B5: 全局优化应用

优化介绍

优化（optimization）的目标是从一组可能的答案中发现问题的最佳解。答案通过使用一个或多个问题变量的实际值目标函数（objective function）进行对比。可能的集合（feasible set）由约束条件（constraints）决定，约束条件通常是关于问题变量的不等式或方程（组）。数学意义上，目的是发现目标函数的最大值（maximizes）或最小值（minimizes）、同时满足（satisfying）约束条件的点，这个点称为极值（extremum）。优化问题定义如下。

$f(x)$ 的最大值（或最小值

约束条件 $g_i(x) \le 0, i = 1..p,$

和 $h_i(x) = 0, i = 1..m$

这里, $x \in R^n$

优化模型由$f(x), g_i(x),$ 和 $h_i(x)$的结构分类。如果所有的函数是 x 线性函数，这个模型是一个线性规划（linear program）。如果 $f(x)$ 是 x 的二次函数，以及 $g_i(x)$ 和 $h_i(x)$ 是 x 的线性函数，这个模型是一个二次规划（quadratic program）。如果 $f(x)$ 是一个平方和函数，这个模型是一个非线性规划（least-squares problem）。对于其他任意结构，模型称为非线性回归（NLP）。Maple 9.5中的优化程序包（Optimization）提供了一系列算法分别求解这些类型的问题。

传统上，优化研究集中在局部搜索算法（local）。局部搜索的目的是发现 f(x) 在可行区域内的局部极值。从一个初始点出发，局部搜索算法迭代（iteratively）搜索当前点领域中的一个点，提高目标函数值，同时维持或逼近可行性、使用当前点上关于$f(x), g_i(x),$ 和 $h_i(x)$的迭代信息。理想情况下，搜索会终止于一个可行的局部极值。优化算法的不同决定它们如何衡量逼近可行，以及它们如何搜索。

当$f(x)$在可行区域内有唯一的局部极值时，局部搜索是有效的，这是因为搜索发现的局部解是问题的全局解。如果$f(x), g_i(x)$都是凸函数（convex），并且所有的$h_i(x)$是仿射函数（affine）时，极值是唯一的。

当 $f(x)$在可行区域内有许多局部极值，局部搜索难以完成。在这种情况下，局部极值的发现依赖于起始点，但可能并不是全局解。当任一个$f(x), g_i(x)$ 是非凸函数，或者任一个$h_i(x)$ 是非线性函数时，存在多个局部最小值。非凸性经常存在于现实问题中，可能是求解优化问题最大的障碍。全局优化（Global）是最新的和最成功方法，能够克服这个障碍。

1. 全局优化的优势

优化程序包（Optimization）中的优化算法是局部搜索方法。局部搜索方法使用一次和二次导数发现局部极值，反复迭代生成点提高目标函数同时维持或逼近可行性。

全局优化工具箱（Global Optimization toolbox）使用了全局搜索方法。不同于使用求导发现局部极值，它们系统地搜索整个可行区域寻找一个全局极值。对于可能有许多局部极值的优化模型，全局搜索方法寻找一个全局极值。这些方法并不依赖于初始点。

尽管全局搜索不依赖于起始点，但它需要所有决策变量的有限边界，限定搜索空间。同时相比局部搜索方法，随着变量数的增加，全局搜索方法的计算工作量会急剧增加。这种情况通常并不明显，除非变量的数目达到上百个。

为了了解全局优化相比局部优化的优势，考虑下面的问题，对比优化程序包（Optimization）中的局部非线性求解器NLPSolve，和全局优化工具箱（Global Optimization toolbox）中的 GlobalSolve 命令。

1. 全局优化工具箱介绍

概述：

全局优化工具箱提供世界级的全局优化技术求解优化模型的最优解，鲁棒和高效。

该工具箱的功能包括：

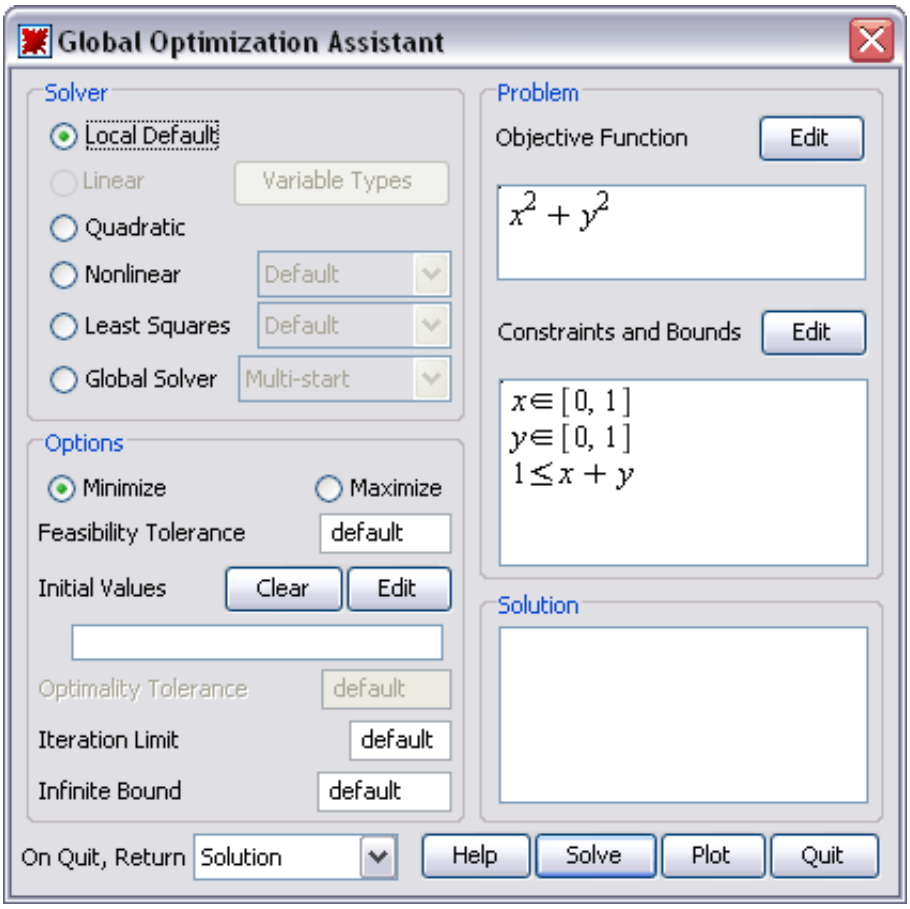
- 提供多个求解器模块求解非线性优化问题，包括branch-and-bound全局搜索、全局自适应随机搜索、全局随机多起点搜索、和一个广义简约梯度局部搜索。
- 可以求解包含数千个变量和约束的模型，利用Maple任意精度计算的功能，大大地降低了数值不稳定性。
- 支持任意类型的目标函数和约束条件，包括预定义的特殊函数（例如，贝赛耳、超几何分布）、导数和积分、分段函数、以及用Maple语言编写的程序体。
- 交互式的全局优化助手，提供简单易用的图形界面，帮助您输入问题和选择求解方法。
- Maple内置的模型可视化功能，可以观察目标函数的一维和二维子空间投影图，以及通过目标表面上的面或线图形显示约束函数。

初始化：

为了使用Global Optimization Toolbox，需要首先使用下面的命令加载GlobalOptimization程序包。

```
> restart;
  with(Optimization):
  with(GlobalOptimization):
```

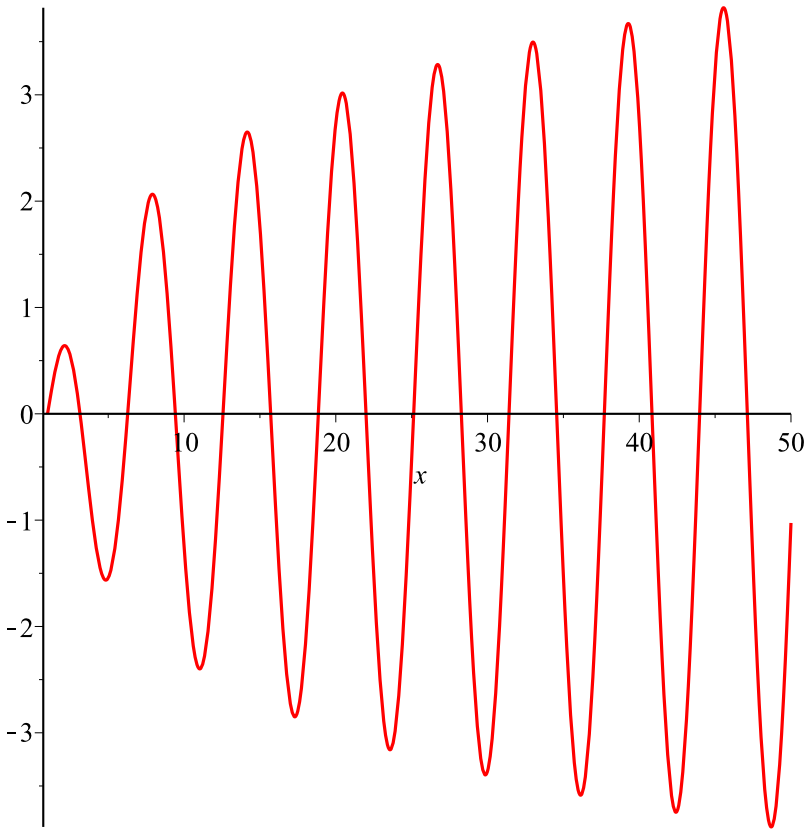
您可以选择使用命令求解预定义的优化模型，或者使用全局优化助手，如下图所示。



使用全局优化工具箱求解问题

对于许多优化问题，简单的方法并不能足以发现最优解。它们仅能发现局部极值，通常是最靠近用户给定的搜索起始点的极值点。

```
> interface(warnlevel = 0):
  with(plots):
> plot(ln(x)·sin(x), x = 1..50)
```



您也许可以通过观察图形发现给定区域内的全局最小值。但如果您不能正确地求它的近似值，使用常规的优化算法并不能发现全局最小值。

```
> Optimization[Minimize](ln(x)·sin(x), x = 1 ..50)
                [-3.39618690740209, [x = 29.8549920107437]]
```
(2.2.1)

根据 Minimize 命令的计算结果，最小值发生在 x = 30。但是您可以通过上面的图形发现它并不是全局最小值。

通过使用全局优化工具箱中有等效功能的命令，可以确保发现指定区间内的全局最小值。

```
> with(GlobalOptimization) :
> GlobalSolve(ln(x)·sin(x), x = 1 ..50)
                [-3.88562417153593120, [x = 48.6999705692544]]
```
(2.2.2)

为了查看使用了何种方法发现全局最小值，改变 infolevel 变量，然后重新执行命令。设置 infolevel = 3，命令显示输入模型的大小和类型，求解器运行模式和参数，以及详细的运行时间信息。

```
> infolevel[GlobalOptimization] := 3 :
> GlobalSolve(ln(x)·sin(x), x = 1 ..50)
```

```
GlobalSolve: calling NLP solver
SolveGeneral: calling global optimization solver
SolveGeneral: number of problem variables 1
SolveGeneral: number of nonlinear inequality constraints 0
SolveGeneral: number of nonlinear equality constraints 0
SolveGeneral: method multistart
SolveGeneral: merit function evaluation limit 1000
SolveGeneral: non-improving merit function evaluation limit 200
SolveGeneral: constraint penalty multiplier 100.0
SolveGeneral: target merit function value -0.10e11
SolveGeneral: local search target objective function value -0.10e11
SolveGeneral: local search feasibility tolerance 0.10e-5
SolveGeneral: local search optimality tolerance 0.10e-5
SolveGeneral: time limit in seconds 100
SolveGeneral: trying evalhf mode
SolveGeneral: total number of function evaluations 1120
SolveGeneral: runtime in external solver 0.
SolveGeneral: maximum constraint infeasibility 0.
SolveGeneral: cycling or stall detected in solver
```

$$[-3.88562417153593120,\ [x = 48.6999705692544]] \qquad (2.2.3)$$

通常，infolevel **设置为默认值** 0。

```
> infolevel[GlobalOptimization] := 0 :
```

下面是一个例子，使用线性方法不能发现全局最小值，但全局求解器可以发现最优值。

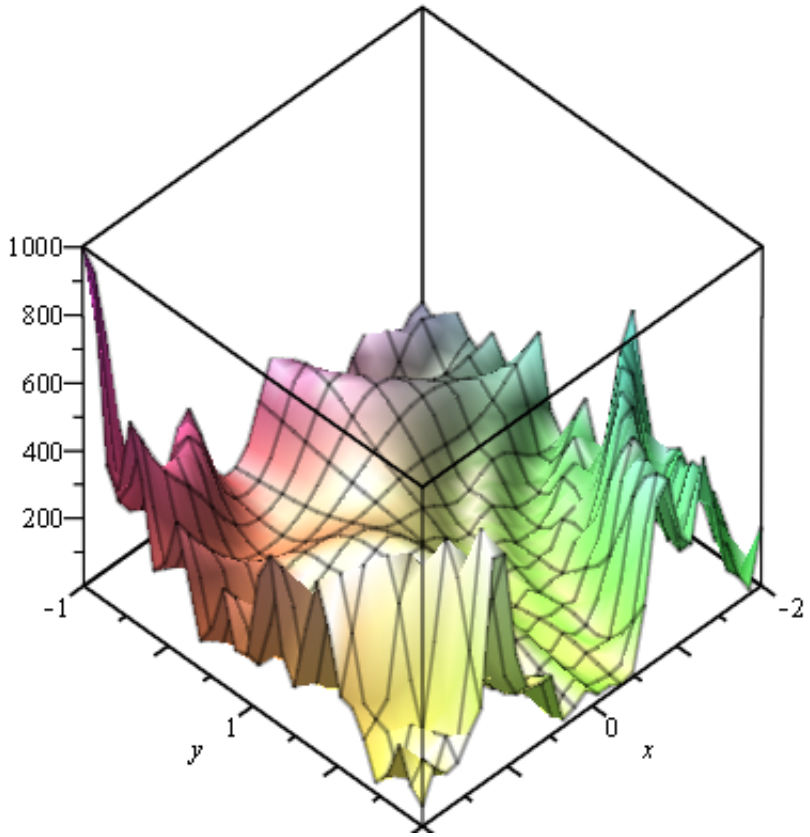
例子：

考虑一个非线性方程组：

```
> eq1 := sqrt(x^2 + y^4) + e^(x - y^2) + 5 sin(2 x - 4 x y) - 12 cos(x y) :
> eq2 := 5 ln(1 + x^2) + e^(-y + x) + 5 sin(6 x y) :
```

最小二乘误差函数有多个极值。

```
> plot3d(eq1^2 + eq2^2, x = -2 ..2, y = -1 ..3, grid = [30, 30], lightmodel = light4, axes = boxed)
```



为了求最小二乘误差的全局最小值，定义要优化的目标函数和约束。

```
> objf := eq1^2 + eq2^2;
```

$$objf := \left(\sqrt{x^2+y^4} + e^{x-y^2} - 5\sin(-2x+4xy) - 12\cos(xy)\right)^2 + \left(5\ln(1+x^2) + e^{-y+x} + 5\sin(6xy)\right)^2 \quad (2.2.4)$$

```
> cons := {eq1, eq2};
```

$$cons := \left\{5\ln(1+x^2) + e^{-y+x} + 5\sin(6xy), \sqrt{x^2+y^4} + e^{x-y^2} - 5\sin(-2x+4xy) - 12\cos(xy)\right\} \quad (2.2.5)$$

首先，使用Maple内置的 Optimization 程序包求局部解。由于方程组的复杂性，线性优化求解器并不能发现可行的解。

```
> localsoln := Optimization[Minimize](objf, {eq1 = 0, eq2 = 0}, x = -1..2, y = -2..1)
Error, (in Optimization:-NLPSolve) no improved point could be found
```

然而，全局优化算法可以发现全局最小值。

```
> sol := GlobalSolve(objf, {eq1 = 0, eq2 = 0}, x = -1..2, y = -2..1):
  sol[1]; sol[2]
```

$$5.24626533431424086\ 10^{-23}$$

$$[x = -0.558972497937727,\ y = -1.58234523718628] \qquad (2.2.6)$$

代入约束函数，结果显示最小二乘逼近是相当精确的解。也就是说，发生在该最小点上的最小二乘逼近的误差非常小。

```
> eval(cons, sol[2])
```

$$\{2.91588975187551\ 10^{-12},\ 6.63025190306143\ 10^{-12}\} \qquad (2.2.7)$$

通过了解上面的例子，您可以有信心使用全局优化工具箱求解任意复杂的数学问题。

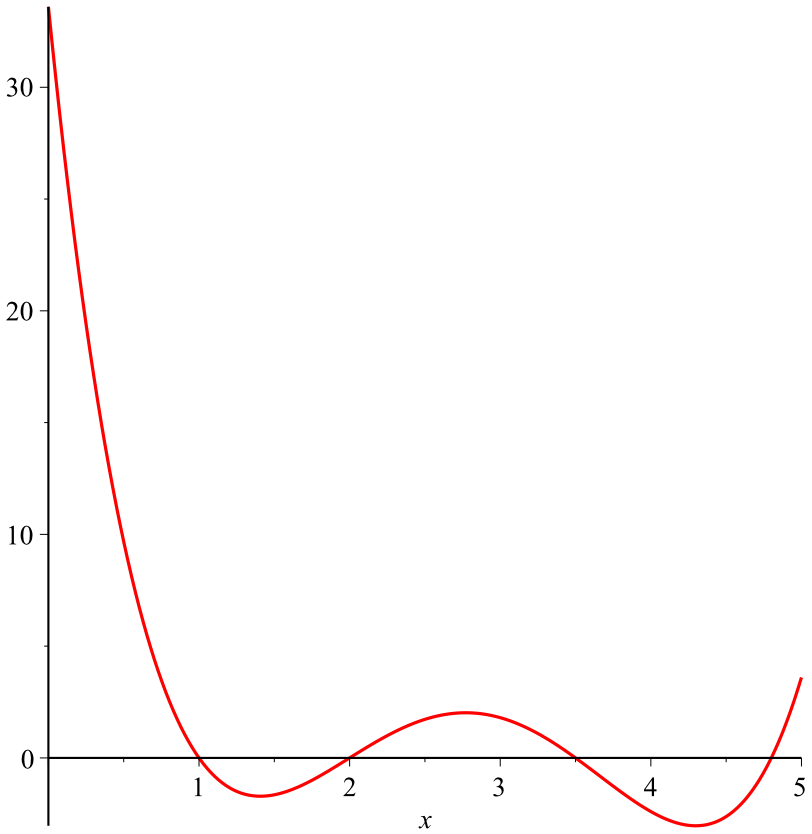
四次多项式

求四次多项式在区间 [0, 5] 上的全局最小值。注意该函数是一个非凸函数，有两个局部最小值。

```
> poly := (x-1) * (x-2) * (x-3.5) * (x-4.8);
```

$$poly := (x-1)\ (x-2)\ (x-3.5)\ (x-4.8) \qquad (2.3.1)$$

```
> plot( poly, x = 0 ..5 );
```



取决于给出的起始点，内置的局部 NLP 求解器返回位于 $x = 4.298$ 处的全局最小值，或者是位于 $x = 1.407$ 处的次优局部最小值。

```
> NLPSolve( poly, initialpoint = [x = 5] );
  NLPSolve( poly, initialpoint = [x = 3] );
  NLPSolve( poly, initialpoint = [x = 2] );
  NLPSolve( poly, initialpoint = [x = 0] );
```

$$[-1.71396627012779357, [x = 1.40679933068723]]$$
$$[-3.03603863879697267, [x = 4.29790996157305]]$$
$$[-1.71396627012779401, [x = 1.40679932978431]]$$
$$[-1.71396627012779468, [x = 1.40679933068591]] \qquad (2.3.2)$$

全局优化求解器总是求全局最小值。注意到调用格式并不依赖于起始点，但需要 x 的有限范围，这个问题定义为 0..5。

```
> GlobalSolve( poly, x = 0..5 );
```

$$[-3.03603863879697311, [x = 4.29790996175692]] \qquad (2.3.3)$$

非凸二次函数

求一个二维非凸二次函数在矩形区域上的全局最小值。

```
> with( LinearAlgebra ) :
```

使用非正定矩阵构建非凸函数。

```
> Q := Matrix( <<1.9, -3.5>|<5, -1/3>> );
```

$$Q := \begin{bmatrix} 1.9 & 5 \\ -3.5 & -\frac{1}{3} \end{bmatrix} \qquad (2.4.1)$$

检查该矩阵是否是正定的。

```
> IsDefinite( Q );
```

$$\text{false} \qquad (2.4.2)$$

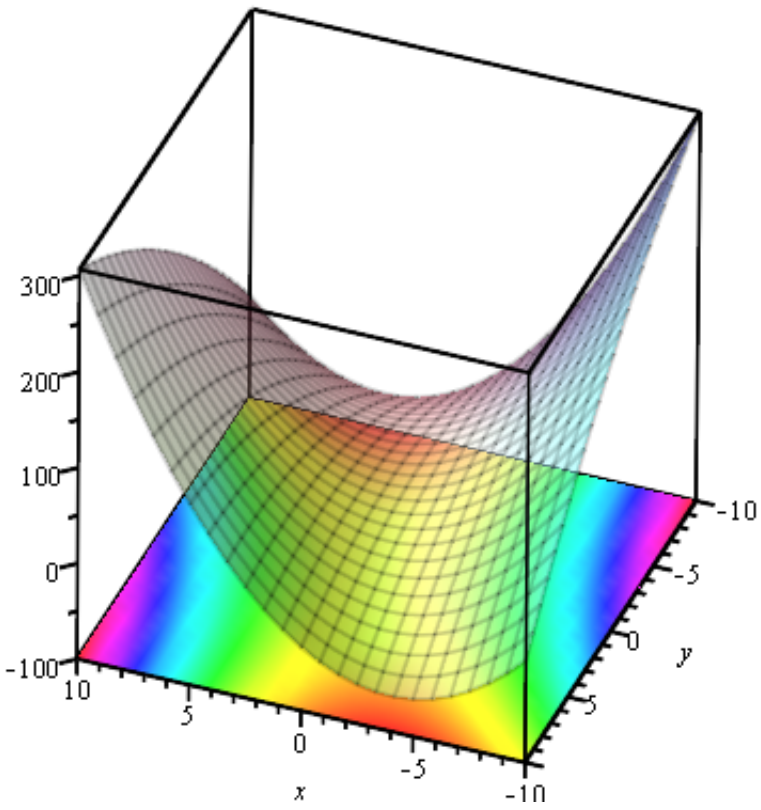
构建函数 $x^t \ Q \ x$。

```
> NonconvexQuadratic := expand( Transpose( <x, y> ) . Q . <x, y> );
```

$$NonconvexQuadratic := 1.9\,x^2 + 1.5\,x\,y - \frac{1}{3}\,y^2 \qquad (2.4.3)$$

对函数绘图并画出等高线图。全局最小值发生在[3.947, -10.000] 和 [-3.947, 10.000]，鞍点发生在 [0, 0]。

```
> p1 := plot3d( NonconvexQuadratic, x = -10..10, y = -10..10, axes = boxed, transparency
        = .7, style = patch) :
  p2 := plot3d( -100, x = -10..10, y = -10..10, style = patchnogrid, color
        = NonconvexQuadratic ) :
  plots[display]( p1, p2, orientation = [125, 60] );
```



从大部分起始点出发，Maple局部优化求解器会发现发生在[3.947, -10.000]上的全局最小值。

```
> NLPSolve( NonconvexQuadratic, x = -10..10, y = -10..10, initialpoint = [x = .1, y = 0]);
```

$$[-62.9385964912280614,\ [x = 3.94736842105263,\ y = -10.]] \qquad (2.4.4)$$

如果起始点选择不佳，局部优化求解器会停止在鞍点 [0, 0] 上。求解诶其停止的原因是它发现在当前点是零梯度。鞍点有零梯度但不是极值。

```
> NLPSolve( NonconvexQuadratic, x = -10..10, y = -10..10, initialpoint = [x = 0, y = 0]);
```

$$[0.,\ [x = 0.,\ y = 0.]] \qquad (2.4.5)$$

全局优化求解器总是发现两个全局最小值中的一个。必须定义x和y的有限范围。

```
> GlobalSolve( NonconvexQuadratic, x = -10..10, y = -10..10);
```

$$[-62.9385964912280684,\ [x = 3.94736842105263,\ y = -10.]] \qquad (2.4.6)$$

非凸可行区域

求一个二维凸函数在一个缺失单位圆盘的平面上的全局最小值，这是一个非凸区域。

```
> obj := x^2 + y^2 - x + y;
```

$$obj := x^2 + y^2 - x + y \qquad (2.5.1)$$

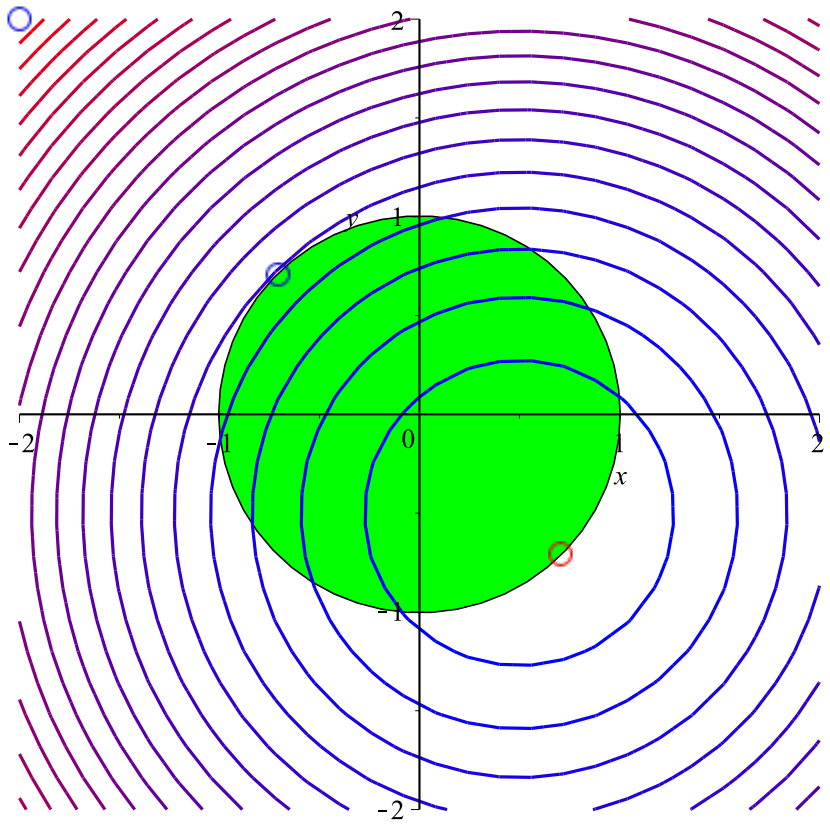
```
> constraint := x^2 + y^2 >= 1;
```

$$constraint := 1 \le x^2 + y^2 \qquad (2.5.2)$$

全局最小值发生在近似点 [.707, -.707]，见下图中的可行区域边界的红圈，同时显示了目标函数的几个等高线。

起点是[-2, 2]，Maple局部优化求解器发现在近似点 [-.707, .707] 处的非最优局部最小值，表示为蓝色。目标函数在[-.707, .707]上的梯度垂直于可行区域的边界，终止了局部搜索。

```
> p1 := plottools[disk]([0, 0], 1, color = green):
  p2 := plots[contourplot](obj, x = -2 ..2, y = -2 ..2, coloring = [blue, red], contours = 20):
  p3 := plots[pointplot]([[.707, -.707]], color = red, symbolsize = 20, symbol = circle):
  p4 := plots[pointplot]([[-.707, .707], [-2, 2]], color = blue, symbolsize = 20, symbol
        = circle):
  plots[display](p1, p2, p3, p4);
```



局部搜索从 [-2, 2] 开始发现次优局部最小值。

```
> NLPSolve( obj, {constraint}, x=-2..2, y=-2..2, initialpoint=[x=-2, y=2]);
```

$$[2.4142135627396621 5, [x = -0.707106781262466, y = 0.707106781262466]] \quad (2.5.3)$$

全局搜索发现全局最小值发生在[.707, -.707]。

```
> GlobalSolve( obj, {constraint}, x=-2..2, y=-2..2);
```

$$[-0.414213564542023516, [x = 0.707106776061393, y = -0.707106781075446]] \quad (2.5.4)$$

高维平坦函数

考虑一个四维函数，这个函数有一个非常平坦、但非定常、靠近全局最小值的区域。局部搜索方法往往发现这个平坦区域，但在发现全局最优值之前终止，原因是函数在该区域上的梯度接近于0。

```
> objWood := 100*(y-x^2)^2+(1-x)^2+90*(z-w^2)^2+(1-w)^2+10.1*((y-1)
      ^2+(z-1)^2)+19.8*(y-1)*(z-1);
```

$$objWood := 100\,(y-x^2)^2+(1-x)^2+90\,(z-w^2)^2+(1-w)^2+10.1\,(y-1)^2 \quad (2.6.1)$$
$$+10.1\,(z-1)^2+19.8\,(y-1)\,(z-1)$$

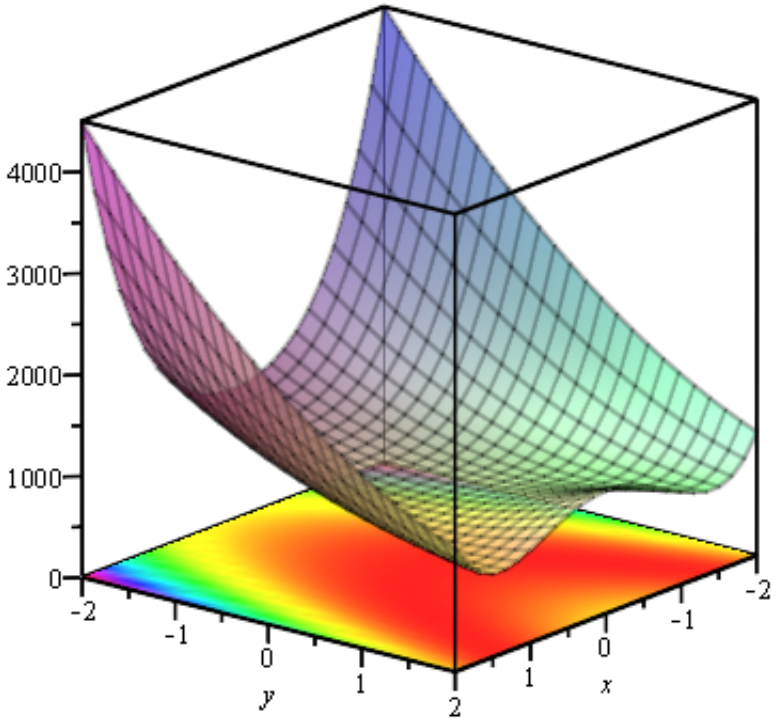
全局最小值发生在 $w=x=y=z=1$。为了查看为什么这个函数难以最小化，考虑它在 $z=1$ 和 $w=1$ 上的2-D投影。

```
> objWoodXY := eval( objWood, { z=1, w=1 } );
```

$$objWoodXY := 100\,(y-x^2)^2+(1-x)^2+10.1\,(y-1)^2 \quad (2.6.2)$$

等高线表明该函数砸原点附近非常平坦。局部搜索方法通常在这样的区域找到局部最小值，并终止搜索。

```
> p1 := plot3d( objWoodXY+800, x=-2..2, y=-2..2, axes=boxed, transparency=.5, style
      =patch):
  p2 := plot3d(0, x=-2..2, y=-2..2, style=patchnogrid, color=objWoodXY):
  plots[display](p1, p2, orientation=[39, 72]);
```



Maple的局部搜索在找到全局最小值之前已经终止，但是全局搜索可以轻松发现最小值。

```
> NLPSolve( objWood, initialpoint = [w = 3, x = -2, y = 2, z = -2] );
  GlobalSolve( objWood, w = -10..10, x = -10..10, y = -10..10, z = -10..10 );

[0.148242645237016024, [w = 1.10990339256281, x = 0.942001104081936, y
      = 0.872155202370579, z = 1.22185294890580]]
[1.46859883390000718 10^-17, [w = 1.00000000100129, x = 0.999999998768496, y
      = 0.999999997733617, z = 1.00000000226602]]
```

(2.6.3)

2. 全局优化的应用

在上一节中，通过几个简单的例子说明了全局搜索方法如何大大优于局部搜索方法。下面是几个应用全局优化解决实际问题的范例。

汽车悬架系统的调校

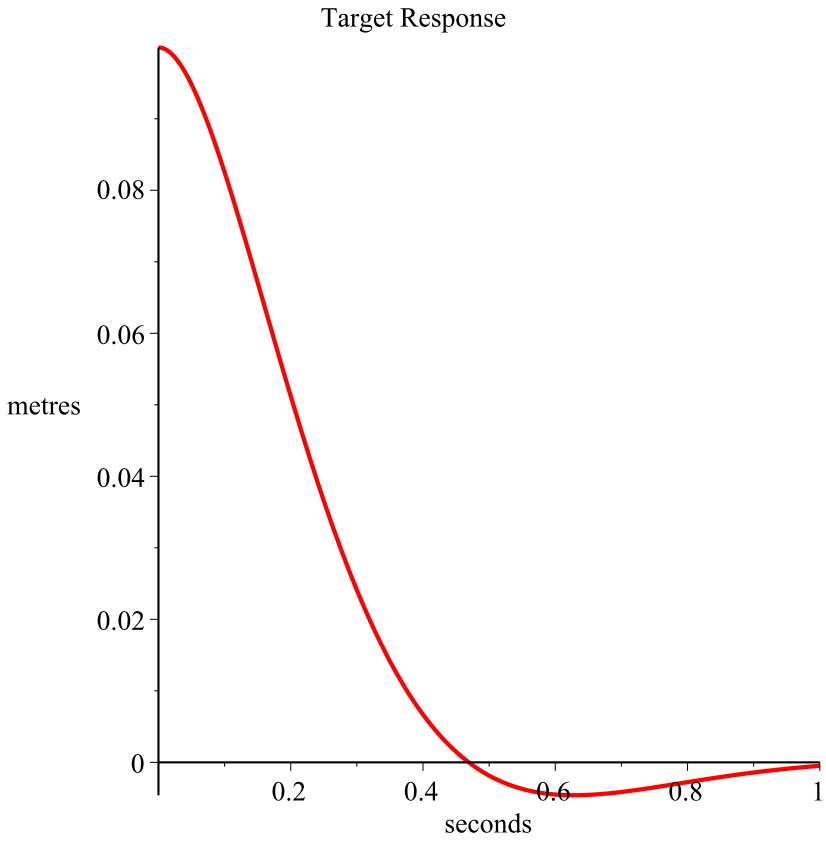
考虑悬架系统设计中的问题，显示路面颠簸时的响应。问题变量是弹簧常数 k 和阻尼常数 b。每个车轮上车重给定，m，以及一个典型的颠簸的期望幅度，求 k 和 b 的值，创建与期望响应匹配的系统响应。

要最小化的目标函数是期望和实际响应之间的平方误差，是关于 k 和 b 在离散时间集上的函数。在通过求解系统的微分方程得到实际响应后，使用优化算法求 k 和 b 的值，最小化误差值。

目标响应

假设目标响应是一个衰减指数函数。当汽车碰到一个幅度为 0.1 米的颠簸时，其振荡幅度呈指数衰减。

```
> restart;
> Target := t → (0.14 * exp( -4.9 * t) * cos(5 * t -.7754)) :
> plot( Target(t), t = 0 ..1, title = "Target Response", labels = ["seconds", "metres"],
      thickness = 2 );
```



计算实际响应

为了得到系统对颠簸的实际响应，求解微分方程组，初始条件是 x(0) = 0.1。

自然状态下质量-弹簧-阻尼系统的微分方程是：

```
> System_Equation := m*diff(x(t), t, t) + b*diff(x(t), t) + k*x(t) = 0;
```

$$System\_Equation := m\left(\frac{d^2}{dt^2}x(t)\right) + b\left(\frac{d}{dt}x(t)\right) + k\,x(t) = 0 \qquad (3.1.2.1)$$

求解微分方程，定义响应为k, b, 和t的函数，质量为m=450 kg。

```
> m := 450 :
> sol := dsolve( {System_Equation, x(0) = .1, D(x)(0) = 0} );
```

$$sol := x(t) \qquad (3.1.2.2)$$

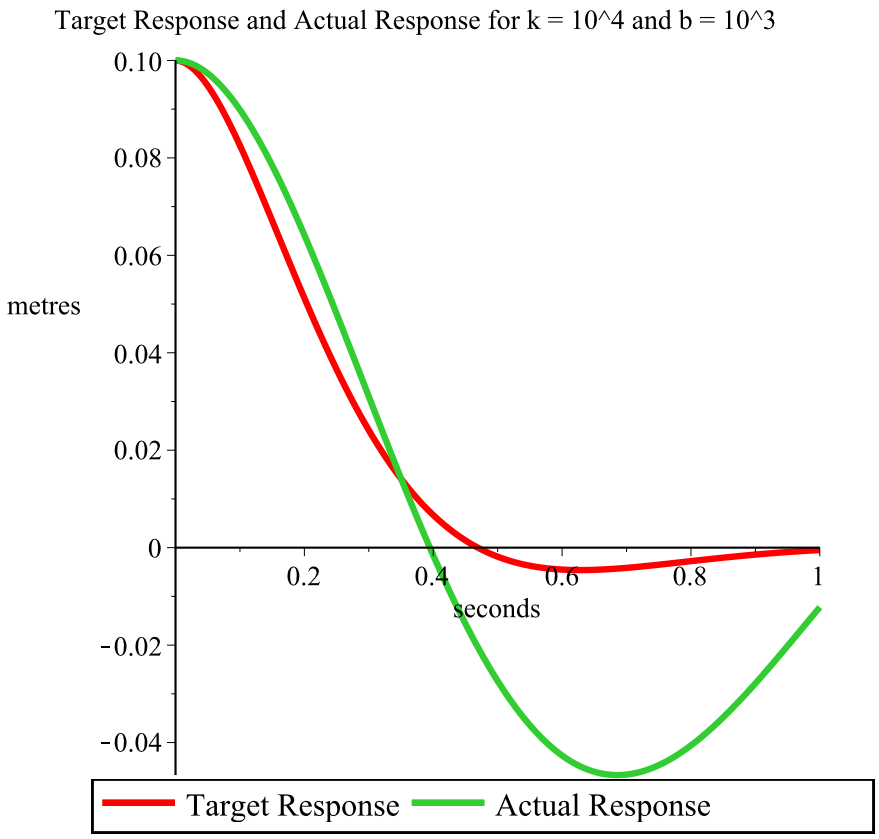
$$= \frac{1}{20}\,\frac{\left(b\sqrt{b^2-1800\,k}+b^2-1800\,k\right)\mathrm{e}^{\left(-\frac{1}{900}b+\frac{1}{900}\sqrt{b^2-1800\,k}\right)t}}{b^2-1800\,k} + \frac{1}{20}\,\frac{\left(b^2-b\sqrt{b^2-1800\,k}-1800\,k\right)\mathrm{e}^{\left(-\frac{1}{900}b-\frac{1}{900}\sqrt{b^2-1800\,k}\right)t}}{b^2-1800\,k}$$

系统的实际响应是关于k, b, 和t的函数。

```
> Actual := unapply( rhs( sol ), k, b, t) :
```

k=10^4 N/m 和 b=10^3 Ns/m 条件下的目标响应和实际响应。

```
> plot( [ Target(t), Actual(10^4, 10^3, t) ], t = 0..1, thickness = 3, labels = ["seconds",
    "metres"],
  title = "Target Response and Actual Response for k = 10^4 and b = 10^3", legend
   = ["Target Response", "Actual Response"] );
```



显然 k 和 b 的值没有很好地匹配目标。提高匹配性的一个方法是代入 k 和 b 不同的组合，直到发现一个可接受的结果。下面的小节描述了一个更加系统的方法：使用优化。

测量目标和实际响应之间的误差

理想情况下，误差是目标和实际响应的平方差的积分。但是在这个应用中由于响应函数的复杂性，求积分值非常困难。作为近似，在一些关键的时间点上取函数样本，函数对这些时间点上的平方差相加。

在时间$t = \frac{1}{2}, 1, \frac{3}{2}, 2$**上的输出样本：**

```
> time_sample := 1/2, 1, 3/2, 2;
```

$$time\_sample := \frac{1}{2}, 1, \frac{3}{2}, 2 \qquad \textbf{(3.1.3.1)}$$

目标函数是实际和目标响应在这4个时间点上的平方差的和。尽管误差函数仅有两个变量，但是可以看出非常复杂。

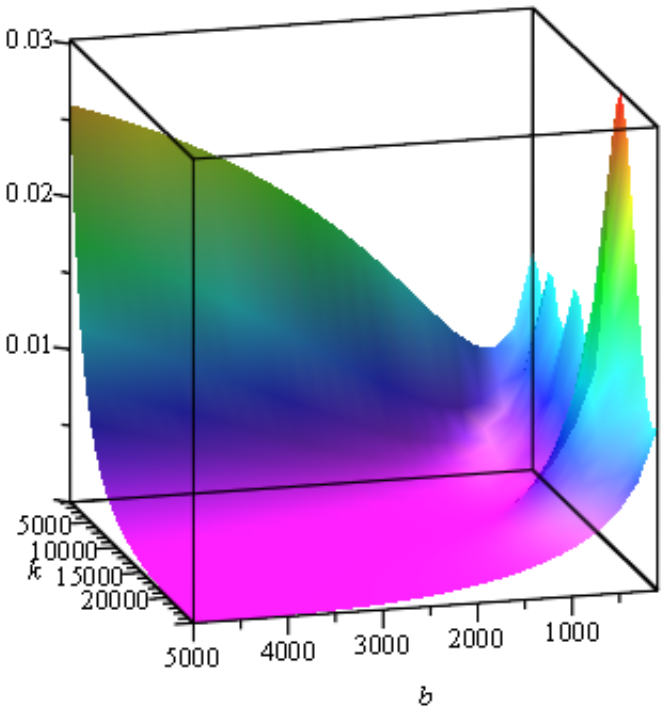
```
> Error := add( (Actual(k, b, time_sample[i]) - Target(time_sample[i]))^2, i = 1..4);
```

$$Error := \left(\frac{1}{20} \frac{(\%1)\, e^{-\frac{1}{1800} b + \frac{1}{1800}\sqrt{b^2 - 1800\,k}}}{b^2 - 1800\,k} + \frac{1}{20} \frac{(\%2)\, e^{-\frac{1}{1800} b - \frac{1}{1800}\sqrt{b^2 - 1800\,k}}}{b^2 - 1800\,k} + 0.001850800734 \right)^2 + \left($$

画出误差函数在 k-b 平面上的图形。函数不仅仅是非凸的，而且在可能含有全局最小值的区域比较平坦。在计算前还不知道下面显示的图形区域是否含有期望的答案。

（注意：根据微分方程理论，通常取消响应函数和误差函数的虚部项。但是，Maple的绘图函数在计算之前并不能检验。为了正确地绘图，对误差函数的虚部绘图，等同于函数绘图。）

```
> plot3d( Re(Error), b = 100 ..5000, k = 1000 ..25000,
      axes = boxed, shading = zhue, transparency = .35, style = patchnogrid, orientation
      = [75, 75] );
```



使用全局优化最小化误差

使用全局优化工具箱求误差函数的最小值。

```
> with(GlobalOptimization);
```

[GetLastSolution, GlobalSolve, Interactive] **(3.1.4.1)**

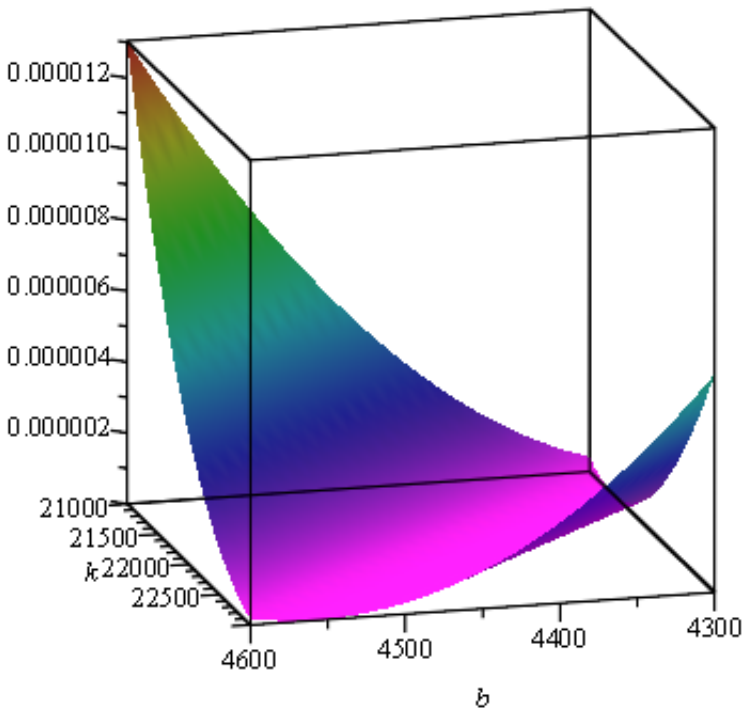
工具箱发现全局最小值。注意到误差值非常小，接近于0，正如期望的一样。

```
> sol := GlobalSolve( Re(Error), b = 100 ..10000, k = 100 ..40000 );
```

$sol := [3.96363826829140810\ 10^{-8}, [b = 4530.56866744741, k = 22691.7138449499]]$ **(3.1.4.2)**

画出全局最小值领域内的误差函数。即使在这个小区域内，图形也不能明显地显示全局最小值的位置。

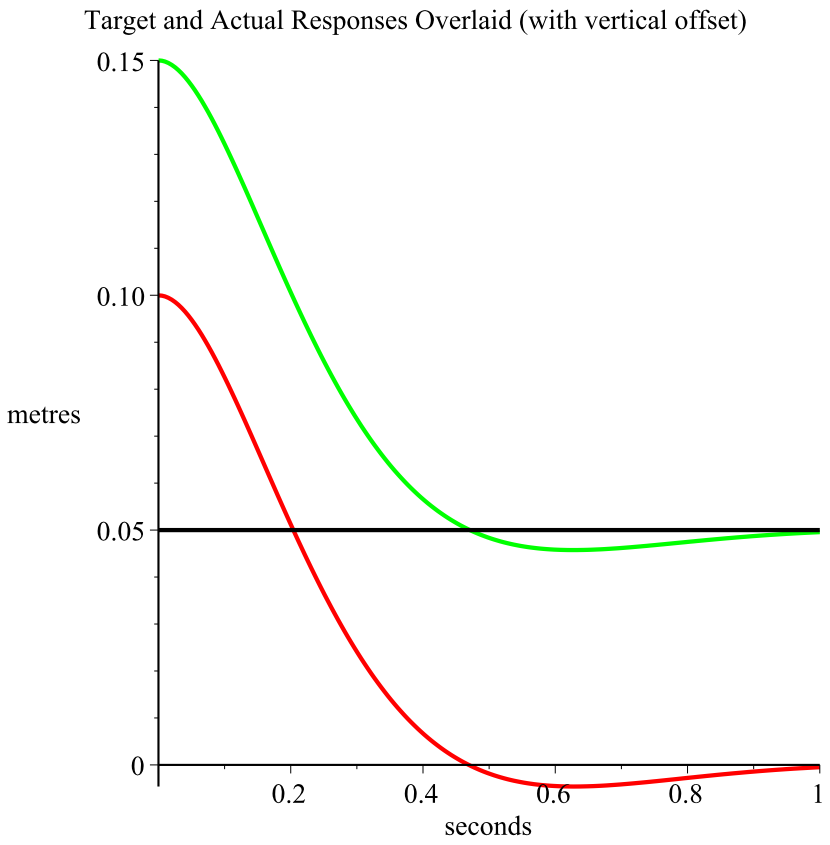
```
> plot3d( Re(Error), b = 4300 ..4600, k = 21000 ..23000,
      axes = boxed, shading = zhue, transparency = .35, style = patchnogrid, orientation
      = [75, 75] );
```



测试结果

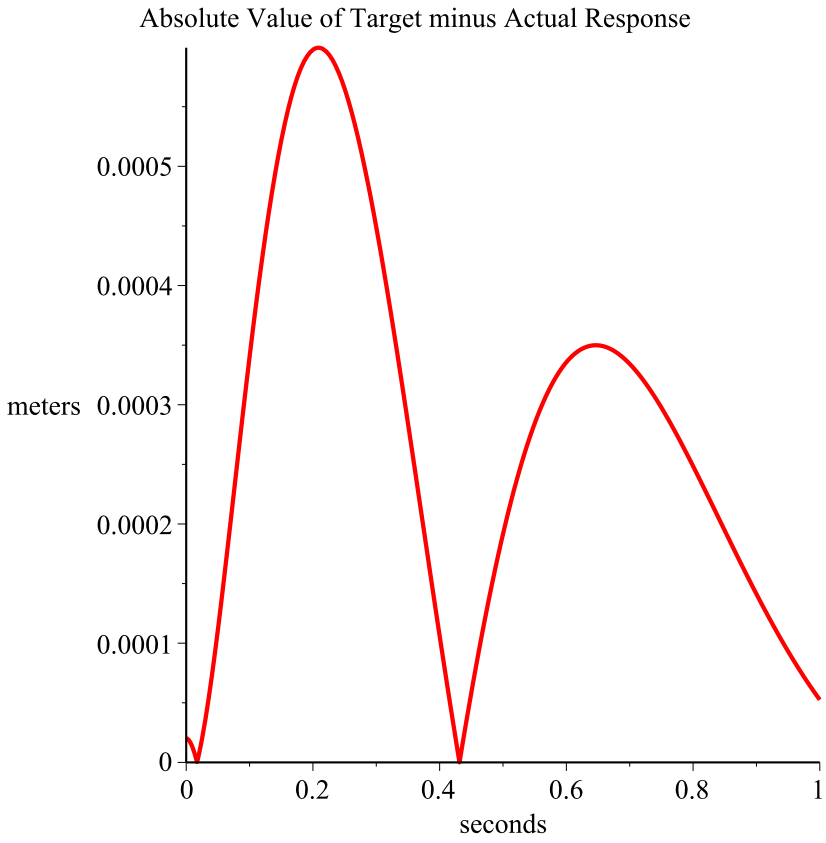
在同一图形上显示实际响应和目标响应，k 和 b 使用全局优化得到的值。它们可以很好地匹配。

```
> assign( sol[2] );
> plot( [Target(t), Actual(k, b, t) + .05, .05 ], t = 0 ..1,
      labels = ["seconds", "metres"], title
     = "Target and Actual Responses Overlaid (with vertical offset)", thickness = 2, color
     = [red, green, black]);
```



从目标响应和实际响应之间差的图形上可以发现，即使是最大差值也可以忽略。

```
> plot(abs(Target(t) - Actual(k, b, t)), t = 0 ..1,
      title = "Absolute Value of Target minus Actual Response", labels = ["seconds",
     "meters"], thickness = 2);
```



香水瓶子设计的优化

一家香水公司希望设计一个新的香水瓶子，要求容量固定的前提下降低运费。运费是与包装盒占有的体积成正比。因此设计任务是最小化占有空间，同时保持瓶子的容量和美观性。

瓶子的设计是椭圆体、平底。这里有4个问题变量：椭圆的3个离心率参数，椭圆体被切除形成平底的高度位置。

瓶子模型

```
> restart;
> interface(warnlevel = 0) : with(plots) : with(plottools) :
> with( GlobalOptimization );
                    [GetLastSolution, GlobalSolve, Interactive]
```

(3.2.1.1)

瓶子形状的模型是一个包含参数 a, b, c 的椭圆体，中心在原点。上面部分是椭圆体，但

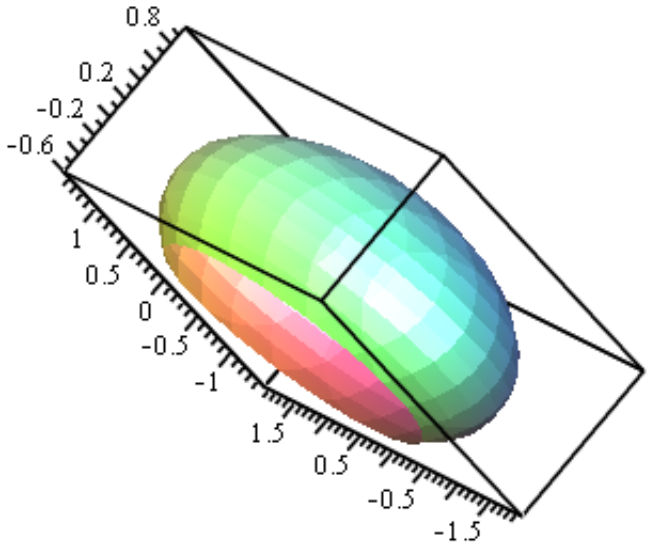
椭圆体地段是平面 $z = -h$，这里 h 是模型的第4个参数。长度单位是 cm，体积是 mL。

```
> bottleShape := x^2/a^2 + y^2/b^2 + z^2/c^2 = 1;
```

$$bottleShape := \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \qquad (3.2.1.2)$$

对应的图形是一个平底瓶。目的是使用优化方法改进这个形状。

```
> implicitplot3d( eval(bottleShape, {a = 2, b = 1.5, c = 1}), x = -2..2, y = -1.5..1.5, z = -6
  /10..1,
        scaling = constrained, orientation = [75, 80], style = patchnogrid, lightmodel
  = light1);
```



目标函数

目标函数：包装体积

包含瓶子的盒子是一个平行六面体，宽度是 2a，长度是 2b，高度是 $h + c + 1$，这里额外的长度是为喷嘴准备的空间。目标是最小化 $4\ a\ b\ (c + h + 1)$。

```
> packageVolume := 4*a*b*(c + h + 1);
```

$$packageVolume := 4\,a\,b\,(h + c + 1) \qquad (3.2.2.1)$$

约束条件

约束条件1：瓶子体积

瓶子必须能够至少装 40 mL 的香水。瓶子的体积是椭圆体的横截面，是关于 z 的函数，在区间 $z = -h..c$ 上的的积分。

```
> z_crossSection := Pi * a * b * (1 - z^2 / c^2);
```

$$z\_crossSection := \pi\,a\,b\left(1 - \frac{z^2}{c^2}\right) \qquad (3.2.3.1)$$

```
> perfumeVolume := Int( z_crossSection, z = -h..c );
  perfumeVolume := value( perfumeVolume );
```

$$perfumeVolume := \int_{-h}^{c} \pi\,a\,b\left(1 - \frac{z^2}{c^2}\right) dz$$

$$perfumeVolume := -\frac{1}{3}\,\frac{\pi\,a\,b\,(c^3 + h^3)}{c^2} + \pi\,a\,b\,(h + c) \qquad (3.2.3.2)$$

通过求它在 $h = c$ 上的值检查公式。变为一个求椭圆体的公式，符合期望。

```
> eval( perfumeVolume, { h = c } );
```

$$\frac{4}{3}\,\pi\,a\,b\,c \qquad (3.2.3.3)$$

```
> constraint1 := perfumeVolume ≥ 40;
```

$$constraint1 := 40 \le -\frac{1}{3}\,\frac{\pi\,a\,b\,(c^3 + h^3)}{c^2} + \pi\,a\,b\,(h + c) \qquad (3.2.3.4)$$

约束条件2：底的宽度

底必须至少有 2 cm 的直径，以保持瓶子的稳定性。作为 h 的函数，在平面 $z = -h$ 上的横截面最小直径是 $2\sqrt{b^2\left(1 - \frac{h^2}{c^2}\right)}$。约束 h 让直径至少是 2 cm。

```
> hSquareMax := solve( 4 * b^2 * (1 - h^2 / (c^2)) = 2^2, h^2 );
```

$$hSquareMax := \frac{(-1 + b^2)\,c^2}{b^2} \qquad (3.2.3.5)$$

得到下面的约束：

```
> constraint2 := h^2 ≤ hSquareMax;
```

$$(3.2.3.6)$$

$$constraint2 := h^2 \le \frac{\left(-1 + b^2\right) c^2}{b^2} \qquad (3.2.3.6)$$

约束条件3：美学比例

客户通常喜欢圆滑瓶。尽管是最经济的设计，但公司不会销售球形香水瓶。限制瓶子的比例满足 $2\ b \le a$。

```
> constraint3 := 2 * b <= a;
```

$$constraint3 := 2\ b \le a \qquad (3.2.3.7)$$

优化

```
> constraints := { constraint1, constraint2, constraint3 };
```

$$constraints := \left\{40 \le -\frac{1}{3}\ \frac{\pi\, a\, b\, \left(c^3 + h^3\right)}{c^2} + \pi\, a\, b\, (h + c), h^2 \le \frac{\left(-1 + b^2\right) c^2}{b^2}, 2\ b \le a\right\} \qquad (3.2.4.1)$$

使用全局优化工具箱发现最优解。得到包装体积的最小值 77.69 mL，显然大于瓶子的体积 40 mL，满足期望。

```
> sol := GlobalSolve( packageVolume, constraints, a = 1..10, b = 1..10, c = 1..10, h = 1
      ..10 );
```

$$sol := [77.6889319096612782, [a = 2.15296455896033, b = 1.07648227948016, c = 5.38623110089699, h = 1.99398838830011]] \qquad (3.2.4.2)$$

测试

检查得到的结果是否在可接受的精度内满足所有的约束条件。为了更精确，使用 GlobalSolve 调用格式中的 feasibilitytolerance 和 penaltymultiplier 参数项。

```
> subs( sol[2], constraints ) : evalf[6](%);
```

$$\{2.15296 \le 2.15296, 3.97599 \le 3.97599, 40. \le 40.0000\} \qquad (3.2.5.1)$$

优化后的椭圆体：

```
> bottleShape := eval( bottleShape, sol[2] );
```

$$bottleShape := 0.215737796445492\ x^2 + 0.862951185781968\ y^2 + 0.0344691071043511\ z^2 = 1 \qquad (3.2.5.2)$$

```
> A := eval(a, sol[2] ) : B := eval(b, sol[2] ) : C := eval(c, sol[2] ) : H := eval(h,
      sol[2]) :
```

优化后的香水瓶的轮廓图：

```
> glass := implicitplot3d( bottleShape, x = -A..A, y = -B..B, z = -H..C,
                grid = [8, 8, 8], style = patchnogrid, lightmodel = light4,
                transparency = .1, color = gold) :
```

```
liquid := implicitplot3d( lhs(bottleShape) = .85, x = -A..A, y = -B..B, z = -.9 * H..(.7)
    * C,  style = patchnogrid, lightmodel = light4, transparency = .8, color = black) :

cap := cylinder( [0, 0, .95 * C], A/6, color = grey) :

display( glass, liquid, cap, scaling = constrained, axes = box);
```