1stOpt 9.0 主要新增功能

1stOpt 9.0 版于 2020 年 10 月 18 日正式发布，主要新增功能及改进如下。12 月 1 日开始符合免费升级或打算升级的用户可申请办理。

1. 多层自回归网络工具箱功能提升

增加了“多模”模型结构，对拟合及分类在效果方面有大幅提高，尤其对低维数据，同时支持单一输入。

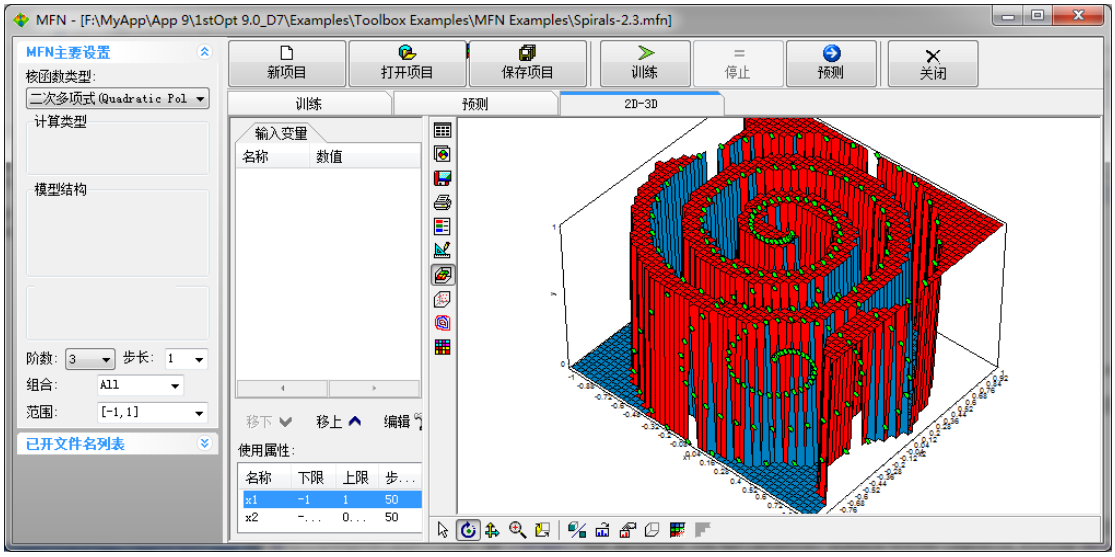

图 1、经典双螺旋（Spirals）分类问题

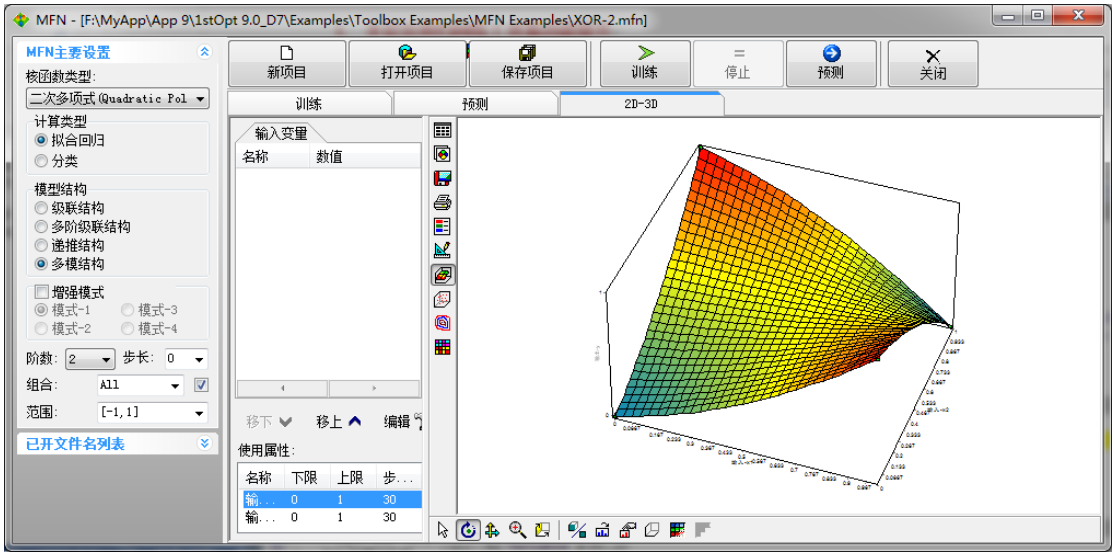

图 2、经典与或门（XOR）问题

2. 增加了线性、多项式及完全多现实拟合计算功能

线性拟合或多项式拟合问题虽然都可视为非线性拟合问题去求解，但求解变量数比较大时，对应的未知参数数也会很多，采用非线性最优化迭代算法进行计算，不仅效率会低下，效果也无法完全保证，“牛刀杀鸡”事倍功半，专门的线性拟合或多项式拟合算法命令，无需迭代计算，在效率上有质的飞跃，同时效果也有保证。

✧ LineFit：多元线性拟合，命令格式：LineFit(x1,x2,…)，自变量数不限，自变量数很多时也可写为：LineFit(x(k))，其中 k 为实际自变量数。

两自变量线性拟合案例代码：

```
Variable x1,x2,y;
Function y=LineFit(x1,x2);
Data;
x1=[23.73,24.31,25.16,26.34,28.05,30.47,33.94,39.27,42.4,47.15,51.35,54.1,55.22];
x2=[-3.329,-4.159,-5.153,-6.63,-8.462,-10.59,-12.83,-14.84,-15.31,-15.09,-13.46,-10.82,-7.7];
y=[21.23,14.36,9.766,6.643,4.542,3.09,2.105,1.434,0.9766,0.6643,0.4542,0.309,0.2105];
```

上述代码的拟合公式等同于：

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2$$

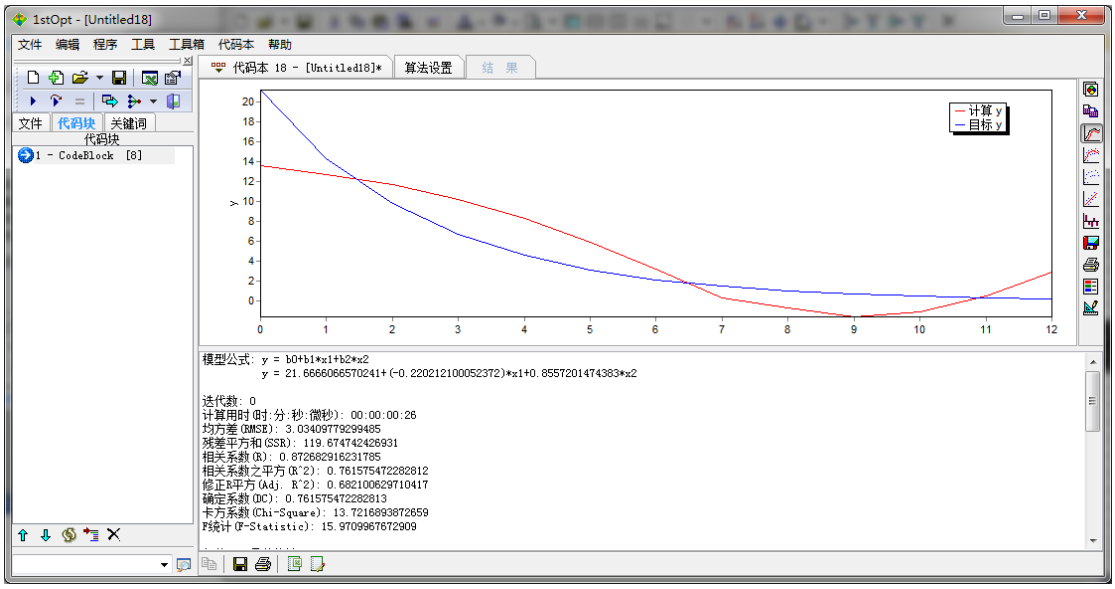

图 3、线性拟合结果

✧ PolyFit：多元多项式拟合，命令格式：PolyFit(x1,x2,…,N)，自变量数不限，自变量数很多时也可写为：PolyFit(x(k),N)，其中 k 为实际自变量数，N 表示多项式阶数。

两自变量 2 阶多项式案例代码：

```
Variable x1,x2,y;
```

```
Function y=PolyFit(x1,x2,2);
Data;
x1=[23.73,24.31,25.16,26.34,28.05,30.47,33.94,39.27,42.4,47.15,51.35,54.1,55.22];
x2=[-3.329,-4.159,-5.153,-6.63,-8.462,-10.59,-12.83,-14.84,-15.31,-15.09,-13.46,-10.82,-7.7];
y=[21.23,14.36,9.766,6.643,4.542,3.09,2.105,1.434,0.9766,0.6643,0.4542,0.309,0.2105];
```

上述代码的拟合公式等同于：

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_1^2 + b_3 \cdot x_2 + b_4 \cdot x_2^2$$

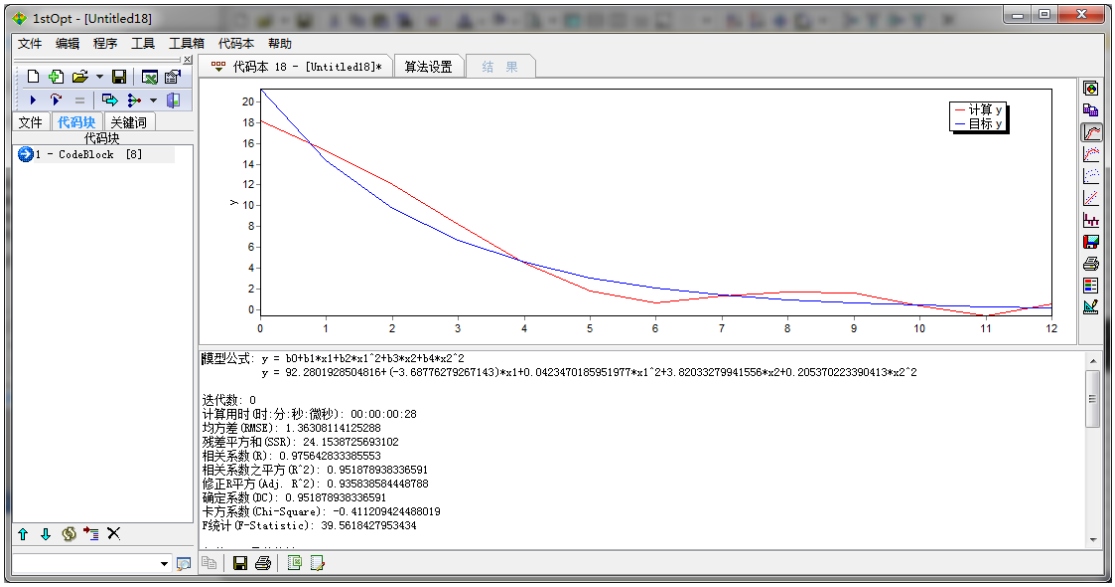

图 4、多项式拟合结果

✧ PolyFit2：完全多项式拟合是在多项式拟合的基础上增加了所有自变量及各阶自变量任意阶数项的组合，命令格式：PolyFit2(x1,x2,…,N)，自变量数不限，自变量数很多时也可写为：PolyFit2(x(k),N)，其中 k 为实际自变量数，N 表示多项式阶数。

案例代码：

```
Variable x1,x2,y;
Function y=LineFit2(x1,x2，2);
Data;
x1=[23.73,24.31,25.16,26.34,28.05,30.47,33.94,39.27,42.4,47.15,51.35,54.1,55.22];
x2=[-3.329,-4.159,-5.153,-6.63,-8.462,-10.59,-12.83,-14.84,-15.31,-15.09,-13.46,-10.82,-7.7];
y=[21.23,14.36,9.766,6.643,4.542,3.09,2.105,1.434,0.9766,0.6643,0.4542,0.309,0.2105];
```

上述代码的拟合公式等同于：

$$y = b_0 + b_1 \cdot x_1 + b_2 \cdot x_1^2 + b_3 \cdot x_2 + b_4 \cdot x_2^2 + b_5 \cdot x_1 \cdot x_2 + b_6 \cdot x_1 \cdot x_2^2 + b_7 \cdot x_1^2 \cdot x_2 + b_8 \cdot x_1^2 \cdot x_2^2$$

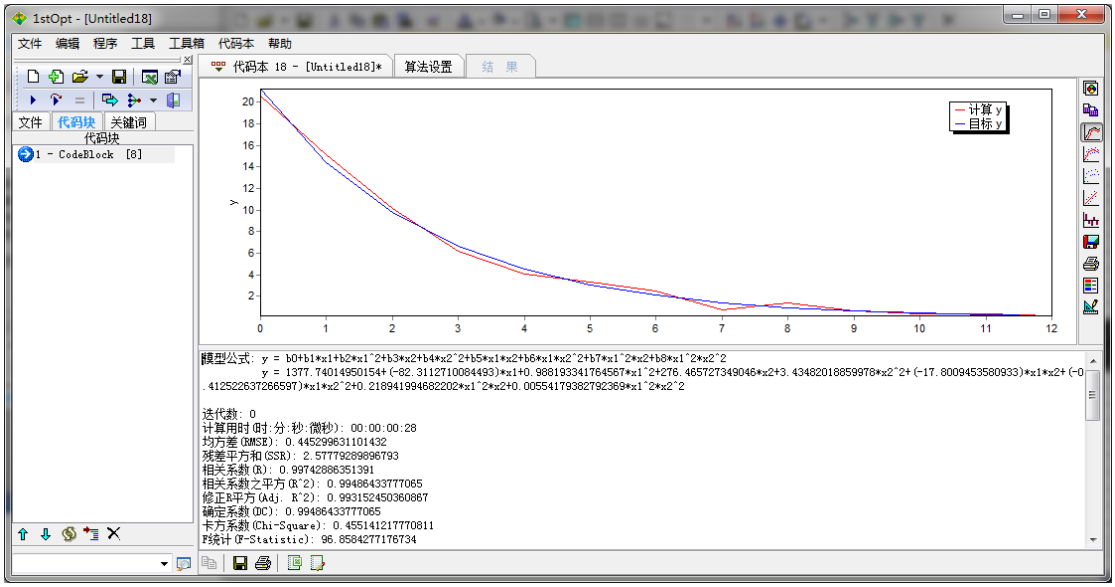

图 5、完全多项式拟合结果

3. 参数赋予初值为 0 时效率大幅提高。

代码：

```
Constant k=1.38*10^(-23);
Parameter ea=0;
Variable f,tm;
Function    f=f0*exp(-Ea/(k*(Tm-Tf)));
Data;
f            Tm (K)
0.1            251.52
0.5            257.31
1             260.93
5             266.99
10             270.07
```

上述代码随机运行 10 次，获得最优解次数 9.0 版相较于 8.0 版提升 9 倍

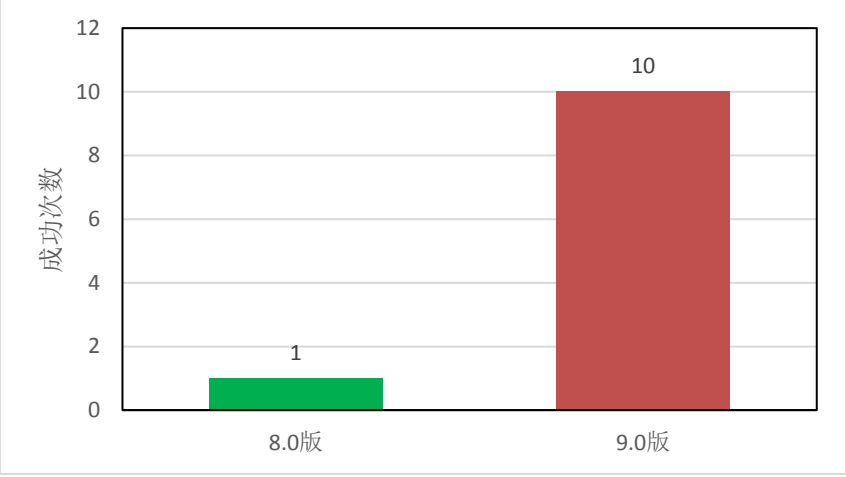

图 6、8.0 版与 9.0 版测试对比

4. 拟合时目标函数可选择最小二乘（实际输出值及对应计算值差值平方和最小）

或确定系数 DC（对应 WeightedReg = 3 命令，也即权重系数为 w = Variance^2），对于多数据多输出且输出值数量级相差很大时，选择后者可兼顾整体效果更好。

代码：

```
WeightedReg = 3;
Variable T,y1,y2;
ODEFunction
y1' = k0;
y2' = k1*y1*y2;
Data;
T = [0 2 4 6 8 10 12];
y1=[307.18 394.39 441.93 516.62 565.13 636.74 653.68];
y2 = [21.06666667 15.4 9.633333333 4.666666667 0.753333333 0.403333333 0.206666667];
```

上述微分方程拟合问题代码，其特点是两个输出 y1 和 y2 在数量级上相差很大，如果按常规最小二乘目标计算，y1 拟合效果很好但 y2 却极差如图 7。在上述代码前增加一句：WeightedReg = 3; 结果如下图 8

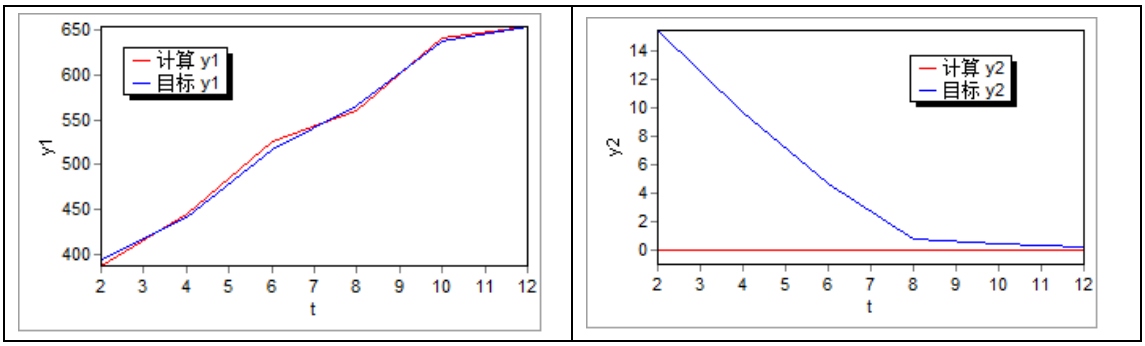

图 7、常规最小二乘法拟合结果对比

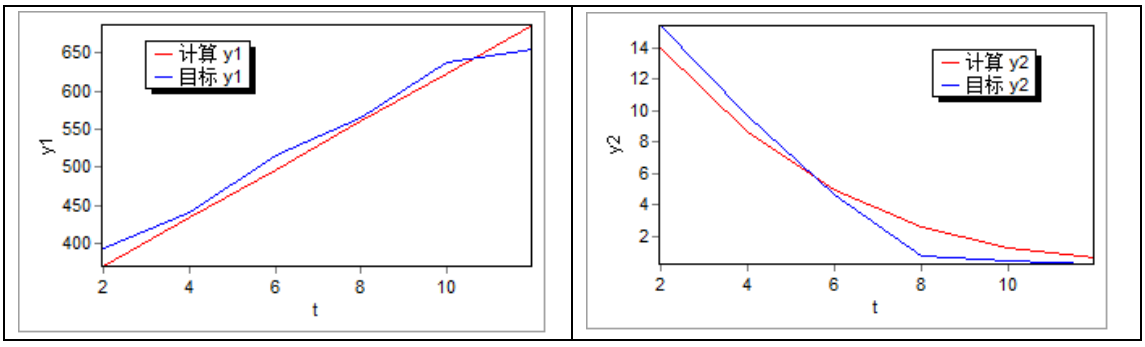

图 8、以确定系数 DC 为目标的拟合结果对比

5. 输出结果多页面保存，多页面预测；

每次计算输出结果都可以设定为自动保存

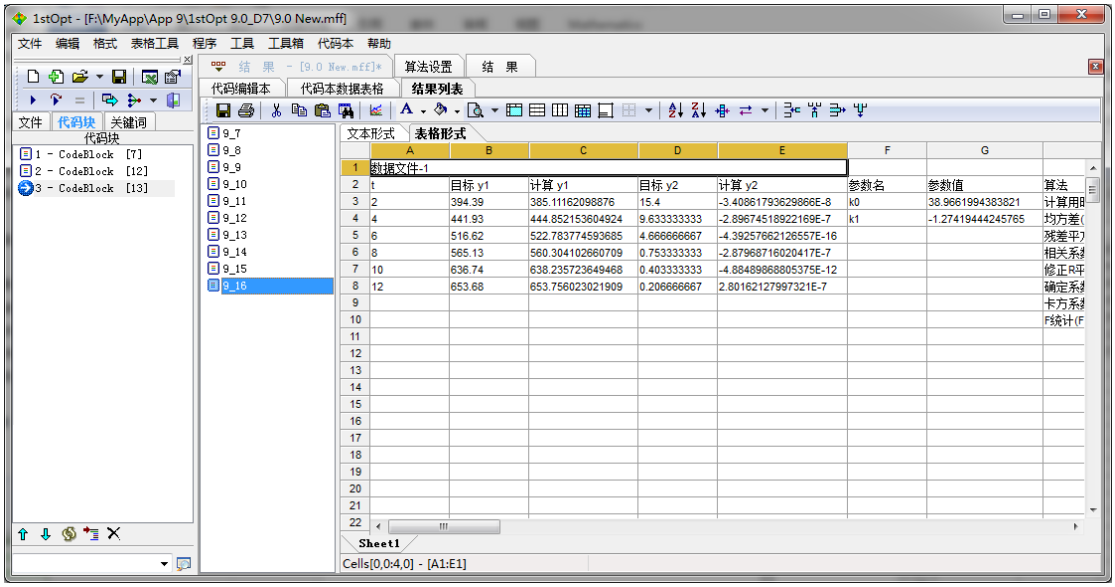

图 9、计算结果自动保存及展示

6. 调用外部高级语言编写的 dll 目标函数时支持循环常数（LoopConstan）传递，同时也支持数据直接从代码本传入目标函数动态库

通过关键字 RunDllModel 实现。

1) RunDllModel = 0；缺省设置，与 8.0 版及之前的版本用法一致，只有参数输入。

dllfunction(para, objfun, confun1, confun2, passpara)

2) RunDllModel = 1；增加循环常数输入。

dllfunction(loopdata, para, objfun, confun1, confun2, passpara)

3) RunDllModel = 2；增加循环常数及三组一维数组输入。

dllfunction(loopdata, dll_xdata, dll_ydata, dll_zdata, para, objfun, confun1, confun2, passpara)

4) RunDllModel = 3；增加循环常数及一组二维数组输入。

dllfunction(loopdata, dll_data, para, objfun, confun1, confun2, passpara)

5) RunDllModel = 4；增加循环常数、三组一维数组及一组二维数组输入。

dllfunction(loopdata, dll_xdata, dll_ydata, dll_zdata, dll_data, para, objfun, confun1, confun2, passpara)

6) RunDllModel = 5；对应“RunDllModel = 2；”，只是没有循环常数输入，只有三组一维数组输入。

dllfunction(dll_xdata, dll_ydata, dll_zdata, para, objfun, confun1, confun2, passpara)

7) RunDllModel = 6；对应“RunDllModel = 3;”，只是没有循环常数输入，只有一组二维数组输入。

dllfunction(dll_data, para, objfun, confun1, confun2, passpara)

8) RunDllModel = 7；对应“RunDllModel = 4;”，只是没有循环常数输入，只有三组一维数组及一组二维数组输入。

dllfunction(dll_xdata, dll_ydata, dll_zdata, dll_data, para, objfun, confun1, confun2, passpara)

数据传递功能仅适用于外部目标函数是动态库 dll 形式而非 EXE 命令行形式。

例：求下列函数在所有 a 和 b 不同组合下的最小值：

$$Min = \sum_{i=0}^{n-2} (a \cdot 100 \cdot (x_i^2 - x_{i+1})^2 + b \cdot (x_i + \sin(x_{i+1}) - 1)^2)$$

其中 n=10，$x \in [-5,10]$，a 和 b 为循环常数：a=[1,2,3], b=[0.3,0.2,0.1]

通常，该问题可用下面快捷代码求解：

```
Constant n=10;
LoopConstant a=[1,2,3], b=[0.3,0.2,0.1];
FullLoopModel;
Parameter x(0:n-1)=[-5,10];
MinFunction Sum(i=0:n-2)(a*100.0*(x[i]^2.0-x[i+1])^2.0+b*(x[i]+sin(x[i+1])-1.0)^2.0);
```

如果用其它高级语言如 C++或 Fortran 如何求解上述问题？高级语言编译的目标函数动态库是独立于 1stOpt 的，循环常数该如何传递处理？以 Fortran 为例：

Fortran 源代码：

```
subroutine dllfunction(loop_data, para, objfun, confun1, confun2, passpara)
integer, parameter :: fp = selected_real_kind(15,300)
real(fp) :: Loop_data(0:1), para(0:9), confun1(0:0), confun2(0:0), passpara(0:0)
real(fp) :: objfun
real(8) temd
integer i, j

temd = 0.0
Do i = 0, 8
    temd = temd + loop_data(0)*100.0*(para(i)**2.0-para(i+1))**2.0+ &
           loop_data(1)*(para(i)+sin(para(i+1))-1.0)**2.0
EndDo

objfun = temd
end subroutine
```

上述代码保存为“c:\mytest\dlltest1.f90”，再编译成动态库文件“c:\mytest\dlltest1.dll”。

此时用到在 1stOpt 里调用时用 RunDllModel = 1 命令。

1stOpt 调用外部动态库：

```
Constant n=10;
LoopConstant a=[1,2,3], b=[0.3,0.2,0.1];
FullLoopModel;
RunDLLModel = 1;
Parameter x(0:n-1)=[-5,10];
MinFunction "c\mytest\dlltest1.dll";
```

计算结果与快捷模式相同。

7. 支持连续定义约束: SubjectTo x1>x2>x3>x4;

```
ParameterDomain = [0,90];
Function
cosd(5*a)+cosd(5*b)+cosd(5*c)+cosd(5*d)+cosd(5*e)+cosd(5*f)=0;
cosd(7*a)+cosd(7*b)+cosd(7*c)+cosd(7*d)+cosd(7*e)+cosd(7*f)=0;
cosd(11*a)+cosd(11*b)+cosd(11*c)+cosd(11*d)+cosd(11*e)+cosd(11*f)=0;
cosd(13*a)+cosd(13*b)+cosd(13*c)+cosd(13*d)+cosd(13*e)+cosd(13*f)=0;
cosd(17*a)+cosd(17*b)+cosd(17*c)+cosd(17*d)+cosd(17*e)+cosd(17*f)=0;
cosd(19*a)+cosd(19*b)+cosd(19*c)+cosd(19*d)+cosd(19*e)+cosd(19*f)=0;
0 <a < b < c < d < e < f < 90;
```

```
ParameterDomain = [0,90];
SubjectTo 0 <a < b < c < d < e < f < 90;
Function
cosd(5*a)+cosd(5*b)+cosd(5*c)+cosd(5*d)+cosd(5*e)+cosd(5*f)=0;
cosd(7*a)+cosd(7*b)+cosd(7*c)+cosd(7*d)+cosd(7*e)+cosd(7*f)=0;
cosd(11*a)+cosd(11*b)+cosd(11*c)+cosd(11*d)+cosd(11*e)+cosd(11*f)=0;
cosd(13*a)+cosd(13*b)+cosd(13*c)+cosd(13*d)+cosd(13*e)+cosd(13*f)=0;
cosd(17*a)+cosd(17*b)+cosd(17*c)+cosd(17*d)+cosd(17*e)+cosd(17*f)=0;
cosd(19*a)+cosd(19*b)+cosd(19*c)+cosd(19*d)+cosd(19*e)+cosd(19*f)=0;
```

```
ParameterDomain = [0,90];
Constant v=[5,7,11,13,17,19];
Function For(i=1:6,v)(cosd(v*a)+cosd(v*b)+cosd(v*c)+cosd(v*d)+cosd(v*e)+cosd(v*f)=0);
         0 < a < b < c < d < e < f < 90;
```

```
ParameterDomain = [0,90];
Constant v=[5,7,11,13,17,19];
Function For(i=1:6,v)(Sum(i=1:6,p)(cosd(v*p))=0);
         For(i=0:6)(if(i=0,0<p[1],if(i=6, p[i]<90, p[i]<p[i+1])));
```

8. 重复定义参数，服从后者

例1. Parameter p1, p2, p1=[0,1]; 等同于： Parameter p1=[0,1], p2;

例2. Parameter p1=[0,], p2, p1=[,1]; 等同于： Parameter p1=[0,1], p2;

9. 连续定义参数

例1. Parameter -0.5<A3<0, 0<t1<t2<10, A1<A2<0;

10. 常数及常字符串可以累加定义

例：Constant a=1, b=3, a=a+b+3;

ConstStr f=2*d, f=f+3;

等同于：Constant b=3, a=1+b+3;

ConstStr f=2*d +3;

11. 增加迭代计算自动保存命令

1) ObjAppendSave：随迭代计算过程动态添加保存目标函数值；
2) ParAppendSave：随迭代计算过程动态添加保存参数组值；
3) ObjAppendIteration：动态添加保存目标函数值的间隔迭代数，对应“ObjAppendSave”
4) ParAppendIteration：动态添加保存参数组值的间隔迭代数，对应“ParAppendSave”
5) SaveResultFile：保存最终计算结果；
6) SaveParameterFile：保存最终计算最佳参数值

12. 直接支持 Fortran 语言

例：求下列函数最小值，其中 n=5

$$\text{Min} = \sum_{i=1}^{n-1}\left(x_i^2 + x_{i+1}\cdot\sin(10\cdot\pi\cdot x_i\cdot x_{i+1}) + 2 + (x_i + x_{i+1})^2\right)$$

快捷模式代码：

```
Constant n =5;
Parameter x(n);
MinFunction Sum(i=1:n-1)(x[i]^2+x[i+1]*sin(10*pi*x[i]*x[i+1])+2+(x[i]+x[i+1])^2);
```

编程模式（Basic）代码：

```
Constant n=5;
Parameter x(n);
StartProgram [Basic];
Sub MainModel
    dim i as integer
    dim td1 as double
    td1 = 0
    for i = 1 to n - 1
        td1 = td1 + x(i)^2+x(i+1)*sin(10*pi*x(i)*x(i+1))+2+(x(i)+x(i+1))^2
    next
    ObjectiveResult = td1
End Sub
EndProgram;
```

编程模式（Fortran）代码：

```
Constant n=5;
Parameter x(n);
ConstStr pi=dacos(-1.D0);
StartProgram [Fortran];
Subroutine MainModel
    integer i
    real*8 td1
    td1 = 0.0
    do i = 1, n - 1
```

```
        td1 = td1 + x(i)^2.0+x(i+1)*sin(10.0*pi*x(i)*x(i+1))+2.0+(x(i)+x(i+1))^2.0
    end do
    ObjectiveResult = td1
End Subroutine
EndProgram;
```

编程模式（Pacsl）代码：

```
Constant n=5;
Parameter x(n);
StartProgram [Pascal];
Procedure MainModel;
var i: integer;
    td1: double;
Begin
    td1 := 0;
    for i := 1 to n - 1 do
        td1 := td1 + sqr(x[i])+x[i+1]*sin(10.0*pi*x[i]*x[i+1])+2.0+sqr(x[i]+x[i+1]);
    ObjectiveResult := td1;
End;
EndProgram;
```

上述四种模式的代码都可以得到相同的计算结果 6.50867235315496

*使用 Fortran 时需要另外安装 GFortran 编译器

13. 支持 Matlab 数据文件（.mat）输入及导出

在 1stOpt 内置电子表格里可以打开和保存 Matlab 数据文件（.mat）.

14. TolIteration 命令

增加了可控制判断迭代收敛次数的命令。

15. 输出多解参数值“MultiSolution”命令

新增加可同时输出多解的命令。

16. 编程模式下新增“DebugModel”命令

该命令可方便编程模式下代码调试。

17. “IncludeFile”，编程模式下增加外部代码单元

编程模式下可通该命令调用相关代码单元。

18. 图形指标格式设定

例：XAxis = Ln(x);可将横坐标设置为对数坐标。

YAxis = Ln(OutPut);可将纵坐标设置为对数坐标。

19. UGO 算法稳定性提高

20. 微分方程预测计算时缺失数据的问题修正

21. 数学函数检查是否定义为参数、常数等

22. 微分方程高阶拟合计算时自动降阶错误订正

23. 结果保存及展示时多输出图形的同时保存