1stOpt 11.0 主要新增及改进功能

1stOpt 11.0 版于 2024 年 11 月 11 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: 神经网络工具箱改进

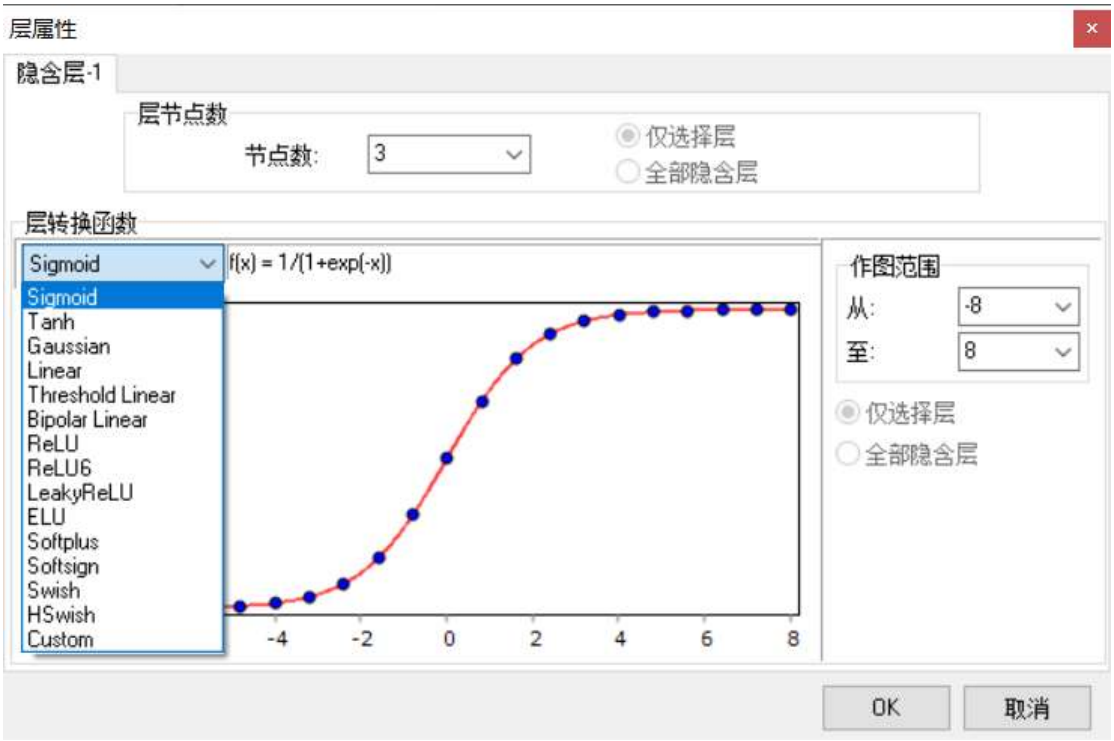
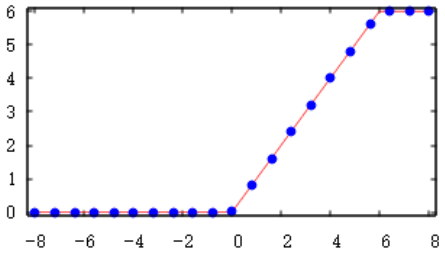
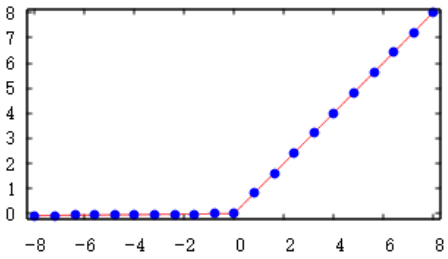
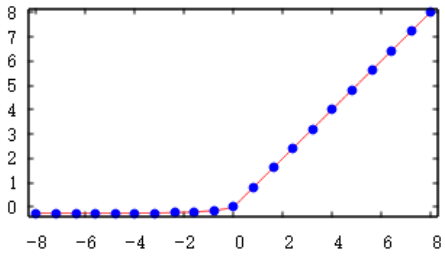
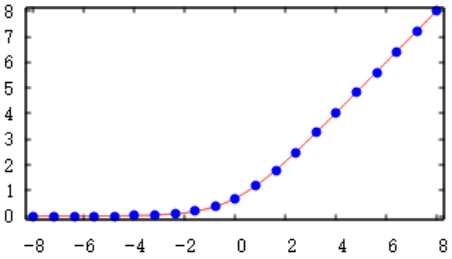
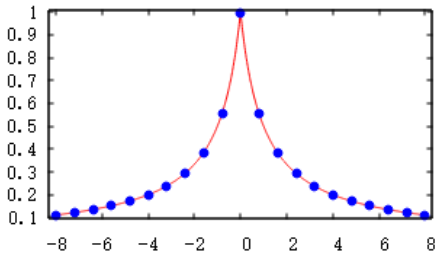
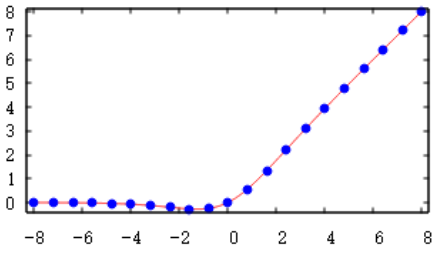


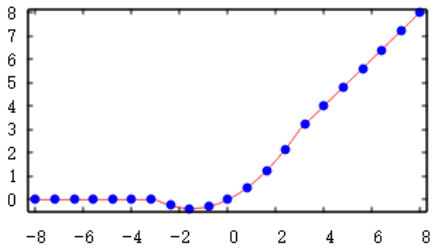
图-1：神经元转换函数

在原有七种神经元转换函数的基础上又新增加七种常见流行的转换函数，同时还增加了自定义转换函数的功能。

表-1：新增神经元转换函数

转换函数名	函数公式
ReLU6	 $f(x) = min(max(0, x), 6)$

LeakyReLU	 $f(x) = if(x > 0, x, a \cdot x)$
ELU	 $f(x) = if\big(x > 0, x, a \cdot (\exp(x) - 1)\big)$
Softplus	 $f(x) = \ln\big(1 + \exp(x)\big)$
Softsign	 $f(x) = \dfrac{1}{1 + \lvert x \rvert}$
Swish	 $f(x) = \dfrac{x}{1 + \exp(-x)}$

HSwish	 $f(x) = \frac{x \cdot \left(min(max(0, x+3), 6)\right)}{6}$

2: 作图功能提升

作图功能提升改进，三维作图网格数据可大于之前的 60*60.

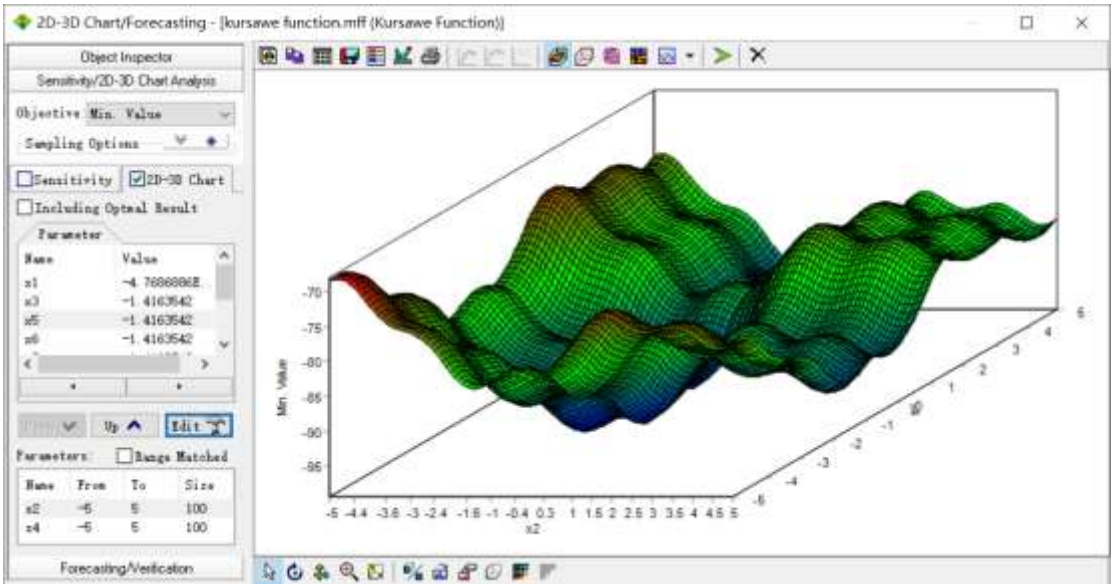


图-2：1stOpt 三维作图 100*100 网格图

3: 集成了著名的绘图引擎 GnuPlot，出图质量大幅提升达论文出版级

Gnuplot 是一款开源免费但功能强大的科研绘图工具，能提供并输出论文出版级的高质量二维及三维图形，集成后的 1stOpt 不仅可以一键转换成 GnuPlot 图形，还可以提供对应的 Gnuplou 基本作图脚本代码，方便用户进行任意修改。

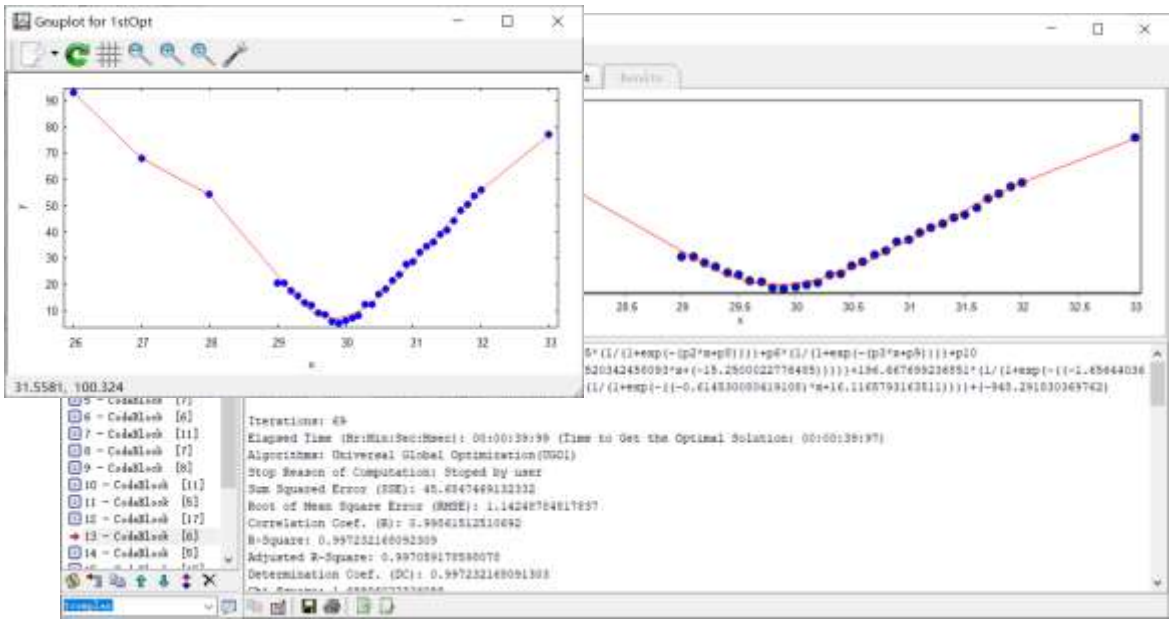


图-3：1stOpt 原图及对应的 GnuPlot 出图

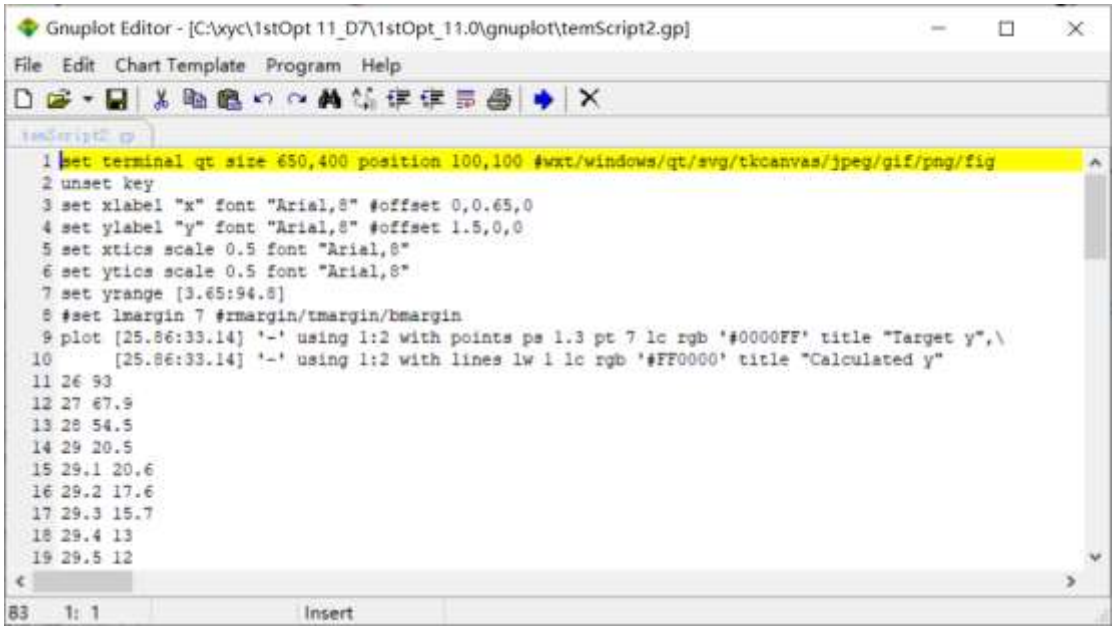


图-4：自动提供对应的 Gnuplot 作图脚本

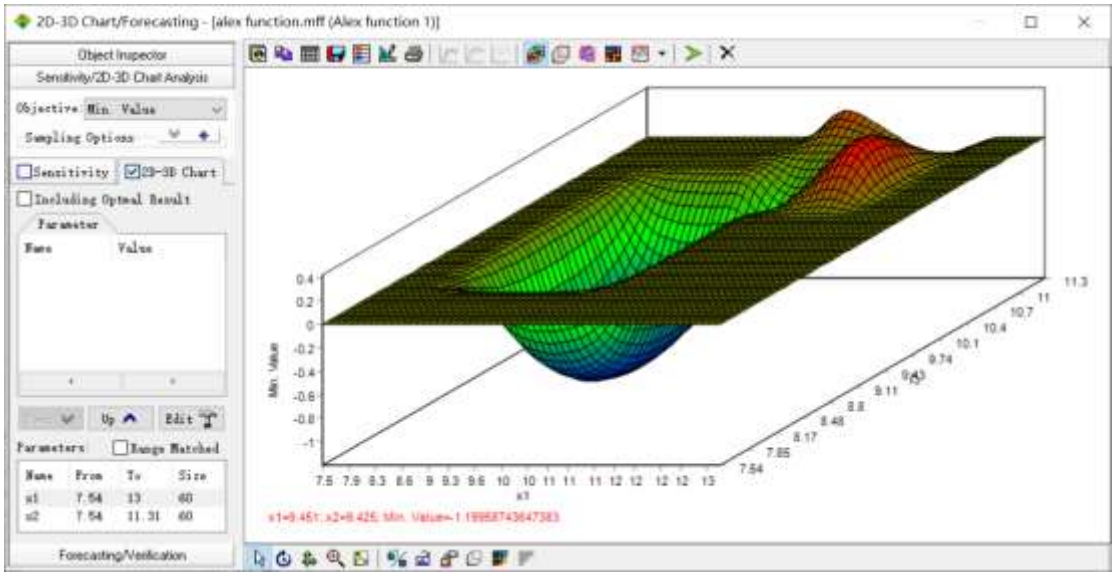


图-5：1stOpt 三维图

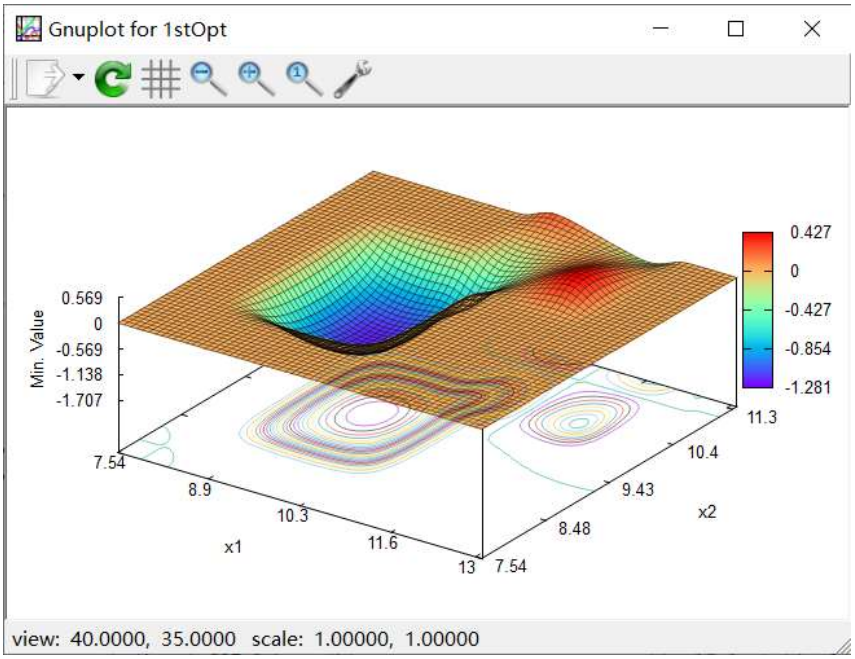


图-6：GnuPlot 对应三维图

4: 代码转换功能

1stOpt 快捷模式代码可一键转换至著名的 GAMS、Lingo、Baron、Matlab、Mathematical 和 Maple 的对应代码，极大方便用户对同样问题采用不同的工具进行对比测试。

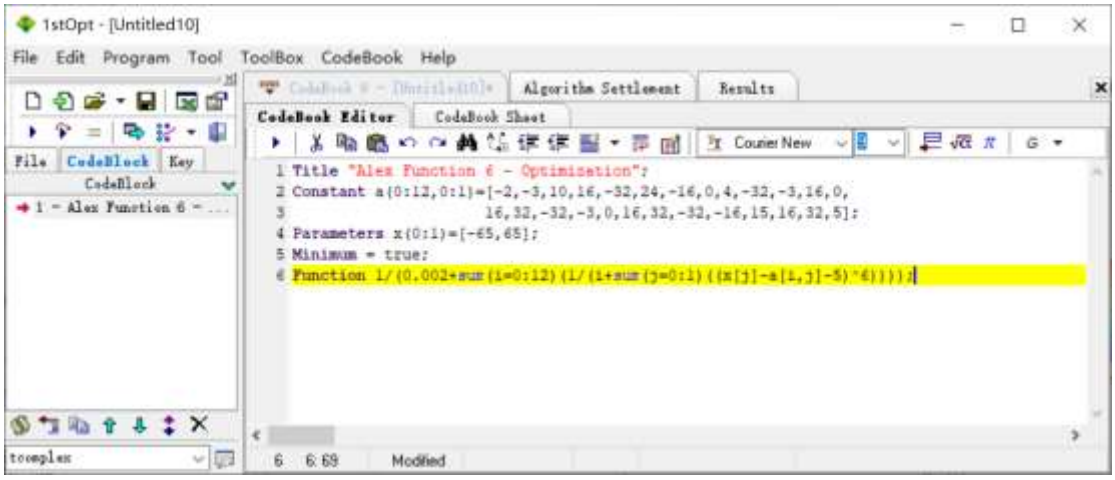


图-7 1stOpt 优化求解代码

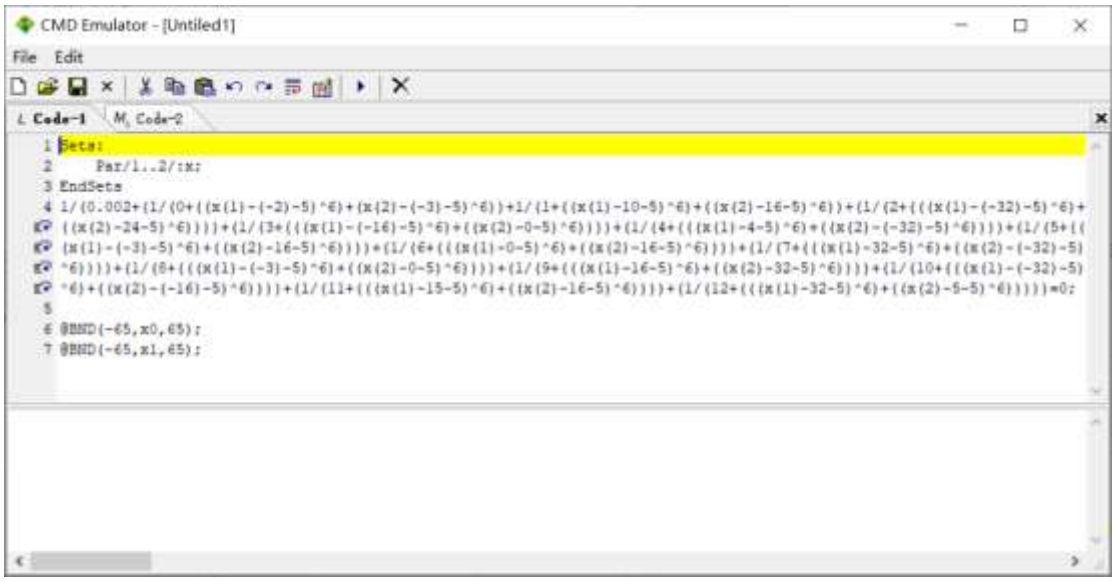


图-8 自动转换为 Lingo 优化求解代码

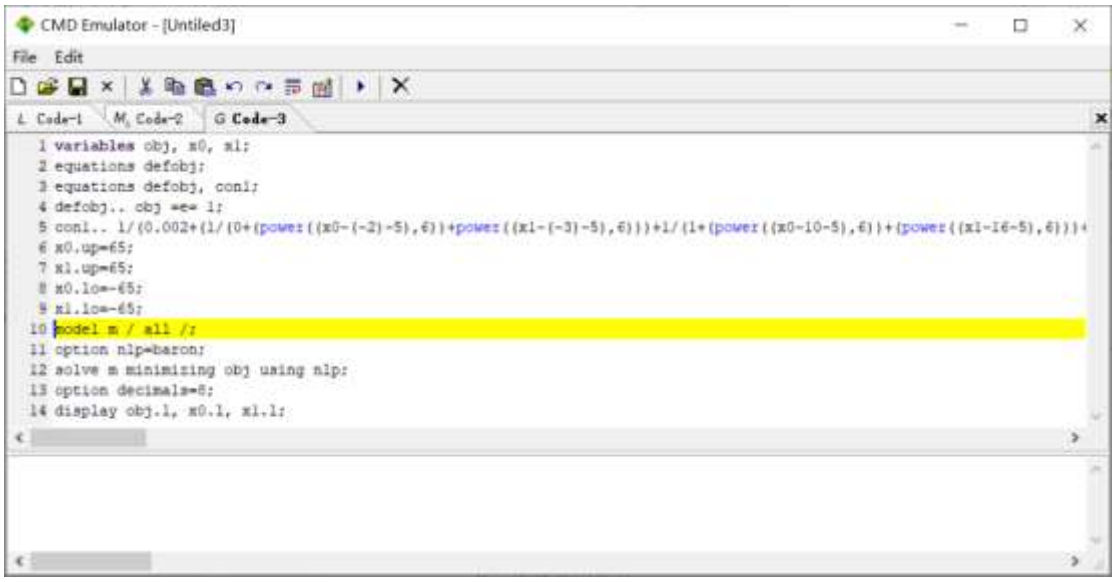


图-9 自动转换为 GAMS 优化求解代码

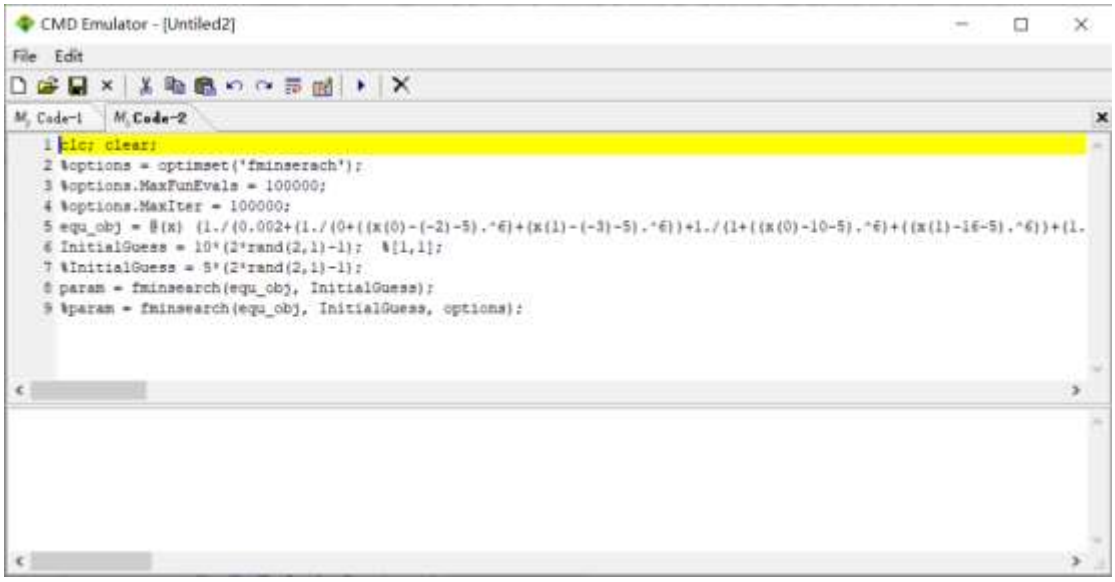


图-10 自动转换为 Matlab 优化求解代码

5: 线性模型求解可直接处理目标函数含有 abs(), MinMax()和 MaxMin()的非线性函数

求解线性规划问题时，如果目标函数含有绝对值 abs()、最大最小 MinMax()或最小最大 MaxMin()等非线性函数，1stOpt 可以直接在后台自动把问题转换为标准的线性问题从而使问题可以采用线性身份处理。

案例-1：目标函数有绝对值函数

$$Min.\ |x_1| + 2\cdot|x_2| + |x_3| + 4\cdot|x_4|$$

$$S.T.\ \begin{cases} x_1 - x_2 - x_3 + x_4 = 12 \\ x_1 - x_2 + 6\cdot x_3 - 3\cdot x_4 = 1 \\ x_1 - x_2 - 2\cdot x_3 + 3\cdot x_4 = -\dfrac{1}{2} \end{cases}$$

案例-2：目标函数有最大最小函数

$$MinMax\ \big((x + 2\cdot y + 10), (3\cdot x + y + 1)\big)$$

$$S.T.\ \begin{cases} 0 \le 3\cdot x + y \le 10 \\ x, y \in [-2,2] \end{cases}$$

上述问题因为目标函数含有非线性函数，之前版本只能按非线性模型去求解，也即只能采用非线性优化算法，现在可以直接采用线性算法（LP）。虽然这两个问题因为求解规模小，非线性与线性两种方式计算效果都一样，当求解规模较大，线性化求解在效率和效果方面就会优势明显。

案例-1 代码及结果

代码	Algorithm = LP;

	MinFunction abs(x1)+2*abs(x2)+abs(x3)+4*abs(x4);
	x1-x2-x3+x4=12;
	x1-x2+6*x3-3*x4=1;
	x1-x2-2*x3+3*x4=-1/2;

结果	====== Output Results ====== Iterations: 12 Elapsed Time (Hr:Min:Sec:Msec): 00:00:00:06 Algorithms: Simplex Linear Program Objective Function(Min.): 61.2500000000104 Best Estimated Parameters: x1: 14.6500000000028 x2: 0 x3: -7.19999999999718 x4: -9.8500000000014 Constrained Functions: 1: x1-x2-x3+x4-(12) = -1.3944401189292E-12 2: x1-x2+6*x3-3*x4-(1) = 2.39772646182246E-11 3: x1-x2-2*x3+3*x4-(-1/2) = -7.02726765666739E-12

案例-2 代码及结果

	代码	结果
非线性	Parameter [x,y]=[-2,2]; ConstStr f1=x+2*y+10, f2=3*x+y+1; MinMax (f1,f2); 0<=3*x+y<=10;	Algorithms: Universal Global Optimization(UGO1) MinMax: 6.66666666666667 x: 0.666666666666667 y: -2 MinMax Function: 1: ((x+2*y+10)) = 6.66666666666667 2: ((3*x+y+1)) = 1 Constrained Functions: 1: 0-(3*x+y) = 0 2: 3*x+y-10 = -10
线性	Algorithm = LP; Parameter [x,y]=[-2,2]; ConstStr f1=x+2*y+10, f2=3*x+y+1; MinMax (f1,f2); 0<=3*x+y<=10;	Algorithms: Simplex Linear Program Objective Function(Min.): 6.66666666666689 Best Estimated Parameters: x: 0.66666666666689 y: -2 Constrained Functions: 1: 0-(3*x+y) = -6.71018796083445E-13 2: 3*x+y-10 = -9.99999999999933

6: 每一次计算结果均可保存并进行预测计算（Result List）

每个代码本的每一次计算结果都可以自动保留，并可在任意时候进行预测验证、二维-三维分析及参数灵敏度计算，还可选择是否保存至主文件（.mff）以便将来打开.mff 文件时亦可再现之前的计算结果。

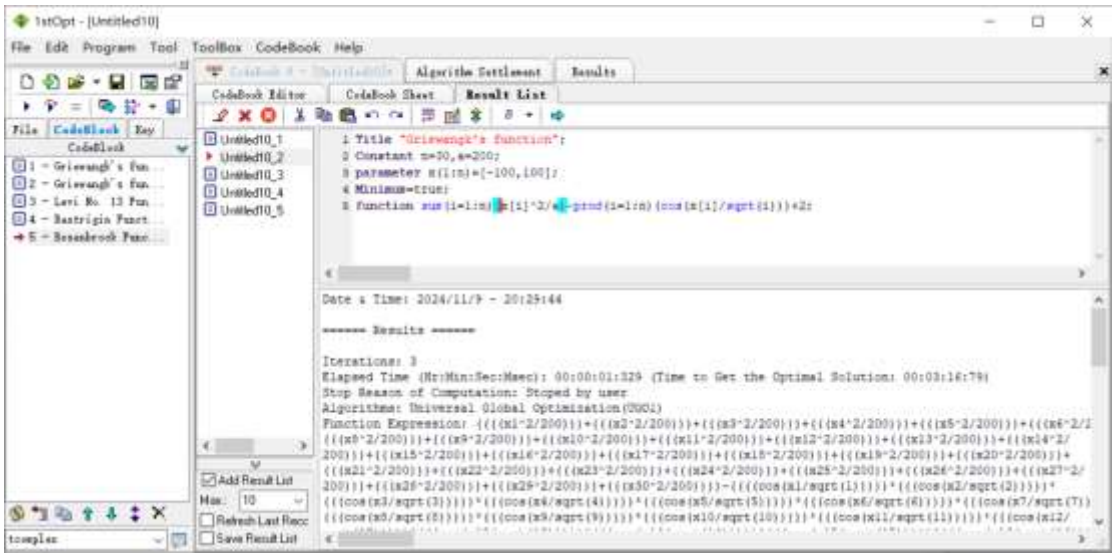


图-11 计算结果自动记录保存

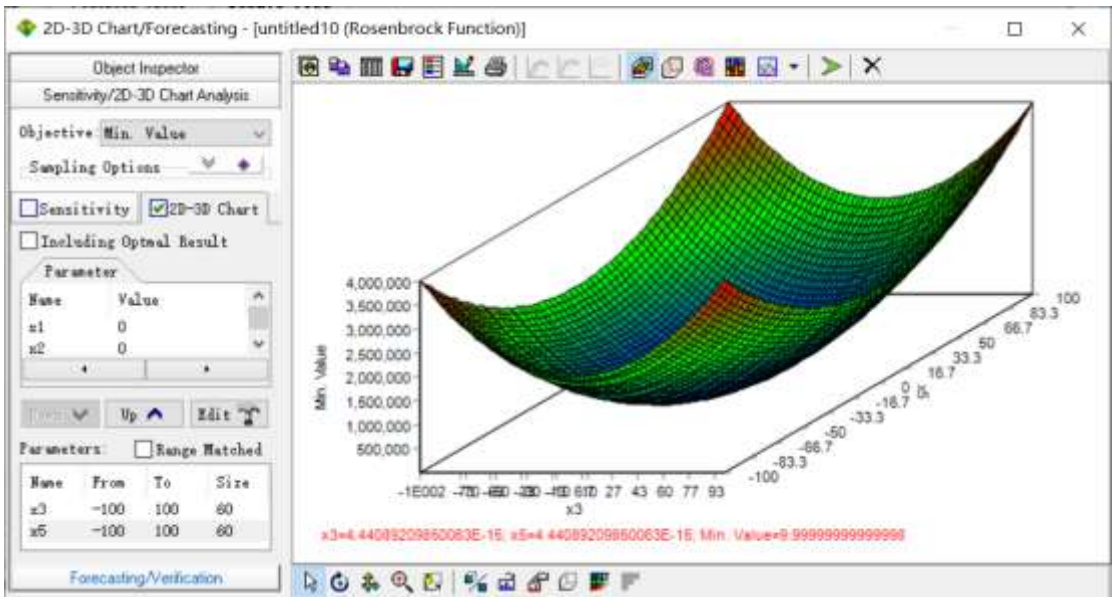


图-12 记录计算结果二维-三维分析

7: 编程模式下复数计算功能提升

Pascal 编程模式复数计算功能的改进，可以对复数直接进行加减乘除等计算，同时引入新的复数函数“cmplx()”，功能与 Fortran 相同函数类似。

案例：已知复数拟合方程如下，i 为虚数符号，y 为复数因变量，a、b、c 为待求实数型参数。

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

案例数据

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	3,2.88462,2.58621,2.20588,1.82927,1.5,1.22951,1.01351,0.8427,0.70755,0.6,0.5137
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

快捷模式代码：

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

Pascal 编程模式代码：

```
ComplexStr = i;
Variable x,y1[OutPut],y2[OutPut];
parameter A,B,C;
StartProgram [Pascal];
Procedure MainModel;
var j: integer;
    ty: TComplex;
Begin
    for j := 0 to DataLength - 1 do begin
        ty := A/(1+(x[j]*B)^2) - i*(A*C)/(1+(x[j]*B)^2);
        y1[j] := ty.x;
        y2[j] := ty.y;
    end;
End;
EndProgram;
Data;
0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.11;
3,2.88462,2.58621,2.20588,1.82927,1.5,1.22951,1.01351,0.8427,0.70755,0.6,0.5137;
-0.789,-0.660,-0.648,-0.511,-0.440,-0.418,-0.338,-0.279,-0.237,-0.218,-0.177,-0.136;
```

上述代码中的“ty := A/(1+(x[j]*B)^2) - i*(A*C)/(1+(x[j]*B)^2);”也可以改为“ty := cmplx(A/(1+(x[j]*B)^2), -1*(A*C)/(1+(x[j]*B)^2));”

Fortran 编程模式代码：

```
ComplexStr = i;
Variable x,y1[OutPut],y2[OutPut];
parameter A,B,C;
StartProgram [Fortran];
Subroutine MainModel
Complex(8) :: ty
integer :: j
real(8) td
    do j = 0, DataLength - 1
        ty = cmplx(A/(1+(x(j)*B)^2), -1*(A*C)/(1+(x(j)*B)^2))
        y1(j) = real(ty)
        y2(j) = aimag(ty)
    end do
```

```
End Subroutine
EndProgram;
Data;
0,0.01,0.02,0.03,0.04,0.05,0.06,0.07,0.08,0.09,0.1,0.11;
3,2.88462,2.58621,2.20588,1.82927,1.5,1.22951,1.01351,0.8427,0.70755,0.6,0.5137;
-0.789,-0.660,-0.648,-0.511,-0.440,-0.418,-0.338,-0.279,-0.237,-0.218,-0.177,-0.136;
```

上述三段代码求解结果一致，而且快捷模式更显简单方便，但如果模型比较复杂，有较多的中间环节计算，必须使用编程模式时，无疑新增直接复数计算功能会方便很多。

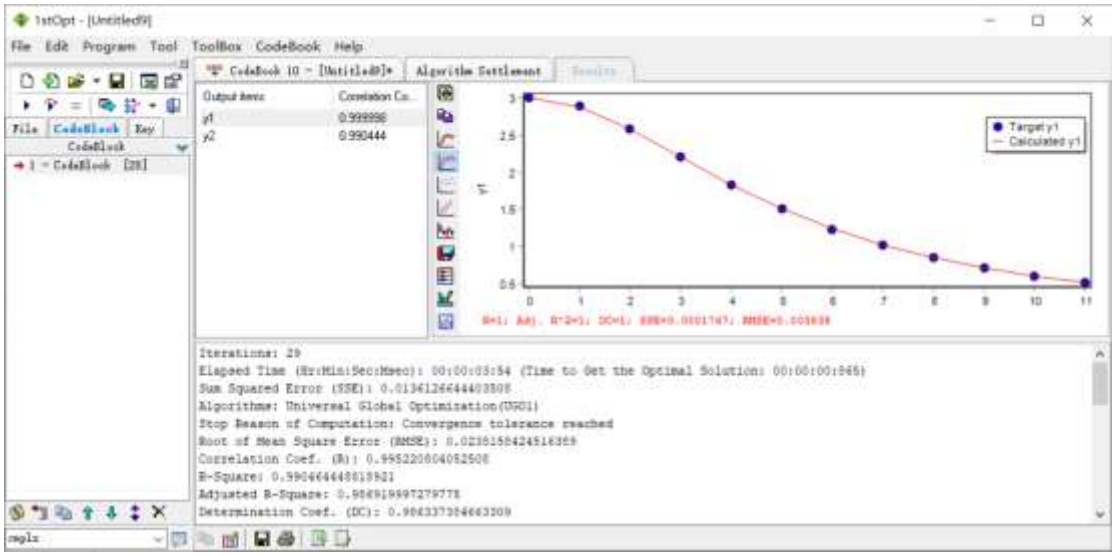


图-13 复数拟合计算

8: 新增关键字“AddResultList”，“SaveResultList”，“Exit”

- AddResultList：是否添加计算结果记录；
- SaveResultList：是否保存计算结果记录；

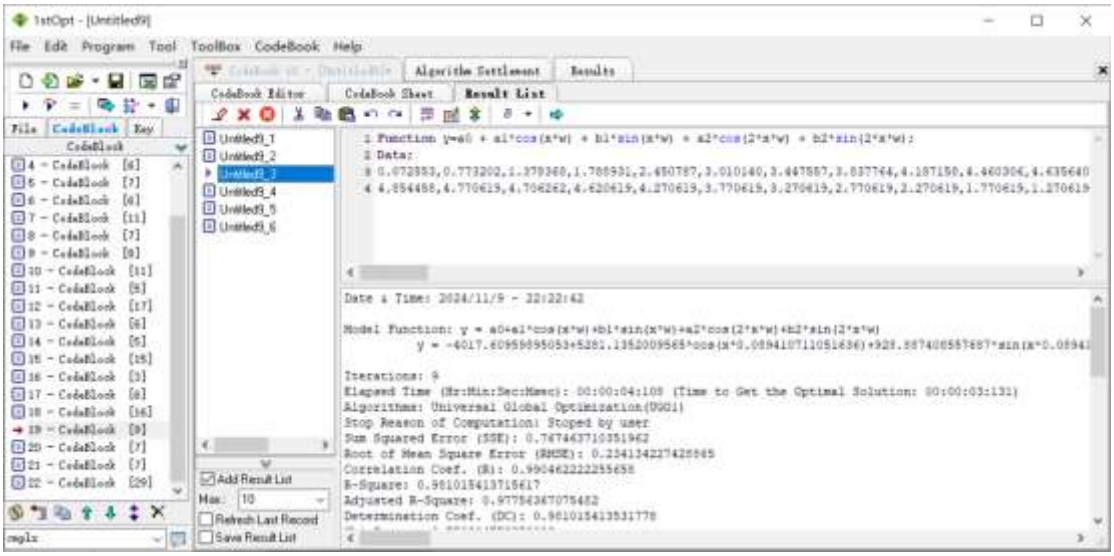


图-14 “AddResultList”和“SaveResultList”对应的设置

- Exit：代码中该关键字之后的所有代码将被忽略。

```
Function y=p1*exp(p2*(x-p6)^2)+(ln(1-p3*x)+p5)^p4;
Data;
0.1   1009.215
0.2   1019.867
0.3   1030.957
0.4   1042.484
0.5   1054.448
0.6   1066.85
0.7   1079.689
0.8   1092.966
0.9   1106.68
Exit;
1     1120.832
1.1   1135.422
1.2   1150.449
1.3   1165.913
1.4   1181.816
1.5   1198.156
1.6   1214.933
```

9: 直接求解导数方程

Diff()函数可直接对含初级函数的表达式求导。如对表达式“x^4+sin(x)”求一阶导数：diff(x^4+sin(x),x)，返回：4*(x^3)+cos(x)；求二阶导数：diff(x^4+sin(x),x,2)，返回：4*(3*(x^2))+(-sin(x))。

导数方程-1：

代码	ConstStr j=(3-k1+2*k2)/(2*k2*(3+2*k1-k1^2-4*k2)); Function diff(j,k1)-9; diff(j,k2)-70;
结果	Objective Function (Min.): 1.78870265954338E-23 k2: -0.0918220762459558 k1: -1.84238285017731

导数方程-2：

代码	ConstStr v=1/2*(2*10^3)*(0.5*cos(t))^2-0.75*9.81*20*cos(t),
	dv=diff(v,t),d2v=diff(v,t,2);
	Parameter t=[0,pi];
	Function dv = 0 ;
	d2v >= 0;

结果	Objective Function (Min.): 8.07793566946316E-28 t: 1.27207332819662 Constrained Functions: 1: -(((1000*((2*(0.5*((-sin(t)))))*(0.5*((-sin(t))))+(2*(0.5*cos(t)))*(0.5*((-(cos(t))))))-147.15*((-(cos(t)))))) - (0)) = -456.693755

10: 支持“Subs()”函数

支持类似于 Matlab 的 Subs()函数，如“Subs(sin(x)*y-c=0,c,2.5)”返回“sin(x)*y-2.5=0”。

示例：

代码	Constant Tin=915,CR=1,xe=1; ConstStr eq=(xe*k^n*((Tin-T)/CR)^n)/(1+k^n*((Tin-T)/CR)^n)=x, eq1=subs(eq,T,888), eq1=subs(eq1,x,5.43/100), eq2=subs(eq,T,840), eq2=subs(eq2,x,73/100); Function eq1; eq2;
结果	Objective Function (Min.): 8.05057205681331E-27 k: 0.0173582081407126 n: 3.77039009302756

11: 新增加了超级加密狗版，提供更多的类型选择；

软件版本类型除了单机版、加密狗版及网络版外，新增加了超级加密狗版，该版本一个加密狗对应多台电脑，这些电脑接入加密狗激活一次，可在 90 天内无需接入加密狗即可正常运行，也即多台电脑可同时运行。与单机版相比，用户多的化性价比会更高；与加密狗版相比，多台电脑可同时使用，无需一直要接入加密狗；与网络版相比无需局域网连接。

12: 非连续非线性优化求解问题性能改进与提升

13: 修正了众多用户反馈/发现的 Bugs