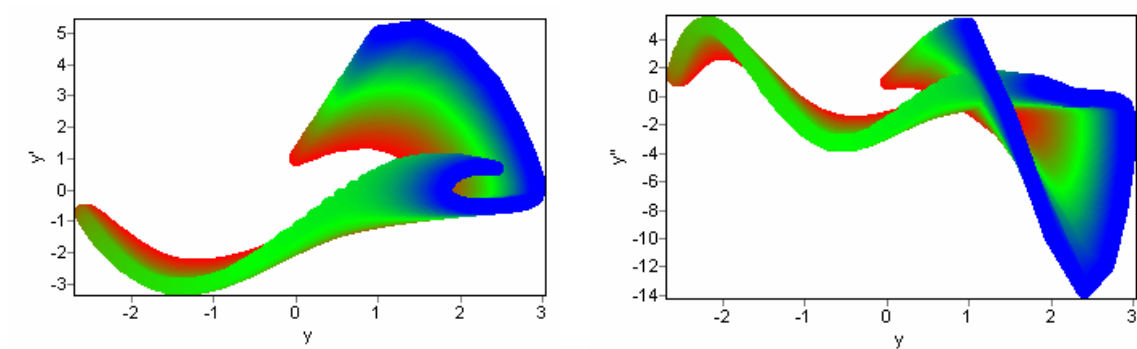


Figure 2-4-10 Result of high-order ODE with variable coefficients

Right click on the graph and choose “graph option...” from pop-up menu



Set the thickness of line from 1 to 10, a graph can be shown as follows:



2.4.6 Boundary value problems (BVP) of ODE

Boundary value problem is an important component of ordinary differential equations. It is solved usually by shooting method. As for linear problem, shooting method is easy to use and has high efficiency. But as for complicated nonlinear ordinary differential equation, shooting method is hard to meet the requirements.

Auto2Fit can recognize initial and boundary problem automatically based on its unique global optimization algorithm. Regarding boundary value problems of ODE, it can solve both linear and nonlinear problems. The solving method is almost the same as solve the problem of IVP of ODE.

Boundary value problem example1:

$$4 \cdot y'' + y \cdot y' = 2 \cdot x^3 + 16 \quad (2-4-10)$$

Initial value:, x is ranged in $[2, 3]$

The analytical solution for this case: $y = x^2 + \frac{8}{x}$

Auto2Fit code

```
Variable x=[2:0.1:3], y=[8,35/3];
Plot x[x],y',y''[y2];
ODEFunction y''=(2*x^3+16-y*y')/4;
```

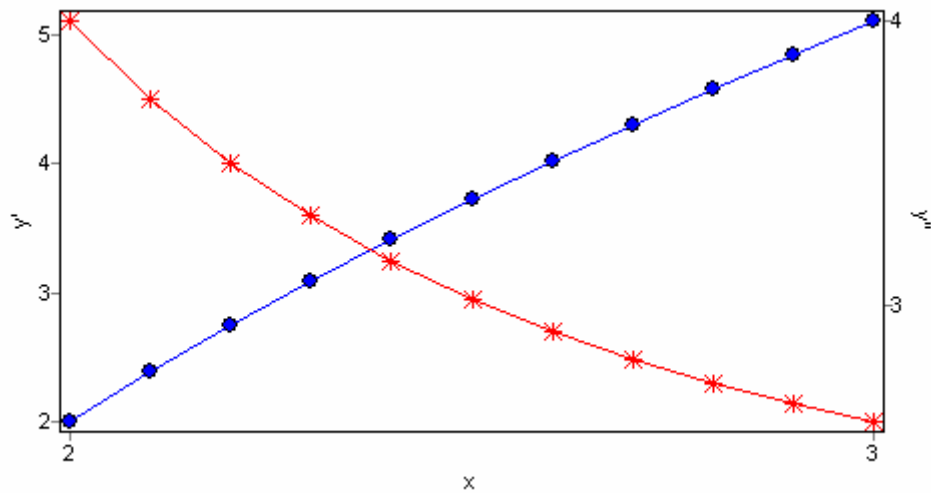


Figure 2-4-11 Result of boundary value problem

Output results:

Ordinary differential equation (Boundary value problem):				
1: $y' = dy/dx = y'$				
2: $y'' = dy'/dx = (2*x^3+16-y*y')/4$				
Objective function: 0				
Boundary value estimation:				
$y'(x=2)$: 1.99999993866552				
Algorithm: Runge-Kutta-Fehlberg Method				
Length of step: 0.1				
Number of step: 10				
Number of group: 5				
Result:				
x	y(x)	y'(x)	y'(x)	y''(x)
2	8	1.99999993866552	1.99999993866552	4.00000012266896
3	11.66666666666673	5.11111104586436	5.11111104586436	2.59259278289476

Comparison of results:

x	Analytical solution	Numerical result	Deviation
2	8	8	0
2.1	8.21952380370944	8.21952381	-6.290560961E-9
2.2	8.47636361925439	8.476363636	-1.674561112E-8
2.3	8.76826084598844	8.76826087	-2.401156074E-8
2.4	9.09333330705989	9.093333333	-2.59401105E-8
2.5	9.44999997396447	9.45	-2.603552929E-8
2.6	9.83692305339298	9.836923077	-2.360702034E-8
2.7	10.2529629436722	10.25296296	-1.632779956E-8
2.8	10.6971428433994	10.69714286	-1.660060001E-8
2.9	11.1686206824299	11.16862069	-7.570099214E-9
3	11.66666666666673	11.66666667	-3.33269945E-9

Boundary value problem example 2:

System equations as follows:

$$\begin{cases} y_1' = y_2 \\ y_2' = 20(y_3 - y_1) \\ y_3' = y_4 \\ y_4' = 10 + 8(y_3 - y_1) \end{cases} \quad (2-4-11)$$

Boundary value condition: when $x = 0$, $y_2 = 0$, $y_3 = 20$, when $x = 3$, $y_1 = 0$, $y_4 = 0$

Auto2Fit code

```
Variable y1=[,0], y2=[0,], y3=[20,], x=[0,3], y4=[,0];
ODEOptions = [SN=50,A=0,P=2];
Plot y1, y2, y3, y4;
ODEFunction y1' = y2;
           y2' = 20*(y3-y1);
           y3' = y4;
           y4' = 10+8*(y3-y1);
```

Result:

Ordinary differential equation (Boundary value problem):

- 1: $y_1' = dy_1/dx = y_2$
- 2: $y_2' = dy_2/dx = 20*(y_3 - y_1)$
- 3: $y_3' = dy_3/dx = y_4$
- 4: $y_4' = dy_4/dx = 10 + 8*(y_3 - y_1)$

Objective function: 1.39840021860598E-28

Boundary value estimation:

$y_1(x=0)$: 13.4694429556798

$y_4(x=0)$: -19.1527204392172

Algorithm: Runge-Kutta-Fehlberg Method

Length of step: 0.06

Number of step: 50

Number of group: 2

Result:

x	y1(x)	y2(x)	y3(x)	y4(x)	y1'(x)	y2'(x)	y3'(x)	y4'(x)
0.0000	13.4694	0.0000	20.0000	-19.1527	0.0000	130.6111	-19.1527	62.2445
3.0000	0.0000	-27.1182	2.1541	0.0000	-27.1182	43.0812	0.0000	27.2325

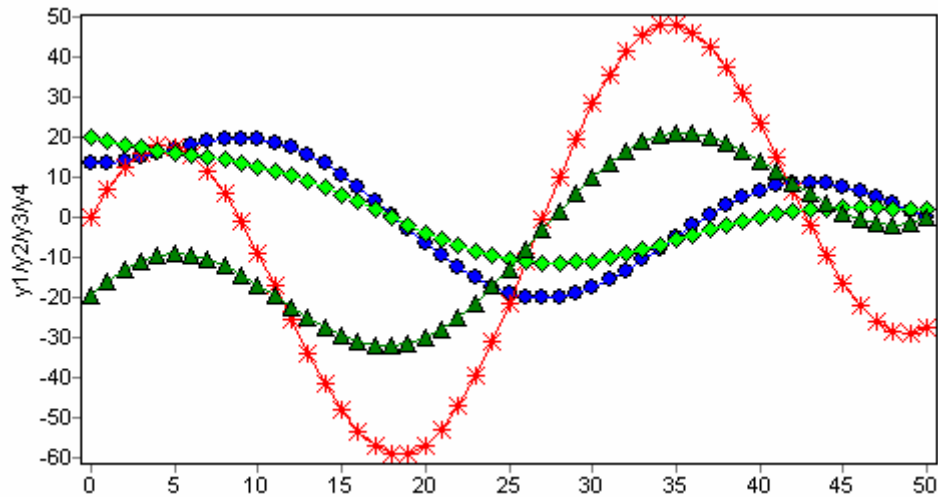


Figure 2-4-12 Result of boundary value problems for ordinary differential equations

Boundary value problem example 3:

$$y'' = y + y^3 \quad (2-4-12)$$

Boundary value condition: when $x = 0$, $y' = 1$ and when $x = 5$, $y' = -1$

Auto2Fit code:

```
StartRange = [-5,5];
ODEOptions = [SN=50,A=0,P=10];
Variable x = [0,5], y' = [1,-1];
Plot x[x], y[y2], y', y'';
ODEFunction y'' = y+y^3;
```

In above code, "Plot x[x], y[y2], y', y''" means x as x axis, y as right vertical axis, y', y'' as default left vertical axis.

Output result:

Ordinary differential equation (Boundary value problem):

1: $y'' = dy'/dx = y + y^3$

2: $y' = dy/dx = y'$

Objective function: 1.7241048121671E-27

Boundary value estimation:

$y(x=0)$: -0.86112059752689

Algorithm: Runge-Kutta-Fehlberg Method

Length of step: 0.1

Number of step: 50

Number of group: 10

Result:

x	$y'(x)$	$y(x)$	$y''(x)$	$y'(x)$
0	1	-0.86112059752689	-1.49966622053288	1
5	-0.999999144142996	-0.861120162291034	-1.4996648170779	

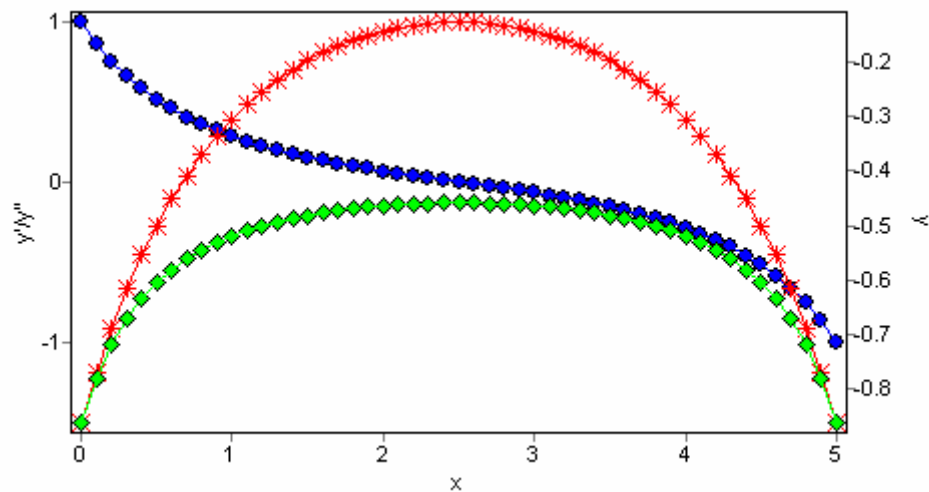


Figure 2-4-13 Result of BVP of ODE

Boundary value problem example 4:

$$y'' = (x+1) \cdot y + \exp(-x) \cdot (x^2 - x + 2) \quad (2-4-13)$$

Boundary value condition: when $x = 2$, $y' = y - \sin(x \cdot (y')^2)$ and when $x = 4$, $y' = -0.1$

```
Variable x = [2,4], y' = [y-sin(x*y'^2),-0.1];
Plot y[x], y', y'';
ChartType = 3;
ODEFunction y'' = (x+1)*y+exp(-x)*(x^2-x+2);
```

Result:

Ordinary differential equation (Boundary value problem):

1: $y'' = dy'/dx = (x+1)*y + \exp(-x)*(x^2-x+2)$

2: $y' = dy/dx = y'$

Objective function: 3.50846657969023E-29

Boundary value estimation:

$y'(x=2)$: -0.102184972000803

$y(x=2)$: -0.0813029529237473

Algorithm: Runge-Kutta-Fehlberg Method

Length of step: 0.02

Number of step: 100

Number of group: 30

Result:

x	$y'(x)$	$y(x)$	$y''(x)$
2	-0.102184972000803	-0.0813029529237473	0.297432274175209
5	-0.999999144142996	-0.861120162291034	-1.4996648170779

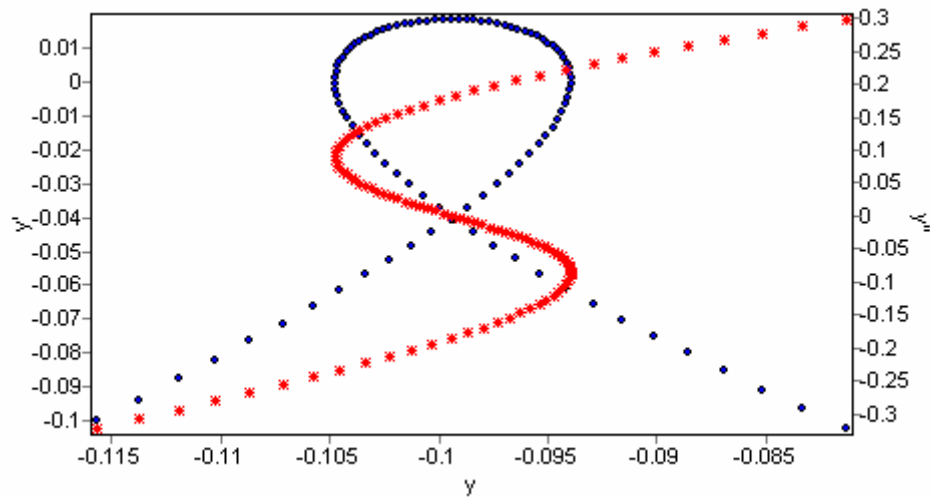


Figure 2-4-14 Result of BVP of ODE

Verify boundary value condition: when $x = 2$, $y' = y - \sin(x \cdot (y')^2)$

Here: $x = 2$, $y' = -0.102184972000803$, $y = -0.0813029529237473$

$$\begin{aligned}
 y - \sin(x \cdot (y')^2) &= -0.0813029529237473 - \sin(2 \cdot (-0.102184972000803)^2) \\
 &= -0.102184972 \\
 &= y'
 \end{aligned}$$

The above verification shows, the result meet the boundary condition well.

2.5 Other application

Lots of problems can be converted to optimization problem. Thus Auto2Fit can be used to solve many other problems.

2.5.1 Plot implicit function

Two dimension: x vs. y

$$\ln(3.5 \cdot x^y) + x - y^x - (\sin(y - x))^2 + 0.6 - \frac{(x + y)^{(0.1 \cdot x)}}{x} = 0 \quad (2-5-1)$$

Here, $x \in [1.5, 10]$

Three dimension: z vs. x, y

$$\ln(z) + \sin(x + y - z)^2 = (x - 10 - z)^3 + \cos(z) * (y - 100)^2 \quad (2-5-2)$$

Here, $x \in [9, 13]$, $y \in [97, 103]$

Auto2Fit code (two dimensions):

```
Parameters x[1.5,10], y;  
StepX = 100;  
PlotFunction ln(3.5*x^y)+x-y^x-(sin(y-x))^2+0.6-(x+y)^(0.1*x)/x = 0;
```

Auto2Fit code (three dimension)

```
Parameters x[9,13], y[97,103], z;  
StepX = 30;  
StepY = 30;  
PlotFunction ln(z)+sin(x+y-z)^2 = (x-10-z)^3 + cos(z)*(y-100)^2;
```

Result:

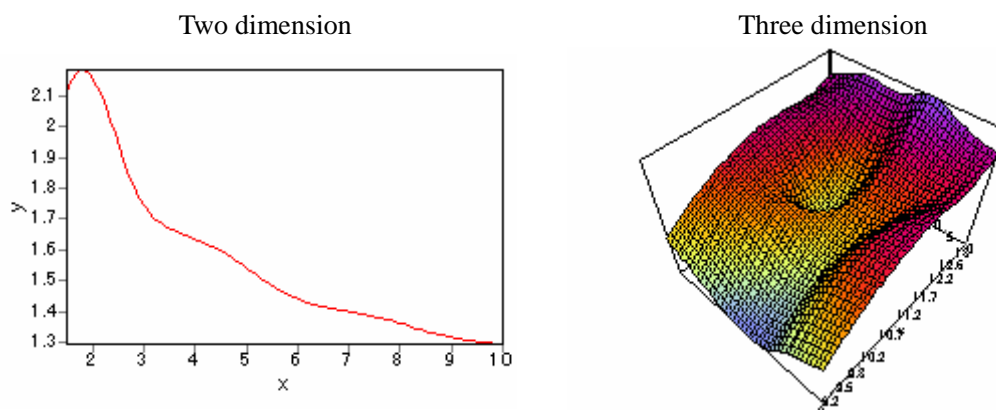


Figure 2-5-1 Result of plotting implicit function

2.5.2 Plot parameter function

Parameter function:

$$\begin{cases} x = \cos(t) \cdot \left(\exp(\cos(t)) - 2 \cdot \cos(4 \cdot t) - \sin\left(\frac{t}{12}\right)^5 \right) \\ y = \sin(t) \cdot \left(\exp(\cos(t)) - 2 \cdot \cos(4 \cdot t) - \sin\left(\frac{t}{12}\right)^5 \right) \end{cases} \quad (2-5-3)$$

Here $t = i \cdot (u + 20 \cdot i)$, u is ranged in $[0, 3\pi]$, $i=1..5$

Auto2Fit code:

```
ConstStr t = i*(u+20*i);
Parameters u[0,3*pi],x,y;
StepX = 1200;
PlotParaFunction For(i=1:5)(x=cos(t)*(exp(cos(t))-2*cos(4*t)-sin(t/12)^5),
y=sin(t)*(exp(cos(t))-2*cos(4*t)-sin(t/12)^5));
```

Result:

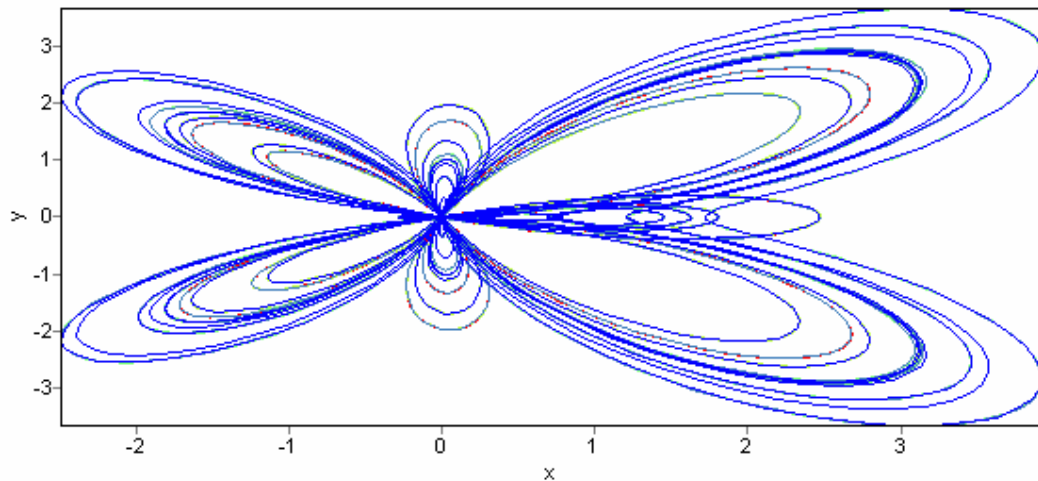


图 2-5-2 Result of plotting parameter function

2.5.3 Use as an advanced calculator

Auto2Fit can be used to calculate any format of expression. It supports special calculation operator, such as sum (Σ), quadrature (Π), integration, and other special functions, such as Gamma Function, Bessel Function in the mean time.

Try to calculate the values of the following expressions.

$$1: \quad f_1 = 5 \cdot \sin(\pi \cdot 6)^2 + \exp(6.45 + \ln(2.14))$$

$$2: f_2 = \int_0^{\pi} \left(x^{\sin(\Gamma(x))} + (\text{abs}(\sin(x)))^x \right) dx + \sum_{i=1}^{10} \left(\prod_{j=1}^i (0.1 \cdot (\ln(i \cdot j))^2) \right)$$

$$3: f_3 = \sqrt{\max(f_1^{0.1}, f_2)}$$

$$4: f_4 = \sum_{i=1}^3 \sum_{j=1}^i \sum_{m=1}^4 \prod_{n=1}^m \prod_{p=1}^{10} \left(f_3^{\frac{i+j+m+n+p}{1000}} \right)$$

Auto2Fit code:

```
f1 = 5*sin(pi*6)^2+exp(6.45+ln(2.14));
f2 = int(x^(sin(Gamma(x)))+(abs(sin(x)))^x, x=0:pi)+sum(i=1:10)(prod(j=1:i)(0.1*ln(i*j)^2));
f3 = sqrt(max(f1^0.1, f2));
f4 = Sum(i=1:3)(Sum(j=1:i)(Sum(m=1:4)(Prod(n=1:m)(Prod(p=1:3)(f3^((i+j+m+n+p)/1000))))));
```

Results: $f_1 = 1353.9829$, $f_2 = 48.669818$, $f_3 = 6.97637$, $f_4 = 28.2972$

2.5.4 Use script language

Auto2Fit support Basic and Pascal Script language, which can be used to control CodeBook, CodeBook spreadsheets and Auto2Fit spreadsheets directly.

Calculation within spreadsheets: see the following spreadsheet, put the sum of three columns A, B and C to column D.

	A	B	C	D	E	F	G
1	0	0.000	0.000				
2	0	0.000	0.000				
3	0	0.000	0.000				
4	0	0.000	0.000				
5	0.3	1.990	3.437				
6	0.6	3.075	6.513				
7	1.8	14.111	17.367				
8	4.4	14.858	31.839				
9	7.5	22.161	49.749				
10	5.35	15.969	37.101				
11	3.2	9.648	23.156				
12	2.35	7.015	17.005				
13	1.5	4.432	10.854				
14	0.9	2.876	6.513				
15	0.3	1.031	2.171				
16							
17							

Figure 2.5.3 Spreadsheets data

Auto2Fit code:

```
StartScript [Pascal];
var i: integer;
Begin
```

```

With Sheet1 do
  for i := 0 to 14 do
    Doubles[3,i] := doubles[0,i]+doubles[1,i]+doubles[2,i];
  end;
EndScript;

```

Results:

	A	B	C	D	E	F	G
1	0	0.000	0.000	0			
2	0	0.000	0.000	0			
3	0	0.000	0.000	0			
4	0	0.000	0.000	0			
5	0.3	1.990	3.437	5.727			
6	0.6	3.075	6.513	10.188			
7	1.8	14.111	17.367	33.278			
8	4.4	14.858	31.839	51.097			
9	7.5	22.161	49.749	79.41			
10	5.35	15.969	37.101	58.42			
11	3.2	9.648	23.156	36.004			
12	2.35	7.015	17.005	26.37			
13	1.5	4.432	10.854	16.786			
14	0.9	2.876	6.513	10.289			
15	0.3	1.031	2.171	3.502			
16							
17							

Figure 2.5.4 Result of using script to control spreadsheets

Controlling CodeBook spreadsheets: it is almost the same as the above example but only the name is different

	A	B	C	D	E	F	G
1	0	0.000	0.000				
2	0	0.000	0.000				
3	0	0.000	0.000				
4	0	0.000	0.000				
5	0.3	1.990	3.437				
6							

Figure 2.5.5 Screen shot of pre-calculation

Auto2Fit code:

```
StartScript [Pascal];
var i: integer;
Begin
  With CodeSheet1 do
    for i := 0 to 14 do
      Doubles[3,i] := doubles[0,i]+doubles[1,i]+doubles[2,i];
    End;
  EndScript;
```

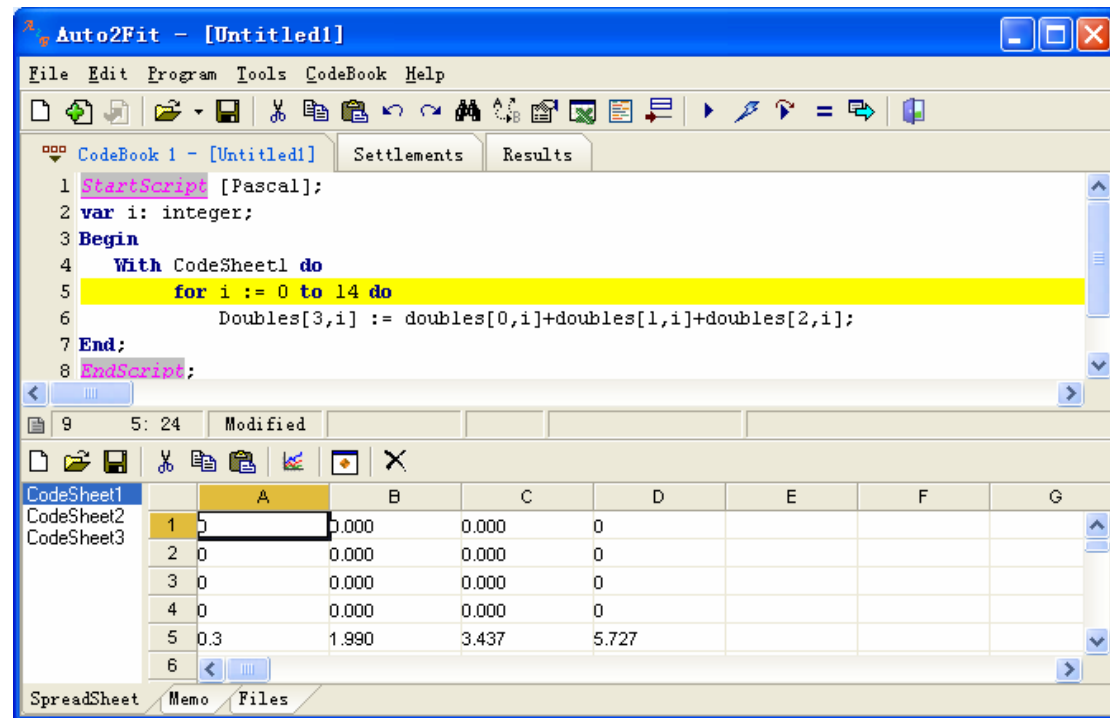


Figure 2.5.6 Screen shot of after calculation

Controlling CodeBook: add content in CodeBook

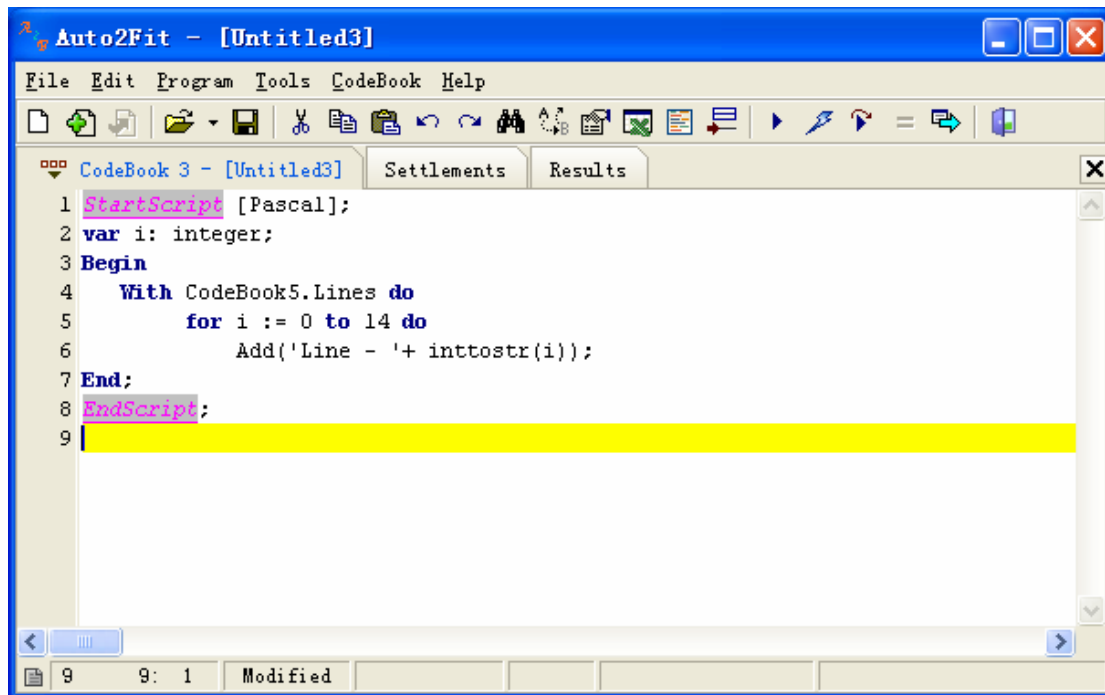


Figure 2.5.7 Screen shot of pre-calculation

Auto2Fit code:

```
StartScript [Pascal];
var i: integer;
Begin
  With CodeBook5.Lines do
    for i := 0 to 14 do
      Add('Line - '+ inttostr(i));
End;
EndScript;
```

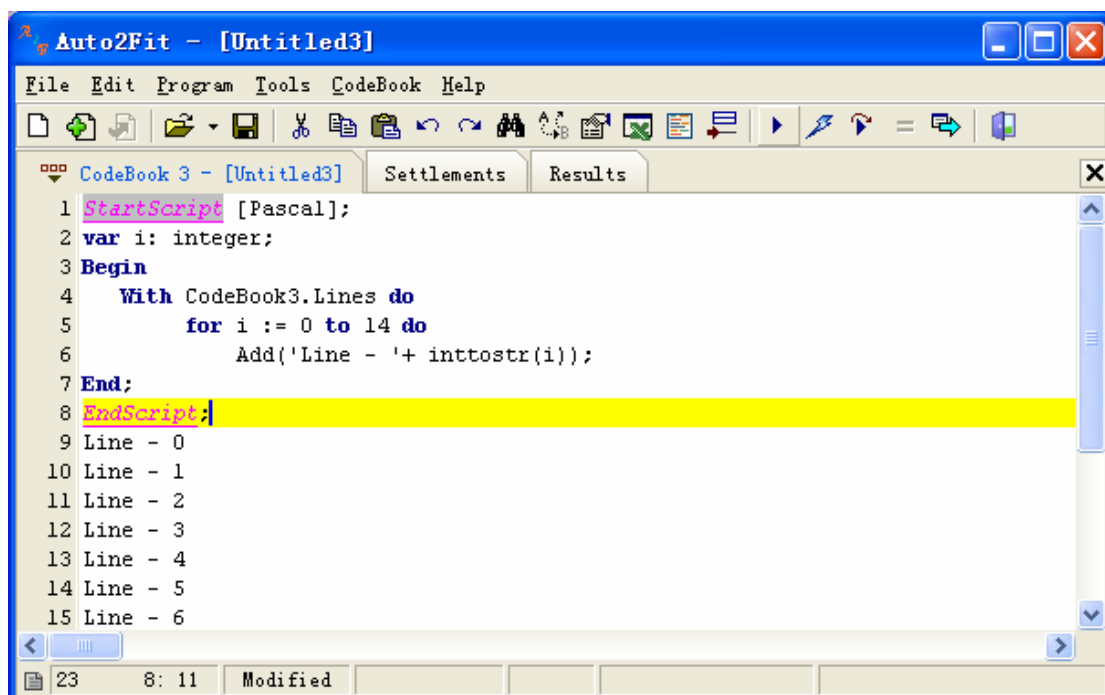


Figure 2.5.7 Screen shot of after calculation

2.5.5 The key word of “PassParameter”

“PassParameter” can be understood as “pass parameter”, which can pass and return the value related to parameters.

Example: Constrained function optimization problems: when doing optimization calculation, it will also calculate and return the value of $x \cdot y$

Auto2Fit code

```
Parameter x[0,100],y[0,100];
PassParameter x*y;
MinFunction 9-x-y;
(x-3)^2+(y-2)^2<=16;
x*y<=14;
```

Result:

```
Number of iteration: 167
Computation time (Hour:Minute:Second:Millisecond): 00:00:00:78
Reason of computation stops: meet the criteria of convergence
Optimization algorithm: Simplex Method (SM) + Universal Global Optimization (UGO)
Function expression: 9-x-y
Value of objective function (Minimum): 0
x: 7
y: 2

PassParameter:
x*y: 14

Constrained functions:
1: (x-3)^2+(y-2)^2-(16) = 0
2: x*y-(14) = 0

===== End of Computation =====
```

Example: Nonlinear constrained curve fit

Data for curve fit

x	-0.08,-0.065,-0.05,-0.03,-0.015,0.015,0.03,0.05,0.065,0.08,0
y	20.26008,19.72613,19.501619,18.72662,18.58769,18.592199,18.88372,19.5453,19.88743,20.9914,18.12336

Curve fit equation

$$y = a \cdot (x - d)^4 + b \cdot (x - d)^2 + c \quad (2-5-4)$$

here: x,y: independent variable and dependent variable; a,b,c,d: parameter to be solved.

Constraints:

1) must larger than 0;

$$2) \text{Max}(\text{abs}(y_i - y'_i)) \leq 0.25, \quad i = 1.11 \quad (2-5-6)$$

Auto2Fit code

```

RowDataSet;
  x=-0.08,-0.065,-0.05,-0.03,-0.015,0.015,0.03,0.05,0.065,0.08,0;
  y=20.26008,19.72613,19.501619,18.72662,18.58769,18.592199,18.88372,19.5453,19.88743,20.9914,18.12336;
EndRowDataSet;
PassParameter CalY(11), PA;
Parameter a=[,0],b,c,d;
Plot y, CalY;
StartProgram [Pascal];
Procedure MainModel;
var i: integer;
    temd, temy, Maxy: double;
Begin
  temd := 0;
  for i := 1 to 11 do begin
    temy := a*(x[i]-d)^4+b*(x[i]-d)^2+c;
    temd := temd + sqr(temy - y[i]);
    if i = 1 then MaxY := abs(y[i]-temy)
    else MaxY := Max(abs(y[i]-temy), MaxY);
    CalY[i] := temy;
  end;
  PA := MaxY;
  ObjectiveResult := temd;
  ConstrainedResult := PA <= 0.25;
End;
EndProgram;

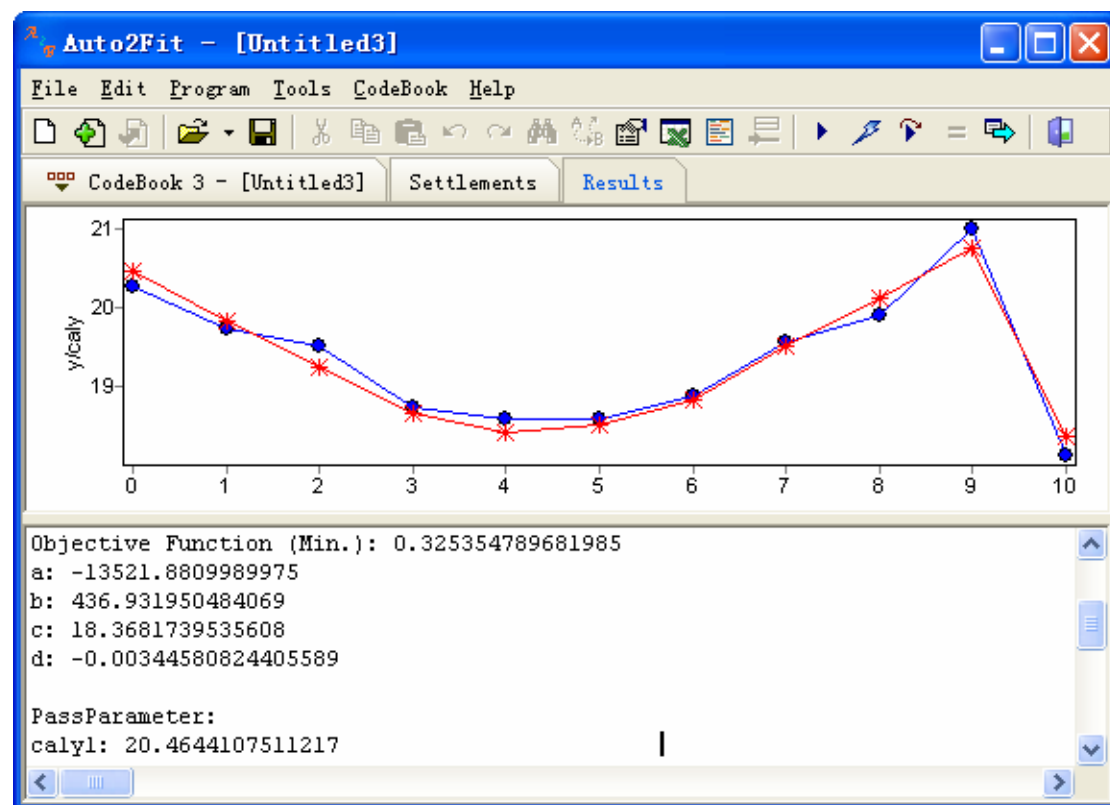
```

Comparison of results:

Nonconstrained (Remove ConstrainedResult := PA <= 0.25;)	Constrained
Number of iteration: 283 Computation time (Hour:Minute:Second:Millisecond) 00:00:00:516 Reason of computation stops: meet the criteria of convergence Optimization algorithm: Simplex Method (SM) + Universal Global Optimization (UGO) Value of objective function (Minimum): 0.309952381552963 a: -6877.78414257241 b: 385.898376776761 c: 18.4260680489247 d: -0.00393418836172759 PassParameter: caly1: 20.4286253611321 caly2: 19.7694556336994 caly3: 19.2139955867598 caly4: 18.6850827208528 caly5: 18.4732190200065 caly6: 18.5635299978685 caly7: 18.8613211998476 caly8: 19.4904089461543 caly9: 20.104521527912 caly10: 20.8033487243363 caly11: 18.4320392738443 pa: 0.308679273844348	Number of iteration: 814 Computation time (Hour:Minute:Second:Millisecond) 00:00:01:219 Reason of computation stops: meet the criteria of convergence Optimization algorithm: Simplex Method (SM) + Universal Global Optimization (UGO) Value of objective function (Minimum): 0.325354789681984 a: -13521.8798280647 b: 436.931945824039 c: 18.3681739539736 d: -0.00344580812519686 PassParameter: caly1: 20.4644107695081 caly2: 19.8295546986496 caly3: 19.251619 caly4: 18.6695425408538 caly5: 18.4262630967072 caly6: 18.5152736946542 caly7: 18.8400155406003 caly8: 19.5059206320813 caly9: 20.118352178724 caly10: 20.7549952476031 caly11: 18.37336 pa: 0.25

===== End of Computation =====

===== End of Computation =====



2.5.6 The key word of “SubDivision”

“SubDivision” can let the code in subdivision section use the parameters, constant and objective function values form the main “NewDivision” section.

Auto2Fit code

```
Parameter x1[0,100],y1[0,100];
MinFunction 9-x1-y1;
    (x1-3)^2+(y1-2)^2<=16;
    x1*y1<=14;

SubDivision;
Parameters p(1:5)[-32.768, 32.768];
MinFunction -x1*exp(-y1*sqrt(1/5*sum(i=5,p)(p^2)))-exp(1/5*sum(i=5,p)(cos(6*(p))))+x1+exp(ObjFunction);
```

Result:

===== Result =====

Number of iteration: 37
Computation time (Hour:Minute:Second:Millisecond): 00:00:00:375
Reason of computation stops: meet the criteria of convergence
Optimization algorithm: Standard Differential Evolution algorithm
Function expression: 9-x1-y1
Value of objective function (Minimum): 0
x1: 7
y1: 2

Constrained Functions:

```

1: (x1-3)^2+(y1-2)^2-(16) = 0
2: x1*y1-(14) = 0

===== End of Computation =====

*****
===== End =====

Number of iteration: 361
Computation time (Hour:Minute:Second:Millisecond): 00:00:01:781
Reason of computation stops: meet the criteria of convergence
Optimization algorithm: Standard Differential Evolution algorithm
Function expression:
-7*exp(-2*sqrt(1/5*((p1^2)+(p2^2)+(p3^2)+(p4^2)+(p5^2))))-exp(1/5*((cos(6*(p1)))+(cos(6*(p2)))+(cos(6
*(p3)))+(cos(6*(p4)))+(cos(6*(p5))))) + 7 + exp(0)
Value of objective function (Minimum): -1.71828182845904
p1: -1.23257171502356E-16
p2: 7.40257379257929E-17
p3: -4.29348910045455E-17
p4: -3.95583172994915E-17
p5: -7.79891019495037E-17

===== End of Computation =====

```

2.5.7 The key word of “PenaltyFactor”

“PenaltyFactor” is penalty coefficient used to solve the constrained optimization problem. The default value is 1000000. But sometimes this value is not the best choice and can be adjusted using “PenaltyFactor” in this case.

Example 1:

$$\text{Min. } 4 - (x_1 + 0.25)^2 + (x_1 + 0.25)^3 + (x_1 + 0.25)^4 - x_2 \quad (2-5-7)$$

$$\text{S.T. } x_1^2 + x_2^2 = 9$$

Auto2Fit code

```

MinFunction 4-(x1+0.25)^2+(x1+0.25)^3+(x1+0.25)^4-x2;
x1^2+x2^2=3^2;

```

Execute the above code; the probability of getting the optimal solution is not high. The default value of penalty coefficient in this case is 1000000. If change this coefficient to 10, the code is as follows:

```

PenaltyFactor = 10;
MinFunction 4-(x1+0.25)^2+(x1+0.25)^3+(x1+0.25)^4-x2;
x1^2+x2^2=3^2;

```

It is easy to find the optimal solution whatever algorithm is used:

```

Optimization algorithm: standard differential evolution algorithm
Function expression: 4-(x1+0.25)^2+(x1+0.25)^3+(x1+0.25)^4-x2

```

Value of objective function (Minimum): 0.244013991092436

x1: -1.3503280560084

x2: 2.67892033124477

Constrained functions:

1: $x_1^2 + x_2^2 - (3^2) = 0$

Example 2:

$$\text{Max. } 0.5 \cdot (x_1^2 + x_2^2) + \frac{0.5 \cdot x_3}{1 + x_1 + x_2} \quad (2-5-8)$$

$$\text{S.T. } x_1^2 + x_2^2 + x_3^2 = 1$$

Here X is larger than 0.

Auto2Fit code

```
ParameterDomain = [0,];  
Maxfunction 0.5*(x1^2+x2^2)+0.5*x3/(1+x1+x2);  
x1^2+x2^2+x3^2=1;
```

Using the above code it is also difficult to find the optimal solution. Adjust the coefficient of penalty function as follows:

```
ParameterDomain = [0,];  
PenaltyFactor = 5;  
Maxfunction 0.5*(x1^2+x2^2)+0.5*x3/(1+x1+x2);  
x1^2+x2^2+x3^2=1;
```

It is easy to find the optimal solution whatever algorithm is used:

Optimization algorithm: standard differential evolution algorithm

Function expression: $0.5 \cdot (x_1^2 + x_2^2) + 0.5 \cdot x_3 / (1 + x_1 + x_2)$

Value of objective function (Maximum): 0.532343902770902

x1: 0.964560753739086

x2: 7.41716656971252E-16

x3: 0.263860857927594

Constrained functions:

1: $x_1^2 + x_2^2 + x_3^2 - (1) = 0$

Chapter 3 Programming mode of Auto2Fit

The quick mode of Auto2Fit is intuitive, simple, clear and easy to grasp and can solve most of the optimization problems. But as for some complicated problems, e.g., when objective function or constrained functions cannot be expressed by simple expression and they only can be expressed by complicated logical determination and mathematical operations, quick model is hard to use, in this case, the programming mode of Auto2Fit can be employed to solve the problem

Auto2Fit support Basic and Pascal directly. In theory, programming model can handle all the problems which quick mode can handle. The main key words in programming mode are:

- 1) StartProgram: Define the start line of programming mode
 - “StartProgram [Basic]” means to use Basic
 - “StartProgram [Pascal]” or “StartProgram” means to use Pascal
- 2) EndProgram: Define the finish line of programming mode
Between “StartProgram” and “EndProgram”, standard Delphi/Pascal or Basic can be used.
- 3) ObjectiveResult: Define objective function, which can be used only once.
- 4) ConstrainedResult: Define constrained functions, which may be used several times.

3.1 Constrained function optimization problems

$$\text{Min. } 10 \cdot x_1 + 9 \cdot x_2 + 8 \cdot x_3 + 7 \cdot x_4 \cdot \sin(x_1 + x_2 + x_3) \quad (3-1)$$

$$\text{S.T. } \begin{cases} (3 \cdot x_2 + 2 \cdot x_4 \cdot \cos(x_1 + x_2 + x_3 + x_4))^2 \leq 90 \\ x_1 + x_2 \geq -30 \\ x_3 + x_4 \geq 30 \\ 3 \cdot x_1 + 2 \cdot x_3 \leq 120 \end{cases}$$

Parameters' ranges are all in [-100,100]

Auto2Fit code in quick model:

```
Parameter x(4)=[-100,100];
MinFunction 10*x1+9*x2+8*x3+7*x4*sin(x1+x2+x3);
(3*x2+2*x4*cos(x1+x2+x3+x4))^2<=90;
x1+x2>=-30;
x3+x4>=30;
3*x1+2*x3<=120;
```

Auto2Fit Basic code in programming mode:

```
Parameter x(4)=[-100,100];
Minimum;
StartProgram [Basic];
Sub MainModel
```

```

ObjectiveResult = 10*x1+9*x2+8*x3+7*x4*sin(x1+x2+x3)
ConstrainedResult = (3*x2+2*x4*cos(x1+x2+x3+x4))^2<=90
ConstrainedResult = x1+x2>=-30
ConstrainedResult = x3+x4>=30
ConstrainedResult = 3*x1+2*x3<=120
End Sub
EndProgram;

```

Auto2Fit Pascal code in programming mode:

```

Parameter x(4)=[-100,100];
Minimum;
StartProgram [Pascal];
Procedure MainModel;
Begin
  ObjectiveResult := 10*x1+9*x2+8*x3+7*x4*sin(x1+x2+x3);
  ConstrainedResult := sqrt(3*x2+2*x4*cos(x1+x2+x3+x4))<=90;
  ConstrainedResult := x1+x2>=-30;
  ConstrainedResult := x3+x4>=30;
  ConstrainedResult := 3*x1+2*x3<=120;
End;
EndProgram;

```

The above three sections of codes have the same results: Min. = -1589.02777245744, X = [-98.78263, 68.78263, -65.84115, 99.09896].

3.2 Calibration of time series model

This case is to study the relationships between E. coli (z) and precipitation (x), discharge (y). Table 3.1 shows the data of three measurements. There are nine similar data. Because E. Coli is not only concerning with precipitation and discharge, but also is affected by its initial value. The adopted mathematical model is as following equation, here, from p1 to p7 are parameters to be determined, t represents time.

$$z_t = \frac{p_1 + p_2 \cdot x_t + p_3 \cdot y_t + p_4 \cdot z_{t-1}}{1 + p_5 \cdot x_t + p_6 \cdot y_t + p_7 \cdot z_{t-1}} \quad (3-2)$$

Example of original data

Rainfall x	3, 4.5, 5.5, 0.5, 0.5, 1, 0.5, 1.5, 2.5, 0, 0, 0
Runoff y	1.23, 2.17, 3.88, 5.90, 4.81, 3.07, 1.99, 1.36, 1.46, 2.11, 1.89, 1.35
E.coli z	43453.54, 28745.68, 16267.61, 9466.555, 8041.477, 6047.688, 4509.077, 3817.827, 2888.571, 2008.761, 2293.048, 2710.212

In this case, the first line in every data file is considered as their initial value. Use key word “VarConstant” to describe initial value of E. coli in every measure process.

Auto2Fit code:

```
Parameter p(1:7);
```



```

Variable x, y, z;
VarConstant z0 = [27177.83, 25288.04, 7751.078, 11028.05, 10725.02, 34615.46,
22479.53, 18309.19, 44856.28];
StartProgram [Pascal];
Procedure MainModel;
var i      :integer;
    Temz : Double;
begin
    Temz := z0;
    for i := 0 to Datalength - 1 do begin
        z[i] := (p1+p2*x[i]+p3*y[i]+p4*Temz)/(1+p5*x[i]+p6*y[i]+p7*Temz);
        Temz := z[i];
    end;
end;
EndProgram;
//x      y      z
Data "2000-03-29"; //file 1
//1  1.0100  27177.83
1  1.2500  22469.71
1  1.5000  18596.52
0.5  1.6000  15013.49
1.5  1.4400  13300.36

```

Detailed code and data can be referred to the attached example to Auto2Fit software package "Examples\Auto Calibration\Time Series.mff".

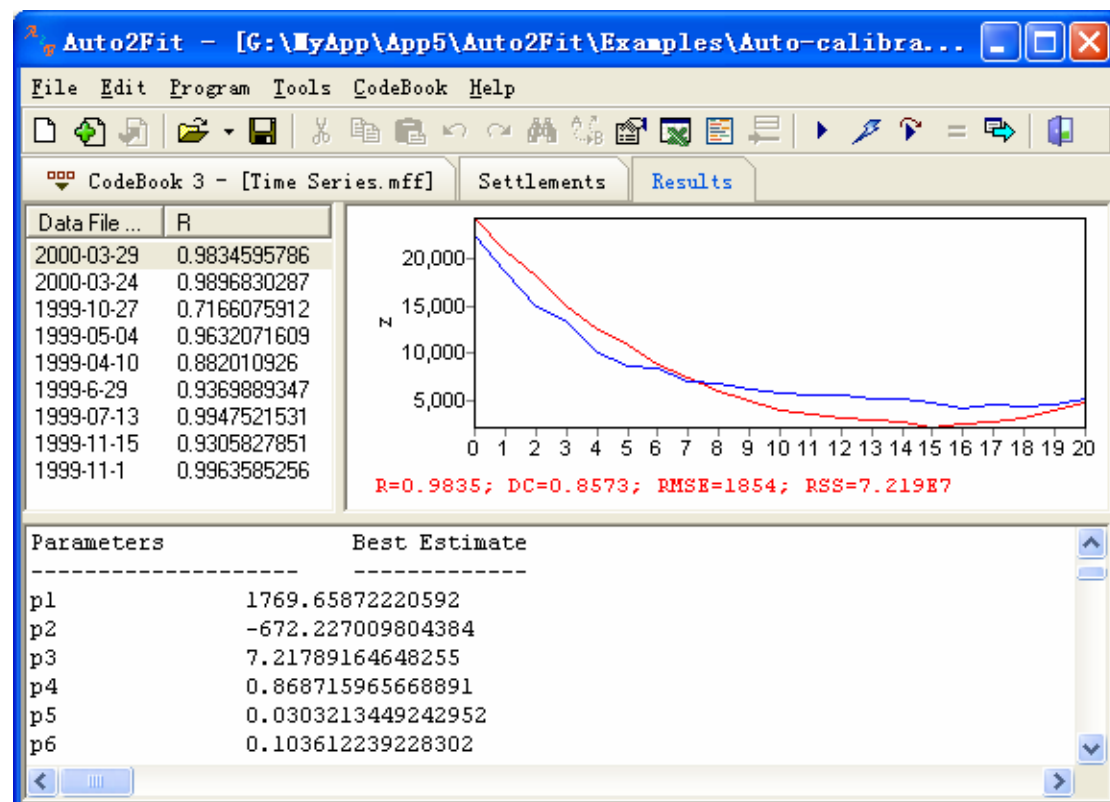


Figure 3-1 Auto-calibration result of time series model

3.3 Transportation problem

There are three railway boxcars, the maximum allowances of weight is 7, 8 and 9 tons respectively. Now there are 16 boxes need to be carried by these three boxcars. The following table lists the weights of these boxes (unit is ton). The problem is how to arrange the boxes to different boxcars to meet the requirements: the boxcar with largest weight allowance carries the boxes with lightest load and the actual load of every boxcar does not exceed the maximum weight allowance.

This category of problem cannot be solved using quick model but only programming mode.

Basic code:

```
Algorithm = SM2[30];
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
IntParameter p(16)=[1,3];
PassParameter v(3);
Minimum;
StartProgram [Basic];
Sub MainModel
dim i as integer
dim ww(3) as double
  for i = 1 to 3
    ww(i) = 0
  next
  for i = 1 to 16
    if p(i) = 1 then
      ww(1) = ww(1) + w(i)
    elseif p(i) = 2 then
      ww(2) = ww(2) + w(i)
    elseif p(i) = 3 then
      ww(3) = ww(3) + w(i)
    end if
  next
  for i = 1 to 3
    v(i) = ww(i)
  next
  ObjectiveResult = ww(3)
  ConstrainedResult = for(i=1:3)(ww(i) <= c(i))
End Sub
EndProgram;
```

Pascal code:

```
Algorithm = SM2[30];
Constant w=[3.4,0.6,0.8,1.7,1.6,0.5,1.3,2.1,2.5,3.1,1.4,1.3,3.3,0.9,2.5,2.5];
Constant c=[7,9,19];
IntParameter p(16)=[1,3];
PassParameter v(3);
Minimum;
StartProgram [Pascal];
Procedure MainModel;
var i: integer;
    ww: array[1..3] of double;
```

```

Begin
  for i := 1 to 3 do
    ww[i] := 0;
  for i := 1 to 16 do
    if p[i] = 1 then ww[1] := ww[1] + w[i]
    else if p[i] = 2 then ww[2] := ww[2] + w[i]
    else if p[i] = 3 then ww[3] := ww[3] + w[i];
  for i := 1 to 3 do
    v[i] := ww[i];
  ObjectiveResult := ww[3];
  ConstrainedResult := for(i=1:3)(ww[i] <= c[i]);
End;
EndProgram;

```

Results:

```

===== Results =====

Number of iteration: 23
Computation time (Hour:Minute:Second:Millisecond): 00:00:00:624
Reason of computation stops: meet the criteria of convergence
Optimization algorithm: Simplex Method (SM) + Universal Global Optimization (UGO)
Value of objective function (Minimum): 13.5
p1: 1
p2: 1
p3: 1
p4: 2
p5: 3
p6: 2
p7: 1
p8: 2
p9: 3
p10: 3
p11: 2
p12: 3
p13: 2
p14: 1
p15: 3
p16: 3

PassParameter:
v1: 7
v2: 9
v3: 13.5

===== End of Computation =====

```

3.4 Allocation of water resources problem

Water resources to be allocated have seven units for three users. The economic benefits of every user when they get different water resources can be found in table 3-1. The problem is to make a water resources allocation plan with highest economic benefits.

Table 3-1 Table of economic benefits

Water usage unit \ User	0	1	2	3	4	5	6	7
1	0	5	15	40	80	90	95	100
2	0	5	15	40	60	70	73	75
3	0	4	26	40	45	50	51	53

Basic code:

```

Algorithm = SM2[3];
Constant Benefit(1:3, 0:7)=[0,5,15,40,80,90,95,100,
                             0,5,15,40,60,70,73,75,
                             0,4,26,40,45,50,51,53];
Parameter p(1:3)=[0,7,0];
Maximum;
StartProgram [Basic];
Sub MainModel
    FunctionResult = Benefit(3,p(3)) + Benefit(1,p(1)) +Benefit(2,p(2))
    ConstrainedResult = p(3)+p(1)+p(2) = 7
End Sub
EndProgram;
```

Pascal code:

```

Algorithm = SM2[3];
Constant Benefit(1:3, 0:7)=[0,5,15,40,80,90,95,100,
                             0,5,15,40,60,70,73,75,
                             0,4,26,40,45,50,51,53];
IntParameter p(1:3)=[0,7];
Maximum;
StartProgram [Pascal];
Procedure MainModel;
Begin
    FunctionResult := Benefit[3,p[3]] + Benefit[1,p[1]] +Benefit[2,p[2]];
    ConstrainedResult := p[3]+p[1]+p[2] = 7;
End;
EndProgram;
```

Result:

```

===== Result =====

Number of iteration: 7
Computation time (Hour:Minute:Second:Millisecond): 00:00:00:47
Reason of computation stops: meet the criteria of convergence
Optimization algorithm: Simplex Method (SM) + Universal Global Optimization (UGO)
Value of objective function (Maximum): 120
p1: 4
p2: 3
p3: 0

===== End of Computation =====
```

3.5 Auto-calibration of Tank model

Because of its simple structure, easy-to-understand, good performance of application, Tank model becomes the most used rainfall-runoff model. Figure 3-2 is a three layers Tank model. Rainfall is input and discharge, Q , is output:

$$Q = Q_1 + Q_3 + Q_5 \quad (3-3)$$

Parameter in use

- Initial storage: L_1, L_2, L_3 (total 3), unit: mm
- Storage capacity: S_1, S_2, S_3 (total 3), unit: mm
- Discharge coefficient: K_1, K_2, K_3, K_4, K_5 (Total 5)

If use 8 times of rainfall-runoff to do the calibration of model, then the initial storage, L , changes to $3 \times 8 = 24$. The final total parameter to be solved is: $24 + 3 + 5 = 32$.

In the following code, special parameter “initial storage” is defined using key word “VapParameter”.

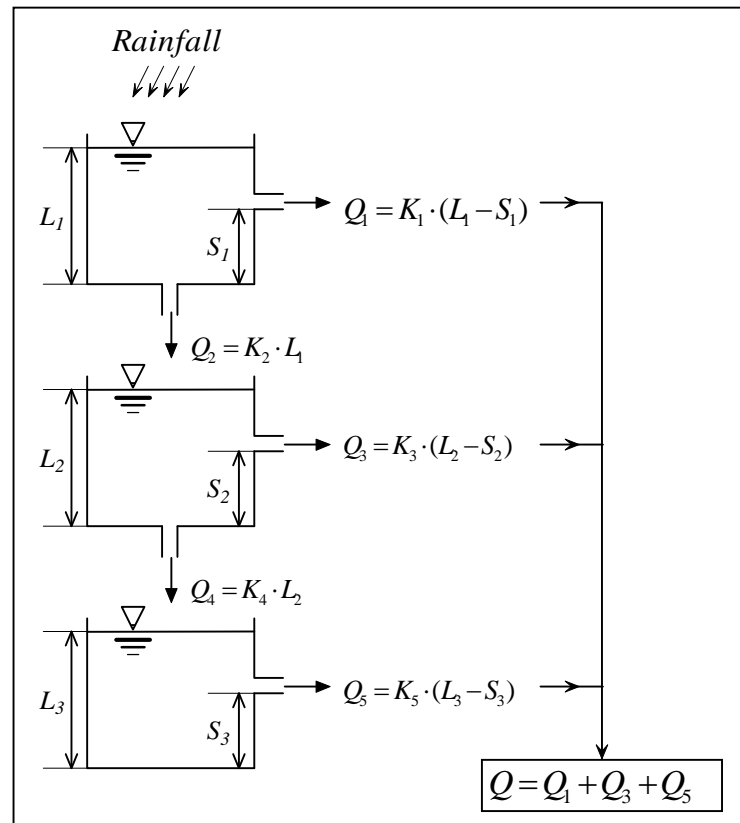


Figure 3-2 Three layer of Tank model

Auto2Fit code:

Variable Rainfall;	//unit: mm
Variable Runoff;	//unit: m ³ /s

```

Constant DT = 1;           //time interval, unit: hr
Constant n = 3;           //number of rainfall process
Parameter Area [0,3];     //catchment area, unit: km^2
Parameter K(1:5) [0.0001,1]; //discharge coef.
Parameter S(1:3) [1,100]; //storage coef.
VarParameter L(1:3) = 3 [0, 40]; //initial storage
StartProgram;
var
  i : integer;
  Q1, Q2, Q3, Q4, Q5: Double;
  SS1, SS2, SS3: Double;
begin
  SS1 := L1;
  SS2 := L2;
  SS3 := L3;
  for i := 0 to DataLength-1 do begin
    //tank 1
    if SS1 > S1 then
      Q1 := K1 * (SS1 - S1)
    else
      Q1 := 0;
    Q2 := K2*SS1;
    SS1 := SS1 + (Rainfall[i] - Q1 - Q2) * DT;
    if SS1 < 0 then SS1 := 0;
    //tank 2
    if SS2 > S2 then
      Q3 := K3*(SS2-S2)
    else
      Q3 := 0;
    Q4 := K4 * SS2;
    SS2 := SS2 + (Q2 - Q4 - Q5) * DT;
    if SS2 < 0 then SS2 := 0;
    //tank 3
    if SS3 > S3 then
      Q5 := K5 * (SS3 - S3)
    else
      Q5 := 0;
    SS3 := SS3 + (Q4 - Q5) * DT;
    if SS3 < 0 then SS3 := 0;
    // change unit from mm -> m^3/s
    Runoff[i] := 3600*(Q1+Q3+Q5)*Area*10/(DT*36);
  end;
end;
EndProgram;
DataFile C:\Applications\Auto2Fit\yangui_1.csv;
DataFile C:\Applications\Auto2Fit\yangui_3.csv;
DataFile C:\Applications\Auto2Fit\yangui_4.csv;
DataFile C:\Applications\Auto2Fit\yangui_6.csv;
DataFile C:\Applications\Auto2Fit\yangui_7.csv;
DataFile C:\Applications\Auto2Fit\yangui_10.csv;
DataFile C:\Applications\Auto2Fit\yangui_11.csv;
DataFile C:\Applications\Auto2Fit\yangui_12.csv;

```

Result: average correlation coefficient, $R = 0.9887$

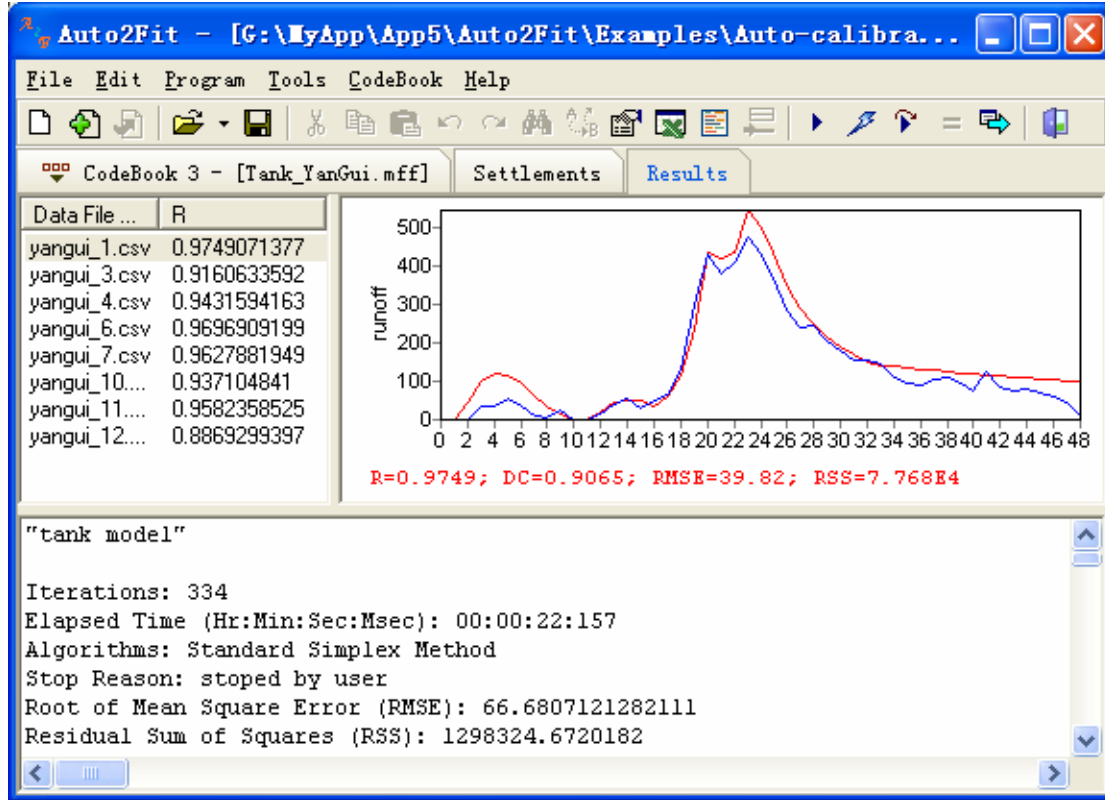


Figure 3-3 Screen shot of TANK model optimization

3.6 Calibration of water quality model of road runoff

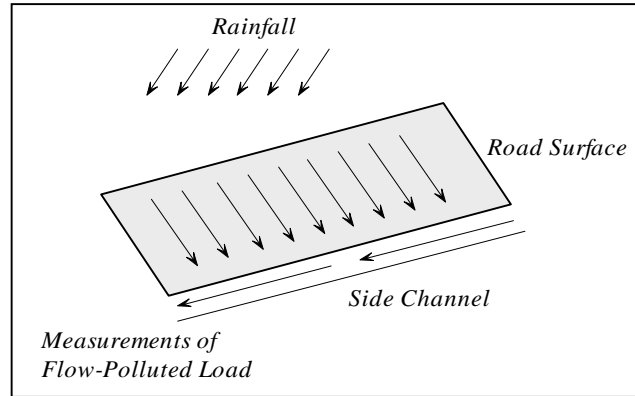


Figure 3-4 Illustration of road surface drainage tests

Figure 3-4 shows the experiment of road surface. The area is 995 m², Rainfall is artificial rainfall. The measurement items are BOD, COD, SS and discharge. The time interval is 10 minutes. There are six sets of measurement data.

The water quality model of road runoff is as following equations:

$$\begin{cases} L_t = k \cdot S_t \cdot Q_t \\ S_{t+1} = S_t - \int L_t dt \end{cases} \quad (3-4)$$

Here, L_t : at time point of t , load of dirty outflow (g/m²/hr)

k : factor of load of dirty outflow (1/mm)

Q_i : at time point of t , intensity of rain outflow (mm/hr)

S_t, S_{t+1} : at time point of t and $t+1$, Residual amount of surface accumulation of load (g/m²)

Parameter in use:

In this case, BOD and COD proceeds the calibration of model parameters at the same time, the data from six measurements are used also at the same time. The parameters are as follows:

- Initial BOD polluted load: S_{BOD} , total 6, unit: g/m²
- Initial COD polluted load: S_{cod} , total 6, unit: g/m²
- Coefficients of BOD and COD polluted outflow: K_{bod}, K_{cod} , total 2, unit: 1/mm

Final total number of parameters: $6 + 6 + 2 = 14$

Auto2Fit code:

```
Variable Flow; //unit: l/min
Variable BOD [Output]; //unit: mg/m^2/hr
Variable COD [Output]; //unit: mg/m^2/hr
Parameter k [0,1]; //unit: 1/mm
VarParameter S0_BOD[1,100,1,]=30; //unit: mg/m^2
Parameter k1 [0,1]; //unit: 1/mm
VarParameter S0_COD[1,]=30; //unit: mg/m^2
Constant Area=995; //unit: m^2
Constant dt=10; //unit: minutes
StartProgram;
var
  i: integer;
  S, S1: Double;
Begin
  S := S0_BOD;
  S1 := S0_COD;
  for i := 0 to DataLength - 1 do begin
    BOD[i] := k * S * Flow[i] * 60 / Area;
    S := S - BOD[i] * dt / 60;
    if S < 0 then S := 0;
    COD[i] := k1 * S1 * Flow[i] * 60 / Area;
    S1 := S1 - COD[i] * dt / 60;
    if S1 < 0 then S1 := 0;
  end;
End;
EndProgram;
```


water level caused by rainfall in every minute during period of t_1 and t_2 should be equal to the measured increments of water level during period of t_1 and t_2 , shown as Figure 3-6. The objective function is as follows,

$$\left\{ \begin{array}{ll} t_1 \text{ period} & \sum_{i=0}^{59} a \cdot R_{1,i}^b - 30.9524 = 0 \end{array} \right. \quad (3-6)$$

$$\left\{ \begin{array}{ll} t_2 \text{ period} & \sum_{i=0}^{59} a \cdot R_{2,i}^b - 0.2547 = 0 \end{array} \right. \quad (3-7)$$

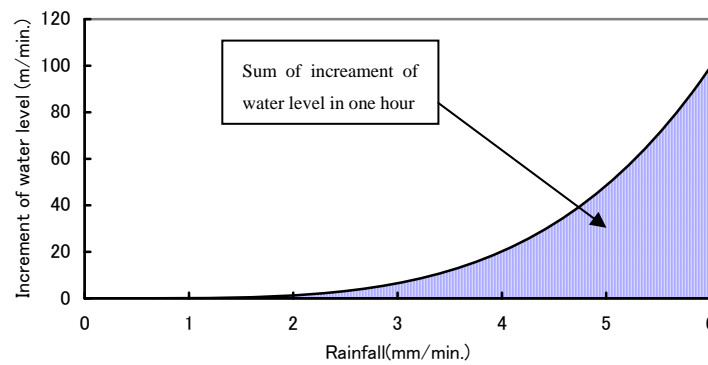


Figure 3-6 Relationship between rainfall and water level every minute

Now the problem can be converted to a system equations, a and b , are the roots of the equations.

Auto2Fit code:

```
Constant Rain1(0:59)=[0.5,0,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,1,0.5,0.5, 1,0.5,1,0.5,
1,1,1,1,1,1.5,1,1,1.5,2,1.5,1,2,1.5,2,1,1.5,2,2,2,2.5,2,2.5,2,
1.5,2.5,2,1.5,2,2,1,1,1,1.5,1,1,1.5,0.5,1,1];
Constant Rain2(0:59)=[0.5,0,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
0,0.5,0,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
0.5,0,0.5,0,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5];
Parameter a, b;
Function Sum(i = 0:59)(a*Rain1[i]^b)-30.9524 = 0;
Sum(i = 0:59)(a*Rain2[i]^b)-0.2547 = 0;
```

Result: $a = 0.086476$, $b = 3.93344$.

3.8 Combination optimization problem

See section 2.1.7.

Chapter 4 Communication with external program in Auto2Fit

With quick model and programming mode, Auto2Fit can handle most of the optimization problem. But as for some very complicated engineering optimization problem, even programming mode cannot meet the requirements, or these optimization problems have already be coded by other programming languages, and those code is difficult to re-write again in programming mode of Auto2Fit, in this case, Auto2Fit needs the ability to mix-language programming, in another word, Auto2Fit provides the optimization algorithm while external program support the computation of objective function and constrained function. From version 3.0, Auto2Fit has the complete capability to support mix-language programming. User can use high-level language, such as C++, FORTRAN, Delphi/Pascal, Basic or any other language which supports standard Windows DLL (.dll) or command-line executive file (.exe) to program any complicated optimization problem and then called by Auto2Fit.

There are two main methods to invoke external compiled program in Auto2Fit, one is through .dll file and the other is through command-line executive file. The efficiency of the first method is much higher than the second one. Therefore the method of invoking .dll file is recommended in most of the time.

The choices of external programming languages are many currently, user can choose the one they prefer and are good at. Now we describe how to use Visual C++, Borland C++, Visual Fortran, Gun Fortran, Delphi, PowerBasic and Free Basic to write program for Auto2Fit to invoke in details. Though Visual Basic (VB) is popular, it is not good at providing fast computation and good migration support because it is a type of interpretative language rather than compiler language. Thus it is not in the list of recommended languages.

The biggest advantage of external program invoke mode is to make full use of the ability of high-level programming language to handle any complicated optimization problem, such as complex calculation problem, though it is not supported by Auto2Fit directly, the complex optimization problem can be solved through the combined usage of C++ or FORTRAN which support the calculation of complex.

4.1 Format of invoke and key words

Whatever Dll or Exe is used, the basic principle is that the calculation of objective functions and constrained functions are provided by external program while Auto2Fit does the job of

parameter optimization through invoking the results from external program. Figure 4.1 and Figure 4.2 show the flow chart of interaction between Auto2Fit and external program.

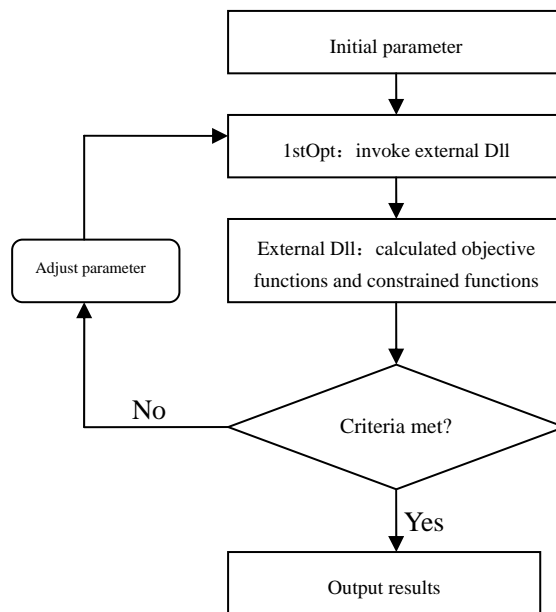


Figure 4-1 Flow chart of invoking external DLL in Auto2Fit

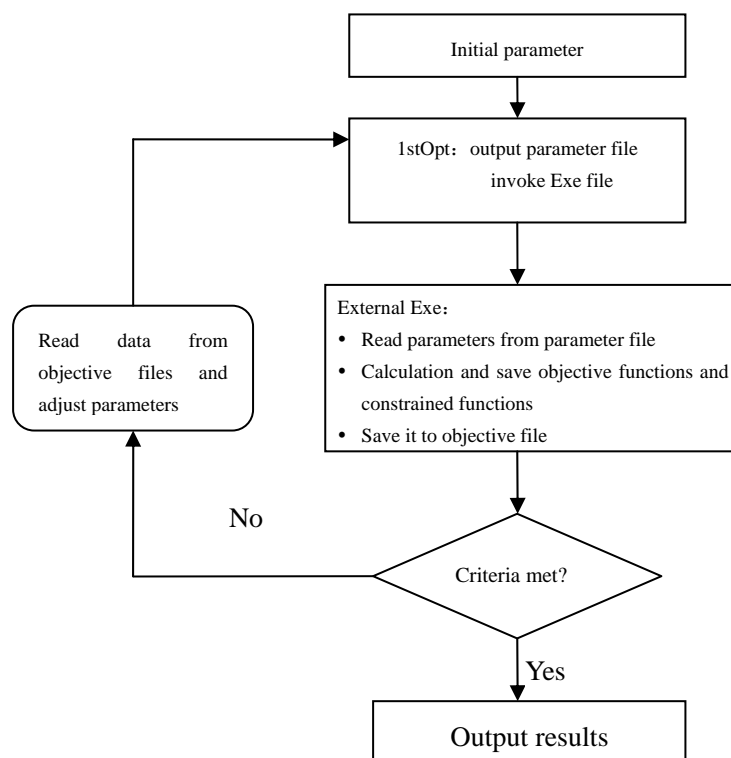


Figure 4-2 Flow chart of invoking external command line executable file in Auto2Fit

4.1.1 Call external Dll file

If the Dll file is “c:\test\dll_test.dll”, the invoke method is as follows :

MinFunction "c:\test\dll_test.dll[dllfunction, N1, N2]" ;

Here, “dllfunction” is the default function name of output file, optional, N1, N2 are the number of constraints for inequality and equality respectively. The format of inequality must be expressed as it is lower or equal to 0.

Key words: Parameter, PassParameter and MinFunction/MaxFunction

Some examples:

- 1) If the function name of output is the default “dllfunction”, the number of constraints for inequality (≤ 0) is 2 and for equality is 0, then the format of invoke expression is :

MinFunction "c:\test\dll_test.dll[2]" ;

- 2) If the function name of output is “myfunction”, the number of constraints for inequality (≤ 0) is 0 and for equality is 2, then the format of invoke expression is:

MinFunction "c:\test\dll_test.dll[myfunction, 0, 2]" ;

- 3) If the function name of output is the default “dllfunction”, the numbers of constraints for inequality (≤ 0) and for equality are both 0, then the format of invoke expression is :

MinFunction "c:\test\dll_test.dll" ;

- 4) If the function name of output is the default “dllfunction”, the numbers of constraints for inequality (≤ 0) is 0 and for equality is 4, then the format of invoke expression is :

MinFunction "c:\test\dll_test.dll[0, 4]" ;

- 5) If the function name of output is self-defined, such as “myfunction”, the numbers of constraint for inequality (≤ 0) is 1 and for equality is 4, then the format of invoke expression is :

MinFunction “c:\test\dll_test.dll[myfunction, 0, 4]" ;

The most important difference between invoking external program mode and programming mode of Auto2Fit is: the constant or constant string defined by “Constant”, “ConstStr”, “DataSet” in the code of Auto2Fit for invoking external program is invisible to external program, in another word, they cannot be passed to external program while they are visible and can be used in programming mode.

4.1.2 Call external Exe file

External .Exe file must be the format of command line executable file. The parameter file produced from Auto2Fit and objective function file must be in text format. In parameter file, a parameter is recorded row by row in order. In objective function file, the first line records the value of objective function and then the following lines record the values of constraints of inequality and equality.

For example, the Exe file is “c:\test\exe_test.exe”, the output of parameter file is “c:\test\exe_par.txt”, the output of objective function file is “c:\test\exe_obj.txt”, the numbers of

constraints of inequality and equality are 3 and 2 respectively, then the invoke format is :

```
ExeParameterFile = " c:\test\exe_par.txt ";
ExeObjectiveFile = " c:\test\exe_obj.txt ";
MinFunction " c:\test\exe_test.exe[3, 2]" ;
```

Some examples:

- 1) The number of constraint for inequality is 3 and for equality is 0: MinFunction " c:\test\exe_test.exe[3]" ;
- 2) The number of constraint for inequality is 0 and for equality is 2: inFunction " c:\test\exe_test.exe[0, 2]" ;
- 3) The number of constraint for inequality and for equality is both 0: MinFunction " c:\test\exe_test.exe" ;

4.2 Communication with C++ compiled Dll file

The format of output function is:

```
extern "C" void __declspec(dllexport) __stdcall
dllfunction(double *para, double *objfun, double *confun1, double *confun2, double *passpara)
{
    //Add code for calculation of objective and constrained functions
}
```

Here:

- dllfunction: default function name of output, which can self-defined
- Para: array of parameters passed from Auto2Fit, subscript starts from 0
- ConFun1, ConFun2: array of inequality and equality, subscript starts from 0
- ObjFun: value of objective functions
- PassPara: array of pass parameters, subscript starts from 0

Example 1: Function optimization problem

$$\text{Min. } 9-x-y \quad (4-1)$$

$$\text{S.T. } \begin{cases} (x-3)^2 + (y-2)^2 \leq 16 \\ x \cdot y \leq 14 \end{cases} \quad (4-2)$$

Here, x,y are ranged in [0, 100].

- 1) C++ code as follows:

```
#include <windows.h>
#include <math.h>

extern "C" void __declspec(dllexport) __stdcall
dllfunction(double *para, double *objfun, double *confun1, double *confun2, double *passpara)
{
    *objfun = 9-para[0]-para[1];
    confun1[0] = pow((para[0]-3),2)+pow((para[1]-2),2)-16;
```

```
confun1[1] = para[0]*para[1]-14;
}
```

- 2) Save file, e.g., "c:\projects\cplus_test1.cpp"
- 3) Use Borland C++5.5 compiler to compile it to a Dll file, e.g., if C++ compiler is located at "d:\Borland C\bin", the compiling command is:
"d:\Borland C\bin\bcc32" -tWM -tWD "c:\projects\cplus_test1.cpp"
- 4) Execute the above command, "cplus_test1.dll" can be produced in "d:\Borland C\bin"
- 5) Auto2Fit call code

```
Parameter x[0,100],y[0,100];
MinFunction " d:\Borland C\bin\ cplus_test1.dll[2]";
```

- 6) Auto2Fit quick model code

```
Parameter x[0,100],y[0,100];
minFunction 9-x-y;
(x-3)^2+(y-2)^2<=16;
x*y<=14;
```

Execute the above steps 5 and 6, the same results can be achieved: Min. = 0, x = 7, y = 2

4.3 Communication with Visual Fortran (VF) compiled Dll file

Visual Fortran is a widely used Fortran compiler. The name of VF command line compiler executable file is "DF.exe". If use it to compile a VF source file "c:\projects\VF_test1.f90" to a Dll file, the command is as follows:

DF.exe /dll "c:\projects\VF_test1.f90"

The format definition of output functions is as follows:

```
subroutine dllfunction(para, objfun, confun1, confun2, passpara)
!dec$attributes dllexport , stdcall :: dllfunction
!dec$attributes reference :: para, objfun, confun1, confun2
integer, parameter :: fp = selected_real_kind(15,300)
real(fp) :: para(0:1), confun1(0:0), confun2(0:0), passpara(0:0)
real(fp) :: objfun
end subroutine
```

Here:

- dllfunction: default function name of output, which can self-defined
- para: array of parameters passed from Auto2Fit, subscript starts from 0
- objfun: value of objective functions
- confun1, confun2: array of inequality and equality, subscript starts from 0
- passpara: array of pass parameters, subscript starts from 0

Statement "real(fp) :: para(0:1), confun1(0:0), confun2(0:0), passpara(0:0)" need to be changed

according to real situation, If the number of parameters is 3, the number of constraints for inequality is 4, the number constraints for equality is 2, the number of pass parameter is 20, then the above statement needs to be changed to :

real(fp) :: para(0:2), confun1(0:3), confun2(0:1), passpara(0:19)

Example 2: Constrained optimization problem

$$\text{Min.} - x_1 \cdot x_2 - x_2 \cdot x_3 - x_3 \cdot x_1 \quad (4-3)$$

$$\text{S.T.} \begin{cases} 0.5(x_1 - 3)^2 + x_2^2 + x_3^3 - 1 \leq 0 \\ \frac{x_1}{0.5 + x_2^2} + 2x_3 - 4 \leq 0 \\ x_1 + x_2 + x_3 = 3 \end{cases}$$

1) Fortran code as follows :

```
subroutine dllfunction(para, objfun, confun1, confun2, passpara)
!dec$attributes dllexport , stdcall :: dllfunction
!dec$attributes reference :: para, objfun, confun1, confun2
integer, parameter :: fp = selected_real_kind(15,300)
real(fp) :: para(0:2), confun1(0:1), confun2(0:0), passpara(0:0)
real(fp) :: objfun
real temd
integer i
    objfun = -para(0)*para(1)- para(1)*para(2)- para(2)*para(0)
    confun1(0) = 0.5*(para(0)-3)**2+para(1)**2+para(2)**3-1
    confun1(1) = para(0)/(0.5+para(1)**2)+2*para(2)-4
    confun2(0) = para(0)+para(1)+para(2)-3
end subroutine
```

2) Save file,e.g., “c:\projects\VFortran_test1.f90”

3) Compile it to Dll file, IF VF compiler is located at“C:\Program Files\Microsoft Visual Studio\DF98\BIN”,then the compile command is as follows:

C:\Program Files\Microsoft Visual Studio\DF98\BIN\DF.exe /dll
“c:\projects\VFortran_test1.f90”

4) Execute the above command, produce a Dll file “Vfortran_test1.dll”

5) Code for calling dll file

```
Parameter x(3);
MinFunction "c:\Projects\VFortran_Test.dll[2,1]";
```

6) Code in quick model

```
MinFunction -x1*x2-x2*x3-x3*x1;
0.5*(x1-3)^2+x2^2+x3^3-1<=0;
x1/(0.5+x2^2)+2*x3-4<=0;
x1+x2+x3=3;
```

The same results can be acquired in both two modes:

Value of objective function (minimum): -2.34512297572644

x1: 1.93394964369396

x2: 0.506937411738282

x3: 0.559112944567762

4.4 Communication with Gun Fortran compiled Dll file

Gun Fortran (FFortran) is an open source, free, cross-platform Fortran compiler. It supports Fortran95/2003 standard and the official website is <http://gcc.gnu.org/>.

GFortran only provides command line compiler; the detailed command can be referred to its user manual and other related references. The name of Gfortran compiler executable file is “Gfortran.exe”. If use it to compile Fortran source file “c:\projects\GF_test1.f90” to a Dll file, the command is:

```
Gfortran.exe -o "c:\projects\GF_test1.dll" -s -shared -mrtd -fno-underscoring
"c:\projects\GF_test1.f90"
```

The format definition of output functions is as follows:

```
subroutine dllfunction(para, objfun, confun1, confun2, passpara)
integer, parameter :: fp = selected_real_kind(15,300)
real(fp) :: para(0:1), confun1(0:0), confun2(0:0), passpara(0:0)
real(fp) :: objfun

end subRoutine
```

here:

- dllfunction: default function name of output, which can self-defined
- para: array of parameters passed from Auto2Fit, subscript starts from 0
- objfun: value of objective functions
- confun1, confun2: array of inequality and equality, subscript starts from 0
- passpara: array of pass parameters, subscript starts from 0

Statement “real(fp) :: para(0:1), confun1(0:0), confun2(0:0), passpara(0:0)” need to be changed according to real situation, If the number of parameters is 3, the number of constraints for inequality is 4, the number constraints for equality is 2, the number of pass parameter is 20, then the above statement needs to be changed to :

```
real(fp) :: para(0:2), confun1(0:3), confun2(0:1), passpara(0:19)
```

Example 3: Unconstrained optimization problem

$$\text{Min. } \sum_{i=1}^{n-1} \left(100 \cdot (x_i^2 - x_{i+1})^2 + (x_i + \sin(x_{i+1}) - 1)^2 \right) \quad (4-4)$$

Here, n = 10, the range of 10 variables are all in [-5, 10].

7) Fortran source code as follows:

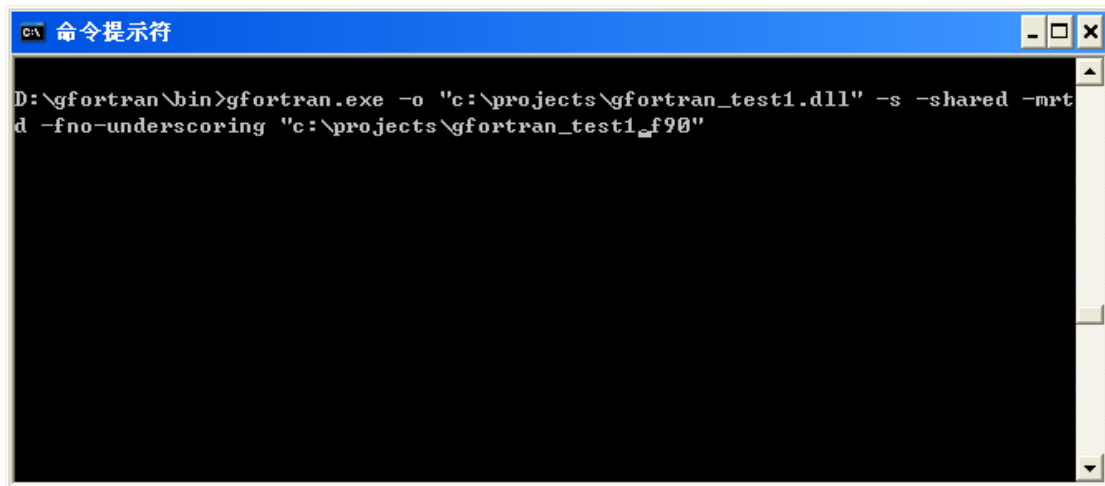
```
subroutine dllfunction(para, objfun, confun1, confun2, passpara)
integer, parameter :: fp = selected_real_kind(15,300)
real(fp) :: para(0:1), confun1(0:0), confun2(0:0), passpara(0:0)
real(fp) :: objfun
real temd
integer i
    temd = 0.0
```

```

Do i = 0, 8
    temd = temd + 100.0*(para(i)**2.0-para(i+1))**2.0+(para(i)+sin(para(i+1))-1.0)**2.0
EndDo
objfun = temd
end subroutine

```

- 8) Save file, e.g., "c:\projects\Gfortran_test1.f90"
- 9) Compile it to a Dll file, if GFortran compiler is located at "d:\gfortran\bin", then the compiling command is as follows:
d:\gfortran\bin\gfortran.exe -o "c:\projects\gfortran_test1.dll" -s -shared -mrt
-fno-underscoring "c:\projects\gfortran_test1.f90"



- 10) Execute the above command, produce a Dll file, "gfortran_test1.dll", in "c:\projects"
- 11) The invoke command in Auto2Fit

```

Constant n=10;
Parameter x(1:n)[-5,10];
MinFunction "c:\Projects\GFortran_Test.dll";

```

- 12) Code in quick model

```

Constant n=10;
Parameter x(1:n)[-5,10];
Minimum;
Function Sum(i=1:n-1)(100*(x[i]^2-x[i+1])^2+(x[i]+sin(x[i+1])-1)^2);

```

The same results can be acquired in two modes: Min. = 3.70328593254089

Example 4: Curve fit problem

Curve fit equation:

$$y = p_1 + (p_2 - p_1) \cdot (1 - \exp(-p_3 \cdot x)) \quad (4-5)$$

Data for curve fit

x	15,30,45,60,75,90,105,120,135,495
---	-----------------------------------

y	0.489,0.427,0.373,0.327,0.285,0.250,0.218,0.191,0.167,0.005
---	---

- 1) Fortran source code

```

subroutine dllfunction(para, objfun, confun1, confun2, passpara)
  integer, parameter :: fp = selected_real_kind(15,300)
  real(fp) :: para(0:2), confun1(0:0), confun2(0:0), passpara(0:0)
  real(fp) :: objfun
  real :: x(0:9)=(/15.000,30.000,45.000,60.000,75.000,90.000,105.000,120.000,135.000,495.000/)
  real :: y(0:9)=(/0.489,0.427,0.373,0.327,0.285,0.250,0.218,0.191,0.167,0.005/)
  integer i
  real temd, temy
  temd = 0.0
  do i = 0, 9
    temy = para(0)+(para(1)-para(0))*(1-exp(-para(2)*x(i)))
    temd = temd + (temy-y(i))*2.0
    passpara(i) = y(i)
    passpara(i + 10) = temy
  enddo
  objfun = temd
end subroutine

```

- 2) Save file,e.g., "c:\projects\Gfortran_reg.f90"
- 3) Compile it to a Dll file, if GFortran compiler is located in "d:\gfortran\bin",then the compiling command is :


```

d:\gfortran\bin\gfortran.exe -o "c:\projects\gfortran_reg.dll" -s -shared -mrtd
-fno-underscoring "c:\projects\gfortran_reg.f90"

```
- 4) Execute the above command,a Dll file, "gfortran_reg.dll", can be produced in "c:\projects"
- 5) The invoke command in Auto2Fit

```

Parameter p(3);
PassParameter ObsY(10), CalY(10);
Plot ObsY, CalY;
MinFunction " c:\projects\gfortran_reg.dll";

```

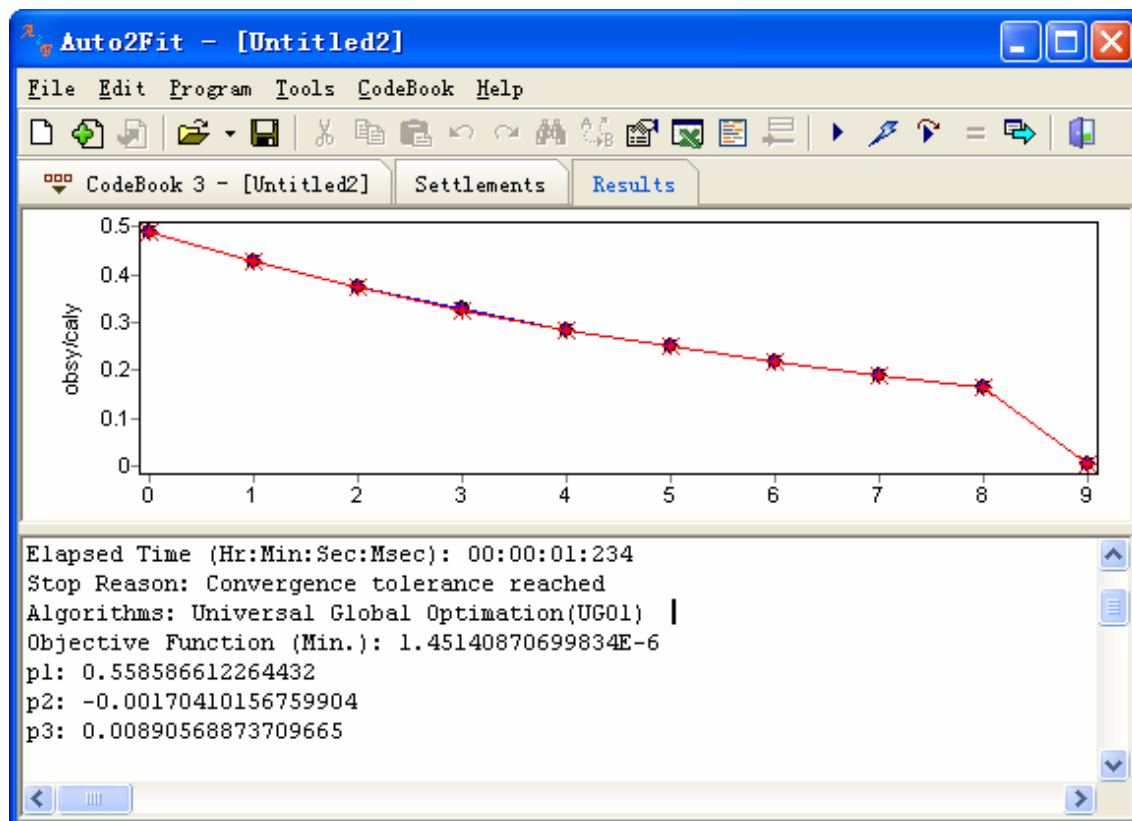


Figure 4-3 Screen shot of running results

6) Code in quick model

```

RowDataSet;
  x=15,30,45,60,75,90,105,120,135,495;
  y=0.489,0.427,0.373,0.327,0.285,0.250,0.218,0.191,0.167,0.005;
EndRowDataSet;
MinFunction Sum(x,y)((p2+(p3-p2)*(1-exp(-p1*x))-y)^2);
  
```

The same results can be acquired in both two modes:Min. = 1.45142373499332E-6

4.5 Communication with Delphi compiled Dll file

Delphi is an excellent software development environment for Windows platform. It is an object-oriented, easy to use, and real compiled and can produce highly efficient Dll. From version 3.0 to version 2009, it can be combined use with Auto2Fit. The following takes Delphi2007 as an example.

The format of output function is defined as follows:

```

library DllProject;

uses  SysUtils, Classes, Math, Consts;

type
  TVector = array[0..1] of Double;
  PVector = ^TVector;
  
```

```

procedure dllfunction(para: pvector; var objfun: double; confun1, confun2, passpara: pvector); stdcall;
begin
    //Add code for calculation of objective and constrained functions
end;

exports dllfunction;

begin
end.

```

Here:

- DllProject : function of Dll library, which can be self-defined
- dllFunction: default name of output, which can self-defined
- Para: array of parameters passed from Auto2Fit, subscript starts from 0
- ConFun1, ConFun2: array of inequality and equality, subscript starts from 0
- objfun: value of objective functions
- PassPara: array of pass parameters, subscript starts from 0
- “TVector = array[0..1] of Double;”: the outbound of array defined should be the largest number of values of outbound of arrays, Para, ConFun1, ConFun2 and PassPara minus 1. If the largest number of outbound of array is 10, then the definition becomes :
TVector = array[0..9] of Double;

Example 5: Constrained optimization problem

The following use a constrained function optimization problem as an example to describe how to use Delphi 2007 to create an external Dll and how to invoke it from Auto2Fit.

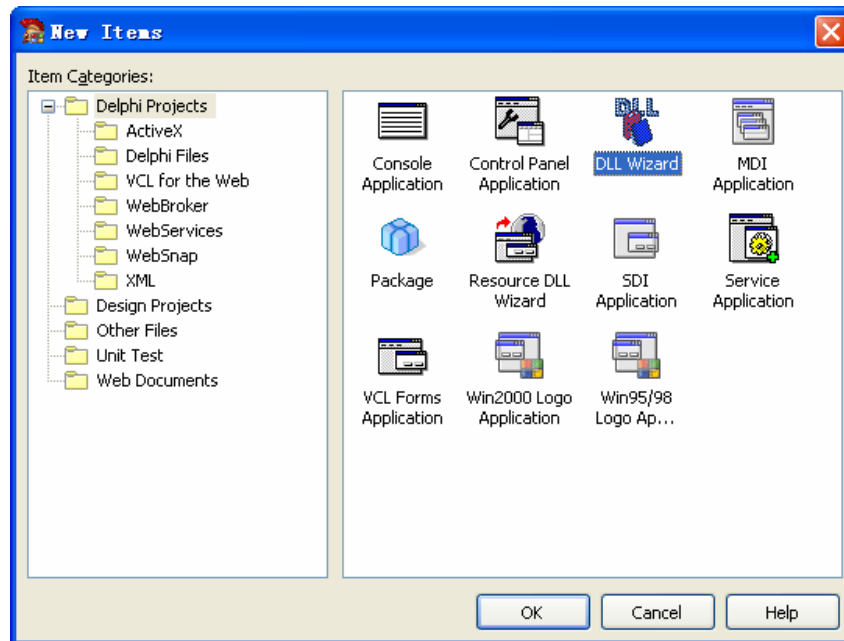
The optimization problem is defined as follows, there is one equality constraint and two inequality constraints in addition to one objective function, there is no limit to the range of parameters.

$$\text{Min. } x_1^2 + x_2 \quad (4-6)$$

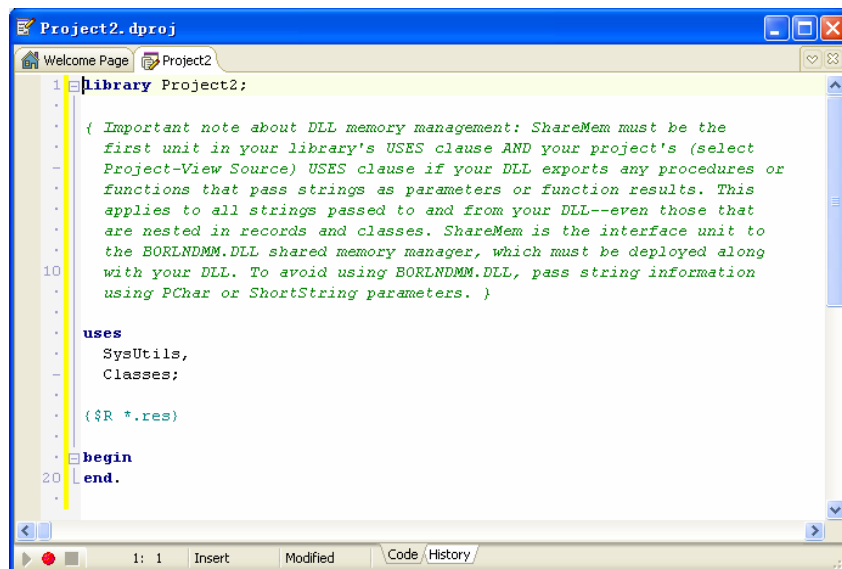
$$\text{S.T. } \begin{cases} x_1^2 + x_2^2 - 9 = 0 \\ x_1 + x_2^2 - 1 \leq 0 \\ x_1 + x_2 - 1 \leq 0 \end{cases}$$

The steps of using Delphi 2007 to create the Dll for the above optimization problem is as follows:

- (1) Launch Delphi2007, choose “DLL Wizard”, press “OK”



- (2) Delphi 2007 will automatically generate the following code



- (3) Add the following code

```
library DllProject;

uses SysUtils, Classes, Math, Consts;

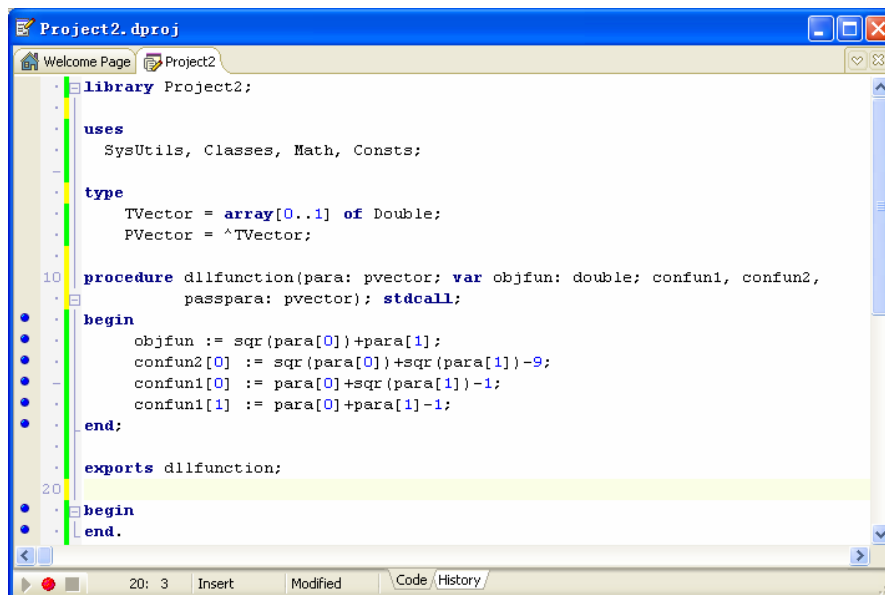
type
  TVector = array[0..1] of Double;
  PVector = ^TVector;

procedure dllfunction(para: pvector; var objfun: double; confun1, confun2, passpara: pvector); stdcall;
begin
  objfun := sqr(para[0]) + para[1];
  confun2[0] := sqr(para[0]) + sqr(para[1]);
  confun1[0] := para[0] + sqr(para[1]) - 1;
  confun1[1] := para[0] + para[1] - 1;
end;
```

```
exports dllfunction;
```

```
begin
```

```
end.
```



(4) Compile it to produce a Dll file, e.g., "C:\Projects\Project2.dll";

(5) Code of Incode in Auto2Fit

```
Parameter x1, x2;
```

```
MinFunction "C:\Projects\Project2.dll[2,1]";
```

After execution, results can be easily acquired: Min = 3.7913455, $x_1 = -2.3722817$, $x_2 = -1.836375$

(6) Compared with quick model, the results in both modes are the same.

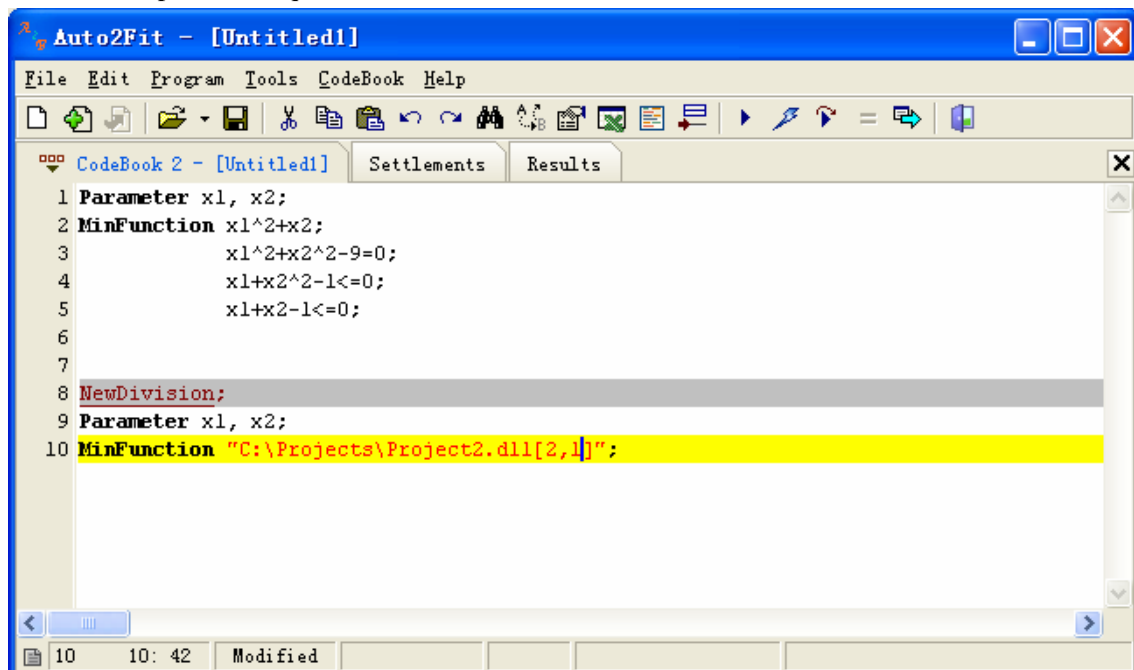


Figure 4-4 Invoke methods of quick model and external invoke mode

Example 6: Constrained curve fit problem

Curve fit equation:

$$y = \frac{b_1(x^2 + b_2x)}{x^2 + b_3x + b_4} \quad (4-7)$$

Constraint: $b_1 + b_2 + b_3 + b_4 = 1$

Data for curve fit

x	4.0000, 2.0000, 1.0000, 0.5000, 0.2500, 0.1670, 0.1250, 0.1000, 0.0833, 0.0714, 0.0625
---	--

y	0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246
---	--

Delphi code is as follows, which could be complied successfully through Delphi 3.0 to Delphi 2009.

Code of Dll file for data curve fit

```
library Pascal_Demo2_Reg;

uses SysUtils, Classes, Math, Consts;

type
  TVector = array[0..3] of Double;
  PVector = ^TVector;

const x : array[0..10] of double =
  (4.0000,2.0000,1.0000,0.5000,0.2500,0.1670,0.1250,0.1000,0.0833,0.0714,0.0625);
  y : array[0..10] of double =
  (0.1957,0.1947,0.1735,0.1600,0.0844,0.0627,0.0456,0.0342,0.0323,0.0235,0.0246);

procedure dllfunction(para: pvector; var objfun: double; confun1, confun2, passpara: pvector); stdcall;
var i, p: integer;
    temD, CalY: double;
begin
  temD := 0;
  for i := 0 to 10 do begin
    CalY := para[0]*(sqr(x[i])+x[i]*para[1])/(sqr(x[i])+x[i]*para[2]+para[3]);
    temD := temD + sqr(CalY - y[i]);
    passpara[i] := x[i];
    passpara[i + 11] := y[i];
    passpara[i + 22] := CalY;
  end;
  confun2[0] := para[0]+para[1]+para[2]+para[3]-1;
  objfun := temD;
end;

exports dllfunction;

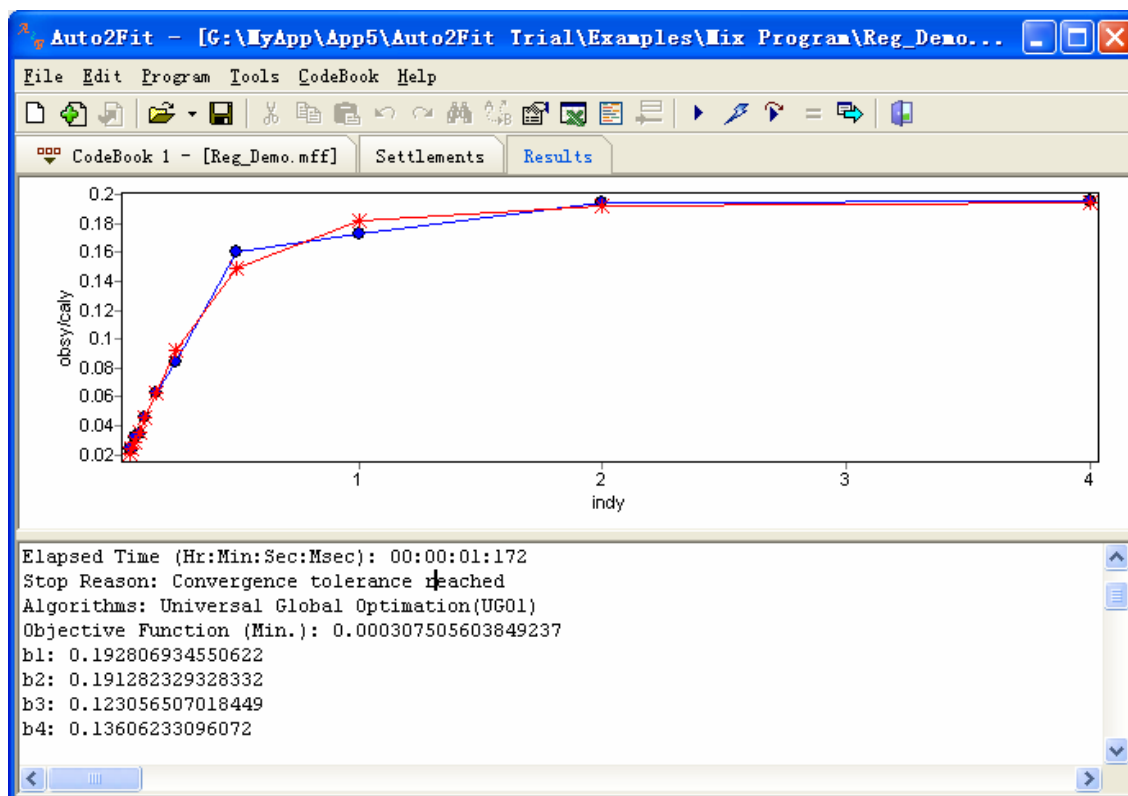
begin
end.
```

If the complied Dll is “C:\Projects\ Pascal_Demo2_Reg”, then the code for invoke in Auto2Fit is as follows.

Code of invoke Dll in Auto2Fit

```
Parameter b1,b2,b3,b4;
PassParameter X(0:10), ObsY(0:10), CalY(0:10);
Plot X[x], ObsY, CalY;
MinFunction "F:\MyApp\App\Auto2Fit\Examples\Mix Program\Pascal_Demo2_Reg.dll";
```

In the above code, “PassParameter” defines three one dimensional array, independent variable x , dependent variable (object) ObsY and dependent variable (calculation) CalY, the length of array and data for curve fit is the same. Note that the way of calculating and return “PassParameter”: no matter how many one-dimensional arrays defined, they are allocated in a one-dimensional array “passpara”. In this case, the elements with subscripts from 0 to 10 in “passpara” are x , from 11 to 21 are ObsY, from 22 to 32 are CalY. Statement “Plot X[x], ObsY, CalY;” means X is the horizontal axis, ObsY and CalY are vertical axis. When execute the above code, the computation results and graphic changes can be displayed dynamically, see below:



4.6 Communication with PowerBasic compiled Dll file

PowerBasic is a powerful Windows application development tool in a wide range of products of Basic family. The official website of it is <http://www.powerbasic.com/>.

PowerBasic compiler allows you to use BASIC to write industry-standard Dll and Exe file for DoS or Windows applications. Though PowerBASIC (PB) or MicroSoft Visual Basic (VB) are both belonged to BASIC family, the only difference is that VB is interpreted language while PB is compiled language. Usually the performance of code compiled by PB is more than three times faster

than the code compiled by VB. Moreover, the executable file generated by PB is smaller than the one generated by VB and it doesn't need the running support from external file.

This section takes PB 8.0 as an example to describe how to build a Dll compatible to Auto2Fit programming interface.

The format of output functions is defined as follows:

```
#Compile Dll
#Dim All
#include "Win32api.inc"

sub dllfunction stdcall alias "dllfunction" (byval para as double ptr, byref objfun as double, byval confun1 as double ptr, byval confun2 as double ptr, byval passpara as double ptr) export
    'Add calculation code for calculation of objective and constrained functions
end sub
```

Here:

- Compile Dll: PB command, compile it to a Dll file
- dllfunction: default function name of output, it can be self defined.
- para: array of parameters passed from Auto2Fit, subscript starts from 0
- confun1, confun2: array of inequality and equality, subscript starts from 0
- objfun: value of objective functions
- passpara: array of pass parameters, subscript starts from 0

Example 7: Integer constrained optimization problem

$$\text{Min. } x_1^{0.1} + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \quad (4-8)$$

$$\text{S.T. } \begin{cases} x_1 + x_4 + x_5 + x_6 + x_7 \geq 50 \\ x_1 + x_2 + x_5 + x_6^{0.2} + x_7 \geq 50 \\ x_1 + x_2^{0.4} + x_3 + x_6 + x_7 \geq 50 \\ x_1 + x_2 + x_3 + x_4 + x_7 \geq 50 \\ x_1 + x_2 + x_3 + x_4 + x_5 \geq 80 \\ x_2 + x_3 + x_4 + x_5 + x_6 \geq 90 \\ x_3 + x_4 + x_5 + x_6 + x_7^{-0.12} \geq 100 \end{cases}$$

Here parameters are all positive integers which are larger than 1, x_1 is ranged in [5, 10]. This optimization problem has one objective function and seven inequality constrained functions.

1) Launch PB IDE, PBEEdit.Exe, input code as follows :

```
#Compile Dll
#Dim All
#include "Win32api.inc"

Sub dllfunction StdCall Alias "dllfunction" (ByVal para As Double Ptr, ByRef objfun As Double, _
    ByVal confun1 As Double Ptr, ByVal confun2 As Double Ptr, ByVal passpara As Double Ptr) Export
    objfun = @para[0]^0.1 + @para[1] + @para[2] + @para[3] + @para[4] + @para[5] + @para[6]
```

```
@confun1[0] = 50 - (@para[0] + @para[3] + @para[4] + @para[5] + @para[6])
@confun1[1] = 50 - (@para[0] + @para[1] + @para[4] + @para[5]^0.2 + @para[6])
@confun1[2] = 50 - (@para[0] + @para[1]^0.4 + @para[2] + @para[5] + @para[6])
@confun1[3] = 50 - (@para[0] + @para[1] + @para[2] + @para[3] + @para[6])
@confun1[4] = 80 - (@para[0] + @para[1] + @para[2] + @para[3] + @para[4])
@confun1[5] = 90 - (@para[1] + @para[2] + @para[3] + @para[4] + @para[5])
@confun1[6] = 100 - (@para[2] + @para[3] + @para[4] + @para[5] + @para[6]^(-0.12))
```

End Sub

Note that the constraints of the inequality change the format from “ \geq ” to “ ≤ 0 ” in the code.

2) Save it as “PowerBasic_Demo1.bas”, and compile it to a Dll file, “PowerBasic_Demo1.dll”

3) The invoke code in Auto2Fit

Parameter x1[5,10,0], x(2:7) = [1,,0];

Algorithm = SM2[50];

MinFunction "F:\MyApp\Auto2Fit\Examples\Mix Program\PowerBasic_Demo1.dll[7]";

4) For easier comparison, the codes of quick model and programming mode are given as follows.

Quick model:

Parameter x1[5,10,0];

ParameterDomain = [0,,0];

MINfunction x1^0.1 + x2 + x3 + x4 + x5 + x6 + x7;

x1 + x4 + x5 + x6 + x7 >= 50;

x1 + x2 + x5 + x6^0.2 + x7 >= 50;

x1 + x2^0.4 + x3 + x6 + x7 >= 50;

x1 + x2 + x3 + x4 + x7 >= 50;

x1 + x2 + x3 + x4 + x5 >= 80;

x2 + x3 + x4 + x5 + x6 >= 90;

x3 + x4 + x5 + x6 + x7^(-0.12) >= 1000;

Programming mode:

```

Parameter x1[5,10,0], x(2:7) = [1.,0];
Algorithm = SM2[50];
Minimum;
StartProgram [Basic];
Sub MainModel
  ObjectiveResult = x1^0.1 + x2 + x3 + x4 + x5 + x6 + x7
  ConstrainedResult = 50 - (x1 + x4 + x5 + x6 + x7) <= 0
  ConstrainedResult = 50 - (x1 + x2 + x5 + x6^0.2 + x7) <= 0
  ConstrainedResult = 50 - (x1 + x2^0.4 + x3 + x6 + x7) <= 0
  ConstrainedResult = 50 - (x1 + x2 + x3 + x4 + x7) <= 0
  ConstrainedResult = 80 - (x1 + x2 + x3 + x4 + x5) <= 0
  ConstrainedResult = 90 - (x2 + x3 + x4 + x5 + x6) <= 0
  ConstrainedResult = 100 - (x3 + x4 + x5 + x6 + x7^(-0.12)) <= 0
End Sub
EndProgram;

```

The same optimization result in the above modes can be acquired: Min. = 102.174618943088, but the values of parameters are not the only.

Among three modes, quick model is undoubtedly simplest and easy to understand, then the programming mode. Relatively speaking, invoke external Dll mode is more complicated but has incomparable advantages. It can make full use of the high-level programming capability to build any levels of objective files.

4.7 Communication with Free Basic compiled Dll file

FreeBasic (FB) is an open source BASIC compiler and is called “Dark horse in Basic community”. It is fully compatible with QuickBASIC. It is easy to use and cross platform. It can generate high quality local machine binary code with fast computation speed. It is real complied and doesn’t need the support from external library. The official Website of it is <http://www.freebasic.net>.

FB only provides command line compiler and the detailed command can be referred to its user manual and related references. The name of FB compiler executable file is “FBC.exe”. If use it to compile the source file “c:\projects\FB_test1.bas” to Dll file, the command is as follows:

```
fbc.exe -dll " c:\projects\FB_test1.bas ";
```

The format of output function is defined as follows:

```

'dllfunction: Default name of function
'Para: Parameter passed from Auto2Fit
'ConFun1, ConFun2: Constraints of inequality and equality
'ObjFun: Value of objective function
'PassPara: Returned parameter

sub dllfunction stdcall alias "dllfunction" (byval para as double ptr, byref objfun as double, byval confun1 as double ptr, byval confun2 as double ptr, byval passpara as double ptr) export

end sub

```

here:

- dllfunction: function name of Dll Library, which can be self defined.

- para: array of parameters passed from Auto2Fit, subscript starts from 0
- confun1, confun2: array of inequality and equality, subscript starts from 0
- objfun: value of objective functions
- passpara: array of passed parameters, subscript starts from 0

Take the example 2 in the chapter here as the problem to be solved:

1) Free Basic code

```
sub dllfunction stdcall alias "dllfunction" (byval para as double ptr, byref objfun as double, byval confun1 as double ptr, byval confun2 as double ptr, byval passpara as double ptr) export
    objfun = -para[0]*para[1]- para[1]*para[2]- para[2]*para[0]
    confun1[0] = 0.5*(para[0]-3)^2+para[1]^2+para[2]^3-1
    confun1[1] = para[0]/(0.5+para[1]^2)+2*para[2]-4
    confun2[0]= para[0]+para[1]+para[2]-3
end sub
```

Save it as “c:\FB_Test1.bas”

- 2) Compile it to Dll file, if FB compiler is located at “C:\Free Basic\BIN”, then the compiler command is as follows :

C:\Free Basic\BIN\fbcc.exe -dll “C:\FB_Test1.bas”

- 3) Execute the above command, generate a Dll file: “FB_Test1.dll”
- 4) Invoke command in Auto2Fit

```
Parameter x(3);
MinFunction "c:\FB_Test1.dll[2,1]";
```


- 5) Code in quick model

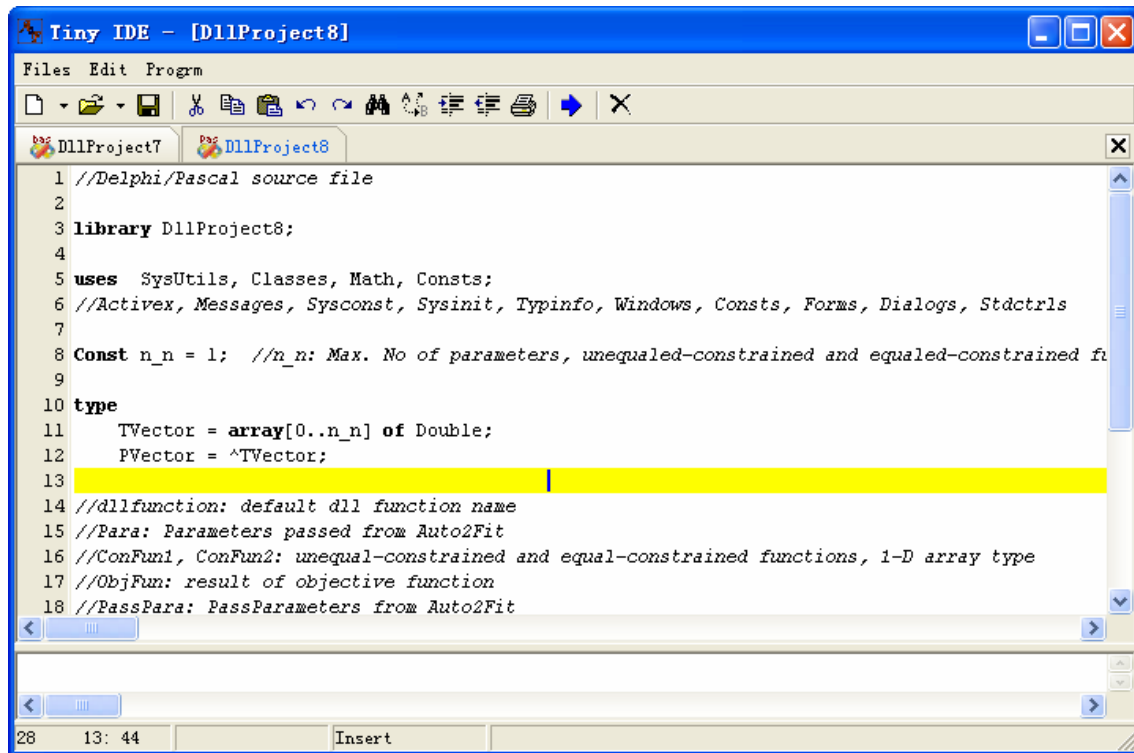
```
MinFunction -x1*x2-x2*x3-x3*x1;
    0.5*(x1-3)^2+x2^2+x3^3-1<=0;
    x1/(0.5+x2^2)+2*x3-4<=0;
    x1+x2+x3=3;
```

The same result can be acquired in both of these two modes.

4.8 Use external program editor of Auto2Fit (IDE)

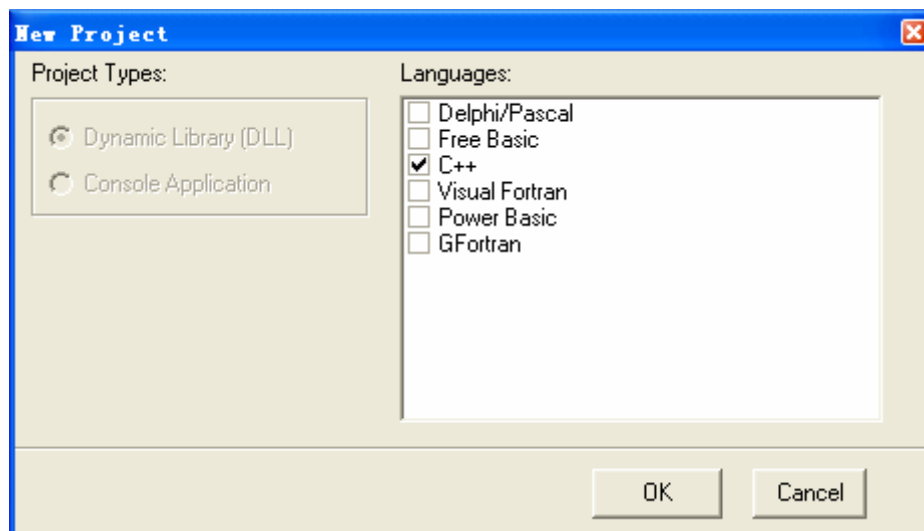
Auto2Fit has one external program editor (IDE), which can be used to edit, compile program using external high-level programming language. This IDE has Dll template for Delphi, FreeBasic, C++, Visual Fortran, PowerBasic and Gun Fortran and is easy for user to create Dll file.

- 1) Launch IDE: Choose “Tools” —> “External program editor”, from main menu, or press Keyboard shortcut “F7”, or press button  :

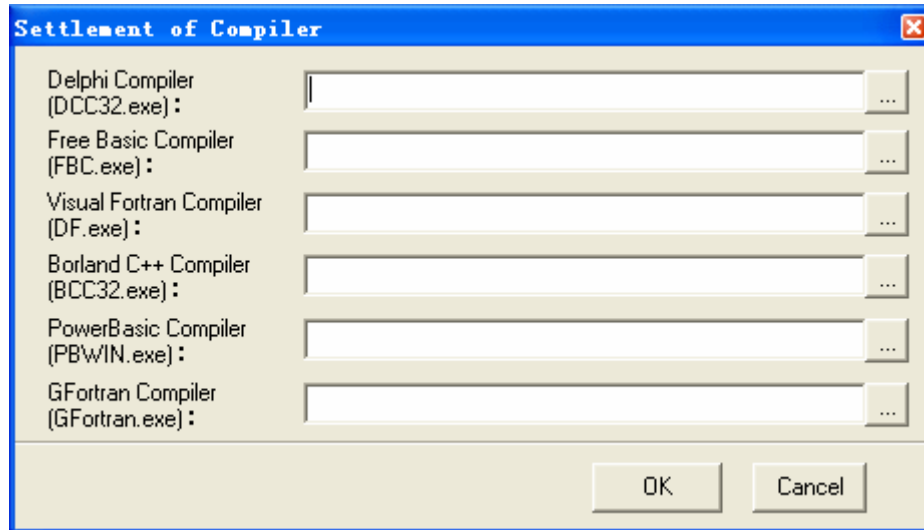


External program editor

- 2) Add new project file : Choose “File” —> “New” to add new project file , user can choose the programming language he prefers



- 3) Settings of compiler : choose compiler, include the path and name of compiler



Pre-define template: choose different language, which would generate different Dll file template

If choose Delphi, the code will be generated automatically as:

```
//Delphi/Pascal source file
```

```
library DllProject2;
```

```
uses SysUtils, Classes, Math, Consts;
```

```
//Activex, Messages, Sysconst, Sysinit, Typinfo, Windows, Consts, Forms, Dialogs, StdCtrls
```

```
Const n_n = 1; //n_n: Parameter, the number of maximum constraints for inequality or equality minus one
```

```
type
```

```
  TVector = array[0..n_n] of Double;
```

```
  PVector = ^TVector;
```

```
//dllfunction: Default name of function
```

```
//Para: Parameter passed from Auto2Fit
```

```
//ConFun1, ConFun2: Constraints for inequality and equality
```

```
//ObjFun: Value of objective function
```

```
//PassPara: Pass returned parameter
```

```
procedure dllfunction(para: pvector; var objfun: double; confun1, confun2, passpara: pvector); stdcall;
```

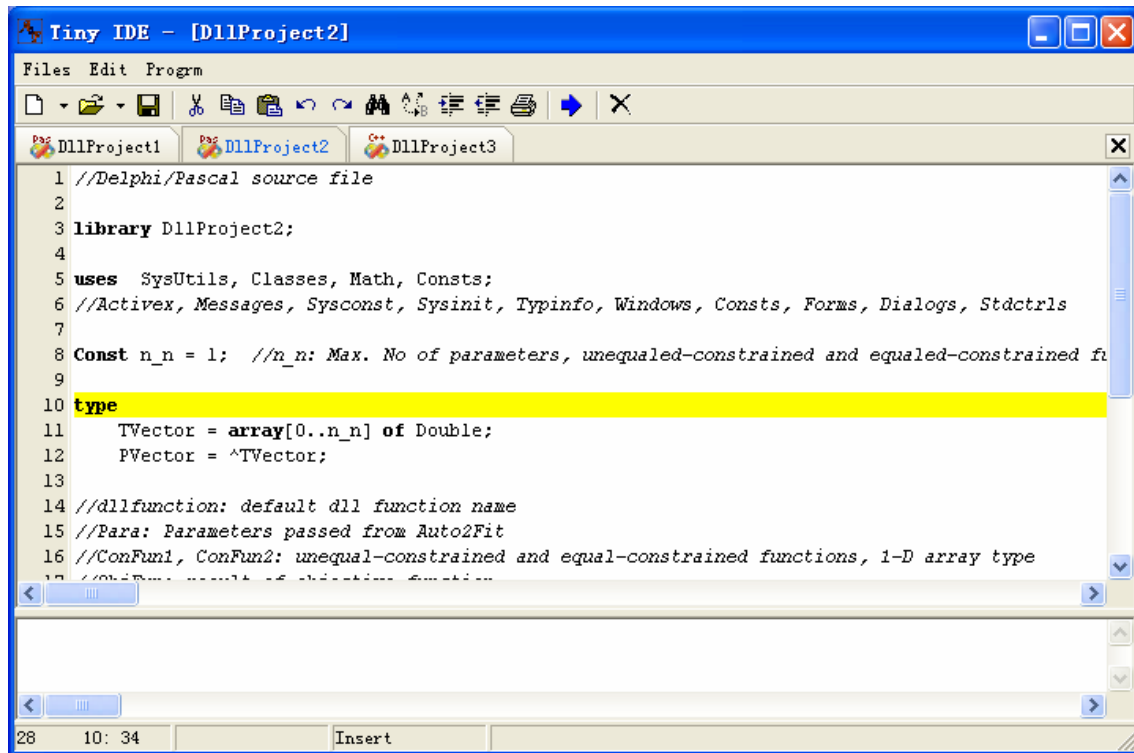
```
begin
```

```
end;
```

```
exports dllfunction;
```

```
begin
```

```
end.
```



Choose C++, the code will be generated like follows:

//C++ source file

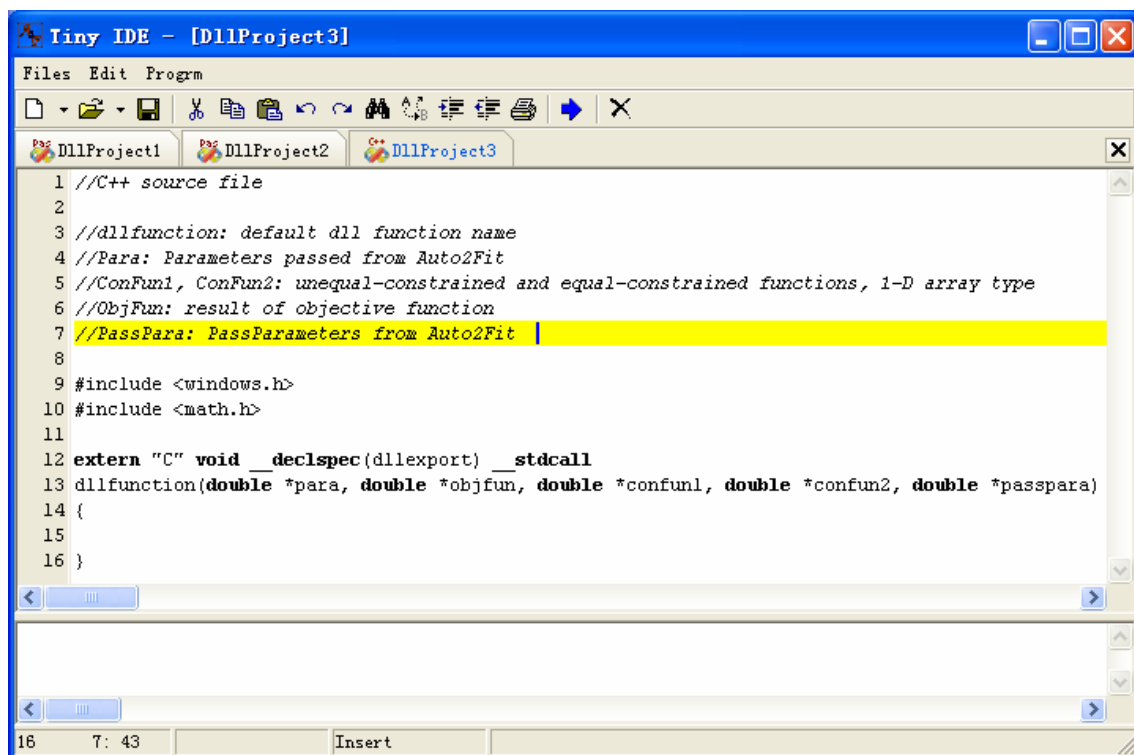
//dllfunction: Name of optional function
 //Para: Parameters passed from Auto2Fit
 //ConFun1, ConFun2: Constraints of inequality and equality
 //ObjFun: Value of objective function
 //PassPara: Pass returned parameter

#include <windows.h>
 #include <math.h>

```

extern "C" void __declspec(dllexport) __stdcall
dllfunction(double *para, double *objfun, double *confun1, double *confun2, double *passpara)
{
}

```

- 4) Generate Dll file: after filling objective and constrained functions, save file, if the setting of compiler is correct, press “compile”, the Dll file to be invoked by Auto2Fit will be generated.