

# 1st0pt 使用手册

七维高科

2012 年 2 月 3 日

## 目录

第一章 1stOpt 基础.....	4
1.1 1stOpt 简介 .....	4
1.1.2 1stOpt 特征 .....	5
1.1.3 1stOpt 计算模式 .....	7
1.2 1stOpt 工作环境.....	8
1.2.1 菜单.....	8
1.2.2 工具栏及设定 .....	10
1.2.3 导航窗口 .....	11
1.2.4 1stOpt 代码本 .....	11
1.2.5 代码本电子表格.....	13
1.2.6 优化算法及其它设定 .....	13
1.2.7 计算结果展示和获取 .....	15
1.2.8 二维-三维及预测 .....	17
1.3 1stOpt 电子表格 .....	17
1.4 1stOpt 关键字 .....	22
1.5 1stOpt 支持的数学函数 .....	25
1.6 基本语法 .....	27
第二章 1stOpt 功能.....	28
2.1 函数优化.....	28
2.1.1 一维函数优化.....	28
2.1.2 多维函数优化.....	28
2.1.3 隐函数优化.....	29
2.1.4 线性规划.....	30
2.1.5 非线性规划问题.....	32
2.1.6 排列组合优化.....	33
2.1.7 多目标函数优化.....	37
2.1.8 极大极小函数优化.....	39
2.1.9 与 Lingo 的比较 .....	40
2.1.10 与 Matlab 的比较 .....	43
2.2 非线性拟合 .....	46
2.2.1 共享模式拟合 .....	50
2.2.2 缺少变量值的特殊拟合 .....	51
2.2.3 批处理拟合 .....	53
2.2.4 权重拟合 .....	55
2.2.5 带约束拟合 .....	55
2.2.6 带积分的拟合 .....	57
2.2.7 最小一乘及其它特殊拟合 .....	58
2.2.8 隐函数拟合 .....	60
2.2.9 分峰拟合.....	63
2.2.10 StepReg 的使用 .....	67
2.2.11 QuickReg 与快速拟合 .....	68
2.2.12 递推公式拟合 .....	69
2.2.13 变系数拟合 .....	70

2.2.14	公式自动搜索拟合 .....	72
2.2.15	设定拟合初始取值范围 .....	73
2.2.16	常微分方程拟合 .....	74
2.2.17	复数拟合 .....	84
2.2.18	非线性问题线性化拟合 .....	88
2.2.18	与 Origin 的比较 .....	91
2.2.19	与 Lingo 的比较 .....	93
2.2.20	与其它数学软件的比较 .....	95
2.3	方程及方程组求解 .....	95
2.3.1	一般方程组求解 .....	95
2.3.2	循环方程求解 .....	96
2.3.3	循环递归方程求解 .....	96
2.3.4	整数方程求解 .....	97
2.3.4	复数方程求解 .....	98
2.4	常微分方程数值求解 .....	99
2.4.1	1stOpt 求常微分方程的关键字 .....	99
2.4.2	常微分方程初值问题 (Initial Value Problem – IVB) .....	100
2.4.3	隐式常微分方程及方程组 .....	102
2.4.4	变系数常微分方程 .....	103
2.4.5	常微分方程边值问题 (Boundary Value Problem – BVB) .....	108
2.4.6	特殊边值问题 .....	114
2.5	其它应用 .....	117
2.5.1	隐函数作图 .....	117
2.5.2	参数函数作图 .....	120
2.5.3	数据作图 .....	122
2.5.4	作为高级计算器使用 .....	125
2.5.5	使用脚本语言 .....	126
2.5.6	关键字 “PassParameter” 的使用 .....	129
2.5.7	关键字 “SubDivision” 的使用 .....	131
2.5.8	关键字 “PenaltyFactor” 的使用 .....	132
2.6	1stOpt 的编程模式 .....	132
2.6.1	约束函数优化问题 .....	133
2.6.2	时系列模型参数优化率定 .....	134
2.7	1stOpt 调用外部程序 .....	136
2.7.1	调用格式及关键字 .....	136
2.7.2	C++编译目标 Dll 文件 .....	138
2.7.3	Visual Fortran (VF) 编译目标 Dll 文件 .....	139
2.7.4	Gun Fortran 编译目标 Dll 文件 .....	141
2.7.5	Delphi 编译目标 Dll 文件 .....	144
2.7.6	PowerBasic 编译目标 Dll 文件 .....	148
2.7.7	Free Basic 编译目标 Dll 文件 .....	151
2.7.8	1stOpt 外部程序编辑器 (IDE) .....	152

# 第一章 1stOpt 基础

## 1.1 1stOpt 简介

1stOpt 名字源自英文 “First Optimization”，是国内目前所知首款通用数值优化仿真计算软件平台，在全局优化能力方面与当今世界上最优秀最流行的优化软件如 Lingo、GAMS 等相比，丝毫不居劣势，甚至表现更优；而在使用方便度方面，更胜一筹，初次使用该软件的用户，参照实例和说明，不需一小时就可基本掌握使用方法并解决自己的实际问题。

该软件平台拥有强大的全局优化能力、简洁易用的用户界面、可扩展的高级语言支持，可广泛应用于非线性回归，曲线拟合，非线性复杂模型参数估算求解，线性/非线性规划等各学科领域优化问题。该平台内含多种经典及现代优化算法如牛顿法、模拟退火、遗传算法、爬山法等；而其独创特有的通用全局优化算法 (Universal Global Optimization - UGO)，在保留经典局部算法快捷高效的同时，全局非线性寻优能力更是得到了极大地提升，对非线性优化问题，虽然还不能保证 100% 的全局成功求解率，但相比目前市场上流行的软件平台或算法已有了质的突破，具备了不依赖初值的特性，这使其在降低使用门槛的同时，还大大地提高了求解成功率。以非线性拟合为例，目前世界上在该领域最有名的软件工具包诸如 Matlab, OriginPro, SAS, SPSS, DataFit, GraphPad 等，均需用户提供适当的参数初始值以便计算能够收敛并找到最优解。如果设定的参数初始值不当则计算难以收敛，其结果是无法求得正确结果。而在实际应用当中，对大多数用户来说，给出(猜出)恰当的初始值是件相当困难的事，特别是在参数量较多的情况下，更无异于是场噩梦。而 1stOpt 凭借其超强的寻优，容错能力，在大多数情况下（大于 90%），从任意随机初始值开始，都能求得正确结果。

美国国家标准与技术研究院 (NIST) 提供有一套 27 道非线性拟合测试题，是公认的权威测试题集，世界上几乎所有著名的数据分析软件包，如 SAS、SPSS 等都以能通过该套测试题集为验证标准，而 1stOpt 是目前世界上唯一不依赖使用 NIST 提供的初始值，却能以任意随机初始值就可求得全部最优解的软件平台（如果使用 NIST 提供的初始值，则更可轻易求得最优解）。从此意义而言，1stOpt 的实用能力达业界领先水平。

此外，1stOpt 除了直接支持 Basic 和 Pascal 高级语言外，还可以与任意其它高级语言如 C++、Fortran 等联合使用，因而能够描述并解决十分复杂的工程优化问题。对终端用户，勿需要求掌握算法，不论求解问题的逻辑关系如何复杂，只需将问题的目标函数、约束函数等描述清楚即可，因而极大地方便了用户，此外由于这些高级语言都是编译性语言，因而计算速度也快。

1stOpt 推出时间不长，知名度远不及 Lingo 等世界知名软件，但在极短的时间内，已拥有了广大的用户群，国外如著名的英国牛津大学、美国斯坦福大学、美国能源部所属的橡树岭国家实验室 (Oak Ridge National Laboratory)、再生能源实验室 (National Renewable Energy Laboratory) 等，国内用户则几乎包括了大部分顶尖科研院所和知名企业，如中国科学院、国家地震局、航天二院、中国工程物理研究院、核九院、清华大学、北京大学、中

国科技大学、浙江大学、上海交大、南京大学等，著名企业如宝钢集团、中石油、中国船舶重工集团公司、中国铝业股份有限公司、大庆集团、霍尼韦尔公司等采用了 1stOpt 计算平台；而其研究领域更是涵盖了航空航天、军事、水利水电、能源、计算机信息、生物科学、社会经济、农业工程、环境等各方面，公开发表的论文已近千篇，国内外用户数以万计。

### 1.1.2 1stOpt 特征

一个优化软件好坏的评价标准主要包括三个方面：

- 1) 效果 (Effective)：优化计算结果是否正确（最优或接近最优），一种算法如果效果不好，即使效率再高再易于使用也毫无价值；
- 2) 效率 (Efficiency)：计算效率是否可接受，虽然计算机硬件功能有了长足进展，但对优化问题而言，计算时间仍是重要衡量指标；
- 3) 易用 (Easy for Use)：是否易于使用，软件平台设计的是否人性化、是否易于理解和使用将会严重影响软件的推广和进入门槛。

上述标准所占比重应该是相同的，各为 33%，1stOpt 软件平台应该说基本满足三者平衡的要求。

#### ✧ 1stOpt 内嵌的优化算法包括：

- 1) 通用全局优化算法 (Universal Global Optimization - UGO)；
- 2) 稳健全局优化算法 (Robust Global Optimization)；
- 3) 下山单体法 (Simplex Method - SM) + 通用全局优化算法 (Universal Global Optimization - UGO)；
- 4) 差分进化法 (Differential Evolution - DE)；
- 5) 遗传算法 (Genetic Algorithms - GA)；
- 6) 模拟退火 (Simulated Annealing - SA)；
- 7) 离子群法 (Particle Swarm Optimization - PSO)；
- 8) 最大继承法 (Max Inherit Optimization - MIO)；
- 9) 自组织群移法 (Self-Organizing Migrating Algorithms - SOMA)；
- 10) 禁忌搜索法 (Tabu Search - TS)；
- 11) 单纯线性规划法 (Simplex Linear Programming)。

#### ✧ 1stOpt 应用范围：

- 1) 模型自动优化率定；
- 2) 参数估算；
- 3) 任意模型公式线性，非线性拟合，回归；
- 4) 非线性连立方程组求解；
- 5) 常微分方程及方程组，初值和边值问题；
- 6) 常微分方程拟合；
- 7) 复数类型拟合及复数方程组求解；
- 8) 任意维函数，隐函数极值求解；

- 9) 隐函数根求解，作图，求极值；
- 10) 线性，非线性及整数规划；
- 11) 组合优化问题；
- 12) 高级计算器。

#### ☆ 1stOpt 特长

- 1) 模型采用自然描述语言，简单易懂，学习周期短
- 2) 线性、非线性、混合整数规划、二次规划、优化组合
- 3) 功能强劲，是目前唯一能以任何初始值而求得美国国家标准与技术研究院 (NIST: National Institute of Standards and Technology) 非线性回归测试题集最优解的软件包。
- 4) 可广泛用于水文水资源及其它工程模型优化计算。内嵌 VB 及 Pascal 语言，可帮助描述处理复杂模型。
- 5) 可连接由任何语言 (C++, Fortran, Basic, Pascal...) 编译而成的外部目标函数动态连接库或命令行可执行文件。
- 6) 独特的隐含数优化、拟合，智能拟合、带约束的拟合功能
- 7) 非线性曲线拟合可处理任意类型模型公式，任意多数目的待求参数及变量
- 8) 直接支持微分方程和复数方程拟合
- 9) 模型自动率定时可同时处理多个数据文件
- 10) 可非常容易处理一些特殊的参数，如降雨径流模型中的流域初期土壤含水量
- 11) 可同时处理多个输出量
- 12) 实时显示计算结果
- 13) 可直接读存 Excel, CSV 等格式文件
- 14) 界面简单友好，使用方便
- 15) 自带有上百个实例，覆盖范围包括几乎所有优化方面。通过不同类型实例，用户可轻松掌握 1stOpt 的用法。

#### ☆ 1stOpt 系统要求

- 1) 操作系统: Win98/WinMe/Win2000/WinXP/WinVista/Win7
- 2) 硬盘空间: 50M
- 3) 内存: 512M 以上

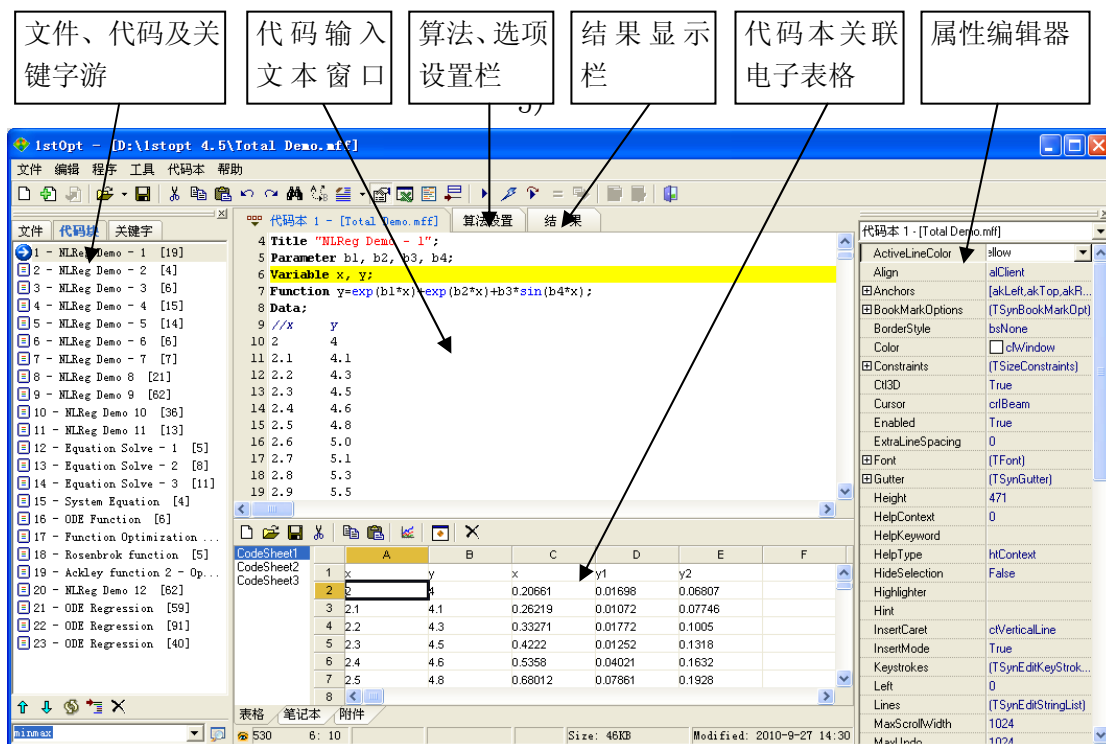


图 1.1 1stOpt 主界面

### 1.1.3 1stOpt 计算模式

1stOpt 的计算模式可分为三个层次：快捷模式、编程模式和混编模式。快捷模式使用类自然描述语言，直观、简单易学，约 80% 的优化问题可用这种方式解决；编程模式直接支持 Basic 和 Pascal 两种高级编程语言，可描述处理较为复杂的问题的优化问题，所占比例约 15%；剩余 5% 的问题可通过混编模式，即与任一高级语言混合编程，可处理任意大型复杂优化问题。

## 1.2 1stOpt 工作环境

1stOpt 主界面如图 1.1 示，主要包括主菜单、工具栏、文件/代码/关键字游览窗口、代码输入文本窗口、算法及选项设置页面、结果显示页面、代码本关联电子表格和属性编辑器等。

## 1.2 1stOpt 工作环境

### 1.2.1 菜单

1stOpt 是标准的 Windows 应用程序，通过菜单能够实现几乎所有功能。

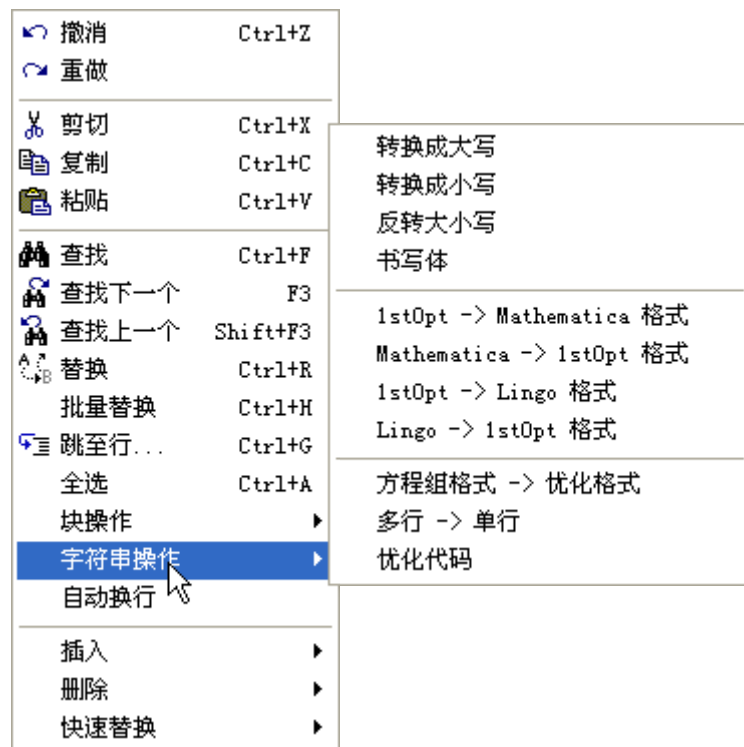
✧ 主菜单： 文件 编辑 程序 工具 代码本 帮助

✧ “文件”子菜单：



✧ “编辑”子菜单：





✧ “程序”子菜单：



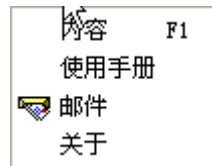
✧ “工具”子菜单：



✧ “代码本”子菜单：



◇ “帮助”子菜单：



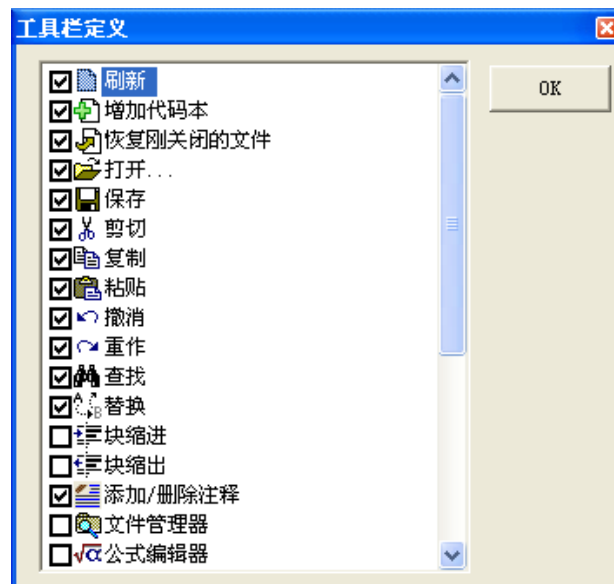
## 1.2.2 工具栏及设定

◇ 工具栏：



◇ 工具栏设定：

在工具栏上点击右键，选“自定义”，弹出如下图工具栏图标设定窗口，一旦设定，下次启动时会自动记住前次设定



### 1.2.3 导航窗口

文件-代码块-关键字导航窗口用于快速浏览选择文件、代码块以及关键字和数学函数，如图 1-2 和 1-3 示。该导航窗口可关闭，通过主菜单或快捷组合键“Ctrl + B”也可重新激活。

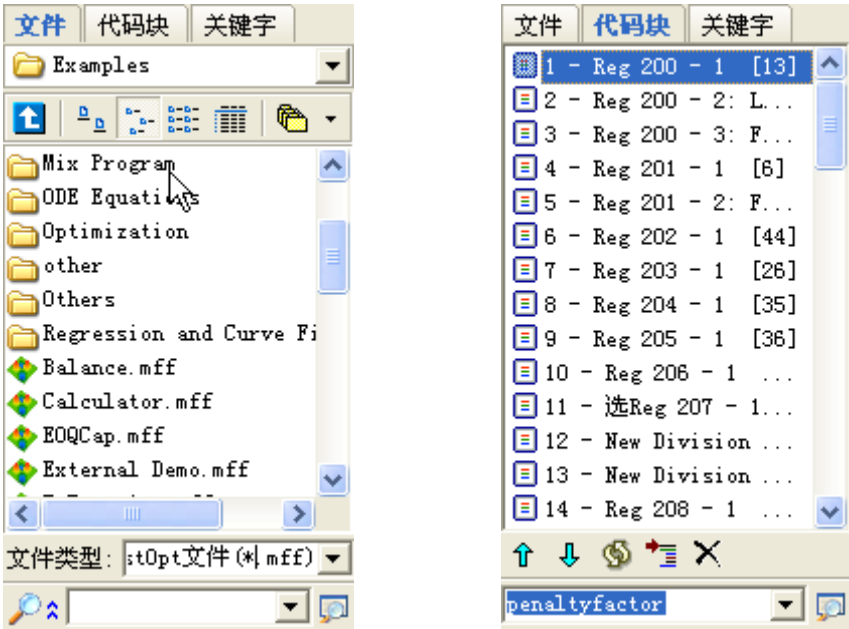


图 1-2 文件及代码块导航窗口



图 1-3 关键字及数学函数导航窗口

### 1.2.4 1stOpt 代码本

代码本是 1stOpt 的主要工作区域，问题代码、数据等均在此输入。在同一代码本中可写多个不同问题的代码，由关键词“NewDivision”来区分。可同时开启多个代码编辑本。同一代码文件中还可加入富文本如图，表，公式等，也可把不同格式的文件添附进来。另外需注意 1stOpt 不区分大小写。

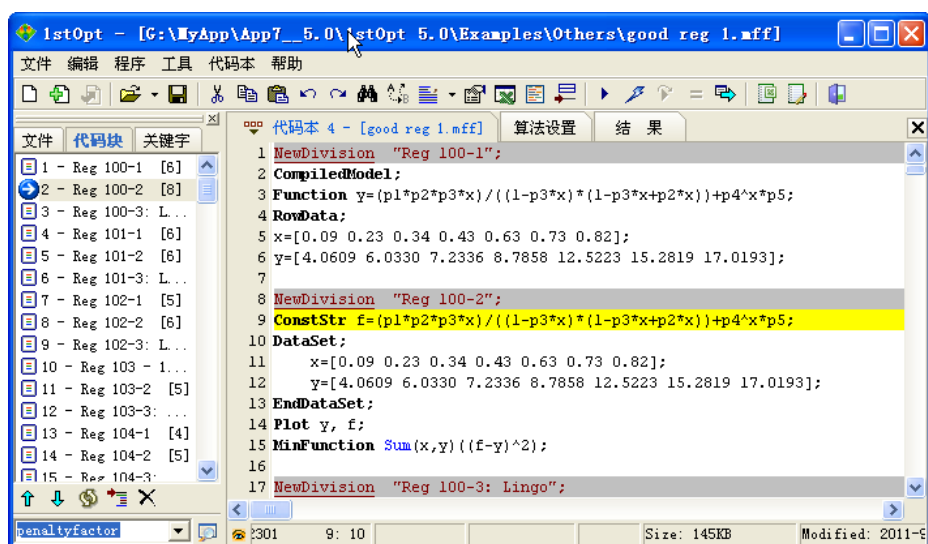
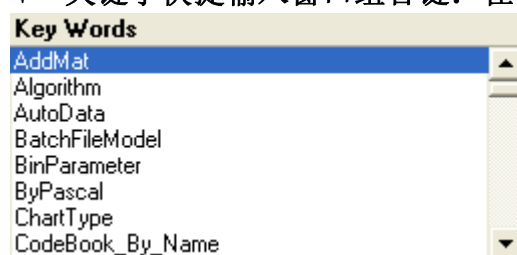


图 1-4 1stOpt 代码本

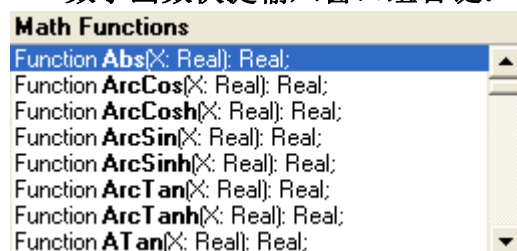
代码本中主要快捷组合键

✧ 关键字快捷输入窗口组合键：在代码本中按“Ctrl+K”



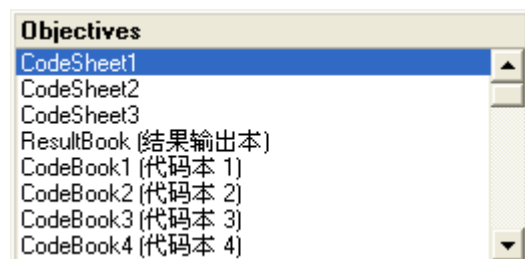
窗口弹出后，按顺序输入关键字字母，可快速查找并输入所需关键字

✧ 数学函数快捷输入窗口组合键：在代码本中按“Ctrl+M”



窗口弹出后，按顺序输入数学函数字母，可快速查找并输入所需数学函数

✧ 代码表格、代码本快捷输入窗口组合键：在代码本中按“Ctrl+J”



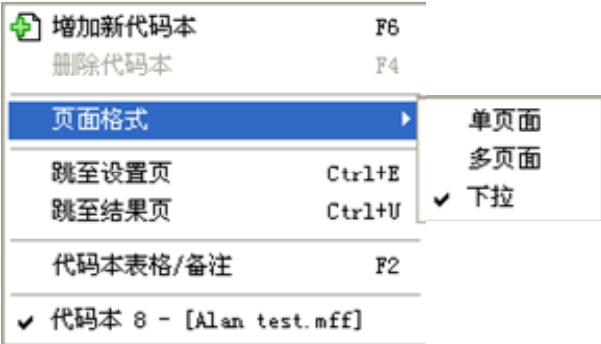
窗口弹出后，可选择所需代码表格或代码本。代码表格列表名不同于表格名时，括号内为表格列表名，此时应选择其对应的表格名

✧ 恢复上一次执行的代码：在代码本中按“Ctrl+Shift+T”

有时或许由于代码输错，或许由于其它不明原因，当输完代码按计算命令时，1stOpt出错而自动退出，如果先前没有进行保存，该如何恢复刚才输入的代码呢？重新启动 1stOpt，开启一新代码本，在代码本中按“Ctrl+Shift+T”即可恢复上一次执行的代码。

关键字和数学函数也可通过导航窗口选择并双击来输入。

代码本有三种排列方式，单页面、多页面和下拉式，可通过主菜单“代码本”→“页面格式”来设定。



## 1.2.5 代码本电子表格

代码本电子表格功能类似于 1.3 节电子表格，其数据可直接在代码本中调用，保存文档时，数据也存于同一文档。

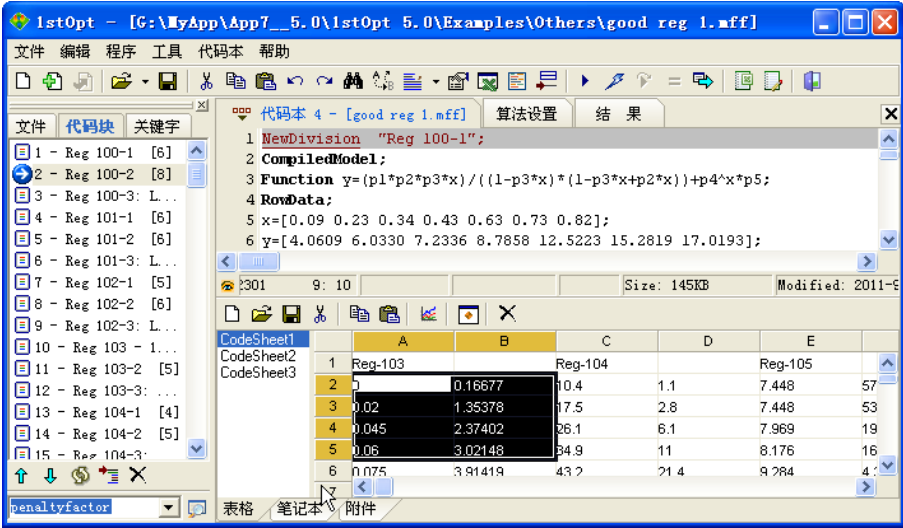


图 1-5 1stOpt 内嵌关联电子表格

## 1.2.6 优化算法及其它设定

### ☆ 优化算法设定

在 1stOpt 中，共有 11 种优化算法供选择。不同的问题该选用何种算法？一般而言：

- 非线性回归、曲线拟合问题、方程及方程组求解、无约束函数优化：
  - 1) 通用全局优化算法(Universal Global Optimization - UGO)
  - 2) 稳健全局优化算法(Robust Global Optimization)
  - 3) 下山单体法 (Simplex Method - SM) + 通用全局优化算法(Universal Global Optimization - UGO)
  - 4) 差分进化法
  - 5) 最大继承法

- 有约束函数优化问题：
  - 1) 下山单纯法 (Simplex Method - SM) + 通用全局优化算法 (Universal Global Optimization - UGO)
  - 2) 差分进化法
  - 3) 通用全局优化算法 (Universal Global Optimization - UGO)
  - 4) 最大继承法
- 线性规划问题：
  - 1) 单纯线性规划法 (Simplex Linear Program - SLP)
  - 2) 下山单纯法 (Standard Simplex Method - SM) + 通用全局优化算法 (Universal Global Optimization - UGO)
  - 3) 差分进化法
- 优化组合问题：
  - 1) 最大继承法
  - 2) 禁忌搜索法
  - 3) 模拟退火
  - 4) 遗传算法

线性规划问题一般不推荐使用非线性全局优化算法去求解，虽然从理论上讲，全局优化算法可解决任何线性规划问题，但在实际应用当中，会出现“牛刀”无法“杀鸡”的现象。此外不同算法对应

有相关的运行参数，绝大多数情况下，缺省状态值都可满足求解需求，个别情况需要根据经验和实际问题进行单独调整。

### ✧ 选项设置一



图 1-6 优化算法设定



图 1-7 选项设置一

- 实时更新间隔：迭代计算结果显示间隔；
- 罚函数系数：运用于控制约束优化问题的系数，关键字“PenaltyFactor”；
- 快速拟合：大规模数据拟合时加快计算速度，关键字“QuickReg”；
- 计算停止指标：迭代计算自动终止判断指标；
- 参数边界控制：对参数有边界限制或约束时起作用，关键字“EnhancedBound”；
- 多重运算：设定自动多次运算，并保留最佳及每次计算结果，关键字“MultiRun”；
- 微分方程选项：设定微分方程求解控制参数，包括算法和步长等，“种群数”仅应用于初值（IVP）隐式或边值（BVP）微分方程，代码本中可通过关键字“ODEOptions”来设定。



## ✧ 选项设置二



图 1-8 选项设置二

- 日志文件自动保存：1stOpt 每一次运行前都会保存一个副本至“Log Files”目录中，文件后缀“.log”，该选项决定是否保存、保存总数等；
- 结果保存：实时计算结果保存；
- 参数保存：实时计算参数保存。

## 1.2.7 计算结果展示和获取

计算结果可实时显示，计算完毕后，结果可以保存为文件，也可通过右键弹出菜单选“数据报表”或工具栏“数据报表”“结果报表”，可获得更整洁和易于理解的数据表格

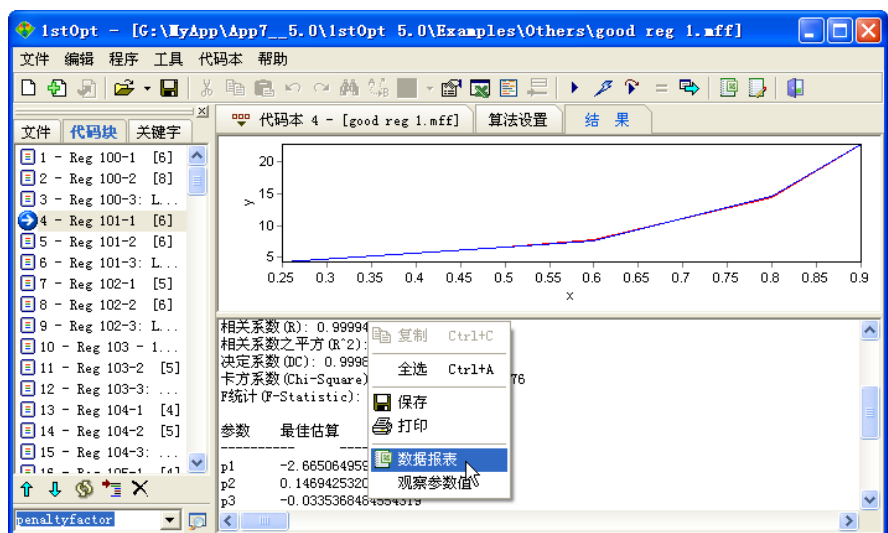


图 1-9 计算结果显示页面

Figure 1-10 shows the '1stOpt' application window displaying a spreadsheet of calculation results. The spreadsheet has columns A, B, C, D, and E. The data includes target values, calculated values, parameter names, and parameter values.

目标y	计算y	参数名	参数值
4.28	4.282677534985	p1	-2.66506495905871
6.63	6.557417267513	p2	0.146942532046548
7.61	7.728759539304	p3	-0.0335368484554319
14.55	14.4862015813	p4	711004.318457527
22.77	22.79065842807		
		均方差(RMSE)	0.0691031971734169
		残差平方和(SSE)	0.0238762592979407
		相关系数(R)	0.999947401869102
		相关系数之平方(R^2)	0.999894806504768
		最大绝对值差	0.11875953930472
		最小绝对值差	0.00267753498556278
		最大相对绝对值差	0.0156057213278213
		最小相对绝对值差	0.00062559228634651

图 1-10 计算结果数据表格显示

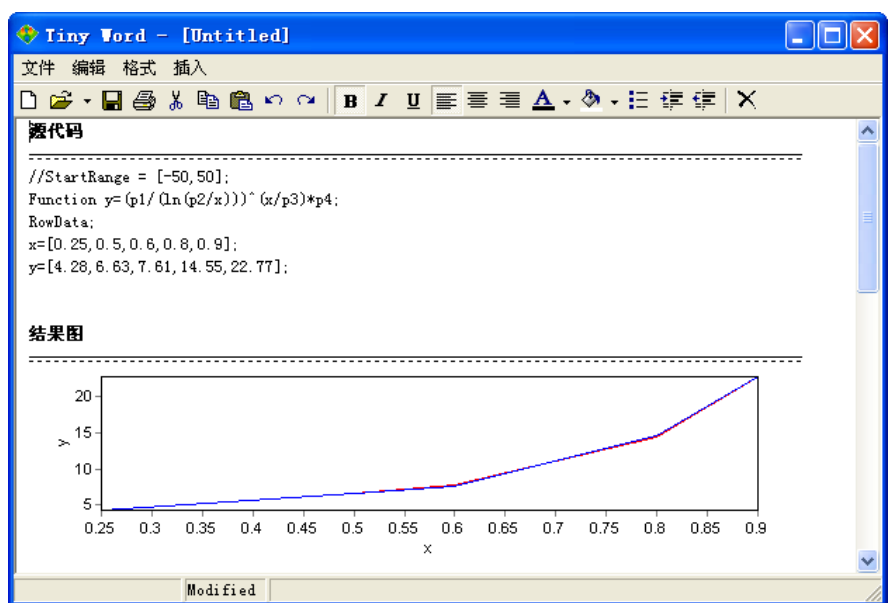


图 1-11 计算结果富文本编辑



## 1.2.8 二维-三维及预测

二维-三维及预测是以非常实用的功能，主要用于优化计算完成后验证、预测、再现、由因变量反求自变量、二维-三维图形分析等，其具体应用可参考后面 1stOpt 应用篇章。

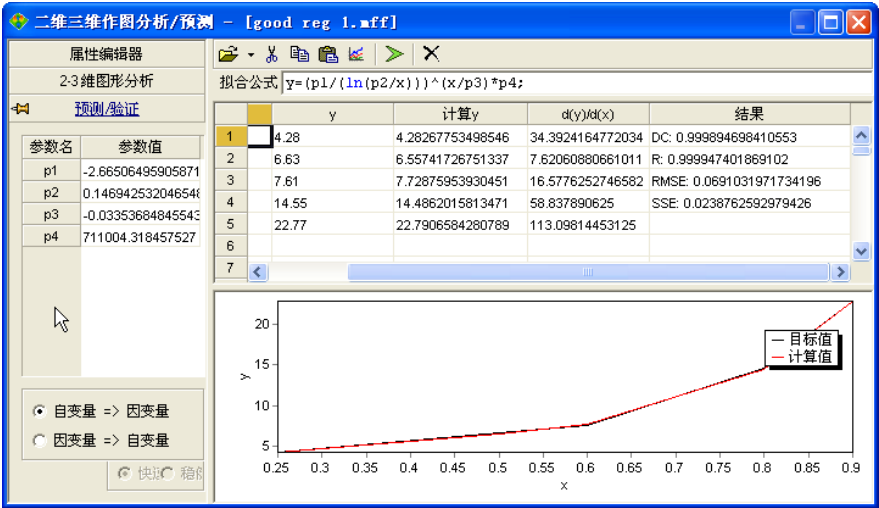


图 1.12 曲线拟合验证和预测

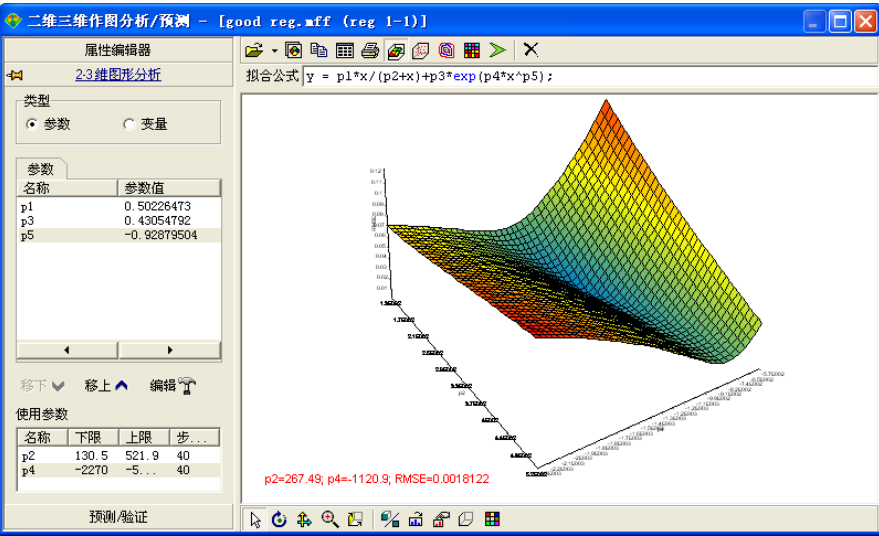


图 1.13 参数三维图形分析

## 1.3 1stOpt 电子表格

1stOpt 附带的电子表格，虽不如 Excel 般强大，但亦基本可满足数据前后处理、显示、作图等，有自己一定的特色：

- 可直接读写 Excel、CSV 和 Txt 格式文件；
- 支持 Windows 粘贴板功能，可和其它软件无缝数据交换；
- 列式多页面；
- 支持表格内公式计算；
- 脚本语言支持，可从代码本编写脚本直接操作表格内容；

- 与代码本内嵌电子表格无缝相连
- 计算结果报表
- 一些特有的数据处理功能

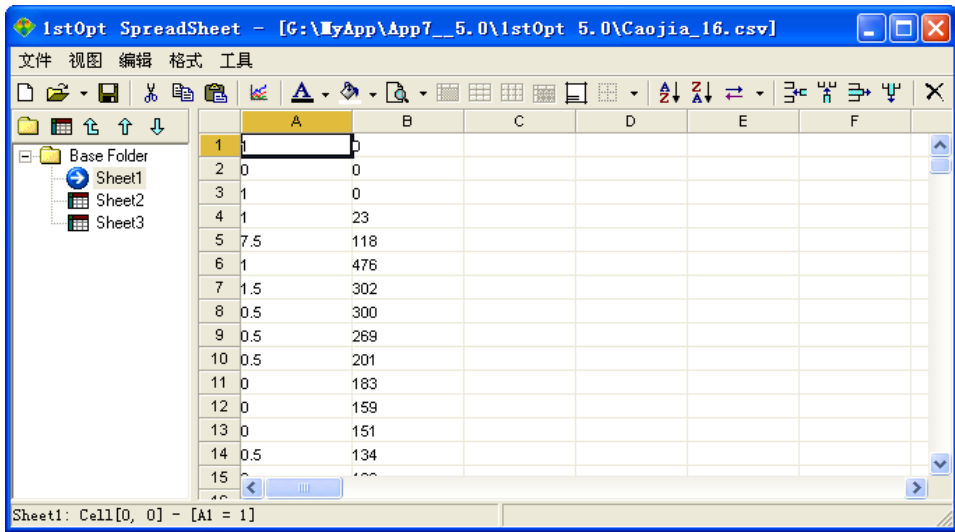


图 1.14 电子表格界面

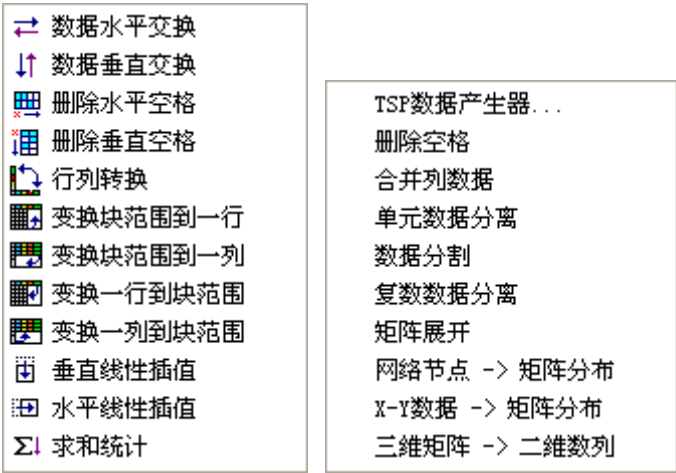


图 1.15 数据处理及特殊工具

### ✧ TSP 数据产生器

旅行商问题（TSP）数据产生器，可随机生成任意城市数目的 TSP 数据，包括矩阵和坐标两种形式。

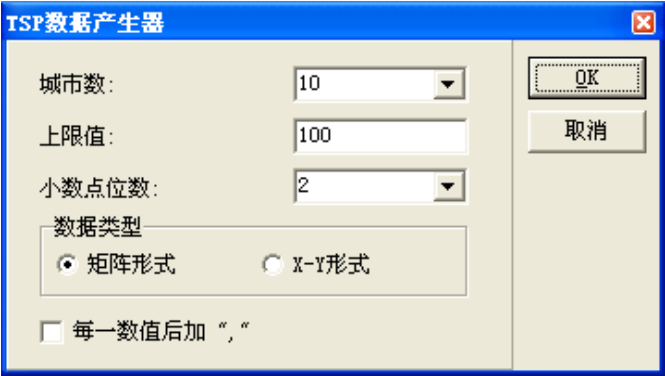


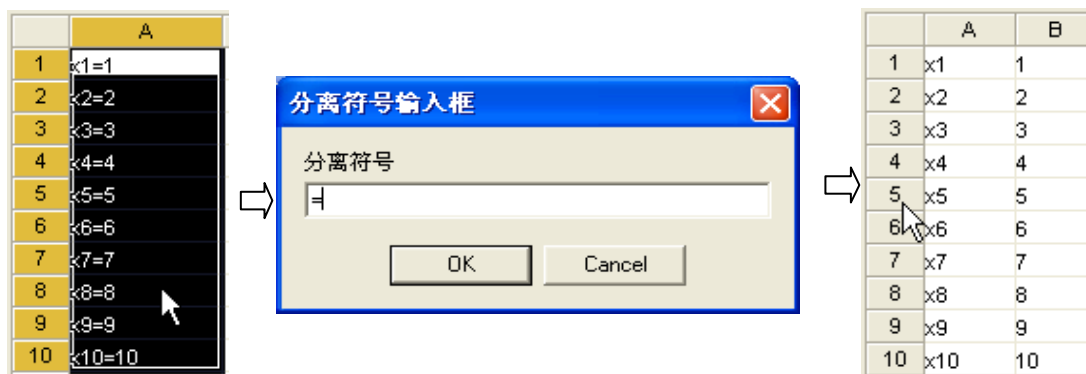
图 1.16 TSP 数据产生器参数设定

	A	B	C	D	E	F	G	H	I	J
1	0	42.51	21.35	8.45	53.87	81.61	4.44	69.34	22.05	24.33
2	42.51	0	42.69	61.49	75.43	95.43	23.84	18.32	29.68	90.82
3	21.35	42.69	0	90.12	39.78	3.73	76.19	0.46	47.34	13.51
4	8.45	61.49	90.12	0	49.59	11.64	33.17	36.90	59.74	39.12
5	53.87	75.43	39.78	49.59	0	8.41	73.30	97.68	83.68	79.85
6	81.61	95.43	3.73	11.64	8.41	0	46.21	3.67	34.68	10.05
7	4.44	23.84	76.19	33.17	73.30	46.21	0	22.43	56.55	85.90
8	69.34	18.32	0.46	36.90	97.68	3.67	22.43	0	40.40	46.11
9	22.05	29.68	47.34	59.74	83.68	34.68	56.55	40.40	0	37.94
10	24.33	90.82	13.51	39.12	79.85	10.05	85.90	46.11	37.94	0

图 1.17 TSP 数据产生结果

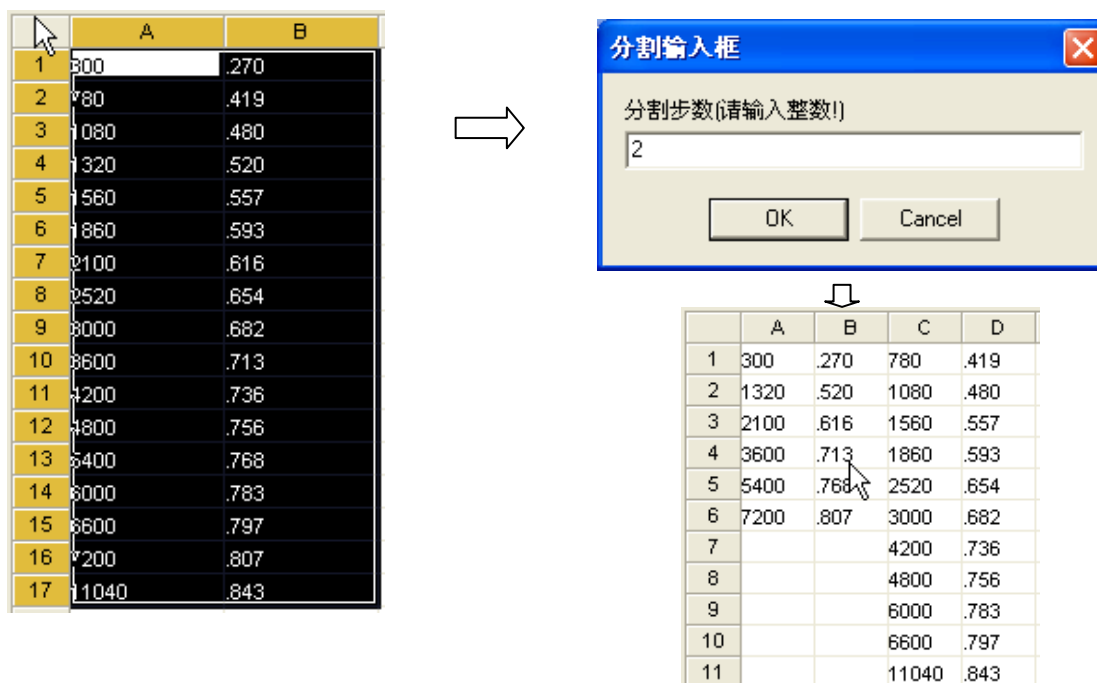
### ✧ 单元数据分离

将选择数据按指定符号进行分离。



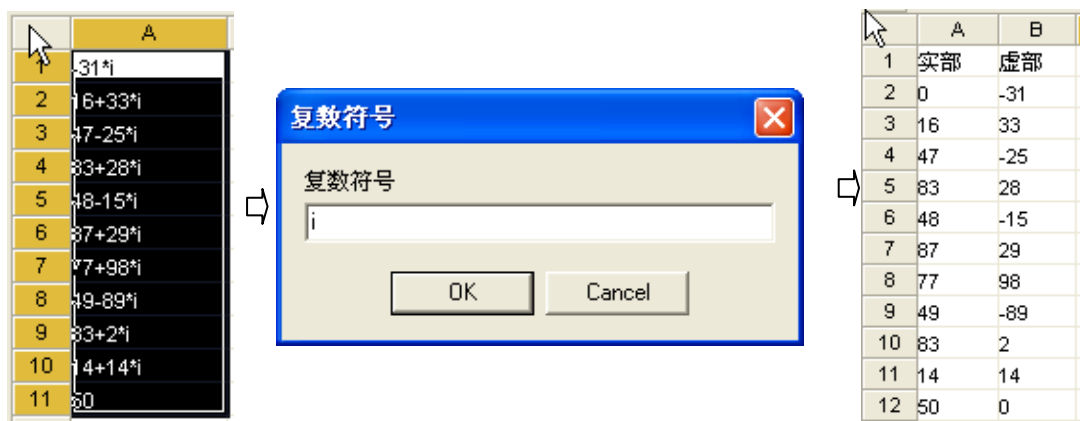
### ✧ 数据分割

将长系列数据按给定步长分割形成两个单独较短系列。



### ✧ 复数数据分离

工具定义的复数符号将复数的实虚部分离



### ◇ 矩阵展开

选择两个矩阵将其相乘展开

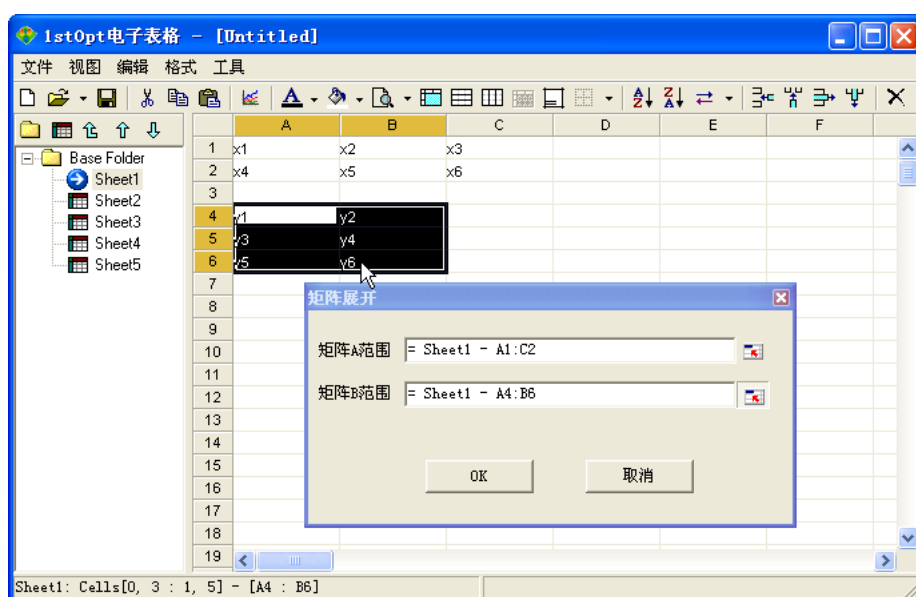


图 1.18 选择两个矩阵

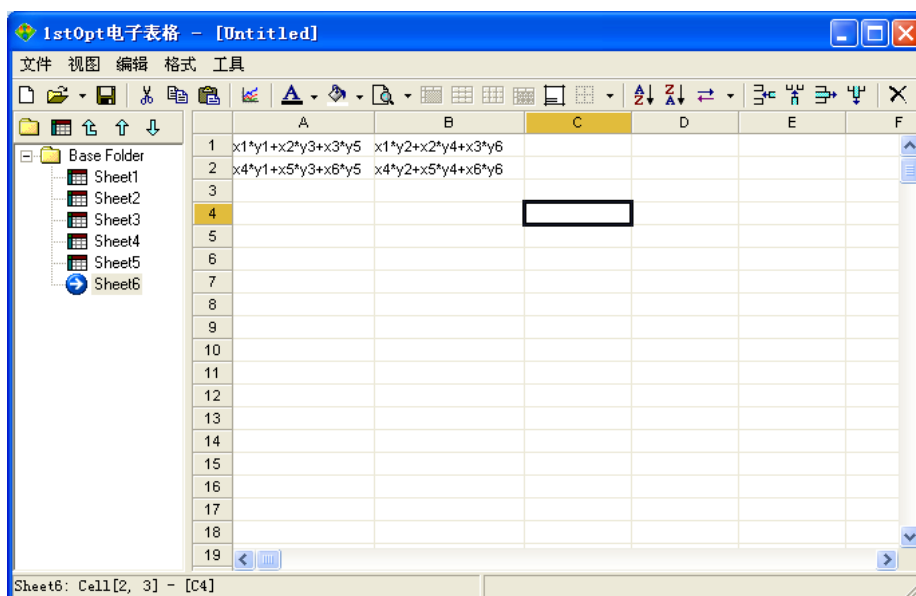


图 1.19 矩阵展开结果

### ◇ 网络节点至矩阵分布转换

如下图网络节点，节点间有的相连而有的没有连接，连接点数据如下：

表 1.1 节点数据

节点号	距离
0 1	34
0 2	38
1 3	63
1 4	34
2 3	91
2 5	9
3 6	62
4 6	65
5 6	26

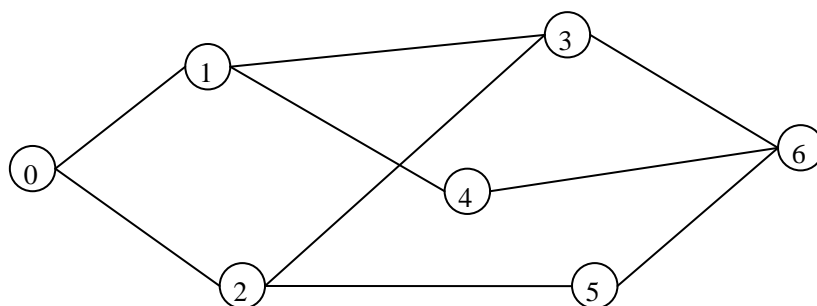


图 1.20 节点网络图

在表格中选择数据，再输入非连接点间的数据，即可进行转换。

	A	B	C
1		1	34
2		2	38
3		3	63
4		4	34
5		3	91
6		5	9
7		6	62
8		6	65
9		6	26



非连接符号

请输入符号)

1

OK Cancel



	A	B	C	D	E	F	G	H
1		0	1	2	3	4	5	6
2	0	0	34	38	-1	-1	-1	-1
3	1	34	0	-1	63	34	-1	-1
4	2	38	-1	0	91	-1	9	-1
5	3	-1	63	91	0	-1	-1	62
6	4	-1	34	-1	-1	0	-1	65
7	5	-1	-1	9	-1	-1	0	26
8	6	-1	-1	-1	62	65	26	0

### ◇ X-Y 数据转换成矩阵分布

将 x-y 坐标数据转换为矩阵形式。

	A	B
1	x	y
2	3.3	14.1
3	5.0	5.5
4	5.1	70.2
5	8.9	13.7

	A	B	C	D	E
1		1	2	3	4
2	1	0	9.21	56.19	0.72
3	2	9.21	0	64.70	9.08
4	3	56.19	64.70	0	56.63
5	4	0.72	9.08	56.63	0

### ◇ 三维矩阵转换成二维数列

	A	B	C	D	E
1		1.1	2.2	3.3	4.4
2		6.70	7.53	7.09	8.43
3		1.02	7.55	1.09	0.55
4		2.93	9.23	4.64	9.66
5		6.38	7.33	4.89	2.40

	A	B	C
1	1	1.1	6.70
2	2	1.1	1.02
3	3	1.1	2.93
4	4	1.1	6.38
5	1	2.2	7.53
6	2	2.2	7.55
7	3	2.2	9.23
8	4	2.2	7.33
9	1	3.3	7.09
10	2	3.3	1.09
11	3	3.3	4.64
12	4	3.3	4.89
13	1	4.4	8.43
14	2	4.4	0.55
15	3	4.4	9.66
16	4	4.4	2.40

## 1.4 1stOpt 关键字

### 主要关键词

关键词名	意义及示例
Parameter	定义参数 例： 定义 a, b, c, d 四个参数： <code>Parameter a, b, c, d;</code> 例： 定义 a1, a2, a3, a4, a5, a6, a7, a8, a9, a10 十个参数： <code>Parameter a1, a2, a3, a4, a5, a6, a7, a8, a9, a10;</code> 也可简写为： <code>Parameter a(1:10);</code> 或： <code>Parameter a(10);</code> 例： 定义参数 a，其取值范围在[-1, 1]，初始值为 0.5 <code>Parameter a = 0.5[-1,1];</code> 例： 定义参数 a 为整数，其取值范围在[-100, 100] <code>Parameter a=[-100,100,0];</code> 例： 定义参数 a、b、c，取值范围在[-1, 1] <code>Parameter -1&lt;=[a,b,c]&lt;=1;</code>
ParameterDomain	批量定义参数范围 例： 定义参数 a 范围为[-1, 1]，其它均为[0, 10]； <code>Parameter a = [-1, 1], b, c, d, e, f, g;</code> <code>ParameterDomain = [0, 10];</code>
BinParameter	定义 0-1 参数 例： 定义 a 为非 0 即 1 的参数 <code>BinParameter a;</code>
IntParameter	定义正整数参数 例： 定义参数 a 为大于 0 的正整数 <code>IntParameter a;</code>
StartRange	定义参数初始值取值范围 例： 定义参数 a 初始值取值范围为[-100, 100] <code>StartRange a = [-100, 100];</code>
Variable	定义变量 例： 定义 x, y, z 三个变量： <code>Variable x, y, z;</code>
QuickReg	设定快速拟合功能
Function	定义函数 例： 两变量曲线拟合： <code>Function y = a + b*exp(c - x);</code> 例： 两变量函数优化： <code>Function (x+((2-x)*(2+y))^2)*sin(x*y);</code>

Constant	定义常量 例: Constant a=1, b=2; 例: Constant p(3)=[1, 2, 3];
ConstStr	定义常字符串量 例: 两变量曲线拟合: Function y = a* (c-x) ^2 + b*exp((c - x)^4); 可写为: ConstStr B = (c-x) ^2 Function y = a*B + b*exp(B^2);
VarConstant	定义变常量
VarParameter	定义变参数
ComplexPar	定义复数型参数
ComplexStr	定义复数符号, 缺省为 i
ComplexType	复数方程求解模式, 0: 实虚部均为 0; 1: 仅实部为 0; 2: 仅虚部为 0. 缺省值为 0
realPart	复数拟合时定义实数变量部分
imagPart	复数拟合时定义虚数变量部分
ODEFunction	定义常微分方程 例: ODEFunction y''''=1/(1-y)^2;
ODEAlgorithm	定义常微分方程计算方法, 包括龙格-库塔-费尔博格法(RKF45)、龙格-库塔 1 至 5 阶 (RK1、RK2、RK3、RK4、RK5)
ODEOptions	微分方程求解选项 例: ODEOptions = [SN=10,A=0,P=5]; SN: 求解步数; A: 算法, 0 表示 RKF45 法; P: 种群数, 仅对边值问题有效
InitialODEValue	微分方程拟合时定义初始值
Data	定义数据开始
RowData	数据以行形式出现
DataFile	定义数据文件
NewDivision	定义新的代码块
SubDivision	定义子代码块
StartProgram	编程模式开始
EndProgram	编程模式结束
Maximum	求最大值
Minimum	求最小值
MinMax	求极大极小函数
PlotFunction	画函数图
Algorithms	定义优化方法
Exclusive	定义问题为排它问题, 如 TSP 问题
MutliRun	定义多次自动计算数
HotRun	定义自动热计算数
SharedModel	定义共享参数问题
DataSet EndDataSet	定义常数 结束定义常数
MinFunction	最小值求优
MaxFunction	最大值求优
PlotParaFunction	画参数方程函数图
Title	定义代码块名
RegType	设定最小一乘法拟合

MDataSet	设定网络节点数据格式，等同于矩阵格式
EndMDataSet	
ConstrainedResult	编程模式中约束函数值
ObjectiveResult	编程模式中目标函数值
BatchFileModel	批处理文件模式
LoopConstant	定义循环常数
FullLoopModel	全循环计算模式，对应 LoopConstant
EnhancedBound	参数有边界限制时进行强化处理
SharedModel	多函数拟合时共享模式
WeightedReg	权重拟合
StepReg	逐步拟合模式
ExeParameterFile	调用外部命令行执行文件时，定义参数输出文件
ExeObjectiveFile	调用外部命令行执行文件时，定义目标函数值输出文件
MaxIteration	最大迭代数

1stOpt 还有三个特殊定义符：

#### ✧ 求和定义 Sum

如  $\sum_{i=1}^n (x_i \cdot \sin(x_i + 1))$ ，在 1stOpt 中表达为：Sum(i=1:n) (x[i]\*sin(x[i]+1))

如果下标号均为 i，上述也可简写为：Sum(i=1:n, x) (x\*sin(x+1))

如果下标号 i 起始值为 1，还可简写为：Sum(i=n, x) (x\*sin(x+1))

可多重定义，如

$$\sum_{i=1}^n \sum_{j=i}^n (x_i \sin(x_i + x_j))$$

在 1stOpt 中可写为：Sum(i=1:n) (Sum(j=i:n) (x[i]\*sin(x[i]+x[j]))));

#### ✧ 求积定义 Prod

如  $\prod_{i=1}^n (x_i \cdot \sin(x_i + 1))$ ，在 1stOpt 中表达为：Prod(i=1:n) (x[i]\*sin(x[i]+1))

如果下标号均为 i，上述也可简写为：Prod(i=1:n, x) (x\*sin(x+1))

如果下标号 i 起始值为 1，还可简写为：Prod(i=n, x) (x\*sin(x+1))

#### ✧ 循环符 For

如方程组：

$$\begin{cases} x_1 \sin(x_1) - 1 + x_1 = 0 \\ x_2 \sin(x_2) - 2 + x_2 = 0 \\ x_3 \sin(x_3) - 3 + x_3 = 0 \\ x_4 \sin(x_4) - 4 + x_4 = 0 \\ x_5 \sin(x_5) - 5 + x_5 = 0 \end{cases}$$

1stOpt 代码：

```
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;
```

用 For 语句，简写如下：

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

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



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

## 1.5 1stOpt 支持的数学函数

### 实数函数

序号	函数	说明	例
1	Abs(X: Real): Real;	绝对值函数	Abs(-0.25) = 0.25
2	Arccos(X: Real): Real;	反余弦函数	Arccos(-0.25) = 1.823476582
3	Arccosh(X: Real): Real;	反余弦双曲函数	Arccosh(-0.25) = 0
4	Arcsin(X: Real): Real;	反正弦函数	Arcsin(-0.25) = -0.2526802551
5	Arcsinh(X: Real): Real;	反正弦双曲函数	Arcsinh(-0.25) = -0.247466461
6	Arctan(X: Real): Real;	反正切函数	Arctan(-0.25) = -0.2449786631
7	Arctanh(X: Real): Real;	反正切双曲函数	Arctanh(-0.25) = -0.255412811
8	ATan(X: Real): Real;	反正切函数	ATan(-0.25) = -0.2449786631
9	ATand(X: Real): Real;		ATand(-0.25) = -0.0043632954
10	BessI(N: Integer, X: Real): Real;	贝塞尔 I 型函数	BessI(2, 0.25) = 0.0078532695
11	BessI0(X: Real): Real;	贝塞尔 I0 型函数	BessI0(0.25) = 1.015686133
12	BessI1(X: Real): Real;	贝塞尔 I1 型函数	BessI1(0.25) = 0.1259791086
13	BessJ(N: Integer, X: Real): Real;	贝塞尔 J 型函数	BessJ(2, 0.25) = 0.0077718892
14	BessJ0(X: Real): Real;	贝塞尔 J0 型函数	BessJ0(0.25) = 0.9844359314
15	BessJ1(X: Real): Real;	贝塞尔 J1 型函数	BessJ1(0.25) = 0.1240259773
16	BessK(N: Integer, X: Real): Real;	贝塞尔 K 型函数	BessK(2, 0.25) = 31.51771458
17	BessK0(X: Real): Real;	贝塞尔 K0 型函数	BessK0(0.25) = 1.541506736
18	BessK1(X: Real): Real;	贝塞尔 K1 型函数	BessK1(0.25) = 3.747025981
19	BessY(N: Integer, X: Real): Real;	贝塞尔 Y 型函数	BessY(2, 0.25) = -20.70126879
20	BessY0(X: Real): Real;	贝塞尔 Y0 型函数	BessY0(0.25) = -0.9315730315
21	BessY1(X: Real): Real;	贝塞尔 Y1 型函数	BessY1(0.25) = -2.704105228
22	Beta(X1: Real; X2: Real): Real;	贝塔函数	Beta(0.3, 0.4) = 5.112091244
23	BetaCDF( X: Real; A[>0]: Real; B[>0]: Real): Real;	贝塔密度累积分布函数	Betacdf(0.6, 0.3, 0.7) = 0.7773849481
24	BetaPDF( X: Real; A[>0]: Real; B[>0]: Real): Real;	贝塔密度分布函数	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;	二项式分布累积函数	BinomialCdf (0.3, 2, 0.5) = 0.25
27	BinomialPDF(n1: Integer; n2[<n1]: Integer; X[0,1]: Real): Real;	二项式分布密度函数	BinomialPdf(0.4, 6, 0.3) = 0.117649
28	ChiSquareCDF(X: Real; n: Integer): Real;	卡方分布累积函数	ChisquareCdf(0.3, 2) = 0.13929
29	ChiSquarePDF(X: Real; n: Integer): Real;	卡方分布密度函数	ChisquarePdf(0.4, 3) = 0.20657
30	Cos(X: Real): Real;	余弦函数	Cos(2.3) = -0.6662760213
31	Cosd(X: Real): Real;		Cosd(3.5) = 0.9981347984
32	Cosh(X: Real): Real;	余弦双曲函数	Cosh(0.6) = 1.185465218
33	Cot(X: Real): Real;	余切函数	Cot(5.4) = -0.8213276958
34	Coth(X: Real): Real;	双曲余切函数	Coth(1.2) = 1.199537544
35	Erf(X: Real): Real;	误差函数	Erf(0.6) = 0.6038561848
36	Erfc(X: Real): Real;	互补误差函数	Erfc(3.2) = 6.031483983e-6
37	Exp(X: Real): Real;	指数函数	Exp(1.8) = 6.049647464
38	ExponentialCDF(X: Real; A: Real): Real;	指数密度累积分布函数	ExponentialCDF(0.5, 0.9) = 0.4262465793
39	ExponentialCDFInv(P[0,1]: Real; A: Real): Real;	指数密度累积分布反函数	ExponentialCDFInv(0.3, 0.4) = 0.1426699776
40	ExponentialPDF(X: Real; A: Real): Real;	指数密度分布函数	ExponentialPDF(0.12, 0.54) =

			1.482847042
41	FCDF(X: Real; n1: Integer; n2: Integer): Real;	F 密度累积分布函数	FCDF(0.6, 3, 1) = 0.2871897411
42	FPDF(X: Real; n1: Integer; n2: Integer): Real;	F 密度分布函数	FPDF(0.3, 3, 2) = 0.5961681487
43	Gamma(X: Real): Real;	伽玛函数	Gamma(0.6) = 1.489192249
44	LnGamma(X: Real): Real	对数伽玛函数	LnGamma(0.6) = 0.398233858069235
45	Igamma(X: Real; a: Real): Real	不完全伽玛函数	IGamma(2,0.6) = 0.121901382249558
46	GammaCDF(X: Real; A[>0]: Real; B[>0]: Real): Real;	伽玛密度累积分布函数	GammaCdf(0.3, 0.1, 0.2) = 0.988655
47	GammaCDFInv(P[0,1]: Real; A[>0]: Real; B[>0]: Real): Real;	伽玛密度累积分布反函数	GammaCDFInv(0.1,2,3) = 1.595434825
48	GammaPDF(X: Real; A[>0]: Real; B[>0]: Real): Real;	伽玛密度分布函数	GammaPDF(0.3, 0.4, 0.2) = 0.3943490019
49	Ln(X: Real): Real;	自然对数函数	Ln(3.2) = 1.16315081
50	Log(X: Real): Real;	10 为底的对数	Log(3.2) = 0.5051499783
51	Logn(Base: Real; X: Real): Real;	Base 为底的对数	Logn(10, 3.2) = 0.5051499783
52	Max(X1: Real; X2: Real): Real;	两数最大	Max(2.3,3.2) = 3.2
53	Min(X1: Real; X2: Real): Real;	两数最小	Min(2.3,3.2) = 2.3
54	NormalCDF(X: Real): Real;	正态分布累积函数	NormalCDF(0.3) = 0.6179114222
55	NormalPDF(X: Real): Real;	正态分布函数	NormalPDF(0.9) = 0.2660852499
56	PoissonCDF(n: integer; X: Real): Real;	泊松密度分布累积函数	PoissonCDF(2,0.5) = 0.985612322
57	PoissonPDF(n: Integer; X: Real): Real;	泊松密度分布函数	PoissonPDF(2,0.5) = 0.07581633246
58	Power(Base: Real; Exponent: Real): Real;	Base 为底的指数函数	Power(2.2, 3.1) = 11.52153413
59	Psi(X: Real): Real	普西函数	Psi(2.5) = 0.703156640645243
60	Real2Bin(X: Real; n: Integer): String;		Real2Bin(0.5, 2) = 0.101
61	Round(X: Real): Real;	四舍五入函数	Round(0.364) = 0
62	Sign(X: Real): Real;	符号函数	Sign(2.3) = 1
63	Sin(X: Real): Real;	正弦函数	Sin(3.2) = -0.05837414343
64	Sind(X: Real): Real;		Sind(0.6) = 0.01047178412
65	Sinh(X: Real): Real;	正弦双曲函数	Sinh(5.6) = 135.2113548
66	Sqr(X: Real): Real;	平方函数	Sqr(4.2) = 17.64
67	Sqrt(X: Real): Real;	平方根函数	Sqrt(3.5) = 1.870828693
68	StudentCDF(X: Real; n: Integer): Real;	学生密度分布累积函数	StudentCDF(0.5, 2) = 0.6666666667
69	StudentPDF(X: Real; n: Integer): Real;	学生密度分布函数	StudentPDF(0.5, 2) = 0.2962962963
70	Tan(X: Real): Real;	正切函数	Tan(3.2) = 0.05847385446
71	Tand(X: Real): Real;		Tand(6.5) = 0.1139356083
72	Tanh(X: Real): Real;	正切双曲函数	Tanh(0.9) = 0.7162978702
73	Trunc(X: Real): Real;	取整函数	Trunc(3.2) = 3
74	Wrap(X: Integer, n: Integer): Integer		Wrap(6,5)=1, Wrap(7,5)=2
75	Wrap0(X: Integer, n: Integer): Integer		Wrap(6,5)=0, Wrap0(7,5)=1

#### 复数函数

abs(), ln(), log(), exp(), power(), sqr(), sqrt(), sin(), cos(), tan(), sec(), csc(), cot(), cosh(), sinh(), tanh(), sech(), csch(), coth(), bessj0(), bessj0(), gamma(), arcsin(), arccos(), arctan(), arccot(), arcsec(), atan()

# 1.6 基本语法

## ✧ 书写方式

代码本书写方式自由，如果一行代码过长，可用回车键转到下一行，不用任何连接符，也不影响计算效果。

## ✧ 行代码结束符

每一句代码一般以 ‘;’ 号作结束符。如

Parameter a, b, c, d;

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

## ✧ 多代码块

每一代码本可写多个代码块描述多个问题，用关键字 “NewDivision” 隔开。

## ✧ 变量及参数自动识别

对曲线拟合，对二维，缺省自变量名为 x，因变量名为 y；对三维或多维，缺省自变量名为 x1, x2, x3..., 因变量名为 y。如下两段代码效果等同，右边代码中无需再定义变量和参数，将由 1stOpt 自动识别。

代码 1	代码 2
Variable 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

对函数优化，如参数没有范围限制，也可省去参数定义，下列左右两段代码效果等同。

代码 1	代码 2
Parameter 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;

## ✧ 关键字和数学函数

为避免出错，代码本关键字和数学函数应尽量避免手工输入，而应选择快捷窗口（“Ctrl + K”、“Ctrl + M”）或从导航窗口输入，正确的关键字或数学函数会自动加粗加亮显示。

## 第二章 1stOpt 功能

### 2.1 函数优化

函数优化是最常见、最基本也是最直观的优化问题。1stOpt 可求任意形式、任意维数、约束或非约束的函数优化问题，约束函数即可是不等式也可是等式。

主要使用关键字：

- Paramter: 定义参数及其范围;
- MinFunction: 定义目标函数并求其最小值;
- MaxFunction: 定义目标函数并求其最大值;
- MinMax: 极大极小函数求解

#### 2.1.1 一维函数优化

求下列一维函数最小值：

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

其中， $x \in [-3\pi, 3\pi]$

如图 2-1 示，在给定区间有两个局部最优和一个全局最优，1stOpt 可以很容易求得全局最优解。

1stOpt 代码

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

或更为简单形式

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

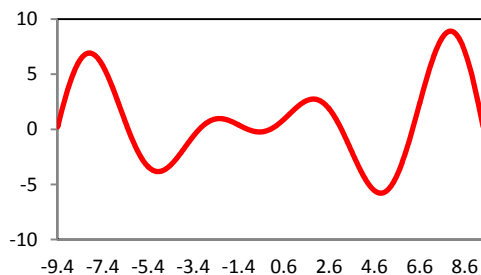


图 2-1 一维函数图

结果：f = -5.7976，x = 4.8808

#### 2.1.2 多维函数优化

✧ 二维针状全局最优的函数

该函数的函数图形如下所示，在(50, 50)处取得全局最大值 1.1512，其第二极大值为 1.12837，它是一个多峰值函数，采用传统优化方法几乎不能找到全局最优点。1stOpt 可轻易求的最优值

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

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

1stOpt 代码

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

✧ 多维函数最小值:

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

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

1stOpt 代码

```
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));
```

结果:  $f = -51.7695$

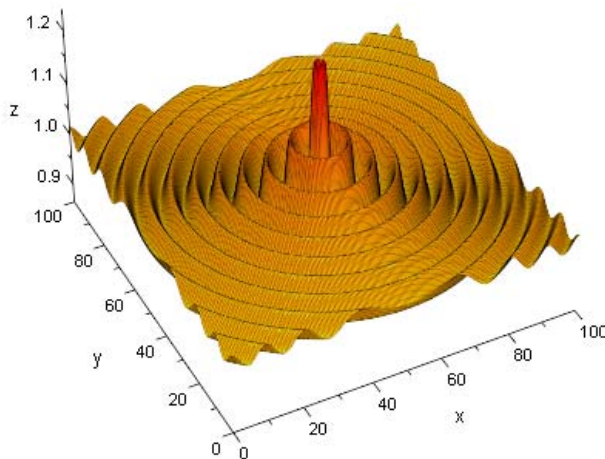


图 2-2 针状函数三维图

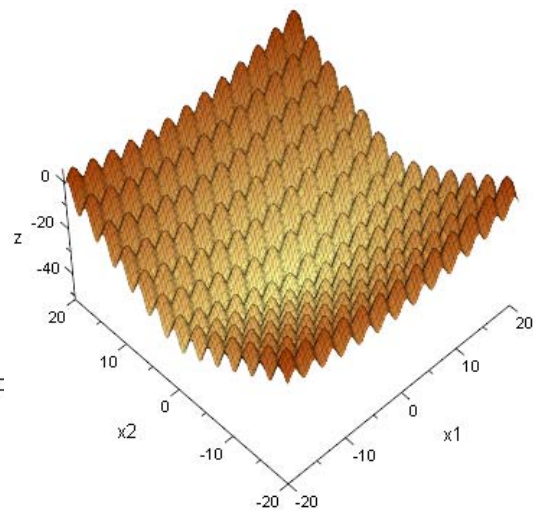


图 2-3 多维函数优化三维图

## 2.1.3 隐函数优化

相对于普通函数, 隐函数在表达式中包含有自身, 且无法转换成普通函数显示, 一般有如下形式:

$$\min y = f(x_i, y) \quad (2-4)$$

1stOpt 隐函数优化与普通函数优化并无太大区别。

例 1, 求下列隐函数  $z$  的最小值:

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

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

1stOpt 代码

```
Parameter 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));
```

结果:  $z = 0.02335$  ( $x = 2.898329$ ,  $y = -0.8573138$ )

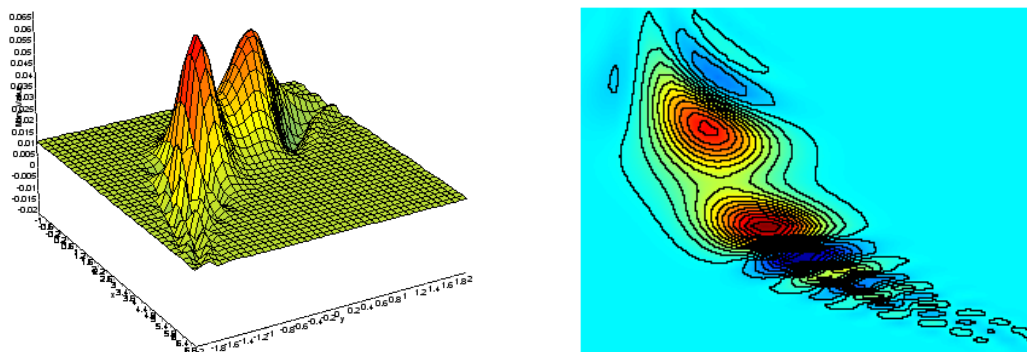


图 2-4 隐函数优化三维图

例 2, 求下列隐函数  $y$  的最小值:

$$\min_y \cos(x_1) \cdot \cos(x_2) - 2 \cdot \exp(-50 \cdot ((x_1 - 1)^2 + (x_2 - 1)^2) + y) + 2.5 \cdot \exp(-51 \cdot ((x_1 + 0.5)^2 + (x_2 + 0.5)^2) + y) \quad (2-6)$$

其中,  $x_1, x_2 \in [0, 2\pi]$

1stOpt 代码

```
Parameter x(2)=[0,2*pi];
Minimum=y;
Function
y=cos(x1)*cos(x2)-2*exp(-50*((x1-1)^2+(x2-1)^2)+
2*y)-2.5*exp(-51*((x1+0.5)^2+(x2+0.5)^2)+2.5*y);
```

结果

```
目标函数值(最小): -1
x1: 3.52484268344853E-9
x2: 3.14159265653279
或
目标函数值(最小): -1
x1: 6.28318530022232
x2: 3.14159264453642
```

## 2.1.4 线性规划

1stOpt 算法中含有专门的线性算法—单纯性算法, 可高效求解线性规划问题。不同于 Lingo 等优化软件包, 在 1stOpt 中, 各待求参数的缺省设置范围是正负无穷。另外书写方式也自由, 如 “ $x_1 + 3x_2 + x_3 \leq 15$ ;” 完全等同于 “ $3x_2 + x_3 + x_1 - 15 \leq 0$ ;”。

线性规划实例之一

$$\max 2 \cdot x_1 + 3 \cdot x_2 + x_3 \quad (2-7)$$

1stOpt 代码

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

$$\text{s. t. } \begin{cases} x_1 + 3 \cdot x_2 + x_3 \leq 15 \\ 2 \cdot x_1 + 3 \cdot x_2 - x_3 \leq 18 \\ x_1 - x_2 + x_3 \leq 3 \\ x_1, x_2, x_3 \geq 0 \end{cases}$$

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

结果:  $x_1=5$ ,  $x_2=3$ ,  $x_3=1$ , 最大值为 20

#### ✧ 线性规划实例之二

min 0.44x1+0.94x2+0.88x3+0.48x4+4x5+3.4x6+2.3x7+0.12x8+1.6x9+19x10+25x11  
 3230x1+2640x2+2500x3+1730x4+2900x5+2230x6+2500x7>2750  
 8.27x1+43x2+40x3+15.4x4+62x5+50x6+45x7>15  
 8.27x1+43x2+40x3+15.4x4+62x5+50x6+45x7<16  
 0.038x1+0.32x2+0.32x3+0.14x4+3.91x5+4.6x6+33.4x8+21x9>2.85  
 0.038x1+0.32x2+0.32x3+0.14x4+3.91x5+4.6x6+33.4x8+21x9<3  
 0.058x1+0.15x2+0.14x3+0.32x4+2.9x5+2.15x6+0.14x8+18.5x9>0.5  
 0.058x1+0.15x2+0.14x3+0.32x4+2.9x5+2.15x6+0.14x8+18.5x9<0.55  
 0.26x1+2.45x2+2.41x3+0.54x4+4.35x5+3.28x6+2.6x7+99x11>0.8  
 0.125x1+0.48x2+0.51x3+0.18x4+1.65x5+1.31x6+0.65x7+99x10>0.31  
 0.298x1+1.08x2+1.4x3+0.58x4+2.21x5+1.74x6+0.83x7+99x10>0.58  
 0.298x1+1.08x2+1.4x3+0.58x4+2.21x5+1.74x6+0.83x7+99x10<0.63  
 0.077x1+0.6x2+0.6x3+0.27x4+0.8x5+0.64x6>0.19  
 x1>0.5, x1<0.66  
 x2+x3>0.1, x2+x3<0.22  
 x4>0.04, x4<0.2  
 x5+x6>0.03, x5+x6<0.07  
 0<x7<0.035  
 x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11=1

1stOpt 代码

```
Parameter x1[0.5,0.66], x4[0.04,0.2],x7[0,0.035];
MinFunction 0.44*x1+0.94*x2+0.88*x3+0.48*x4+4*x5+3.4*x6+2.3*x7+0.12*x8+1.6*x9+19*x10+25*x11;
3230*x1+2640*x2+2500*x3+1730*x4+2900*x5+2230*x6+2500*x7>2750;
8.27*x1+43*x2+40*x3+15.4*x4+62*x5+50*x6+45*x7>15;
8.27*x1+43*x2+40*x3+15.4*x4+62*x5+50*x6+45*x7<16;
0.038*x1+0.32*x2+0.32*x3+0.14*x4+3.91*x5+4.6*x6+33.4*x8+21*x9>2.85;
0.038*x1+0.32*x2+0.32*x3+0.14*x4+3.91*x5+4.6*x6+33.4*x8+21*x9<3;
0.058*x1+0.15*x2+0.14*x3+0.32*x4+2.9*x5+2.15*x6+0.14*x8+18.5*x9>0.5;
0.058*x1+0.15*x2+0.14*x3+0.32*x4+2.9*x5+2.15*x6+0.14*x8+18.5*x9<0.55;
0.26*x1+2.45*x2+2.41*x3+0.54*x4+4.35*x5+3.28*x6+2.6*x7+99*x11>0.8;
0.125*x1+0.48*x2+0.51*x3+0.18*x4+1.65*x5+1.31*x6+0.65*x7+99*x10>0.31;
0.298*x1+1.08*x2+1.4*x3+0.58*x4+2.21*x5+1.74*x6+0.83*x7+99*x10>0.58;
0.298*x1+1.08*x2+1.4*x3+0.58*x4+2.21*x5+1.74*x6+0.83*x7+99*x10<0.63;
0.077*x1+0.6*x2+0.6*x3+0.27*x4+0.8*x5+0.64*x6>0.19;
x2+x3>0.1;
x2+x3<0.22;
x5+x6>0.03;
x5+x6<0.07;
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11=1;
```

结果

迭代数: 13	x2: -0.028913011
计算用时(时:分:秒:毫秒): 00:00:00:15	x3: 0.212176502

算法：单纯形线性规划法 该线性规划的最小(Min)为：0.651708124 参数最优解为： x1: 0.66 x4: 0.04 x7: 0	x5: 0.041774768 x6: -0.011774768 x8: 0.068574559 x9: 0.017195715 x10: 0.000731968 x11: 0.000234268
---	---

## 2.1.5 非线性规划问题

✧ 非线性规划问题

$$\max \frac{\sin(2 \cdot \pi \cdot x_1)^3 \cdot \sin(2 \cdot \pi \cdot x_2)}{x_1^3 \cdot (x_1 + x_2)} \quad (2-8)$$

$$\text{s. t.} \begin{cases} x_1^2 - x_2 + 1 \leq 0 \\ 1 - x_1 + (x_2 - 4)^2 \leq 0 \\ 0 \leq x_1, x_2 \leq 10 \end{cases}$$

lstOpt 代码

```
Parameter 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;
```

结果：x1= 1.22797, x2= 4.24537, 最大值为 0.095825

✧ 非线性混合整数规划问题

有两种方法定义整数参数，如 p1 为区间[-100, 100]的整数参数，可定义如下：

Parameter p1=[-100, 100, 0]; 或 IntParameter p1=[-100, 100];

$$\min 1.5 \cdot (x_1 - \sin(x_1 - x_2))^2 + 0.5 \cdot x_2^2 + x_3^2 - x_1 \cdot x_2 - 2 \cdot x_1 + x_2 \cdot x_3 \quad (2-9)$$

$$\text{s. t.} \begin{cases} -20 < x_1 < 20 \\ -20 < x_2 < 20 \\ -10 < x_3 < 10 \end{cases} \quad x_1, x_2 \text{ 为实数, } x_3 \text{ 为整数}$$

lstOpt 代码

```
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;
```

结果：当 x1= 4.49712, x2= 9.147501, x3= -4 时，得最小值为-10.51832

✧ 非线性整数规划问题

$$\min -2 \cdot x_1 - x_2 - 4 \cdot x_3 - 3 \cdot x_4 - x_5 \quad (2-10)$$

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

其中，x1 至 x5 均为整数。



1stOpt 代码

```
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;
```

结果有两组:  $x_1, x_2, x_3, x_4, x_5 = (15, 0, 6, 11, 2)$  或  $x_1, x_2, x_3, x_4, x_5 = (19, 0, 4, 10, 5)$ , 最小值为-89

## 2.1.6 排列组合优化

1stOpt 亦可用与解决组合优化问题。最大继承法 (MI0) 在解决该类问题时, 比其它诸如遗传算法, 模拟退火及禁忌算法等表现更优。

✧ TSP 问题

TSP 问题是非常著名的组合优化问题: 有N个城市, 从某一城市出发, 每个城市访问一次, 最后回到起始城市, 试求最短距离的访问路线。下面以某地 15 个城市为例。



图 2-5 十五城市分布图

表 2-1 十五城市距离表

	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	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	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

### 1stOpt 代码

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]];

ObjectiveResult := TemSum + Distance[Cities[n-1], Cities[0]];

End;

EndProgram;

结果：最短距离为 2 0 7 6 K M, 访问路径为：

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

上面求解代码也可用快捷模式，代码中用到了“Wrap0”函数。

### 1stOpt 快捷模式代码

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;

MinFunction Sum(i=0:n-1)(Distance[Cities[i], Cities[Wrap0(i+1,n-1)]]);

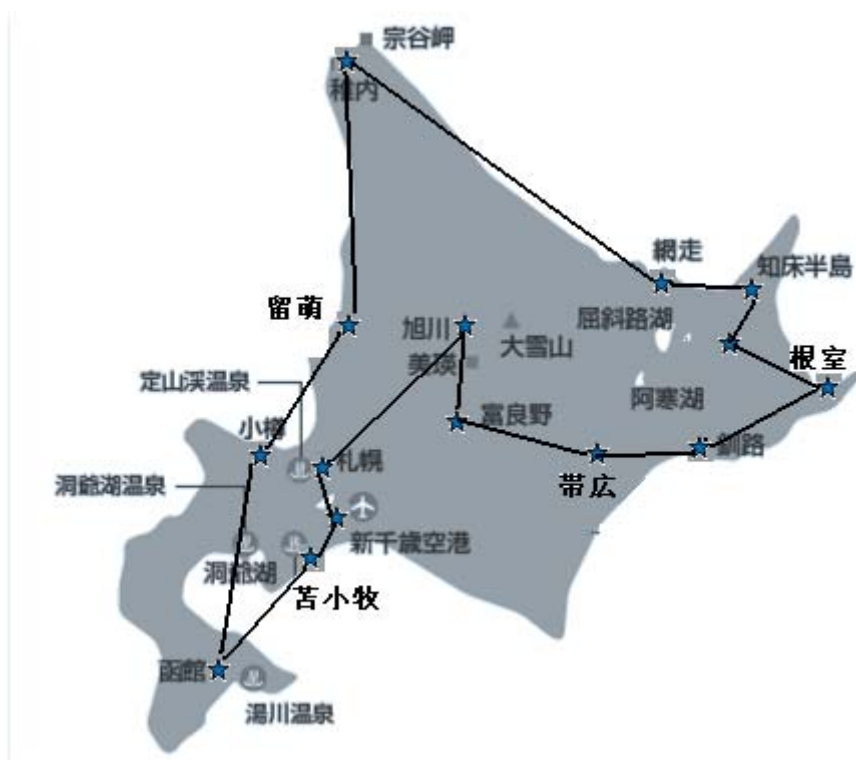


图 2-6 十五城市最短路径地图

# ◇ 最短路径问题

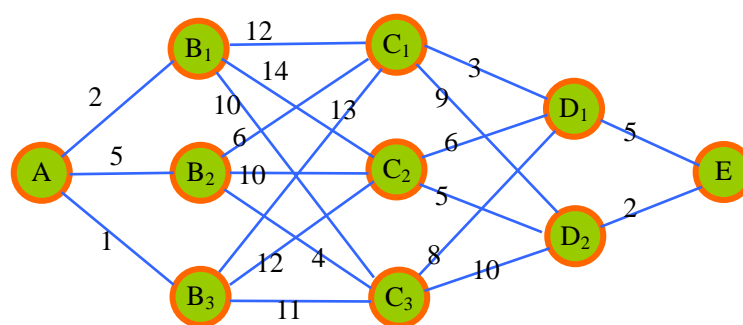


图 2-7 最短路径示意图

图 2-7 表示从起点 A 到终点 E 之间各点的距离。求 A 到 E 的最短路径。

此问题也乃经典的组合优化问题。不同于传统的用动态线性规划方法求解，1stOpt 使用全局优化来寻优。

表 2-2 城市距离表(无联结城市间的距离用虚拟值 1000 表示)：

	序号	0	1	2	3	4	5	6	7	8
序号		A	B1	B2	B3	C1	C2	C3	D1	D2
0	A	0	2	5	1	1000	1000	1000	1000	1000
1	B1	2	0	1000	1000	12	14	10	1000	1000
2	B2	5	1000	0	1000	6	10	4	1000	1000
3	B3	1	1000	1000	0	13	12	11	1000	1000
4	C1	1000	12	6	13	0	1000	1000	3	9
5	C2	1000	14	10	12	1000	0	1000	6	5
6	C3	1000	10	4	11	1000	1000	0	8	10
7	D1	1000	1000	1000	1000	3	6	8	0	1000
8	D2	1000	1000	1000	1000	9	5	10	1000	0

1stOpt 代码

```

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;

```

结果：最短路径值为 19，路线图为：0 -> 2 -> 4 -> 7 -> 9 也既：A -> B2 -> C1 -> D1 -> E。上面代码也可用下面快捷模式，注意使用了关键字“MData”来描述距离矩阵数据，计算结果相同。

1stOpt 快捷模式代码

```

Constant n = 8;
Parameters Cities(1:n)[1,n,0];
MDataSet[1000];
i,j,Dis=
    0 1    2
    0 2    5
    0 3    1
    1 4   12

```

```

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

```

EndMDataSet;

MinFunction Dis[0,Cities[1]]+Sum(i=1:n-1)(Dis[Cities[i],Cities[i+1]])+ Dis[Cities[n],n+1];

## 2.1.7 多目标函数优化

多目标规划的基本思想大都是将多目标问题转化为单目标规划，基本方法有理想点法、线性加权法、最大最小法等。1stOpt 中没有专门的多目标优化命令，只能通过前述方法进行转换后求解。

$$\text{例 1. } \min \begin{cases} f_1(x) = 0.5 \cdot x_1 + 0.6 \cdot x_2 + 0.7 \cdot \exp\left(\frac{x_1 + x_3}{10}\right) \\ f_2(x) = (x_1 - 2 \cdot x_2)^2 + (2 \cdot x_2 - 3x_3)^2 + (5 \cdot x_3 - x_1)^2 \end{cases} \quad (2-11)$$

s. t.  $x_1 \in [10, 80], x_2 \in [20, 90], x_3 \in [15, 100]$

### ✧ 理想点法

1stOpt 代码

```

Parameter 10<=x1<=80,20<=x2<=90,15<=x3<=100;
ConstStr f1=0.5*x1+0.6*x2+0.7*exp((x1+x3)/10), f2=(x1-2*x2)^2+(2*x2-3*x3)^2+(5*x3-x1)^2;
MinFunction f1;

```

1) 求出单目标函数 f1 的最优值，f1=25.52774，x=[10, 20, 15];

2) 求出单目标函数 f2 的最优值，f2=300，x=[65, 27.5, 15];

3) 构筑新的单目标函数： $\min \sqrt{(f_1 - 25.52774)^2 + (f_2 - 300)^2}$

1stOpt 代码

```

Parameter 10<=x1<=80,20<=x2<=90,15<=x3<=100;
ConstStr f1=0.5*x1+0.6*x2+0.7*exp((x1+x3)/10), f2=(x1-2*x2)^2+(2*x2-3*x3)^2+(5*x3-x1)^2;
MinFunction Sqrt((f1-25.52774)^2+(f2-300)^2);

```

运算结果：

函数表达式：

$\text{sqrt}((((0.5 \cdot x_1 + 0.6 \cdot x_2 + 0.7 \cdot \exp((x_1 + x_3)/10)) - 25.52774)^2 + (((x_1 - 2 \cdot x_2)^2 + (2 \cdot x_2 - 3 \cdot x_3)^2 + (5 \cdot x_3 - x_1)^2) - 300)^2)$

目标函数值(最小): 578.27926596784

x1: 48.9910529644766

```

x2: 23.4556259778595
x3: 15
传递参数(PassParameter):
((0.5*x1+0.6*x2+0.7*exp((x1+x3)/10))): 459.483666124556
(((x1-2*x2)^2+(2*x2-3*x3)^2+(5*x3-x1)^2)): 684.443782170933

```

即理想点目标函数为： 578. 279

此时, f1=459. 48366, f2=684. 4437, x=[48. 9910529644766, 23. 4556259778595, 15]

## ✧ 线性加权法

每个目标函数赋予一权重系数，各权重系数之和等于 1。

$$\min a \cdot f_1(x) + (1 - a) \cdot f_2(x)$$

$$\text{s. t. } x_1 \in [10, 80], x_2 \in [20, 90], x_3 \in [15, 100]$$

$\alpha$  为目标函数 f1 的权重，范围在 0 至 1 之间，显然目标函数 f2 的权重为  $(1-\alpha)$ ， $\alpha$  取值不同，结果也不尽相同。下面代码从 0 到 1 取  $\alpha$  得值，注意使用了关键字 “LoopConstant”。

1st0pt 代码如下，结果如图 2-8 示。

```

LoopConstant a=[0:0.05:1];
Parameter 10<=x1<=80,20<=x2<=90,15<=x3<=100;
ConstStr f1=0.5*x1+0.6*x2+0.7*exp((x1+x3)/10), f2=(x1-2*x2)^2+(2*x2-3*x3)^2+(5*x3-x1)^2;
PlotLoopData a[x], f1, f2[y2];
MinFunction a*f1+(1-a)*f2;

```

表 2-3 计算结果

$\alpha$	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
f1	2135.67	1336.52	988.43	771.49	614.77	491.27	387.74	296.28	210.99	126.91	25.53
f2	300.00	334.62	394.23	465.75	549.77	650.74	777.61	948.64	1208.72	1705.84	5150.00

## ✧ 非劣解集法

- 1) 求出单目标函数 f1 的最优值，f1=25. 52774，x=[10, 20, 15]；
- 2) 将 f1 设为等式约束，f1 的变化从其最小值（取 26）开始递增
- 3) 求对应于 f1 的 f2 最小值
- 4) 得到 f1 与 f2 的非劣解集

1st0pt 代码如下，结果如图 2-9 示。

```

LoopConstant a=[26:50:1000];
Parameter 10<=x1<=80,20<=x2<=90,15<=x3<=100;
ConstStr f1=0.5*x1+0.6*x2+0.7*exp((x1+x3)/10), f2=(x1-2*x2)^2+(2*x2-3*x3)^2+(5*x3-x1)^2;
PlotLoopData f1[x], f2;
MinFunction f2;
f1=a;

```

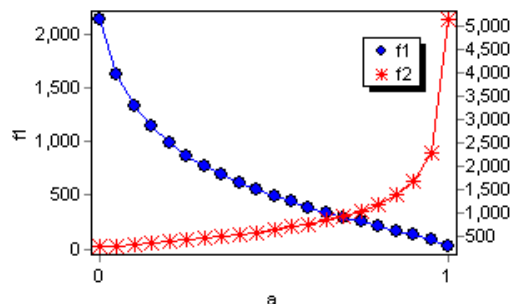


图 2-8 多目标可行解图

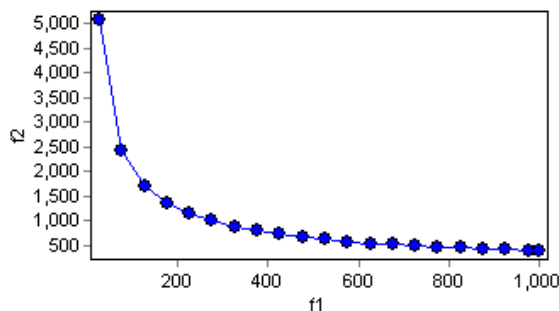


图 2-9 多目标非劣解集

表 2-4 多目标非劣解集结果

f1	f2	x1	x2	x3
26	5084.632	10.345	20.000	15
76	2411.745	27.715	20.000	15
126	1714.123	34.298	20.000	15
176	1369.823	38.298	20.625	15
226	1151.854	41.171	21.408	15
276	998.730	43.418	22.007	15
326	884.472	45.261	22.490	15
376	795.668	46.822	22.897	15
426	724.595	48.176	23.246	15
476	666.445	49.370	23.553	15
526	618.051	50.439	23.826	15
576	577.228	51.405	24.073	15
626	542.412	52.288	24.298	15
676	512.453	53.099	24.504	15
726	486.483	53.850	24.694	15
776	463.832	54.549	24.871	15
826	443.977	55.203	25.037	15
876	426.498	55.817	25.192	15
926	411.058	56.395	25.338	15
976	397.381	56.943	25.476	15
1000	391.374	57.195	25.540	15

## 2.1.8 极大极小函数优化

极大极小优化问题也是一种特殊的多目标优化问题，定义如下

$$\min f(x) = \max\{f_i(x), i=1, 2, \dots, m\} \quad (2-12)$$

1stOpt 中对应的函数为“MinMax”。

$$f_1 = 10 + 2 \cdot x_1 + 2 \cdot x_2 - x_1^2 + x_2^2 - x_1 \cdot x_2$$

$$\text{例 1. } f_2 = 2 + x_1 - x_2 + 2 \cdot x_1^2 - x_2^2 + 3 \cdot x_1 \cdot x_2 \quad (2-13)$$

$$f_3 = x_1^2 + x_2^2$$

1stOpt 代码

```
ConstStr f1=10+2*x1+2*x2-x1^2+x2^2-x1*x2,
f2=2+x1-x2+2*x1^2-x2^2+3*x1*x2,
f3=x1^2+x2^2;
MinMax f(3);
```

结果

优化算法: 标准简面体爬山法 + 通用全局优化法 (SM1)

极大极小值(MinMax): 4.25

x1: -2

x2: 0.5

极大极小函数:

1: ((10+2\*x1+2\*x2-x1^2+x2^2-x1\*x2)) = 4.25

2: ((2+x1-x2+2\*x1^2-x2^2+3\*x1\*x2)) = 4.25

3: ((x1^2+x2^2)) = 4.25

$$f_1 = x_1^4 + x_2^2$$

$$\text{例 2. } f_2 = (2 - x_1)^2 + (2 - x_2)^2 \quad (2-14)$$

$$f_3 = 2 \cdot \exp(x_2 - x_1)$$

1stOpt 代码

结果

```
ConstStr f1=x1^4+x2^2,
          f2=(2-x1)^2+(2-x2)^2,
          f3=2*exp(x2-x1);
MinMax f1,f2,f3;
```

优化算法: 标准简面体爬山法 + 通用全局优化法(SM1)  
 极大极小值(MinMax): 2  
 x1: 1  
 x2: 1  
 极大极小函数:  
 1: ((x1^4+x2^2)) = 2  
 2: (((2-x1)^2+(2-x2)^2)) = 2  
 3: ((2\*exp(x2-x1))) = 2

例 3. 约束极大极小问题

$$\begin{aligned} f_1 &= -5 \cdot x_1 + x_2 \\ f_2 &= 4 \cdot x_2 + x_1^2 + x_2^2 \\ f_3 &= 5 \cdot x_1 + x_2 \\ g_1(x) &= -2x_1 - x_2 \leq 0 \end{aligned} \quad (2-15)$$

1stOpt 代码

```
ConstStr f1=-5*x1+x2, f2=4*x2+x1^2+x2^2, f3=5*x1+x2;
MinMax f(3);
-2*x1-x2<=0;
```

结果

优化算法: 通用全局优化法(UGO1) 极大极小值(MinMax): 0 x1: 0 x2: 0	极大极小函数: 1: ((-5*x1+x2)) = 0 2: ((4*x2+x1^2+x2^2)) = 0 3: ((5*x1+x2)) = 0 约束函数: 1: -2*x1-x2-(0) = 0
---	---

## 2.1.9 与 Lingo 的比较

Lingo 是世界上著名的优化运筹学软件, 据称世界财富排行榜前 500 名得企业中有约三分之二在使用 Lingo, 其认可度可见一斑。与 Lingo 相比, 1stOpt 在知名度方面相差甚远, 使用方便度、全局优化能力等方面丝毫不差, 甚至更胜一筹。下面仅以几个函数优化例子进行比较, 用 Lingo 进行计算时均启用求解器选项 “Use Global Solver”。

例 1. 二维函数优化, 两个参数 x1、x2, X [-50, 50]。

$$\min \cos(x_1) \cdot \cos(x_2) - \sum_{i=1}^5 \left( (-1)^i \cdot i \cdot 2 \cdot \exp \left( -500 \cdot \left( (x_1 - i \cdot 2)^2 + (x_2 - i \cdot 2)^2 \right) \right) \right) \quad (2-16)$$

该测试题虽然只有两个参数, 但在给定的搜索域, 目标函数值的变化却是非常复杂, 如图 2-10 示; 图 2-11 是其局部放大图, 可看出最小极值点如同位于起伏不平的丘陵地带中的一口深井, 要发现位于 “井” 中的最深点, 是极其困难的。1stOpt 能以大于 90% 的概率获得最优解-7.97883233, 而 Lingo 获得的最好结果为-1, 实为一局部最优。

表 2-5 例题 1 Lingo 与 1stOpt 代码和结果

软件	代码	最优结果
1stOpt	Algorithm = SM2[100]; ParameterDomain = [-50,50]; MinFunction cos(x1)*cos(x2)-Sum(i=1:5)((-1)^i*i*2*exp(-500*((x1-i*2)^2+(x2-i*2)^2)));	目标函数值= -7.97883233 x1: 7.9999820 x2: 7.9999820
Lingo	Sets: P/1..5/;	目标函数值= -1 x1: 0



EndSets Min=@cos(x1)*@cos(x2)-@Sum(P(j): (-1)^j*j*2*@exp(-500*((x1-j*2)^2+(x2-j*2)^2))); @Bnd(-50,x1,50); @Bnd(-50,x2,50);	x2: -47.12389
---	---------------

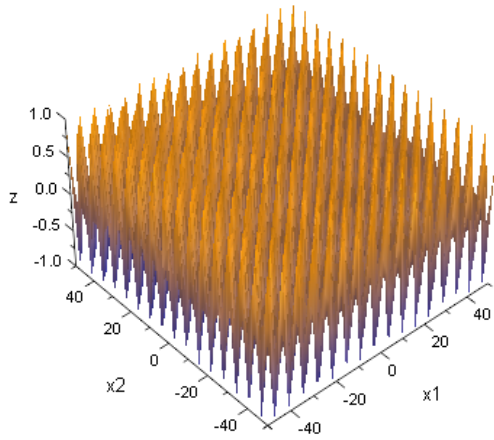


图 2-10. 函数三维图,  $X \in [-50, 50]$ , 密度:  $100 \times 100$

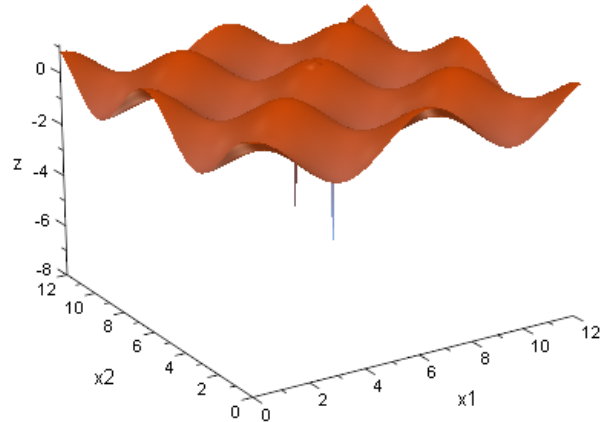


图 2-11. 函数三维图,  $X \in [0, 12]$ , 密度:  $100 \times 100$

例 2. 六参数多维函数优化,  $n=6$ ,  $p=10$ ,  $X \in [-100, 100]$

$$\min \sum_{j=1}^{n-1} \sum_{i=1}^p \left( - (1 + 0.1 \cdot (i-1)) \cdot \cos(x_j) \cdot \cos(x_{j+1}) \cdot \exp \left( - \left( (x_j - i \cdot \pi - i)^{(2+2 \cdot (i-1))} + (x_{j+1} - i \cdot \pi)^{(2+2 \cdot (i-1))} \right) \right) \right) \quad (2-17)$$

例题 2 有六个待求参数, 如图 2-12 示, 搜索域的大部分区间几乎没有变化梯度, 给寻优搜索带来了很大困难; 而从局部放大图 2-13 所示, 又有不少局部陷阱。1stOpt 所得最优解为-5.047, 为多解, 而 Lingo 则为-2.74239, 是一局部最优。

表 2-6 例题 2 Lingo 与 1stOpt 代码和结果

软件	代码	最优结果
1stOpt	Algorithm = SM2[50]; Constant M = 100, n = 6, p = 10; Parameter x(1:n)[-M,M]; MinFunction Sum(j=1:n-1)(Sum(i=1:p)(-(1+(i-1)*0.1)* cos(x[j])*cos(x[j+1])*exp(-((x[j]-i*pi-i)^ (2+(i-1)*2)+(x[j+1]-i*pi)^ (2+(i-1)*2))))));	目标函数值: -5.047127847 x1: 32.291404, x2: 5.1327345 x3: 8.8495559, x4: 8.3339692 x5: 1.4987781, x6: 5.7079632
Lingo	Sets: P/1..5/ K/1..10/ Par/1..6/:x; EndSets pi = 3.1415926; Min= @Sum(P(j): @Sum(K(i):-(1+(i-1)*0.1)*@cos(x(j))* @cos(x(j+1))*exp(-((x(j)-i*pi-i)^(2+(i-1)*2)+(x(j+1)-i*pi)^(2+ (i-1)*2)))))); @For(Par: @Bnd(-100,x,100));	目标函数值: -1.201742 x1: 9.623649 x2: 3.258190 x3: -1.665323 x4: 3.774672 x5: 3.472771 x6: 3.141593

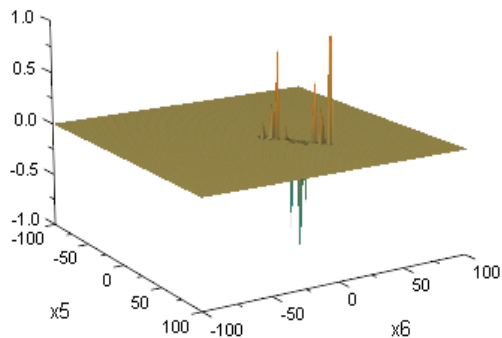


图 2-12. 函数三维图,  $X \in [-100, 100]$ ,  
密度:  $100 \times 100$

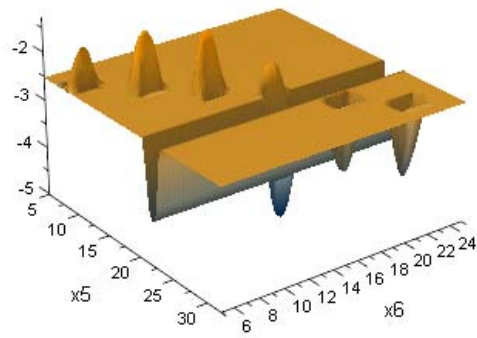


图 2-13. 函数三维图,  $X \in [0, 30]$ ,  
密度:  $100 \times 100$

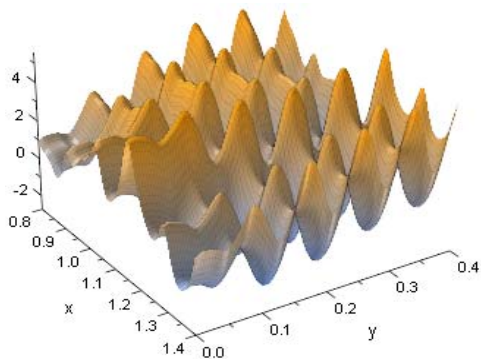


图 2-14. 函数三维图

例 3. 带等式约束的优化问题, 两个参数  $x_1$ 、 $x_2$ ,  $X \in [-50, 50]$

$$\begin{aligned} & \exp(\sin(50 \cdot x)) + \sin(60 \cdot \exp(y)) + \sin(70 \cdot \sin(x)) + \\ \min & \quad \sin(\sin(80 \cdot y)) - \sin(10 \cdot (x + y)) + \frac{(x^2 + y^2)^{\sin(y)}}{4} \quad (2-18) \\ \text{s. t. } & x - ((\cos(y))^x - x)^y = 0 \end{aligned}$$

例题 3 为一有不等式约束的优化问题, 该问题的特殊点是其函数三维表明是非连续的, 因而对大多数容错能力低的求解器来说都极其困难。其三维示意图如图 2-14 所示; 1stOpt 和 Lingo 所得最优解分别为-3.46659 和-1.034522。

表 2-7 例题 3 Lingo 与 1stOpt 代码和结果

软件	代码	最优结果
1stOpt	<pre> Algorithm = DE1[100]; PenaltyFactor = 5; ParameterDomain = [-50,50]; 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)^sin(y)/4; x-((cos(y))^x-x)^y&lt;=0; </pre>	目标函数值: -3.4665958617 x: -49.921967437 y: 4.8499336803
Lingo	<pre> Min=@exp(@sin(50*x)) +@sin(60*@exp(y)) + @sin(70*@sin(x)) +@sin(@sin(80*y))- @sin(10*(x+y))+(x^2+y^2)^@sin(y)/4; x-((@cos(y))^x-x)^y&lt;=0; @bnd(-50,x,50); @bnd(-50,y,50); </pre>	目标函数值: -1.034522 x: -6.679913 y: -0.5638658

例 4. 二维隐函数优化,  $x_1, x_2 \in [-100, 100]$

$$\begin{aligned} & y = -\cos(x_1) \cdot \cos(x_2) \cdot \exp\left(-\left((x_1 - \pi)^{2 \cdot y} + (x_2 - \pi)^2\right)\right) - \\ & 1.1 \cdot \cos(x_1) \cdot \cos(x_2) \cdot \exp\left(-\left((x_1 - 2 \cdot \pi)^4 + (x_2 - 2 \cdot \pi)^4\right) + 1.1 \cdot y\right) - \\ \min & 1.2 \cdot \cos(x_1) \cdot \cos(x_2) \cdot \exp\left(-\left((x_1 - 3 \cdot \pi)^6 + (x_2 - 3 \cdot \pi)^6\right) + 1.2 \cdot y\right) - \\ & 1.3 \cdot \cos(x_1) \cdot \cos(x_2) \cdot \exp\left(-\left((x_1 - 3.5 \cdot \pi)^6 + (x_2 - 3.5 \cdot \pi)^6\right) + 1.3 \cdot y\right) - \\ & 1.4 \cdot \cos(x_1 \cdot \pi) \cdot \cos(x_2 \cdot \pi) \cdot \exp\left(-\left((x_1 - 4 \cdot \pi)^8 + (x_2 - 4 \cdot \pi)^8\right) + 1.4 \cdot y\right) \end{aligned} \quad (2-19)$$

Lingo 求解隐函数的方法是将隐函数转换成一有等式约束的优化问题。1stOpt 与 Lingo 的代码如下, 1stOpt 很容易得到最优值-0.9998873, 而 Lingo 的最好值仅为-0.5807459。

表 2-8 例题 4 Lingo 与 1stOpt 代码和结果

软件	代码	最优结果
1stOpt	<pre>parameter x(2)=[-100,100]; Minimum = y; Function y=-cos(x1)*cos(x2)*exp(-((x1-pi)^(2*y)+(x2-pi)^2)) -1.1*cos(x1)*cos(x2)*exp(-((x1-2*pi)^4+(x2-2*pi)^4)+1.1*y) -1.2*cos(x1)*cos(x2)*exp(-((x1-3*pi)^6+(x2-3*pi)^6)+1.2*y) -1.3*cos(x1)*cos(x2)*exp(-((x1-3.5*pi)^6+(x2-3.5*pi)^6)+1.3*y) -1.4*cos(x1*pi)*cos(x2*pi)*exp(-((x1-4*pi)^8+(x2-4*pi)^8)+1.4*y);</pre>	目标函数值: -0.9998873 x1: 97.38937465 x2: 3.141592655
Lingo	<pre>DATA:   pi=3.1415926; ENDDATA Min=y; y=-@cos(x1)*@cos(x2)*@exp(-((x1-pi)^(2*y)+(x2-pi)^2)) -1.1*@cos(x1)*@cos(x2)*@exp(-((x1-2*pi)^4+(x2-2*pi)^4)+1.1*y) -1.2*@cos(x1)*@cos(x2)*@exp(-((x1-3*pi)^6+(x2-3*pi)^6)+1.2*y) -1.3*@cos(x1)*@cos(x2)*@exp(-((x1-3.5*pi)^6+(x2-3.5*pi)^6)+1.3*y) -1.4*@cos(x1*pi)*@cos(x2*pi)*@exp(-((x1-4*pi)^8+(x2-4*pi)^8)+1.4*y); @bnd(-100,x1,100); @bnd(-100,x2,100); @free(y);</pre>	目标函数值: -0.5807459 x1: 6.283192 x2: 6.282755

## 2.1.10 与 Matlab 的比较

Matlab 已被公认为当今科学数值计算软件平台的杰出代表, 它即可以作为一个应用平台, 也可以作为一个开发平台。它的优化工具箱提供了线性规划, 二次规划, 非线性优化、非线性最小二乘及非线性方程求解的工具或命令。虽然 Matlab 整体上是优秀的数学软件, 但在局部领域如全局非线性最优化方面, 其提供的现成工具与 1stOpt 等专用的优化软件相比, 仍有一些缺陷, 主要有:

- 软件本身庞大, 对硬件系统要求高;
- 解释性语言, 对大计算量优化问题, 计算速度偏慢
- 采用的优化算法不够先进, 对稍微复杂点的问题难以获得最优解
- 编写代码相对比较繁琐, 需专门的培训和学习

下面以一些简单的例子对 Matlab 处理优化问题和 1stOpt 进行对比。

例 1. 线性规划问题

$$\min f = 3 \cdot x_1 - 6 \cdot x_2 \quad (2-20)$$

$$\text{s. t. } \begin{cases} x_1 \leq 4 \\ x_2 \leq 3 \\ x_1 + 2x_2 \leq 10 \\ x_1 \geq 0 \\ x_2 \geq 0 \end{cases}$$

Matlab 线性规划调用命令为“linprog”，编写程序如下：

Matlab 代码

```
f=[3 -6];
A=[1 0;0 1;1 2;-1 0;0 -1];
b=[4;3;10;0;0];
[x,fval]=linprog(f,A,b)
f=fval*(-1)
```

lstOpt 代码

```
Algorithm = LP;
Parameter 0<=x1<=4, 0<=x2<=3;
MinFunction 3*x1-6*x2;
x1+2*x2<=10;
```

运行结果

```
x =
    0.0000
    3.0000
fval =
-18.0000
```

运行结果

该线性规划的最小(Min)为: -18

参数最优解为:

x1: 0

x2: 3

约束函数:

1: x1+2\*x2-10 = -4

2: x1-0 = 0

3: x1-4 = -4

4: x2-0 = 3

5: x2-3 = 0

例 2. 数学模型如下:

$$\max f = 0.13 \cdot x_1 + 0.09 \cdot x_2 + 0.06 \cdot x_3 + 0.14 \cdot x_4 \quad (2-21)$$

$$\text{s. t } \begin{cases} x_1 - x_2 - x_3 - x_4 \leq 0 \\ x_2 + x_3 - x_4 \geq 0 \\ x_1 + x_2 + x_3 + x_4 = 1 \\ x_j \geq 0, j = 1 \dots 4 \end{cases}$$

Matlab 只能求解最小值，因此求解最大值时需将目标函数转换成求最小值的标准形式：

$$\min f = -0.13 \cdot x_1 - 0.09 \cdot x_2 - 0.06 \cdot x_3 - 0.14 \cdot x_4$$

同时约束函数也必须转换成小于等于 0 的形式，即

$$\text{s. t } \begin{cases} x_1 - x_2 - x_3 - x_4 \leq 0 \\ -x_2 - x_3 + x_4 \leq 0 \\ x_1 + x_2 + x_3 + x_4 = 1 \\ x_j \geq 0, j = 1 \dots 4 \end{cases}$$

Matlab 代码

```
f = [-0.13;-0.09;-0.06;-0.14];
A = [1 -1 -1 -1
      0 -1 -1 1];
b = [0; 0];
Aeq=[1 1 1 1];
```

运行结果

```
x =
    0.5000
    0.2500
    0.0000
    0.2500
```

```

beq=[1];
lb = zeros(4,1);
[x,fval,exitflag] = linprog(f,A,b,Aeq,beq,lb)
f=-fval

```

```

1stOpt 代码

Algorithm = LP;
ParameterDomain = [0,];
MaxFunction
0.13*x1+0.09*x2+0.06*x3+0.14*x4;
    x1-x2-x3-x4<=0;
    x2+x3-x4>=0;
    x1+x2+x3+x4=1;

```

```

fval =
    -0.1225
exitflag =
     1
f =
    0.1225

```

```

运行结果

该线性规划的最大(Max)为: 0.1225
参数最优解为:
    x1: 0.5
    x2: 0.25
    x3: 0
    x4: 0.25
约束函数:
    1: x1-x2-x3-x4-0 = -5.551115123E-017
    2: x2+x3-x4-0 = 5.551115123E-017
    3: x1+x2+x3+x4-1 = 0
    4: x1-0 = 0.5
    5: x2-0 = 0.25
    6: x3-0 = 0
    7: x4-0 = 0.25

```

例 3. 有约束非线性规划问题

$$\min \quad 100 \cdot (x_2 - x_1)^2 + (1 - x_1)^2 \tag{2-22}$$

$$\text{s. t} \quad \begin{cases} x_1 \leq 2 \\ x_2 \leq 2 \end{cases}$$

```

Matlab: 首先建立 fun1.m 文件:
function f=fun1(x)
f=100*(x(2)-x(2)^2)+(1-x(1))^2;
然后在工作空间键入程序:
x0=[1.1, 1.1];
A=[1 0;0 1];
b=[2;2];
[x, fval]=fmincon(@fun1, x0, A, b)
结果:    x =
           1.0000    1.0000
fval =
           3.1936e-011

```

```

1stOpt 代码

ParameterDomain = [2];
MinFunction 100*(x2-x1)^2+(1-x1)^2;

```

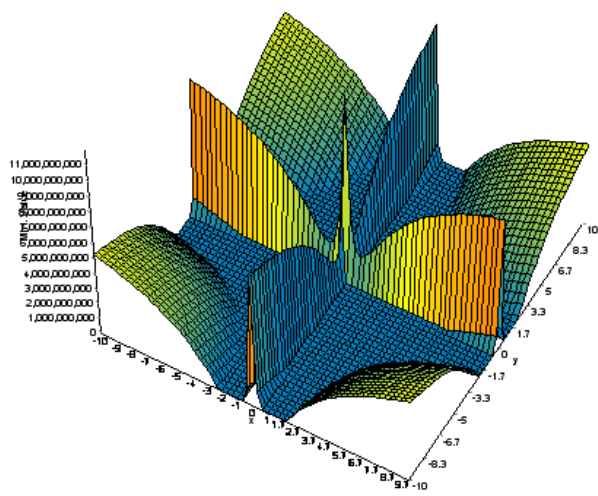


图 2-15. 非线性约束函数三维图

```

运行结果

函数表达式: 100*(x2-x1)^2+(1-x1)^2
目标函数值(最小): 0
x2: 1
x1: 1

```

相对于 Matlab 处理优化问题，1stOpt 代码简单、直观、易于理解，同时也不需猜初值。下列非线性约束优化问题，Matlab 获得最优解的难度较大，如图 2-15 示。

$$\min \quad -(x \cdot \sin(9 \cdot \pi \cdot y) + y \cdot \cos(25 \cdot \pi \cdot x) + 20) \quad (2-23)$$

$$\text{s. t} \quad x^2 + y^2 \leq 81$$

1stOpt 代码

```
Algorithm = MIO1;
ParameterDomain = [-10,10];
minfunction -(x*sin(9*PI*y)+y*cos(25*pi*x)+20);
x^2+y^2<=81;
```

运行结果

目标函数值(最小): -32.7178878068835  
x: -6.44002582226023  
y: -6.27797201413613  
约束函数:  
1: x^2+y^2-(81) = -0.1131347983

## 2.2 非线性拟合

非线性拟合是科研工作中最常用的方法之一。1stOpt 的非线性拟合功能强于目前任何已知软件包，如著名的 SPSS, SAS, Matlab, Origin, Systat, DataFit 等。其最大特点是，在绝大多数情况下，不需要使用者提供(猜测)任何初始值，仅依靠自身的全局搜索能力，从任意随机值出发，既可求得最优解。美国国家标准与技术研究院(NIST: National Institute of Standards and Technology) 提供有一套 27 道非线性拟合测试题，世界上几乎所有著名的数据分析软件包都以能通过该套测试题集为验证标准。经对比测试，1stOpt 是目前唯一不依赖使用 NIST 提供的初始值，而能以任意随机初始值就可求得全部最优解的软件包(如果使用 NIST 提供的初始值，则更可轻易求得最优解)。由于在实际应用当中，选择确定合理的初始值组是一件非常困难的事，尤其是在参数量比较多的情况下。从此意义而言，1stOpt 的实用能力达业界领先水平。

表 2-9 NIST 测试题结果

序号	测试题名	难度	待定参数量	初始值	1stOpt 使用算法	成功率(%)
1	Misrala	低	2	1stOpt 随机给出	通用优化算法	100
2	Chwirut2		3			100
3	Chwirut1		3			100
4	Lanczos3		6			100
5	Gauss1		8			100
6	Gauss2		8			100
7	DanWood		2			100
8	Misralb		2			100
9	Kirby2	中	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			100
16	Misralc		2			100
17	Misrald		2			100
18	Roszman1		4			100

19	ENS0		9			100
20	MGH09		4			100
21	Thurber		7			100
22	BoxBod		2			100
23	Rat42	高	3			100
24	MGH10		3			100
25	Eckerle4		3			100
26	Rat43		4			100
27	Bennett5		3			>90

注：NIST 网址：[http://www.itl.nist.gov/div898/strd/nls/nls\\_main.shtml](http://www.itl.nist.gov/div898/strd/nls/nls_main.shtml)

1stOpt 的曲线拟合均为自定义拟合，主要关键字有：

- Function：定义拟合公式；
- Parameter：定义参数；
- Variable：定义变量；
- Data、DataFile：定义数据；

“Function”、“Data”或“DataFile”为必须的关键字，其它为可选关键字。对二维曲线拟合，缺省自变量名为 x，因变量名为 y；对三维有两种缺省方式，一为自变量名为 x1 和 x2，因变量名为 y，第二种为自变量名为 x 和 y，因变量名为 z；对多维，缺省自变量名为 x1, x2, x3...，因变量名为 y。如下表中两段代码效果等同，右边代码中无需用“Variable”和“Parameter”定义变量和参数，而将由 1stOpt 自动识别。

拟合代码比较之一

代码 1	代码 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

当数据较长，为了节省代码本空间，可考虑将数据以行的形式给出，4.0 版及以前须用关键字“RowData”取代“Data”，同时数据形式亦做相应改动，每行数据以“;”号结束：  
拟合代码比较之二

代码 1	代码 2
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); RowData; 0.05,0.15,0.25,0.35; 0.13,0.13,0.19,0.34;

数据也可存为文件形式，用关键字“DataFile”调用，文件格式包括标准文本格式和 Excel 文件格式。如 Excel 数据如图 2-16 并存储为“c:\test1.xls”，调用形式如下。注意使用“DataFile”时，关键字“Variable”不能省略。

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

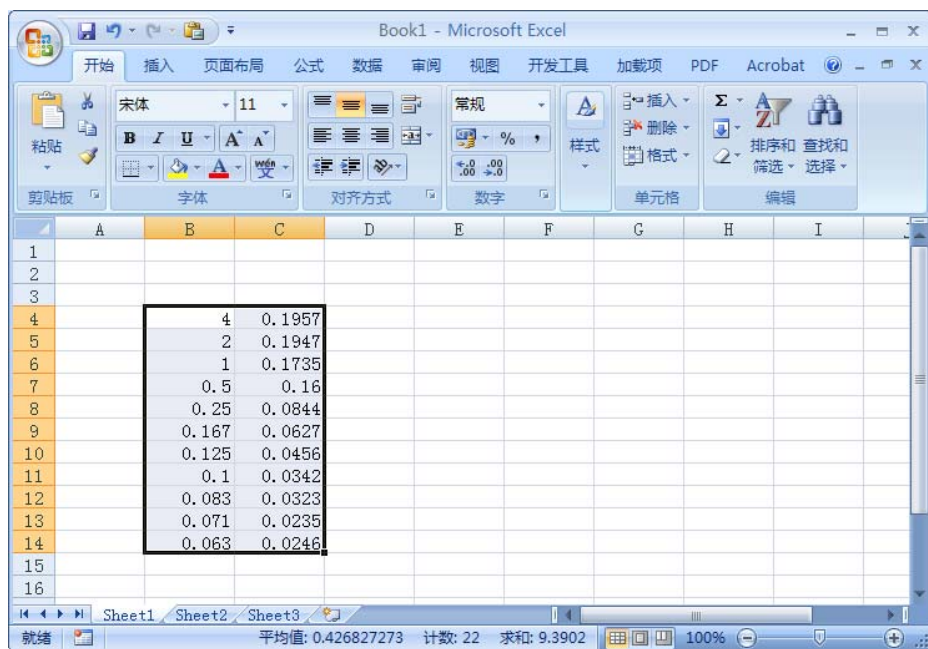


图 2-16 从 Excel 中读取数据

拟合计算完成后，在“二维-三维/预测”功能窗口还可进行多项其它工作：

- 预测：输入任意自变量值计算因变量
- 导数：计算拟合方程任一点的导数

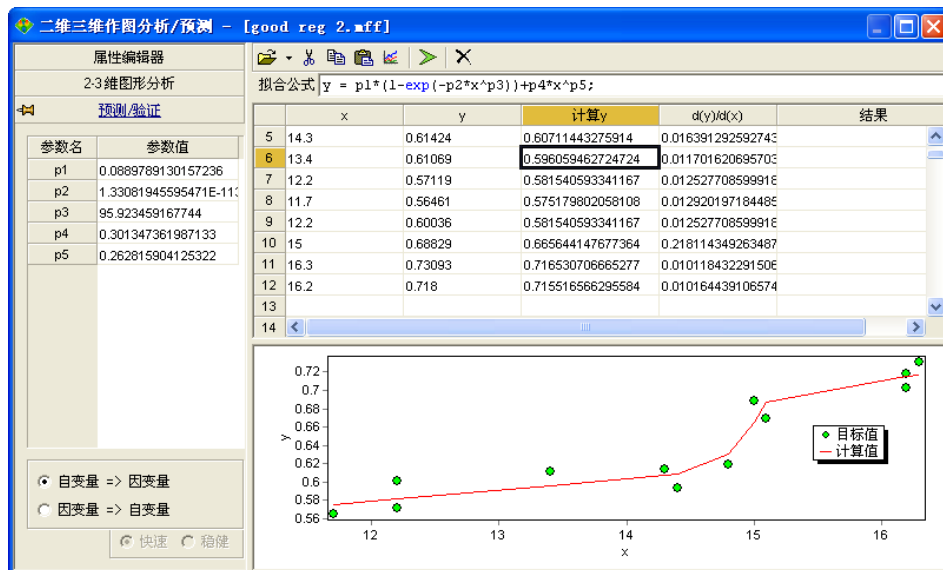


图 2-17 拟合预测验证计算

- 逆计算：输入任一因变量值计算对应的自变量



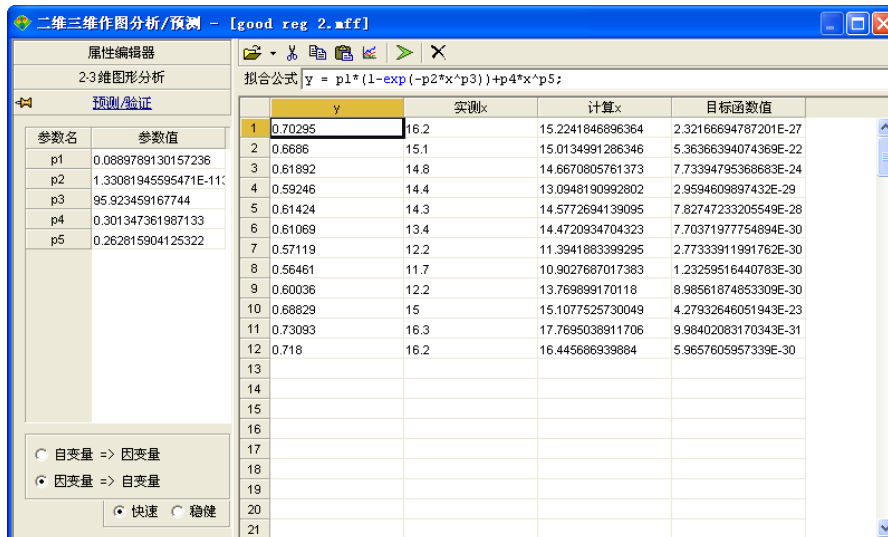


图 2-18 由因变量反求自变量

- 参数分析：参数影响二维-三维图形

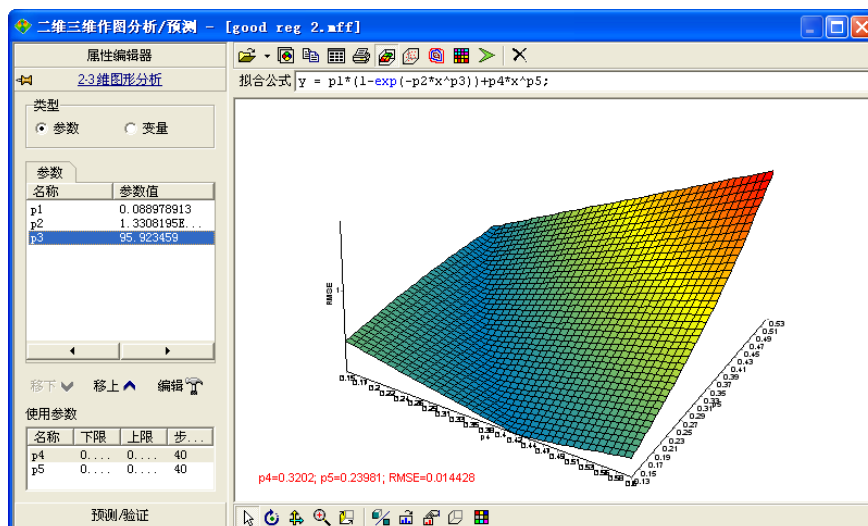


图 2-19 参数影响分析三维示图

- 变量二维-三维图形展示

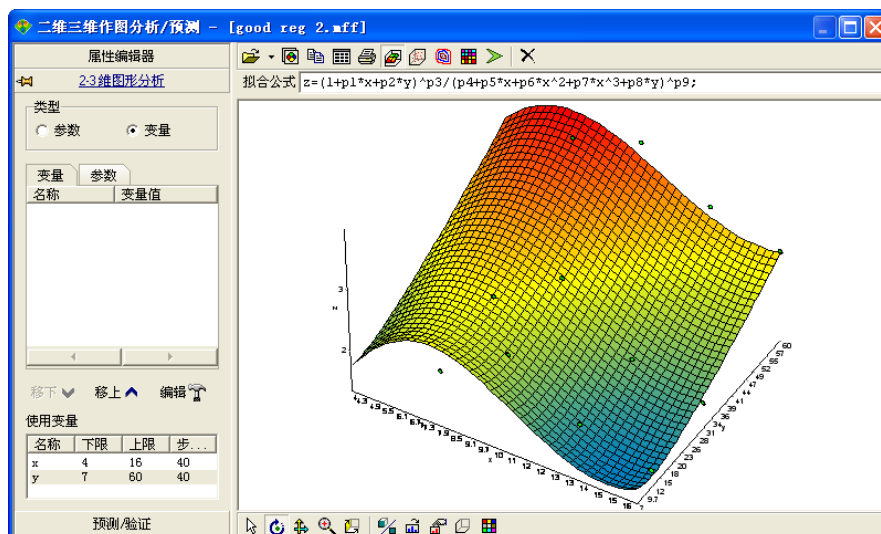


图 2-20 三维空间拟合示图

## 2.2.1 共享模式拟合

共享模式拟合指不同的拟合公式拥有一个或几个相同的参数，称之为共享参数。求解主要由关键字“SharedModel”来实现。

本例中有一个自变量，四个因变量，共有 11 个参数：m1, m2, v, p1, p2, p3, p4, c1, c2, c3, c4, 其中前三个参数 m1, m2, v 为共享参数，数据见表 2-10。

$$\text{四个拟合公式: } \begin{cases} y_1 = \frac{v \cdot x}{k_1 + x} + p_1 \cdot x + c_1 \\ y_2 = \frac{v \cdot x}{k_2 + x} + p_2 \cdot x + c_2 \\ y_3 = \frac{v \cdot x}{k_3 + x} + p_3 \cdot x + c_3 \\ y_4 = \frac{v \cdot x}{k_4 + x} + p_4 \cdot x + c_4 \end{cases} \quad (2-24)$$

$$\text{其中: } k_1 = m_1 \cdot (1 + \frac{1}{m_2}), k_2 = m_1 \cdot (1 + \frac{3}{m_2}), k_3 = m_1 \cdot (1 + \frac{10}{m_2}), k_4 = m_1 \cdot (1 + \frac{30}{m_2})$$

表 2-10 拟合数据

x	y1	y2	y3	y4
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

1stOpt 代码

```
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

运行结果:

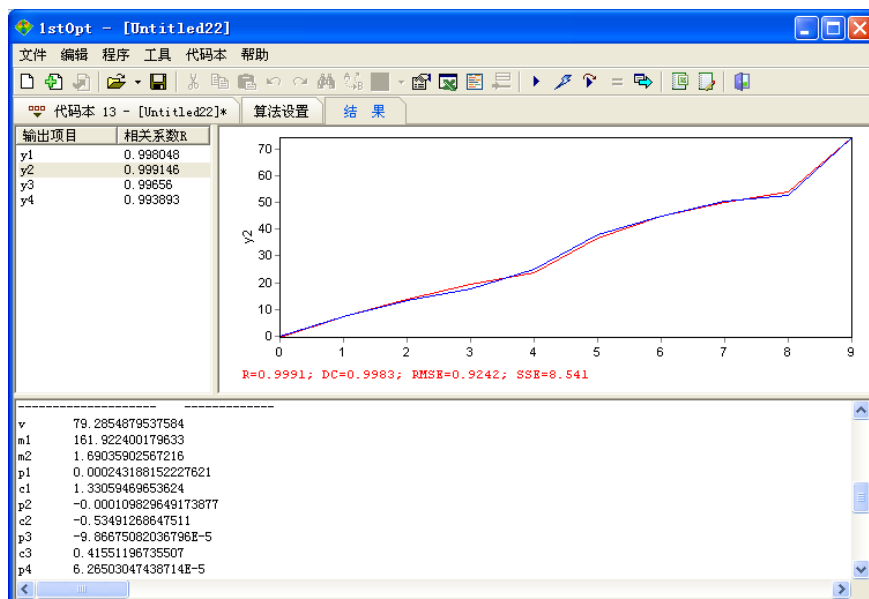


图 2-21 共享模式拟合结果

## 2.2.2 缺少变量值的特殊拟合

例 1. 如已知参数方程及 x、y 数据如下，

$$\begin{aligned} x &= r \cdot (a - \sin(a+b)) \cdot \cos(b) + r \cdot (1 - \cos(a \cdot b)) \cdot \sin(b) \\ y &= r \cdot (a - \cos(a-b)) \cdot \cos(b) - r \cdot (a - \sin(a \cdot b)) \cdot \sin(b) \end{aligned} \quad (2-25)$$

表 2-11 拟合数据

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

现在需要用 x、y 数据对上面参数方程进行曲线拟合，该参数方程是摆线方程，其中 r 是摆线的滚圆半径，a 是摆角，其值范围从 0 到  $2\pi$ ，b 是曲线的旋转初始角，如何求出参数 r 和 b？

公式 2-25 中两式均含有变量 a，如果对应于 x 和 y 各点的 a 已知，则问题就比较好解决，用关键字 SharedModel 即可求解，但本题中 a 值为未知数，并且通过一定的数学变换也无法消除 a 而形成诸如  $y=f(x)$  的一般二维拟合形式。1stOpt 中关键字 ParVariable 专门用于定义这类未知变量。代码如下：

```
Variable x,y;
ParVariable a[0,2*pi];
SharedModel;
Function x=r*(a-sin(a+b))*cos(b)+r*(1-cos(a*b))*sin(b);
        y=r*(1-cos(a-b))*cos(b)-r*(a-sin(a*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;
```

本题求解有一定难度，很多时候会陷入一局部最优如下表。1stOpt 不仅可求出参数 r 和 b 值，而且给出了对应与 x 和 y 的 a 值。

结果

最优解	局部最优解																																																												
均方差(RMSE): 0.473947739744703 残差平方和(SSE): 5.3910350402187 相关系数(R): 0.999958522301281 相关系数之平方(R^2): 0.999917046322962 决定系数(DC): 0.999917046322961 F 统计(F-Statistic): -712.581671464623	均方差(RMSE): 0.512540059756795 残差平方和(SSE): 6.30473550853197 相关系数(R): 0.999951492276122 相关系数之平方(R^2): 0.999902986905244 决定系数(DC): 0.999902986905244 F 统计(F-Statistic): -190.095206680829																																																												
<table><tr><th>参数</th><th>最佳估算</th></tr><tr><td>r</td><td>14.9570078354002</td></tr><tr><td>b</td><td>-6.36129367898334</td></tr><tr><td>a0</td><td>2.80018799250962</td></tr><tr><td>a1</td><td>2.97946628688341</td></tr><tr><td>a2</td><td>3.18398254437925</td></tr><tr><td>a3</td><td>3.43219540311849</td></tr><tr><td>a4</td><td>3.60947190670764</td></tr><tr><td>a5</td><td>3.75895134334236</td></tr><tr><td>a6</td><td>4.0020662840376</td></tr><tr><td>a7</td><td>4.31207489147291</td></tr><tr><td>a8</td><td>4.5384582291118</td></tr><tr><td>a9</td><td>4.7531139284785</td></tr><tr><td>a10</td><td>5.28361370615172</td></tr><tr><td>a11</td><td>6.06628119473038</td></tr></table>	参数	最佳估算	r	14.9570078354002	b	-6.36129367898334	a0	2.80018799250962	a1	2.97946628688341	a2	3.18398254437925	a3	3.43219540311849	a4	3.60947190670764	a5	3.75895134334236	a6	4.0020662840376	a7	4.31207489147291	a8	4.5384582291118	a9	4.7531139284785	a10	5.28361370615172	a11	6.06628119473038	<table><tr><th>参数</th><th>最佳估算</th></tr><tr><td>r</td><td>14.8678109018936</td></tr><tr><td>b</td><td>-0.0676584084342278</td></tr><tr><td>a0</td><td>2.77832408158674</td></tr><tr><td>a1</td><td>2.9868819988592</td></tr><tr><td>a2</td><td>3.15581577602873</td></tr><tr><td>a3</td><td>3.35738415118238</td></tr><tr><td>a4</td><td>3.56254331491974</td></tr><tr><td>a5</td><td>3.74652240154359</td></tr><tr><td>a6</td><td>4.00578269856165</td></tr><tr><td>a7</td><td>4.22454273139407</td></tr><tr><td>a8</td><td>4.48622591894176</td></tr><tr><td>a9</td><td>4.77012549091088</td></tr><tr><td>a10</td><td>5.1522561082451</td></tr><tr><td>a11</td><td>5.67387853331666</td></tr></table>	参数	最佳估算	r	14.8678109018936	b	-0.0676584084342278	a0	2.77832408158674	a1	2.9868819988592	a2	3.15581577602873	a3	3.35738415118238	a4	3.56254331491974	a5	3.74652240154359	a6	4.00578269856165	a7	4.22454273139407	a8	4.48622591894176	a9	4.77012549091088	a10	5.1522561082451	a11	5.67387853331666
参数	最佳估算																																																												
r	14.9570078354002																																																												
b	-6.36129367898334																																																												
a0	2.80018799250962																																																												
a1	2.97946628688341																																																												
a2	3.18398254437925																																																												
a3	3.43219540311849																																																												
a4	3.60947190670764																																																												
a5	3.75895134334236																																																												
a6	4.0020662840376																																																												
a7	4.31207489147291																																																												
a8	4.5384582291118																																																												
a9	4.7531139284785																																																												
a10	5.28361370615172																																																												
a11	6.06628119473038																																																												
参数	最佳估算																																																												
r	14.8678109018936																																																												
b	-0.0676584084342278																																																												
a0	2.77832408158674																																																												
a1	2.9868819988592																																																												
a2	3.15581577602873																																																												
a3	3.35738415118238																																																												
a4	3.56254331491974																																																												
a5	3.74652240154359																																																												
a6	4.00578269856165																																																												
a7	4.22454273139407																																																												
a8	4.48622591894176																																																												
a9	4.77012549091088																																																												
a10	5.1522561082451																																																												
a11	5.67387853331666																																																												

例 2. 已知参数方程如下：

$$\begin{aligned}x &= r \cdot (\cos(t) + (t - b) \cdot \sin(t)) + x_0 \\ y &= r \cdot (\sin(t) - (t - b) \cdot \cos(t)) + y_0\end{aligned}\tag{2-26}$$

表 2-12 拟合数据

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

中间变量 t 未知，待求参数 x0、y0、r 和 b。

求解方法如例 1，采用 SharedModel 和 ParVariable 两个关键字。该道题求解难度较大，上面 1stOpt 代码能以大概 50% 的概率求得正解。

1stOpt 代码

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
---

结果

迭代数: 21 计算用时(时:分:秒:微秒): 00:00:57:32 优化算法: 通用全局优化法(UGO1) 计算结束原因: 达到收敛判定标准 均方差(RMSE): 1.34537356121028E-5 残差平方和(RSS): 3.25805403456652E-9 相关系数(R): 0.999999999999975 相关系数的平方( $R^2$ ): 0.999999999999951 决定系数(DC): 0.999999999999951 F 统计(F-Statistic): -2008250717607.4	
参数	最佳估算

62.7992	58.0891
68.3962	50.8058
73.1584	42.9946
79.9955	26.1718
83.0531	8.4225

-----	
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 批处理拟合

当公式已知且数据有多组时，可用此功能快速拟合所有数据。

例：已知 x、y 数据如下表 2-13，试拟合 B = 1, 2, 3, 4 时的 x-y 关系。

表 2-13 拟合数据

序号	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

拟合公式：
$$y = y_0 + \frac{A}{W \cdot \sqrt{\frac{\pi}{2}}} \cdot \exp\left(-\frac{(2 \cdot (x - x_0)^2)}{W^2}\right) + B$$
(2-27)

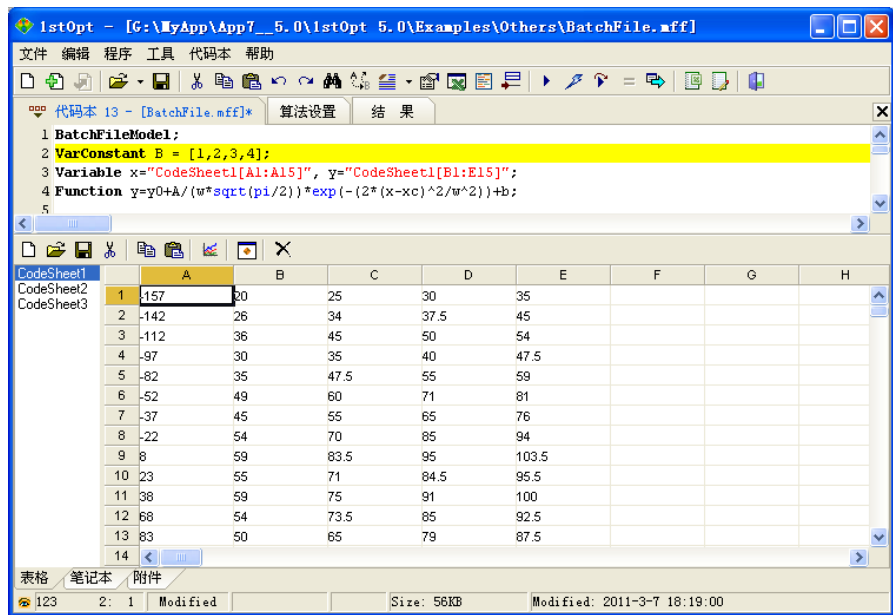


图 2-22 批处理模式拟合

### 1stOpt 代码

```
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 - xc)^2 / w^2)) + B;
```

在上述代码中注意三点：

- 关键字 “BatchFileModel”：表示将分别求各 y 与 x 的关系
- 关键字 “VarConstant”：定义不同拟合时对应的 B 值
- “Variable” 定义变量时，直接从代码表格中读取数据

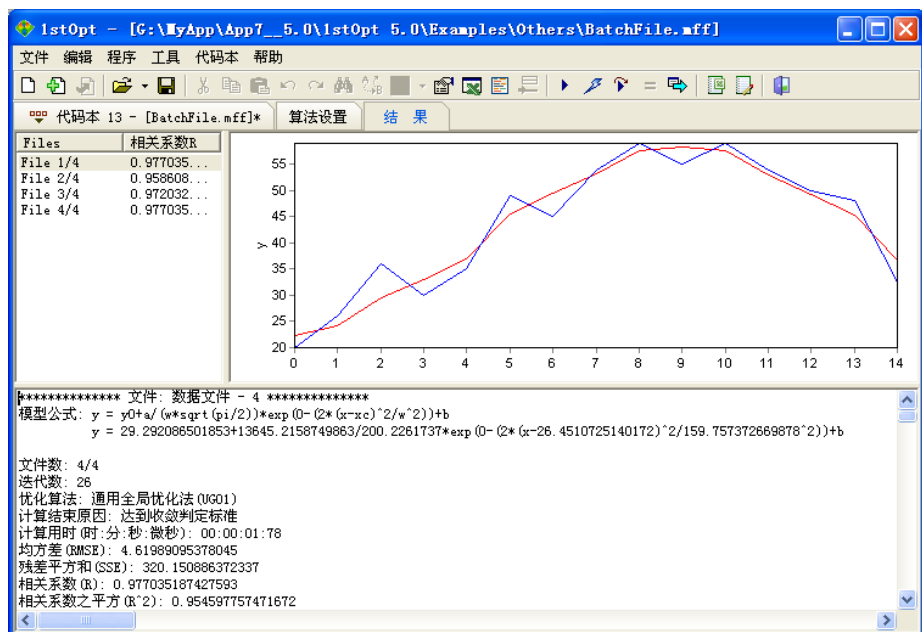


图 2-23 批处理模式拟合结果

2.2.4 权重拟合

带权重的拟合可由关键字 “WeightedReg” 来设定，定义见表 2-14。

表 2-14 权重拟合定义

WeirhtedReg 值	目标函数	说明
1	$\sum_{i=1}^n \left( \frac{(y_i - y'_i)^2}{y_i^2} \right)$	y: 实际因变量 y': 计算因变量 n: 拟合数据长度 StdDev: 标准偏差
2	$\sum_{i=1}^n \left( \frac{(y_i - y'_i)^2}{StdDev} \right)$	
3	$\sum_{i=1}^n \left( \frac{(y_i - y'_i)^2}{y_i} \right)$	

例. 权重拟合

表 2-15 权重拟合数据

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

拟合公式:

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

(2-28)

1stOpt 代码

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

2.2.5 带约束拟合

例 1. 有约束的非线性回归

拟合公式及数据如下:

公式: 
$$y = \frac{a + b \cdot x}{1 + c \cdot x + d \cdot x^2}$$

(2-29)

表 2-16 约束拟合数据一

x	1	2	8	12	17	21	24
y	1	2	3	6	6	4	4

与一般拟合问题不同之处是，该例拟合后的曲线必须通过点 1 和 3，即点 (x1, y1)=(1, 1)，(x2, y2)=(8, 3)；此约束条件即：

$$y_1 = \frac{a+b \cdot x_1}{1+c \cdot x_1+d \cdot x_1^2} \quad \text{及} \quad y_2 = \frac{a+b \cdot x_2}{1+c \cdot x_2+d \cdot x_2^2} \quad (2-30)$$

1stOpt 代码:

```
Constant x1=1,y1=1,x2=8,y2=3;
Function y=(a+b*x)/(1+c*x+d*x^2);
      y1=(a+b*x1)/(1+c*x1+d*x1^2);
      y2=(a+b*x2)/(1+c*x2+d*x2^2);
Data;
1      1
2      2
8      3
12     6
17     6
21     4
24     4
```

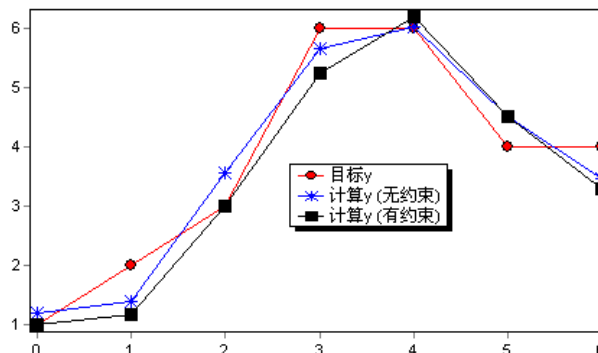


图 2-24 约束拟合对比图

结果:  $a = 0.86154$ ,  $b = 0.036450$ ,  $c = -0.105584$ ,  $d = 0.0035789$

如果无约束条件, 本例答案为:  $a = 1.011742$ ,  $b = 0.056511$ ,  $c = -0.103843$ ,  $d = 0.003781868$

## 例 2. 约束非线性回归

约束优化可以直接将约束写在拟合模式下, 也可写成约束函数优化的形式。

$$\text{拟合公式: } y = \frac{p_3 + p_2 \cdot x^{p_4}}{p_5 + p_1 \cdot x^{p_4}} \quad (2-31)$$

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

表 2-17 约束拟合数据二

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

1stOpt 代码

```
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
```

左边拟合的代码也可写成优化形式

```
Algorithm = UGO[100];
DataSet;
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;
EndDataSet;
MinFunction
Sum(x,y)/((p3+p2*p1*x^p4)/(p5+p1*x^p4)-y)^2);
p2=3*p3+sum(i=1:3)(p[i]);
```



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	20.3 >=p1+p2>=20;
--	-------------------

本题虽然看似简单，但却是一道极有难度的优化问题，一个极易陷入的局部最优解是：  
Min. = 2.32592892, p1 = 7.65783913843992E-02, p2 = 20.0086306479937,  
p3 = -1.91445978460306E-02, p4 = .423355212168873,  
p5 = 4.99918501444038E-02  
而最优解为：  
Min. = 2.235333, p1 = -3.83814476105978E-17, p2 = 20.0000612864639,  
p3 = -2.41557981892391E-17, p4 = .486170393747984,  
p5 = -2.66116003491166E-17  
注意代码中“Algorithm = UG01[100];”，本句设定算法为通用优化算法，第一种局部搜索类型，并行数为 100；如果使用缺省的并行数 30，将较难获得最优解。

### 2.2.6 带积分的拟合

1stOpt 支持定积分函数，用“Int()”来表示，如  $\int_{t=0.1}^c (t+x)^t dt$  可写为  
“Int((t+x)^t, t=0.1, c)”

例 1：拟合公式和数据如下，结果见图 2-25。

$$y = a - b \cdot \exp(-c_1 \cdot x^d) \cdot \int_{t=0.1}^c (t+x) dt \tag{2-32}$$

表 2-18 积分拟合数据一

x	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
y	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
1stOpt 代码	
Parameter a,b,c,d; Variable x,y; Function y=a-b*exp(-c*x^d)*int((t+x),t=0.1,c); Data; 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;	

例 2. 拟合公式和数据如下，结果见图 2-26。

$$y = \frac{\int_{u=0}^x \left( (p_1 \cdot (x-u) + x^{p_2}) \cdot \exp(-p_3 \cdot (x-u)^2 + p_4) \right)^2 dt}{x^{p_5}} \quad (2-33)$$

表 2-19 积分拟合数据二

x	0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9,9.5
y	0.063,0.102,0.086,0.075,0.062,0.056,0.051,0.045,0.041,0.040,0.038,0.034,0.030,0.028,0.028,0.028,0.028,0.027,0.024

1stOpt 代码

```
Parameter p(5);
Variable x,y;
Function y=int(((p1*(x-u)+x^p2)*exp(-p3*(x-u)^2)+p4)^2,u=0,x)/x^p5;
Data;
0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9,9.5;
0.063,0.102,0.086,0.075,0.062,0.056,0.051,0.045,0.041,0.040,0.038,0.034,0.030,0.028,0.028,0.028,0.028,0.027,0.024;
```

结果

均方差(RMSE): 0.0012265480246249	参数	最佳估算
残差平方和(SSE): 2.85839810775136E-5		
相关系数(R): 0.998394035240528	p1	-0.418582664997966
相关系数之平方(R^2): 0.996790649603864	p2	-0.142141026724806
决定系数(DC): 0.996790632184511	p3	-1.43008621946461E-5
F 统计(F-Statistic): 1090.55745592881	p4	-1.08555143919631
	p5	3.49832672593817

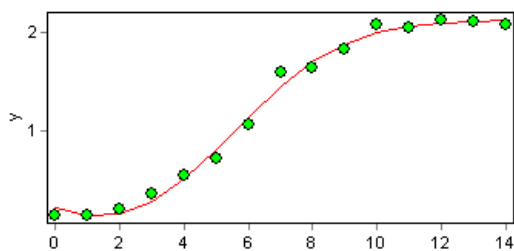


图 2-25 积分拟合结果

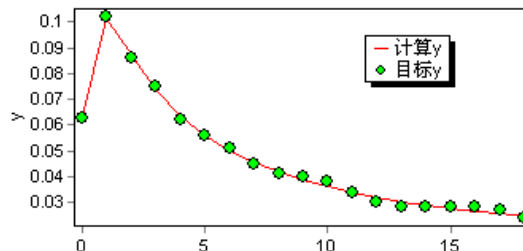


图 2-26 积分拟合结

## 2.2.7 最小一乘及其它特殊拟合

非线性拟合一般均指最小二乘法,即计算因变量与实际因变量间误差平方和最小,如下:

$$\min RSS = \sum_{i=1}^n (Y_i - y_i)^2 \quad (2-34)$$

其中, Y、y 分别为计算和实际因变量值, RSS (Residue of Sum of Square) 为平方和残差值, n 为拟合数据长度。

但有的情形或特殊情况下需要采取有别于最小二乘法的判断准则, 如最小一乘等。1stOpt 有四种方法, 如下表示。1stOpt 中用关键字 “RegType” 进行定义。

表 2-20 拟合类型方法

RegType	拟合方法	拟合准则
---------	------	------

0	最小二乘法: RSS (Residue of Sum of Square — 残差平方和)	$\min. RSS = \sum_{i=1}^n (Y_i - y_i)^2$
1	最小一乘法: SAE (Sum of Absolute Error — 绝对差之和)	$\min. SAE = \sum_{i=1}^n  Y_i - y_i $
2	最大绝对差值最小: MAE (Max. Absolute Error)	$\min. MAE = \text{Max}( Y_1 - y_1 ,  Y_2 - y_2 , \dots,  Y_n - y_n )$
3	最大相对绝对差值最小: MRAE (Max. Relative Absolute Error)	$\min. MRAE = \text{Max}\left(\left \frac{Y_1 - y_1}{y_1}\right , \left \frac{Y_2 - y_2}{y_2}\right , \dots, \left \frac{Y_n - y_n}{y_n}\right \right)$

例 1: 拟合公式及数据如下

$$\text{拟合公式: } y = a \cdot (x - x_0)^4 + b \cdot (x - x_0)^2 + c \quad (2-35)$$

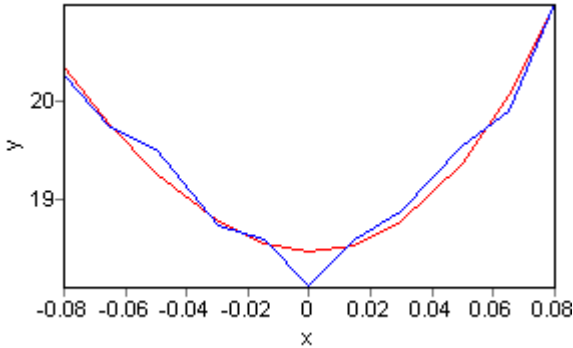
表 2-21 拟合数据

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

1stOpt 最小二乘拟合代码

Parameter a,b,c,x0;
Variable x,y;
Function y=a*(x-x0)^4+b*(x-x0)^2+c;
Data;
-0.08,-0.065,-0.05,-0.03,-0.015,0.015,0.03,0.05,0.065,0.08,0;
20.26,19.73,19.50,18.73,18.59,18.59,18.88,19.55,19.89,20.99,18.12;

由“通用全局优化法”可以很容易得到如下结果:

	迭代数: 36 计算用时(时:分:秒:微秒): 00:00:03:120 优化算法: 通用全局优化法(UGO2) 计算结束原因: 达到收敛判断标准 均方差(RMSE): 0.154577732398945 残差平方和(RSS): 0.262837028889598 相关系数(R): 0.981806182300711 相关系数之平方(R^2): 0.963943379603898 决定系数(DC): 0.963943379603897 卡方系数(Chi-Square): 0.00694913498803648 F 统计(F-Statistic): 64.7130348795956
	参数 最佳估算 <hr/> a                    421.902418131742 b                    -170.887335709735 c                    35.768453358506 x0                   -0.448782169655507

缺省状态下 RegType = 0, 在代码中通过加入和改变“RegType”的值即可得到不同拟合准则下的计算结果。本例题当 RegType 值不等于 0 时, 即当拟合准则不是最小二乘法时, 从任意随




机值出发，1stOpt 较难求得稳定的最优参数组值，这时可采用 1stOpt 的热执行方式：首先以最小二乘法（RegType = 0）进行拟合，在得到稳定最优解后，在代码中改变 RegType 值，再按热执行键进行计算。当出现下列提示时，选择“OK”按钮，对于 RegType 不等于 0 时的其它三种情况，均可获得理想结果。

表 2-22 不同拟合类型计算结果对比

目标值	计算值			
	RegType = 0	RegType = 1	RegType = 2	RegType = 3
20.26	20.3313	20.2600	20.4821	20.4977
19.73	19.7514	19.7125	19.8319	19.8664
19.5	19.2625	19.2520	19.2779	19.2713
18.73	18.7752	18.7959	18.7173	18.6567
18.59	18.5513	18.5900	18.4528	18.3942
18.59	18.5311	18.5900	18.4040	18.4777
18.88	18.7655	18.8259	18.6579	18.8155
19.55	19.3674	19.4201	19.3302	19.5036
19.89	20.0577	20.0967	20.1121	20.1233
20.99	20.9718	20.9900	21.1562	20.7438
18.12	18.4649	18.5165	18.3421	18.3326

表 2-23 不同拟合类型参数结果对比

RegType	参数				RSS	SAE	MAE	MRAE
	a	b	c	x <sub>0</sub>				
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	-16517.0595	464.1539	18.3283	-0.0030	0.35030	1.78730	0.24622	0.011730

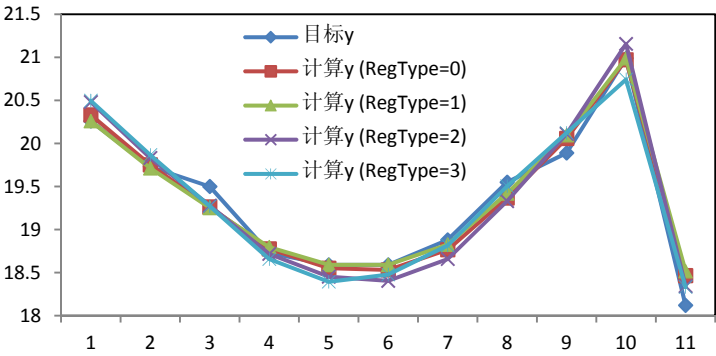


图 2-27 不同准则拟合结果对比

### 2.2.8 隐函数拟合

在有些情况下，非线性回归的模型式为隐函数，如下式：

$$\begin{aligned}
 &-(x-p_4)\cdot\sin(p_3)+(y-p_5)\cdot\cos(p_3)+p_5= \\
 &p_1+p_2\cdot\sin(p_3)\cdot\ln((x-p_4)*\cos(p_3)+(y-p_5)\cdot\sin(p_3)+p_4)
 \end{aligned}
 \tag{2-36}$$

其中，x 为自变量，y 为因变量, 上式分解成标准的显示。对于此类问题，1stOpt 可轻易处理，书写格式与一般无异。

例 1：拟合公式及数据如下

$$y = a - b \cdot \exp(-c \cdot x^d + y) \quad (2-37)$$

表 2-24 隐函数拟合数据一

x	0.05,0.15,0.25,0.35,0.45,0.55,0.65,0.75,0.85,0.95
y	0.13,0.13,0.19,0.34,0.53,0.71,1.06,1.6,1.64,1.83
1stOpt 代码	
Parameters a, b, c, d; Function y=a-b*exp(-c*x^d+y); Data; 0.05,0.15,0.25,0.35,0.45,0.55,0.65,0.75,0.85,0.95; 0.13,0.13,0.19,0.34,0.53,0.71,1.06,1.6,1.64,1.83;	
结果	
目标函数值(最小): 0.0185659981797815 均方差(RMSE): 0.0198740405308961 残差和(SSE): 0.003949774870237 相关系数(R): 0.999602497389836 决定系数(DC): 0.999020888521126 参数 最佳估算 -----	
a	0.16951239446793
b	-0.28277738509661
c	0.00075789453416385
d	-6.79556201175718

例 2. 渐开线渐隐函数非线性回归

已知渐开线参数方程及数据如下：

$$\begin{cases} x = R \cdot (\cos(t) - (t - B) \cdot \sin(t)) + x_0 \\ y = R \cdot (\sin(t) - (t - B) \cdot \cos(t)) + y_0 \end{cases} \quad (2-38)$$

其中，t 是自变量，R，B，x0，y0 为待求渐开线方程参数。

表 2-25 隐函数拟合数据二

序号	x	y	序号	x	y
1	15.5910	86.2142	6	68.3962	50.8058
2	24.6601	83.7114	7	73.1584	42.9946
3	33.3732	80.2618	8	79.9955	26.1718
4	49.3445	70.7216	9	83.0531	8.4225
5	62.7992	58.0891			

此题难点是不知到对应于 x 及 y 的自变量 t 值。运用 1stOpt，此题有三种解法：

- ✧ 方法 1：将未知 t 变量值当作参数来求，即有 9 个 t 变量参数及 R，B，x0，y0 共 13 个参数。优化目标函数是最小化下列函数：

$$\begin{aligned} f &= \sum_{i=1}^9 (y_i - y_i')^2 + \sum_{i=1}^9 (x_i - x_i')^2 \\ \min &= \sum_{i=1}^9 (y_i - (R \cdot (\sin(t) - (t - B) \cdot \cos(t)) + y_0))^2 + \\ &\quad \sum_{i=1}^9 (x_i - (R \cdot (\cos(t) - (t - B) \cdot \sin(t)) + x_0))^2 \end{aligned} \quad (2-39)$$

1stOpt 代码

```
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);	

结果: 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

该种解法虽能得到正解,但当数据较多时,如 500 组数据,即有 500 个 t 值要被当作参数来求,如此多的参数量将会大大增加求解的难度,况且中间变量 t 值也并非我们想获得的值。

✧ 方法 2: 用关键字 “ParVariable” 定义缺失变量 t, 再用共享模式 “SharedModel” 进行直接拟合

1stOpt 代码	最好结果
Parameter R ,B,x0,y0;	均方差(RMSE): 1.34537356112074E-5
ParVariable t;	残差平方和(SSE): 3.25805403413287E-9
Variable x, y;	相关系数(R): 0.999999999999975
SharedModel;	相关系数之平方(R^2): 0.999999999999951
Function x = R*(cos(t)+(t-B)*sin(t))+x0;	决定系数(DC): 0.999999999999951
Function y = R*(sin(t)-(t-b)*cos(t))+y0;	F 统计(F-Statistic): -2008243491476.2
Data;	
15.5910 86.2142	参数 最佳估算
24.6601 83.7114	-----
33.3732 80.2618	r -3.19952276416274
49.3445 70.7216	b -27.5800926938431
62.7992 58.0891	x0 -0.000329646830102739
68.3962 50.8058	y0 0.000374659541623336
73.1584 42.9946	t0 -0.215438422270183
79.9955 26.1718	t1 -0.323157595214207
83.0531 8.4225	t2 -0.430878058310167
	t3 -0.646316755110223
	t4 -0.861756728203396
	t5 -0.969475917532324
	t6 -1.07719566111866
	t7 -1.29263476604708
	t8 -1.50807217876265

✧ 方法 3: 将公式进行变换, 消除 t 变量  
由公式

$$\begin{cases} x = R \cdot (\cos(t) + (t - B) \cdot \sin(t)) + x_0 \\ y = R \cdot (\sin(t) + (t - B) \cdot \cos(t)) + y_0 \end{cases} \quad (2-40)$$

变换得

$$\frac{(x-x_0)^2}{R^2} + \frac{(y-y_0)^2}{R^2} = 1 + (t-B)^2 \quad (2-41)$$

即

$$t = \sqrt{\frac{(x-x_0)^2}{R^2} + \frac{(y-y_0)^2}{R^2} - 1} + B \quad (2-42)$$

目标函数如方法 1

$$f = \sum_{i=1}^9 \left( y_i - \left( R \cdot (\sin(t) - (t-B) \cdot \cos(t)) + y_0 \right) \right)^2 + \sum_{i=1}^9 \left( x_i - \left( R \cdot (\cos(t) - (t-B) \cdot \sin(t)) + x_0 \right) \right)^2 \quad (2-43)$$

再将上述  $t$  代入目标函数用于消除  $t$ 。注意关键词 ConstStr。

lstOpt 代码:

```
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);
```

结果:  $R = 3.199541$ ,  $B = 0.694399$ ,  $x_0 = -0.0003171$ ,  $y_0 = 0.00035955$

## 2.2.9 分峰拟合

分峰拟合主要包括两个步骤: 拟合与分峰。

例 1. 使用三个高斯 (Gauss) 和三个洛伦兹 (Lorentz) 函数进行多峰拟合, 数据和公式如下:

表 2-26 分峰拟合数据

序号	x	y	序号	x	y	序号	x	y
1	1	6.6	21	21	23.7	41	41	440
2	2	9	22	22	41.1	42	42	480
3	3	10.6	23	23	60	43	43	520
4	4	12.9	24	24	88	44	44	540
5	5	13.5	25	25	132	45	45	555
6	6	13.6	26	26	170	46	46	570
7	7	13.3	27	27	233	47	47	600
8	8	12.9	28	28	322	48	48	515

9	9	12	29	29	375	49	49	400
10	10	11.3	30	30	400	50	50	330
11	11	10.6	31	31	416	51	51	285
12	12	9.9	32	32	425	52	52	260
13	13	9.4	33	33	420	53	53	245
14	14	8.9	34	34	412	54	54	250
15	15	8.3	35	35	408	55	55	260
16	16	7.9	36	36	405	56	56	290
17	17	7.7	37	37	400	57	57	377
18	18	8	38	38	405	58	58	492
19	19	10.5	39	39	409	59	59	523
20	20	16.6	40	40	420			

高斯函数:  $y = a \cdot \exp\left(-0.5 \cdot \left(\frac{x-b}{c}\right)^2\right)$ , a、b、c 为待求参数。

洛伦兹函数:  $y = d + \frac{e}{4 \cdot (x-f)^2 + g^2}$ , d、e、f、g 为待求参数。


1stOpt 代码

```
ParameterDomain = [0,];
Function y= Sum(i=1:3,a,b,c)(a*exp(-0.5*((x-b)/c)^2))+Sum(i=1:3,d,e,f,g)(d + e/(4*(x-f)^2 + g^2));
Data;
1    6.6
2    9
3    10.6
4    12.9
...
```

三个高斯和三个洛伦兹函数共计 21 个待求参数，且要求均大于 0。该问题代码如上，要拟合出最优答案难度极大。所得最佳结果如下：

均方差(RMSE): 3.71031606427381	参数 最佳估算	e1	433.393060771498
残差平方和(SSE): 812.220272511688	-----	f1	28.676300872629
相关系数(R): 0.999833423094657	a1	g1	2.71578884800157
相关系数之平方(R^2): 0.99966687393718	b1	d2	1.28294006664369
决定系数(DC): 0.99966687393718	c1	e2	1600.30233847297
卡方系数(Chi-Square): 12.048957519855	a2	f2	47.2507551223418
F 统计(F-Statistic): 5703.54653104543	b2	g2	2.78837542950084
	c2	d3	1.33531345829328
	a3	e3	10973.3932201694
	b3	f3	58.8304994945306
	c3	g3	4.79876604799094
	d1		

拟合计算完毕后进行分峰，

✧ 从工具栏中选“二维-三维分析/预测”按钮，



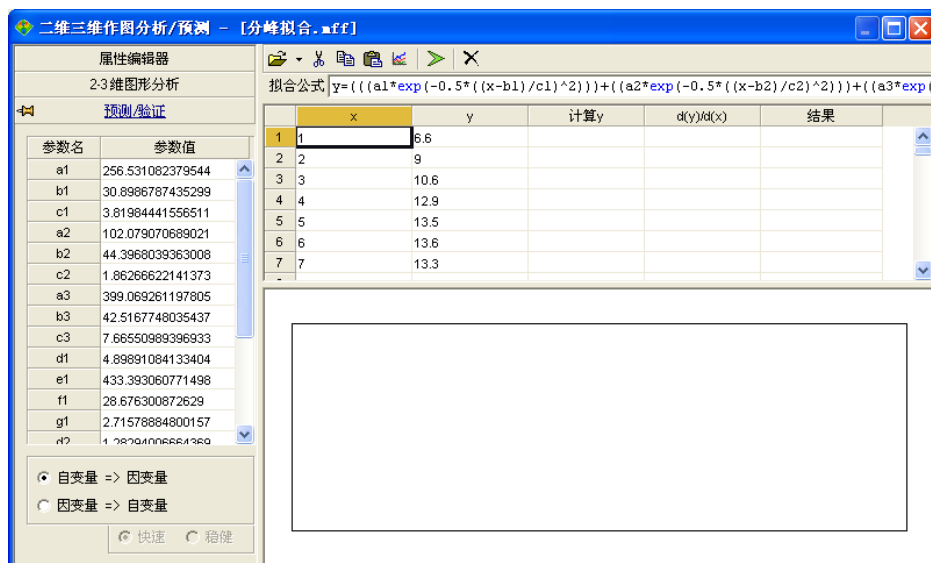


图 2-28 拟合验证预测一

- ✧ 选 2-3 维图形分析，双击“拟合公式”标签或拖动放大拟合公式编辑框

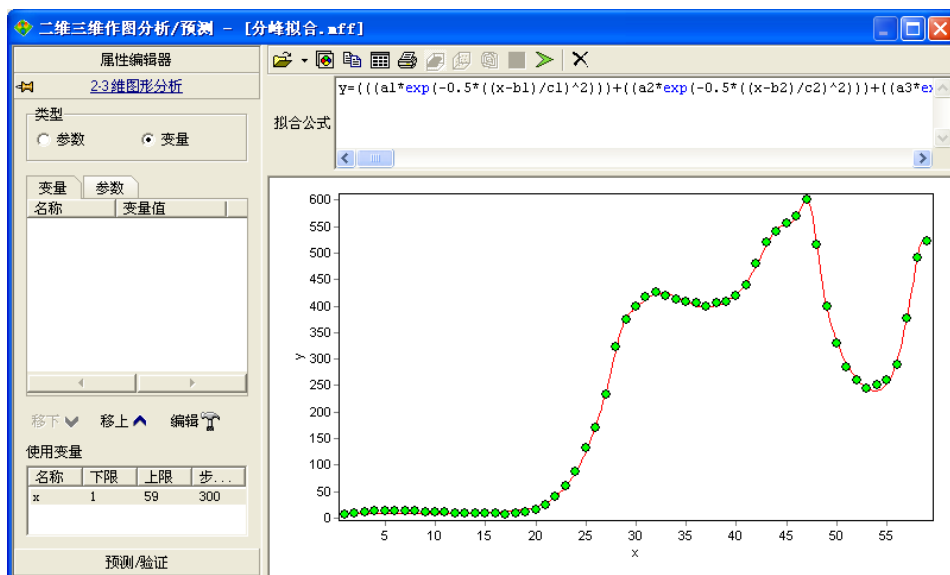


图 2-29 拟合验证预测二

- ✧ 在公式编辑框里依次选择每一个高斯和洛伦兹公式，按右键，从弹出菜单选“添加成公式”

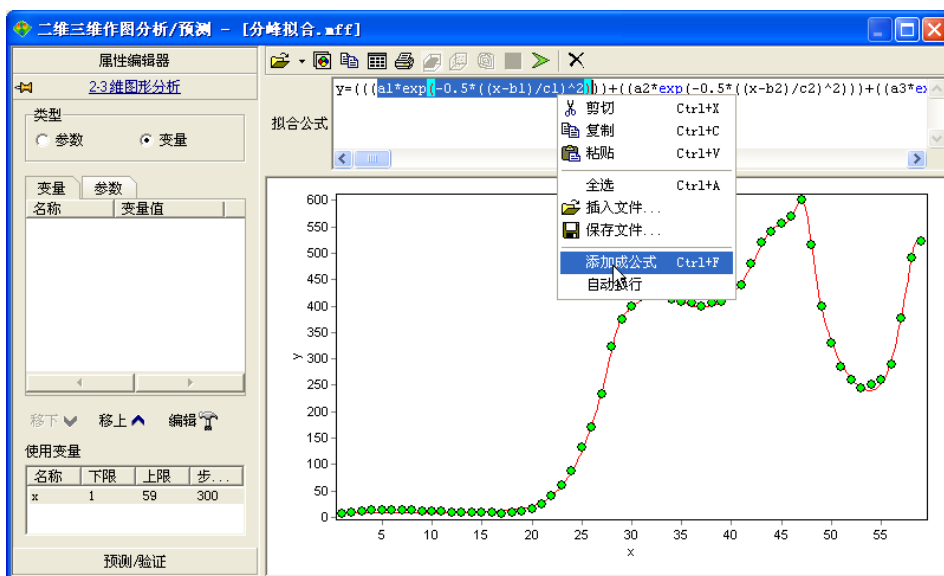


图 2-30 拟合验证预测三

✧ 获得分峰结果

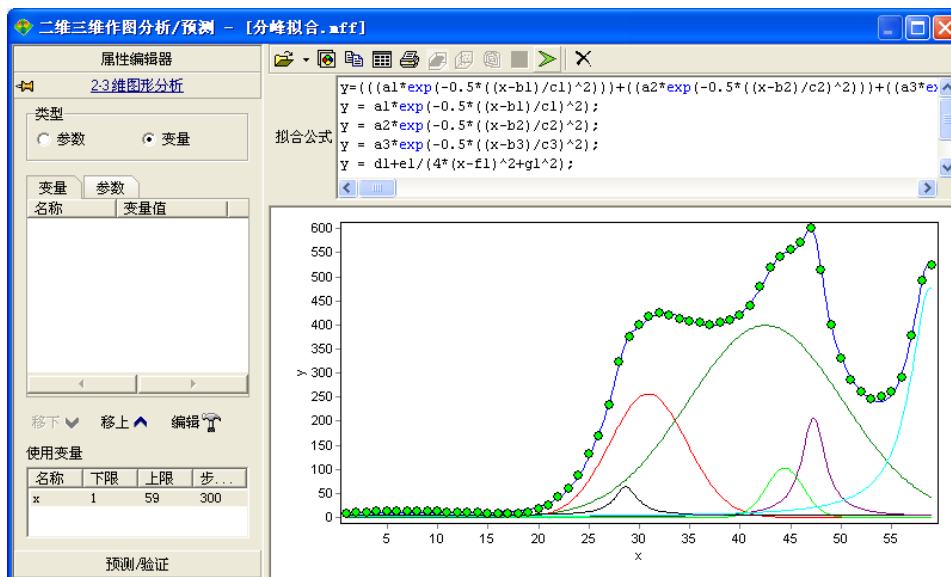



图 2-31 拟合验证预测四

✧ 选“查看数据”按钮, 可获取详细数据

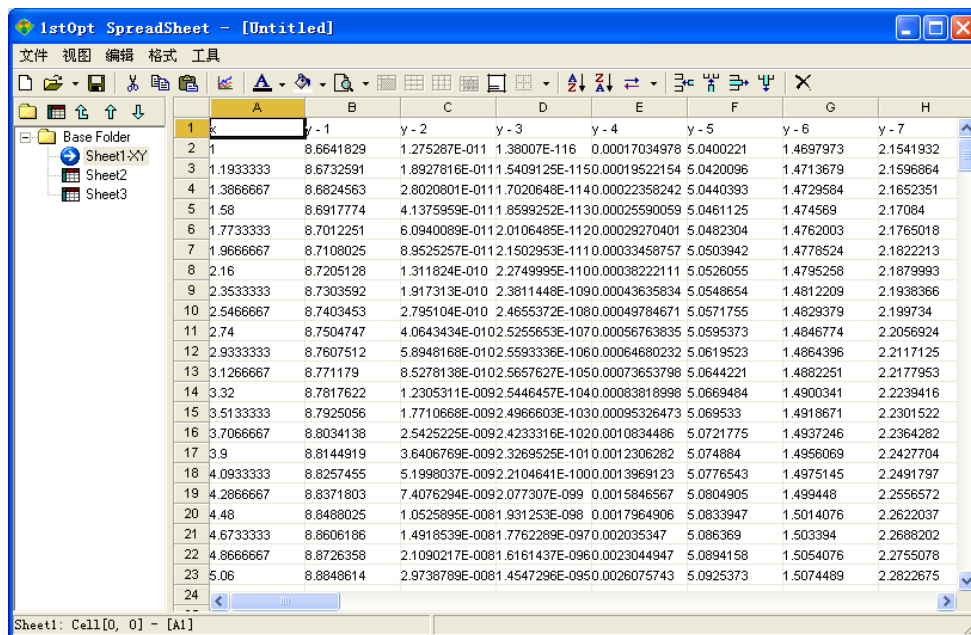


图 2-32 拟合验证预测四

## 2.2.10 StepReg 的使用

StepReg 是用于拟合的一个特殊命令，主要用于周期性变化的数据拟合，在某些情况下非常有效。使用格式是：StepReg[b1,b2]，b1 和 b2 均为正整数，b1 代表初始参与拟合计算的数据行，b2 代表每次增加的数据行数，两者大小均应小于拟合数据的总行数。下面拟合实例显示，在运行代码中如果不加“StepReg[20,10];”一行，求解成功的概率不超过 10%，而加上之后，则能以 100%概率求得最优解。

例 1. 拟合公式：
$$y = \left( \frac{p_1 \cdot x^2 + p_2 \cdot x + p_3}{p_4} - \text{Trunc} \left( \frac{p_1 \cdot x^2 + p_2 \cdot x + p_3}{p_4} \right) \right) \cdot p_4 \quad (2-44)$$

表 2-27 StepReg 拟合数据

序号	x	y	序号	x	y	序号	x	y
1	0	0.00e+00	14	0.039	5.23e-03	27	0.078	1.21e-03
2	0.003	9.00e-05	15	0.042	7.67e-03	28	0.081	6.04e-03
3	0.006	3.60e-04	16	0.045	2.91e-04	29	0.084	1.06e-03
4	0.009	8.10e-04	17	0.048	3.09e-03	30	0.087	6.26e-03
5	0.012	1.44e-03	18	0.051	6.08e-03	31	0.09	1.66e-03
6	0.015	2.25e-03	19	0.054	9.25e-03	32	0.093	7.24e-03
7	0.018	3.24e-03	20	0.057	2.60e-03	33	0.096	3.01e-03
8	0.021	4.41e-03	21	0.06	6.13e-03	34	0.099	8.97e-03
9	0.024	5.76e-03	22	0.063	9.85e-03	35	0.102	5.12e-03
10	0.027	7.30e-03	23	0.066	3.75e-03	36	0.105	1.47e-03
11	0.03	9.01e-03	24	0.069	7.84e-03	37	0.108	8.00e-03
12	0.033	9.02e-04	25	0.072	2.11e-03	38	0.111	4.73e-03
13	0.036	2.98e-03	26	0.075	6.57e-03			

1stOpt 代码

StepReg[20,10];

Function y=((p1\*x^2+p2\*x+p3)/p4)-trunc(((p1\*x^2+p2\*x+p3)/p4))\*p4;

data;
0 0.00e+00
0.0039.00e-05
0.0063.60e-04
...

结果	
均方差(RMSE): 6.02539567057278E-5	参数 最佳估算
残差平方和(SSE): 1.37960493350437E-7	-----
相关系数(R): 0.999790850070571	p1 10.2795526425588
相关系数之平方(R^2): 0.999581743884835	p2 -0.0132630651577513
决定系数(DC): 0.999581743884835	p3 0.0705392697694426
卡方系数(Chi-Square): 0.000173518141499186	p4 0.010060405100386
F 统计(F-Statistic): 27096.6351056757	

## 2.2.11 QuickReg 与快速拟合

QuickReg 也是用于拟合的一个特殊命令，与 StepReg 不同，主要用于大样本且变化不是很剧烈的数据拟合，可明显加快拟合计算速度，但要注意并非适合于每一种情况。

基本格式为“QuickReg = 20”，QuickReg 的赋值大约取参与拟合数据量的 1/8 至 1/5。下面例子共有 141 组数据

表 2-28 QuickReg 拟合数据

序号	x	y	序号	x	y	序号	x	y
1	1428	0.0159	48	134286	76.3657	95	268571	101.5635
2	2857	0.0294	49	137143	76.8343	96	271429	125.4770
3	5714	0.0540	50	140000	76.4166	97	274286	123.0295
4	8571	0.0832	51	142857	73.9337	98	277143	116.8548
5	11429	0.1769	52	145714	83.6326	99	280000	132.5786
6	14286	0.3010	53	148571	79.5735	100	282857	146.0718
7	17143	0.4822	54	151429	77.3333	101	285714	152.3447
8	20000	0.8909	55	154286	82.7579	102	288571	149.5277
9	22857	1.4673	56	157143	74.8436	103	291429	149.2917
10	25714	2.6091	57	160000	70.2986	104	294286	167.8167
11	28571	3.9487	58	162857	69.1072	105	297143	140.3957
12	31429	6.1861	59	165714	62.9654	106	300000	146.2560
13	34286	10.2854	60	168571	63.9911	107	302857	140.2877
14	37143	14.1971	61	171429	63.3608	108	305714	150.3306
15	40000	20.8154	62	174286	54.3341	109	308571	151.0844
16	42857	27.9622	63	177143	54.7983	110	311429	135.0494
17	45714	34.1893	64	180000	54.1602	111	314286	124.1520
18	48571	50.9944	65	182857	50.5028	112	317143	128.2794
19	51429	52.8896	66	185714	46.6765	113	320000	121.8187
20	54286	74.7148	67	188571	37.3148	114	322857	127.7168
21	57143	78.2739	68	191429	35.4733	115	325714	109.3237
22	60000	94.8341	69	194286	31.4011	116	328571	98.8645
23	62857	97.1913	70	197143	31.6141	117	331429	101.0413
24	65714	106.9074	71	200000	29.3181	118	334286	95.4619
25	68571	110.5764	72	202857	29.2092	119	337143	82.9761
26	71429	99.7094	73	205714	25.6456	120	340000	108.2798
27	74286	95.2368	74	208571	23.5434	121	342857	173.0225
28	77143	100.9800	75	211429	21.6426	122	345714	125.9043
29	80000	95.3051	76	214286	20.5469	123	348571	55.3894
30	82857	84.7236	77	217143	21.0379	124	351429	48.4749
31	85714	72.5028	78	220000	23.0230	125	354286	40.5538

32	88571	63.7613	79	222857	23.3786	126	357143	32.5134
33	91429	53.8688	80	225714	22.6132	127	360000	30.7203
34	94286	45.3433	81	228571	28.5862	128	362857	26.0722
35	97143	47.6868	82	231429	29.2285	129	365714	21.6562
36	100000	39.2611	83	234286	34.6253	130	368571	19.8353
37	102857	41.2603	84	237143	40.5241	131	371429	16.4291
38	105714	38.4826	85	240000	43.5328	132	374286	14.3602
39	108571	41.0614	86	242857	44.4078	133	377143	11.6957
40	111429	42.8718	87	245714	55.3422	134	380000	8.8610
41	114286	52.4247	88	248571	63.8790	135	382857	7.2047
42	117143	54.6309	89	251429	70.3837	136	385714	6.0873
43	120000	58.5655	90	254286	64.7925	137	388571	5.1435
44	122857	57.6910	91	257143	84.1060	138	391429	4.1757
45	125714	62.2335	92	260000	85.2346	139	394286	3.0479
46	128571	63.6598	93	262857	96.3957	140	397143	2.7644
47	131429	74.5593	94	265714	101.5065	141	400000	2.1079

```
1stOpt 代码:
QuickReg = 20;
Function y = p1*exp(-2.77*((x-p2)/p3)^2)+p1*exp(-2.77*((x-5*p2)/p4)^2)+
          a0*exp(-0.5*((x-a1)/a2)^2)+2*a0*exp(-0.5*((x-2*a1)/a2)^2);
Data;
1428      0.0159
2857      0.0294
5714      0.0540
8571      0.0832
....
```

结果	
均方差(RMSE): 4.15952506651653	参数 最佳估算
残差平方和(SSE): 2439.53247783609	-----
相关系数(R): 0.995615366256638	p1 103.322600494197
相关系数之平方(R^2): 0.99124995752634	p2 68678.0879083805
决定系数(DC): 0.991249536510768	p3 36696.0095727035
卡方系数(Chi-Square): 13.388457439654	p4 5496.22604752625
F 统计(F-Statistic): 2552.24575941785	a0 76.4782210308997
	a1 148671.24116907
	a2 -35012.6480980637

拟合结果见图 2-33，采用“QuickReg”比不采用计算速度快了约 3 倍。

## 2.2.12 递推公式拟合

递推公式主要是因变量不仅与自变量有关，而且与因变量自身有关，也可认为是一种时间系列拟合，基本形式如下：

$$y_i = f(x_i, y_{i-1}) \tag{2-45}$$

例 1. 已知拟合公式： $y_i = a + b \cdot x_i^c + d \cdot y_{i-1}$  (2-46)

表 2-29 递推公式拟合数据

x	0.829,0.816,0.643,0.570,0.472,0.365,0.245,0.118,0.372,0.558
y	0.125,0.146,0.243,0.247,0.268,0.284,0.307,0.271,0.300,0.324

## 1stOpt 代码

```
Parameter a,b,c,d;
Variable x, y;
StartProgram [Pascal];
Procedure MainModel;
var i: integer;
    y0: double;
Begin
    y0 := 0.1;
    for i := 0 to DataLength - 1 do begin
        y[i] := a+b*x[i]^c+d*y0;
        y0 := y[i];
    end;
End;
EndProgram;
Data;
0.829,0.816,0.643,0.570,0.472,0.365,0.245,0.118,0.372,0.558;
0.125,0.146,0.243,0.247,0.268,0.284,0.307,0.271,0.300,0.324;
```

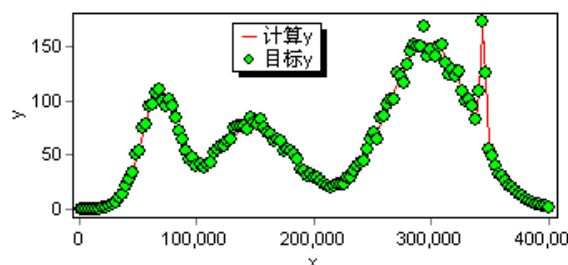


图 2-33 快速拟合效果图

## 结果

均方差(RMSE): 0.0148704873348888  
 残差平方和(SSE): 0.00221131393577087  
 相关系数(R): 0.971697391812805  
 相关系数之平方(R^2): 0.944195821255808  
 决定系数(DC): 0.944190448967862  
 F 统计(F-Statistic): 35.8361599943407

参数 最佳估算

参数	最佳估算
a	0.135164154284012
b	-3.42087548246511
c	20.9530572489513
d	0.552692143388021

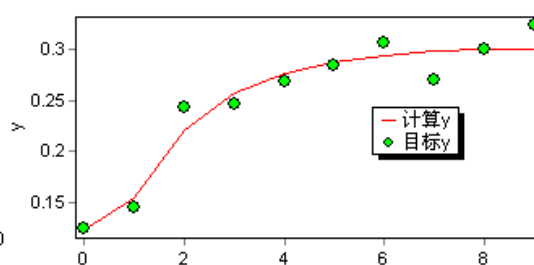


图 2-34 递推公式拟合计算结果

## 2.2.13 变系数拟合

变系数拟合一般指多文件数据拟合时，不同的文件数据对于有不同的参数或常数，对前者用“VarParameter”表示，对后者用“VarConstant”。在编程模式处理复杂工程问题时用的较多。下面举例说明。

$$\text{拟合公式: } y = p_1 - p_1 \cdot \exp(-p_2 \cdot x^{p_3}) + p_4 \cdot x^{p_5} + p_2 \quad (2-47)$$

P5 为一常数，不同 p5 下的几组数据如下。

表 2-30 变系数拟合数据

x	y			
	P5=-1	P5=-0.5	P5=0	P5=0.5
0.25	46.298	11.685	-7.287	-15.662
0.5	18.621	6.637	-2.140	-7.421
1	3.244	2.918	3.091	3.109
1.5	-0.529	2.263	6.091	10.275
2	-1.950	2.181	7.929	16.731
2.5	-2.331	2.252	8.741	18.366
3	-2.509	2.429	10.954	26.494
3.5	-2.264	2.310	11.590	30.990
4	-2.333	2.475	13.236	32.158
4.5	-2.333	2.592	11.895	31.943
5	-2.392	2.303	14.169	40.370
5.5	-2.213	2.654	13.692	36.195

6	-2.101	2.516	15.614	40.402
6.5	-1.759	2.748	14.483	48.785
7	-1.561	3.149	15.110	44.280
7.5	-1.530	2.887	15.129	50.702
8	-1.294	3.038	15.130	45.932
8.5	-1.164	3.104	16.964	58.362
9	-1.117	3.044	14.957	55.357

1stOpt 代码:

```
VarConstant p5=[-1,-0.5,0,0.5];
Function y= p1-p1*exp(-p2*x^p3))+p4*x^p5+p2;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
46.3,18.6,3.2,-0.5,-2.0,-2.3,-2.5,-2.3,-2.3,-2.4,-2.2,-2.1,-1.8,-1.6,-1.5,-1.3,-1.2,-1.1;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
11.7,6.6,2.9,2.3,2.2,2.3,2.4,2.3,2.5,2.6,2.3,2.7,2.5,2.7,3.1,2.9,3.0,3.1,3.0;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
-7.3,-2.1,3.1,6.1,7.9,8.7,11.0,11.6,13.2,11.9,14.2,13.7,15.6,14.5,15.1,15.1,15.1,17.0,15.0;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
-15.7,-7.4,3.1,10.3,16.7,18.4,26.5,31.0,32.2,31.9,40.4,36.2,40.4,48.8,44.3,50.7,45.9,58.4,55.4;
```

结果

均方差(RMSE): 1.46844242313118 残差平方和(SSE): 163.880559403904 相关系数(R): 0.996016782544658 相关系数之平方(R^2): 0.992049431110612 决定系数(DC): 0.992049411263255 F 统计(F-Statistic): 286.286358082532	参数	最佳估算
	p1	-51.3319085911549
	p2	0.616524036910651
	p3	-0.436464991756911
	p4	19.2688951414747

如果本例中不同文件所对应的 p5 值未知,此时需要将 p5 视为未知待求参数,其数目与要拟合的文件数据一致,参数数目变为 8 个。用关键字“VarParameter”进行定义,上面代码改为如下:

1stOpt 代码

```
VarParameter p5;
Function y= p1-p1*exp(-p2*x^p3))+p4*x^p5+p2;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
46.3,18.6,3.2,-0.5,-2.0,-2.3,-2.5,-2.3,-2.3,-2.4,-2.2,-2.1,-1.8,-1.6,-1.5,-1.3,-1.2,-1.1;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
11.7,6.6,2.9,2.3,2.2,2.3,2.4,2.3,2.5,2.6,2.3,2.7,2.5,2.7,3.1,2.9,3.0,3.1,3.0;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
-7.3,-2.1,3.1,6.1,7.9,8.7,11.0,11.6,13.2,11.9,14.2,13.7,15.6,14.5,15.1,15.1,15.1,17.0,15.0;
Data;
0.25,0.5,1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,6.5,7,7.5,8,8.5,9;
-15.7,-7.4,3.1,10.3,16.7,18.4,26.5,31.0,32.2,31.9,40.4,36.2,40.4,48.8,44.3,50.7,45.9,58.4,55.4;
```

结果

均方差(RMSE): 1.39242564092599 残差平方和(SSE): 147.35253657862 相关系数(R): 0.996419220636458 相关系数之平方(R^2): 0.992851263253767 决定系数(DC): 0.992851260564925	参数	最佳估算
	p1	-45.3794451220797
	p2	0.742955165657522
	p3	-0.42685291955804

F 统计(F-Statistic): 93.1203182769692	p4	19.3722687734958
	p5(数据文件 - 1)	-0.953248362800597
	p5(数据文件 - 2)	-0.479474719326709
	p5(数据文件 - 3)	0.0150096101253355
	p5(数据文件 - 4)	0.506538874630483

### 2.2.14 公式自动搜索拟合

在很多情况下，仅知道自变量和因变量的数据，而其模型公式却未知，这时该如何确定模型公式呢？1stOpt 提供的拟合公式自动搜索功能专门用于解决此类问题。

公式自动搜索主要用于在模型公式未知时，搜寻与数据匹配最好的公式，并按结果好坏排列给出公式列表。对于二维和三维数据，函数库分别包含了 3700 和 1000 余种不同类型的公式；而对于高于三维的拟合，公式库则必须用户自己添加。不论是二维、三维或高维，用户自行添加公式于库中时必须符合添加规则：

- 二维：x 表示自变量，y 表示因变量，p1、p2、p3、... 表示参数
- 三维：x、y 表示自变量，z 表示因变量，p1、p2、p3、... 表示参数
- 高于三维：x1、x2、x3、... 表示自变量，y 表示因变量，p1、p2、p3、... 表示参数
- 参数以字母 p 表示，下标从 1 开始，必须连续，如有三个参数，只能定义为 p1、p2、p3，而不能定义为 p1、p3、p4

自动搜索时有两种模式：“快速模式”和“稳健模式”，前者速度快但效果略差，后者相反，效果较好但速度下降。不论何种模式，搜索得出的公式都不能保证 100% 为最优解，因此最好选定公式后再用常规拟合方法进行验证。如果模型参数还有一定的物理意义限制，则还应进行相应的约束处理。

进行快速搜索时，仅需关键字“Data”即可，数据可竖排也可横排，横排时以“;”号结束。

#### 公式自动搜索例

表 2-31 公式自动搜索拟合数据

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
1stOpt 代码	
竖排格式	横排格式
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	Data; 300,780,1080,1320,1560,1860,2100,2520,3000,3600,4200,4800,5400,6000,6600,7200,11040; .270,.419,.480,.520,.557,.593,.616,.654,.682,.713,.736,.756,.768,.783,.797,.807,.843;



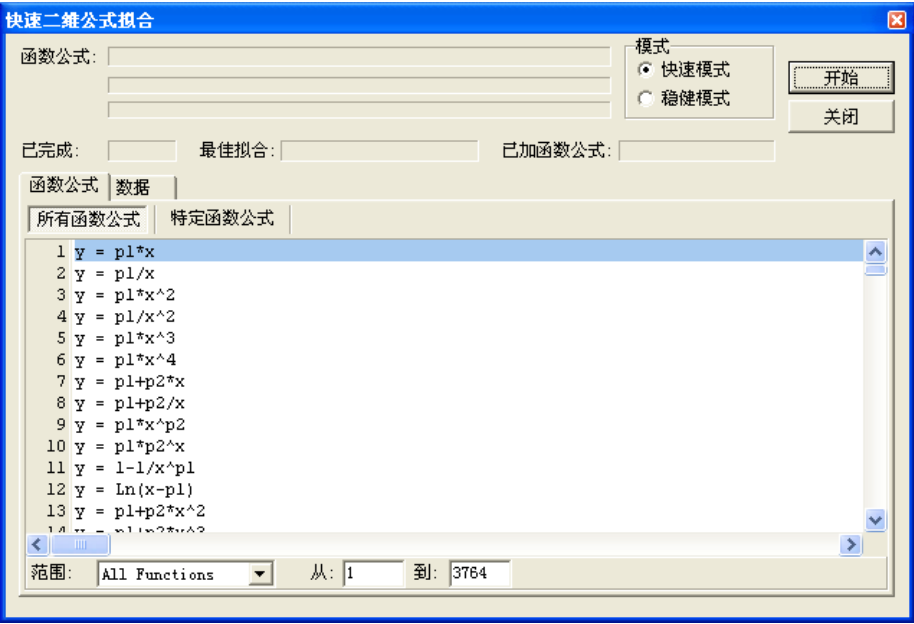


图 2-35 公式自动搜索画面

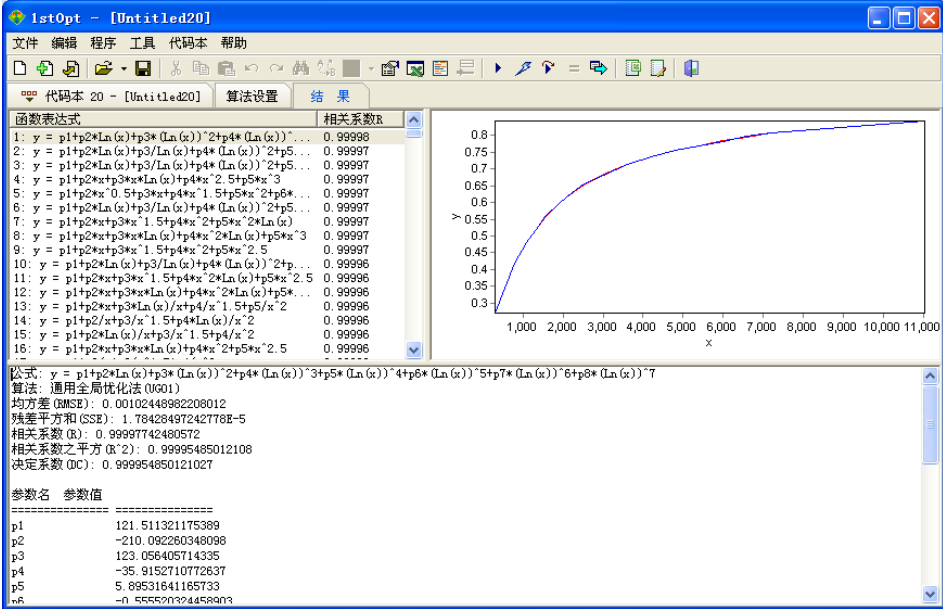


图 2-36 公式自动结果

## 2.2.15 设定拟合初始取值范围

一般情况下，拟合参数初值都是随机赋值的，但某些情况下将会出现异常情况，如下例

拟合公式：
$$y = 1 - \exp\left(\frac{-d \cdot (b + c \cdot x) \cdot x}{a + x}\right)$$
 (2-48)

表 2-32 设定初始范围的拟合数据

x	7.50e+21,1.10e+22,1.70e+22,3.40e+22,1.70e+23,3.40e+23,6.70e+23,8.40e+23,1.10e+24,1.70e+24,2.10e+24, 3.40e+24,4.20e+24,6.70e+24,8.40e+24
---	--

y	0.01,0.018,0.029,0.053,0.09,0.11,0.12,0.14,0.15,0.17,0.24,0.31,0.32,0.49,0.5
---	--

特点:

- x 数据值非常大，与 y 数据完全不在一个数量级；
- 从拟合公式看，c、b、d 三个参数有可能不是唯一的，也即是过参数拟合措施；
- 1stOpt 的参数缺省起始区间是【0， 5】，显然，在本例中无法满足要求，因此将参数 a 和 c 的起始区间分别设为【0， 1E+30】和【-1E-35， 1E-10】；
- 虽然对拟合问题，麦考特+全局通用算法为首选，但该例中差分算法（DE）却是最有效的。
- 1stOpt 代码如下：

```
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+21,1.10e+22,1.70e+22,3.40e+22,1.70e+23,3.40e+23,6.70e+23,8.40e+23,1.10e+24,1.70e+24,2.10e+24,3.40
e+24,4.20e+24,6.70e+24,8.40e+24;
0.01,0.018,0.029,0.053,0.09,0.11,0.12,0.14,0.15,0.17,0.24,0.31,0.32,0.49,0.5;
```

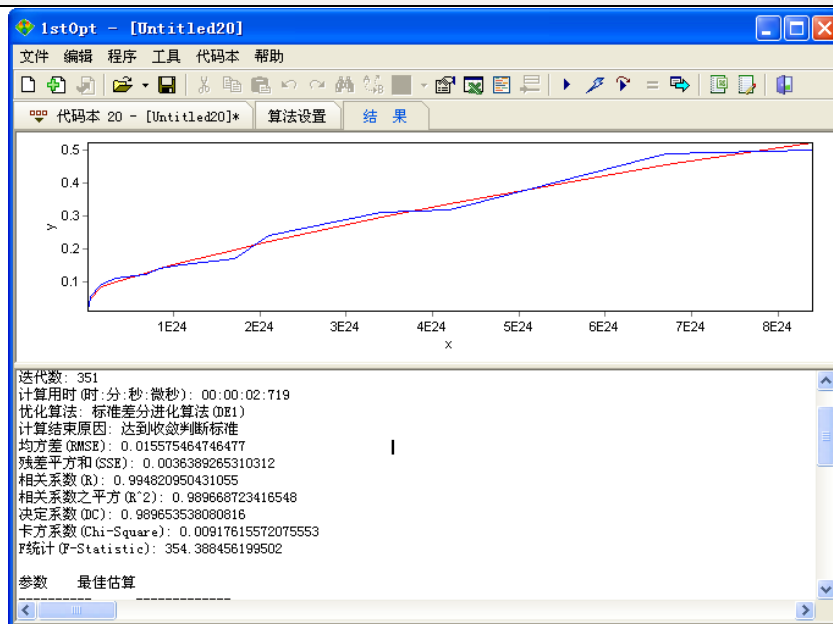


图 2-37 拟合结果画面

## 2.2.16 常微分方程拟合

常微分方程拟合是 4.0 后才有的功能，除了几个关键字不同外，与普通拟合几乎一样。几个常用的关键字：

- ODEFunction: 用于定义微分方程；
- InitialODEValue: 定义初值；
- ODEAlgorithm: 设定常微分方程算法
- ODERegStep: 定义步长；
- ODEi: 用于缺失数据的标示。

微分表示如下：

一阶微分  $\frac{dy}{dx}$  可写成  $y'$ ，高阶微分  $\frac{dy^2}{dx^2}$  可写成  $y''$ ，依次类推，不用人为降阶。

例 1. 一阶微分方程

拟合微分方程：
$$\frac{dy}{dx} = p_1 \cdot \exp(p_2 \cdot x) - \frac{p_3 \cdot x^{(p_4-x)}}{(x^{p_4} + p_5)^2}$$
 (2-49)

表 2-33 微分方程拟合数据一

x	.01,.05,.1,.15,.2,.25,.28,.3,.35,.4,.45,.5
y	.167,.721,1.33,1.676,1.854,1.921,1.932,1.937,1.934,1.935,1.935,1.935

1stOpt 代码如下	结果												
<pre>Variable x,y; ODEFunction y'=p1*exp(p2*x)-                 p3*x^(p4-x)/(x^p4+p5)^2; Data; .01 .167 .05 .721 .1 1.33 .15 1.676 .2 1.854 .25 1.921 .28 1.932 .3 1.937 .35 1.934 .4 1.935 .45 1.935 .5 1.935</pre>	<p>常微分方程算法：龙格 - 库塔 - 费尔博格法 (Runge-Kutta-Fehlberg Method) 优化算法：通用全局优化法(UGO1) 计算结束原因：达到收敛判定标准 计算用时(时:分:秒:微秒): 00:00:29:94 均方差(RMSE): 0.005020709979258 残差平方和(SSE): 0.00027728281565403 相关系数(R): 0.999906547397225 相关系数之平方(R^2): 0.999813103527838 决定系数(DC): 0.999813103402988 F 统计(F-Statistic): 8025.828313515</p> <table><tr><td>参数</td><td>最佳估算</td></tr><tr><td>p1</td><td>73.4912009714643</td></tr><tr><td>p2</td><td>-15.6279333083522</td></tr><tr><td>p3</td><td>1409949.54338989</td></tr><tr><td>p4</td><td>-2.43859004084862</td></tr><tr><td>p5</td><td>9128.6867360705</td></tr></table>	参数	最佳估算	p1	73.4912009714643	p2	-15.6279333083522	p3	1409949.54338989	p4	-2.43859004084862	p5	9128.6867360705
参数	最佳估算												
p1	73.4912009714643												
p2	-15.6279333083522												
p3	1409949.54338989												
p4	-2.43859004084862												
p5	9128.6867360705												

注意如果代码中没有使用“InitialODEValue”给出初值，数据第一行将被自动视为初值，如果缺失对应列数据，则该初值将被视为一待求参数。

微分方程拟合计算量大，缺省的龙格-库塔-费尔博格法（RKF45）求解精度高、适用面广，但相对于龙格-库塔法比较耗时，实际应用时根据具体情况可先用 4 阶龙格-库塔法试算，得到较为满意的结果后再用 RKF45 进一步计算。

例 2. 高阶微分方程组

已知微分方程：
$$\begin{cases} \frac{dx_1^2}{dt^2} = p_1 \cdot (x_2 - x_1) + \frac{dx_2}{dt} \cdot \frac{t \cdot \cos\left(\frac{dx_1}{dt} - x_2\right)}{p_2} \\ \frac{dx_2^2}{dt^2} = p_3 \cdot (x_1 - x_2) - \frac{dx_1}{dt} \cdot t - \frac{\sin\left(\frac{dx_2}{dt} + x_1\right)}{p_4} \end{cases}$$
 (2-50)

表 2-34 微分方程拟合数据二

t	0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,1.1
x1	-1,-0.447,0.149,0.767,1.387,1.984,2.539,3.030,3.441,3.761,3.981,4.092
$x_2' = \frac{dx_2}{dt}$	-2.9,-3.474,-3.916,-4.216,-4.357,-4.324,-4.110,-3.723,-3.184,-2.500,-1.658,-0.636

```

1stOpt 代码如下

Variable t,x1,x2';
ODEFunction x1''=p1*(x2-x1)+x2'*t*cos((x1'-x2))/p2;
              x2''=p3*(x1-x2)-x1'*t*sin((x2'+x1))/p4;

Data;
0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1,1.1;
-1,-0.447,0.149,0.767,1.387,1.984,2.539,3.030,3.441,3.761,3.981,4.092;
-2.9,-3.474,-3.916,-4.216,-4.357,-4.324,-4.110,-3.723,-3.184,-2.500,-1.658,-0.636;

```

结果	
常微分方程算法: 龙格-库塔-费尔博格法(Runge-Kutta-Fehlberg Method)	参数
优化算法: 通用全局优化法(UGO1)	最佳估算
计算结束原因: 用户中止	-----
计算用时(时:分:秒:微秒): 00:00:37:594	p1 2.00211675003467
均方差(RMSE): 0.0013286214119067	p2 3.83231003442367
残差平方和(SSE): 3.88351668358927E-5	p3 2.00354746973598
相关系数(R): 0.999999911222671	p4 2.87775739034647
相关系数之平方(R^2): 0.999999822445351	x1'初值 5.25341831094185
决定系数(DC): 0.999999818551957	x2 初值 2.00552563414669

结果中包括  $x_1'$  和  $x_2$  的初值估值。

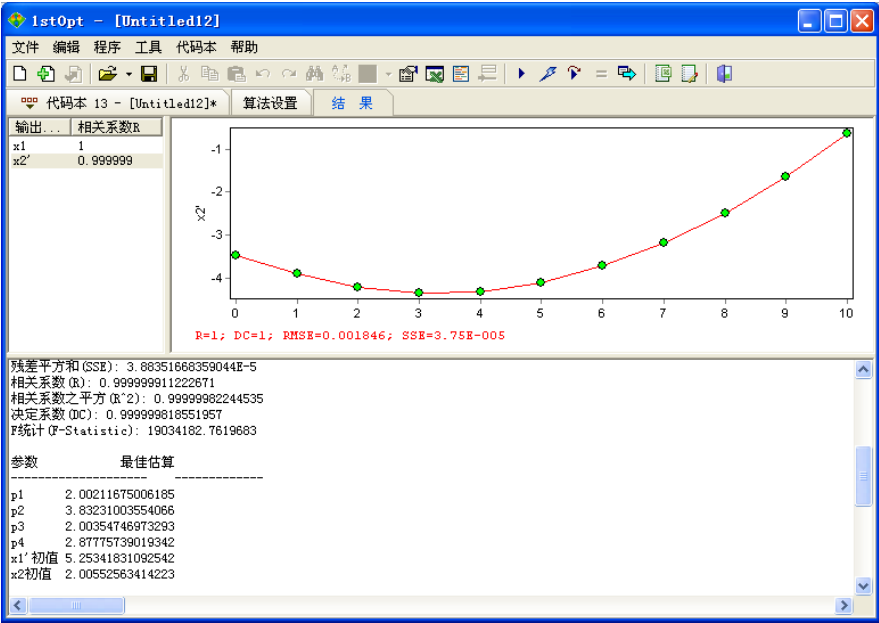


图 2-38 微分方程拟合计算结果一

### 例 3. 微分方程组拟合

$$\text{已知微分方程组: } \begin{cases} \frac{dy_1}{dx} = k_1 \cdot y_1 \\ \frac{dy_2}{dx} = k_1 \cdot y_1 - k_2 \cdot y_2 \\ \frac{dy_3}{dx} = k_2 \cdot y_2 - k_3 \cdot y_3 \\ \frac{dy_4}{dx} = k_1 \cdot y_1 + k_2 \cdot y_2 + k_3 \cdot y_3 \end{cases} \quad (2-51)$$

表 2-35 微分方程组拟合数据

x	0,3,28,53,78,103,183,303,463,603
y1	0.9922,0.9602,0.8352,0.5316,0.2798,0.1241,0.0175,0.0000,0.0000,0.0000
y4	0.0000,0.0169,0.0746,0.2409,0.4348,0.6058,0.7922,0.8666,0.8777,0.8799

数据缺少 y2 和 y3，但这并不影响正常计算。

1stOpt 代码如下

```
Parameter k1,k2,k3 ;
Variable x,y1,y4;
ODEFunction y1'=k1*y1;
          y2'=k1*y1-k2*y2;
          y3'=k2*y2-k3*y3;
          y4'=k1*y1+k2*y2+k3*y3;

data;
0,3,28,53,78,103,183,303,463,603;
0.9922,0.9602,0.8352,0.5316,0.2798,0.1241,0.0175,0.0000,0.0000,0.0000;
0.0000,0.0169,0.0746,0.2409,0.4348,0.6058,0.7922,0.8666,0.8777,0.8799;
```

结果

常微分方程算法: 龙格-库塔-费尔博格法(Runge-Kutta-Fehlberg Method)	参数	最佳估算
优化算法: 通用全局优化法(UGO1)	-----	-----
计算结束原因: 达到收敛判定标准	k1	-0.0141381652040465
计算用时(时:分:秒:微秒): 00:05:09:515	k2	0.03629115032839
均方差(RMSE): 0.0534684363911388	k3	0.0109004605968108
残差平方和(SSE): 0.0514597264220387	y2 初值	-2.22151838445331
相关系数(R): 0.995530888011698	y3 初值	8.29247046878423
相关系数之平方(R^2): 0.99108174898536		
决定系数(DC): 0.990709937258519		
F 统计(F-Statistic): 23.4120649778941		

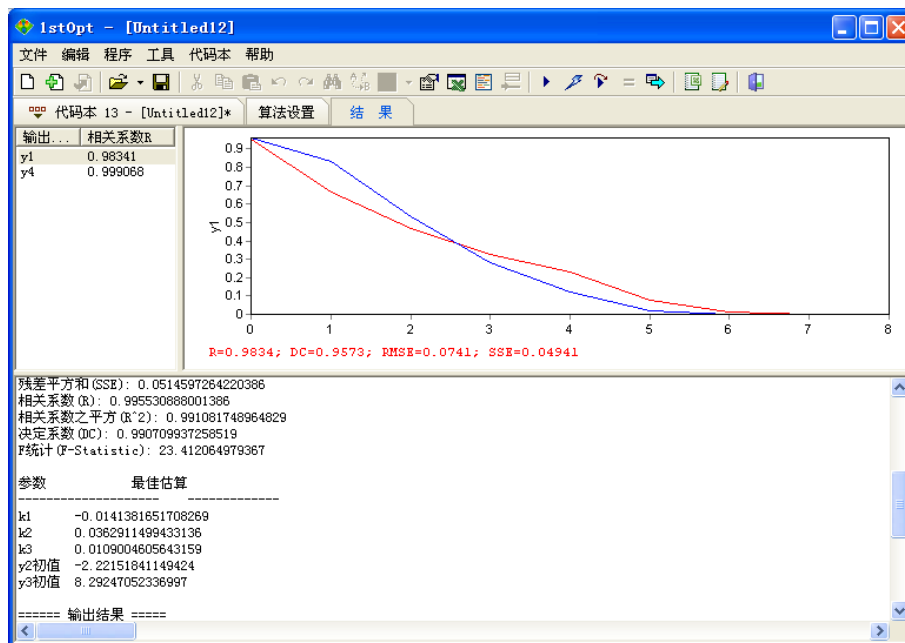


图 2-39 微分方程拟合计算结果二

#### 例 4. 多数据约束高阶微分方程

已知微分方程:  $\frac{dy^2}{dx^2} = p_1 \cdot \cos(x) \cdot x^{p_4} - p_2 \cdot x \cdot \sin(x - a + p_3)$  (2-52)

当 a 取 1 和 2 时, 分别获得了两组数据如下, 试根据实验数据求参数 p1 至 p4。对参数有如下约束:

$$\text{s. t. } \begin{cases} p_1 + p_2 \geq 2 \\ p_2 + p_3 \leq 3 \end{cases}$$

表 2-36 多数据约束高阶微分方程拟合数据

a=1			a=3		
x	y	$y' = \frac{dy}{dx}$	x	y	$y' = \frac{dy}{dx}$
0	1	-2	0	-1	2
0.2	0.58	-1.55	0.2	-0.59	1.96
0.4	0.33	-1.21	0.4	-0.19	2.11
0.6	0.16	-0.73	0.6	0.21	2.39
0.8	0.06	-0.42	0.8	0.71	2.59
1	0.00	-0.15	1	1.19	2.84
1.2	-0.02	0.00	1.2	1.82	3.52
1.4	-0.01	0.03	1.4	2.39	3.87
1.6	-0.01	-0.09	1.6	3.63	4.38
1.8	-0.05	-0.35	1.8	4.44	3.91
2	-0.15	-0.81	2	4.77	3.89
2.2	-0.40	-1.40	2.2	5.23	3.60
2.4	-0.65	-1.96	2.4	6.53	2.60
2.6	-1.24	-3.06	2.6	6.47	0.64
2.8	-1.79	-3.67	2.8	6.16	-1.92
3	-2.99	-4.34	3	5.45	-5.21

3.2	-4.01	-5.41			
3.4	-4.95	-6.97			
3.6	-6.74	-7.90			
3.8	-8.44	-8.09			
4	-9.16	-8.96			

```

1stOpt 代码如下

VarConstant a=[1,3];
Variable x, y, y';
Parameter p(4);
ODEFunction y''=p1*cos(x)*x^p4 - p2*x*sin(x-a+p3);
    p1+p2>=2;
    p2+p3<=3;

Data;
0,0.2,0.4,0.6,0.8,1,1.2,1.4,1.6,1.8,2,2.2,2.4,2.6,2.8,3,3.2,3.4,3.6,3.8,4;
1,0.58,0.33,0.16,0.06,0.00,-0.02,-0.01,-0.01,-0.05,-0.15,-0.40,-0.65,-1.24,-1.79,-2.99,-4.01,-4.95,-6.74,-8.44,-9.16;
-2,-1.55,-1.21,-0.73,-0.42,-0.15,0.00,0.03,-0.09,-0.35,-0.81,-1.40,-1.96,-3.06,-3.67,-4.34,-5.41,-6.97,-7.90,-8.09,-8.96;

Data;
0,0.2,0.4,0.6,0.8,1,1.2,1.4,1.6,1.8,2,2.2,2.4,2.6,2.8,3;
-1,-0.59,-0.19,0.21,0.71,1.19,1.82,2.39,3.63,4.44,4.77,5.23,6.53,6.47,6.16,5.45;
2,1.96,2.11,2.39,2.59,2.84,3.52,3.87,4.38,3.91,3.89,3.60,2.60,0.64,-1.92,-5.21;

```

结果	参数	最佳估算
常微分方程算法: 龙格-库塔-费尔博格法(Runge-Kutta-Fehlberg Method)		
优化算法: 通用全局优化法(UGO1)		
计算结束原因: 达到收敛判定标准	p1	3.593077906536
计算用时(时:分:秒:微秒): 00:00:22:875	p2	0.661506050558973
均方差(RMSE): 0.532524478121196	p3	-3.85810134176421
残差平方和(SSE): 19.8507623858777	p4	0.637602095698861
相关系数(R): 0.990318220968059		
相关系数之平方(R^2): 0.980730178781342		
决定系数(DC): 0.980109235297229		
F 统计(F-Statistic): 89.2407741159852		

例 5. 多数据微分方程组

已知微分方程组：

$$\begin{cases} \frac{dy_1}{dx} = p_1 \cdot y_2 \\ \frac{dy_2}{dx} = y_3 \\ \frac{dy_3}{dx} = p_2 \cdot y_4 \\ \frac{dy_4}{dx} = y_5 \\ \frac{dy_5}{dx} = \frac{p_3 \cdot y_3 \cdot y_4 \cdot y_5 - p_4 \cdot y_4^3}{y_3^2} \end{cases} \quad (2-53)$$

两组不同的初值对应两组不同的数据，用 “InitialODEValue” 来定义两组初值。

表 2-37 多数据微分方程拟合数据一 (初值: x=0, y1=0. 1, y2=1, y3=0. 5, y4=2. 5, y5=0. 6)

x	y1	y2	y3	y4	y5
0.136	0.242	1.087	0.715	-0.268	-34.688
0.273	0.396	1.171	0.462	-2.465	3.788
0.409	0.559	1.214	0.198	-1.329	8.184

0.545	0.726	1.232	0.078	-0.539	3.701
0.682	0.895	1.238	0.030	-0.210	1.460
0.818	1.064	1.241	0.012	-0.081	0.565
0.955	1.233	1.242	0.004	-0.031	0.218
1.091	1.403	1.242	0.002	-0.012	0.084
1.227	1.572	1.243	0.001	-0.005	0.032
1.364	1.741	1.243	0.000	-0.002	0.013
1.500	1.911	1.243	0.000	-0.001	0.005

表 2-38 多数据微分方程拟合数据二(初值:  $x=0$ ,  $y_1=0.1$ ,  $y_2=1$ ,  $y_3=0.5$ ,  $y_4=2.5$ ,  $y_5=0.6$ )

x	y1	y2	y3	y4	y5
0.136	2.097	0.923	3.205	1.501	-0.089
0.273	2.253	1.374	3.408	1.474	-0.316
0.409	2.473	1.852	3.605	1.414	-0.574
0.545	2.759	2.357	3.792	1.316	-0.856
0.682	3.117	2.886	3.963	1.180	-1.147
0.818	3.547	3.437	4.112	1.004	-1.429
0.955	4.055	4.006	4.235	0.792	-1.680
1.091	4.641	4.590	4.326	0.548	-1.880
1.227	5.307	5.184	4.383	0.282	-2.009
1.364	6.055	5.784	4.403	0.004	-2.055
1.500	6.884	6.383	4.384	-0.274	-2.012

1stOpt 代码如下

```

Parameter p(4);
InitialODEValue x=[0,0], y1=[0.1,2], y2=[1,0.5], y3=[0.5,3], y4=[2.5,1.5], y5=[0.6,0.1];
Variable x, y1,y2, y3, y4, y5;
ODEFunction y1'=p1*y2;
           y2'=y3;
           y3'=p2*y4;
           y4'=y5;
           y5'=(p3*y3*y4*y5-p4*y4^3)/y3^2;

Data;
0.1360.2421.0870.715-0.268    -34.688
0.2730.3961.1710.462-2.465    3.788
0.4090.5591.2140.198-1.329    8.184
0.5450.7261.2320.078-0.539    3.701
0.6820.8951.2380.030-0.210    1.460
0.8181.0641.2410.012-0.081    0.565
0.9551.2331.2420.004-0.031    0.218
1.0911.4031.2420.002-0.012    0.084
1.2271.5721.2430.001-0.005    0.032
1.3641.7411.2430.000-0.002    0.013
1.5001.9111.2430.000-0.001    0.005

Data;
0.1362.0970.9233.2051.501-0.089
0.2732.2531.3743.4081.474-0.316
0.4092.4731.8523.6051.414-0.574
0.5452.7592.3573.7921.316-0.856
0.6823.1172.8863.9631.180-1.147
0.8183.5473.4374.1121.004-1.429
0.9554.0554.0064.2350.792-1.680
1.0914.6414.5904.3260.548-1.880
1.2275.3075.1844.3830.282-2.009
1.3646.0555.7844.4030.004-2.055
1.5006.8846.3834.384-0.274    -2.012

```

结果



均方差(RMSE): 0.00257571999820954	参数	最佳估算
残差平方和(SSE): 0.000729776686009421		
相关系数(R): 0.999999878835271	p1	1.00008656860854
相关系数之平方(R^2): 0.999999757670557	p2	0.998024737425075
决定系数(DC): 0.999999821341232	p3	4.9919369240034
F 统计(F-Statistic): 7035419.25334065	p4	3.99541080935836

### 例 6. 缺失数据的标示及拟合

实验数据有不规则的缺失，如下数据表 2-39，“\*\*\*\*”表示该数据缺失，这种情况下该如何对下列微分方程组进行拟合呢？这里要用到关键字“ODEi”，该关键字专门用于标示数据，“1”代表有对应的数据，“0”表示无对应数据。重新整理后的数据见表 2-40，缺失数据“\*\*\*\*”可用任意有效数代表，此例中用“0”表示。

$$\begin{cases} \frac{dx_1}{dt} = x_3 \\ \frac{dx_2}{dt} = x_4 \\ \frac{dx_3}{dt} = (1-k_1) \cdot \sin(k_2 \cdot t + k_4) - 2 \cdot k_3 \cdot x_3 + 2 \cdot k_3 \cdot x_4 - x_1 + x_2 \\ \frac{dx_4}{dt} = \frac{k_1 \cdot \sin(k_2 \cdot t + k_4) + 2 \cdot k_3 \cdot x_3 - 2 \cdot k_3 \cdot (1+k_5) \cdot x_4 - (1+k_6) \cdot x_2}{k_7} \end{cases} \quad (2-54)$$

表 2-39 缺失拟合数据

t	x1	x2	x3	x4
0	0.005	-3	0	0
1	-1.445	-2.045	-1.914	1.782
2	-2.994	****	-0.346	1.273
3	****	0.550	1.784	****
4	0.974	0.576	****	-0.291
5	3.164	****	0.789	0.531
6	****	1.347	-0.862	****
7	1.346	1.951	****	0.676
8	0.393	****	-0.011	-0.903
9	****	0.808	0.043	****
10	0.687	-0.570	0.065	-1.363

表 2-40 缺失数据整理表

t	x1	x2	x3	x4	x1(ODEi)	x2(ODEi)	x3(ODEi)	x4(ODEi)
0	0.005	-3	0	0	1	1	1	1
1	-1.445	-2.045	-1.914	1.782	1	1	1	1
2	-2.994	0	-0.346	1.273	1	0	1	1
3	0	0.550	1.784	0	0	1	1	0
4	0.974	0.576	0	-0.291	1	1	0	1
5	3.164	0	0.789	0.531	1	0	1	1
6	0	1.347	-0.862	0	0	1	1	0
7	1.346	1.951	0	0.676	1	1	0	1
8	0.393	0	-0.011	-0.903	1	0	1	1
9	0	0.808	0.043	0	0	1	1	0
10	0.687	-0.570	0.065	-1.363	1	1	1	1

语句“Variable t,x(4), x(4)[ODEi];”等同于

“Variable t,x1,x2,x3,x4,x1[ODEi],x2[ODEi],x3[ODEi],x4[ODEi];”

1stOpt 代码如下

```
Variable t,x(4), x(4)[ODEi];
ODEFunction x1'=x3;
          x2'=x4;
          x3'=(1-k1)*sin(k2*t+k4)-2*k3*x3+2*k3*x4-x1+x2;
          x4'=(k1*sin(k2*t+k4)+2*k3*x3-2*k3*(1+k5)*x4+x1-(1+k6)*x2)/k7;

Data;
0  0.005-3.000    0.0000.0001    1    1    1
1  -1.445    -2.045    -1.914    1.7821    1    1    1
2  -2.994    0.000-0.346    1.2731    0    1    1
3  0.0000.5501.7840.0000    1    1    0
4  0.9740.5760.000-0.291    1    1    0    1
5  3.1640.0000.7890.5311    0    1    1
6  0.0001.347-0.862    0.0000    1    1    0
7  1.3461.9510.0000.6761    1    0    1
8  0.3930.000-0.011    -0.903    1    0    1    1
9  0.0000.8080.0430.0000    1    1    0
10 0.687-0.570    0.065-1.363    1    1    1    1
```

结果	
计算用时(时:分:秒:微秒): 00:00:41:359	参数 最佳估算
均方差(RMSE): 0.0413730788707377	-----
残差平方和(SSE): 0.198560872008337	k1 1.96525670623102
相关系数(R): 0.998086118654787	k2 2.9930200395685
相关系数之平方(R^2): 0.996175900251378	k4 1.15908907630355
决定系数(DC): 0.99614428909135	k3 0.0500003543637648
F 统计(F-Statistic): 0.799404156425403	k5 1.26394944649003
	k6 0.99814132412302
	k7 2.98481139101963

例 7. 等步长拟合

当微分数据步长不一致且间隔较大时，可用关键字“ODERegStep”使计算按给定的步长进行，对提高拟合精度有很好的帮助。在某些情况下，如微分方程算法使用一至四阶“龙格-库塔法”时，步长过大对拟合精度有很大影响。

拟合微分方程组：

$$\begin{cases} \frac{dx}{dt} = y - k_1 \cdot \cos(t + k_2) - k_3 \cdot y \\ \frac{dy}{dt} = k_4 \cdot x + y + k_5 \cdot t \end{cases} \quad (2-55)$$

表 2-41 等步长拟合数据（初值 t=0, x=1, y=1）

t	x	y
0.3	0.675	2.065
2.1	-0.623	9.552
3	8.539	29.316

1stOpt 代码如下

```
ODERegStep = 0.3;
ODEAlgorithm = RK4;
InitialODEValue t=0,x=1,y=1;
Variable t, x, y;
ODEFunction x'=y-k1*cos(t+k2)-y*k3;
          y'=k4*x+y+k5*t;

Data;
0.3  0.6752.065
```

2.1	-0.623	9.552
3	8.539	29.316

结果	
使用“ODERegStep”	不使用“ODERegStep”
常微分方程算法: 四阶龙格-库塔法(Fourth Order Runge-Kutta Method) 优化算法: 通用全局优化法(UGO1) 计算结束原因: 用户中止 计算用时(时:分:秒:微秒): 00:00:15:703 均方差(RMSE): 0.0214386989947079 残差平方和(SSE): 0.00551541377502829 相关系数(R): 0.999998984095299 相关系数之平方(R^2): 0.999997968191629 决定系数(DC): 0.999998061673587 F 统计(F-Statistic): 5.12122320427093	常微分方程算法: 四阶龙格-库塔法(Fourth Order Runge-Kutta Method) 优化算法: 通用全局优化法(UGO1) 计算结束原因: 用户中止 计算用时(时:分:秒:微秒): 00:00:15:922 均方差(RMSE): 0.0111158581235333 残差平方和(SSE): 0.000741373810935133 相关系数(R): 0.999999639704033 相关系数之平方(R^2): 0.999999279408196 决定系数(DC): 0.999999279303729 F 统计(F-Statistic): -36447.5765681835
参数 ----- 最佳估算	参数 ----- 最佳估算
k1 5.50432128217711 k2 -1.3552990742272 k3 0.267620859873006 k4 2.22570187589488 k5 -0.131203327608557	k1 5.58686555773678 k2 -1.33060029053298 k3 0.263465825271026 k4 2.31887169889776 k5 -0.0580247710613677

使用“ODERegStep”所得数据			
目标 x	计算 x	目标 y	计算 y
0.675	0.74389490394211	2.065	2.04247917833449
0	0.299319814670909	0	3.16098820012847
0	-0.236383467347927	0	4.27029976470085
0	-0.749275891831205	0	5.3371702668181
0	-1.10487986330609	0	6.41258526353131
0	-1.1372311640575	0	7.6806330650717
-0.623	-0.624838564462221	9.552	9.53732771035425
0	0.756585991279738	0	12.7234492788511
0	3.52186044747193	0	18.5541407434378
8.539	8.53423863014645	29.316	29.3205156983214

不使用“ODERegStep”所得数据			
目标 x	计算 x	目标 y	计算 y
0.675	0.700897895060419	2.065	2.05867690390189
-0.623	-0.624184110329092	9.552	9.54702059567901
8.539	8.53750088955258	29.316	29.3174991282418

# 例 8. 复合数据的拟合

微分方程拟合时, 变量声明关键字“Variable”可定义复合型数据, 如:  
Variable t, x1+x2, x2;

$$\text{拟合微分方程组: } \begin{cases} \frac{dx_1}{dt} = k_1 \cdot x_1 + k_2 \cdot x_2 + k_5 \\ \frac{dx_2}{dt} = k_3 \cdot x_1 + k_4 \cdot x_2 + k_5 \end{cases} \quad (2-56)$$

表 2-42 复合拟合数据 (初值 t=0, x1=0.01, x2=-1)

t	$x_1 + \exp(x_2)$	$x_2' = \frac{dx_2}{dt}$
0.1	-0.401	0.747
0.2	-1.036	0.129
0.3	-1.641	-0.340
0.4	-2.094	-0.784
0.5	-2.658	-1.075
0.6	-3.224	-1.541
0.7	-3.817	-1.962
0.8	-4.536	-2.226
0.9	-6.048	-2.994
1	-7.070	-3.325

1stOpt 代码如下

```
InitialODEValue t=0,x1=0.01,x2=-1;
Variable t,x1+exp(x2),x2';
ODEFunction x1'=k1*x1+k2*x2+k5;
              x2'=k3*x1+k4*x2+k5;
Data;
0.1 -0.401    0.747
0.2 -1.036    0.129
0.3 -1.641   -0.340
0.4 -2.094   -0.784
0.5 -2.658   -1.075
0.6 -3.224   -1.541
0.7 -3.817   -1.962
0.8 -4.536   -2.226
0.9 -6.048   -2.994
1   -7.070   -3.325
```

结果

均方差(RMSE): 0.124311451756304  
 残差平方和(SSE): 0.309066740755197  
 相关系数(R): 0.999147470801841  
 相关系数之平方(R^2): 0.998295668409715  
 决定系数(DC): 0.998293753008998  
 F 统计(F-Statistic): 212.012531077907

参数	最佳估算
k1	-0.977487871515061
k2	9.15933001093156
k5	0.417624873683557
k3	0.908685664659853
k4	-1.22762153799123

## 2.2.17 复数拟合

复数拟合一般是指拟合公式是复数形式，有两种处理方法，一是将复数公式分解成两个独立的实部和虚部公式，再按常规方式拟合即可，其优点是可利用实数计算的所有数学函数，但缺点是公式分解有时比较困难甚至不可能，即使借助一些专门的数学软件；二是直接对复数公式进行拟合，无需分解成虚实两部分，但缺点是支持复数计算的数学函数远不如实数的多。1stOpt4.0 起支持复数类型公式直接拟合。几个关键字：

- **ComplexStr:** 定义虚数符号
- **ComplexPar:** 定义复数型参数
- **realPart:** 指定变量实部数据
- **imagPart:** 指定变量虚部数据

$$\text{例 1. 拟合公式: } y = \frac{a}{1+(b \cdot x)^2} - \frac{a+c}{1+(b \cdot x)^2} i \quad (2-57)$$

其中 y 为复数因变量，x 实数自变量，i 虚数符号，a、b、c：实数参数。

表 2-43 复数拟合数据一

x	0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.11
---	---

y 实部	3,2.885,2.586,2.206,1.829,1.5,1.230,1.014,0.843,0.708,0.6,0.5137
y 虚部	-0.789,-0.660,-0.648,-0.511,-0.440,-0.418,-0.338,-0.279,-0.237,-0.218,-0.177,-0.136

✧ 分解后拟合

复数公式可很容易分解为实虚两部：

$$\text{实部: } y_1 = \frac{a}{1+(b \cdot x)^2} \quad (2-58)$$

$$\text{虚部: } y_2 = -\frac{a+c}{1+(b \cdot x)^2} \quad (2-59)$$

✧ 直接拟合

“realPart” “ImagPart” 用以指定实部和虚部变量，“ComplexStr”定义虚数符号，“ComplexPar”定义复数型参数，因为都是实数型参数，此例中不使用该关键字。

1stOpt 分解拟合代码

```
Variable x,y1,y2;
SharedModel;
Function y1 = a/(1+(b*x)^2);
          y2 = -(a*c)/(1+(n*x)^2);
Data;
0   3   -0.789
0.01 2.88462 -0.660
0.02 2.58621 -0.648
0.03 2.20588 -0.511
0.04 1.82927 -0.440
0.05 1.5   -0.418
0.06 1.22951 -0.338
0.07 1.01351 -0.279
0.08 0.8427  -0.237
0.09 0.70755 -0.218
0.1  0.6   -0.177
0.11 0.5137  -0.136
```

1stOpt 直接拟合代码

```
ComplexStr = i;
Variable x,y[realPart],y[imagPart];
Function y= A/(1+(x*B)^2) - i*(A*C)/(1+(x*B)^2);
Data;
0   3   -0.789
0.01 2.88462 -0.660
0.02 2.58621 -0.648
0.03 2.20588 -0.511
0.04 1.82927 -0.440
0.05 1.5   -0.418
0.06 1.22951 -0.338
0.07 1.01351 -0.279
0.08 0.8427  -0.237
0.09 0.70755 -0.218
0.1  0.6   -0.177
0.11 0.5137  -0.136
```

上面两段代码可求得一样结果

均方差(RMSE): 0.0238158527186631	参数	最佳估算
残差平方和(SSE): 0.0136126761772093	-----	
相关系数(R): 0.999835177171414	a	2.99420404711415
相关系数之平方(R^2): 0.999670381509393	b	19.9055986193202
决定系数(DC): 0.999670381509393	c	0.250607997082915
F 统计(F-Statistic): 229580.784019992		

$$\text{例 2. 拟合公式: } y = \frac{a \cdot i \cdot \tan(b \cdot x - e \cdot i)}{c + d \cdot i \cdot \tan(b \cdot x - e \cdot i)} \quad (2-60)$$

y 为复数且仅有实部数据，a、b、c 为实数参数，d、e 为复数参数，i 为虚数符号；参数范围[-1, 1]

表 2-44 复数拟合数据二

序号	x	y 实部	序号	x	y 实部	序号	x	y 实部
1	50	0.159	11	100	0.517	21	150	0.597
2	55	0.322	12	105	0.173	22	155	0.382

3	60	0.525	13	110	0.361	23	160	0.08
4	65	0.65	14	115	0.562	24	165	0.335
5	70	0.627	15	120	0.671	25	170	0.497
6	75	0.483	16	125	0.596	26	175	0.502
7	80	0.039	17	130	0.261	27	180	0.433
8	85	0.452	18	135	0.238	28	185	0.262
9	90	0.629	19	140	0.491	29	190	0.332
10	95	0.606	20	145	0.604	30	195	0.515

```
1stOpt 代码
Variable x, y[realPart];
Parameter a,b,c;
ComplexPar d,e;
ParameterDomain = [-1,1];
ComplexStr = i;
Function y=(a*i*tan(b*x-e*i))/(c+d*i*tan(b*x-e*i));
Data;
50,55,60,65,70,75,80,85,90,95,100,105,110,115,120,125,130,135,140,145,150,155,160,165,170,175,180,185,190,
195;
0.159,0.322,0.525,0.65,0.627,0.483,0.039,0.452,0.629,0.606,0.517,0.173,0.361,0.562,0.671,0.596,0.261,0.238,0.4
91,0.604,0.597,0.382,0.08,0.335,0.497,0.502,0.433,0.262,0.332,0.515;
```

有多解，其中一组结果如下

均方差(RMSE): 0.0530444337955333 残差平方和(SSE): 0.0844113587006615 相关系数(R): 0.952244453354437 相关系数之平方(R^2): 0.90676949894429 决定系数(DC): 0.90676949894429 F 统计(F-Statistic): 41.1167299320072	参数                      最佳估算 ----- d.realpart -0.0429872483458212 d.imagpart 0.00278970209813672 e.realpart -0.0773380514610092 e.imagpart -0.22364338475115 a -0.0266398114223544 b -0.11647416267366 c 0.0174007240861673
--	---

例 3. 拟合公式：
$$y = p_1 \cdot x^{(p_2 + p_5 \cdot xi)} + p_3 \cdot x^{p_4 \cdot i} \tag{2-61}$$

其中：y 为复数型数据，x 为实数型数据，参数均为实数，i 为虚数符号

表 2-45 复数拟合数据三

x	41.976,303.16,328.6,349.8,367.82,418.7,473.82,508.8,543.78,573.46,642.36,700.66,745.18,801.36,833.16,855.42,874.5,907.36
y 实部	222.93,138.84,224.46,297.57,285.91,334.81,409.65,470.78,525.78,573.44,602.50,621.38,677.63,752.38,833.78,898.98,932.05,970.90
y 虚部	14.553,-48.33,-13.831,24.595,17.977,46.885,96.638,141.8,185.64,225.93,251.51,268.52,321,394.75,479.99,551.79,589.37,634.47

```
1stOpt 代码
Variable x,y[realPart],y[imagPart];
ComplexStr = i;
Function y=p1*x^(p2+p5*x*i)+p3*x^(p4*i);
Data;
41.976,303.16,328.6,349.8,367.82,418.7,473.82,508.8,543.78,573.46,642.36,700.66,745.18,801.36,833.16,855.42,
874.5,907.36;
222.93,138.84,224.46,297.57,285.91,334.81,409.65,470.78,525.78,573.44,602.50,621.38,677.63,752.38,833.78,89
8.98,932.05,970.90;
-14.553,-48.33,-13.831,24.595,17.977,46.885,96.638,141.8,185.64,225.93,251.51,268.52,321,394.75,479.99,551.7
9,589.37,634.47;
```

结果如下

均方差(RMSE): 38.1516946285731	参数	最佳估算
残差平方和(SSE): 52399.8649091482		
相关系数(R): 0.996832798253507	p1	0.408299964413126
相关系数之平方(R^2): 0.993675627673917	p2	1.17127614375748
决定系数(DC): 0.993675627673917	p5	7.58087930817352E-5
F 统计(F-Statistic): 122.97550081656	p3	-143.587067352712
	p4	-0.986604476741416

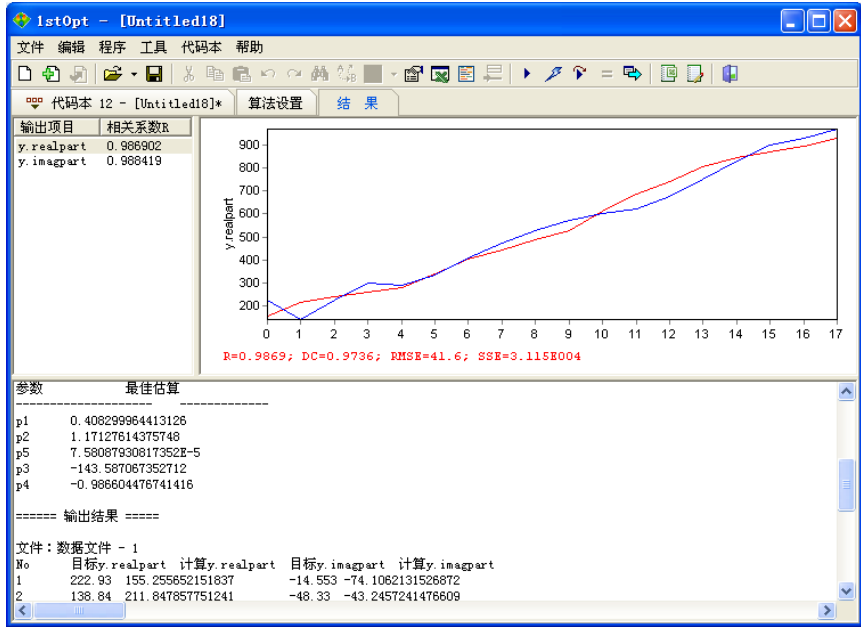


图 2-40 复数拟合结果对比一

例 4. 拟合公式：
$$y = \frac{p_1 \cdot x^{p_2}}{p_3^{x^{p_4}}} \quad (2-62)$$

其中：x、y 均为复数型数据，参数均为复数。

表 2-46 复数拟合数据四

x 实部	200,400,1600,3200,6400,12800,25600,51200,102400
X 虚部	279,270,255,230,202,170,150,125,100
y 实部	105.60,65.13,76.18,100.76,114.51,117.56,114.84,108.54,90.43
y 虚部	519.43,458.25,254.00,149.45,110.53,80.29,40.82,8.46,-15.41

1stOpt 代码如下：

```

Variable x[realPart],x[imagPart],y[realPart],y[imagPart];
ComplexPar p(4);
Function y = p1*x^p2/p3^(x^p4);
Data;
//x 实部,x 虚部,y 实部,y 虚部
200 279 105.60 519.43
400 270 65.13 458.25
1600 255 76.18 254.00
3200 230 100.76 149.45
6400 202 114.51 110.53
12800 170 117.56 80.29
25600 150 114.84 40.82
51200 125 108.54 8.46
102400 100 90.43 -15.41

```

绝大部分优化算法或其它软件对本题都难以获得稳定最优解。

结果如下：

均方差(RMSE): 6.29415476974529	参数	最佳估算
残差平方和(SSE): 713.094916779133		
相关系数(R): 0.99947453817744		
相关系数之平方(R^2): 0.998949352465007		
决定系数(DC): 0.998949352465007		
F统计(F-Statistic): 4.04786461338704		
	p1.realpart	-1345.23722291759
	p1.imagpart	-532.874722012776
	p2.realpart	-0.243007055415745
	p2.imagpart	-0.314215108612732
	p3.realpart	6.08007872575963E-7
	p3.imagpart	-1.86509528329051E-7
	p4.realpart	-0.612407492032328
	p4.imagpart	1.06627199949592

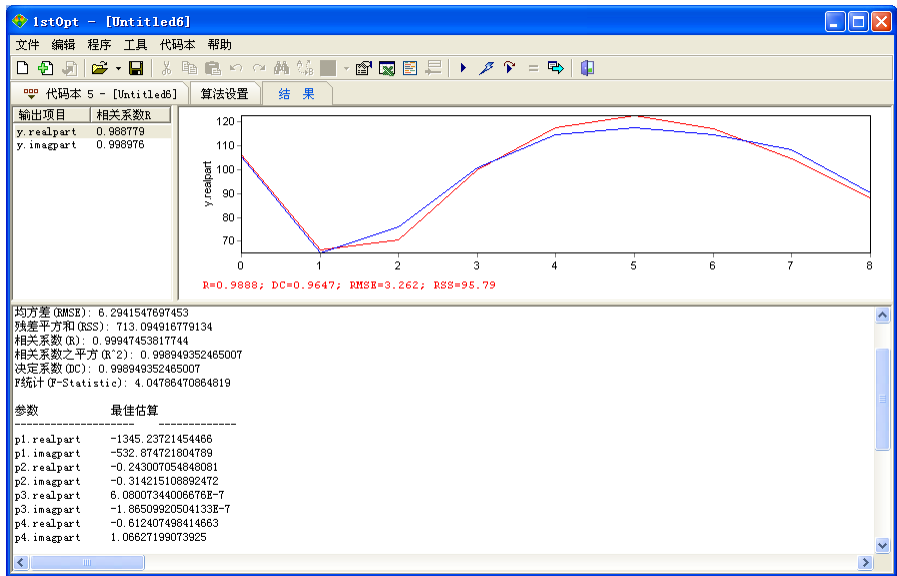


图 2-41 复数拟合结果对比一

## 2.2.18 非线性问题线性化拟合

非线性拟合，因为涉及到初值的猜测拟合难度大，因而在可能的情况下将非线性公式进行一定的数学变换使其成为线性公式后再进行拟合，被视为一种有效可行并广而用之的方法，其优点是对线性问题无需迭代可直接计算出正确结果。但这种方法所得结果精度如何呢？线性化后所得结果的确是最优，但仅对线性化后的问题，而逆推回原非线性方程，其结果往往并不是最优的，有时误差还很大。下面以实例说明。

例 1. 二维拟合公式： $y = a \cdot x^b$  (2-63)

其中 a、b 为待求系数。

表 2-47 线性化拟合数据一

x	0.091,0.2543,0.3121,0.3792,0.4754,0.4410,0.4517,0.5595,0.8080
y	0.7171,0.8964,1.0202,1.1962,1.4928,1.6909,1.8548,2.1618,2.6638
y1=ln(y)	-2.3969,-1.3692,-1.1644,-0.9697,-0.7436,-0.8187,-0.7947,-0.5807,-0.2132
x1=ln(x)	-0.3325,-0.1094,0.0200,0.1791,0.4007,0.5253,0.6178,0.7709,0.9798

✧ 直接非线性拟合

✧ 公式线性化处理： $\ln(y) = \ln(a) + b \cdot \ln(x)$ ，令  $y1 = \ln(y)$ ， $a1 = \ln(a)$ ， $x1 = \ln(x)$ ，有  $y1 = a1 + b \cdot x1$



1stOpt 代码如下

直接非线性拟合	线性化后拟合
Variable x,y; Function y=a*x^b; Data; 0.0910.7171 0.2543 0.8964 0.3121 1.0202 0.3792 1.1962 0.4754 1.4928 0.4410 1.6909 0.4517 1.8548 0.5595 2.1618 0.8080 2.6638	PassParameter a=exp(a1); Variable x1,y1; Function y1=a1+b*x1; Data; -2.3969 -0.3325 -1.3692 -0.1094 -1.1644 0.0200 -0.9697 0.1791 -0.7436 0.4007 -0.8187 0.5253 -0.7947 0.6178 -0.5807 0.7709 -0.2132 0.9798

结果如下（图 2-42）

直接非线性拟合	线性化后拟合
均方差(RMSE): 0.188131839723942 残差平方和(SSE): 0.318542302061233 相关系数(R): 0.951138940848744 相关系数之平方(R^2): 0.90466528479887 决定系数(DC): 0.901998244902145 卡方系数(Chi-Square): 0.157952574904411 F 统计(F-Statistic): 71.4272922256392  参数 最佳估算 ----- a 3.23204821806461 b 0.859500527740241	均方差(RMSE): 0.168304727114165 残差平方和(SSE): 0.254938330520761 相关系数(R): 0.910610280469846 相关系数之平方(R^2): 0.829211082897371 决定系数(DC): 0.829211082897371 卡方系数(Chi-Square): -24.9410197450973 F 统计(F-Statistic): 40.9862660806826  参数 最佳估算 ----- a1 0.979643636674007 b 0.636949401737495  传递参数(PassParameter): a: 2.66350689660705

注意在线性化代码中使用了“PassParameter”，用以给出线性化后得到的原非线性公式参数。

表 2-48 线性化拟合结果分析一

参数值	直接非线性拟合 (1)	线性化后拟合 (2)	误差 (%) $\frac{ (1) - (2) }{(1)} \cdot 100$
a	3.232048	2.663512	17.5906
b	0.859500	0.6369517	25.8928
RMSE	0.1881318	0.23698751	25.9689
SSE	0.3185423	0.50546773	58.6815
R <sup>2</sup>	0.904665	0.87728300	3.0268

例 2. 非线性拟合公式:  $y = \exp(p_1 \cdot x^{p_2} \cdot p_3^x)$  (2-64)

其中 p1、p2、p3 为待求系数。

公式线性化处理:

1) 公式两边去对数:  $\ln(y) = p_1 \cdot x^{p_2} \cdot p_3^x$

2) 再取一次对数:  $\ln(\ln(y)) = \ln(p_1) + p_2 \cdot \ln(x) + x \cdot \ln(p_3)$

3) 令  $z = \ln(\ln(y))$ ,  $y = \ln(x)$ ,  $b_1 = \ln(p_1)$ ,  $b_2 = p_2$ ,  $b_3 = \ln(p_3)$ , 则线性化后公式:

$$z = b_1 + b_2 \cdot y + b_3 \cdot x$$

表 2-49 线性化拟合数据二

x	y	z=ln(ln(y))	y=ln(x)
41.670	66.980	1.4361	3.7298
83.330	45.380	1.3390	4.4228
125.000	31.460	1.2380	4.8283
166.700	22.510	1.1359	5.1162
208.300	16.720	1.0355	5.3390
250.000	13.100	0.9449	5.5215
291.700	11.180	0.8813	5.6757
333.300	10.150	0.8405	5.8090
375.000	9.739	0.8225	5.9269
416.700	10.050	0.8362	6.0324
458.300	10.160	0.8409	6.1275
500.000	10.570	0.8578	6.2146
541.700	11.170	0.8810	6.2947
583.300	11.800	0.9034	6.3687

1stOpt 代码

直接非线性拟合	线性化后拟合
Variable x, y; Function Y = exp(p1*x^p2*p3^x); Data; 41.670    66.980 83.330    45.380 125.000    31.460 166.700    22.510 208.300    16.720 250.000    13.100 291.700    11.180 333.300    10.150 375.000    9.739 416.700    10.050 458.300    10.160 500.000    10.570 541.700    11.170 583.300    11.800	PassParameter p1=exp(b1), p2=b2, p3=exp(b3); Function z=b1+b2*y+b3*x; Data; x    y    z 41.670    3.7298    1.4361 83.330    4.4228    1.3390 125.000    4.8283    1.2380 166.700    5.1162    1.1359 208.300    5.3390    1.0355 250.000    5.5215    0.9449 291.700    5.6757    0.8813 333.300    5.8090    0.8405 375.000    5.9269    0.8225 416.700    6.0324    0.8362 458.300    6.1275    0.8409 500.000    6.2146    0.8578 541.700    6.2947    0.8810 583.300    6.3687    0.9034

结果 (图 2-43)

直接非线性拟合	线性化后拟合
均方差(RMSE): 2.74396207457536 残差平方和(SSE): 105.410590133911 相关系数(R): 0.986028946841199 相关系数之平方(R^2): 0.972253084008765 决定系数(DC): 0.971944844669758 卡方系数(Chi-Square): 3.96978151310412 F 统计(F-Statistic): 196.04240059478	均方差(RMSE): 0.0535938055907557 残差平方和(SSE): 0.040212143967796 相关系数(R): 0.962846296535731 相关系数之平方(R^2): 0.927072990752572 决定系数(DC): 0.927072990752572 卡方系数(Chi-Square): 0.0193574675440479 F 统计(F-Statistic): 75.4178740737813

参数 最佳估算	参数 最佳估算
-----	-----
p1 7.38339690571875	b1 2.86897153064041
p2 -0.141282517616353	b2 -0.369142972963057
p3 0.999247320538982	b3 0.000548956707081793
	传递参数(PassParameter):
	p1: 17.618883915984
	p2: -0.369142972963057
	p3: 1.00054910741139

表 2-50 线性化拟合结果分析二

参数值	直接非线性拟合 (1)	线性化后拟合 (2)	误差 (%) $\frac{ (1) - (2) }{(1)} \cdot 100$
p1	7.38340	17.61889	138.6284
p2	-0.14128	-0.36914	161.2825
p3	0.99925	1.00055	0.13009
RMSE	2.743962	8.126508	196.1596
SSE	105.41059	924.5618	777.1052
R <sup>2</sup>	0.904665	0.87728300	3.0268

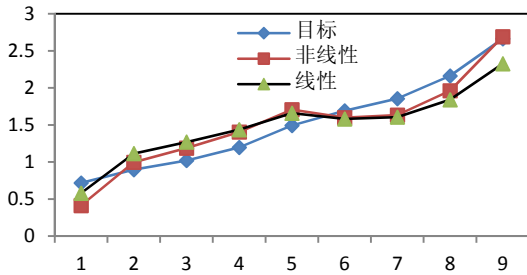


图 2-42 线性化拟合对比结果一

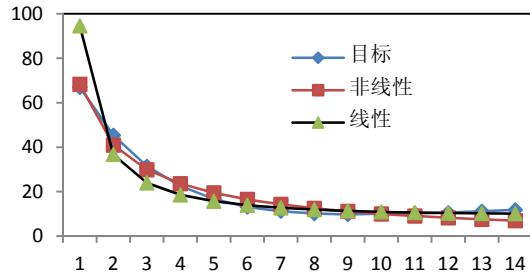


图 2-43 线性化拟合对比结果二

仅从结果看，误差是非常大的。有了 1stOpt 这样拥有先进全局优化算法的软件，进行拟合须提供猜测的初值已成为历史，除非个案，基本已没必要将非线性问题线性化来处理了。

## 2.2.18 与 Origin 的比较

Origin 是当今著名的科学绘图和分析计算软件，而非线性拟合就是其引以为豪的主要功能之一，但与 1stOpt 相比，不论是在易用性还是拟合能力与效果方面，都有相当的差距。下面给出几个实例加以对比说明。

例 1. 自定义非线性拟合

$$\text{模型公式: } y = p_1 + p_2 \cdot \exp\left(-\frac{p_3 \cdot x}{p_5}\right) + \frac{p_4}{1 + p_4 \cdot p_5 \cdot x} \quad (2-65)$$

表 2-51 对比拟合数据一

x	0,0.098,0.195,0.293,0.391,0.488,0.586,0.684,0.781,0.879,0.977,1.074,1.172,1.27,1.367,1.465,1.562,1.66,1.758;
y	0.928,1.02,1.12,1.125,1.42,1.7,2.01,2.26,2.46,2.63,2.82,3.01,3.2,3.41,3.59,3.72,3.85,3.98,4.08

本例公式在 Origin 中也是自定义公式。Origin 虽然有自动赋初值的功能，但由于其采用的优化算法均为局部最优算法，因而很难得到下面 1stOpt 可轻松获得的结果。

1stOpt 代码如下：

Function y=p1+p2*Exp(-p3*x/p5)+p4/(1+p4*p5*x); Data; 0,0.098,0.195,0.293,0.391,0.488,0.586,0.684,0.781,0.879,0.977,1.074,1.172,1.27,1.367,1.465,1.562,1.66,1.758; 0.928,1.02,1.12,1.25,1.42,1.7,2.01,2.26,2.46,2.63,2.82,3.01,3.2,3.41,3.59,3.72,3.85,3.98,4.08;	
1stOpt 计算结果:	
均方差(RMSE): 0.0333771635317068 残差平方和(SSE): 0.0211666658630237 相关系数(R): 0.999497560321499 相关系数之平方(R^2): 0.998995373088629 决定系数(DC): 0.998995373088629 卡方系数(Chi-Square): 0.00540325819887849 F 统计(F-Statistic): 3483.88039418846	参数 最佳估算 ----- p1 6.85548568165676 p2 4.81345041356596 p3 -0.542980694450919 p5 -0.151650295309416 p4 -10.7289846802742

例 2. 高斯函数拟合

与上例不同，高斯函数（Gauss）是 Origin 的内置函数，定义如下：

$$y = y_0 + \left( \frac{A}{w \cdot \sqrt{2 \cdot \pi}} \right) \cdot \exp \left( -2 \cdot \left( \frac{x - x_c}{w} \right)^2 \right) \quad (2-66)$$

表 2-52 对比拟合数据二

x	325,350,375,400,425,450,475,500,525,550
y	0.111,0.189,0.253,0.276,0.245,0.189,0.120,0.068,0.034,0.015

1stOpt 代码如下：

Function y=y0 + (A/(w*sqrt(PI/2)))*exp(-2*((x-xc)/w)^2); Data; 325,350,375,400,425,450,475,500,525,550; 0.111,0.189,0.253,0.276,0.245,0.189,0.120,0.068,0.034,0.015;	
1stOpt 计算结果:	
均方差(RMSE): 0.00334020818797656 残差平方和(SSE): 0.000111569907390256 相关系数(R): 0.999296976874618 相关系数之平方(R^2): 0.99859444799075 决定系数(DC): 0.99859444799075 卡方系数(Chi-Square): 0.000580957937930361 F 统计(F-Statistic): 1422.92849132422	参数 最佳估算 ----- y0 0.00976258766817963 a 37.2410336365238 w 112.063492168364 xc 400.407929317179

Origin 即使对上面比较简单的拟合问题也难以快速正确处理。

在非线性拟合方面，与 1stOpt 相比，Origin 有以下两点不足：

- 操作繁琐，人为干涉因素多；
- 采用优化算法的缺陷使其全局寻优能力偏弱，求得正解的概率低。

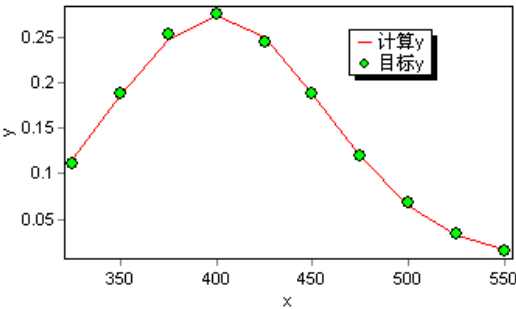


图 2-44 高斯函数拟合结果

### 2.2.19 与 Lingo 的比较

非线性拟合一般采用最小二乘法，其从本质上来说就是一个优化问题。目前公认和使用最广泛的拟合优化算法是麦夸特法，该算法虽然高效但一最大的缺点就是其属于局部最优算法，也即计算结果依赖合适初值的猜测。Lingo 具有享有盛名的全局优化求解器，在此通过几个实例检验其非线性拟合方面的效果，并同时与 1stOpt 进行比较。

优化模型构筑时，均将非线性拟合转换成函数优化形式，即计算因变量与实际因变量间误差平方和最小：

$$\text{Min } RSS = \sum_{i=1}^n (Y_i - y_i)^2 \tag{2-67}$$

例 1. 拟合公式：
$$y = \frac{p_1}{p_2 + p_3 \cdot \exp(-p_3 \cdot x + p_1)} + p_4 \cdot x^{p_5} \tag{2-68}$$

表 2-53 对比拟合数据三

x	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26
y	33815,33981,34004,34165,34212,34327,34344,34458,34498,34476,34483,34488,34513,34497,34511,34520,34507,34509,34521,34513,34515,34517,34519,34519,34521,34521

优化模型：
$$\min \sum_{i=1}^n \left( \left( \frac{p_1}{p_2 + p_3 \cdot \exp(-p_3 \cdot x_i + p_1)} + p_4 \cdot x_i^{p_5} - y_i \right)^2 \right) \tag{2-69}$$

1stOpt 代码如下

```
Algorithm = UGO1[50];
DataSet;
  x=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26;
  y=33815,33981,34004,34165,34212,34327,34344,34458,34498,34476,34483,34488,34513,34497,34511,34520,34507,34509,34521,34513,34515,34517,34519,34519,34521,34521;
EndDataSet;
MinFunction Sum(x,y)((p1/(p2+p3*exp(-p3*x+p1))+p4*x^p5-y)^2);
```

Lingo 代码如下

```
Sets:
  Dat/1..26/:x,y;
  Par/1..5/:p;
EndSets
Data:
  x=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26;
  y=33815,33981,34004,34165,34212,34327,34344,34458,34498,34476,34483,34488,34513,34497,34511,34520,34507,34509,34521,34513,34515,34517,34519,34519,34521,34521;
EndData
Min=@Sum(dat:((p1)/(p2+p3)*exp(-p3*x+p1))+p(4)*x^p(5)-y)^2);
@For(Par: @Free(p));
```

本例题看似并不复杂，求解参数也只有 5 个，但全局最优解却非常难以获得。上述代码，1stOpt 能以约 20% 的概率得到下面最优解，而 Lingo 却只能得到局部最优解。

结果

1stOpt	Lingo
目标函数值(最小): 9589.39821870966 p1: -8.98527141247 p2: -0.000260147025317434 p3: 1.49792015327337 p4: -7364.33144471589 p5: -1.99617448648483	目标函数值(最小): 86610.67 P(1) 13.47931 P(2) -50.50229 P(3) 1.069133 P(4) 33866.44 P(5) 0.6590676E-02

例 2. 三维拟合

$$\text{拟合公式: } z = a \cdot \exp(b \cdot x) + c \cdot \exp(d \cdot y) + p_1 \cdot (p_2 \cdot x + p_3 \cdot y)^{p_4} \quad (2-70)$$

表 2-54 对比拟合数据四

x	1000,600,1200,500,300,400,1300,1100,1300,300
y	5,7,6,6,8,7,5,4,2,9
z	100,75,80,70,50,65,90,100,110,60

1stOpt 代码如下

```
DataSet;
  x= 1000,600,1200,500,300,400,1300,1100,1300,300;
  y= 5,7,6,6,8,7,5,4,2,9;
  z= 100,75,80,70,50,65,90,100,110,60;
EndDataSet;
MinFunction Sum(x,y,z)((a*exp(b*x)+c*exp(d*y)+p1*(p2*x+p3*y)^p4-z)^2);
```

Lingo 代码如下

```
Sets:
  Dat/1..10/:x,y,z;
EndSets
Data:
  x= 1000,600,1200,500,300,400,1300,1100,1300,300;
  y= 5,7,6,6,8,7,5,4,2,9;
  z= 100,75,80,70,50,65,90,100,110,60;
EndData
Min= @Sum(Dat: (a*@exp(b*x)+c*@exp(d*y)+p1*(p2*x+p3*y)^p4-z)^2);
@Free(a);@Free(b);@Free(c);@Free(d);@Free(p1);@Free(p2);@Free(p3);@Free(p4);
```

1stOpt 对此问题求得最优解的概率接近 100%，而 Lingo 的最好结果如下，与最优解相距甚远。

结果

1stOpt	Lingo
目标函数值(最小): 47.9086168767324 a: -282.436803450259 b: -0.00145706447117894 c: 147.361098525535 d: 0.175764449240005 p1: -85.7004829842418 p2: 0.000304789664983706 p3: 0.222966231588406 p4: 2.31132682754351	目标函数值(最小): 323.3838 A 15.59318 B -0.8412757E-01 C 68.10107 D -0.1562307 P1 65.56637 P2 0.6539922E-03 P3 -0.1036598E-01 P4 0.2515391

在非线性拟合方面，Lingo 明显不如 1stOpt 快捷和准确。

## 2.2.20 与其它数学软件的比较

非线性拟合应用非常广泛，包含有此项功能的软件众多，如 Matlab、Mathematica、Maple、SAS、SPSS 都有专门的命令或功能用于非线性拟合，同时这些综合性软件有的还含有全局优化工具箱或求解器；此外还有一些专门的非线性拟合软件如 NLReg、DataFit 等，但目前为止，基于大量的实例测试，不论是从易用性还是求解准确率方面，1stOpt 无疑都是处于领先地位。

## 2.3 方程及方程组求解

1stOpt 可求解任意形式、约束或无约束线性、非线性方程或方程组，由于采用的是全局优化算法，因此也不需要提供初值。其主要关键字是：

- Function: 定义方程
- Parameter: 定义求解参数

### 2.3.1 一般方程组求解

方程组求解 例 1.

$$\begin{cases} (x-0.3)^{y^z} + \frac{x}{y \cdot z} - x \cdot y \cdot \sin(z) + (x+y-z)^{\cos(x-1)} = 1 \\ (y-0.2)^{z^x} + \frac{y}{z \cdot x} - y \cdot z \cdot \sin(x) + (y+z-x)^{\cos(y-2)} = 2 \\ (z-0.1)^{x^y} + \frac{z}{x \cdot y} - z \cdot x \cdot \sin(y) + (z+x-y)^{\cos(z-3)} = 3 \end{cases} \quad (2-71)$$

1stOpt 代码:

```
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;
```

结果:  $x = 0.79390634413219$ ,  $y = 0.902585377949916$ ,  $z = 1.21622367662841$

方程组求解 例 2.

$$\begin{cases} \exp(0.1 \cdot x_1) - \exp(0.1 \cdot x_2) - x_3 \cdot (\exp(-0.1) - \exp(-1)) = 0 \\ \exp(0.2 \cdot x_1) - \exp(0.2 \cdot x_2) - x_3 \cdot (\exp(-0.2) - \exp(-2)) = 0 \\ \exp(0.3 \cdot x_1) - \exp(0.3 \cdot x_2) - x_3 \cdot (\exp(-0.3) - \exp(-3)) = 0 \end{cases} \quad (2-72)$$

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

1stOpt 代码:

```
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(-1))=0;
exp(-0.2*x1)-exp(0.2*x2)-x3*(exp(-0.2)-exp(-2))=0;
exp(-0.3*x1)-exp(0.3*x2)-x3*(exp(-0.3)-exp(-3))=0;
```

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

## 2.3.2 循环方程求解

如下列方程组,  $k$  为变量, 范围为  $[0, 1]$ , 变化步长为 0.05, 试求对应与不同  $k$  值的  $x$  和  $y$  值。

$$\begin{cases} 0.23 + 0.32 \left( 1 + 1.5 \left( \frac{x}{0.18} - 1 \right) k - 0.5 \left( \frac{x}{0.18} - 1 \right)^3 \right) - y = 0 \\ x - k \cdot y^{0.5} = 0 \end{cases} \quad (2-73)$$

1stOpt 中, 可用关键字 “LoopConstant” 来描述循环赋值求解的问题, 代码如下, 结果见图 2-45。

```
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;
```

## 2.3.3 循环递归方程求解

如下列递归方程组, 已知  $n = 50$ ,  $x_n = 200$ ,  $y_n = 0.3$ , 试求  $x_n$ ,  $x_{n-1}$ ,  $x_{n-2} \cdots x_1$  及  $y_n$ ,  $y_{n-1}$ ,  $y_{n-2} \cdots y_1$

$$\begin{cases} (x_{i+1} - x_i) \cdot y_{i+1} = \frac{L}{2} \cdot \left( -4.5 \cdot (\sin(y_{i+1}) + \sin(y_i)) + 0.02 \cdot ((1 + 2x_{i+1})^{0.5} \cdot \cos(y_{i+1})^2 + (1 + 2x_i)^{0.5} \cdot \cos(y_i)^2) \right) \\ (y_{i+1} - y_i) \cdot x_{i+1} = \frac{L}{x_{i+1} + x_i} \cdot \left( -4.5 \cdot (\cos(y_{i+1}) + \cos(y_i)) + 2 \cdot ((1 + 2x_{i+1})^{0.5} \cdot \sin(y_{i+1})^2 + (1 + 2x_i)^{0.5} \cdot \sin(y_i)^2) \right) \end{cases} \quad (2-74)$$

本例是在已知  $x_{i+1}$  及  $y_{i+1}$  的基础上求解  $x_i$  及  $y_i$ , 求出  $x_i$  及  $y_i$  后, 在其基础上再求解  $x_{i-1}$  及  $y_{i-1}$ , 以此类推至求出  $x_1$  及  $y_1$ , 1stOpt 代码如下, 结果见图 2-46。

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



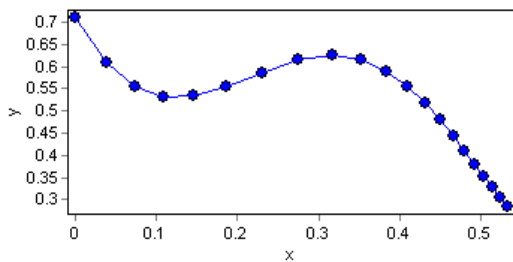


图 2-45 循环拟合结果

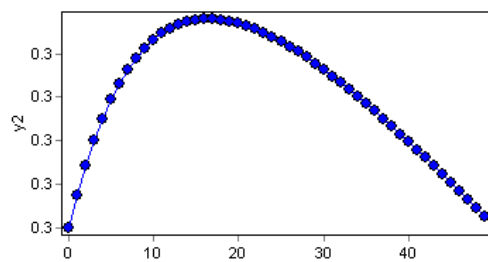


图 2-46 递归求解结果

表 2-55 递归方程计算结果

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

## 2.3.4 整数方程求解

### 魔法矩阵问题

将 1 至 16 个数排列成 4 行 4 列矩阵如表. 12, 每个数只用一次, 如何排列使其每行, 每列及两对角 4 数之和均分别等于 34? 也即求  $X_1$  至  $X_{16}$  的值.

表 2-56 魔法矩阵表

	$X_1$	$X_2$	$X_3$	$X_4$	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

表 2-57 魔法矩阵结果

	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

1stOpt 代码

```

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;

```

该问题可归结为整数方程求解。1stOpt 代码及结果如下，代码中使用了关键字“Exclusive”，意即每个整数参数值是排他的，不能相等。

见表 2-57，计算结果如下： $x_1 = 2$ ,  $x_2 = 14$ ,  $x_3 = 15$ ,  $x_4 = 3$ ,  $x_5 = 7$ ,  $x_6 = 11$ ,  $x_7 = 10$ ,  $x_8 = 6$ ,  $x_9 = 9$ ,  $x_{10} = 5$ ,  $x_{11} = 8$ ,  $x_{12} = 12$ ,  $x_{13} = 16$ ,  $x_{14} = 4$ ,  $x_{15} = 1$ ,  $x_{16} = 13$

## 2.3.4 复数方程求解

复数方程的求解与普通方程并无二意，只需用到两个关键字“ComplexStr”和“ComplexPar”，前者用于定义虚数符号，后者定义复数型参数，没有定义的参数将自动视为实数型参数。

例 1. 方程组: 
$$\begin{cases} x \cdot y \cdot i + i \cdot 4 \cdot \pi = 3 \\ x^y - (2 \cdot x - y) \cdot i = 0 \end{cases} \quad (2-75)$$

其中： $x$ 、 $y$  为复数型参数， $i$  为虚数符号。

1stOpt 代码如下

```
ComplexStr = i;
ComplexPar x,y;
Function x*y*i+i*4*pi=3;
        x^y-(2*x-y)*i=0;
```

结果

```
目标函数值(最小): 4.09337433674202E-29
x.realpart: 0.297311264077123
x.imagpart: -2.52439153859475
y.realpart: 0.593883730135222
y.imagpart: -5.04792491262037
```

“realPart”和“imagPart”分别表示实部和虚部。

例 2. 方程组: 
$$\begin{cases} z_1^2 = z_1 \cdot (z_2 + i \cdot z_1) \\ z_1 + z_2 = 0.5 \cdot \left(1 + (3 \cdot i + a)^{\frac{1}{3}i}\right) \end{cases} \quad (2-76)$$

其中： $z_1$ 、 $z_2$  为复数型参数， $i$  为虚数符号， $a$  为变系数，范围 $[0, 3]$ ，变动步长 0.2。

1stOpt 代码如下

```
LoopConstant a=[0:0.2:3];
ComplexPar z1, z2;
ComplexStr = i;
PlotLoopData z1[realPart],z1[imagPart][y2];
Function z1^2=z1*(z2+i*z1);
        z1+z2=(1+(3*i+a)^(1/3)*i)/2;
```

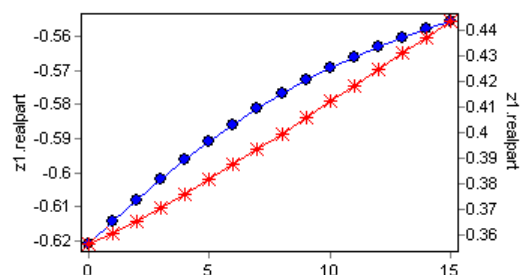


图 2-47 复数拟合结果对比图

上面代码中使用关键字“PlotLoopData”，在计算的同时分别画出了  $z_1$  实部和虚部的变化图形，虚部以右边为  $y$  轴 ( $y_2$ )。按“数据报表”，详细结果显示如下。

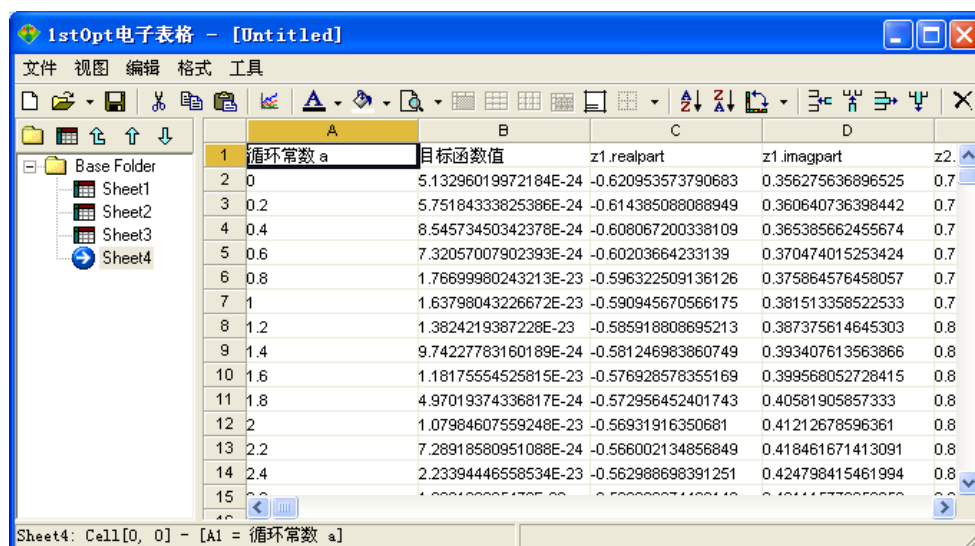


图 2-48 复数拟合结果数据

例 3. 方程组: 
$$\begin{cases} 5 \cdot x^{y_1} + (x - y) \cdot i = 3 \cdot x + 2 \cdot i \\ y \cdot y_1 \cdot i \cdot x - 3 \cdot i \cdot y_1 \cdot x = 1 + 3 \cdot i \cdot \cos(y_1 \cdot i + y - x) \end{cases} \quad (2-77)$$

其中:  $x$ 、 $y$  为复数型参数,  $i$  为虚数符号,  $y_1$  为  $y$  的共轭复数, 共轭函数用“Conjugate()”表示。

1stOpt 代码如下

```
ComplexStr = i;
ComplexPar x,y;
ConstStr y1=Conjugate(y);
Function 5*x^y1+(x-y)*i=3*x+2-2*i;
        y*y1*i*x-3*i*y1*x=1+3*i*Cos(y1*i+y-x);
```

结果

目标函数值(最小): 1.295385093459E-22  
 $x$ .realpart: -0.970529493525754  
 $x$ .imagpart: 0.672924123012695  
 $y$ .realpart: 0.518780437588187  
 $y$ .imagpart: -0.500921769272053

## 2.4 常微分方程数值求解

常微分方程的求解是数学研究与应用的重要组成, 在工程应用方面也极为广泛的。常微分方程一般有解析解和数值解两类求解方法, 而在很多情况下, 解析解是很难或无法获取的, 这时只能用数值方法去求解。1stOpt 只支持数值解。

### 2.4.1 1stOpt 求常微分方程的关键字

表 2-58 常微分方程主要关键字

关键字	意义
Variable	定义变量名及其区间, 必需
ODEFunction	定义微分方程或方程组, 必需
Plot, PlotLoopData	作图命令, 选项
ChartType	设定曲线类型, 1: 点-线图, 2: 线-线图, 3: 点-点图, 选项
InitialODEValue	设定初始值, 仅用于边值和微分方程拟合问题
iValue	功能等同于 InitialODEValue

ODEStep	设定计算步长，等同于“ODEOptions”中的“SN”项
ODEOptions	<p>常微分方程求解选项，基本形式如下：  ODEOptions = [SN=10,A=0,P=5]  其中：</p> <ul style="list-style-type: none"> <li>• SN: 求解的步长数，正整数，也可变为 SS，表示步长大小；</li> <li>• A: 设定微分方程求解算法，0：龙格-库塔-费尔博格法 (Runge-Kutta-Fehlberg Method)，1：欧拉法，2：二阶龙格-库塔法，3：三阶龙格-库塔法，4：四阶龙格-库塔法，5：五阶龙格-库塔法</li> <li>• P: 种群数，仅用于边值问题优化求解，数值越大，计算收敛可能性越高，但计算时间越长。</li> </ul> <p>例：ODEOptions = [SN=10,A=0,P=5]  步长数为 10，算法为龙格-库塔-费尔博格法，种群数为 5</p> <p>例：ODEOptions = [SS=0.1,A=4,P=20]  步长值为 0.1，算法为四阶龙格库塔法，种群数为 20</p>

常微分方程求解选项也可通过设置面板设定。代码级设定优先于面板设置。

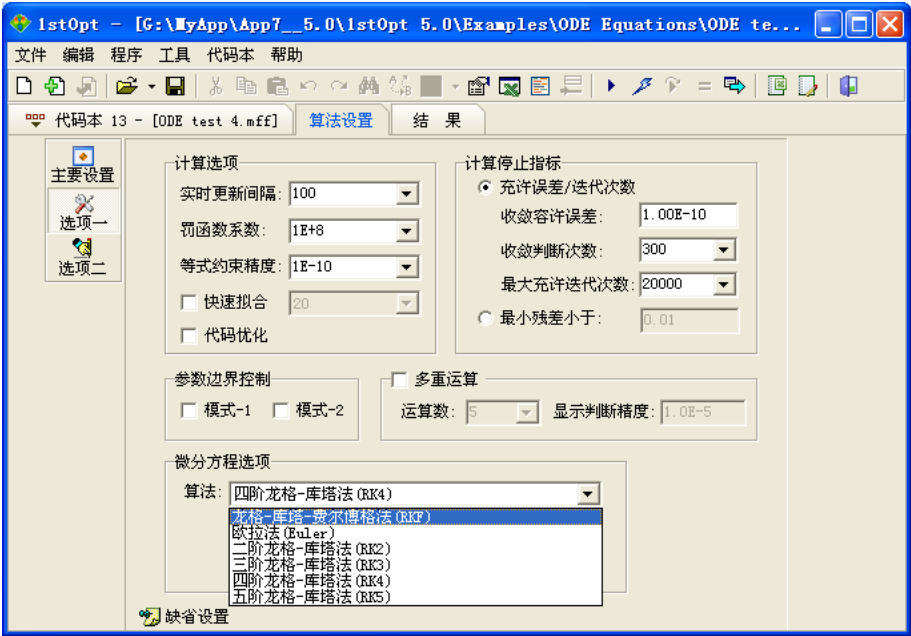


图 2-49 常微分方程算法设定

### 2.4.2 常微分方程初值问题（Initial Value Problem – IVB）

初值问题是常微分方程最常见的类型。1stOpt 可自动处理任意高阶常微分方程或方程组，不需人为降阶处理。书写格式是必须将最高阶项单独写在等式左面。

例1.  $y' = y^{\sin(t-y)} - \ln(yt)$  (2-78)

初值：  $t = 0.2$  时  $y' = 1.5$ ，  $t$  区间  $[0.2, 4]$

1stOpt 代码

```
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);
```

上面代码中，“Plot t[x], y, y' [y2]”意即 t 为 x-轴，y 为左竖轴，y' 为右竖轴，被积区间 t 的变化步长为 0.1。

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

(2-79)

这里 f(t) 是一个关于 t 的分段函数：

$$f(t)=\begin{cases} t & t<1 \\ 2 \cdot t & 1 \leq t < 2 \\ 4 & 2 \leq t < 5 \\ 0 & t>5 \end{cases}$$

初值 y(0) = 0， y'(0) = 0，积分区间[0, 6]，步长 0.1。 . 试画出 y、y''、y' 随着 t 变化的曲线。

微分方程求解代码编写时，需将最高阶数项单独分离写在方程等式的左边。

1stOpt 代码

Variable t=[0.0:0.1:6], y=0,y'=0;

ConstStr f=if(t<=1,t,if(t<=2,2\*t,if(t<=5,4,0)));

Plot t[x],y,y',y'';

ODEFunction y''=f-2\*t\*y'-y\*t;

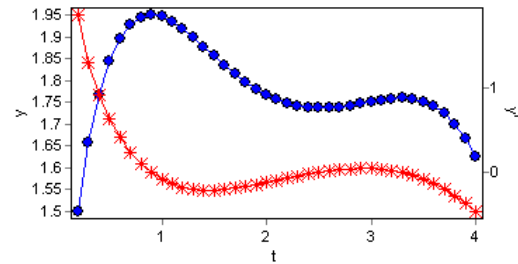


图 2-50 常微分方程例 1 结果

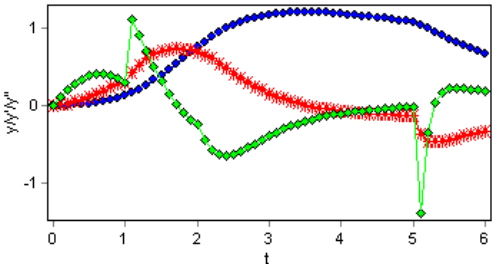


图 2-51 常微分方程例 2 结果

例 3.二阶常微分方程

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

(2-80)

初值： x=0时 y' = -0.1， y = 0， x 范围[0, 14]

1stOpt 代码

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;

结果

x	y(x)	y'(x)	y''(x)
0	0	-0.1	-0.1
14	0.2274063516	7.5406670511	3.967022721

上面语句中，“Plot y[x]”意即 y 作为 x 轴。

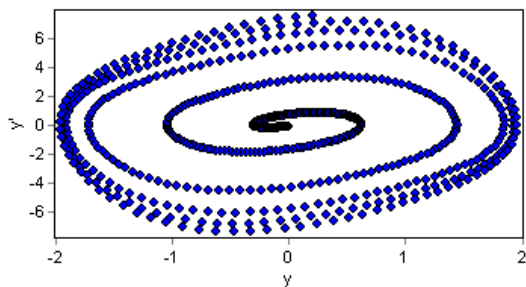


图 2-52 二阶常微分方程计算结果

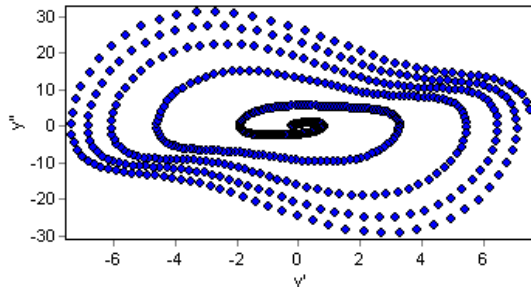


图 2-53 二阶常微分方程计算结果

上述 1stOpt 代码中，如果将 “Plot y[x], y;” 改为 “Plot y'[x], y'';”，可得图形如图 5-23

## 2.4.3 隐式常微分方程及方程组

隐式微分方程通常指方微分程中最高阶微分项不可能分离并单独写在公式的左边。无需特殊处理，1stOpt 可以求解任一阶隐式常微分方程或方程组。

例 1. 微分方程组如下，x 范围 [0, 2.5]，变化步长 0.05，初值条件： $y_1(0) = 1$   
 $y_2(0) = 0.25$ 。

$$\begin{cases} \frac{dy_1}{dx} = \cos\left(y_1 - \sin(x + y_2) + \frac{dy_2}{dx}\right) - \sin\left(\frac{2 \cdot x}{y_1} + y_2\right) \\ \frac{dy_2}{dx} = -2 \cdot x \cdot y_2 + y_1 + \sin\left(x - \frac{dy_1}{dx}\right) \cdot y_1 \end{cases} \quad (2-81)$$

1stOpt 代码

```
Variable y1=1,y2=0.25, x=[0:0.05:2.5];
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;
```

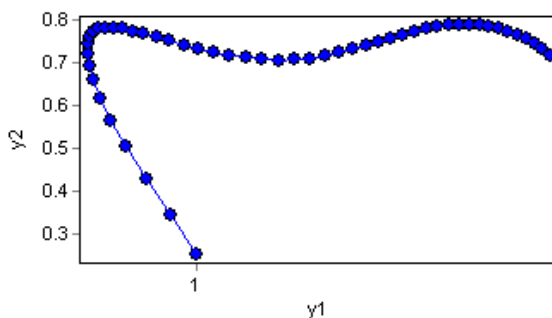


图 2-54 隐式常微分方程计算结果

例 2. 隐式微分方程组

$$\begin{cases} \frac{df^2}{dx^2} = \sin(x) \cdot \frac{dg}{dx} \\ \frac{dg}{dx} = \cos(x) \cdot f \end{cases} \quad (2-82)$$

x 范围 [0, 2π]，变化步长 0.1，初值条件： $f(0) = 10$ ， $f'(0) = 10$ ， $g = 0$

1stOpt 代码

```
Variable x=[0:0.1:2*pi],f=10,f'=1,g=0;
Plot f[x],g,f[y2];
ODEFunction f''=sin(x)*g';
g'=cos(x)*f;
```

例 3. 隐式微分方程组：

$$\begin{cases} \frac{dy^2}{dt^2} = -0.15 \cdot \frac{dx^2}{dt^2} - 0.02 \cdot \pi \cdot \frac{dy}{dt} - 0.16 \cdot y - 0.02 \cdot \pi \cdot x^2 \\ \frac{dx^2}{dt^2} = 8 \cdot \left( -0.003 \cdot \pi \cdot \frac{dy}{dt} - (0.025 + 0.003 \cdot \pi) \cdot x - 0.12 \cdot \frac{dy^2}{dt^2} \right) \end{cases} \quad (2-83)$$

x 范围 $[0, 2\pi]$ ，变化步长 0.1，初值条件： $f(0) = 10$ ， $f'(0) = 10$ ， $g = 0$ 。

隐式微分方程的求解涉及到迭代优化计算，缺省状态下微分方程选项中的“种群数”为 5，增加该数值可提高求解精度（但降低求解速度），该选项在代码中可通过“ODEOptions”来设定，如欲将种群数设为 30，其基本格式为：“ODEOptions = [p=30];”；此外“Plot”关键字可输出复合型数值，如“Plot sin(x+y)\*y'^2;”。

1stOpt 代码

```
Variable t[0:0.5:15], x=0,x'=0, y=1, y'=0.0;
ODEOptions = [p=30];
Plot x,y',y'^2;
ODEFunction y''=-0.15*x''-0.02*pi*y'-0.16*y-0.02*pi*x^2;
x''=8*(-0.003*pi*y'-(0.025+0.003*pi)*x-0.12*y'');
```

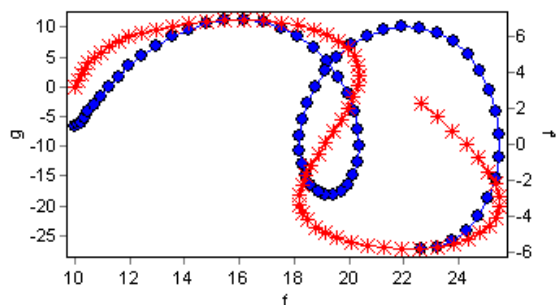


图 2-55 隐式常微分方程组计算结果图

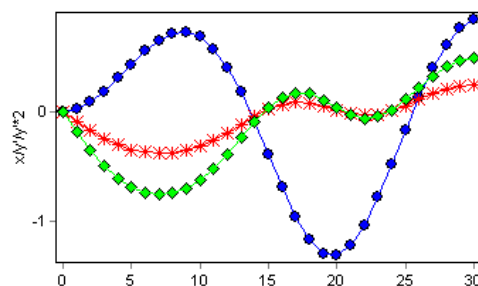


图 2-56 隐式微分方程组结果

## 2.4.4 变系数常微分方程

变系数微分方程指微分方程中有超过一个变动的常数，求其对应的微分方程解。这里主要用到了关键字“LoopConstant”。

例 1 变系数常微分方程组如下：

$$\begin{cases} \frac{dx}{dt} = y - \cos(x + y - a) + y \cdot (\sin(x^2 \cdot y + a)) \\ \frac{dy}{dt} = x - x^3 - 0.1 \cdot y + 0.5 \cdot \cos(0.2 \cdot t \cdot y + x - a) \cdot t \end{cases} \quad (2-84)$$

其中： $a$  为变参数，范围 $[0, 3]$ ，变幅 0.02， $t$  范围 $[0, 2]$ ，初值  $x=0$ ， $y=0$ 。

1stOpt 代码

```
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;
```

例 2. 变边界初值微分方程

$$y'' = \frac{dy}{dt} = (1 - y^2 + a) \cdot y' - \sin(t \cdot y') \cdot y + \cos(t \cdot y') \quad (2-85)$$

边界条件:  $t = 0$  时  $y = a$ ,  $y' = 2 \cdot a$ ,  $a$  为变动系数, 范围[0, 1], 变幅 0.01。

1stOpt 代码

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

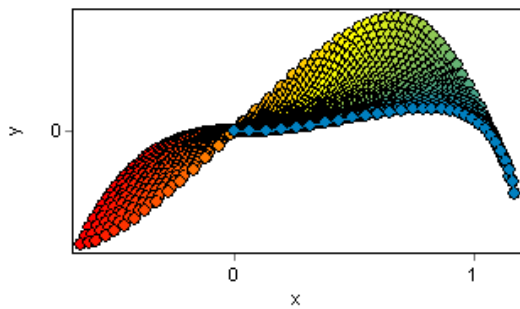


图 2-57 变系数常微分方程计算结果

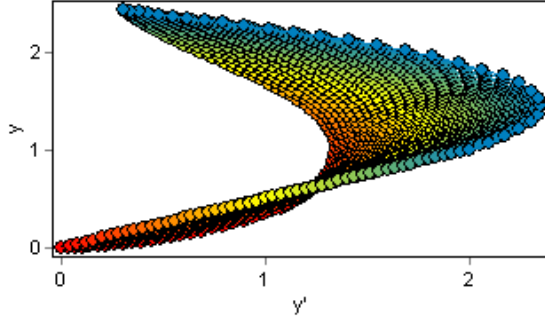


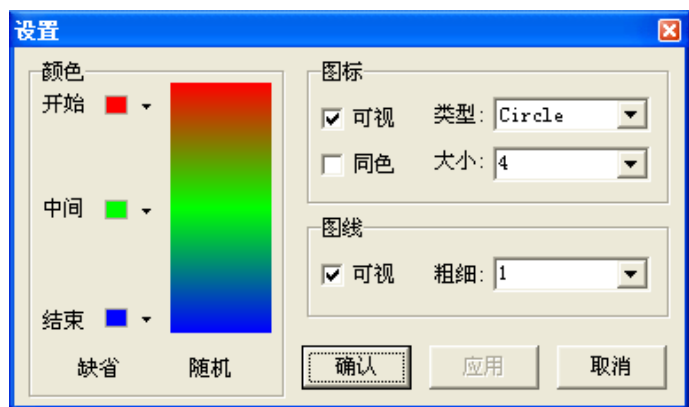
图 2-58 变边界常微分方程计算结果

点击“数据报表”按钮或右键弹出菜单选“获取数据”：

t	y(t)	y'(t)	y''(t)	y'''(t)
0	0	0	0	1
0.04	0.000810774036435141	0.04081075276439	0.04081075276439	1.04080806999216
0.08	0.00328705819293907	0.083286349462698	0.083286349462698	1.08324135128114
0.12	0.00749673951934843	0.127491139923877	0.127491139923877	1.12725226066027
0.16	0.0135102160405982	0.17348566077579	0.17348566077579	1.17269381031866
0.2	0.021400158206794	0.221322165892486	0.221322165892486	1.21929434105363
0.24	0.031241070404446	0.271039075885422	0.271039075885422	1.26662879634708
0.28	0.0431086066500797	0.322654218803254	0.322654218803254	1.31408717044519
0.32	0.0570785901693746	0.37615677415952	0.37615677415952	1.36084154236485
0.36	0.073225684462776	0.431497909425633	0.431497909425633	1.40581418654351
0.4	0.0916216652332328	0.488580224372863	0.488580224372863	1.44765062725988
0.44	0.112333251136259	0.547246304588004	0.547246304588004	1.48470308681045
0.48	0.13541946722856	0.607266935240646	0.607266935240646	1.51503133075501
0.52	0.160928544131523	0.668329830631976	0.668329830631976	1.5364290050697
0.56	0.188894395792806	0.730030063553759	0.730030063553759	1.54648354897247
0.6	0.21933277266616	0.791863671702832	0.791863671702832	1.54267588485524
0.64	0.252237250053179	0.853226087232334	0.853226087232334	1.52252166582588
0.68	0.2875757375002348	0.9134458072517410	0.9134458072517410	1.49274697376988

图形上点击右键选“图形选项...”：





将图线粗细定为 10，可得图像如下：

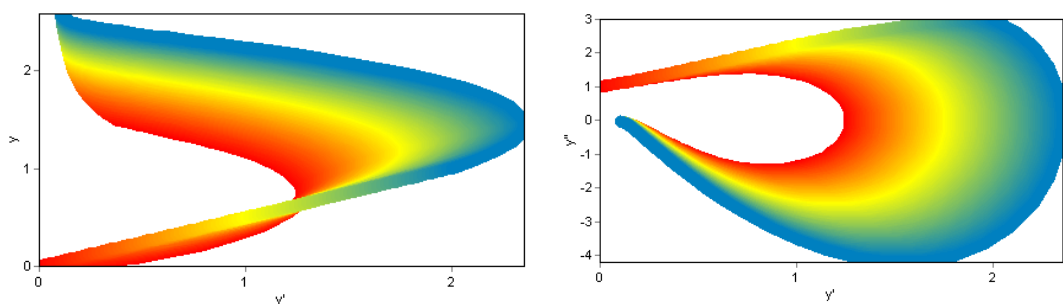


图 2-59 变边界微分方程效果图

例 3. 变系数隐式常微分方程组：

微分方程为隐式的变系数微分方程求解，微分方程的求解即要用到关键字“LoopConstant”，也要涉及到优化迭代计算。

隐式微分方程组：

$$\begin{cases} \frac{dy_1}{dx} = \cos\left(y_1 - \sin(x + y_2) + \frac{dy_2}{dx} \cdot a\right) - \sin\left(\frac{2 \cdot x}{y_1} + y_2 - a\right) \\ \frac{dy_2}{dx} = -2 \cdot x \cdot y_2 + y_1 + \sin\left(x - \frac{dy_1}{dx} + a\right) \cdot y_1 \end{cases} \quad (2-86)$$

其中：a 为变参数，范围[0, 1]，变幅 0.1，x 范围[0, 2.5-a]。

1stOpt 代码

```
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;
```

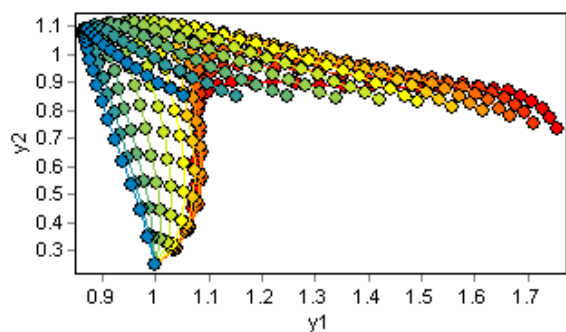


图 2-60 变系数隐式常微分方程计算结果

例 4. 高阶变系数微分方程

$$\left\{ \frac{dy^4}{dx^4} = -\frac{dy^3}{dx^3} x - a \cdot \sin(x - a) \cdot y - \frac{dy^3}{dx^3} \right. \quad (2-87)$$

其中：a 为变参数，范围[1,5]，变幅 0.05；x 范围[0,5]，步长 0.1；初值  $y = 0$ ，

$$y' = \frac{dy}{dx} = 0, \quad y'' = \frac{dy^2}{dx^2} = 1, \quad y''' = \frac{dy^3}{dx^3} = 2.$$

1stOpt 代码

```
LoopConstant a=[1:0.05:5];
Variable x=[0:0.1:5],y=0,y'=0,y''=1,y'''=2;
Plot y,y'[x],y'',y''',y'''';
ODEFunction y''''=-y''*x-a*sin(x-a)*y-y''';
```

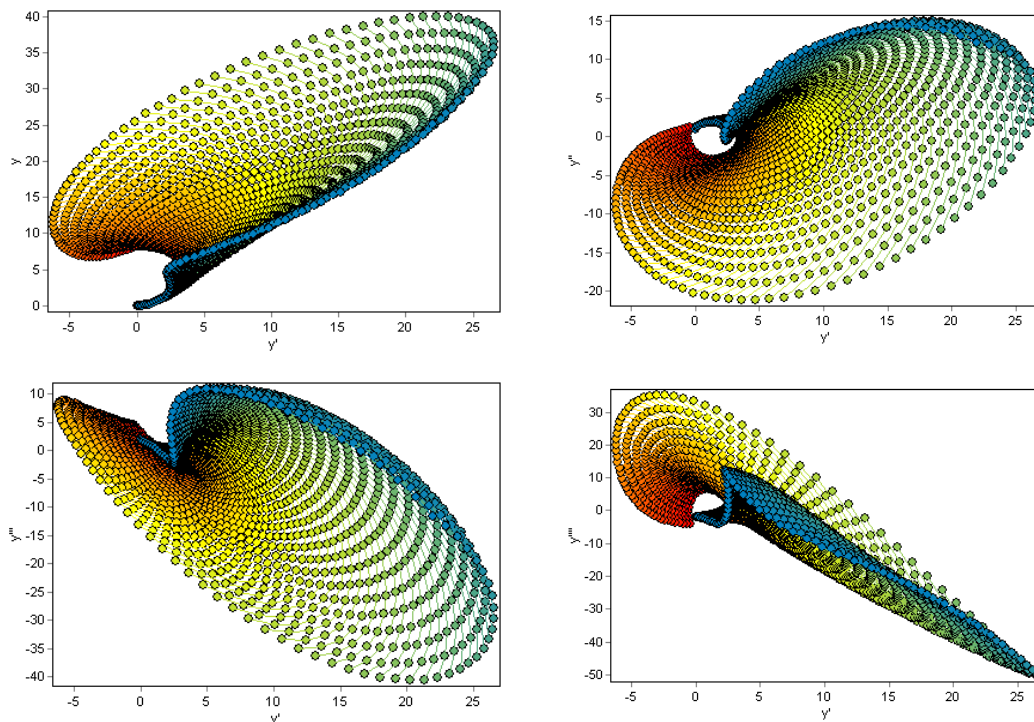


图 2-61 变系数微分方程系列示图

例 5. 高阶变系数微分方程组

$$\begin{cases} \frac{dx_1^2}{dt^2} = -0.1 \cdot x_1 - x_2 + a \\ \frac{dx_2^2}{dt^2} = x_1 - 0.1 \cdot x_2 \cdot a \end{cases} \quad (2-88)$$

其中：a 为变参数，范围[-1,6]，变幅 0.1；t 范围[0,10π]，步长数 200；初值  $x_1=1$ ， $x_2=1$ 。

1stOpt 代码

```
LoopConstant a=[-1:0.1:6];
ODEOptions = [SN=200];
Variable t=[0,10*pi],x1=1,x2=1;
Plot x1[x],x2,x1'[y2];
ODEFunction x1'=-0.1*x1-x2+a;
             x2'=x1-0.1*x2*a;
```

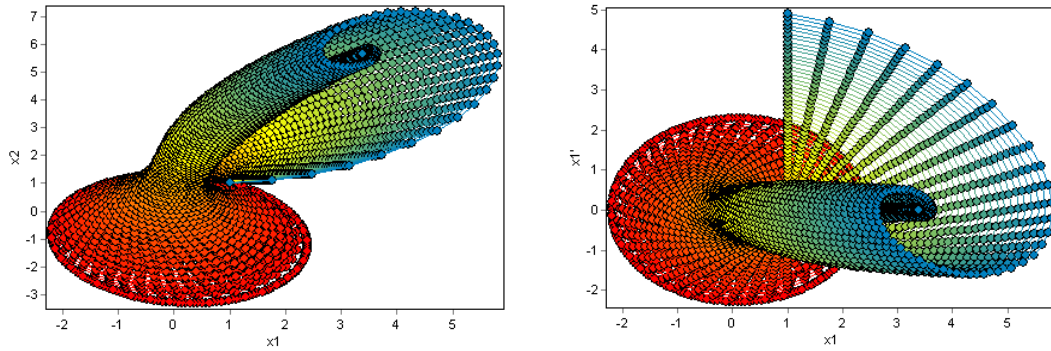


图 2-62 高阶变系数微分方程组示图

#### 例 6：变初值微分方程组

$$\begin{cases} \frac{dx}{dt} = \frac{2 \cdot (\cos(t) - x + y)}{\sqrt{(\sin(t) \cdot y - x - 0.1)^2 + (\cos(t) \cdot x - y)^2}} \\ \frac{dy}{dt} = \frac{2 \cdot (\sin(t) - y + x)}{\sqrt{(\sin(t) \cdot y - x - 0.1)^2 + (\cos(t) \cdot x - y)^2}} \end{cases} \quad (2-89)$$

其中：a 为变参数，范围 $[-3.5, 3.5]$ ，变幅 0.1；t 范围 $[0, 10]$ ，步长 0.2；初值  $x = 2 \cdot a$ ， $y = a^2$ 。

1stOpt 代码

```
LoopConstant a=[-3.5:0.1:3.5];
Variable t=[0:0.2:10],x=2*a,y=a^2;
Plot x[x],y,y+x/2;
ODEFunction x'=2*(Cos(t)-x+y)/Sqrt((Sin(t)*y-x-0.1)^2+(Cos(t)*x-y)^2);
y'=2*(Sin(t)-y+x)/Sqrt((Sin(t)*y-x-0.1)^2+(Cos(t)*x-y)^2);
```

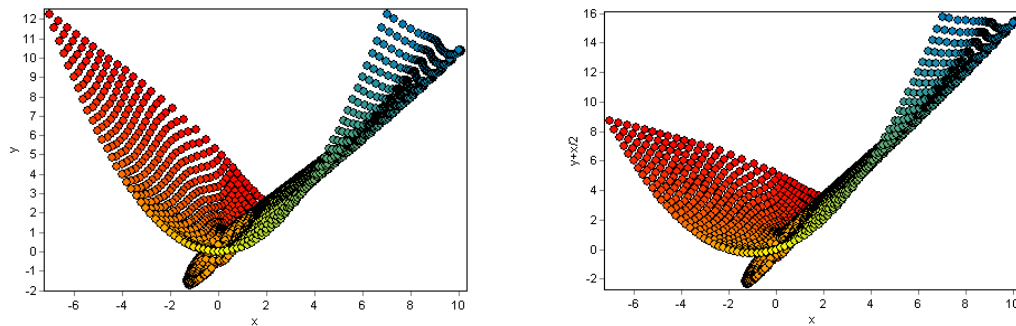


图 2-63 变初值微分方程组示图

#### 例 7. 边值变系数微分方程

$$\left\{ \frac{dy^2}{dx^2} = 0.3 \cdot (1 - y^2) \cdot \frac{dy}{dx} - y \cdot b \right. \quad (2-90)$$

其中：b、c 均为变参数，b 范围 $[1, 1.5]$ ，变幅 0.1，c 范围 $[3, 4]$ ，变幅 0.2；x 范围 $[1, 2]$ ；

边值条件  $y(1) = c$ ， $y(2) = \frac{dy}{dx} \cdot \sin(x) \cdot b$ 。此例的边值条件比较特殊，是动态变化的。

1stOpt 代码

```

LoopConstant b=[1:0.1:1.5], c=[3:0.2:4];
Variable x=[1,2], y=[c,y'*sin(x)*b];
Plot y,y'[x],y''[y2];
ODEFunction y''=0.3*(1-y^2)*y'-y*b;

```

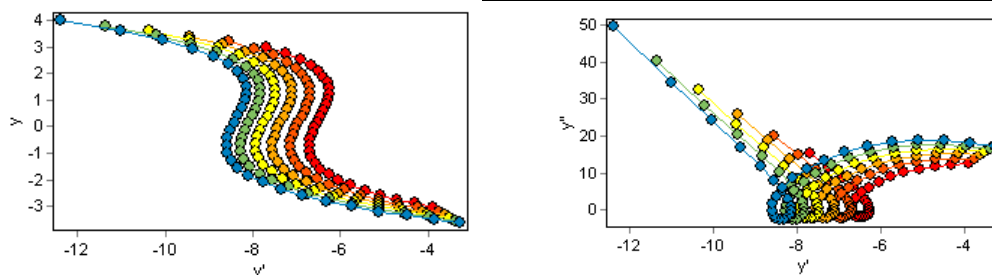


图 2-64 边值变系数微分方程示图

例 8. 变系数高阶微分方程组

$$\begin{cases} \frac{dx^2}{dt^2} = \frac{c \cdot \left( -16000 \cdot t - 160 - \left( 200000 \cdot x - 800 \cdot \frac{dy}{dt} - 80000 \cdot y + 2000 \cdot \frac{dx}{dt} \right) \right)}{185} \\ \frac{dy^2}{dt^2} = \frac{c \cdot \left( 1600 \cdot t + 16 - \left( -8000 \cdot x + 80 \cdot \frac{dy}{dt} - 80 \cdot \frac{dx}{dt} + 8000 \cdot y \right) \right)}{16} \end{cases} \quad (2-91)$$

其中：c 为变参数，范围 [1, 3]，变幅 0.1；t 范围 [1, 3]，步长 0.01；初值条件

$$x = -0.0021512, \quad x' = \frac{dx}{dt} = -0.012185, \quad y = -0.0021512, \quad y' = \frac{dy}{dt} = -0.2.$$

1stOpt 代码

```

LoopConstant c=[1:0.1:3];
Variable t=[0:0.01:3], x'=-0.012185, x=-0.0021512, y=-0.0021512, y'=-0.2;
Plot x[x], x', y[y2];
ODEFunction x''=c*(-16000*t-160-(200000*x-800*y'-80000*y+2000*x'))/185;
y''=c*(1600*t+16-(-8000*x+80*y'-80*x'+8000*y))/16;

```

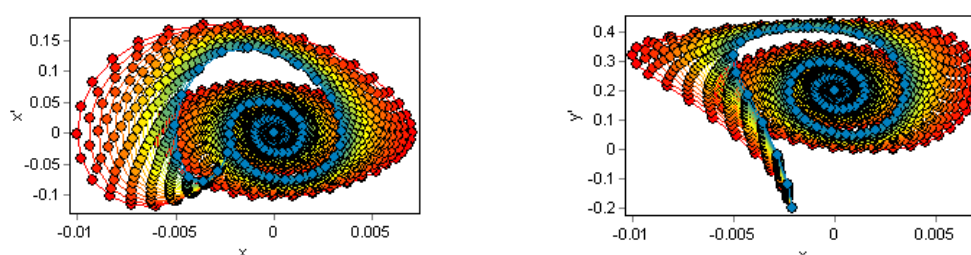


图 2-65 变系数高阶微分方程组示图

## 2.4.5 常微分方程边值问题（Boundary Value Problem – BVB）

边值问题是常微分方程的重要组成部分，其求解方法多用打靶法，对线性问题，打靶法简单易用、效率也高，但对复杂的非线性常微分方程，打靶法则难以满足要求。1stOpt 不仅自动识别初值与边值问题，基于其独特的全局优化算法，对常微分方程边值问题，不论是线性还是非线性都能进行直接求解。1stOpt 求解边值问题有两种模式，第一种形式与求解

初值常微分方程问题基本一致，简单易懂，但对复杂的边值问题难以求解；第二种模式与数据拟合形式相似，稍显复杂但可解决几乎所有类型的边值问题。

例 1.  $4 \cdot \frac{dy^2}{dx^2} + y \cdot \frac{dy}{dx} = 2 \cdot x^3 + 16$  (2-92)

x 区间[2, 3]；边值条件：  $y(2) = 8, y(3) = 35 / 3$ 。该例的解析解为：  $y = x^2 + \frac{8}{x}$

1stOpt 代码

```
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;
```

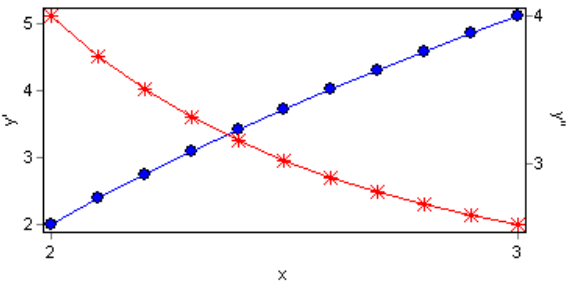


图 2-66 边值问题计算结果

输出结果

常微分方程(边值问题):  
1:  $y' = dy/dx = y'$   
2:  $y'' = dy'/dx = (2 \cdot x^3 + 16 - y \cdot y')/4$   
目标函数: 0  
边值估算:  
     $y'(x=2): 1.99999993866552$   
算法: 龙格-库塔-费尔伯格法(Runge-Kutta-Fehlberg Method)  
步长值: 0.1  
步长数: 10  
种群数: 5

结果:  

x	y(x)	y'(x)	y''(x)
2	8	1.99999993866552	4.00000012266896
3	11.6666666666673	5.1111104586436	2.59259278289476

表 2-59 结果比较

x	解析解	数值解	误差
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	11.6666666666673	11.66666667	-3.33269945E-9

例 2. 微分方程组: 
$$\begin{cases} \frac{dy_1}{dx} = y_2 \\ \frac{dy_2}{dx} = 20 \cdot (y_3 - y_1) \\ \frac{dy_3}{dx} = y_4 \\ \frac{dy_4}{dx} = 10 + 8 \cdot (y_3 - y_1) \end{cases} \quad (2-93)$$

微分区间  $x=[0,3]$ ; 边值条件:  $x=0$  时  $y_2=0$ 、 $y_3=20$ ,  $x=3$  时  $y_1=0$ 、 $y_4=0$

1stOpt 代码

```
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);
```

结果如下表及图 2-67.

常微分方程(边值问题):								
1: $y_1' = dy_1/dx = y_2$								
2: $y_2' = dy_2/dx = 20 \cdot (y_3 - y_1)$								
3: $y_3' = dy_3/dx = y_4$								
4: $y_4' = dy_4/dx = 10 + 8 \cdot (y_3 - y_1)$								
目标函数: 1.39840021860598E-28								
边值估算:								
y1(x=0): 13.4694429556798								
y4(x=0): -19.1527204392172								
算法: 龙格-库塔-费尔博格法(Runge-Kutta-Fehlberg Method)								
步长值: 0.06								
步长数: 50								
种群数: 2								
结果:								
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

例 3. 
$$\frac{dy^2}{dx^2} = y + y^3 \quad (2-94)$$

微分区间  $x=[0,5]$ ; 边界条件:  $x=0$  时  $y' = \frac{dy}{dx} = 1$  及  $x=5$  时  $y' = -1$ 。

1stOpt 代码

```

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;

```

上面语句中，“Plot x[x], y[y2], y', y''”意即 x 作为 x 轴、y 为右边竖轴、y', y''为缺省的左边竖轴

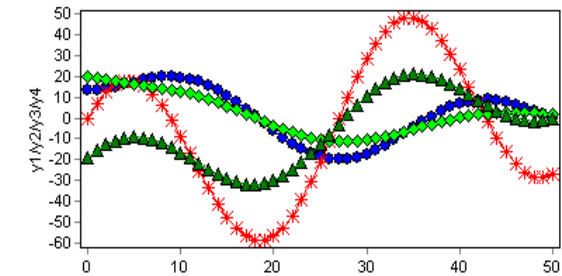


图 2-67 常微分方程组边值问题计算结果

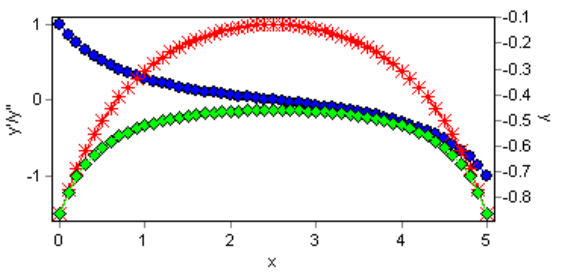


图 2-60 变系数隐式常微分方程计算结果

常微分方程(边值问题):

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

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

目标函数: 1.7241048121671E-27

边值估算:

$y(x=0)$ : -0.86112059752689

算法: 龙格-库塔-费尔博格法(Runge-Kutta-Fehlberg Method)

步长值: 0.1

步长数: 50

种群数: 10

结果:

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

例 4.  $\frac{dy^2}{dx} = (x+1) \cdot y + \exp(-x) \cdot (x^2 - x + 2)$  (2-95)

微分区间  $x=[2,4]$ ; 边界条件:  $x=2$  时  $\frac{dy}{dx} = y - \sin\left(x \cdot \left(\frac{dy}{dx}\right)^2\right)$  及  $x=4$  时  $y' = \frac{dy}{dx} = -0.1$ 。

上述边值条件比较特殊，在  $x=2$  时的  $y'$  值不是常数而是一动态等式约束，1stOpt 中可直接进行定义：“Variable x = [2,4], y' = [y-sin(x\*y'^2),-0.1];”

1stOpt 代码:

```
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);
```

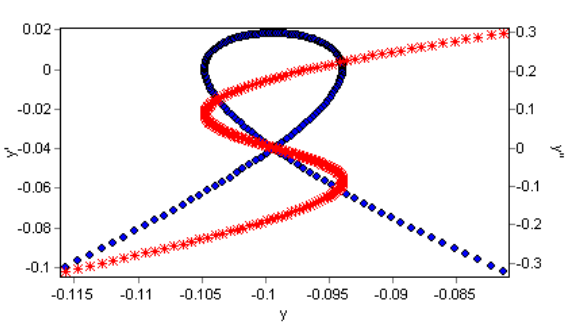


图 2-69 常微分方程组边值问题计算结果

结果			
常微分方程(边值问题):			
1: $y'' = \frac{dy'}{dx} = (x+1)*y + \exp(-x)*(x^2-x+2)$			
2: $y' = \frac{dy}{dx} = y'$			
目标函数: 5.09932323235297E-29			
边值估算:			
$y'(x=2)$ : -0.102184972000791			
$y(x=2)$ : -0.0813029529237538			
算法: 龙格-库塔-费尔伯格法(Runge-Kutta-Fehlberg Method)			
步长值: 0.02			
步长数: 100			
种群数: 30			
结果:			
x	y'(x)	y(x)	y''(x)
2	-0.102184972000791	-0.0813029529237538	0.297432274175189
4	-0.1000000000000008	-0.115680210959961	-0.321982110357528

验证边值条件:  $x = 2$  时,  $y' = y - \sin(x \cdot (y')^2)$

其中:  $x = 2$ 、 $y' = -0.102184972000803$ 、 $y = -0.0813029529237473$

$$y - \sin(x \cdot (y')^2) = -0.0813029529237473 - \sin(2 \cdot (-0.102184972000803)^2)$$

$$= -0.102184972$$

$$= y'$$

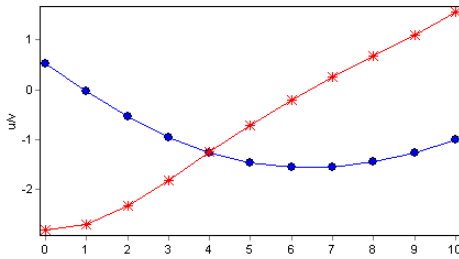
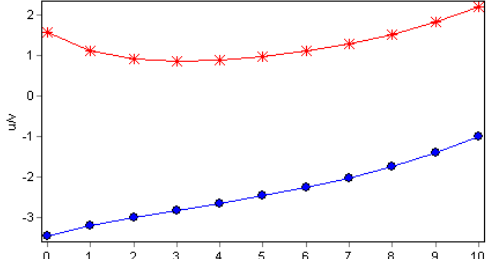
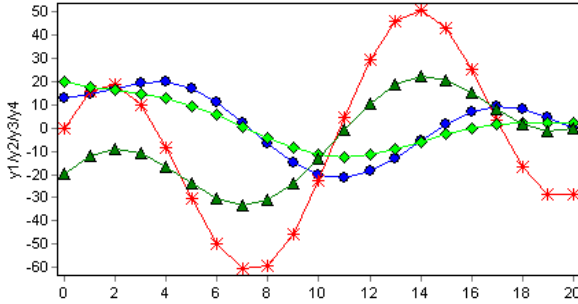
上面验证表明, 计算结果完全满足边界条件。

例 5. 
$$\begin{cases} \frac{du}{dx} = v \\ \frac{dv}{dx} = x + \left(1 - \frac{x}{5}\right) \cdot u \cdot v \end{cases} \quad (2-96)$$

微分区间  $x=[1,3]$ ; 边界条件:  $x = 1$  时  $u = 2 + v \cdot u$ ,  $x = 3$  时  $u = -1$ 。

```
1stOpt 代码
Variable u=[2+v*u,-1], x=[1,3], v;
Plot u, v;
ODEOptions = [SN=10,A=0,P=10];
```



ODEFunction u' = v; v'=x+(1-x/5)*u*v;	
运行上述代码可得如下两组结果	
第一组	第二组
	
常微分方程(边值问题): 1: u' = du/dx = v 2: v' = dv/dx = x+(1-x/5)*u*v  目标函数: 1.15493705914422E-23 边值估算: u(x=1): 0.525411907053075 v(x=1): -2.80653725800423 结果: x u(x) v(x) 1 0.525411907053075 -2.80653725800423 3 -1.00000000005658 1.56242790381221	常微分方程(边值问题): 1: u' = du/dx = v 2: v' = dv/dx = x+(1-x/5)*u*v  目标函数: 2.75908468069419E-23 边值估算: u(x=1): -3.46016007271143 v(x=1): 1.57800794124894 结果: x u(x) v(x) 1 -3.46016007271143 1.57800794124894 3 -1.0000000001061 2.20438043932999
边值条件验证: x=1 时 u=0.52541 2+v*u=2+0.52541*(-2.806537)=0.52541=u 满足条件。	边值条件验证: x=1 时 u=-3.46016 2+v*u=2-3.46016*1.57800=-3.46016=u 满足条件。
例 6.	
$\begin{cases} \frac{dy_1}{dx} = y_2 \\ \frac{dy_2}{dx} = 20 \cdot (y_3 - y_1) \\ \frac{dy_3}{dx} = y_4 \\ \frac{dy_4}{dx} = 8.4 + 2 + 8 \cdot (y_3 - y_1) \end{cases} \quad (2-97)$	
图 2-70 边值问题计算图	
微分区间 x=[0,3]; 边界条件: x = 0 时 y3 = 20, x = 3 时 y1 = 0, y2 = -28.483675, y4 = 0。	
上述边界条件中, y3 是位于起始端, 而 y1、y2、y4 位于终止端, 需求一个边界值即 y3 在 x=3 时的值。	
1stOpt 代码	
Variable y1=[0], y2=[-28.483675], y3=[20.], x=[0,3], y4=[0];	

ODEOptions = [SN=20,A=0,P=2]; Plot y1, y2, y3, y4; ODEFunction y1'=y2; y2'=20*(y3-y1); y3'=y4; y4'=8.4+2+8*(y3-y1);				
结果				
常微分方程(边值问题):				
1: y1' = dy1/dx = y2				
2: y2' = dy2/dx = 20*(y3-y1)				
3: y3' = dy3/dx = y4				
4: y4' = dy4/dx = 8.4+2+8*(y3-y1)				
目标函数: 0				
边值估算:				
y3(x=3): 2.14535283841153				
结果:				
x	y1(x)	y2(x)	y3(x)	y4(x)
3	0	-28.483675	2.14535283841154	0
0	13.0876429039712	0.00165835986955809	20	-19.8058666560522

## 2.4.6 特殊边值问题

例 1. 已知微分方程组如下，微分区间  $t=[0,2]$ ；边界条件： $t=0$  时  $y=1$ ， $t=1$  时  $x=0.43235$ ， $t=2$  时， $z=-2.28072$ 。

$$\begin{cases} \frac{dx}{dt} = 0.25 \cdot y \cdot \left(1 - \frac{y}{20}\right) \\ \frac{dy}{dt} = 2 \cdot x + y - 2 \cdot z \\ \frac{dz}{dt} = 3 \cdot x + 2 \cdot y + z \end{cases} \quad (2-98)$$

该问题的特殊点在于边界值位于不同点，常规求解边值的方法无法求解，在此采用优化拟合的方式，参照微分方程拟合一节，使用关键字“NAN”代表未知或不起作用的数据，数据整理如下表 2-60。

问题的求解实际上变为微分方程拟合，待求参数有两个，即  $x$  和  $z$  的初值。

表 2-60 使用的数据

t	x	y	z
0	待求参数	1	待求参数
1	0.4323539	NAN	NAN
2	NAN	NAN	-2.2807201

1stOpt 代码	结果
InitialODEValue t=0,y=1; Variable t,x,y,z;	均方差(RMSE): 1.60246890531964E-16 残差平方和(SSE): 7.70371977754894E-32

ODEFunction x' = 0.25\*Y\*(1-Y/20);  
y'=2\*x+y-2\*z;  
z'=3\*x+2\*y+z;  
Data;  
1 0.43235 NAN NAN  
2 NAN NAN -2.28072

上面代码也可写为：  
  
Parameter p1,p2;  
InitialODEValue t=0,x=p1,y=1,z=p2;  
Variable t,x,y,z;  
ODEFunction x' = 0.25\*Y\*(1-Y/20);  
y'=2\*x+y-2\*z;  
z'=3\*x+2\*y+z;  
Data;  
1,0.43235,NAN,NAN  
2,NAN,NAN,-2.28072

输出结果变为：  
  
计算用时(时:分:秒:微秒): 00:00:00:656  
均方差(RMSE): 9.61481343191782E-17  
残差平方和(SSE): 2.77333911991762E-32

参数  
-----

x 初值 -2.45504133646076E-6  
z 初值 -0.999994541382211

参数  
-----

p1 -2.4550413362965E-6  
p2 -0.999994541382213

例 2. 已知微分方程组如下，微分区间 t=[0,2]；边界条件：x(0) = 1，y(0) + y(2) = 0。

$$\begin{cases} \frac{dx}{dt} = y \\ \frac{dy}{dt} = -\sin(x) \end{cases} \tag{2-99}$$

本例的第二个边界条件较为特殊：y 的起始值与终点值之和为 0，在此用到关键字“SubjectTo”，求解代码如下：

ODEStep = 0.2; Parameter p; InitialODEValue t=0,x=1,y=p; SubjectTo y[2]=-p; Variable t,x,y; ODEFunction x'=y; y'=-sin(x); Data; 2,NAN,NAN	
结果：	
参数	最佳估算
-----	
p	0.959887101392875

---

微分方程拟合约束(SubjectTo):  
y[2]-(-p): 0

---

例 3. 已知微分方程组如下:

$$\begin{cases} \frac{dx_1}{dt} = x_3 \cdot x_2 \\ \frac{dx_2}{dt} = x_3 \cdot (-x_1 + \sin(x_2)) \\ \frac{dx_3}{dt} = 1 \\ \frac{dx_4}{dt} = -2 \cdot x_1 \end{cases} \quad (2-100)$$

微分区间  $t=[0,1]$ ; 边界条件:  $x_2(0) = 0.6$ ,  $x_3(0) = 2 \cdot x_1(0)$ ,  $x_1(1) = x_4(1) + x_1(0)$ ,

$x_3(0.33) = x_4(0.43)$ 。

上述边界条件更为特殊, 1stOpt 求解代码如下, 注意 “iValue” 与 “InitialODEValue” 功能一样:

```
iValue t=0,x1=p1,x2=0.6,x3=2*p1,x4=p2;
SubjectTo x1[1]=x4[1]+p1,x3[0.33]=x4[0.43];
Variable t,x1,x4;
ODEFunction x1'=x3*x2;
           x2'=x3*(-x1+sin(x2));
           x3'=1;
           x4'=-2*x1;

Data;
1,NAN,NAN
```

结果:

---

参数	最佳估算
----	------

---

p1	-0.134397444549774
p2	-0.0680489509345457

微分方程拟合约束(SubjectTo):  
x1[1]-(x4[1]+p1): -1.38777878078145E-17  
x3[0.33]-(x4[0.43]): 0

===== 输出结果 =====

文件: 数据文件 - 1

No	t	目标 x1	计算 x1	目标 x4	计算 x4	目标 x3	计算 x3
1	0.33	NAN	-0.1545386716152	NAN	0.030844257333876	NAN	0.0612051109004517
2	0.43	NAN	-0.14808947146895	NAN	0.0612051109004517	NAN	0.161205110900452
3	1	NAN	0.0228380203984985	NAN	0.157235464948273	NAN	0.731205110900452

---

## 2.5 其它应用

很多问题都可转化为优化问题，1stOpt 因此还可用于解决许多其它问题。

### 2.5.1 隐函数作图

主要使用关键字：

PlotFunction：定义作图函数方程，也可简写为“PlotFunc”；

Parameter：定义参数及其范围。

#### ✧ 二维隐函数作图

已知方程如下试画出  $x$  与  $y$  的关系图

$$\ln(3.5 \cdot x^y) + x - y^x - (\sin(y - x))^2 + 0.6 - \frac{(x + y)^{(0.1 \cdot x)}}{x} = 0 \quad (2-101)$$

其中， $x \in [1.5, 10]$ 。

“StepX”用于设定计算步长数。

1stOpt 代码(二维)

```
Parameter 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;
```

#### ✧ 三维隐函数作图：

已知方程如下试画出  $z$  与  $x$  和  $y$  的关系图

$$\ln(z) + \sin(x + y - z)^2 = (x - 10 - z)^3 + \cos(z) * (y - 100)^2 \quad (2-102)$$

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

“Mesh”用于设定空间计算密度。

1stOpt 代码

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

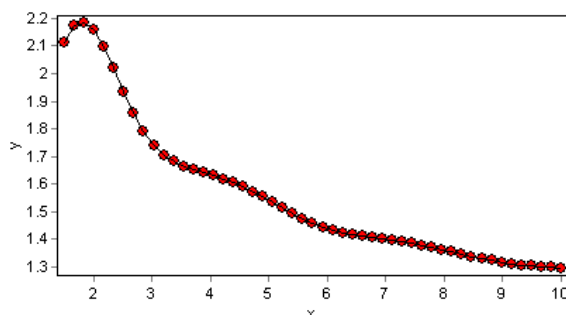


图 2-71 二维隐函数图

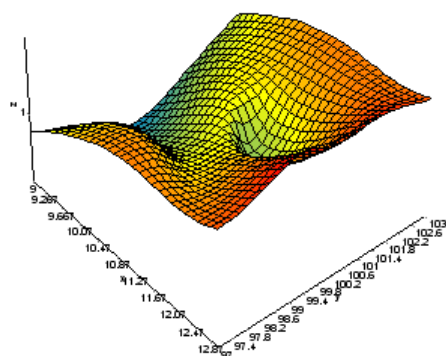


图 2-72 三维隐函数作图结果

✧ 三维隐函数作图：

已知方程如下试画出  $z$  与  $x$  和  $y$  的关系图

$$z = -\frac{1}{x^2 + 1} + \frac{2}{z + y^2 + 1} + \frac{0.5 \cdot \sin(5 \cdot r)}{r} \tag{2-103}$$

其中， $x \in [-5,5], y \in [-5,5]$ 。

```
1st0pt 代码
ConstStr r = sqrt(x^2+y^2+z);
Parameters x[-5,5], y[-5,5],z;
Mesh = [50,50];
PlotFunction  z=-1/(x*x+1)  +  2/(z+y*y+1)  +
0.5*sin(5*r)/r;
```

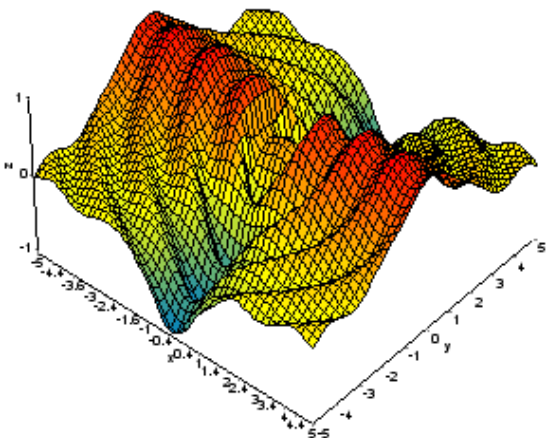
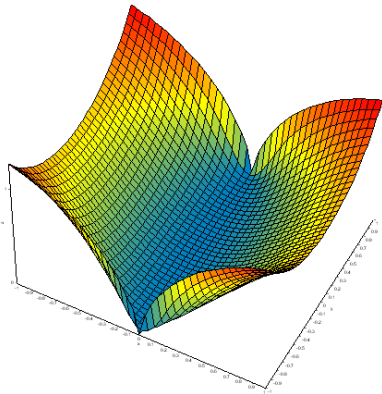
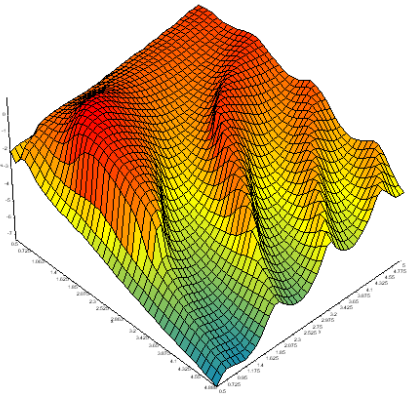
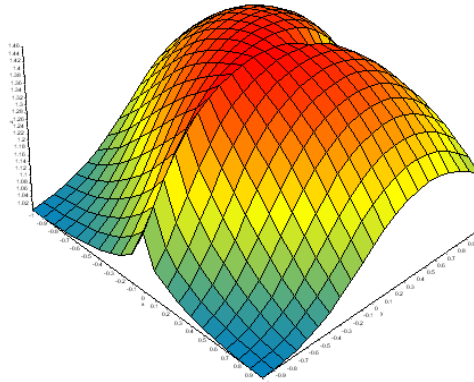
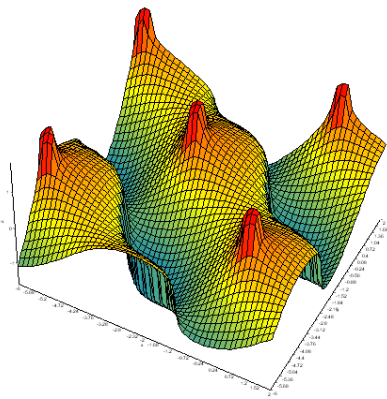
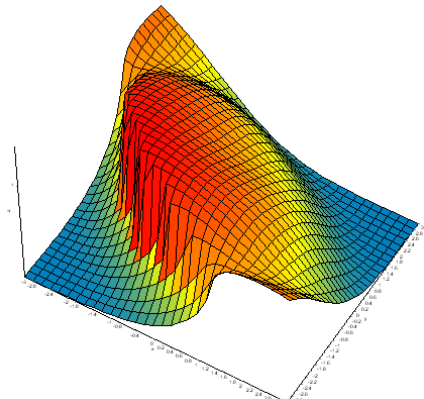


图 2-73 三维隐函数图

表 2-61 三维隐函数实例：

实例 1	方程	$\frac{\cos(z - y^2)}{\exp\left(\left(\sqrt{ x } - y\right)^2 + x^2 + y^2 - z\right)} - (z - 1) = 0 \quad , \quad x \in [-1,1], y \in [-1.5,1.5]$	
	代 码 及 图形	Parameters x[-1,1], y[-1.5,1.5], z; Mesh = [50,50]; PlotFunc cos(z-y^2)/exp((sqrt(abs(x))-y)^2+x^2+y^2-z)-(z-1)=0;	
实例 2	方程	$\cos(z) \cdot x^2 + \sin(z) \cdot y^2 - z = 0 \quad , \quad x \in [-1,1], y \in [-1,1]$	

	代 码 及 图 形	Parameters x[-1,1],y[-1,1],z; Mesh = [20,20]; Plotfunction cos(z)*x^2+sin(z)*y^2-z;	
实例 3	方程	$4\cos(x \cdot y + \sin^2(x^z)) + \sinh(x - y + z) + \frac{x}{y} + 2.5 - 2 \cdot x^{\sin(x)} + \frac{y^2}{x} + \sin(z) = 0$ $x \in [0.5, 5], y \in [0.5, 5]$	
	代 码 及 图 形	Parameters x[0.5,5], y[0.5,5], z; Mesh = [40,40]; Plotfunction 4*cos(x*y+sin(x^z)^2)+ sinh(x-y+z)+x/y+2.5-2*x^sin(x)+y^2/x+sin(z);	
实例 4	方程	$\frac{\cos(z)}{\exp\left(\left(\sqrt{\text{abs}(x)} - y\right)^2 + x^2 + y^2 - z\right)} - (z - 1) = 0, \quad x \in [-1, 1], y \in [-1, 1]$	
	代 码 及 图 形	Parameters x[-1,1], y[-1,1], z; PlotFunc cos(z)/exp((sqrt(abs(x))-y)^2+x^2+y^2-z)-(z-1)=0;	
实例 5	方程	$z = \sin(z - x) \cos(z - y) - \cos(x + y), \quad x \in [-6, 2], y \in [-6, 2]$	

	代 码 及 图 形	Parameters x[-6,2],y[-6,2],z; Mesh = [40,40]; PlotFunc z=sin(z-x)*cos(z-y)-cos(x+y);	
实例 6	方 程	$0.5 + \frac{\left(\sin\left(\sqrt{x^2 + z + y^2}\right)\right)^2 - 0.5 \cdot z \cdot \cosh(x + y + z^3)}{(1 + 0.001 \cdot (x^2 + y^2 + z^2))} - z \cdot \ln(z) = 0$ $x \in [-3,3], y \in [-3,3]$	
	代 码 及 图 形	Parameters x[-3,3], y[-3,3], z; Mesh = [30,30]; PlotFunc 0.5+(sqr(sin(sqrt(x^2+z+y^2)))- 0.5*z*cosh(x+y+z^3))/sqr(1+0.001*(x^2+y^2+z ^2))-z*ln(z)=0;	

## 2.5.2 参数函数作图

主要使用关键字：

PlotParaFunction：定义参数函数方程

例 1. 参数方程：

$$\begin{cases} x = \cos(t) \cdot \left( \exp(\cos(t)) - 2 \cdot \cos(4 \cdot t) - \sin\left(\frac{t}{12}\right)^5 \right) \\ y = \sin(t) \cdot \left( \exp(\cos(t)) - 2 \cdot \cos(4 \cdot t) - \sin\left(\frac{t}{12}\right)^5 \right) \end{cases} \quad (2-104)$$

其中  $t = i \cdot (u + 20 \cdot i)$ ， $u$  范围  $= [0, 3 \cdot \pi]$ ， $i=1..5$ 。

有 5 组参数方程 ( $i=1..5$ )，下面代码中使用“For”语句来实现。

1st0pt 代码：

```
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));
```



例 2：隐式参数方程：

$$\begin{cases} x = \sin(t - x^2) \\ y = \cos(2 \cdot t + 0.03 + y) - y \cdot x^2 \end{cases}$$

(2-105)

其中  $t$  范围  $= [0, 2\pi]$ 。

因为参数方程两端均含有因变项  $x$  和  $y$ ，构成了隐式参数方程。

1stOpt 代码

Parameters t[0,2\*pi],x,y;  
StepX = 100;  
PlotParaFunction x=sin(t-x^2), y=cos(t\*2+0.03+y)-y\*x^2;

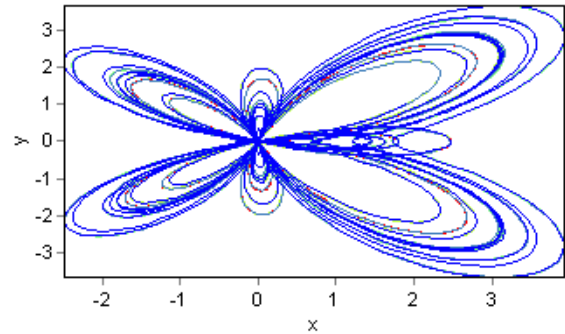


图 2-74 参数函数作图结果

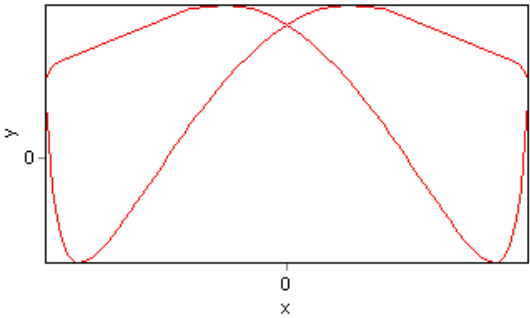
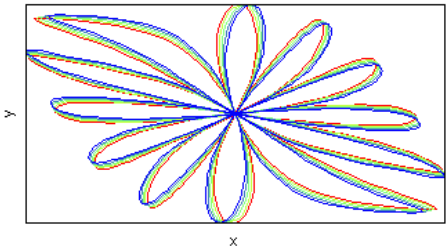
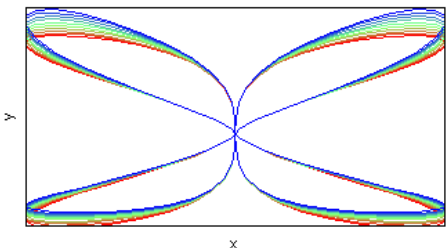
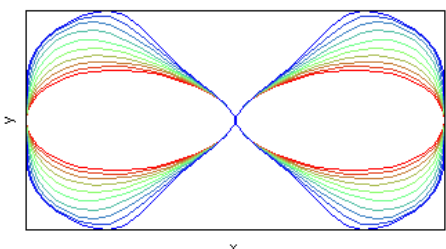


图 2-75 隐式参数方程作图结果

表 2-62 参数函数作图实例：

实例 1	方程	$\begin{cases} x = \sin^3(t) \\ y = \cos(t - y \cdot x \cdot i \cdot 0.1) \cdot \sin(t + y \cdot x \cdot 0.1 \cdot i) \end{cases}, t \in [0, 2\pi], i = 1 \dots 10$	
	代码及图形	Parameters t[0,2*pi],x,y; StepX = 100; PlotParaFunction for(i=1:10)(x=sin(t)^3, y=cos(t-y*x*i*0.1)*sin(t+y*x*0.1*i));	
实例 2	方程	$\begin{cases} x = \cos^2(t) \\ y = \cos(t - 0.2 \cdot i \cdot x \cdot y) \cdot \sin(x) \end{cases}, t \in [0, 2\pi], i = 1 \dots 10$	
	代码及图形	Parameter t[0,2*pi],x,y; PlotParaFunction for(i=1:10)(x=cos(t)^2, y=cos(t-0.2*i*x*y)*sin(x));	
实例 3	方程	$\begin{cases} x = r \cdot \cos(t + x^2 \cdot 0.01) \\ y = r \cdot \sin(t) \end{cases}, r = a^2 \cdot \cos(n \cdot t + i \cdot 0.1), t \in [-\pi, \pi], i = 1 \dots 5$	

	代 码 及 图 形	Constant a=3,n=6; ConstStr r = a^2*cos(n*t+i*0.1); Parameters t[-pi,pi],x,y; StepX = 440; PlotParaFunction for(i=1:5)(x=r*cos(t+x^2*0.01),y=r*sin(t));	
实 例 4	方 程	$\begin{cases} x = \sin^3(t) \cdot \cos^3(t) \\ y = \cos(t - y \cdot x \cdot i \cdot 0.3) \cdot \sin^2(t + y \cdot x \cdot 0.1 \cdot i) \end{cases}, t \in [0, 2\pi], i=1..10$	
	代 码 及 图 形	Parameters t[0,2*pi],x,y; StepX = 100; PlotParaFunction for(i=1:10)(x=sin(t)^3*cos(t)^3, y=cos(t-y*x*i*0.3)*sin(t+y*x*0.1*i)^2);	
实 例 5	方 程	$\begin{cases} x = \sin^3(t) \\ y = \cos(t - y \cdot x \cdot i \cdot 0.1) \cdot \sin^2(t + y \cdot x \cdot 0.1 \cdot i) \end{cases}, t \in [0, 2\pi], i=1..10$	
	代 码 及 图 形	Parameters t[0,2*pi],x,y; StepX = 100; PlotParaFunction for(i=1:10)(x=sin(t)^3, y=cos(t-y*x*i*0.1)*sin(t+y*x*0.1*i)^2);	

### 2.5.3 数据作图

主要使用关键字：

- PlotData: 画二维点线数据图；
- PlotPoint2D: 画二维点数据图；
- PlotPoint3D: 画三维点数据图，数据格式为 x-y-z；
- PlotMeshData: 画三维面数据图，矩阵数据格。

例 1: “PlotData” 的使用，数据均作为 y 轴数据，x 轴数据自动赋予。

1st0pt 代码

PlotData;
0.22 0.0511
0.55 0.0872
0.5420.1685
1.0210.1951
1.6330.2341
1.7720.3616
1.9730.4249
2.15 0.4597
2.21 0.4902
2.5420.5764

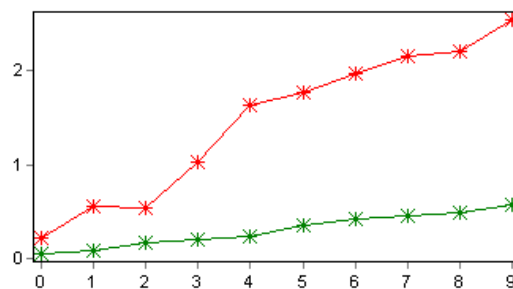


图 2-76 “PlotData” 数据图示例

例 2: “PlotPoint2D” 的使用, 第一列数据自动作为 x 轴数据, 其余均作为 y 轴数据。

lst0pt 代码

PlotPoint2D;
0.22 0.0511
0.55 0.0872
0.5420.1685
1.0210.1951
1.6330.2341
1.7720.3616
1.9730.4249
2.15 0.4597
2.21 0.4902
2.5420.5764

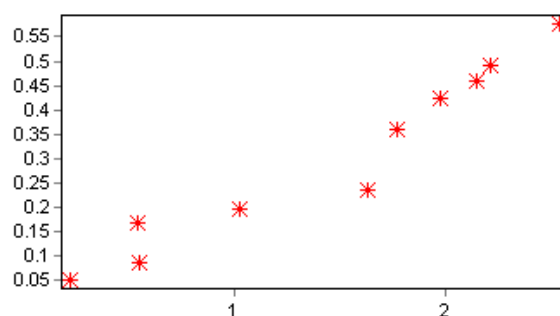


图 2-77 “PlotPoint2D” 数据图示例

例 3: “PlotPoint3D” 的使用, 第一、二列数据分别作为 x 和 y 轴数据, 第三列为 z 轴数据。

lst0pt 代码

PlotPoint3D;
x y z
0.0628 0.0002 0.0039
0.1257 0.0020 0.0156
0.2513 0.0154 0.0599
0.3142 0.0295 0.0910
0.4398 0.0772 0.1650
0.5027 0.1118 0.2053
0.6283 0.2031 0.2852
0.6912 0.2590 0.3216
0.8168 0.3874 0.3796
0.8796 0.4574 0.3983
1.0053 0.6019 0.4089
1.0681 0.6729 0.3991
1.1938 0.8038 0.3479
1.2566 0.8602 0.3071
1.3823 0.9478 0.2001
1.4451 0.9765 0.1368
1.5708 1.0000 0.0000
1.6336 0.9941 -0.0695
1.7593 0.9478 -0.2001
1.8221 0.9087 -0.2574
1.9478 0.8038 -0.3479
2.0106 0.7408 -0.3788
2.1363 0.6019 -0.4089
2.1991 0.5295 -0.4084
2.3248 0.3874 -0.3796
2.3876 0.3208 -0.3536

结果

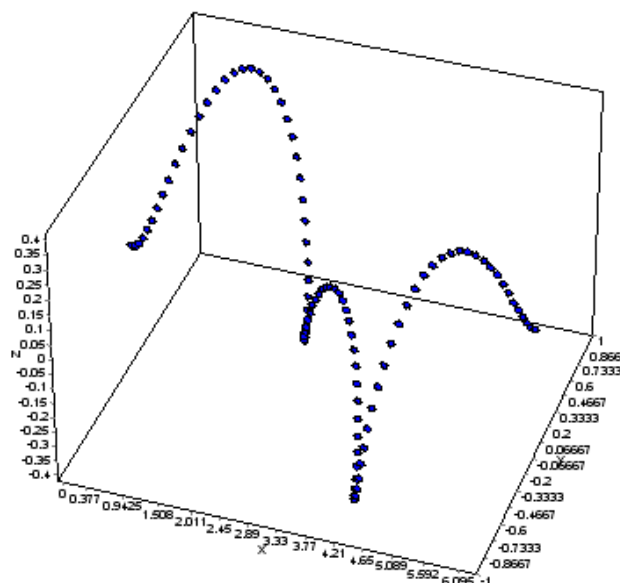


图 2-78 “PlotPoint3D” 数据图示例

2.5133	0.2031	-0.2852
2.5761	0.1538	-0.2459
2.7018	0.0772	-0.1650
2.7646	0.0499	-0.1264
2.8903	0.0154	-0.0599
2.9531	0.0066	-0.0345
3.0788	0.0002	-0.0039
3.1416	0.0000	0.0000
3.2673	-0.0020	-0.0156
3.3301	-0.0066	-0.0345
3.4558	-0.0295	-0.0910
3.5186	-0.0499	-0.1264
3.6442	-0.1118	-0.2053
3.7071	-0.1538	-0.2459
3.8327	-0.2590	-0.3216
3.8956	-0.3208	-0.3536
4.0212	-0.4574	-0.3983
4.0841	-0.5295	-0.4084
4.2097	-0.6729	-0.3991
4.2726	-0.7408	-0.3788
4.3982	-0.8602	-0.3071
4.4611	-0.9087	-0.2574
4.5867	-0.9765	-0.1368
4.6496	-0.9941	-0.0695
4.7752	-0.9941	0.0695
4.8381	-0.9765	0.1368
4.9637	-0.9087	0.2574
5.0265	-0.8602	0.3071
5.1522	-0.7408	0.3788
5.2150	-0.6729	0.3991
5.3407	-0.5295	0.4084
5.4035	-0.4574	0.3983
5.5292	-0.3208	0.3536
5.5920	-0.2590	0.3216
5.7177	-0.1538	0.2459
5.7805	-0.1118	0.2053
5.9062	-0.0499	0.1264
5.9690	-0.0295	0.0910
6.0947	-0.0066	0.0345
6.1575	-0.0020	0.0156
6.2832	0.0000	0.0000

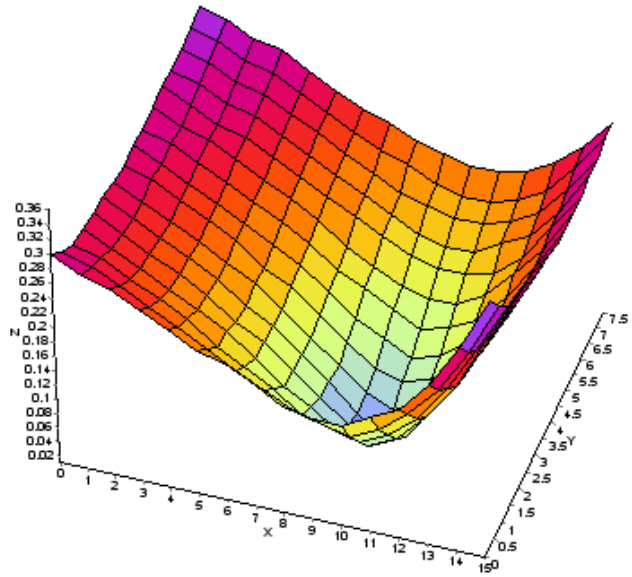


图 2-79 “PlotMeshData” 数据图示例

例 3: “PlotMeshData” 的使用, 从矩阵数据绘制三维面图。

1stOpt 代码:

```
PlotMeshData;
0.36,0.33,0.31,0.30,0.29,0.28,0.27,0.27,0.26,0.26,0.26,0.26,0.27,0.27,0.28,0.28
0.29,0.26,0.24,0.23,0.22,0.21,0.20,0.20,0.20,0.20,0.20,0.21,0.21,0.22,0.23,0.24
0.23,0.20,0.18,0.16,0.15,0.15,0.14,0.14,0.14,0.15,0.16,0.16,0.17,0.18,0.19,0.21
0.19,0.16,0.13,0.11,0.10,0.09,0.09,0.09,0.10,0.11,0.12,0.13,0.15,0.16,0.17,0.19
0.17,0.13,0.10,0.07,0.05,0.05,0.05,0.06,0.07,0.09,0.10,0.12,0.13,0.15,0.16,0.18
0.15,0.11,0.08,0.05,0.03,0.01,0.03,0.05,0.06,0.08,0.10,0.12,0.13,0.15,0.16,0.18
0.15,0.12,0.09,0.06,0.05,0.04,0.05,0.06,0.08,0.10,0.11,0.13,0.14,0.16,0.17,0.19
0.16,0.13,0.10,0.09,0.08,0.08,0.08,0.09,0.10,0.12,0.13,0.15,0.16,0.17,0.19,0.20
0.18,0.15,0.13,0.11,0.11,0.11,0.11,0.12,0.13,0.14,0.15,0.17,0.18,0.19,0.20,0.22
0.20,0.17,0.15,0.14,0.14,0.14,0.14,0.15,0.16,0.17,0.18,0.19,0.20,0.21,0.22,0.23
0.21,0.19,0.17,0.17,0.16,0.16,0.17,0.17,0.18,0.19,0.20,0.21,0.22,0.23,0.24,0.25
0.23,0.21,0.20,0.19,0.19,0.19,0.19,0.20,0.20,0.21,0.22,0.23,0.24,0.25,0.26,0.27
```

0.25,0.23,0.22,0.21,0.21,0.21,0.21,0.22,0.22,0.23,0.24,0.25,0.26,0.27,0.28,0.29
0.27,0.25,0.24,0.23,0.23,0.23,0.23,0.24,0.24,0.25,0.26,0.26,0.27,0.28,0.29,0.30
0.28,0.27,0.26,0.25,0.25,0.25,0.25,0.26,0.26,0.27,0.27,0.28,0.29,0.30,0.31,0.32
0.30,0.28,0.27,0.27,0.27,0.27,0.27,0.27,0.28,0.28,0.29,0.30,0.30,0.31,0.32,0.33

## 2.5.4 作为高级计算器使用

可直观用于计算任意表达式之值。支持特殊计算符号如求和 ( $\Sigma$ )，求积 ( $\Pi$ )，积分，同时也支持一些特殊函数如伽玛函数(Gamma Function)，贝塞尔函数等(Bessel Function)。

✧ 一般运算

试求下列各式之值

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

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

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

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

lstOpt 代码

```
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=1:10)(f3^((i+j+m+n+p)/1000))))));
```

结果

f1 = 1353.98290661822	f3 = 6.97723694350016
f2 = 48.6818353657435	f4 = 28.297511489

✧ 复数计算：复数计算时用关键字 “Complex()”，“i” 为缺省的虚数符号。

试求下列复数之值：

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

$$2: f_2 = f_1 + \sin(5.6 + i)$$

lstOpt 代码

```
f1 = Complex(5*sin(pi*6)^2*i+exp(6.45+ln(2.14)*i));
f2 = Complex(f1+cos(5.6+i));
```

结果

```
f1 = 458.254010532336+436.251593877789*i
f2 = 459.450771220489+436.993459184124*i
```

✧ 求函数导数

已知函数：  $y = x^2 + \exp(x + 2) \cdot \sin(x)$ ，试求

- 一阶导数  $y' = \frac{dy}{dx}$  的表达式;
- $x = 0.5$  时的  $y'$  值
- $y$  的三阶导数表达式
- $x = 0.5$  时的  $y'''$  值

1stOpt 代码

```
Conststr y=x^2+exp(x+2)*sin(x);
Diff(y,x);
Diff(y,x=0.5);
Diff(y,x,3);
Diff(y,x=0.5,3);
```

结果

```
diff(y,x) = 2*x+(exp(x+2))*sin(x)+exp(x+2)*(cos(x))
diff(y,x=0.5) = 2*x+(exp(x+2))*sin(x)+exp(x+2)*(cos(x)) = 17.53174299
diff(y,x,3) =
(exp(x+2))*sin(x)+exp(x+2)*(cos(x))+(exp(x+2))*cos(x)+exp(x+2)*((-sin(x)))+(exp(x+2))*
cos(x)+exp(x+2)*((-sin(x)))+(exp(x+2))*((-sin(x)))+exp(x+2)*((-cos(x))))
diff(y,x=0.5,3) =
(exp(x+2))*sin(x)+exp(x+2)*(cos(x))+(exp(x+2))*cos(x)+exp(x+2)*((-sin(x)))+(exp(x+
2))*cos(x)+exp(x+2)*((-sin(x)))+(exp(x+2))*((-sin(x)))+exp(x+2)*((-cos(x)))) = 9.701091063
```

## 2.5.5 使用脚本语言

1stOpt 支持 Basic 和 Pascal 两种脚本语言。脚本语言可直接控制代码本、代码本电子表格及 1stOpt 电子表格。

✧ 控制电子表格：如下图，欲将三项之和放入 D 列

	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							

图 2-80 电子表格数据

1stOpt 代码

```
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;

```

运行结果

	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							

图 2-81 脚本语言控制电子表格结果

✧ 控制代码本电子表格: 与上面控制电子表格基本一致, 仅名称不同, 这里使用 Basic 语言

```

1 StartScript [Basic];
2 Sub MainModel
3   with CodeSheet1
4     for i = 0 to 14
5       Doubles[3,i] = Doubles[0,i] + Doubles[1,i] + Doubles[2,i]
6     next
7   end with
8 End Sub
9 EndScript;

```

The resulting spreadsheet (CodeSheet1) is as follows:

	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							

图 2-82 运行前画面

1stOpt 代码

```

StartScript [Basic];
Sub MainModel
  with CodeSheet1

```

```

for i = 0 to 14
    Doubles[3,i] = Doubles[0,i] + Doubles[1,i] + Doubles[2,i]
next
end with
End Sub

```

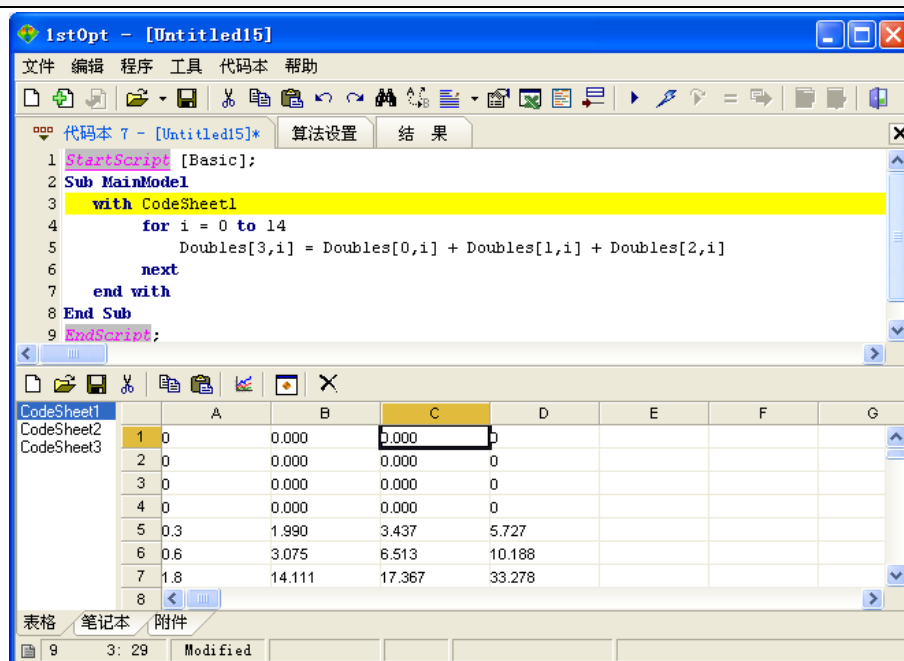


图 2-83 运行后画面

✧ 控制代码本：代码本中增加内容

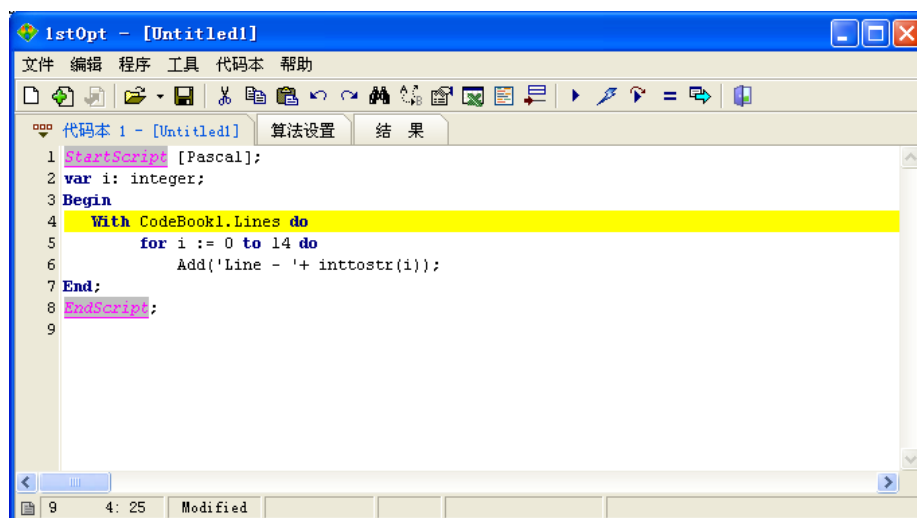


图 2-84 运行前画面

1stOpt 代码

```

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

```



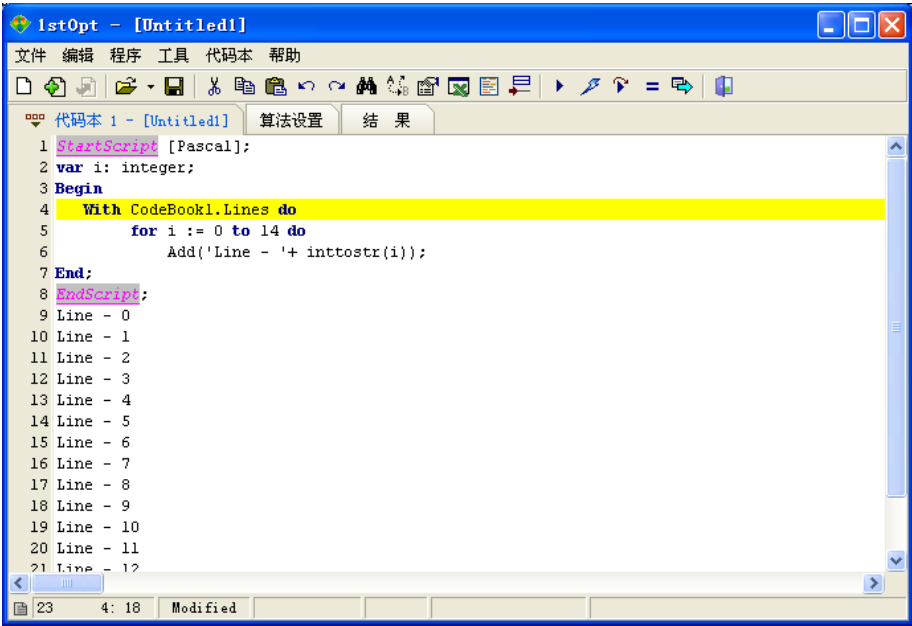


图 2-85 运行后画面

### 2.5.6 关键字 “PassParameter” 的使用

“PassParameter”可理解为“传递参数”，可传递返回与求解参数有关的数值。

例 约束函数优化问题：PassParameter 在优化的同时，将给出  $x \cdot y$  的值

1stOpt 代码

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

结果

迭代数: 167  
计算用时(时:分:秒:毫秒): 00:00:00:78  
计算中止原因: 达到收敛判定标准  
优化算法: 标准简面体爬山法 + 通用全局优化法  
函数表达式: 9-x-y  
目标函数值(最小): 0  
x: 7  
y: 2

传递参数(PassParameter):  
x\*y: 14

约束函数:  
1:  $(x-3)^2+(y-2)^2-(16) = 0$   
2:  $x*y-(14) = 0$

例 非线性约束拟合

表 2-63 拟合数据

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

拟合公式:  $y = a \cdot (x - d)^4 + b \cdot (x - d)^2 + c$  (2-106)

其中: x、y: 自变量和因变量; a、b、c、d: 待求参数。

约束条件:

- 1) 参数 a 必须小于 0;

2) 计算  $y$  与实际  $Y$  的最大差值的绝对值不大于 0.25, 即:

$$\text{Max}(\text{abs}(y_i - y'_i)) \leq 0.25, \quad i = 1..11 \quad (2-107)$$

1stOpt 代码

```
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;
```

结果比较

不加约束 (去除 ConstrainedResult := PA <= 0.25;)	有约束
迭代数: 283 计算用时(时:分:秒:毫秒): 00:00:00:516 计算中止原因: 达到收敛判定标准 优化算法: 标准简面体爬山法 + 通用全局优化法 目标函数值(最小): 0.309952381552963 a: -6877.78414257241 b: 385.898376776761 c: 18.4260680489247 d: -0.00393418836172759  传递参数(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	迭代数: 814 计算用时(时:分:秒:毫秒): 00:00:01:219 计算中止原因: 达到收敛判定标准 优化算法: 标准简面体爬山法 + 通用全局优化法 目标函数值(最小): 0.325354789681984 a: -13521.8798280647 b: 436.931945824039 c: 18.3681739539736 d: -0.00344580812519686  传递参数(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

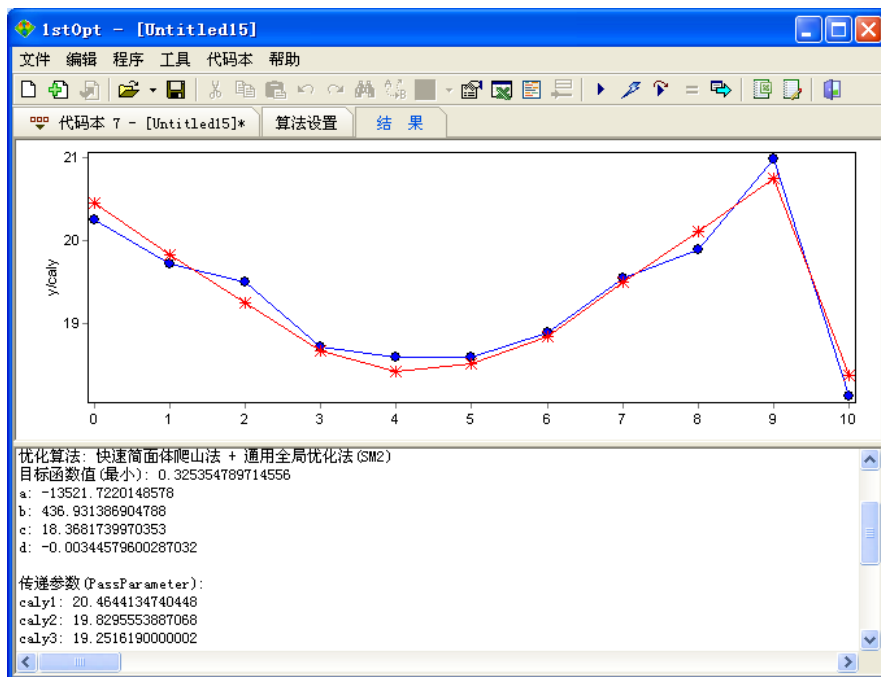


图 2-86 有无约束拟合对比图

## 2.5.7 关键字“SubDivision”的使用

“SubDivision”将能使本问题代码块直接使用主模块的参数、常数和目标函数值等。

优化问题 1:

$$\min 9 - x_1 - y_1 \quad (2-108)$$

$$\text{s. t. } \begin{cases} (x_1 - 3)^2 + (y_1 - 2)^2 \leq 16 \\ x_1 \cdot y_1 \leq 14 \end{cases} \quad x_1, y_1 \in [0, 100]$$

优化问题 2:

$$\min -x_1 \exp\left(-y_1 \cdot \sqrt{\frac{1}{5} \cdot \sum_{i=1}^5 p_i^2}\right) - \exp\left(\frac{1}{5} \cdot \sum_{i=1}^5 \cos(6 \cdot p_i)\right) + x_1 + \exp(\text{ObjFunction}) \quad (2-109)$$

$$\text{s. t. } p_i \in [-32.768, 32.768], i = 1..5$$

优化问题 2 中要使用到优化问题 1 中求得的  $x_1$  和  $y_1$  值以及目标函数值 (ObjFunction)。一般情况下需先求解问题 1，得出结果后再代入问题 2 进一步求解，在 1stOpt 中可以借助“SubDivision”一步完成。

1stOpt 代码

```
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);	
结果	
迭代数: 37 计算用时(时:分:秒:毫秒): 00:00:00:375 计算中止原因: 达到收敛判定标准 优化算法: 标准差分进化算法 函数表达式: 9-x1-y1 目标函数值(最小): 0 x1: 7 y1: 2  约束函数: 1: (x1-3)^2+(y1-2)^2-(16) = 0 2: x1*y1-(14) = 0  ===== 计算结束 =====	迭代数: 361 计算用时(时:分:秒:毫秒): 00:00:01:781 计算中止原因: 达到收敛判定标准 优化算法: 标准差分进化算法 函数表达式:  -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) 目标函数值(最小): -1.71828182845904 p1: -1.23257171502356E-16 p2: 7.40257379257929E-17 p3: -4.29348910045455E-17 p4: -3.95583172994915E-17 p5: -7.79891019495037E-17

## 2.5.8 关键字“PenaltyFactor”的使用

“PenaltyFactor”是罚函数系数，用于处理约束优化问题，其缺省值为 1E+8，但有时该数值却并非最理想，此时可通过“PenaltyFactor”来进行调整。

例 1:

$$\min \cos(x) - \exp((x - 0.5) \cdot y) \tag{2-110}$$

$$\text{s. t. } x^2 + y^{2 \cdot x} \leq 1$$

其中， $x, y \in [-10, 10]$ 。

1stOpt 代码一

ParameterDomain = [-10,10]; Algorithm = UGO1[100]; MinFunction cos(x)-exp((x-0.5)*y); x^2+y^(2*x)<=1;
--

1stOpt 代码二

ParameterDomain = [-10,10]; PenaltyFactor = 1E+30; Algorithm = UGO1[100]; MinFunction cos(x)-exp((x-0.5)*y); x^2+y^(2*x)<=1;
--

运行上述代码一，很难获得最优解，此时的罚函数系数为缺省的 1E+8，如果将该系数改为 1E+30，代码如右，则可容易得到最优解：

目标函数值(最小): -147.413159102577 x: 0 y: -10	约束函数: 1: x^2+y^(2*x)-(1) = 0
--	---------------------------------

## 2.6 1stOpt 的编程模式

1stOpt 的快捷模式直观、简单、明了、易于掌握，可以解决大部分优化问题，但对于一些复杂的问题，如目标函数或约束函数无法用简单的表达式来表述计算，而是必须通过复

杂的逻辑判断、循环运算等来实现，快捷模式无法满足，此时可用 1stOpt 的编程模式来解决。

1stOpt 直接支持 Basic 和 Pascal 两种语言。从理论上来说，编程模式可以处理解决全部快捷模式下的问题。

编程模式的主要关键字：

- StartProgram：定义编程模式的起始行  
“StartProgram [Basic]”表示用 Basic 语言  
“StartProgram [Pascal]”或“StartProgram”表示用 Pascal 语言
- EndProgram：定义编程模式的终止行
- 在“StartProgram”和“EndProgram”间按标准的 Delphi/Pascal 或 Basic 语言编写。
- ObjectiveResult：定义目标函数，仅可有一次。
- ConstrainedResult：定义约束函数，可有多，约束之间可用“and”相连。

下面给出两个实例演示如何使用编程模式。

## 2.6.1 约束函数优化问题

已知如下优化问题，使用快捷方式和编程模式分别实现。

$$\min 10 \cdot x_1 + 9 \cdot x_2 + 8 \cdot x_3 + 7 \cdot x_4 \cdot \sin(x_1 + x_2 + x_3) \quad (2-111)$$

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

参数范围均在  $[-100, 100]$  之间。

1stOpt 快捷模式代码

```
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;
```

1stOpt 编程模式 Basic 代码

```
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)and(x3+x4>=30)and(3*x1+2*x3<=120)
End Sub
EndProgram;
```

1stOpt 编程模式 Pascal 代码

```

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)and(x3+x4>=30)and(3*x1+2*x3<=120);
End;
EndProgram;

```

上面三段代码均可获得相同的结果：Min. = -1589.02777245744, X = [-98.78263, 68.78263, -65.84115, 99.09896]。

## 2.6.2 时系列模型参数优化率定

此案例研究的是大肠杆菌(z)与降雨量(x)及流量(y)间的关系。表 3.1 是三次实测过程数据，共有 9 次类似数据。因为大肠杆菌不仅与降雨量及流量有关，还受其初始值影响，采用的数学模型公式如下式，其中，p1 至 p7 为待定参数，t 表示时间。

$$z_t = \frac{p_1 + p_2 \cdot x_t + p_3 \cdot y_t + p_4 \cdot z_{t-1}}{1 + p_5 \cdot x_t + p_6 \cdot y_t + p_7 \cdot z_{t-1}} \quad (2-112)$$

表 2-64 原始数据例

降雨 x	3, 4.5, 5.5, 0.5, 0.5, 1, 0.5, 1.5, 2.5, 0, 0, 0
径流 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
大肠杆菌 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

在此例中，每一数据文件的第一行数值被当作初始值，使用关键词” VarConstant” 用以描述各次过程的初始大肠杆菌量。

1st0pt 代码(Pascal)

```

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

1stOpt 代码(Basic)

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[Basic];
Sub mainmodel
Dim i as integer
Dim Temz as Double
Temz = z0
for i = 0 to Datalength - 1
z[i] = (p1+p2*x[i]+p3*y[i]+p4*Temz)/(1+p5*x[i]+p6*y[i]+p7*Temz)
Temz = z[i]
next
End Sub
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
```

详细代码和数据可参考 1stOpt 附带实例 “Examples\Auto Calibration\Time Series.mff” .

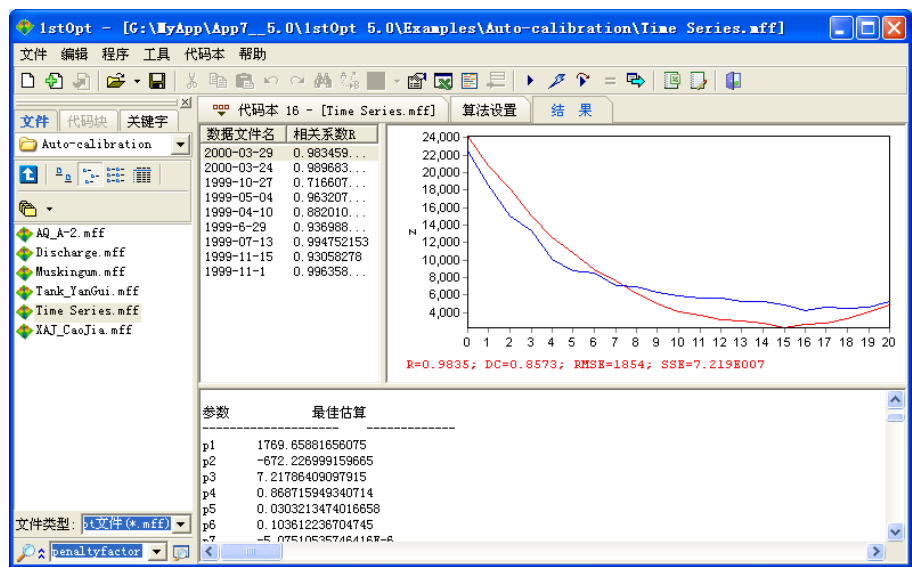


图 2-87 时系列模型优化结果示意图

## 2.7 1stOpt 调用外部程序

1stOpt 已经有了快捷和编程两种模式，可以处理绝大多数优化问题。然而对于一些非常复杂的工程优化问题，即使编程模式也无法满足要求，或这些优化问题已经由其它程序语言编写，较难改写成 1stOpt 编程模式支持的两种语言 Basic 和 Pascal，这时就需要 1stOpt 具有混合编程的能力，即 1stOpt 提供优化算法而外部程序则提供目标函数和约束函数的计算。自 3.0 版本起，1stOpt 已具有完全混合编程能力，用户可用高级语言如 C++、Fortran、Delphi/Pascal、Basic 或其它任何支持 Window 标准动态库(.dll)或命令行执行文件(.exe)的编程语言编写任意复杂的优化问题，然后由 1stOpt 调用。

1stOpt 主要通过两种方式调用外部编译的程序，一是通过动态库 (.dll) 文件，二是通过命令行执行文件 (.exe)。在效率上第一种调用方式要远远高于第二种方式，因此在万不得已的情况下，推荐使用动态库调用方式。

目前可使用的外部编译语言有很多，用户可根据自己的喜好特长选择相应的高级语言。下面详细介绍如何用 Visual C++、Borland C++、Visual Fortran、Gun Fortran、Delphi、PowerBasic 和 Free Basic 进行外部程序编写并由 1stOpt 调用求解。Visual Basic (VB) 虽然很流行，但由于其是解释性语言而非真正的编译语言，不论是在计算速度还是在可移植方面都存在不足，因此不在推荐之列。

外部调用模式之最大优点是可以充分发挥高级编程语言的能力，处理任何复杂的优化计算问题。

### 2.7.1 调用格式及关键字

不论是 Dll 格式还是 Exe 格式，其基本路线都是由外部程序提供计算目标函数和约束函数的功能，而 1stOpt 则通过调用这些计算结果进行参数优化。图 4.1 及图 4.2 展示了 1stOpt 与外部程序互动的流程图。

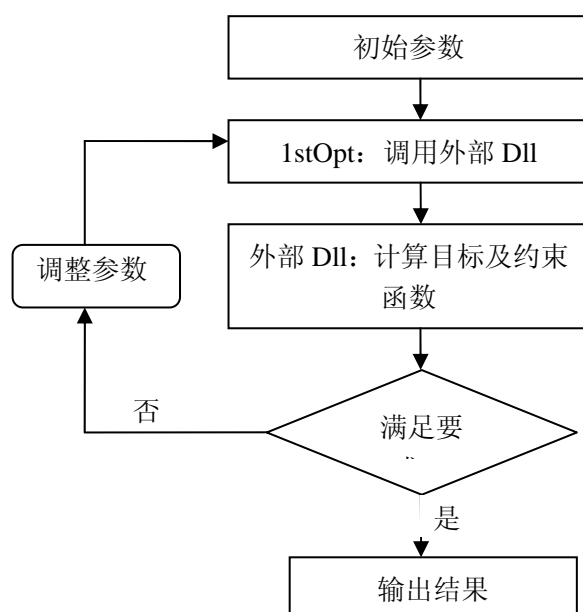




图 2-88 1stOpt 调用动态链接库流程图

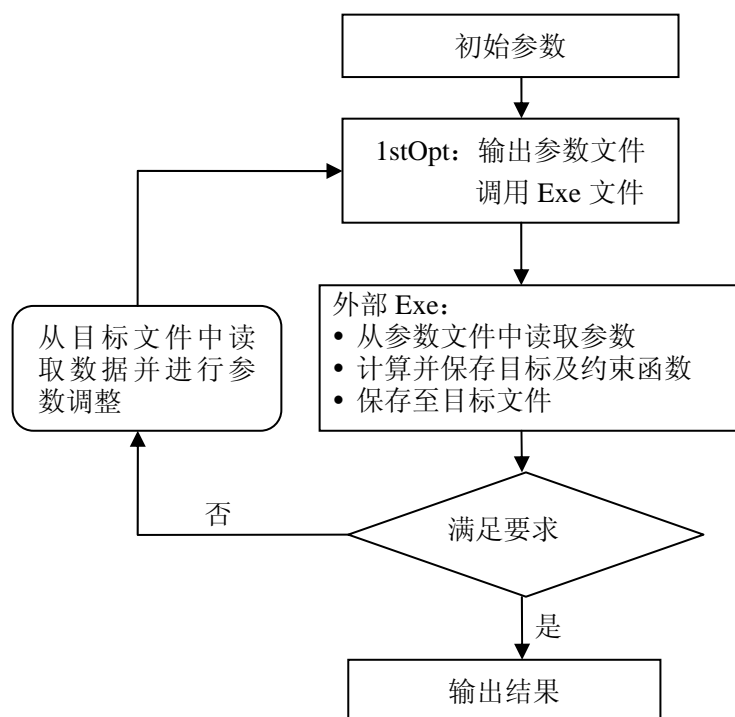


图 2-89 1stOpt 调用命令行执行文件流程图

#### ✧ 4.1.1 调用外部 Dll 文件

如 Dll 文件为 “c:\test\dll\_test.dll”，则调用方式如下：

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

其中“dllfunction”为缺省的输出函数名，可省略，N1、N2 为不等式和等式约束数目。

不等式函数形式必须写成小于等于 0 的形式。

所使用的关键字有：Parameter, PassParameter 和 MinFunction/MaxFunction

一些例子：

- 1) 如输出函数名为缺省的“dllfunction”，不等式 ( $\leq 0$ ) 约束为 2 个，等式约束为 0 个，则调用格式如下：

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

- 2) 如输出函数名为“myfunction”，不等式 ( $\leq 0$ ) 约束为 0 个，等式约束为 2 个，则调用格式如下：

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

- 3) 如输出函数名为缺省的“dllfunction”，不等式 ( $\leq 0$ ) 及等式约束均为 0 个，则调用格式如下：

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

- 4) 如输出函数名为缺省的“dllfunction”，不等式 ( $\leq 0$ ) 约束为 0 个，等式约束为 4 个，则调用格式如下：

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

- 5) 如输出函数名为自定义，如“myfunction”，不等式 ( $\leq 0$ ) 约束为 1 个，等式约束为 4 个，则调用格式如下：

MinFunction " c:\test\dll\_test.dll[myfunction, 1, 4]" ;

调用外部程序模式与 1stOpt 的编程模式的一个重要区别是：1stOpt 调用外部程序的代码中用“Constant”、“ConstStr”、“DataSet”等定义的常数或常字符串等对外部程序来说是不可见或不可用的，也即无法自动传递到外部程序；而在编程模式下却是完全可视和通用的。

#### ✧ 4.1.2 调用外部 Exe 文件

外部 Exe 文件必须是命令行 Exe 格式，从 1stOpt 中输出的参数文件格式及从 Exe 中输出的目标函数文件格式必须是文本格式；在参数文件中，每一行按顺序记录一个参数值，在目标函数文件中，第一行记录目标函数值，然后依次记录不等式约束及等式约束值。

如 Exe 文件为“c:\test\exe\_test.exe”，参数输出文件为“c:\test\exe\_par.txt”，目标函数输出文件为“c:\test\exe\_obj.txt”，不等式及等式约束分别为 3 和 2，则调用方式如下：

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

一些例子：

- 1) 不等式约束为 3 个，等式约束为 0 个：MinFunction " c:\test\exe\_test.exe[3]" ;
- 2) 不等式约束为 0 个，等式约束为 2 个：MinFunction " c:\test\exe\_test.exe[0, 2]" ;
- 3) 不等式及等式约束均为 0 个：MinFunction " c:\test\exe\_test.exe" ;

## 2.7.2 C++编译目标 Dll 文件

输出函数格式如下

```
extern "C" void __declspec(dllexport) __stdcall
dllfunction(double *para, double *objfun, double *confun1, double *confun2, double *passpara)
{
    //加入目标及约束函数计算代码
}
```

其中：

- dllfunction: 缺省输出函数名，可自己命名
- Para: 从 1stOpt 传入的参数数组，下标从 0 开始
- ConFun1, ConFun2: 不等式及等式约束数组，下标从 0 开始
- ObjFun: 目标函数值
- PassPara: 返回传递参数数组，下标从 0 开始

例 1. 函数优化问题

$$\begin{aligned} \min \quad & 9-x-y & (2-113) \\ \text{s.t.} \quad & \begin{cases} (x-3)^2 + (y-2)^2 \leq 16 \\ x \cdot y \leq 14 \end{cases} \end{aligned}$$

其中 x、y 范围均在 [0, 100] 之间。

步骤：

- 1) 编写 C++源代码如下

```
#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) 保存文件，如 “c:\projects\cplus\_test1.cpp”
- 3) 用 Borland C++5.5 编译器编译成动态库文件，如 C++编译器位于“d:\Borland C\bin”，则编译命令如下：  
“d:\Borland C\bin\bcc32” -tWM -tWD “c:\projects\cplus\_test1.cpp”
- 4) 执行上述命令，在 “d:\Borland C\bin\” 目录下可生成动态库文件 “cplus\_test1.dll”
- 5) 1stOpt 调用代码

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

- 6) 1stOpt 快捷代码

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

执行上述 5、6 步，均可得到相同结果：Min. = 0, x = 7, y = 2

## 2.7.3 Visual Fortran（VF）编译目标 Dll 文件

Visual Fortran 是使用非常广泛的 Fortran 编译器。VF 命令行编译器执行文件名为 “DF. exe”，如要将 VF 源文件 “c:\projects\VF\_test1. f90” 编译成动态库文件，其命令如下：

DF. exe /dll “c:\projects\VF\_test1. f90”

输出函数格式定义如下

```
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
```

其中：

- dllfunction: 缺省输出函数名，可自己命名
- para: 从 1stOpt 传入的参数数组，下标从 0 开始
- objfun: 目标函数值
- confun1, confun2: 不等式及等式约束数组，下标从 0 开始
- passpara: 返回传递参数数组，下标从 0 开始

语句“real(fp) :: para(0:1), confun1(0:0), confun2(0:0), passpara(0:0)”要根据实际情况变动，如参数数为 3，不等式约束为 4，等式约束为 2，传递参数为 20，则上述语句要改为：

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

例 2. 约束优化问题

$$\min -x_1 \cdot x_2 - x_2 \cdot x_3 - x_3 \cdot x_1 \quad (2-114)$$

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

步骤

1) 编写 Fortran 源代码如下

```
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) 保存文件，如 “c:\projects\VFortran\_test1.f90”

3) 编译成动态库文件，如 VF 编译器位于 “C:\Program Files\Microsoft Visual Studio\DF98\BIN”，则编译命令如下：

C:\Program Files\Microsoft Visual Studio\DF98\BIN\DF.exe /dll

“c:\projects\VFortran\_test1.f90”

4) 执行上述命令，产生一动态库文件 “Vfortran\_test1.dll”

5) 1stOpt 中调用命令

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

6) 快捷模式代码

```
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;
```

两种模式均可得到相同的结果：

目标函数值(最小): -2.34512297572644

x1: 1.93394964369396

x2: 0.506937411738282

x3: 0.559112944567762

## 2.7.4 Gun Fortran 编译目标 Dll 文件

Gun Fortran(GFortran)是开源、免费、跨平台的 Fortran 编译器,支持 Fortran95/2003 标准,其官方网站在 <http://gcc.gnu.org/>。

GFortran 仅提供命令行编译器,详细命令可参考其使用指南和相关参考书。Gfortran 编译器执行文件名为“Gfortran.exe”,如要将 Fortran 源文件“c:\projects\GF\_test1.f90”编译成动态库文件,命令如下:

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

输出函数格式定义如下

```
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
```

其中:

- dllfunction: 缺省输出函数名,可自己命名
- para: 从 1stOpt 传入的参数数组,下标从 0 开始
- objfun: 目标函数值
- confun1, confun2: 不等式及等式约束数组,下标从 0 开始
- passpara: 返回传递参数数组,下标从 0 开始

语句“real(fp) :: para(0:1), confun1(0:0), confun2(0:0), passpara(0:0)”要根据实际情况变动,如参数数为 3,不等式约束为 4,等式约束为 2,传递参数为 20,则上述语句要改为:

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

例 3. 无约束优化问题

$$\min \sum_{i=1}^{n-1} (100 \cdot (x_i^2 - x_{i+1})^2 + (x_i + \sin(x_{i+1}) - 1)^2) \quad (2-115)$$

其中,  $n = 10$ , 10 个变量范围均在  $[-5, 10]$  之间。

步骤:

1) 编写 Fortran 源代码如下

```
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
```

- 2) 保存文件，如 “c:\projects\Gfortran\_test1.f90”
- 3) 编译成动态库文件，如 GFortran 编译器位于 “d:\gfortran\bin”，则编译命令如下：  
d:\gfortran\bin\gfortran.exe -o “c:\projects\gfortran\_test1.dll” -s -shared -mrt d -fno-underscoring “c:\projects\gfortran\_test1.f90”

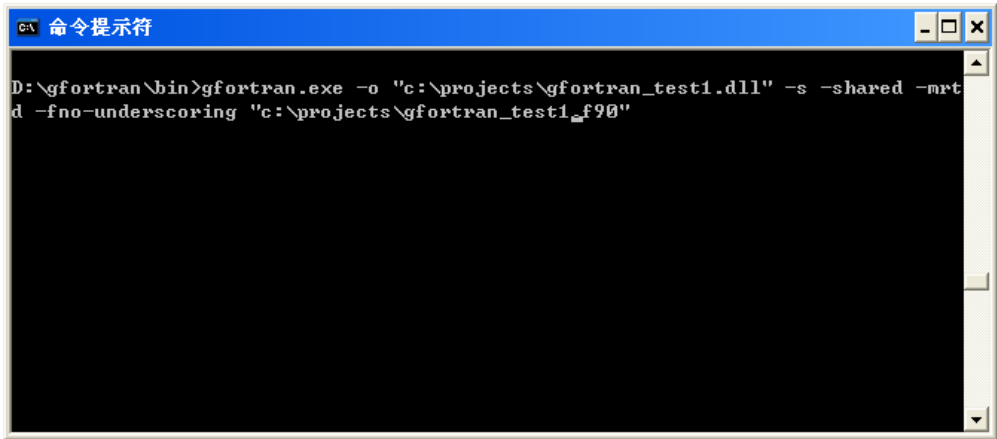


图 2-90 Gfortran 编译界面

- 4) 执行上述命令，可在 “c:\projects” 目录下产生一动态库文件 “gfortran\_test1.dll”
- 5) 1stOpt 中调用命令

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

- 6) 快捷模式代码

```
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);
```

两种模式均可得到相同的结果：Min. = 3.70328593254089

例 4. 拟合问题

拟合公式：
$$y = p_1 + (p_2 - p_1) \cdot (1 - \exp(-p_3 \cdot x))$$
 (2-116)

表 2-65 拟合数据

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

步骤

- 1) Fortran 源代码

```
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) 保存文件，如 “c:\projects\Gfortran\_reg.f90”
- 3) 编译成动态库文件，如 GFortran 编译器位于 “d:\gfortran\bin”，则编译命令如下：  
d:\gfortran\bin\gfortran.exe -o “c:\projects\gfortran\_reg.dll” -s -shared -mrtld  
-fno-underscoring “c:\projects\gfortran\_reg.f90”
- 4) 执行上述命令，可在 “c:\projects” 目录下产生一动态库文件 “gfortran\_reg.dll”
- 5) 1stOpt 中调用命令

```

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

```

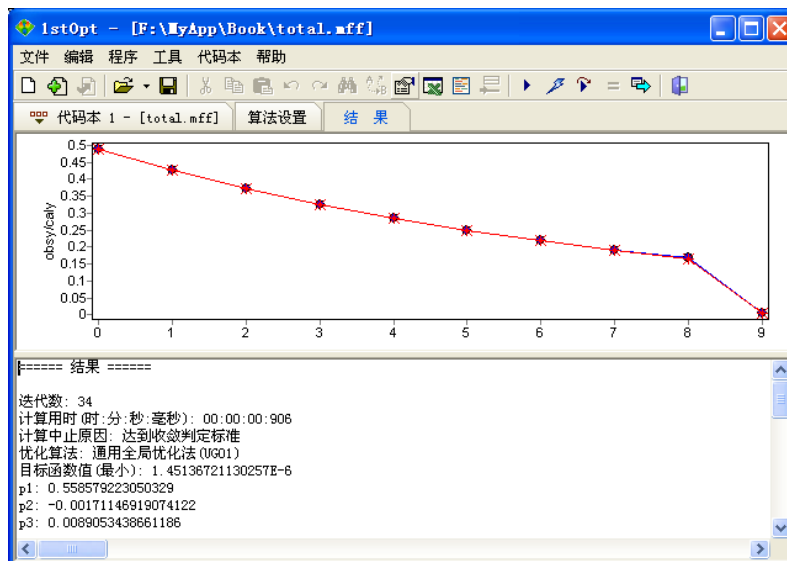


图 2-91 运行结果画面

- 6) 快捷模式代码

```

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);

```

两种模式均可得到相同的结果：Min. = 1.45142373499332E-6

## 2.7.5 Delphi 编译目标 Dll 文件

Delphi 是优秀的 Windows 平台下的软件开发环境，面向对象、易于使用、真编译，可创建高效的动态链接库，从 3.0 版到 2011 版，均可用于与 1stOpt 混合编程。下面以 Delphi2007 为样本。

输出函数格式定义如下

```
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
    //加入目标及约束函数计算代码
end;

exports dllfunction;

begin
end.
```

其中：

- DllProject: Dll 库文件名，可自己命名
- dllfunction: 缺省输出函数名，可自己命名
- Para: 从 1stOpt 传入的参数数组，下标从 0 开始
- ConFun1, ConFun2: 不等式及等式约束数组，下标从 0 开始
- ObjFun: 目标函数值
- PassPara: 返回传递参数数组，下标从 0 开始
- “TVector = array[0..1] of Double;”: 定义的数组上限值应取 Para、ConFun1、ConFun2 及 PassPara 数组上限值最大者再减去 1，如数组上限最大值为 10，则定义变为：

TVector = array[0..9] of Double;

### 例 5. 约束优化问题

下面以一约束函数优化问题为实例，讲解如何用 Delphi 2007 创建外部动态链接库，及如何从 1stOpt 中调用。

优化问题定义如下，除一个目标函数外，还有一个等式约束和两个不等式约束，参数范围自由。

$$\min. x_1^2 + x_2 \quad (2-117)$$

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

用 Delphi 2007 创建上述优化问题的动态链接库步骤如下：



- (1) 启动 Delphi2007，选择“DLL Wizard”，按“OK”

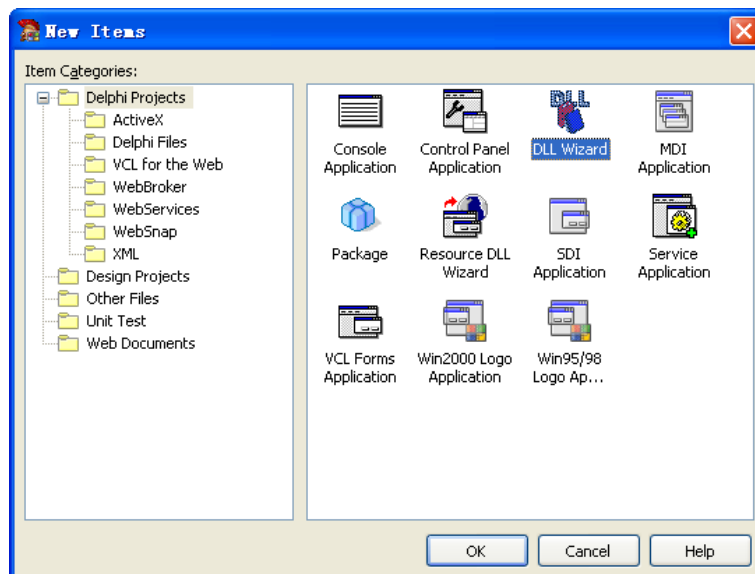


图 2-92 Delphi 工程选项

- (2) Delphi 2007 自动产生如下代码

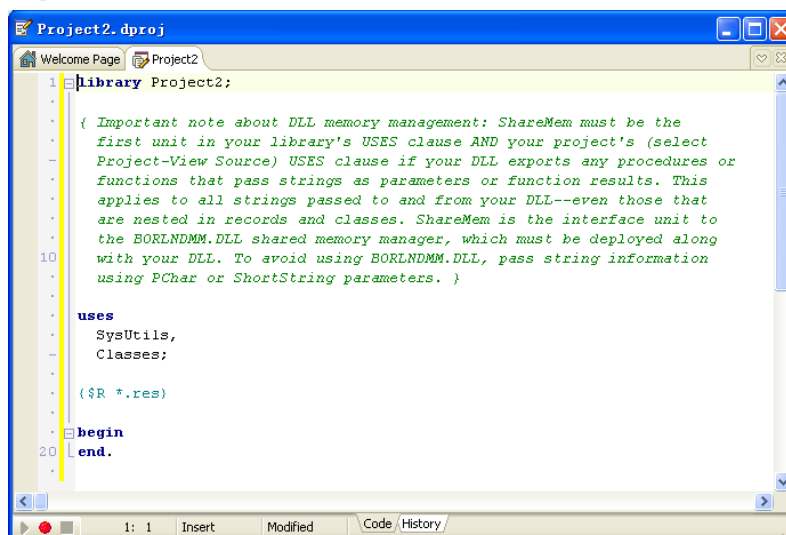


图 2-93 Delphi 动态库自动生成

- (3) 添加如下代码

```
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.
```

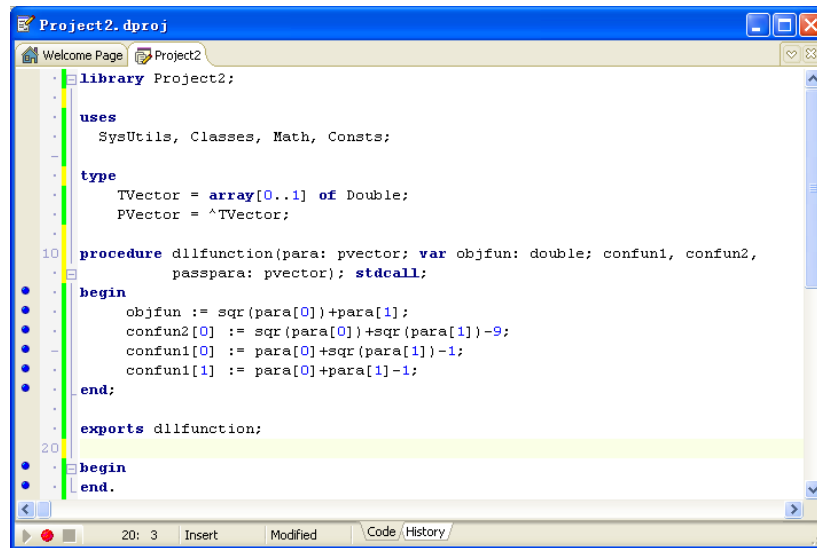


图 2-94 Delphi 动态库代码

- (4) 编译生成动态库，如"C:\Projects\Project2.dll";
- (5) 从 1stOpt 中调用，其代码如下

```
Parameter x1, x2;
MinFunction "C:\Projects\Project2.dll[2,1]";
```

执行可容易得到结果：Min = 3.7913455, x1 = -2.3722817, x2 = -1.836375

- (6) 与快捷模式进行比较：两者的结果是完全相同的。

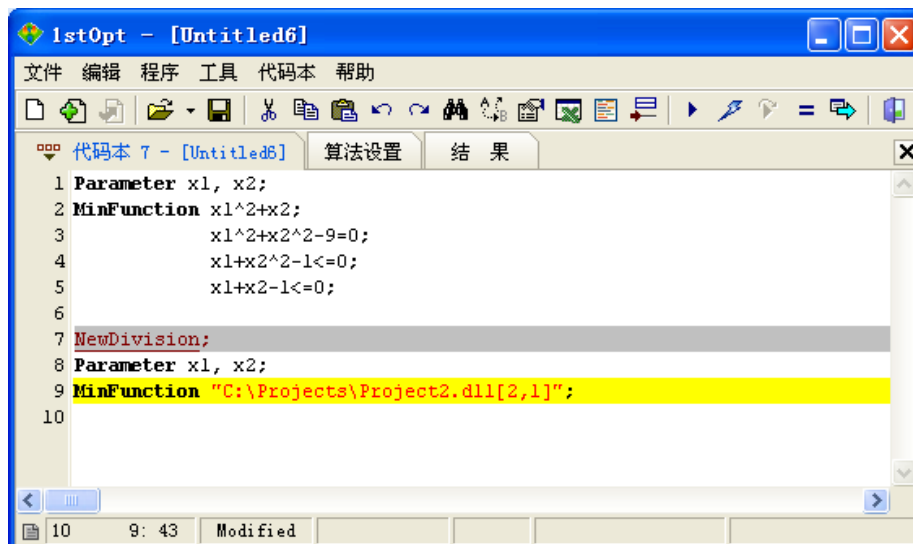


图 2-95 快捷模式与外部调用模式

例 6. 带约束的拟合问题

$$\text{拟合公式: } y = \frac{b_1(x^2 + b_2x)}{x^2 + b_3x + b_4} \quad (2-118)$$

约束:  $b_1 + b_2 + b_3 + b_4 = 1$

表 2-66 拟合数据

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 代码如下, 该代码可在 Delphi 3.0 至 Delphi 2011 任意版本中编译通过。  
数据拟合外部动态库源代码

```
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.
```

如编译完成的动态链接库为“C:\Projects\ Pascal\_Demo2\_Reg”, 则 1stOpt 调用的代码如下。

1stOpt 调用动态库代码

```
Parameter b1,b2,b3,b4;
PassParameter X(0:10), ObsY(0:10), CalY(0:10);
Plot X[x], ObsY, CalY;
MinFunction "F:\MyApp\App\1stopt\Examples\Mix Program\Pascal_Demo2_Reg.dll";
```

上述代码中, “PassParameter” 定义了三个一维数组, 即自变量 **x**, 因变量 (目标) **ObsY** 和因变量 (计算) **CalY**, 数组长度与拟合数据长度一致。注意上述源码中如何计算并返回这三个数组: “PassParameter” 不论定义几个一维数组, 在 Delphi 源码中都累加为一个一维

数组 “passpara”，此例中，下标 0 至 10 的 “passpara” 为 x，11 至 21 为 ObsY，22 至 32 为 CalY。语句 “Plot X[x], ObsY, CalY;” 表示以 X 为横坐标、ObsY 和 CalY 为纵坐标作图。运行上述代码可动态显示计算结果和图形变化，如下：

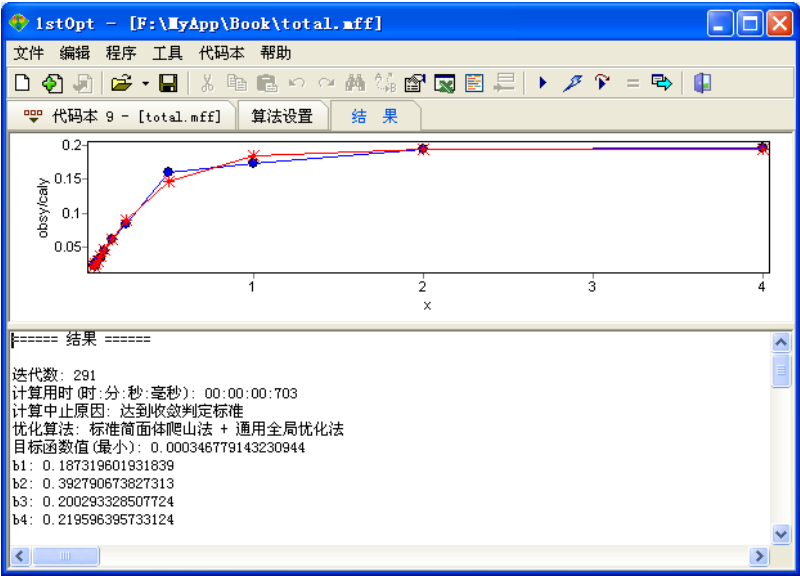


图 2-96 调用动态库拟合结果展示

### 2.7.6 PowerBasic 编译目标 Dll 文件

PowerBasic 是当今 Basic 家族众多产品中一款功能非常强大的 Windows 开发工具，其官方网站在 <http://www.powerbasic.com/>。

PowerBasic 编译器允许用你可以用熟悉的 BASIC 语言编写基于 Dos 或 Windows 工业标准的动态连接库 (DLL) 和可执行程序 (EXE)。虽然 PowerBASIC (PB) 与流行的微软 Visual Basic (VB) 都属 Basic 家族，但一根本的不同之处是 VB 属于解释性语言而 PB 为编译性语言，其结果是：通常 PB 编译的代码在性能上比 VB 编译的代码快 3 倍以上，同时 PB 产生的可执行文件比 VB 生成的可执行文件小很多且也不需要外部文件的运行支持。

本节以 PB 8.0 为例，介绍如何创建符合 1stOpt 接口的动态链接库。  
输出函数格式定义如下

```
#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
    '加入目标及约束函数计算代码
end sub
```

其中：

- Compile Dll: PB 命令，编译成动态库文件
- dllfunction: 缺省输出函数名，可自己命名
- para: 从 1stOpt 传入的参数数组，下标从 0 开始
- confun1, confun2: 不等式及等式约束数组，下标从 0 开始

- objfun: 目标函数值
- passpara: 返回传递参数数组，下标从 0 开始

例 7. 整数约束优化问题

$$\min. x_1^{0.1} + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \quad (2-119)$$

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

其中参数均为大于 1 的正整数，而  $x_1$  范围为  $[5, 10]$ 。该优化问题有一个目标函数，七个不等式约束函数。

步骤：

1) 启动 PB 可视编译器 PBEEdit.Exe，并输入代码如下：

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

Sub dllfunction StdCall Alias "dllfunction" (ByVal para As Double Ptr, ByVal 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
```

注意上面代码中不等式约束均由“ $\geq$ ”形式变为了“ $\leq 0$ ”的形式。

2) 保存为“PowerBasic\_Demo1.bas”，并编译成动态库文件“PowerBasic\_Demo1.dll”

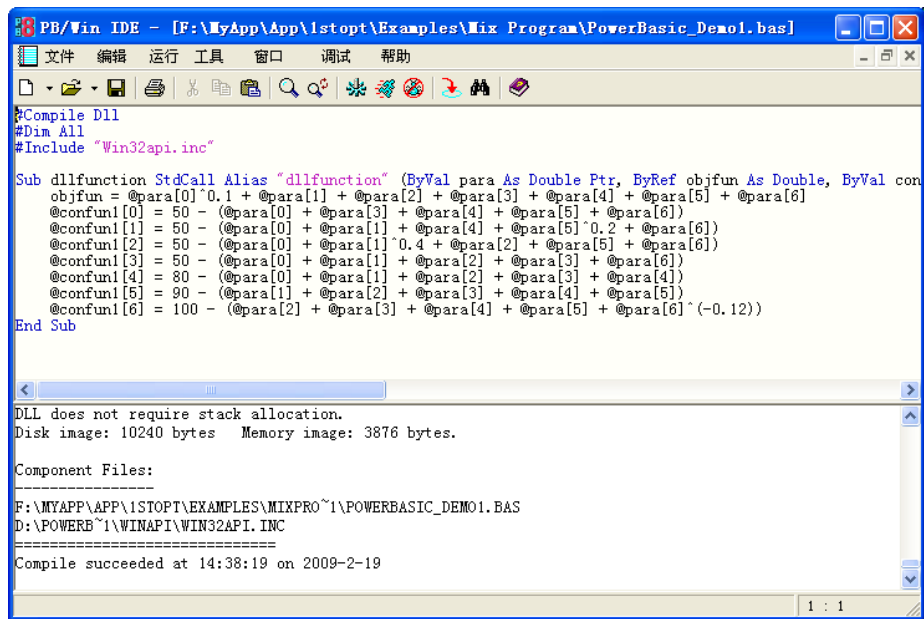


图 2-97 PowerBasic 动态库代码界面

### 3) 1stOpt 调用代码

```
Parameter x1[5,10,0], x(2:7) = [1,,0];
Algorithm = SM2[50];
MinFunction "F:\MyApp\App\1stopt\Examples\Mix Program\PowerBasic_Demo1.dll[7]";
```

### 4) 为便于比较，下面给出快捷模式和编程模式的代码

#### 快捷模式

```
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;
```

#### 编程模式：

```
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;
```

上面几种模式运行都可以得到相同的最优结果：Min. = 102.174618943088，但参数组值却不是唯一的。

三种模式中，快捷模式无疑是最简单易懂的，编程模式次之，调用外部动态库模式相对来说比较麻烦，但却有着无可比拟的优势：不加约束地充分发挥高级语言的编程能力，创建任意难度的目标函数文件。

### 2.7.7 Free Basic 编译目标 DLL 文件

FreeBasic (FB) 是一款开源免费的 Basic 编译器，被称之为 BASIC 语言界的黑马，完全兼容 QuickBASIC，易学易用，跨平台，能够产生高品质的原生本地机器码，运行速度快，真编译，无需外部运行库支持。其官方网站在 <http://www.freebasic.net/>。

FB 仅提供命令行编译器，详细命令可参考其使用指南和相关参考书。FB 编译器执行文件名名为“FBC.exe”，如要将源文件“c:\projects\FB\_test1.bas”编译成动态库文件，命令如下：

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

输出函数格式定义如下

```
'dllfunction: 缺省函数名
'Para: 从 1stOpt 传入的参数
'ConFun1, ConFun2: 不等式及等式约束
'ObjFun: 目标函数值
'PassPara: 传递返回数

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
```

其中：

- dllfunction: 缺省输出函数名，可自己命名
- para: 从 1stOpt 传入的参数数组，下标从 0 开始
- confun1, confun2: 不等式及等式约束数组，下标从 0 开始
- objfun: 目标函数值
- passpara: 返回传递参数数组，下标从 0 开始

取本篇例 2 相同约束优化问题：

步骤：

1) Free Basic 代码

```
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
```

保持为“c:\FB\_Test1.bas”

2) 编译成动态库文件，如 FB 编译器位于“C:\Free Basic\BIN”，则编译命令如下：

```
C:\Free Basic\BIN\fbc.exe -dll "C:\FB_Test1.bas"
```

3) 执行上述命令，产生一动态库文件“FB\_Test1.dll”

4) 1stOpt 中调用命令

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


5) 快捷模式代码

```
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;
```

两种模式均可得到相同的结果：-2.345129

## 2.7.8 1stOpt 外部程序编辑器（IDE）

1stOpt 自带有一外部程序编辑器（IDE），可用于编写、编译外部高级程序语言以生成可供 1stOpt 调用的动态库文件（Dll 文件），该 IDE 含有 Delphi、FreeBasic、C++、Visual Fortran、PowerBasic 和 Gun Fortran 动态库模板，可方便用户创建动态库文件。

- 1) 启动 IDE：选主菜单“工具”→“外部程序编辑器”，或按快捷键“F7”，或从快捷按钮 ：

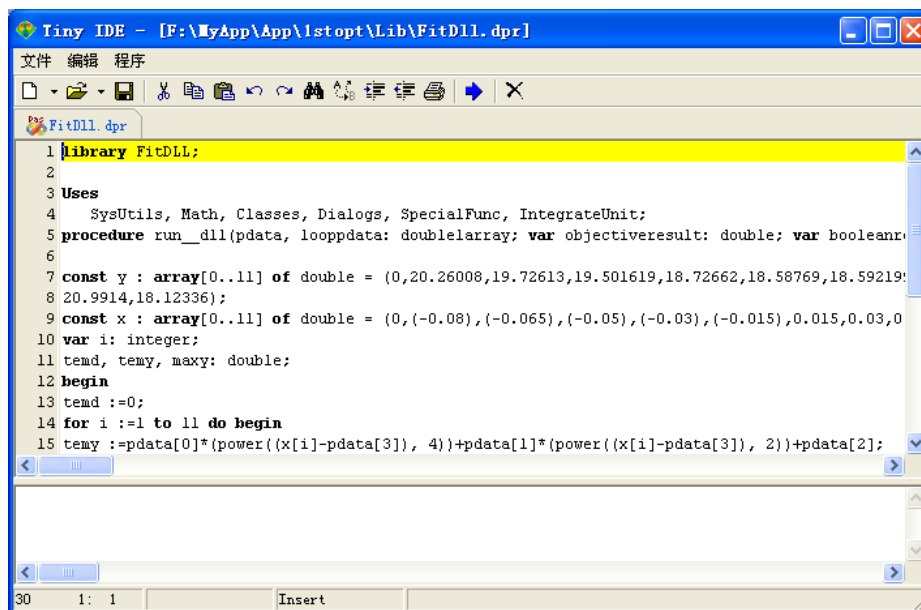


图 2-98 1stOpt 外部程序编辑器

- 2) 添加新工程文件：按“文件”→“新建”，可选择编程语言



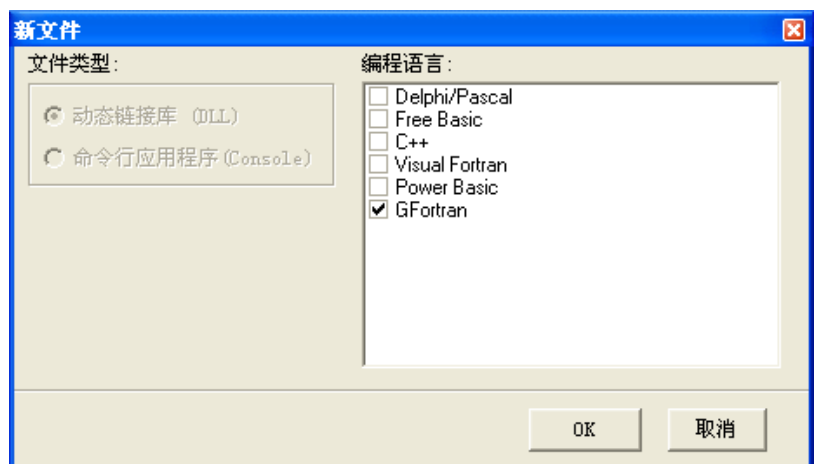


图 2-99 语言选项窗口

3) 编译器设定：选择编译器，包括路径及编译器名

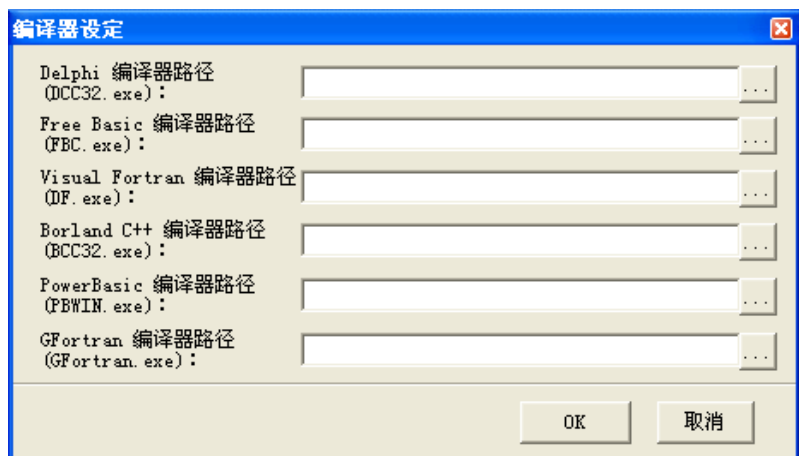


图 2-100 编译器路径设定

预设模板：选择不同的语言，将会产生不同的动态库文件模板

如选择 Delphi，自动产生代码

```
//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: 参数、不等式及等式约束数目之大者减 1

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

//dllfunction: 缺省函数名
//Para: 从 1stOpt 传入的参数
//ConFun1, ConFun2: 不等式及等式约束
//ObjFun: 目标函数值
//PassPara: 传递返回数

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

```

end;

exports dllfunction;

begin
end.

```

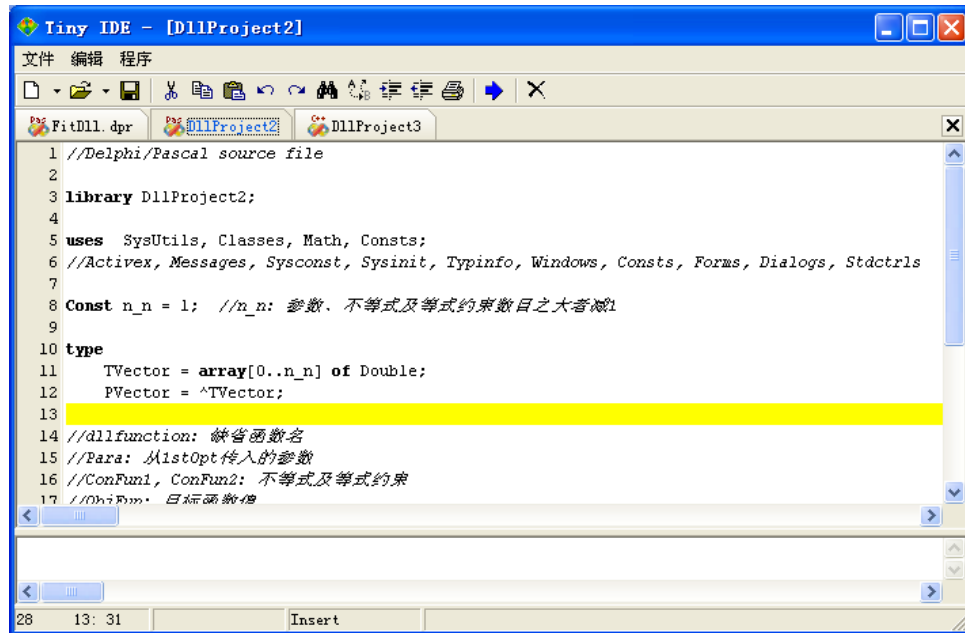


图 2-101 编译器代码 Pascal 语言编辑界面

选择 C++, 产生如下代码:

```

//C++ source file

//dllfunction: 缺省函数名
//Para: 从 1stOpt 传入的参数
//ConFun1, ConFun2: 不等式及等式约束
//ObjFun: 目标函数值
//PassPara: 传递返回数

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

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

```

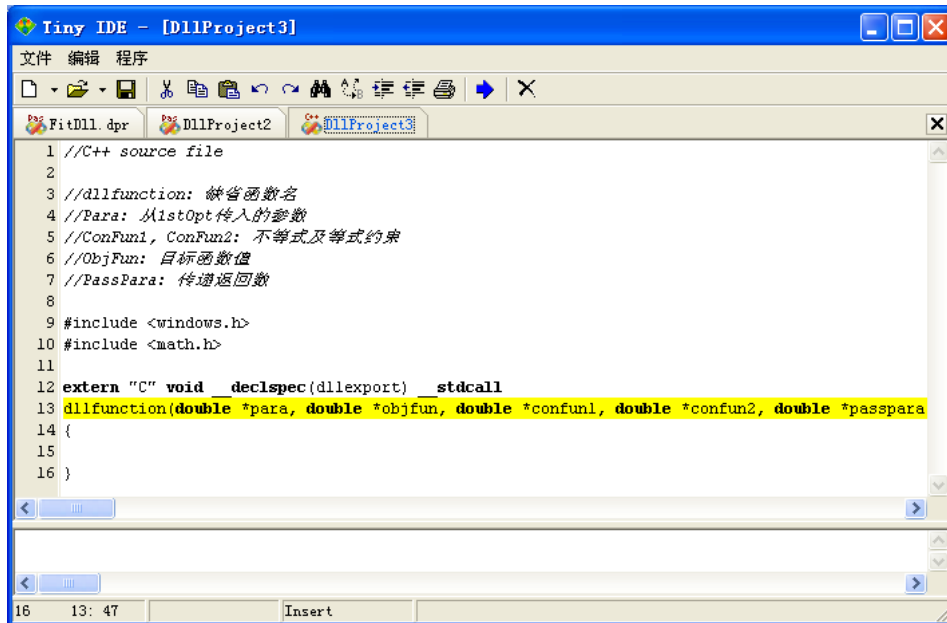


图 2-102 编译器代码 C++语言编辑界面

- 4) 生成动态库文件：填写目标函数及约束函数后，保存文件，如果编译器设定正确，按“编译”，可生成供 1stOpt 调用的 Dll 文件。