第 3 章 工程计算：最优与优雅

或

链 π 数

本章介绍了有负载和无负载的下垂链的解决方案。它展示了如何构建作用在链条上的力的图表。对不同形状的容器进行了优化。

3.1. 无负载的链条

想象一下，作为工程师和设计师(архитектору[1], художнику)，您需要设计和建造一个装饰围栏——在地面上安装铸铁柱子并在其上悬挂链条(图 3.1)。如何做到这一点，使围栏在工程计划中既漂亮又最佳。有这样一种隐含的联系：一座桥梁或其他工程结构如果漂亮，先验地被认为是坚固的！例如，看看舒霍夫塔或其他类似的结构，它们无可争议地受到了人们的钦佩！同时，我们也不能忘记经济——建筑的成本不应太贵。但是，如果一座建筑是工程和艺术表现的杰作，它的实际价格可能非常高。或者根本就没有。谁能确定舒霍夫塔的现代价值？它是无价的！

图 3.1. 悬空的链条形成了一条链线
（https://ru.wikipedia.org/wiki/Цепная_линия）

[1] 在许多工程结构上，你可以看到一个标明设计者和建筑师的牌匾。

第 3 章

让我们根据这样的技术，经济和美学位置来计算我们的链条围栏[1]。当然，您可以尽量节省链条——取极短的部分并几乎水平地将它们固定在柱子上，把柱子本身做得很低，放得很少。但从工程的角度来看，这样的围栏既丑陋又不切实际。需要很大的力（见图 3.1 中的力 F）才能将这样的链条（请原谅重述）连接到相邻的柱子上。在这种情况下，链条会像弦一样绷紧。如果有人无意中踩到这样的链条，甚至坐在上面，并且还试图骑自行车[2]碾过它，那么它可能会断裂，柱子就会倒下。如果你延长链条的长度，作用在杆子上的力首先会减少（绳子的张力会减少），然后会因为链条重量的增加而增加。当然，除非链子直接落在地上。这样一来，根据链条的长度，在你的脑海中立即产生了一个链条在杆子上连接点的拉力的平滑函数。这个函数必须有一个最小值。让我们计算一下——确定图 3.1 中力 F 最小的链长。

图 3.2 显示了在 SMath 中进行的这样一个简单计算。

在图中计算的第 1 点中。变量 L 和 g_c 被赋予了柱子之间的距离和链条的重量（每米的链条有多少牛顿的重量）的值。这些数字是任意的，对计算结果没有影响--最佳链条的长度与连接点之间的距离之比。我们将使用数值求解方法，而不必组成和求解方程（分析方法），因为它被证明是相当复杂的：它包含一个反双曲切线。这种不组成方程的技术将在第九章解决追击问题时重复使用。

$$L := 1\ m \qquad g_c := 1\ \frac{N}{m} \tag{1}$$

$$y(x, a) := a \cdot \mathrm{ch}\left(\frac{x - \frac{L}{2}}{a}\right) - a \qquad y(0\ m, 1\ m) = 0.1276\ m \tag{2}$$

$$S(a) := \int_{0\ m}^{L} \sqrt{1 + \left(\frac{\mathrm{d}}{\mathrm{d}\,x} y(x, a)\right)^2}\ \mathrm{d}\,x \qquad S(1\ m) = 1.0422\ m \tag{3}$$

$$S(a) := 2\sqrt{(y(0\ m, a) + a)^2 - a^2} \qquad S(1\ m) = 1.0422\ m \tag{4}$$

$$a(S) := \mathrm{solve}(S(a) = S, a, 0.1\ m, 2\ m) \qquad a(1.5\ m) = 0.3082\ \frac{N}{N \cdot m^{-1}} \tag{5}$$

$$F_y(S) := \frac{g_c \cdot S}{2} \qquad F_y(1.5\ m) = 0.75\ N \tag{6}$$

$$F_x(S) := a(S) \cdot g_c \qquad F_x(1.5\ m) = 0.3082\ N \tag{7}$$

$$F(S) := \sqrt{F_x(S)^2 + F_y(S)^2} \qquad F(1.5\ m) = 0.8109\ N \tag{8}$$

[2] 数学意义上的顶，而不是曲线的外观（见图 3.1 和 3.8）。如果我们把参数设为负数，就会得到一个 ″链式 ″拱门，顶点在它应该在的地方--最大点，而不是最小点。美国的圣路易斯市建造了一个非常有趣的工程结构，一个高 192 米的倒链形状的拱门(https://ru.wikipedia.org/wiki/Ворота_Запада).

第 3 章

图 3.2. 寻找理想链的尺寸：形成目标函数

在计算第 2 项，介绍了一个链函数，取自维基百科（见图 3.1 下的标题）。经典的链线方程是 $y = a\ ch(x/a)$。我们对其进行了一些修改，使链线的顶端不在通常（经典）的点 x=0, y=a, 处，而是在点 x=L/2, y=0 处——见图 3.1。

注意。一旦所有的自定义计算函数都写好了，就可以调用它们进行测试：用数值代替参数，并检查函数的返回值。这些测试运算符可以从完成的计算中删除。

像其他任何形式的 $y = f(x)$ 的曲线一样，链长可以用定积分（图 3.2 中的第 3 项）来计算，其中包含了勾股定理（斜边等于边平方和的平方根），也可以用专门为链线得出的更简单的公式（第 4 项）。这个公式很容易在上面的链线信息网站上找到，然后根据我们的计算对其进行修改，同时考虑到我们不是在处理经典函数，而是在处理一个稍加修改的函数。

第 5 点是计算中的关键点。它根据 SMath 内置的用于数值解方程的 solve 函数指定了用户函数 $a(S)$。在我们的例子中，这个方程是一个形式为 $S(a) = S$ 的方程。它在变量 a 上通过在 10 厘米到 2 米的段上进行半除法的方法进行数值求解。solve 函数不会返回数字答案（方程的根），而是返回一个名为 a 和参数 S 的新函数。变量 a 从一个参数变成一个函数名，变量 S 从一个函数名变成一个参数。这就是程序员的蜕变!

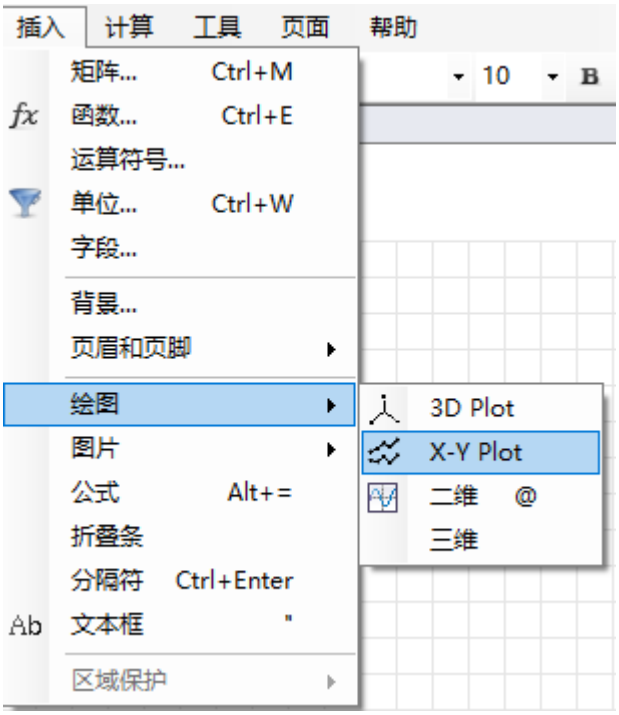
链向下和侧拉柱顶端的力 F 被分解为垂直力投影 F_y（向下拉）和水平力投影 F_x（侧拉）。在图 3.1 中，这个力指向相反的方向。力的垂直投影线性地取决于链的长度——参见图 3.2 中的第 6 项：当两根柱子的高度相同时，链的重量（g_c S）均匀地（除以 2）分布在两个锚固点上。如果考虑链状公式中参数 a 的物理意义，力的水平投影也不难确定（见图 3.2 中的第 2 项）。这种物理意义是通过求解作用在基本链段上的力的微分平衡方程来推导链线公式的关键[3]。但几乎所有数学和物理参考书——纸质和电子参考书（例如在同一个维基百科中）都对这个重要点保持沉默。让我们纠正这一点并注意链状公式中的参数 a 是拉伸链的力 F_x, 的水平投影之比（它在链的长度上是常数）与链的比重，其值我们在图 3.2 第 1 项中给出了第二个算子。因此, 在第 5 项中测试调用 $a(S)$ 时, 我们不仅记录了米, 而且还记录了牛顿除以牛顿乘以米的第一级减去。当然，所有这些都可以简化为米，但最好不要这样做，以免失去问题的“物理性”。根据比值 $a = F_x / g_c$ 确定力的水平投影——见图 3.2 第 7 项。力的垂直投影 F_y 是一个变量。它在链的顶点（在链的最大下垂点）为零，并在链与柱的连接点取最大值。

如果不了解上述参数 a 的物理意义，则可以通过其垂直投影和已经存在双曲正弦 sh 的链状线函数的导数来确定力的水平投影，而不是双曲余弦 ch。

图 3.2 中可以看到的最后一点是创建所需的（目标）函数 $F(S)$（第 8 项）。

[3] 在第 5.3 节中，我们将真正的、基于物理学的粘度单位返回到计算中--见图 5.13、5.13.

图 3.3 显示了这三个函数：力 F_x的水平分量随着链条长度 x 的增加而减少，垂直分量 F_y线性增加，而力 F本身先减少后增加--它有一个最小点。让我们来计算一下！我们怎样才能做到这一点呢？



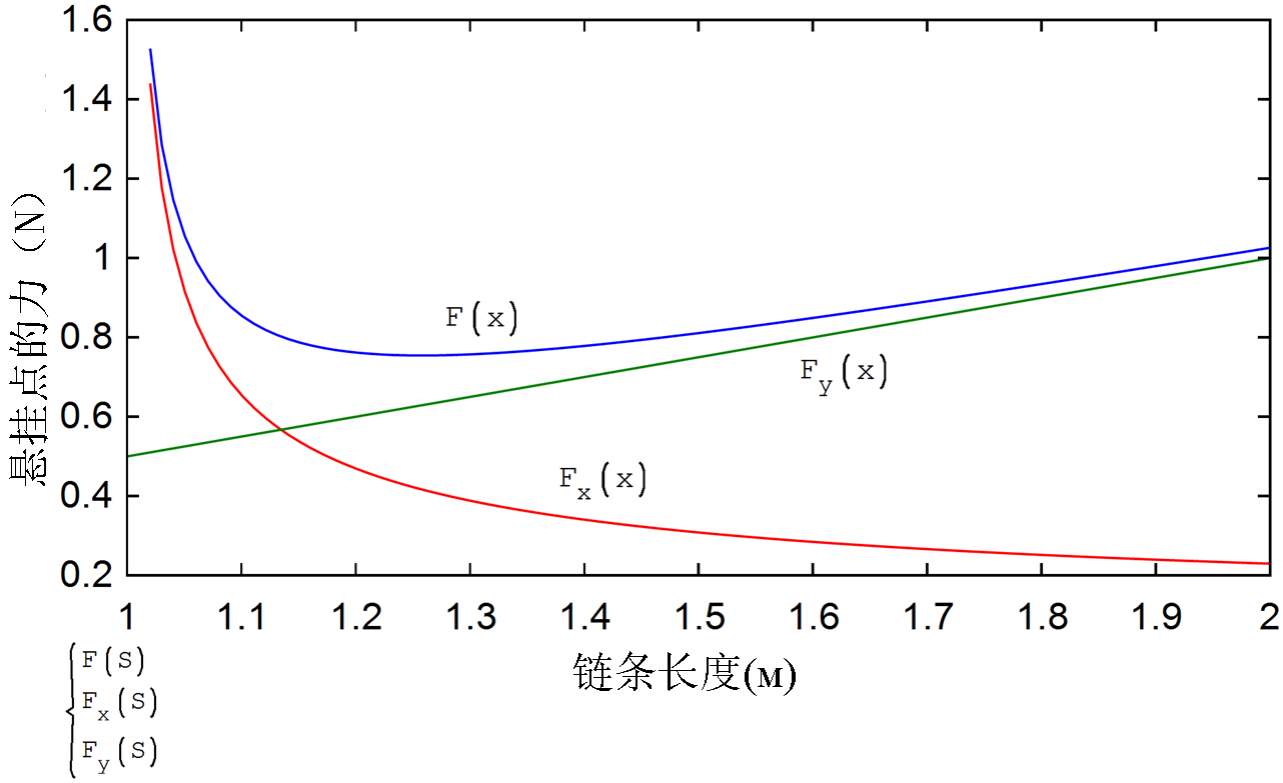


图 3.3. 链条对柱子的变化力及其水平和垂直投影的图示

曾经，第一作者要求他的学生在电脑上打开图形编辑器 ″画图″，然后拖动鼠标，将绘图区域设置为学生认为最漂亮、最相称的高和宽的比例的矩形[4]。这就是我们把学生的矩形集合在同一个图形上时得到的结果（见图 3.4）。矩阵 *Data* 以像素为单位存储学生矩形的尺寸。这些数字然后被转移到 *X* 和 *Y* 向量上。

$$Data := \begin{bmatrix} 1430 & 868 & 955 & 799 & 950 & 878 & 928 & 1056 & 1053 & 1017 & 1080 & 950 & 1120 & 615 & 722 \\ 851 & 388 & 604 & 471 & 505 & 626 & 619 & 632 & 501 & 675 & 716 & 479 & 682 & 412 & 582 \end{bmatrix}$$

$$n := \mathrm{cols}(Data) = 14 \qquad X := \mathrm{row}(Data, 1)^{T} \qquad Y := \mathrm{row}(Data, 2)^{T}$$

$$k := \frac{\sum_{i=1}^{n} \frac{Y_i}{X_i}}{n} = 0.611 \qquad i := [1..n] \qquad Pr_i := \begin{bmatrix} 0 & Y_i \\ X_i & Y_i \\ X_i & 0 \end{bmatrix} \qquad C := \begin{cases} Pr \\ k \cdot x \\ 0.618 \cdot x \end{cases}$$

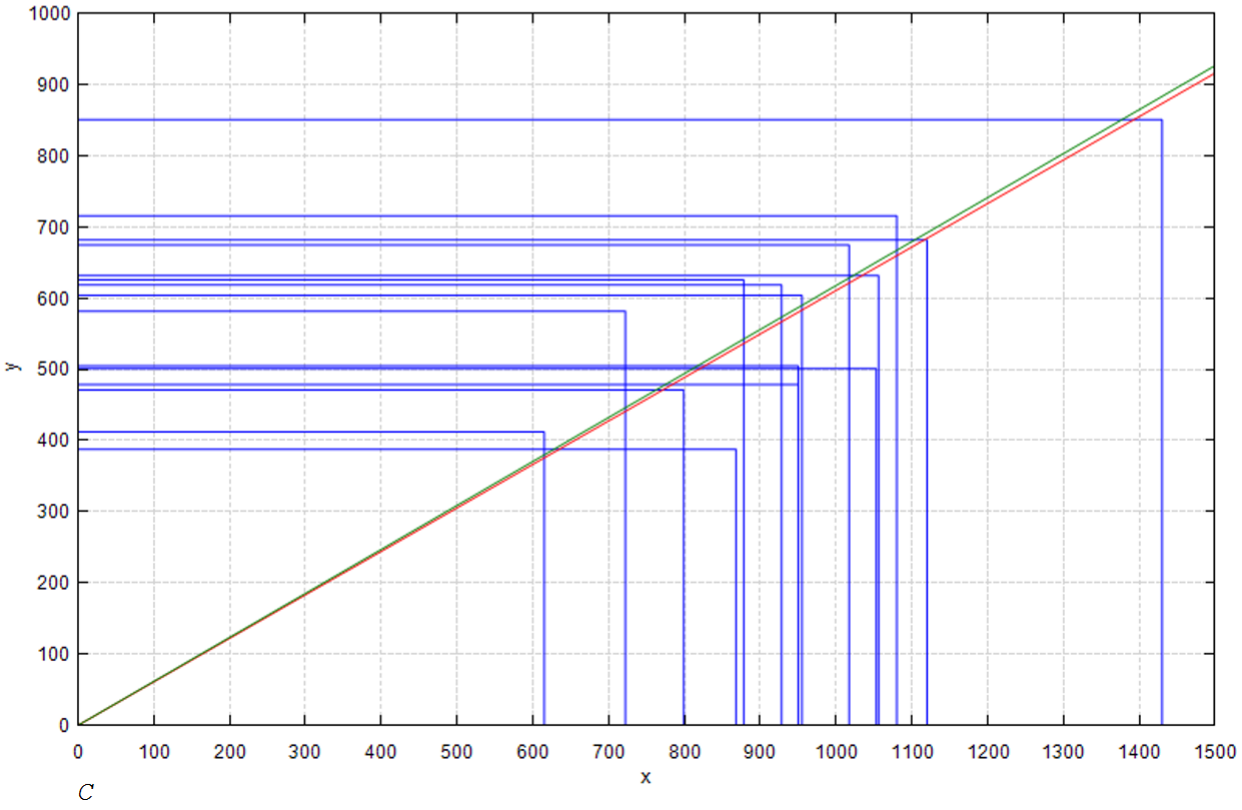


图 3.4. 学生的 ″黄金 ″矩形

经过简单的统计处理——确定矩形高度与其宽度的平均算术比值后，结果证明该值等于 0.611。在图 3.4 中，画出一条带有相应倾斜角正切的绿色直线。这只不过是一个大致的*黄金比例，黄金分割率*，当一个矩形的宽度与它的高度相关时，就像宽度和高度的总和与宽度（红色直线）相关。这就是 М Э И 学生所拥有的完美艺术品味！他们画了不同的长方形，

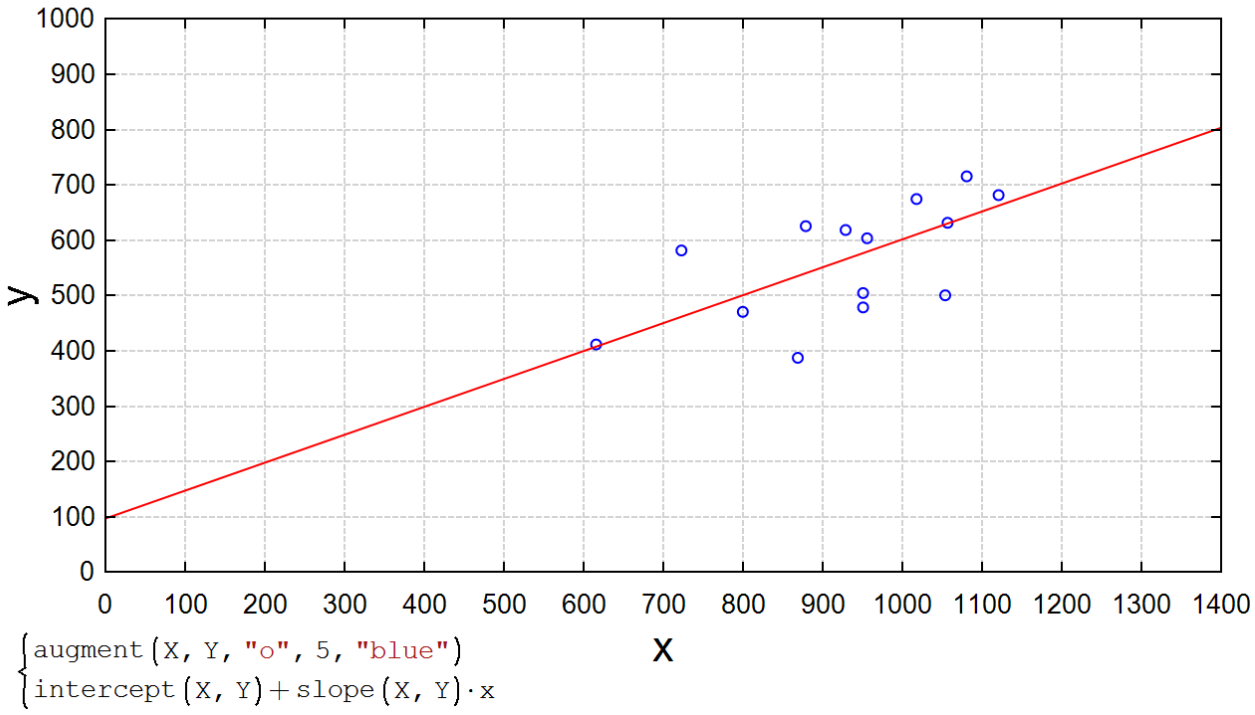
[4] 然后在这个矩形中绘制椭圆——见第 10 章中的图 10.9。

但他们中间的长方形竟然是金色的！在黄金比例中隐藏着一个数字，其四舍五入值为 0.618。

经典的更复杂的统计处理（回归分析）是最小二乘法。图 3.5 显示了线性回归如何应用于学生矩形-向量 x 和 y（见图 3.4）。在 Excel 表格处理器、数学程序 Mathcad 和许多其他程序的环境中，这项工作由内置的 intercept（交叉）和 slope（倾斜）函数完成。标准 SMath 配置中没有这些功能，但您自己创建它们并不困难——参见图 3.5，依赖于在 Internet 上获得的信息。

$$intercept(X, Y) := \left| \begin{array}{l} n := \mathrm{rows}(X) \\ \dfrac{\sum_{i=1}^{n} X_i^{\,2} \cdot \sum_{i=1}^{n} Y_i - \sum_{i=1}^{n} X_i \cdot \sum_{i=1}^{n} X_i \cdot Y_i}{n \cdot \sum_{i=1}^{n} X_i^{\,2} - \left(\sum_{i=1}^{n} X_i\right)^2} = 97.704 \end{array} \right.$$

$$slope(X, Y) := \left| \begin{array}{l} n := \mathrm{rows}(X) \\ \dfrac{n \cdot \sum_{i=1}^{n} X_i \cdot Y_i - \sum_{i=1}^{n} X_i \cdot \sum_{i=1}^{n} Y_i}{n \cdot \sum_{i=1}^{n} X_i^{\,2} - \left(\sum_{i=1}^{n} X_i\right)^2} = 0.5046 \end{array} \right.$$



第 3 章
图 3.5. 最小二乘法

对学生艺术品味的统计评价结果是完全可以预见的。他们中的许多人在博物馆中看到过边框中的画作，边框的比例接近黄金比例（见图 3.7）。黄金比例无处不在地围绕着一个人——例如，在建筑中。许多人在音乐中找到了它。但有什么可走的——普通电脑屏幕的比例接近黄金。书写这段文字的笔记本电脑屏幕尺寸为 207 x 332 毫米。这些数字的比率 0.6235 接近黄金（0.618）。这些都不由得影响了学生们画长方形的过程。对从未见过任何风景画的人进行所描述的矩形实验会很有趣——例如，亚马逊地区的野蛮人，如果还有其他人的话。在某些方面，这种体验将类似于另一种类似的“语言学”体验——孩子从一出生就不被允许听人类语言，等待他最终会说什么语言。当然，这种经历是幼稚和不人道的，但它确实发生了。所以他们试图找出哪种语言是人类交流的主要基本语言。

第 10 章介绍了通过最小化目标函数来实现最小二乘法的一般情况（图 10.10 和 10.11）。

黄金比例不仅广泛用于艺术，而且还用于应用数学。有一些数值方法用于解决黄金比例也存在的问题。其中一个数值化寻找一个参数的函数的最小值的方法是基于黄金比例的。而这正是我们需要的!

图 3.6 显示了用黄金分割法搜索这种极小值的示意图。

最小值所在的区间（从一到二）按黄金比例从两侧划分。这项工作由一个名为 *GR* (Golden Ratio)的辅助用户定义函数执行，该函数具有两个参数。然后检查在哪个黄金分割点（我们在左边是 1.382，在右边是 1.618），分析的函数的值较小。如果在左截面（1.382）时，则最小值搜索间隔随左偏移而减小，如果在右截面（1.618）时，则随右偏移而减小。在 1 到 1.618 的新减少区间中，在左侧计算了一个新的黄金分割率（1.2361），非常重要的是，使用了第二个已经计算过的旧分割率（1.382）。重复这些操作，直到间隔末端的值变得大致相等。黄金比例允许循环只计算被分析的函数值一次而不是两次。这就减少了计算时间，使程序变得 ″漂亮″。而一个好的程序，就像一个好的工程建设一样，必须是美丽的!

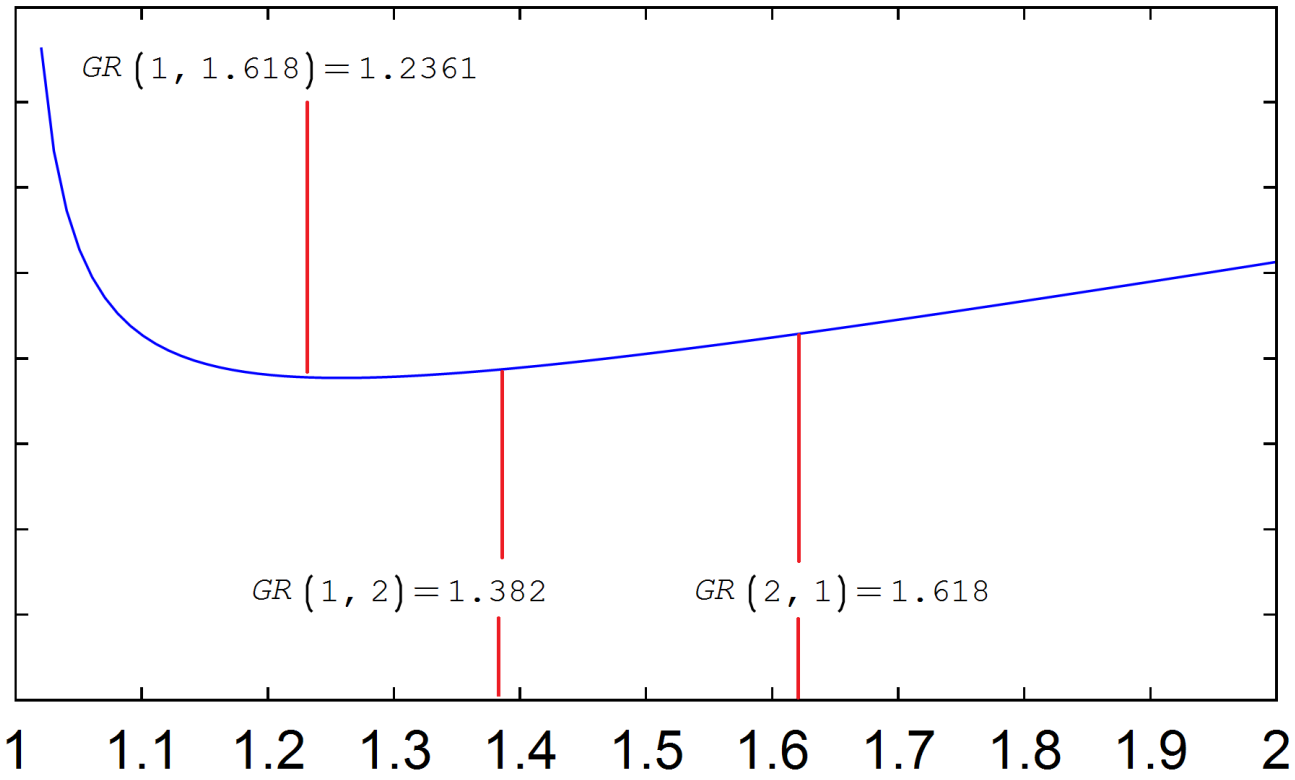


图 3.6. 使用黄金比例法进行最小搜索的示意图

图 3.7 显示了用嵌入在 SMath 包中的语言编写的程序，实现了所描述的黄金分割法。这个程序值得放在一个郁郁葱葱的金色法棍框架中，框架的两边大致是黄金比例。在同一个地方-金色框架显示了对 *MinGR* 函数的调用，以解决我们寻找函数 $F(S)$ 最小值的问题。为了让链条“温柔而美丽”地落在支撑它的柱子上，您需要使其长度超过柱子之间距离的四分之一*左右*。在这种情况下，链条在其固定点的倾斜角*约*为一个弧度。粗略的，而不是绝对精确的：美不喜欢绝对精确。

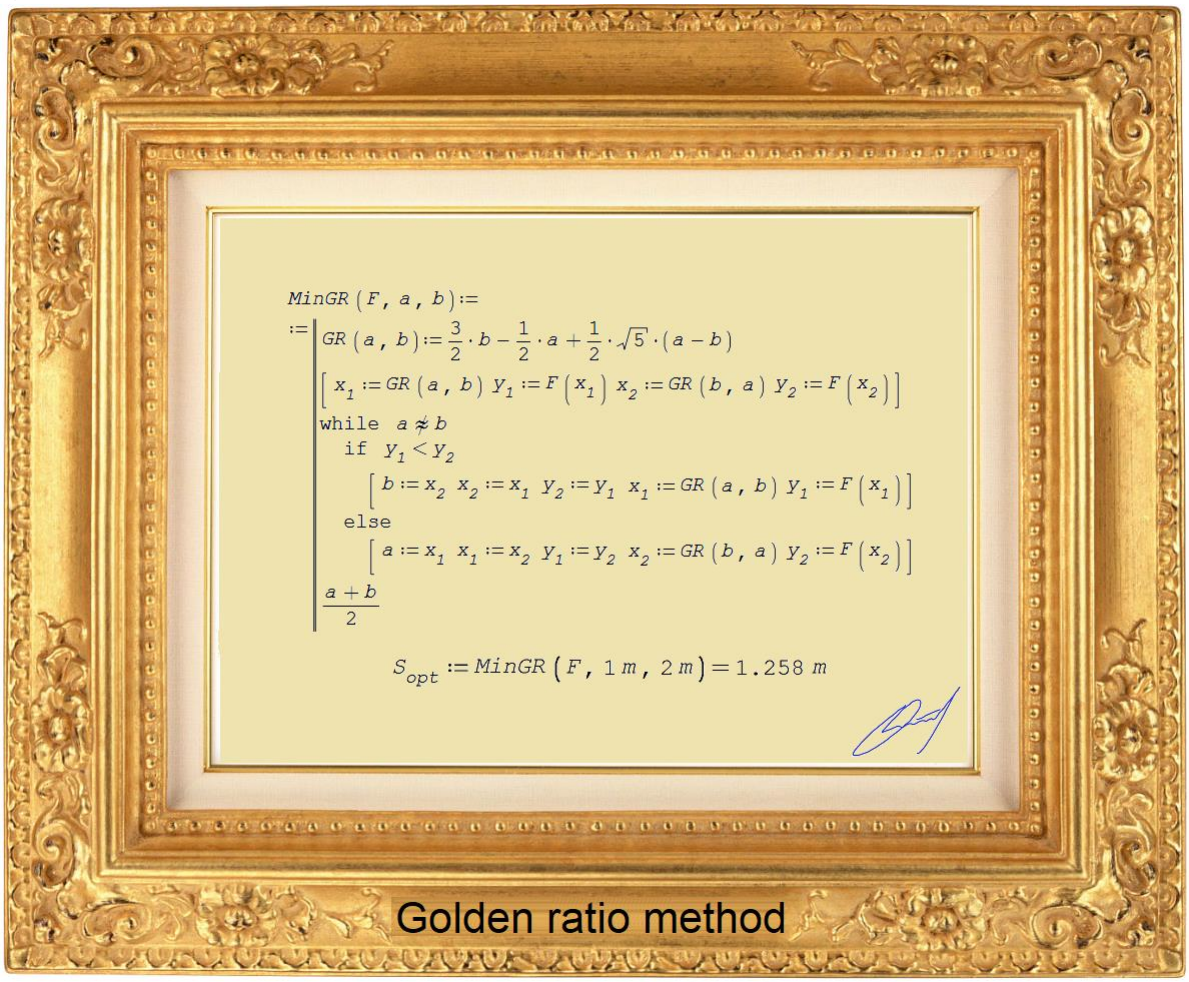


图 3.7. 绘制 ″黄金分割法的最小查找程序″

图 3.7.中关于方案的三个备注

1. 为了使程序更加紧凑，完全适合于显示屏幕（正如我们已经指出的，大致相当于黄金比例），几个运算符被写在一行上，这就是一个有一行和几列的矩阵。图 3.7 中这样的矩阵（水平向量）在其两端用方括号标记。
2. 该程序使用了 ″不约等于 ″运算符。这个布尔运算符和另一个布尔运算符 ″近似相等 ″在 SMath 的库中是可用的。这项工作是本书的第四位作者应第一位作者的要求完成的。
3. 为了程序的紧凑性，重复了第一个赋值语句“:=”。这是通过按下“Ctrl+Enter”键来完成的。

寻找一个以上参数的函数的最小值的程序如第 10 章中的图 10.11 所示。

图 3.8 显示了我们的计算链，悬挂在两个和四个柱上，其高度(*H*)与柱之间的距离(*L*)按银色而不是黄金比例对应。人们对银色比例的了解比对金色比例的了解要少。但对于我们的案例，在作者看来，它更适合。银截面公式如图 3.8 所示。如果你从这个公式中去掉 2，你就会得到黄金分割公式。许多“美学家”有充分的理由认为，银色的比例更优雅，如果我可以

这么说，它比金色更数学，因为它隐藏了 2 的平方根，而不是 5 的平方根，就像在黄金分割中一样（见图 3.7 中的 *GR* 函数）。两的平方根以其数学优雅让古希腊人感到惊讶，他们证明了这个数字是无理数。而美总是非理性的。智能手机屏幕和大幅面显示屏的比例接近银色比例。在第 10 章中，我们还将提到超黄金比例。

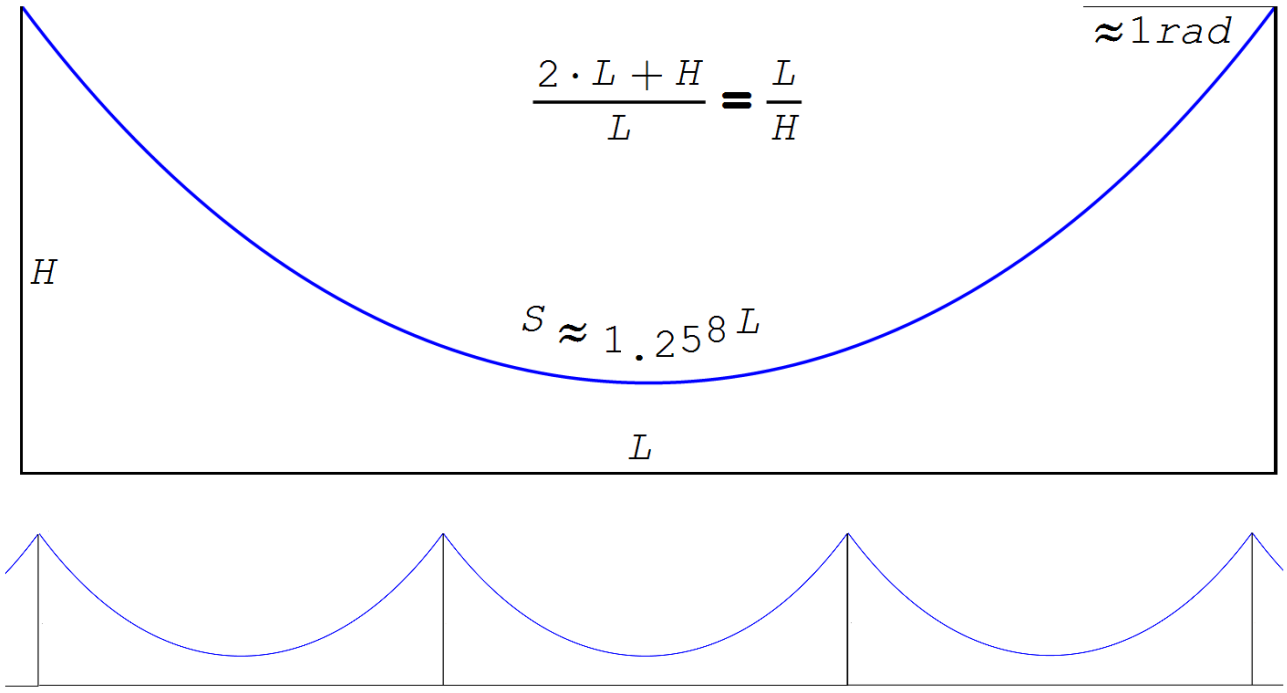


图 3.8. 理想的 "银色 "链条在两个和四个支柱上

你可以让学生从学生对这种简单的工程结构之美的理解的角度来画一个挂在柱子上的链条，你可以处理这些图纸，并得出它们与图 3.8 所示的形状有多接近的结论。

近似值为 1.258（最佳链的长度与其在同一水平上的固定点之间的距离之比）的数字可以被视为某种新的数学常数，并将其命名为 π꜀（链- catenary，chain - π 数）。您可以开始计算竞赛——在这个常数中找到越来越多的符号，就像对待通常的“循环”数 π （周长与其直径的比值）一样，其近似值为 3.14 几乎每个人都知道。

附注

该教程的第一作者在 2015 年准备出版一本书[4]时计算了链数 π꜀（约 1.258），其中有一章是关于链的。人们对发现一个新的数学常数产生了希望。在互联网上搜索了“1.258 catenary”，将这本教程的第一作者带到了同一 2015 年出版的[5]，因此该书的第一作者在这个常数上的优先级有点晚了。另一方面，认为美籍华人王和……这本教程的第一作者几乎同时独立地发现了这个常数，也是完全可以接受的。

顺便说一句,链线公式本身也是由三位数学家——荷兰的惠更斯、瑞士的约翰·伯努利[5]和德国的莱布尼茨——同时独立地发现的,不是两位,而是三位。在他们之前，人们认为链条沿着抛物线下垂。即使是伟大的伽利略也这么认为。诚然，在他生命的尽头，他承认自己错了，但没有时间推导出这条美妙曲线的公式。

在任务的第 2 段中，读者被要求对锚定在不同高度的链条进行优化计算。寻找最佳的链条长度将在这项任务之前进行。

任务。一条链条挂在高度为 h_1和 h_2的两根柱子上。两根柱子之间的距离也已给出并存储在变量 L 中。在已知链条长度 S 的情况下，求链条的几何参数（特别是链条与地面的间隙）和受力参数（连接点的链条拉力）。

图 3.9 显示了 SMath 环境中此问题的可能解决方案之一。

输入原始数据——变量 h_1、h_2和 L 的数值，先输入长度为 m := 1 的伪单位。由于 SMath 包的一些工具无法处理维度量，因此必须这样做。然后，在计算的第二行，输入一个非经典的链线公式，附加参数 x_0和 h --与最小点的坐标（链线的顶点--见上文）。

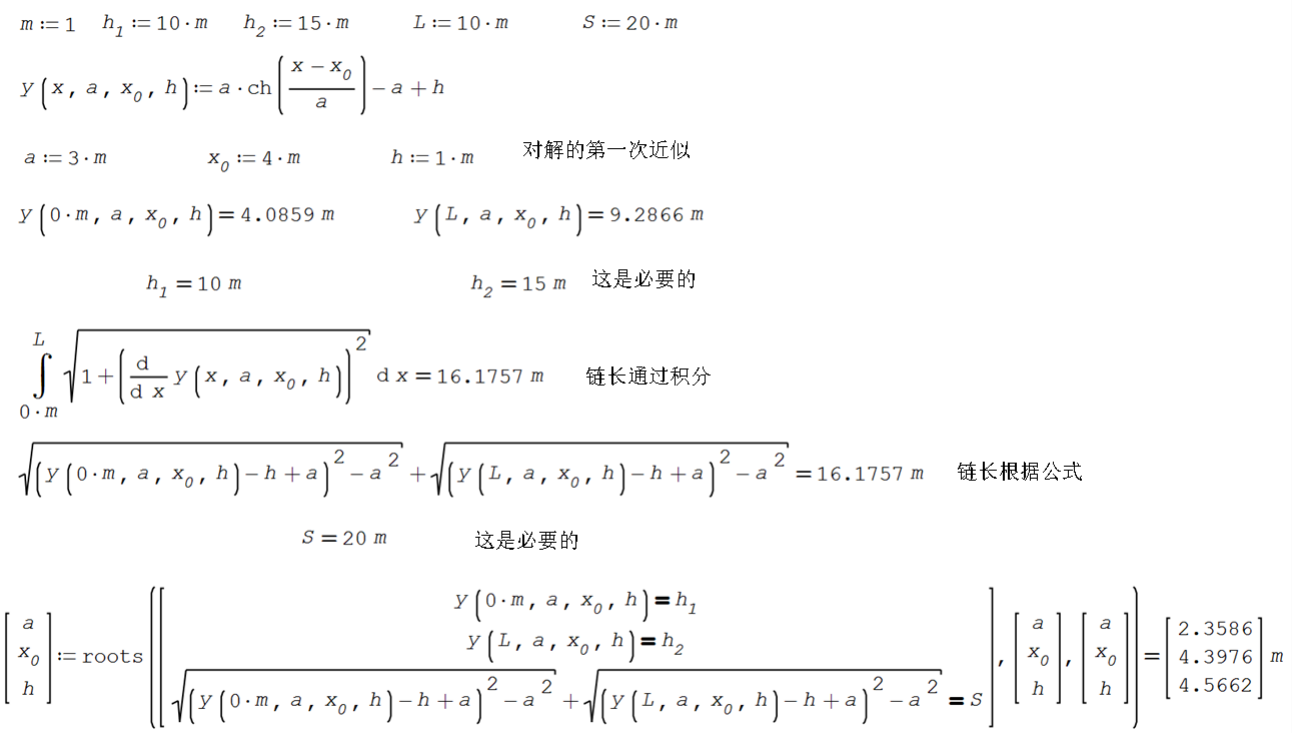


图 3.9. 在不同水平的两端悬挂的链条的计算

问题的解归结为求具有三个未知数 a, x_0和 h 的三个非线性代数方程组的根。这是 SMath 中内置的具有两个或三个参数的 roots c 函数的目的。第一个参数是带有待解方程的向量，第二个参数是指示系统未知的向量，第三个可选参数存储解的初始近似向量。当 roots 函数没

[5] 这位瑞士科学家的姓氏加上了一个名字，因为伯努利王朝有九位伟大的数学家和物理学家（其中三位是伟大的）。我们必须澄清哪一个伯努利发现了链状线公式。

有给出解时，或者当有多个解并且需要得到其中一个解时，需要初始近似。输入第一次近似后，检查方程的左侧给出了什么值。它们应尽可能接近右侧部分的值。

调用 roots 函数并获得解决方案后，对其进行检查-见图 3.10。为此，从图 3.9 中的解决方案中复制四个语句并将它们粘贴在具有 roots 函数的语句之后就足够了。

检查解决方案

$$y\left(0\cdot m, a, x_0, h\right)=10\ m \qquad y\left(L, a, x_0, h\right)=15\ m$$

$$\int_{0\cdot m}^{L}\sqrt{1+\left(\frac{\mathrm{d}}{\mathrm{d}\,x}y\left(x, a, x_0, h\right)\right)^2}\,\mathrm{d}\,x=20\ m$$

$$\sqrt{\left(y\left(0\cdot m, a, x_0, h\right)-h+a\right)^2-a^2}+\sqrt{\left(y\left(L, a, x_0, h\right)-h+a\right)^2-a^2}=20\ m$$

图 3.10：检查有三个未知数的三个方程是否得到正确解决

图 3.11 以图形方式显示了松弛链问题的解决方案。所用的图是 SMath 包内核中的图。本章上面的图显示了 SMath 扩展中包含的图形。

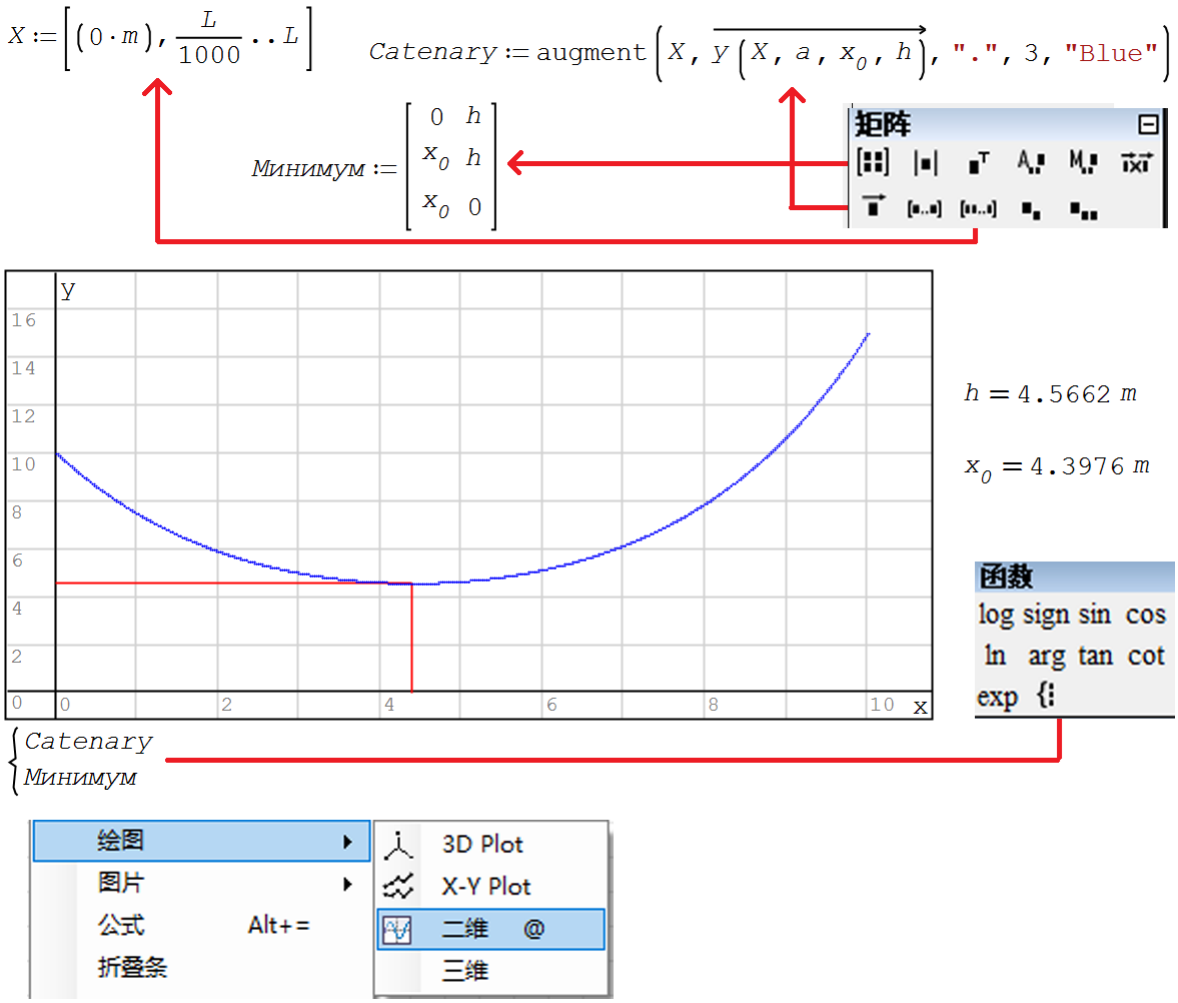


图 3.11. 松弛链问题的解决方案的图形表示

3.2. 带负载的链条

最近，莫斯科出现了一种新的交通方式——缆车。第一个连接 Лужники 和 Воробьевы Горы（图 3.12），第二个即将出现在 ВДНХ.....

目前，它更像是对 "莫斯科人和首都游客 "的一种吸引力，而不是一种成熟的交通工具。但随着时间的推移，在这种新的高架线路建成后，所有这些都将成为真正的公共交通，抛开水障碍和主要道路--到难以铺设传统地面和地下交通--公共汽车、有轨电车或地铁的地方。本手册第一章提到了另一种奇特的公共交通手段--直升机。

第 3 章

图 3.12. 卢日尼基-麻雀山架空索道

任何工程结构的建造之前都有一个设计和计算阶段。让我们从两个支撑物开始，在它们上面挂上一根绳子（一种灵活的、同质的、不可拉伸的、沉重的线），在绳子上附加一个负载（一个缆车舱--见图 3.12），并计算它将如何下垂，以及沿着绳子的长度将有哪些力作用在绳子上。这项任务将像道路本身一样有趣和美丽，因为道路的绳索是由各个电线编织而成的。我们的任务将由物理学、理论力学，当然还有数学 "编织 "而成，没有数学，其他任何精确的科学都是不可想象的。当然，也离不开现代信息技术，它现在不仅渗透到学习过程中，而且渗透到我们所有的生活中。这种现代学习技术被称为 STEM: Science, Technology, Engineering, Mathematics…

图 3.13 显示了原始数据和两个用户功能的计算输入。原始数据为：

- 固定点之间下垂的绳索长度 S
- 链固定点坐标：左（x_L - y_L）右（x_R - y_R）
- 载重（缆车车厢）M
- 钢丝绳比(线)重 m_c
- 左支座至吊重点水平距离 x_M

最后一个量可以从 x_L 值更改为 x_R 值，从而产生缆车车厢从左支架到右支架运动的动画帧。

$$S := 12 \text{ м}$$

$$x_L := 0 \text{ м} \qquad y_L := 2 \text{ м}$$

$$x_R := 10 \text{ м} \qquad y_R := 5 \text{ м}$$

$$M := 2 \text{ кг} \qquad m_c := 1 \frac{\text{кг}}{\text{м}} \qquad x_M := 7 \text{ м}$$

$$y\left(x ; a ; x_0 ; h\right) := a \cdot ch\left(\frac{x - x_0}{a}\right) - a + h$$

$$y'\left(x ; a ; x_0\right) := \text{sh}\left(\frac{x - x_0}{a}\right)$$

图 3.13. 输入输入数据和用户功能

再次给出了图 3.13 中链[3]的表达式（见图 3.2）不是以通常的简洁规范形式 $y = a \text{ ch}(x / a)$，顶点（我们有最小值）在点 $x = 0$ $y = a$，而是以扩展（非规范）形式，顶点在 $x = x_0$，$y = h$。通过给定或计算参数 a，x_0 和 h 的值，不仅可以画出参数 a 给出的任意曲率的链状线，还可以画出笛卡尔图中任意位置的链状线。这就是我们要做的。如果将负载悬挂到链条上——柔性均匀不可拉伸的重丝（接下来我们将简称为链条），那么链条将分解为两个独立的部分，在公式中参数 a 的值相同，而参数 x_0 和 h 的值将不同。这可以通过记住参数 a 的物理意义来证明，它与通过链的元素部分（环节）末端力平衡微分方程的解导出链线公式有关。参数 a 是，我们重复一遍，在链的任何位置拉伸链的力的水平投影与链的比重的比值。参数 a 的单位是牛顿（力）除以牛顿，牛顿又除以米（链的比重）。简化后，得到的只是一个仪表，它将出现在我们的计算中。如果一百分之一的人知道链条不是在抛物线上下垂，而是在链条上下垂(见上文)。同样，在这 100 人中，有一人知道链的物理意义，并将理解为什么两个悬链段的 x_0 和 h 的值不同，但 a 的值相同。拉伸链的力的水平投影在最小点（那里这个力的垂直投影为零）和链的悬挂点（那里这个力的垂直投影取最大值）是相同的。

图 3.13 中最后一个项用双曲正弦引入了链线的导数。顺便说一下，有一个简单的分析表达式，没有双曲函数的链线长度，但只有平方根，但我们将在下面使用积分，其数字 ″取 ″是相当合法的。此外，积分本身是非常美丽和神秘的。以下是经典著作对它的描述：

“自爱”莱文说，被他哥哥的话刺痛了，“我不明白。 如果他们在大学里告诉我，其他人了解积分计算，但我没有，那么自尊”（列夫 • 托尔斯泰，小说《安娜 • 卡列尼娜》）。

“120 天后，积分的建设结束。一个伟大的历史性时刻即将来临，届时第一个积分将上升为世界空间。一千年前，你们英勇的祖先征服了一个国家的统治。你将有一个更光荣的壮举：用玻璃、电、喷火积分整合宇宙的无限方程……”（米哈伊尔 • 扎米亚京，反乌托邦小说《我们》）

“大家去，去乌拉尔！/我们正在清理战斗的地方/钢铁机器，在那里积分呼吸，/与蒙古野生鄂尔多斯！”（亚历山大 • 布洛克，诗“斯基泰人”）

在前计算机时代，即使是取最简单的积分也是一个很大的计算问题，需要时间、智慧和人的才能。这在上面的引文中有所反映。现在在计算机上，它可以相当容易和简单地完成。这就是为什么在图 3.13 中给出了电路长度公式的原始纯度--用关键词 ″曲线长度 ″在互联网上找到某些积分。

我们的负载下垂问题的解决方案是解决一个有五个未知数的五个方程组--见图 3.14。如果没有计算机，这又将是一个很大的计算问题，就像积分的情况一样。现在只需调用数学软件包中的一些函数来返回未知数的值。现在有一个从进口程序到国内程序的逐步过渡。所以我们将展示我们的问题是如何在 Mathcad（Mathvad.com）和 SMath（smath.com）这两个软件包中解决的。

在 Mathcad（进口软件）中，方程是在 Р е ш и т ь 块中解决的--见图 3.14a，以及在 SMath（国产程序）中使用 roots 函数（图 3.14b）。该函数的第一个参数向量存储方程组本身。第二个参数向量存储未知数。还必须指定解的第一个近似值（第三个参数向量）。

第 3 章

Решить

Начальные приближения

$$\begin{bmatrix} a \\ x_{0L} \\ h_L \\ x_{0R} \\ h_R \end{bmatrix} := \begin{bmatrix} 6 \\ 4 \\ 1 \\ 3 \\ 1 \end{bmatrix} \boldsymbol{m}$$

Ограничения

$$y_L = y\left(x_L, a, x_{0L}, h_L\right) \qquad y_R = y\left(x_R, a, x_{0R}, h_R\right)$$

$$y\left(x_M, a, x_{0L}, h_L\right) = y\left(x_M, a, x_{0R}, h_R\right)$$

$$S = \int_{x_L}^{x_M} \sqrt{1 + y'\left(x, a, x_{0L}\right)^2}\, dx + \int_{x_M}^{x_R} \sqrt{1 + y'\left(x, a, x_{0R}\right)^2}\, dx$$

$$a \cdot m_c \cdot g \cdot \left|y'\left(x_L, a, x_{0L}\right)\right| + a \cdot m_c \cdot g \cdot \left|y'\left(x_R, a, x_{0R}\right)\right| = M \cdot g + m_c \cdot g \cdot S$$

Решатель

$$\begin{bmatrix} a \\ x_{0L} \\ h_L \\ x_{0R} \\ h_R \end{bmatrix} := \mathbf{Find}\left(a, x_{0L}, h_L, x_{0R}, h_R\right) = \begin{bmatrix} 6.3346454 \\ 3.9229197 \\ 0.745987 \\ 2.256402 \\ -0.3524955 \end{bmatrix} \boldsymbol{m}$$

(a)

$$\begin{bmatrix} a \\ x_{0L} \\ h_L \\ x_{0R} \\ h_R \end{bmatrix} := \mathrm{roots}\left(\begin{bmatrix} y_L = y\left(x_L; a; x_{0L}; h_L\right) \\ y_R = y\left(x_R; a; x_{0R}; h_R\right) \\ y\left(x_M; a; x_{0L}; h_L\right) = y\left(x_M; a; x_{0R}; h_R\right) \\ S = \int_{x_L}^{x_M} \sqrt{1 + y'\left(x; a; x_{0L}\right)^2}\, dx + \int_{x_M}^{x_R} \sqrt{1 + \left(y'\left(x; a; x_{0R}\right)\right)^2}\, dx \\ a \cdot m_c\ g_з \cdot \left|y'\left(x_L; a; x_{0L}\right)\right| + a \cdot m_c\ g_з \cdot \left|y'\left(x_R; a; x_{0R}\right)\right| = M\ g_з + m_c\ g_з \cdot S \end{bmatrix}; \begin{bmatrix} a \\ x_{0L} \\ h_L \\ x_{0R} \\ h_R \end{bmatrix}; \begin{bmatrix} 6 \\ 4 \\ 1 \\ 3 \\ 1 \end{bmatrix} \text{м}\right)$$

$$\begin{bmatrix} a \\ x_{0L} \\ h_L \\ x_{0R} \\ h_R \end{bmatrix} := \begin{bmatrix} 6,3346578 \\ 3,9229196 \\ 0,745987 \\ 2,2564019 \\ -0,3524954 \end{bmatrix} \text{м}$$

第 3 章
b)

图 3.14. 解方程组 - а）Mathcad б）SMath

一个有五个未知数的五元方程组，通过 Find（Mathcad）或 roots（SMath）进行数值求解，显示出这样的时刻：

- 固定在左支座上的链条
- 固定在右支座上的链条
- 链段在货物固定点相遇
- 链长保持在给定的常数 S
- 链在支座上的固定力的垂直投影之和等于链与负载的重量

只有第五个等式需要一些解释。正如我们所指出的，通过链线参数 a 和链的比重，可以计算出链的张力的水平投影，它在整个链长上都是恒定的。如果这个力乘以链线在特定点的一阶导数的值，那么我们就得到了链张力力的垂直投影值。回想一下，函数的导数在数值上等于切线斜率的正切。并且链的张力也与链相切。角的正切是对边的长度（力的垂直投影）与相邻边的长度（力的水平投影）的比值。

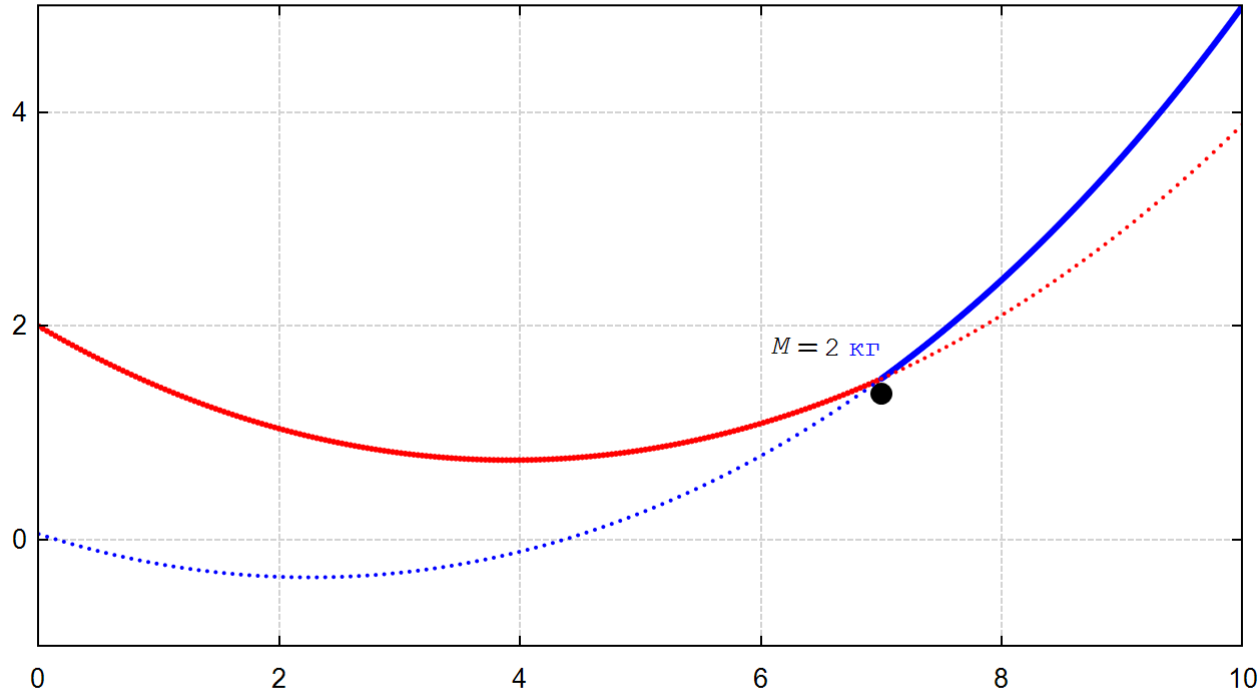
图 3.15 显示了当找到方程组的未知数--链线两段的参数时，我们的带负载的链条将如何下垂。虚线表示链线的幻象延伸。

$$xx_L := \left[x_L, x_L + \frac{x_M - x_L}{200} .. x_M \right] \qquad y1 := \text{augment}\left(xx_L, \overrightarrow{y\left(xx_L, a, x_{0L}, h_L \right)}, ".", 2, "red" \right)$$

$$xx_R := \left[x_M, x_M + \frac{x_R - x_M}{500} .. x_R \right] \qquad y2 := \text{augment}\left(xx_R, \overrightarrow{y\left(xx_R, a, x_{0R}, h_R \right)}, ".", 2, "blue" \right)$$

$$xx_L := \left[x_L, x_L + \frac{x_M - x_L}{100} .. x_M \right] \qquad y3 := \text{augment}\left(xx_L, \overrightarrow{y\left(xx_L, a, x_{0R}, h_R \right)}, ".", 1, "blue" \right)$$

$$xx_R := \left[x_M, x_M + \frac{x_R - x_M}{50} .. x_R \right] \qquad y4 := \text{augment}\left(xx_R, \overrightarrow{y\left(xx_R, a, x_{0L}, h_L \right)}, ".", 1, "red" \right)$$




```
y1
y2
augment(x_M, y(x_M, a, x_0L, h_L) − 0.14, ".", 10, "black")
y3
y4
```

图 3.15. 带负载的松弛链图

图 3.16 显示了作用于链上不同点的力的值的图形显示——该力的水平 F_H（红线）和垂直 F_V（绿线）投影和总力 F（蓝线）。这张图通常被称为作用于结构元素的力图——横梁、悬臂或我们的负载链。图形通常与作用于弯曲的力有关。但链条没有这样的力量。只有断裂的力量。

力的水平投影沿链长是恒定的。垂直投影值是通过了解链线的导数——切线斜率的正切值来确定的。力的总值是根据毕达哥拉斯定理计算的。

$$F_H := a \cdot m_c \; g_s = 62.1215 \qquad F_V(x) := F_H \cdot \begin{vmatrix} \text{if } x < x_M \\ \quad y'(x, a, x_{0L}) \\ \text{else} \\ \quad y'(x, a, x_{0R}) \end{vmatrix} \qquad F(x) := \sqrt{{F_H}^2 + F_V(x)^2}$$

$$X := \left[x_L , x_L + 0.001 .. x_R \right] \qquad XF := \text{augment}\left(X, \overrightarrow{F(X)}\right) \qquad XF_V := \text{augment}\left(X, \overrightarrow{F_V(X)}\right)$$

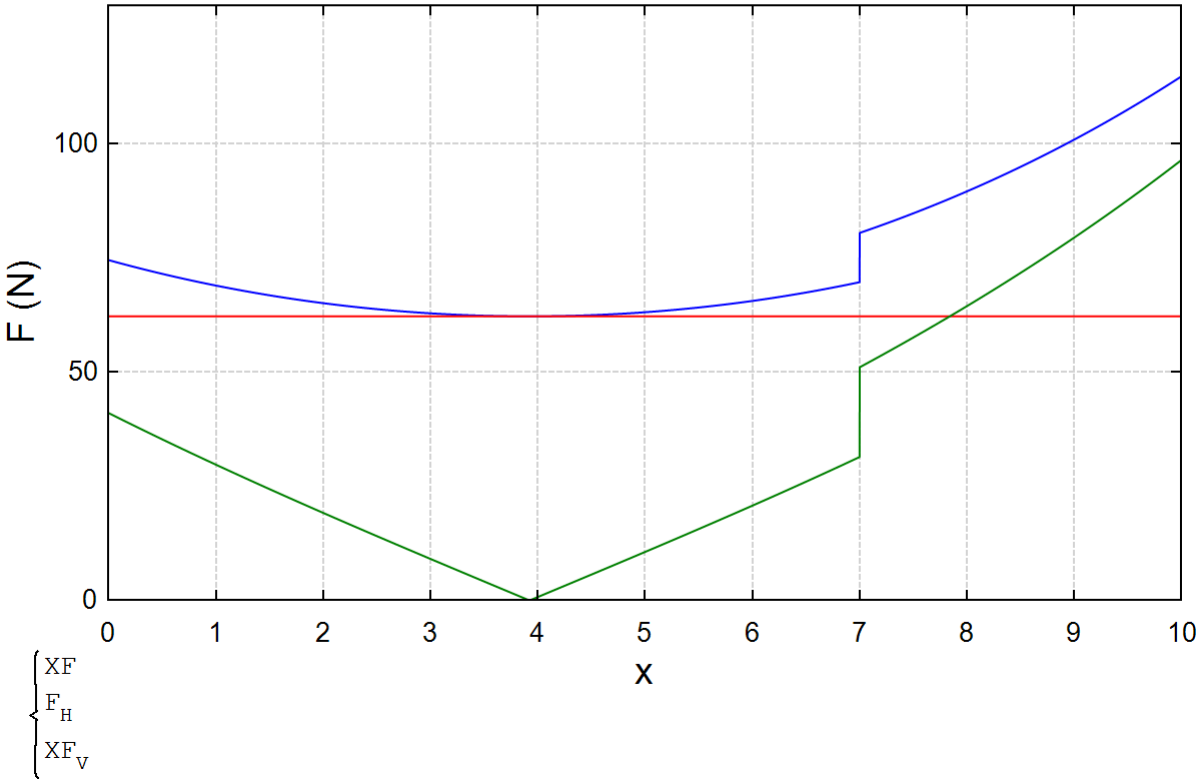


图 3.16. 链条和负载的受力情况

计算的正确性可以通过设置初始条件来评估，以便事先知道答案。在没有负载的情况下，两个链段应该合并成一条链线。如果负载的质量远远大于链条的质量，链条部分将呈现出几乎是直线的字符串形状。这就不再是一个带负载的链条，而是一个所谓的绳状多边形[6]。

力图是理论力学和材料阻力的工具之一。在 SMath 中有许多计算各种结构的例子--见图 3.17。

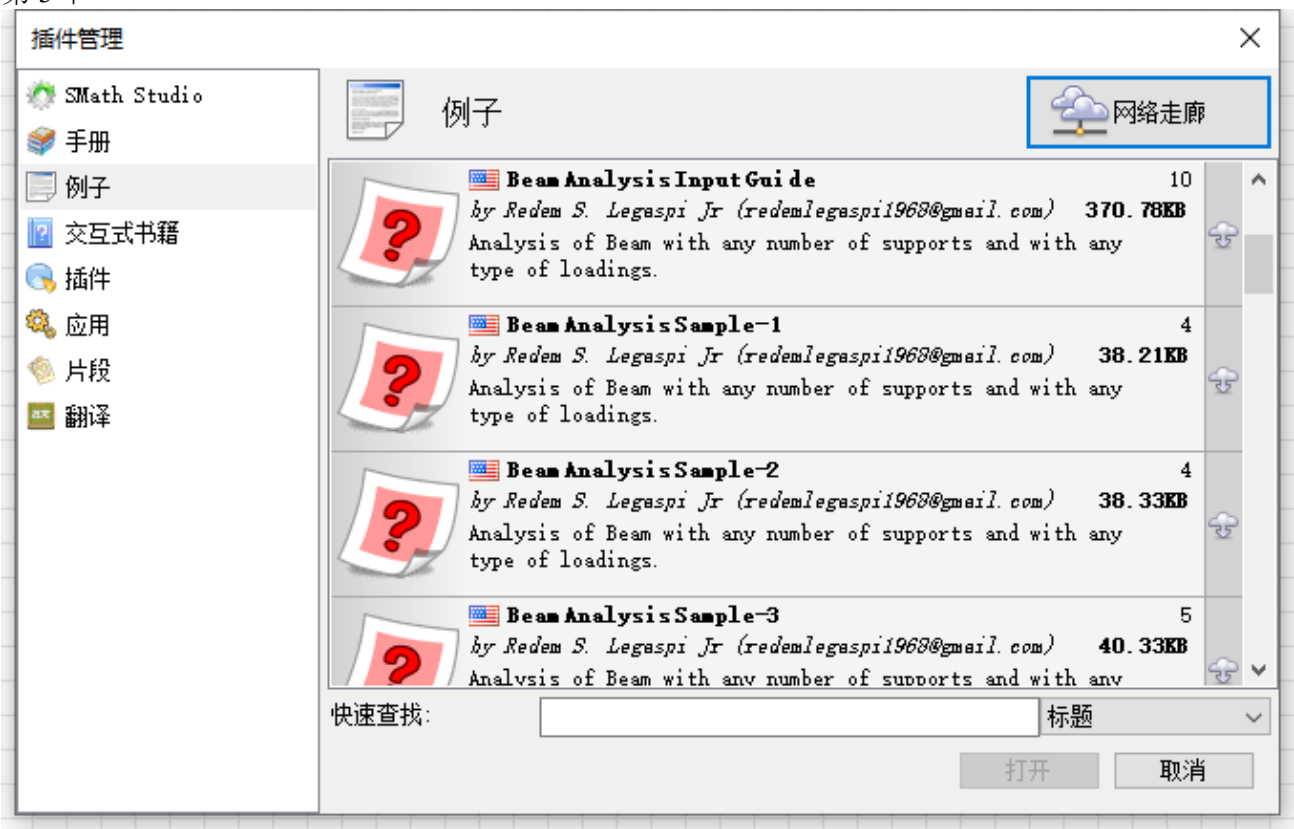


图 3.17. 用于横梁计算的 SMath 库

3.3. 缆车舱

现在让我们设计用于货物运输的缆车的最佳摇篮。

3.3.1. 直角平行六面体

有一个 "流行 "的优化问题。流行，是指它在互联网上被广泛 "搜索"，但其作者却不为人知。如果你在任何搜索引擎上使用关键词 "最大体积箱 "进行搜索，就很容易看到这一点。

任务的本质。取一张正方形的纸，在其边角剪出四个相同的较小正方形——参见图 3.18 中动画的第一帧。此外，从这样的十字形工件中，通过向上弯曲矩形区域来折叠一个盒子-参见动画的其他三帧。

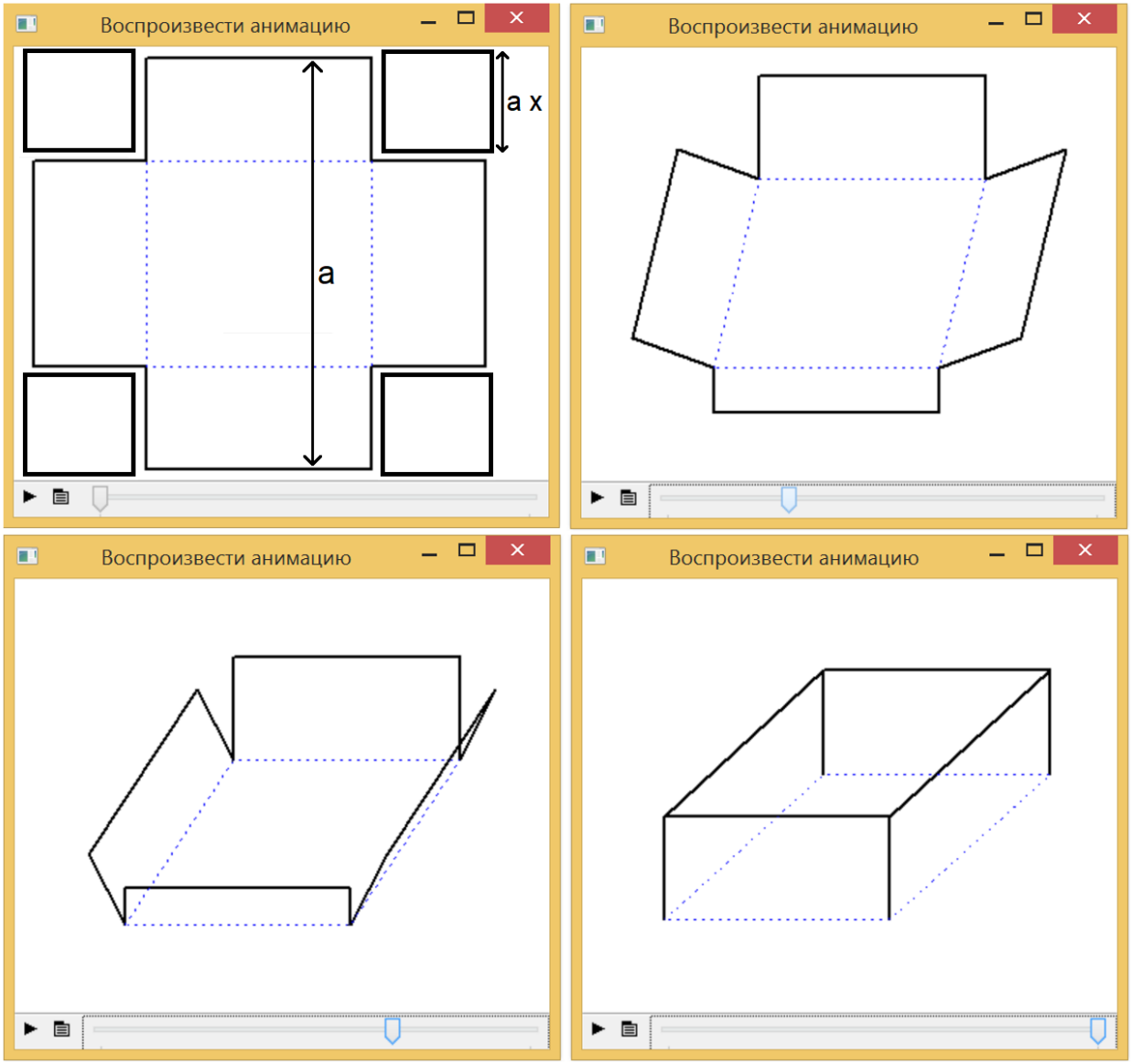


图 3.18. 从正方形空白制作盒子的动画帧

动画的四帧如图 3.18 所示，是这样制作的：在计算中引入了存储十字形工件（动画的第一帧）和相同点的十二个直角顶点坐标的向量。折叠的盒子（最后一帧）。然后，工件角的顶点坐标值平滑地变化为折叠盒的坐标值。最后一帧动画缺乏透视，所以盒子的底座似乎不是正方形，而是长方形。但动画是可以改进的。

问题是，为了使收到的盒子的体积最大，切割的正方形应该是多少？在问题的示意图中，参数 a 是正方形工件的边长，变量 x 是正方形切口边长与参数 a 的比值。

图 3.19 显示了使用 wolframalpha.com 网站解决这个问题的方法，该网站的输入窗口包含一个函数，该函数返回具有单位边长的折叠盒子的体积-盒子底座面积的乘积$(1 - 2x)^2$

到其高度 x。这个函数是三次多项式，取一阶导数。得到一个求零的平方多项式。所有这些都可以通过在纸上描述解决方案的过程在头脑中完成，但我们现在越来越多地在计算机上进行类似的分析转换。平方多项式有两个零：1/2 是函数的局部最小值（盒子的零体积）和 1/6 是所需的最大值。这是我们“流行”的纸箱切割优化问题的解决方案：切割的四个正方形的边长必须等于原始正方形毛坯边长的六分之一。

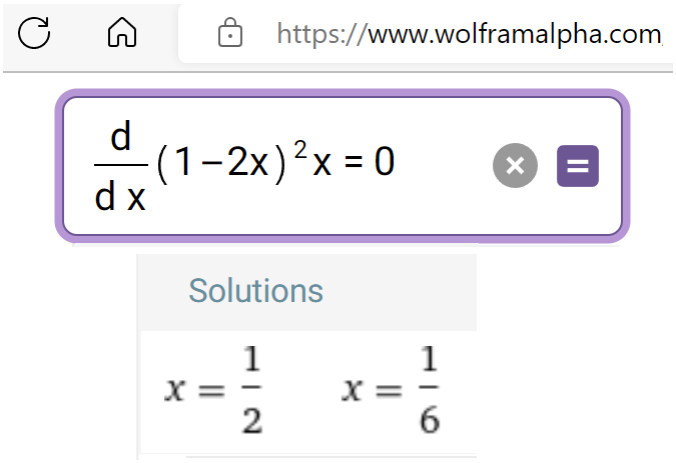


图 3.19. 解决最大体积盒问题

我们重申，图 3.19 所示的解决方案可以在许多论文和互联网上找到。但是最优箱体问题有一个相当意外和有趣的延续[7]，由于可以理解的原因，在前计算机时代人们害怕 ″进入 ″这个问题。

从原工件上切下的四个方块（见图 3.18 中动画的左上角一格），可以用同样的切割模式制作四个相同的、更小的新盒子。同样，16 个(4·4)新方块可以用来制作带有新边角料的新盒子。六十四个(16·4)新的切割方块又可以用来制作更小的新盒子，以此类推，无穷无尽。在数学中，这种通过重复相同操作但在新的层次上形成的物体被称为分形[8]。如果用这个关键词在同一个互联网上进行搜索，可以找到各种形式的分形的图片和描述，包括由正方形或长方体组成的分形，也包括以正方形为基础的尺寸递减的平行四边形（盒子），如我们刚才描述的那些。

但我们的 ″盒状 ″分形可以尝试进行优化--确定正方形切口的边的尺寸，在这个尺寸下，所形成的盒子的总体积将达到最大。

图 3.20 显示了找到将一个正方形切割成五个盒子的最佳方法--一个大盒子和四个小盒子。很明显，在切割四个小方块的第二步中，必须保持 1/6 的比例，以便使这四个小盒子的

总体积达到最大。这个比例（x/6）被固定在图中的输入窗口中。在这里，许多人倾向于认为在切割一个大盒子时必须保持 1/6 的比例。但事实并非如此，这个比例略高于六分之一（这将在图 3.21 中显示）。

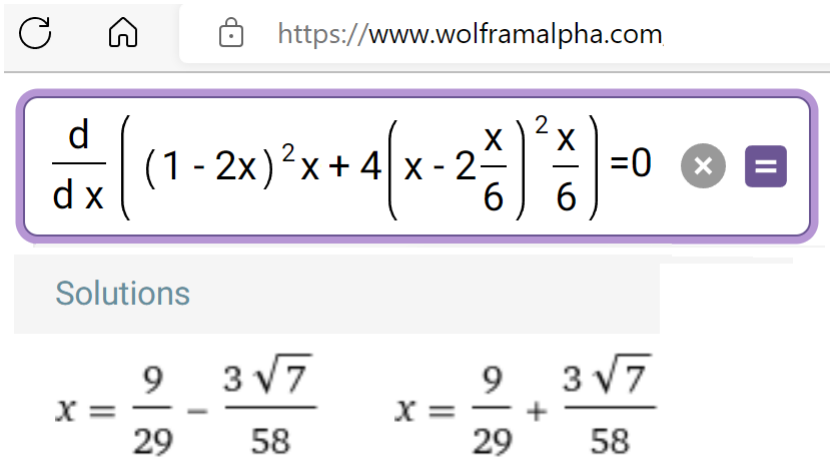


图 3.20. 求解五个盒子的最大体积

第三个嵌套步骤，给出了 21 个盒子--一个大的、四个中的和 16 个小的--也显示在图 3.21 中。3.21.使用的方法不是分析性的（图 3.19 和 3.20），而是数值性的，特别是名为 solve 的 SMath 函数。同时，在数值上重复的是在分析解中得到的答案，六分之一（图 3.19）和第一个表达式的平方根为 7（图 3.20）。

第 3 章

$a := 1\ \mathrm{m} \qquad V_1(x) := (a - 2\cdot a\cdot x)^2 \cdot a \cdot x$

$$x_{opt1} := \mathrm{solve}\left(\frac{\mathrm{d}}{\mathrm{d}\,x} V_1(x),\ x,\ 0.1,\ 0.2\right) = 0.16666666649975$$

$$V_5(x) := V_1(x) + 4\cdot\left(a\cdot x - 2\cdot a\cdot x\cdot x_{opt1}\right)^2 \cdot a\cdot x\cdot x_{opt1}$$

$$x_{opt5} := \mathrm{solve}\left(\frac{\mathrm{d}}{\mathrm{d}\,x} V_5(x),\ x,\ 0.1,\ 0.2\right) = 0.17349562184136$$

$$\frac{9}{29} - \frac{3\cdot\sqrt{7}}{58} = 0.173495621841487$$

$$V_{21}(x) := V_5(x) + 16\cdot\left(a\cdot x\cdot x_{opt1} - 2\cdot a\cdot x\cdot x_{opt1}\cdot x_{opt5}\right)^2 \cdot a\cdot x\cdot x_{opt1}\cdot x_{opt5}$$

$$x_{opt21} := \mathrm{solve}\left(\frac{\mathrm{d}}{\mathrm{d}\,x} V_{21}(x),\ x,\ 0.1,\ 0.2\right) = 0.17363620367393$$

$x := [0, 0.0001..0.5]$

$$xV_1 := \mathrm{augment}\left(x, \frac{\overrightarrow{V_1(x)}}{\mathrm{L}}, ".", 0.5, \text{"red"}\right)$$

$$xV_5 := \mathrm{augment}\left(x, \frac{\overrightarrow{V_5(x)}}{\mathrm{L}}, ".", 0.5, \text{"blue"}\right)$$

$$xV_{21} := \mathrm{augment}\left(x, \frac{\overrightarrow{V_{21}(x)}}{\mathrm{L}}, ".", 0.5, \text{"black"}\right)$$

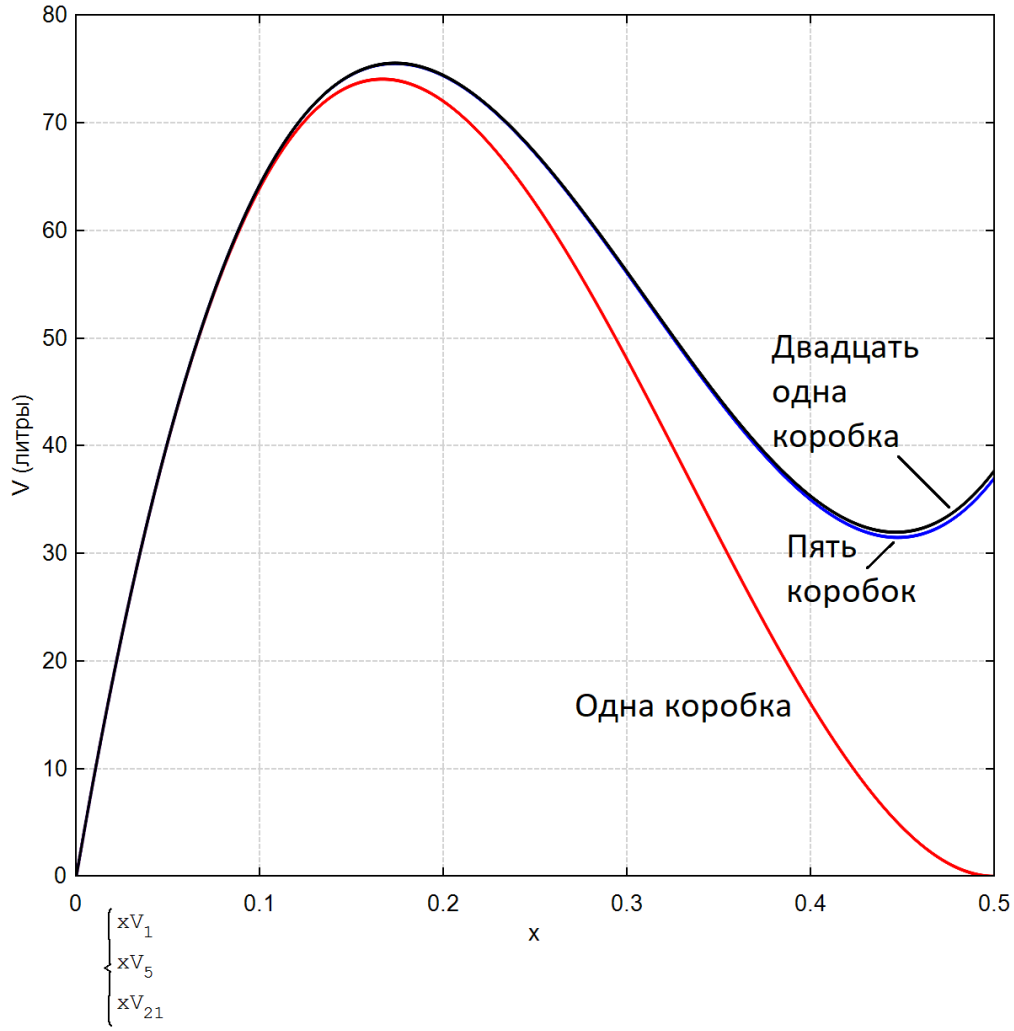


图 3.21 解决 1 个、5 个和 21 个最大体积盒的问题

从图 3.21 的图表和数值可以看出，第四步的切割将使 21 盒（1+4+16）的总体积增加非常小。

五个盒子的问题——一个大的和四个小的问题可以以不同的方式解决，暂时忘掉 1/6 的比例（通过切割四个小的外围方块-见图 3.20）分析 V_5 函数，它不只有一个参数（图 3.21），而是有两个参数：x（切割第一步的比例）和 y（切割第二步的比例-见图 3.22，其中构建了相应的曲面）。

$$f(x,y):=\begin{bmatrix}(1-2\cdot x)^2\cdot x+4\cdot(x-2\cdot x\cdot y)^2\cdot x\cdot y\\ x\\ y\end{bmatrix}$$

$$S:=\text{CreateMesh}\left(f(x,y),-0.1,0.7,-0.1,0.7,40,40\right)$$

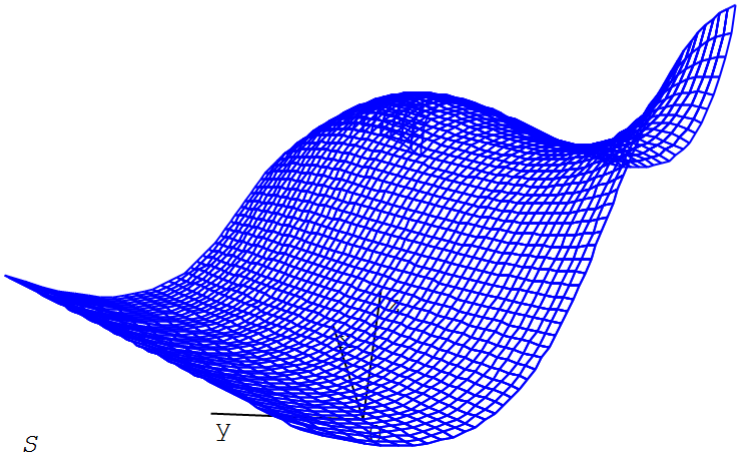
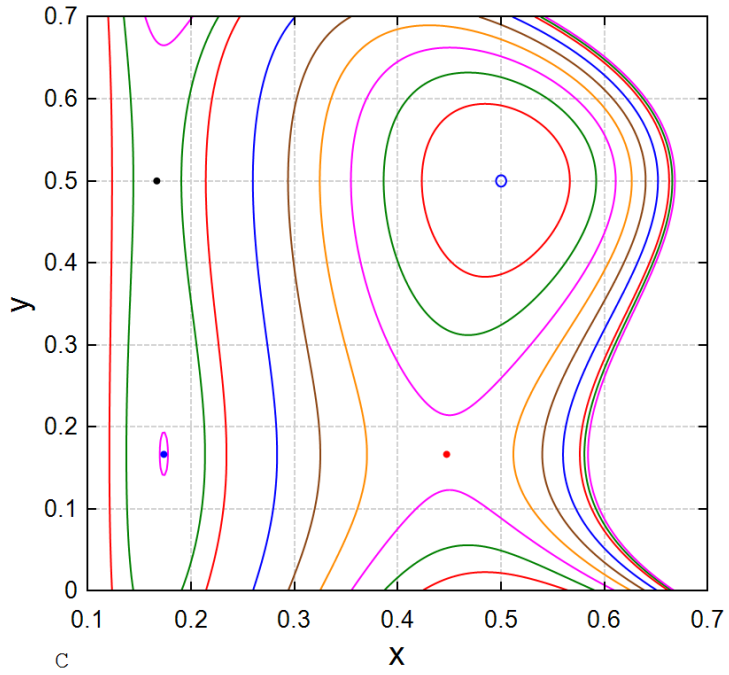


图 3.22. 最大体积五箱问题的图解——曲面

在曲面上，如果您用鼠标旋转它（不是在文本中，而是在 Smath 环境中），如果您愿意，您可以看到特殊点-最多、最少和两个鞍点。这些点在图 3.23 中清晰可见，这里构造了一个名为 V5 的隐式函数的线族。因此，得到的不是曲面（见图 3.22），而是等高线图——一级线图，上面还标有三点：最大点和两个鞍点。最小点(x=0.5，y=0.5)由等值 0.00005 的圆线圈出。最大值点用同一水平的椭圆形线圈定。从图 3.21 中的图表可以看出这是最大值。

$$V_5(x,y) := (1-2\cdot x)^2\cdot x + 4\cdot(x-2\cdot x\cdot y)^2\cdot x\cdot y$$

$$C := \begin{pmatrix} V_5(x,y)-0.00005 \\ V_5(x,y)-0.01 \\ V_5(x,y)-0.02 \\ V_5(x,y)-0.03 \\ V_5(x,y)-0.04 \\ V_5(x,y)-0.05 \\ V_5(x,y)-0.06 \\ V_5(x,y)-0.07 \\ V_5(x,y)-0.073 \\ V_5(x,y)-0.0755 \\ \left[\frac{1}{6}\ \frac{1}{2}\ \text{"."}\ 3\ \text{"black"}\right] \\ \left[\frac{9}{29}+\frac{3\cdot\sqrt{7}}{58}\ \frac{1}{6}\ \text{"."}\ 3\ \text{"red"}\right] \\ \left[\frac{9}{29}-\frac{3\cdot\sqrt{7}}{58}\ \frac{1}{6}\ \text{"."}\ 3\ \text{"Blue"}\right] \end{pmatrix}$$



$$\text{roots}\left(\begin{bmatrix}\frac{d}{dx}V_5(x,y)=0\\ \frac{d}{dy}V_5(x,y)=0\end{bmatrix},\begin{bmatrix}x\\y\end{bmatrix},\begin{bmatrix}0.15\\0.15\end{bmatrix}\right)=\begin{bmatrix}0.173495621841487\\0.166666666666666\end{bmatrix}\quad\begin{matrix}\frac{9}{29}-\frac{3\cdot\sqrt{7}}{58}=0.173495621841487\\ \frac{1}{6}\end{matrix}$$

$$\text{roots}\left(\begin{bmatrix}\frac{d}{dx}V_5(x,y)=0\\ \frac{d}{dy}V_5(x,y)=0\end{bmatrix},\begin{bmatrix}x\\y\end{bmatrix},\begin{bmatrix}0.4\\0.15\end{bmatrix}\right)=\begin{bmatrix}0.447194033330927\\0.166666666666666\end{bmatrix}\quad\begin{matrix}\frac{9}{29}+\frac{3\cdot\sqrt{7}}{58}=0.447194033330927\\ \frac{1}{6}\end{matrix}$$

$$\text{roots}\left(\begin{bmatrix}\frac{d}{dx}V_5(x,y)=0\\ \frac{d}{dy}V_5(x,y)=0\end{bmatrix},\begin{bmatrix}x\\y\end{bmatrix},\begin{bmatrix}0.15\\0.5\end{bmatrix}\right)=\begin{bmatrix}0.166666666666666\\0.499999999999999\end{bmatrix}\quad\begin{matrix}\frac{1}{6}\\ \frac{1}{2}\end{matrix}$$

图 3.23. 五箱最大容积问题的图形和数值解法——一级线

在图 3.23 的轮廓图下，显示了对 Smath 中内置的 roots 函数的三次调用，在未知 x 和 y 的不同初值下求解两个方程的相同系统。

这是 MEI A I.Tikhonov 教授如何在 Python 生态系统中解决这个问题，为此连接符号计算库 Sympy（程序设计中文字母-见第 6 章）。您还需要库来处理 numpy 数组和呈现 matplotlib。所有计算都将在 Jupyter 笔记本(JN)或 Jupyter Lab(JL)中进行。

```
%matplotlib inline
Import numpy as NP
Import matplotlib.pyplot as plt
```

第 3 章

```
Import Sympy as SMP
```

首先，让我们声明字符变量并使用 LaTeX 初始化字符表达式的输出：

```
a,x,v1=smp.symbols('a x v1'
smp.init_printing()
```

盒子的体积、体积的 x 导数和确定极值必要条件的方程的解如下所示：

```
V1=A**3*(1-2*X)**2*X
v1s=smp.diff(v1,x)
x1=smp.solve(v1s,x)
```

我们将以字符(x1)和数字(x1_)的形式显示值。在后一种情况下，我们将不得不调用 evalf 方法，该方法允许您获得小数点后给定有效位数的结果。

```
x1_=[xx.evalf(5)for xx in x1]
x1,x1_
```

我们会得到结果：

$$\left(\left[\frac{1}{6},\frac{1}{2}\right],[0.16667,0.5]\right)$$

我们有两个解，所以我们将它们代入盒子体积的字符表达式，得到正数和小数形式的解。此时，我们认为盒子的边长 a=1：

```
V11=[V1.subs(x,xx).subs(a,1)for xx in x1]
V11_=[V1.subs(x,xx).subs(a,1).evalf(5)for xx in x1]
V11,V11_
```

结果如下所示：

$$\left(\left[\frac{2}{27},0\right],[0.074074,0]\right)$$

该解决方案与 Smath 中获得的解决方案没有区别。让我们用剩下的四个方块重复这个操作。与 Smath 不同，这里不固定第一步获得的 x 值，而是在每一步确定满足必要条件极值的 x 值。

```
V2=smp.expand(smp.simplify(V1+4*(a*x)**3*(1-2*x)**2*x))
V2
```

在这里，使用 Simplify 方法简化了五个盒子体积的字符表达式：

$16a^3x^6-16a^3x^5+4a^3x^4+4a^3x^3-4a^3x^2+a^3x$

和前面一样，取 x 的导数：

$96a^3x^5-80a^3x^4+16a^3x^3+12a^3x^2-8a^3x+a^3$

解方程 v2(X)=0：

```
x2=smp.solve(v2s,x)
```

正如预期的那样，我们收到了五个解决方案，但由于它们的繁琐，我们在这里没有以字符形式给出，而是立即计算数值，如下所示：

```
x2_=[xx.evalf(5)for xx in x2]
x2_
[0.5,0.31116-0.40393 i,0.31116+0.40393 i,0.17333,-0.46232]
```

我们得到了两个完全没有物理意义的解和两个实数，我们对第一个不感兴趣，因为对应于零箱容积。紧接着可以为所有解决方案输出五箱的体积：

```
[v2.subs(x,xx).subs(a,1).evalf(5)for xx in x2_]
```

我们感兴趣的结果以粗体突出显示

[0.12632+0.18435i, 0.12632-0.18435i, **0.075528**, -1.0356]

我们可以通过编写盒子体积的递归公式来更进一步：

```
v3=smp.expand(smp.simplify(v2+16*(a*x**2)**3*(1-2*x)**2*x))
v3s=smp.diff(v3,x)
x3=smp.solve(v3s,x)
x3_=[xx.evalf(5)for xx in x3]
```

唯一的困难是选择正确的解决方案。判据为零虚部和根不等式 0.5。

因此得到解 x3=0.17364, V3= 0.07556, x4= 0.17365, V4= 0.07556…

如果我们在上一步(n-1)中有盒子的体积，我们将其指定为 Vp，那么可以计算第 n 步 Vn 中的盒子体积：

```
n, Vp, Vn = smp.symbols('n, Vp, Vn')
Vn = 4**(n-2)*(a*x**(n-2))**3*(1-2*x)**2*x
```

现在让我们转向数值计算，计算一步体积的附加值。让我们以 Python 函数的形式执行此操作：

```
def vn(x, n, a=1):
    return 4**(n-1)*(a*x**(n-1))**3*(1-2*x)**2*x

m = 1000
xnum = np.linspace(0,.5, m)

styles = 'k-', 'k--', 'k-.', 'k:'
for q in range(3,7):
    plt.plot(xnum, vn(xnum, q), lw=3, label=f'n={q}')
plt.grid()
plt.yscale('log')
plt.ylim(1e-9, 1)
```

```
plt.legend(loc='best');
plt.xlabel('x')
plt.ylabel('Vn')
```

图 3.24 显示了步骤 n 的体积增加的依赖性

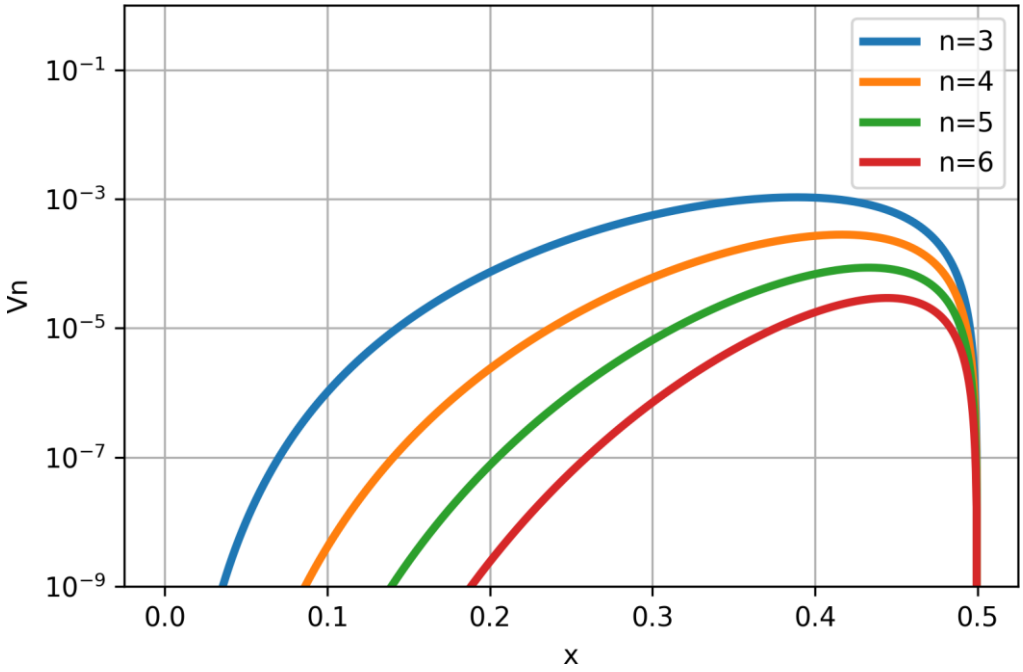


图 3.24. 对步骤 3-7，添加剂与切口相对值 x 的体积关系

根据切口的大小，在施工步骤 n 处的箱体体积见图 3.25。为了计算它，使用了一个函数，其源代码如下所示：

```
def v(x, n, a=1):
    vv = 0
    for i in range(1, n+1):
        vv += vn(x, i, a=a)
    return vv
```

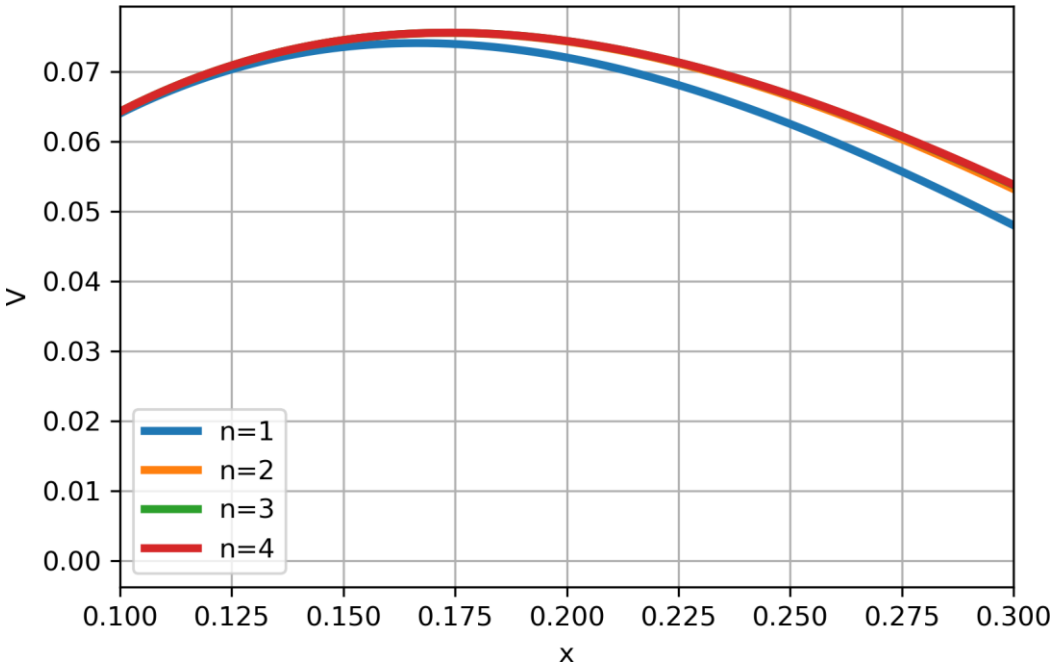


图 3.25. 根据施工步骤 n=1…4 的切口大小，纸箱体积的关系式

从图 3.25 可以看出，第二步和后续步骤中的解决方案几乎没有区别。

3.3.2. 锥体

这是另一个相当有趣的任务，用于使用废物处理来切割工件。在缆车上运输散装货物时，需要的不是长方体形式的摇篮（箱-见上文），而是锥体形式的摇篮。

根据简单的技术，从圆形工件中制作成直圆锥体形式的容器：切割角度为 α (alpha) 的扇形，然后将所得图案折叠成锥体，并焊接焊缝（焊接、粘合-见图 3.26）。曾几何时，商店里的卖家巧妙地用一张粗糙的灰色包装纸卷起类似的圆锥，并将砂糖、谷物和其他杂货倒入由此产生的碎片中。

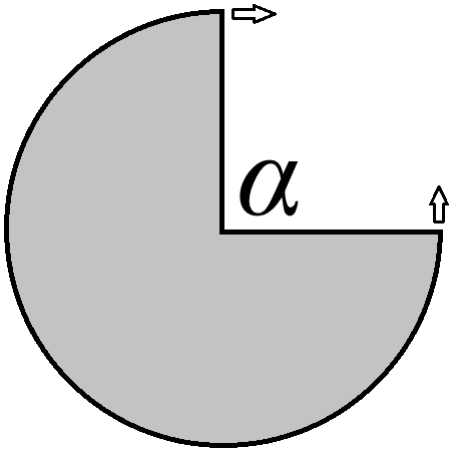


图 1。3.26. 锥形电容问题：一种解法方案

要求求出圆锥体积最大的切角 α 。消防桶一般呈锥形。

我们将立即考虑 Smath 上问题的数值解-见图 3.27，其中以圆形工件的半径为单位。更准确地说，一米（这里使用测量单位，以便在输入公式时不会出错）。创建了两个辅助函数，分别名为 r（锥体基底半径）和 h（其高度），然后形成函数 V_1（单个锥体的体积）。里脊没有被扔掉-第二个容器以锥体的形式从里面卷起。函数 V_2 是两个锥体的总和。接下来，绘制两个图形，显示一个圆锥和两个圆锥的体积随切口角度的变化。图中还引入了第二个（右）纵坐标轴，其格式化工具如图 3.28 所示。它们是通过双击图形来调用的。出现的菜单格式化了第二个纵坐标轴的参数（Y2-Axis-参见图 3.28 的顶部），并通过<list...>位置调用 Series 集合编辑器窗口，其中 ISY2Data 位置设置为 true。描述所有这些是相当困难的-您只需要使用这些工具。或者您可以不引入额外的纵坐标轴，而是在另一个下面构建两个独立的图形，使它们的横坐标轴的比例相同。这也更可取，因为曲线的交点与方程组的图形解有关（例如，见第 10 章图 10.13）。曲线的交点就是方程组的根。在图 3.27 中，曲线的交点没有物理意义，但你可以为了练习而查找其坐标。

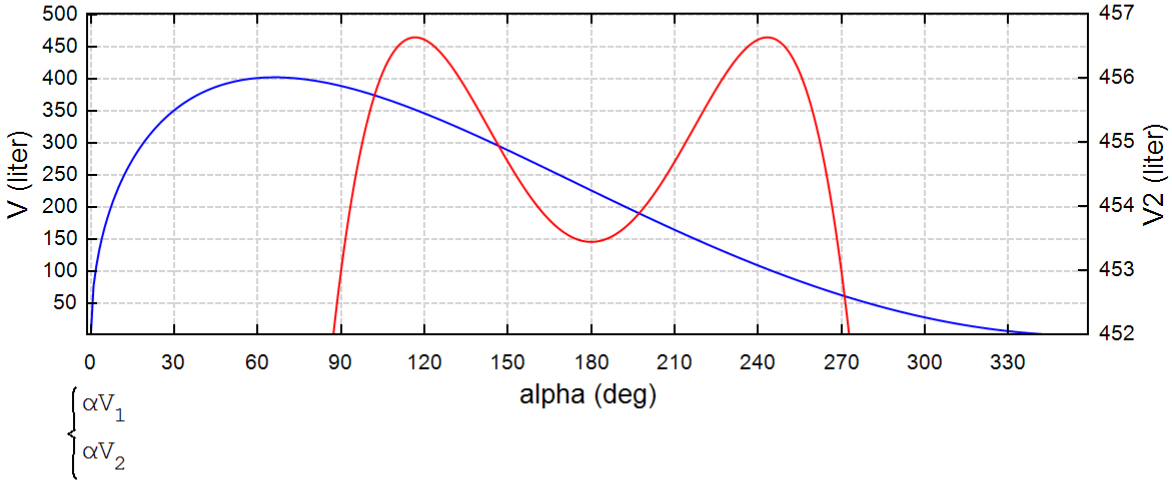
第 3 章

$R := 1\ \text{m}$ $\quad r(\alpha) := R \cdot \left(1 - \frac{\alpha}{2 \cdot \pi}\right)$ $\quad h(\alpha) := \sqrt{R^2 - r(\alpha)^2}$

$V_1(\alpha) := \frac{1}{3} \cdot \pi \cdot r(\alpha)^2 \cdot h(\alpha)$ $\quad V_2(\alpha) := V_1(\alpha) + V_1(2 \cdot \pi - \alpha)$

$\alpha := [\ 0\ \text{deg}\ ,1\ \text{deg}\ ..\ 360\ \text{deg}\ ]$

$\alpha V_1 := \text{augment}\left(\frac{\alpha}{\text{deg}}, \frac{\overrightarrow{V_1(\alpha)}}{\text{L}}\right)$ $\quad \alpha V_2 := \text{augment}\left(\frac{\alpha}{\text{deg}}, \frac{\overrightarrow{V_2(\alpha)}}{\text{L}}\right)$



$\begin{bmatrix} \alpha_{opt1} \\ No \end{bmatrix} := \left(\text{solve}\left(\frac{d}{d\alpha} V_1(\alpha) = 0,\ \alpha\right)\right) = \begin{bmatrix} 66.0613 \\ 360 \end{bmatrix} \text{deg}$ $\quad V_1(\alpha_{opt1}) = 403.07\ \text{L}$

$\begin{bmatrix} \alpha_{opt2} \\ No \\ No \end{bmatrix} := \left(\text{solve}\left(\frac{d}{d\alpha} V_2(\alpha) = 0,\ \alpha\right)\right) = \begin{bmatrix} 116.6448 \\ 180 \\ 243.3549 \end{bmatrix} \text{deg}$ $\quad V_2(\alpha_{opt2}) = 456.64\ \text{L}$

图 3.27. 一个和两个锥形容器的图形和数值解决方案

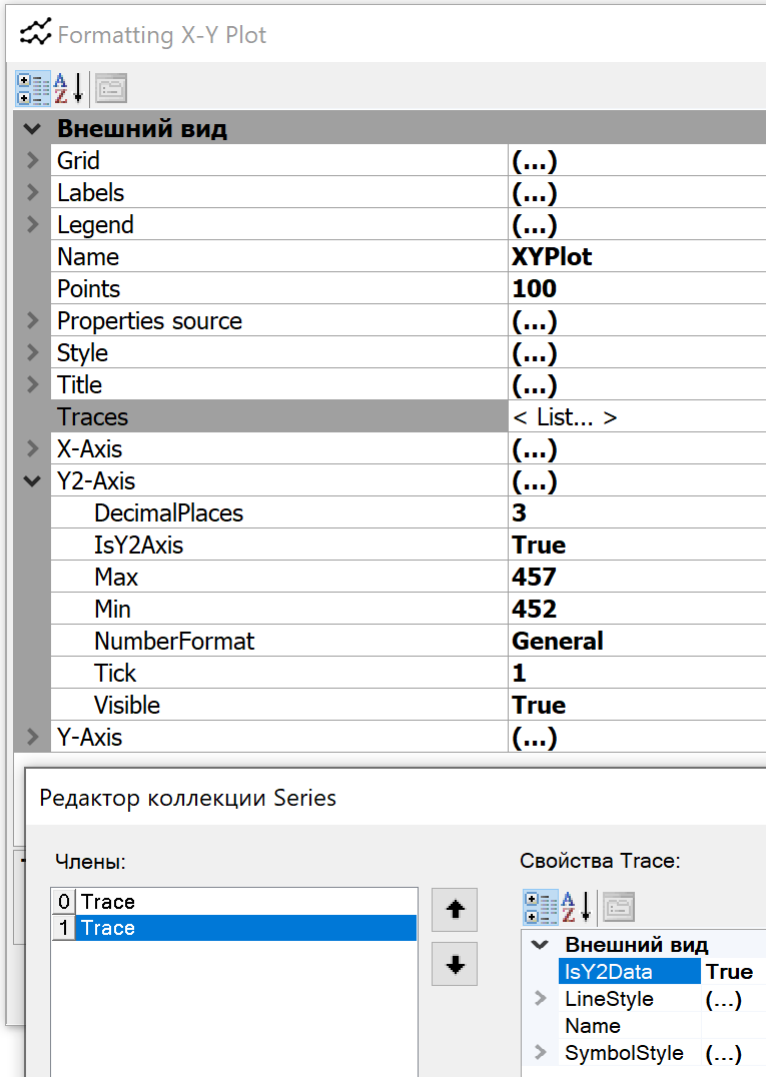


图 3.28. 笛卡尔图形第二条序数轴的创建工具

为了准确确定最佳切割角度，我们使用了我们已经熟悉的 solve 函数，但不是四个参数，而是两个参数。图 3.21 中的判决中有四个论点。双参数 solve 函数返回的不是一个，而是两个（一个锥形电容）和三个（两个电容）响应。向量的第一个元素是我们问题的解决方案。

我们将使用数值方法解决 Python 中的两个锥体问题（MEI A.I.Tikhonov 教授的解决方案）。首先，让我们使用单行准备函数：

```
R = 1 # 工件半径
r = lambda alpha: R * (1 - alpha/360) # alpha 的单位是度
h = lambda alpha: np.sqrt(R**2 - r(alpha)**2)
V1 = lambda alpha: 1/3 * np.pi * r(alpha)**2 *h(alpha)
V2 = lambda alpha: V1(alpha) + V1(360 - alpha)
```

为了绘制图表，准备一个以一千点为单位的 α 角值阵列。

```
m = 1000
alpha = np.linspace(0, 360, m)
```

与 Smath 一样，不同比例的图形在一张图中兼容，为此我们用虚线绘制前两条曲线（V1 和 V2）。

```
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax1.plot(alpha, V1(alpha), ls='dotted', lw=3, label=r'$V1(\alpha)$')
ax1.plot(alpha, V2(alpha), ls='dotted',lw=3, label=r'$V2(\alpha)$')
ax1.set_xlabel(r'$\alpha, г р а д у с ы$')
ax1.set_ylabel(r'$V1(\alpha), V2(\alpha)$')
```

请注意，我们必须创建一个 ax1 图对象，我们进一步将其与另一个 ax2 图进行了匹配。这两幅画都有一个共同的横坐标轴。

第二张图与第一张图的不同之处在于，图表是用实线绘制的，我们手动设置了纵坐标轴上的边界值，因此 V1 的依赖关系没有出现在图中。通过调用 twinx 方法实现图形叠加。第二张图的数字化位于右边。结果如图 3.29 所示。

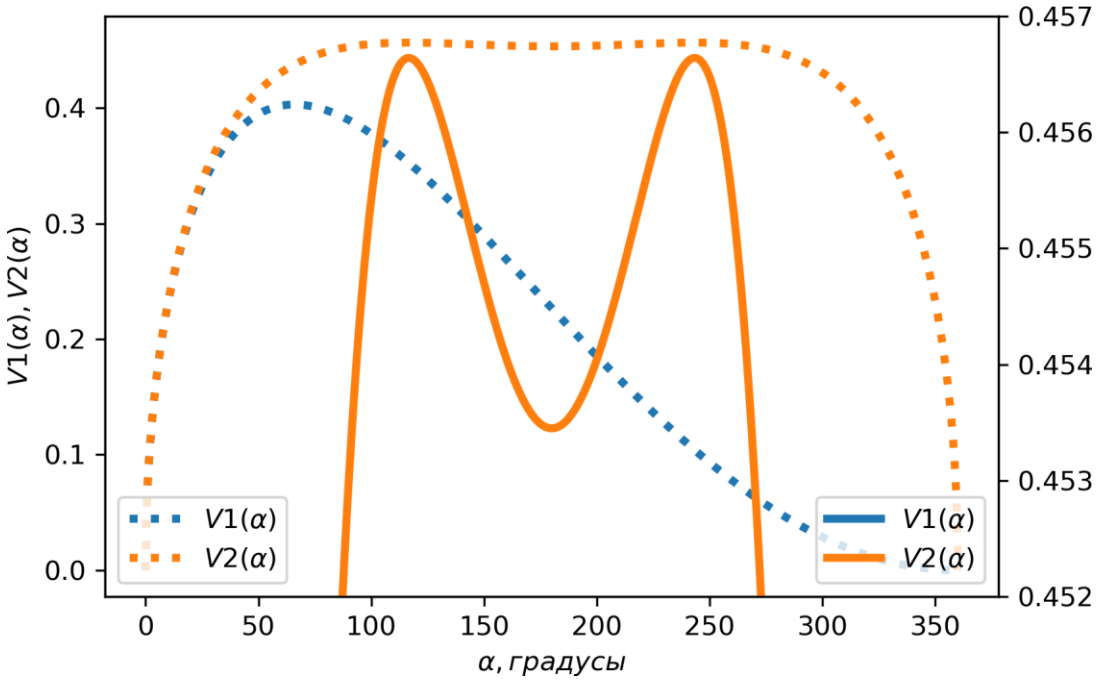


图 3.29. 在 Python 生态系统中构建的一个和两个锥体的体积依赖关系α

我们需要找出锥体最大体积的确切值，近似值可以从图 3.29 中获得。为此，我们将使用 minimize 函数，它至少需要传递最小化函数和初始近似值。返回函数的一个相当复杂的构造，如下所示，此外，由于我们不是在寻找最小值，而是在寻找最大值，因此我们需要更改函数的符号。

```
from scipy.optimize import minimize_scalar, minimize
minimize(lambda alpha: -V2(alpha), 105)
```

我们将得到以下结果：

```
fun: -0.4566383639770894
 hess_inv: array([[108604.40395527]])
      jac: array([-5.5283308e-06])
  message: 'Optimization terminated successfully.'
     nfev: 22
      nit: 3
     njev: 11
   status: 0
  success: True
        x: array([115.83290868])
```

我们对 fun 元素感兴趣-函数在最小点的值，success-复选框，其真值表示优化过程成功完成，以及对应于极值的参数值的 x 数组。使用图 3.29 中的初始值，我们得到：

```
(minimize(lambda alpha: -V2(alpha), 105).x[0],
 minimize(lambda alpha: -V2(alpha), 250).x[0])
```

所需的最大值点：

```
(115.832908677458, 243.717433728684)
```

是否可以用一个圆形工件制造不是两个，而是三个三个锥形容器，其总体积将超过两个容器的体积？在 Smath 环境中解决这个问题的方法在图 3.30 中概述，在图 3.30 中，通过带有余切和正弦的简单几何变换，创建了所谓的三角图，乍一看，第三个锥形电容将是多余的。三角形的边是图 3.27 和 3.29 所示的双峰曲线的俯视图。条边上都有两个最大值（总共有六个），在等边矩形的约束下寻找函数 $\Sigma V(\alpha, \beta)$的最大值时就能达到。

$$r(\alpha) := \frac{\alpha}{2\cdot\pi} \qquad h(\alpha) := \sqrt{1 - r(\alpha)^2} \qquad V(\alpha) := \frac{1}{3}\cdot\pi\cdot r(\alpha)^2\cdot h(\alpha)$$

$$\Sigma V(\alpha, \beta) := \left|\begin{array}{l} \Sigma V'(\alpha, \beta) := V(\alpha) + V(\beta) + V(2\cdot\pi - \alpha - \beta) \\ \alpha' := \alpha - \beta\cdot \mathrm{ctg}\left(\frac{\pi}{3}\right) \\ \beta' := \frac{\beta}{\sin\left(\frac{\pi}{3}\right)} \\ \text{if } \left((\alpha' + \beta' > 2\cdot\pi) \vee (\alpha' < 0)\right) \vee (\beta' < 0) \\ \quad 0 \\ \text{else} \\ \quad \Sigma V'(\alpha', \beta') \end{array}\right.$$

$$C := \begin{pmatrix} \Sigma V(x, y) - 0.45 \\ \Sigma V(x, y) - 0.44 \\ \Sigma V(x, y) - 0.43 \\ \Sigma V(x, y) - 0.42 \\ \Sigma V(x, y) - 0.41 \\ \Sigma V(x, y) - 0.40 \\ \Sigma V(x, y) - 0.39 \\ \Sigma V(x, y) - 0.38 \\ \Sigma V(x, y) - 0.37 \\ \Sigma V(x, y) - 0.36 \\ \Sigma V(x, y) - 0.35 \\ \Sigma V(x, y) - 0.34 \\ \Sigma V(x, y) - 0.33 \\ \Sigma V(x, y) - 0.32 \\ \Sigma V(x, y) - 0.31 \\ \Sigma V(x, y) - 0.10 \\ \Sigma V(x, y) - 0.00 \end{pmatrix}$$

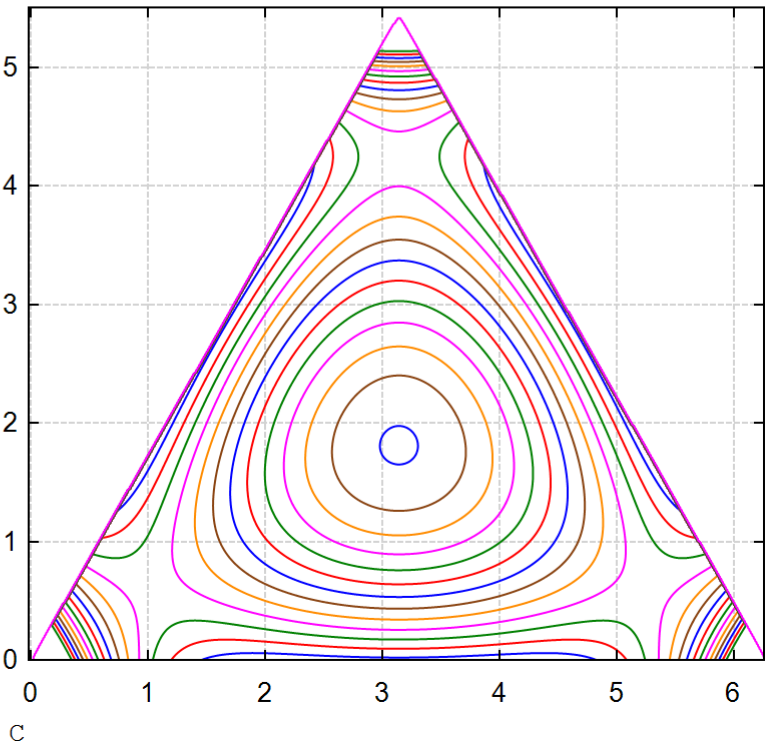


图 3.30. 第三个锥形容器问题：三角坐标系

如果将图 3.30 所示的轮廓图转换为类似于图 3.22 所示的曲面，则会得到一种三角形座椅，座椅中间有局部最小值，三角形座椅的三边有六个最大值。

需要在这里使用三角图，因为所分析函数 ΣV 的参数由 $\alpha+\beta$ 的比值小于或等于 2π 联系起来。如果 $\alpha + \beta$ 等于 2π，那么我们在三角形的边缘。如果 $\alpha+\beta$ 小于 2π，那么我们在三角形内。角度 α 和 β 是，让我们想起从一个圆形工件中切割的角度，用于制造两个锥形容器。

如果函数 ΣV 不受三角形 "$\alpha + \beta$ 小于等于 2π "的限制，而是显示在整个参数值范围内，在这个范围内函数 ΣV 产生有效的值（"螳螂脸"--也可参见第十章图 10.16 中类似的神秘 "脸"），就会得到一个有趣的同一水平线的三角图（图 3.31）。在所有参数值的大等边倒三角形的中间是图 3.30 中的小非倒等边三角形的圆弧，概述了与我们的任务相关的真实参数的区域。在图 3.31 中，可以清楚地看到大三角形角上的三个极大点，小三角形角上和边中间的六个鞍点，以及两个三角形中心的一个局部极小点。在小等边三角形的边上有六个点--解决了两个体积之和最大的圆锥体的问题。第三个锥体是多余的！

$$C := \begin{cases} \Sigma V(x,y) - 1.2 \\ \Sigma V(x,y) - 1.1 \\ \Sigma V(x,y) - 1 \\ \Sigma V(x,y) - 0.9 \\ \Sigma V(x,y) - 0.8 \\ \Sigma V(x,y) - 0.7 \\ \Sigma V(x,y) - 0.6 \\ \Sigma V(x,y) - 0.5 \\ \Sigma V(x,y) - 0.4 \\ \Sigma V(x,y) - 0.3 \\ \Sigma V(x,y) - 0.2 \\ \Sigma V(x,y) - 0.1 \\ \Sigma V(x,y) - 0.09 \\ \Sigma V(x,y) - 0.08 \\ \Sigma V(x,y) - 0.07 \\ \Sigma V(x,y) - 0.06 \\ \Sigma V(x,y) - 0.05 \\ \Sigma V(x,y) - 0.05 \\ \Sigma V(x,y) - 0.04 \\ \Sigma V(x,y) - 0.03 \\ \Sigma V(x,y) - 0.02 \\ \Sigma V(x,y) - 0.01 \\ \Sigma V(x,y) - 0.0 \end{cases}$$

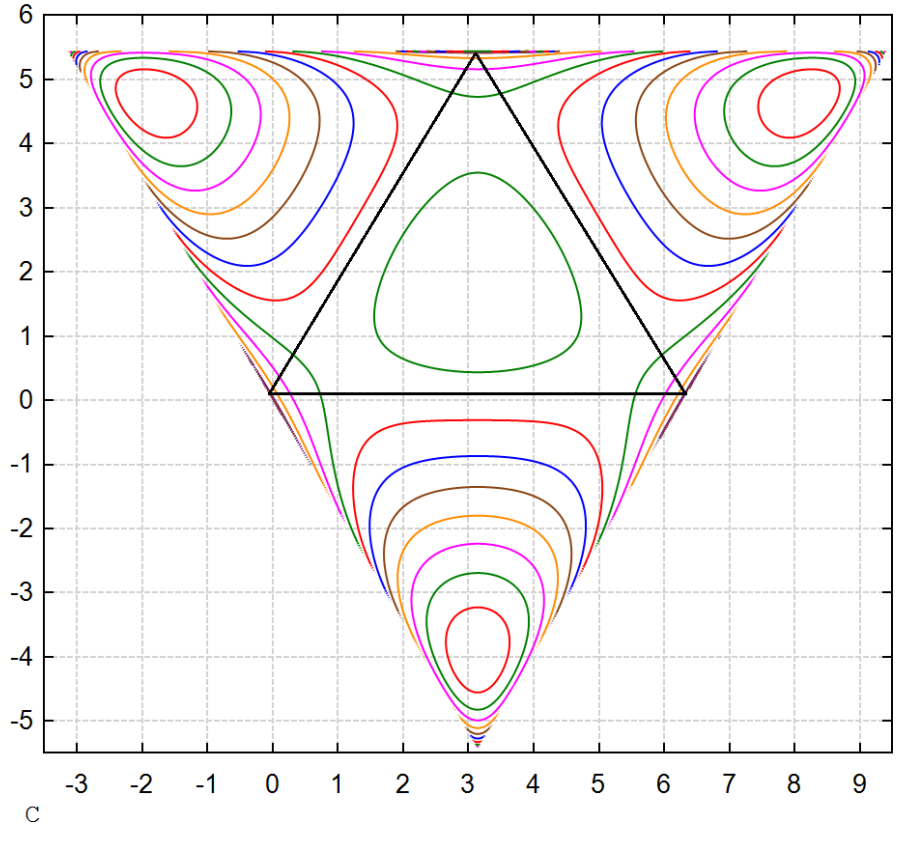


图 3.31. 具有两个参数的全域的三个锥形容器的问题

Python 生态系统让我们有理由练习 NumPy 数组和 3D 图形，尽管您可以使用 matplotlib contour 函数绘制等高线图，就像在 Smath 中一样成功。同时，我们将在 JN 环境中创建一个交互式应用程序。

要绘制，我们需要一个通用的 numpy 函数。通用函数在标量和数组两个方面都同样工作得很好。

```
def SV(alpha, beta):
```

```
    r = lambda alpha: alpha
    h = lambda alpha: np.sqrt(1 - r(alpha)**2)
    V = lambda alpha:  1/3*np.pi*r(alpha)*h(alpha)
    SVS = lambda alpha, beta: V(alpha) + V(beta) + V(1 - alpha - beta)
    alphas = alpha - beta*np.cos(np.pi/3)/np.sin(np.pi/3)
    betas = beta/np.sin(np.pi/3)
    return np.where((alphas+betas>1) | \
           alphas<0) | (betas<0), 0, SVS(alphas, betas))
```

该函数再现了图 13 中的计算，不同之处在于它们位于线段[0,1]上。在 SV 函数中定义了 r, h, V 和 SVS 四个单行函数。请注意，不使用数组的条件 Python 语句已被 NumPy where 函数替换。α,β

要可视化两个变量的函数，您需要在二维网格上计算它们的值：

```
m = 500
alpha, beta = np.linspace(0, 1, m), np.linspace(0, 1, m)
Alpha, Beta = np.meshgrid(alpha, beta)
VAB = SV(Alpha, Beta)
```

这里我们创建了一维数组α,β，二维网格数组 Alpha,Beta，并在上面计算 SV 函数值。我们可以将得到的结果可视化（图 3.23）。

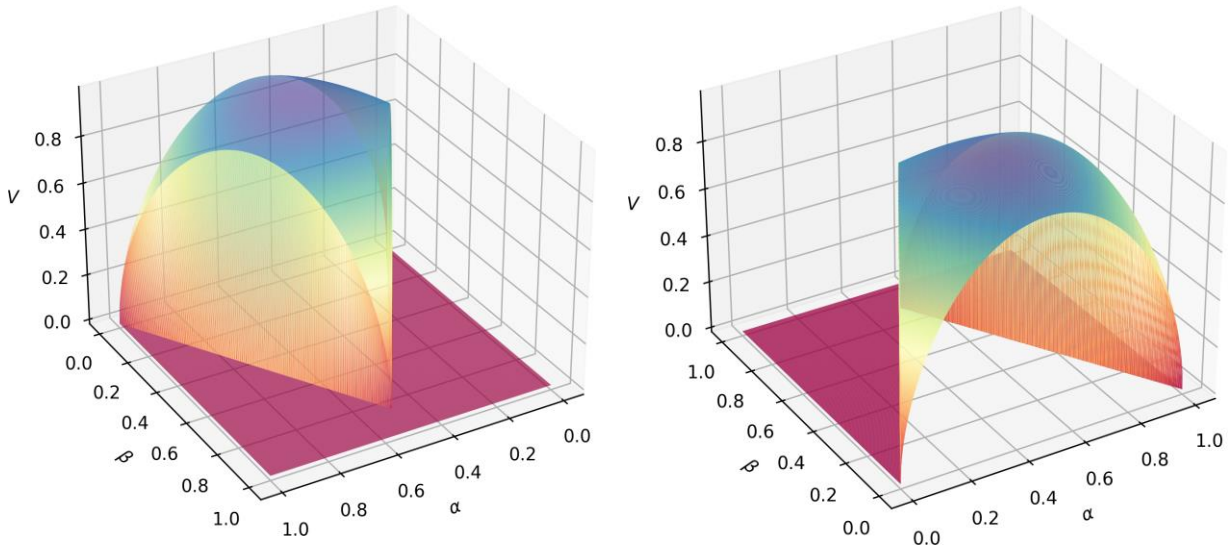


图 3.32. Python 中的三个锥形容器问题

a）围绕垂直轴的旋转角度为 60 度；b）围绕垂直轴的旋转角度为-120 度

在绘制复杂曲面时，建议从不同的角度来观察它们，交互地更改它。您可以在 JN 环境中使用两行原始文本来完成此操作。

```
from ipywidgets import interact_manual, IntSlider
```

```
@interact_manual(azim=IntSlider(min=-180, value=60, max=180),
                 elev=IntSlider(min=-90, value=30, max=90))
def view(azim, elev):
    global fignum
    fig = plt.figure(figsize=(6,6))
    ax = plt.axes(projection='3d')
    stride = 1
    ax.plot_surface(Alpha, Beta, VAB, cmap='Spectral',
                    cstride=stride, rstride=stride)
    ax.set_xlabel(r'$\alpha$')
    ax.set_ylabel(r'$\beta$')
    ax.set_zlabel('$V$')
    ax.view_init(azim=azim, elev=elev)
```

我们将可视化本身设计为视图函数，我们将绕 Azim 垂直轴和 Elev 水平轴的旋转角度作为参数传递给它。可视化分三个步骤实现：使用 Axes 功能创建三维图形，在轴上显示表面和标签，最后将生成的结构绕轴旋转。

要将用户界面添加到 JN 记事本单元格中，导入 IPYWidgets 库并使用 Interact_Manual 装饰函数就足够了（图 3.33）。

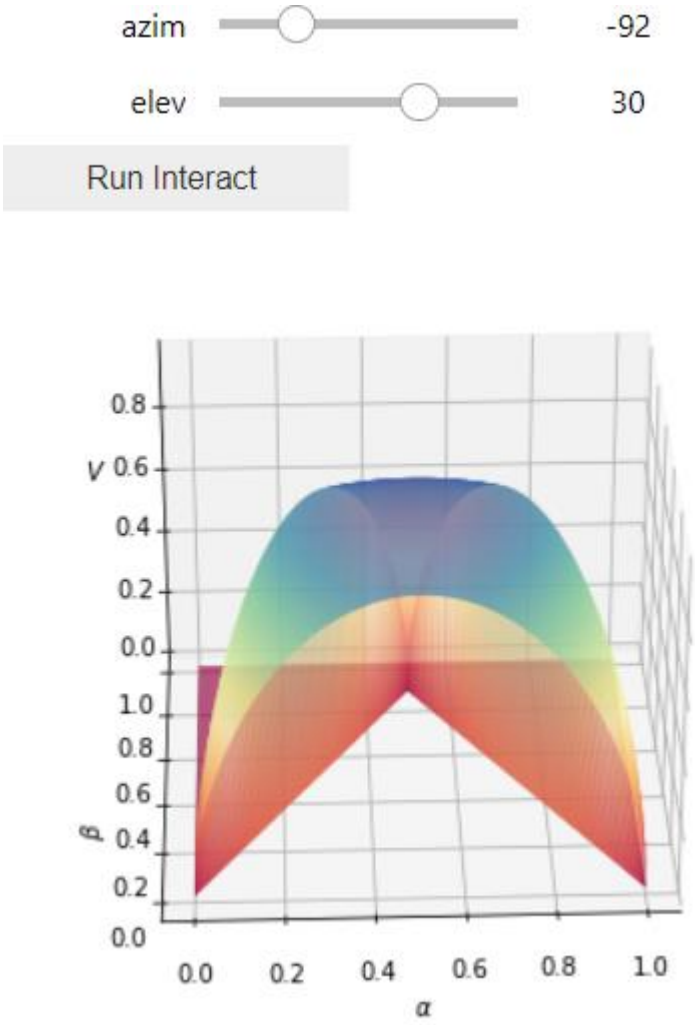


图 3.33. 添加用户界面进行计算实验

将函数的输入参数传递给装饰器。如有必要，就像我们的例子一样，可以显式地设置用户界面元素以指定参数值、它们的边界值和初始值。使用应用程序用户界面的工作包括设置 azim 和 elev 角度并单击 Run Interact 按钮，之后将以新的视角显示曲面。

上面讨论的方法并不能提供漂亮的用户界面，但相当可行。要在 JN 环境中构建完整的图形用户界面，JL 需要参考 ipywidets 库文档[3]。

读者作业：

1. 重现上面描述的计算。
2. 寻找链条悬挂在不同长度的支柱上时的最佳长度值（图 3.11）。有必要最小化链条悬挂的两个点处的力之和。
3. 绘制拉力链长度的变化图。电杆的高度不同——见任务书第 2 条。

4. 将两个、三个、四个不同质量的重物挂在链条上，计算其下垂。
5. 不要设置从左支座开始的货物缩进量，而是设置从左支座到货物的链条长度。
6. 给出链条的弹性模量后，通过解除对链条不可拉伸性的限制，计算其垂度。
7. 通过给出链材料（钢）的热膨胀系数，计算其冬夏季参数。
8. 尝试取积分-用具有基本函数的表达式替换它们。
9. 找到关于 21、85 等问题的解析解。盒子（第三、第四等切割步骤）。
10. 用解析的方法证明，他们的圆形工件制造的第三个锥形容器将是多余的，如果考虑到最大化总体积。
11. 构建三脚架的立体图像，俯视图如图 3.30 所示。
12. 三脚架的俯视图如图 3.30 所示，在顶部有一个凹槽，中心有一个局部极小点。确定可以将多少体积的液体倒入这个凹陷。

参考文献：

1. Очков В. Ф., Попова К., Камалов М. Цепная линия // Физика для школьников. № 3. 2018. С. 24 - 32 (http://twt.mpei.ac.ru/ochkov/fizika_dlya_shkolnikov_2018_03.pdf)
2. Очков В.Ф., Шевяков М.Ю., Чудова Ю.В. Цепная линия: переступить или переехать? // Математика в школе, № 3, 2021 (http://www.twt.mpei.ac.ru/ochkov/Bike-Chain.pdf)
3. Меркин Д.Р. Введение в механику гибкой нити. — М.: Наука, 1980. — 240 с. (https://bookree.org/reader?file=469121&pg=3 или https://dwg.ru/lib/1317)
4. Очков В.Ф., Богомолова Е.П., Иванов Д.А. Физико-математические этюды с Mathcad и Интернет. Издательство Лань. 2016 (http://www.twt.mpei.ac.ru/ochkov/T-2018/PhysMathStudies.pdf)
5. C Y Wang The optimum spanning catenary cable European Journal of Physics, Volume 36, Number 2 (https://iopscience.iop.org/article/10.1088/0143-0807/36/2/028001)
6. Очков В.Ф., Ленер Ф., Чудова Ю. В., Капитонец В. К., Тараканова Д. Ю. Физика vs информатика: веревочный многоугольник с гирьками в статике, кинематике и динамике Или Ньютон vs

Лагранж // Cloud of Science Том 4 № 2. 2017. С. 147-180 (http://www.twt.mpei.ac.ru/ochkov/Polygon.pdf)

7. 1. Очков В. Ф., Калова Яна, Никульчев Е.В. Оптимизированный фрактал или ФМИ // Cloud of Science. 2015. Т. 2. № 4. С. 544-561 (http://www.twt.mpei.ac.ru/ochkov/Opt_Fractal.pdf)
8. 2. Очков В.Ф., Тихонов А.И. Узоры и фракталы на Python // Cloud of Science. 2020. Т. 7. № 4 С. 764-789 (http://www.twt.mpei.ac.ru/ochkov/CoS_28.pdf)
9. 2. Очков В.Ф., Тихонов А.И. Узоры и фракталы на Python // Cloud of Science. 2020. Т. 7. № 4 С. 764-789 (http://www.twt.mpei.ac.ru/ochkov/CoS_28.pdf)
10. 3.Jupyter Widgets.URL：https://ipywidgets.readthedocs.io/en/stable/（日期为2022年10月20日）