Optimization in Maple

Go to Maple Portal Maple Portal for Engineers

The Optimization package in Maple provides a collection of commands for numerically solving optimization problems, which involve in finding the minimum or maximum of an objective function which could be subject to constraints. The package enables you to solve linear programs (LPs), quadratic programs (QPs), nonlinear programs (NLPs), and both linear and nonlinear least-squares problems. The package accepts a wide range of input formats including algebraic form, Matrix form and Maple programming (or operator) notation. More information on the input forms accepted by the Optimization package can be found in the InputForms help page.

This document will use several different examples to illustrate how Maple can solve linear programs, quadratic programs, non-linear programs, and least-square problems.

The commands in the Optimization package can be accessed by loading the package into Maple using the with command.

$with(Optimization)$:

Warning, not a built-in function (`rtable_alias`)

Solving Linear Programs
Solving Non-Linear Programs
Solving Quadratic Programs
Solving Least-Square Problems
Example Worksheet
Applications
See Also

Solving Linear Programs

As mentioned earlier, the Optimization package can handle a wide variety of input formats. This section will show how linear programs can be solved for Algebraic and Matrix Input Notations.

Algebraic Input Notation

This example shows how to enter and solve a linear program in algebraic notation.

Problem Description

Assume a problem where you want to:

Maximize the function $x + y$

Subject to: $x + 2 \cdot y \mathrm{R}\ 2$
$x - y \le 2$
$4 \cdot x - y \mathrm{R}\ 0$
$x \le 4$
$y \le 6.5$

Solution

First, define the objective function:

$$ObjectiveFunctionLP1 := x + y$$

$$ObjectiveFunctionLP1 := x + y \quad (1)$$

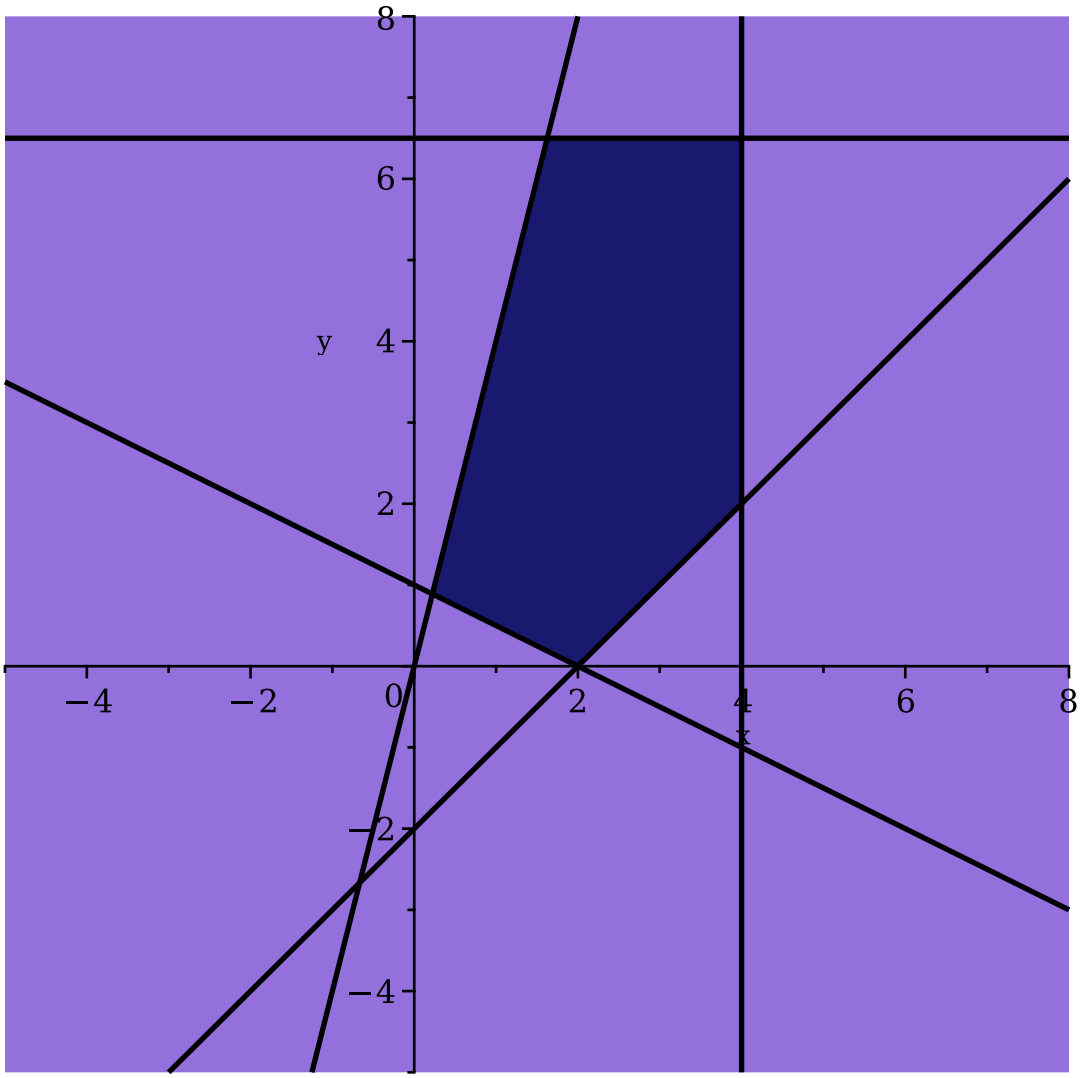
Then define a set containing all the constraint equations:

$$ConstraintEquationsLP1 := \{x + 2 \cdot y \mathrm{R}\ 2, x - y \le 2, 4 \cdot x - y \mathrm{R}\ 0, x \le 4, y \le 6.5\}$$

$$ConstraintEquationsLP1 := \{0 \le 4\,x - y, 2 \le x + 2\,y, x \le 4, y \le 6.5, x - y \le 2\} \quad (2)$$

For a two-dimensional linear program, you can easily graph the feasible region, shown in blue below. The black lines indicate the constraints, and the red lines indicate the contours of the objective function $x + y$.

$$p1 := plots[inequal](ConstraintEquationsLP1, x = -5..8, y = -5..8, thickness = 2, optionsexcluded = (color = \text{"MediumPurple"}), optionsfeasible = (color = \text{"MidnightBlue"}))\ :$$



$p2 := plots[contourplot](ObjectiveFunctionLP1, x = -5..8, y = -5..8, thickness = 2)$:

Error, (in LinearAlgebra:-Multiply) Vector dimension (0) must be the same as the Matrix column dimension (3)

$plots[display](p1, p2)$

Error, (in plots:-display) expecting plot structure but received: p2

Using the Maximize command, find the solution that maximizes the objective function subject to the constraint equations. The first entry returned, 10.5, is the objective value at the optimal solution, while the second entry refers to a point, (4, 6.5), which is a solution point to the optimization problem. The point (4, 6.5) lies at the upper-right vertex of the feasible region.

$Maximize(ObjectiveFunctionLP1, ConstraintEquationsLP1)$

$[10.5000000000000, [\ ]]$ **(3)**

Similarly, the Minimize command can be used to find the minimum solution.

$Minimize(ObjectiveFunctionLP1, ConstraintEquationsLP1)$

$$[1.11111111111111, [\,]] \quad (4)$$

Alternatively, you can use the **LPSolve** command to find the minimum and maximum values of the objective function. LPSolve is more flexible than the **Maximize** and **Minimize** commands in that it can also accept problems in Matrix format. By default, LPSolve minimizes. To maximize, include the option maximize. For more information on the LPSolve command, see LPSolve.

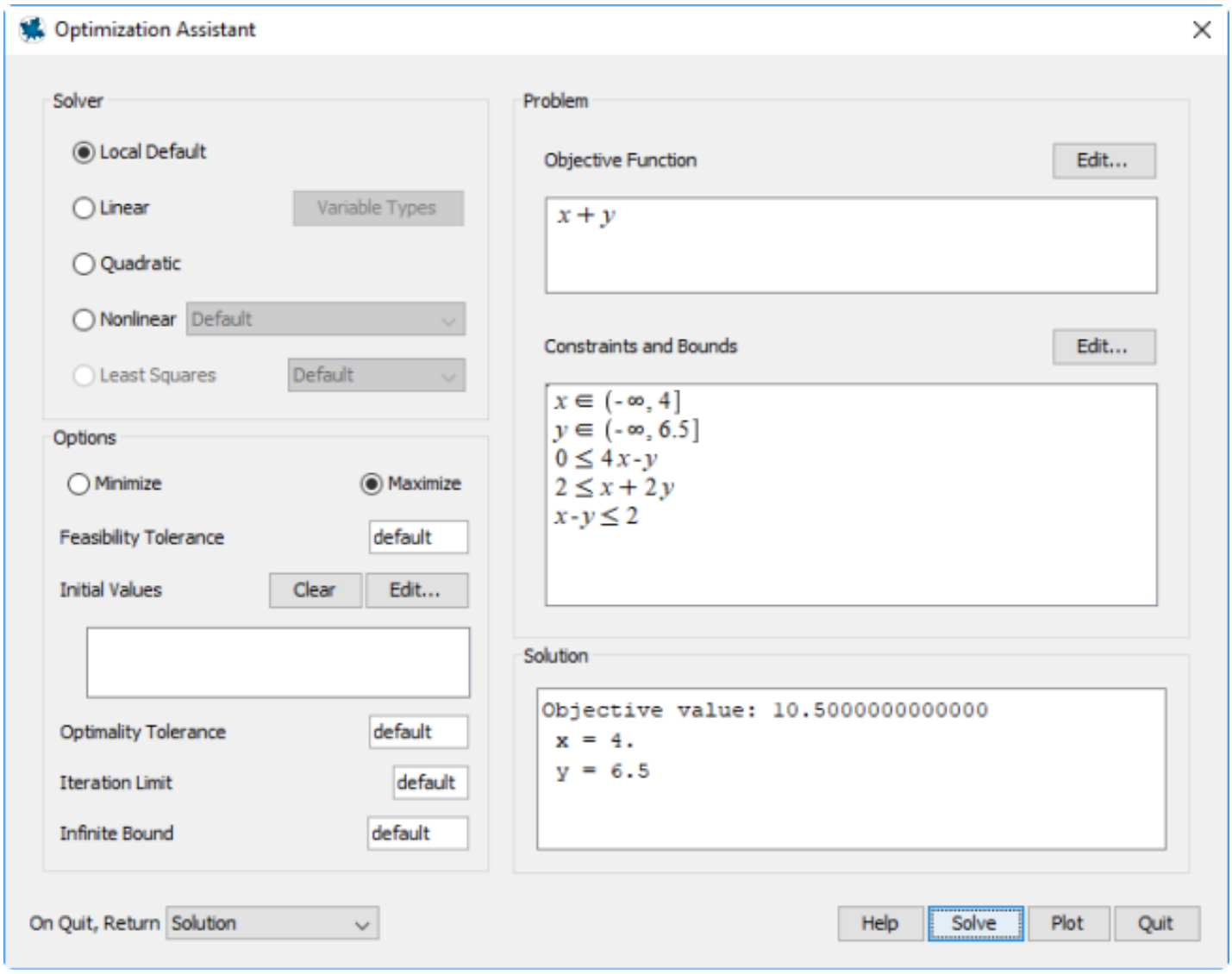
$$LPSolve(ObjectiveFunctionLP1, ConstraintEquationsLP1\ );$$
$$LPSolve(ObjectiveFunctionLP1, ConstraintEquationsLP1, maximize);$$

$$[1.11111111111111, [\,]]$$
$$[10.5000000000000, [\,]] \quad (5)$$

Finally, you can also solve the problem using the interactive Optimization Assistant. The Assistant, seen in the following figure, allows you to enter, edit, and solve (or re-solve) the problem by pointing and clicking, rather than typing commands. It is accessible through the Interactive command or through the Tools menu (**Tools > Assistants**). See Optimization Interactive for information on using the Optimization Assistant.

In the assistant, choose **Minimize** or **Maximize** under Options, and then click **Solve**.

$$MinValue := Interactive(ObjectiveFunctionLP1, ConstraintEquationsLP1)$$
$$MinValue := no\ solution\ available \quad (6)$$
$$MaxValue := Interactive(ObjectiveFunctionLP1, ConstraintEquationsLP1)$$
$$MaxValue := no\ solution\ available \quad (7)$$



Solving the Problem with the Optimization Assistant

Note: The Optimization Assistant only accepts inputs of algebraic form.

Matrix Notation

Problem Description

The Diet Problem is a Linear Program Optimization problem inspired by George Dantzig in the 1940's. The Diet Problem tries to find the cheapest diet that meets the daily nutritional needs given the nutritional values and unit costs of a set of foods.

Let the variable $Cost$ be defined as the cost of 1000 different foods, where $Cost_i$ is the cost per unit weight of food i.

$$Cost := LinearAlgebra[RandomVector](1000, generator = 0..1.0)$$

$$Cost := \begin{bmatrix} 0.866749896999318703 \\ 0.0600188197794759848 \\ 0.443964155018810369 \\ 0.950894415378135238 \\ 0.635661388861376908 \\ 0.505636617571756153 \\ 0.852263890343845643 \\ 0.789073514938958276 \\ 0.814539772900651271 \\ 0.317427863655849629 \\ \text{«} \end{bmatrix} \quad \textbf{(8)}$$

1000 element Vector[column]

Similarly, define the variable $Nutrients$, as a Matrix with dimension 600x1000, where $Nutrients_{i,j}$ is the amount of nutrients i in a unit amount of food j.

$$Nutrients := LinearAlgebra[RandomMatrix](600, 1000, generator = 0..1.0) \quad \textbf{(9)}$$

$Nutrients :=$

$$\begin{bmatrix} 0.690252004749263293 & 0.119368763106342701 & \cdots \\ 0.995993624688443435 & 0.0318037936081185801 & \cdots \\ 0.487363612060525475 & 0.544820537944287864 & \cdots \\ 0.854644185204962104 & 0.844992410967547070 & \cdots \\ 0.860149908324415069 & 0.729909270833515089 & \cdots \\ 0.396795347809815846 & 0.989448062025400477 & \cdots \\ 0.558734511424267999 & 0.274919205132928668 & \cdots \\ 0.816493106302319305 & 0.301092896782379493 & \cdots \\ 0.558864704312516780 & 0.0432740022175199801 & \cdots \\ 0.0953116583646012039 & 0.735743375670887545 & \cdots \\ \text{«} & \text{«} & \end{bmatrix}$$

600 × 1000 Matrix

Lastly, define the variable $DailyRequirements$ to be a vector which specifies a daily requirement of the 600 nutrients, where $DailyRequirements_i$ is the daily requirement of nutrient i

$DailyRequirements := LinearAlgebra[RandomVector](600, generator = 2..20.0)$

$$DailyRequirements := \begin{bmatrix} 19.2033771853635251 \\ 19.9823798534601558 \\ 5.66029616745757913 \\ 16.5125162782306738 \\ 8.79846996533427195 \\ 18.7783614530733729 \\ 15.3137222526290184 \\ 8.27957144508377674 \\ 16.7026099446744922 \\ 16.6353005027060128 \\ \text{«} \end{bmatrix} \quad \textbf{(10)}$$

600 element Vector[column]

Letting vector x be the daily amounts of the foods you choose to buy, the problem can be stated:

Minimize
$Cost^T \cdot x$

Subject to: $Nutrients \cdot x$
R $DailyRequire\backslash ments$
x R 0

Solution

You can solve the linear problem as follows:

$LPSolve(Cost, [Nutrients, DailyRequirements])$ solves min $Cost^T \cdot x$ such that $Nutrients \cdot x \le DailyRequirements$. To tell Maple $Nutrients \cdot x$ R $DailyRequirements$ you must pass $[-Nutrients, -DailyRequirements]$. See LPSolve (Matrix Form) for more information on using the Matrix input for for LPSolve.

$MinCost := LPSolve(Cost, [-Nutrients, -DailyRequirements], assume = nonnegative)$

$$MinCost := \left[0.340161950963048, \begin{bmatrix} 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ 0. \\ \text{«} \end{bmatrix} \right] \quad \textbf{(11)}$$

1000 element Vector[column]

Which elements of the solution are positive? Of the 1000 available foods, it looks like the diet will consist of a select few, which hopefully are palatable.

$$nops(select(x \to x > 0, convert(MinCost[2], list))) = 0$$

Solving Non-Linear Programs

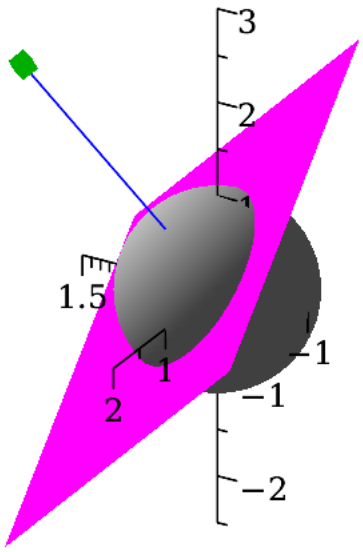
Problem Description

Find the point on the circle formed by the intersection of the unit sphere with the plane $x + y + z = \frac{1}{2}$ that is closest to the point (1, 2, 3).

Solution

This problem can be easily conceptualized by plotting the unit sphere and the equation defining the intersecting plane.

```
p1 := plottools[sphere]( [0, 0, 0], 1, color = "Grey" ) :
p2 := plot3d( 1/2 - u - v, u = -1..1.5, v = -1..1.5, color = "Magenta" ) :
p3 := plottools[line]( [0, 0, 0], [1, 2, 3], color = blue, thickness = 1 ) :
p4 := plots[pointplot3d]( [[1, 2, 3]], color = "Green", symbolsize = 30 ) :
plots[display]([ p1, p2, p3, p4], scaling = constrained, style = patchnogrid, axes
    = normal, orientation = [120, 64], lightmodel = light3 );
```



The objective function is the squared distance between a point (x, y, z) and the point (1, 2, 3).

$ObjectiveFunctionNLP1 := (x-1)^2 + (y-2)^2 + (z-3)^2$

$$ObjectiveFunctionNLP1 := (x-1)^2 + (y-2)^2 + (z-3)^2 \quad \textbf{(12)}$$

The point (x, y, z) is constrained to lie on both the unit sphere and the given plane. Thus, the constraints are:

$ConstraintsEquationsNLP1 := \{ x^2 + y^2 + z^2 = 1,\ x + y + z = 1/2 \}$

$$ConstraintsEquationsNLP1 := \left\{x + y + z = \frac{1}{2},\ x^2 + y^2 + z^2 = 1\right\} \quad \textbf{(13)}$$

You can minimize the objective function subject to the constraints using the NLPSolve command.

$sol := NLPSolve(ObjectiveFunctionNLP1, ConstraintsEquationsNLP1)$

$$sol := [10.2919871984546809, [\ ]] \quad \textbf{(14)}$$

$$[10.2919871984546809, [\ ]] \quad \textbf{(15)}$$

Thus, the minimum distance is 10.29, and the closest point is (-0.51, 0.17, 0.84).

Solving Quadratic Programs

Problem Description

The Markowitz model is an optimization model for balancing the return and risk of a portfolio. The decision variables are the amounts invested in each asset. The objective is to minimize the variance of the portfolio's total return, subject to the following constraints: (1) the expected growth of the portfolio is at least some target level, and (2) the investment should not be more than the capital.

Solution

Let:

$x_1 .. x_n$ be the amounts you buy

c the amount of capital you have

R the random vector of asset returns over some period

r the expected value of R

G the minimum growth you hope to obtain

Q the covariance matrix of R

The objective function is $Var\left(\sum_{i=1}^{n} x_i R_i\right)$, which can be shown to be equal to $x^T Q x$.

If, for example, $n = 4$, you would try to:

Minimize the function $x^T \cdot Q \cdot x$

Subject to:
$$x_1 + x_2 + x_3 + x_4 \le c$$
$$G \le x_1 \cdot r_1 + x_2 \cdot r_2 + x_3 \cdot r_3 + x_4 \cdot r_4$$
$$0 \le x$$

Suppose you have the following data.

$n := 4$:

$X := \langle seq(\, x[i], i = 1 .. n \,) \rangle$:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad (16)$$

$c := 10000$:

$G := 1000$:

$r := \langle .05, -.20, .15, .30 \rangle$:

$$Q := \begin{bmatrix} .08 & -0.05 & -0.05 & -0.05 \\ -0.05 & 0.16 & -0.02 & -0.02 \\ -0.05 & -0.02 & 0.35 & 0.06 \\ -0.05 & -0.02 & 0.06 & 0.35 \end{bmatrix} :$$

This is a quadratic function of X, and so the quadratic programming can be used to solve the problem.

$ObjectiveFunctionQP := expand(LinearAlgebra[Transpose](X).QX)$

Error, (in LinearAlgebra:-Transpose) rtable expected

$ConstraintEquationsQP := \{(x_1 + x_2 + x_3 + x_4 \le c), (x_1 \cdot r[1] + x_2 \cdot r[2] + x_3 \cdot r[3] + x_4 \cdot r[4] \mathrm{R}\ G)\}$

$ConstraintEquationsQP := \{1000 \le 0.05\,x_1 - 0.20\,x_2 + 0.15\,x_3 + 0.30\,x_4, x_1 + x_2 + x_3 + x_4 \le 10000\}$ **(17)**

$QPSolve(\ ObjectiveFunctionQP, ConstraintEquationsQP, assume = nonnegative\ )$

Warning, calling LPSolve as linear objective function was found

$[0., [\ ]]$ **(18)**

Thus, the minimum variance portfolio that earns an expected return of at least 10% is $x_1 = 3452, x_2 = 0, x_3 = 1068, x_4 = 2223$. Asset 2 gets nothing because its expected return is -20% and its covariance with the other assets is not sufficiently negative for it to bring any diversification benefits.

For more information about the QPSolve command, see QPSolve.

Solving Least-Square Problems

Problem Description

Neural networks are machine-learning programs that can learn to approximate functions using a set of sample function values called training data. To generate an output, a neural network applies a smooth function (typically, sigmoidal or hyperbolic tangent) to a linear weighting of the input data. To train the network, one assigns these weights so as to minimize the sum of squared differences between the desired and generated outputs over all data points in the training set. This is an optimization problem that can often be solved by least-squares techniques.

Solution

Assume that you have a sample training data set with five points. The first element of each point is the input, the second is the desired output.

$$x := [[1, 1], [.7, -1], [.1, -1], [.5, 1], [-.8, 1]]$$

$$x := [[1, 1], [0.7, -1], [0.1, -1], [0.5, 1], [-0.8, 1]] \quad (19)$$

Suppose that for a given input value x, the network outputs $\tanh(w_2\, x + w_1)$, where w_1 and w_2 are the weights that are to be set. The residuals are the differences between these outputs and the desired outputs given in the training set:

$$Residual := seq(x[i][2] - \tanh(w[2] \cdot x[i][1] + w[1]), i = 1..5)$$

$$Residual := 1 - \tanh(w_1 + w_2), -1 - \tanh(w_1 + 0.7\, w_2), -1 - \tanh(w_1 + 0.1\, w_2), 1 - \tanh(w_1 + 0.5\, w_2), 1 - \tanh(w_1 - 0.8\, w_2) \quad (20)$$

You can extract elements of the list x using the selection operation. Then, use seq to create a sequence.

The objective function is the sum of squares of the residuals:

$$ObjectiveFunctionLeastSquares := add(Residual[i]^2, i = 1..5)$$

$$ObjectiveFunctionLeastSquares := (1 - \tanh(w_1 + w_2))^2 + (-1 - \tanh(w_1 + 0.7\, w_2))^2 + (-1 - \tanh(w_1 + 0.1\, w_2))^2 + (1 - \tanh(w_1 + 0.5\, w_2))^2 + (1 - \tanh(w_1 - 0.8\, w_2))^2 \quad (21)$$

Now, solve the optimization problem with the LSSolve command. Note that you only have to pass Maple the list of residuals.

no solution available

$$LSSolve([Residual])$$

$$[2.36675848172506, [\,]] \quad (22)$$

The weights needed to minimize the sum of squared differences between the desired and generated outputs for w_1 and w_2 are 0.25123 and 0.174677.

Example Worksheet

The Optimization example worksheet provides additional examples.

Applications

The following are applied optimization examples:
Optimizing the Design of a Helical Spring
Welded Beam Design Optimization

See Also

Optimization, selection, seq