Introduction to Global Optimization

Tibor Csendes

University of Szeged

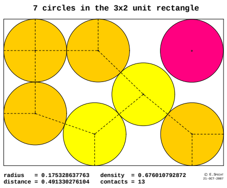
Vienna, October 22-30, 2012

Real life problems

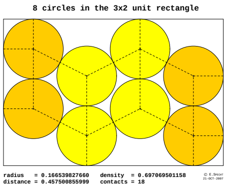
- The case of Hans-Paul Schwefel: the problem was to find a better efficiency positioning of the heating elements of a nuclear power station produced by Siemens. The approximate solution found by an evolutionary method was more than 1% better than the know solution. Although it was not to be cleared whether the found point was even a local minimizer, or how far it was to the optimum, the project partner was satisfied ($> 10^8$ DEM).
- Cutting iron rods of given profils for the building firm (yearly savings at ten millions of HUF): bin packing + storage management, more than 1% improvement against earlier algorithm on real life data by applying a fitting mix of known heuristics.

Real life problems 2

Optimal packing of buckets with printing ink on euro palettes: the aim was not saving place but to ensure that the outlying bucket do not open due to movements on trucks.

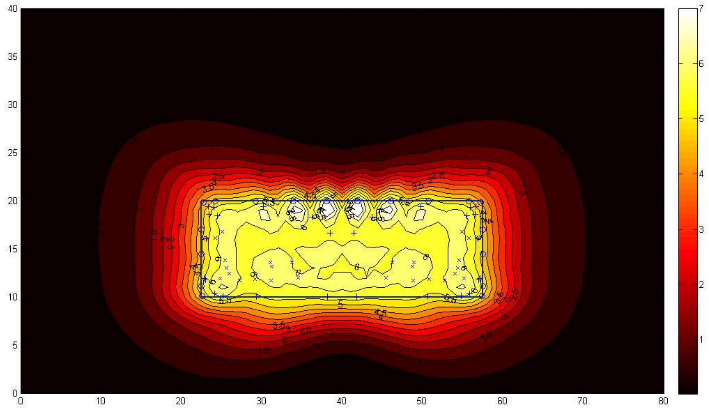


```
radius   = 0.175328637763   density  = 0.676010792872
distance = 0.491330276104   contacts = 13
```



```
radius   = 0.166539827660   density  = 0.697069501158
distance = 0.457500855999   contacts = 18
```

How to design LED based lighting units that meet regulations?



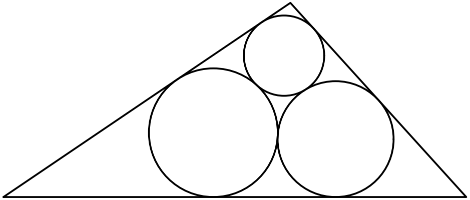
Theoretical questions

- Keppler conjecture: what is the densenst packing of circles on the plane?
- Fekete problem: find n points on a sphere that are the farthest to each other.
- Circle packing: e.g. into the unit square, or the Malfatti circles
- Kissing numbers: what is the maximal number of n-dimensional unit spheres that touch a given unit sphere?
- $\min\left(a^n + b^n - c^n\right)^2 + \sin^2 a\pi + \sin^2 b\pi + \sin^2 c\pi + \sin^2 n\pi$, where a, b, and c nonnegative and $n > 2$.

Malfatti Problem

Gianfrancesco Malfatti (1731–1807) (Chokuen Ajima (1732–1798)):

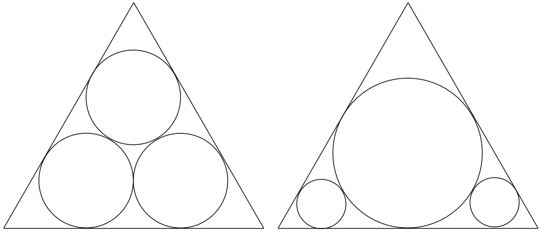
Consider an orthogonal side triangular prism. How can we cut out 3 (possibly of different radii) cylinders such, that the summed volume of the cylinders is maximal?



The Malfatti circles.

The solution of the Malfatti problem

In 1930 H. Lob and H. W. Richmond showed, that there exist a more dense packing of three circles in the equilateral triangle than the Malfatti circles.


Malfatti circles, and a more dense packing for the equilateral triangle.

Optimization problems requiring a reliable solution

- Yes or no questions in practical problems: is there a solution that is more than 1% more profitable than the existing one?
- Critical applications like surgery robot control: there are too many sources of possible failure – once there exist reliable computational techniques, why not use them.

The global optimization problem

Definition

The bound constrained global optimization problem is formulated as

$$\min_{x \in X \subseteq \mathbb{R}^n} f(x).$$

Here X is an n-dimensional interval: $a_i \leq x_i \leq b_i$, where $a_i, b_i \in \mathbb{R}$ are real bounds on the variable x_i, $i = 1, 2, \ldots, n$. $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a two times continuously differentiable objective function.

The global optimization problem is unconstrained when set of feasibility, X is the n-dimensional Euclidean space.

The global optimization problem is constrained, when we add equality and/or inequality constraints: $g_i(x) = 0$ and/or $h_j(x) \leq 0$, where $g_i(x), h_j(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ are real functions, $i = 1, 2, \ldots, m_1$, and $j = 1, 2, \ldots, m_2$ for some positive integers m_1, m_2.

Problem subclasses

For the *unconstrained global optimization problems* $m_1 = m_2 = 0$, and there are no bounds on the variables either.

For the *bound constrained problems* $m_1 = m_2 = 0$.

In case of the *sum-of-squares* problem, the objective function can be written as $f(x) = \sum_{j=1}^{m}(f_j - f_{mod}(j,x))^2$, where m is a positive integer, f_j is real $(j = 1, 2, \ldots, m)$, and $f_{mod}(j,x)$ is the model function.

In the case of the *Mixed integer nonlinear optimization problem (MINLP)*, some of the variables are integer.

On the basis of the properties of the objective function, we have convex, concave, smooth (or two times continuously differentiable), nondifferentiable, quadratic, Lipschitz-continuous etc.

Minima and extremal points

Local minimizer point is x^*, if there exists such a neighbourhood of it $N(x^*)$, that for all points $x \in N(x^*)$ the relation $f(x) \geq f(x^*)$ holds.

Local minimum value of a local minimizer point x^* is the respective objective function value $f(x^*)$.

x^* is a *separated local minimizer point*, if it is not an accumulation point of minimizer points.

A global minimizer point is a local minimizer point, if for all points x of the set of feasibility X the relation $f(x) \geq f(x^*)$ holds.

The *global minimum* is the objective function value at the point x^*: $f(x^*)$.

A *saddle point* is such a point, where although the gradient of the objective function is zero, still in all of its neighbourhood we have smaller and higher objective function values.

Minima and extremal points 2

The region of attraction of a local minimizer point is the set of points of those curves that end in that local minimizer point, and along which the objective function value decreases monotonically.

$f(x)$ is a separable function, if it can be written in the form of $f_1(x) + f_2(x)$, and $f_1(x)$ has a global minimizer point that is a global minimizer of $f_2(x)$ as well (and thus that of $f(x)$ too).

We call a set to be *convex*, if for all its points x_1 and x_2, all points of the line segment (x_1, x_2) are points of the set as well.

$f(x)$ is a convex function, if all its level sets are convex.

The negative of a convex function is a *concave function*.

Classes of global optimization methods

Direct search techniques use only the subroutine that calculates the objective function value at an argument point: mesh search, random walk, Monte Carlo search, adaptive random search, nonlinear simplex method (Nelder-Mead), genetic algorithm, evolutionary methods and simulated annealing.

Gradient based procedures: gradient methods, conjugated gradient method, quasi-Newton algorithm, and the Gauss-Newton method.

The algorithms that utilize the second order derivative information, are the variants of the Newton method.

As a framework for the above procedures, several algorithms serve: multistart method, adaptive start point selection, filter-function approach, Bayesian models, clustering, estimation of the Lipschitz constant, convex underestimation functions, symbolic presolve procedures, branch-and-bound method.

Optimization with tolerances

The reformulated nonlinear optimization problem for handling production tolerances: find such an n-dimensional interval X^* for which

$$\begin{aligned} f(x) &\leq f_\varepsilon := f(x^*) + \varepsilon, & & (1)\\ g_j(x) &\leq 0 & j = 1, 2 \ldots, m. & & (2)\end{aligned}$$

holds for a given $\varepsilon > 0$ and *for all* $x \in X^*$.

We have introduced an interval arithmetic based algorithm for the solution of the above problem, that identifies a *maximal* interval X^* containing a user given seed point. X^* has sides parallel to the coordinate axes, and all points of it are feasible suboptimal points. Maximal interval is meant here in the way that it cannot be increased while keeping the given properties.

Automatic Differentiation

Substitute each appearance of a variable x with the pair $(x, 1)$, and each constant c by the pair $(c, 0)$. Calculate the first element of the resulting pair as for real numbers, while use the derivation rules for determining the second elements of the pairs:

$y = f(x)$	$a \pm x$	$a * x$	a/x	$\sqrt{x}$	$\log(x)$	$\exp(x)$	$\cos(x)$
$f'(x)$	± 1	a	$-y/x$	$0.5/y$	$1/x$	y	$-\sin(x)$

Example: calculate the derivative of $f(x) = (x-1)^2$ at $x = 2$. The derivative function is $f'(x) = 2(x-1)$, and its value 2.

The expression in the brackets is $(2, 1) - (1, 0) = (1, 1)$. Implement square by multiplication: $(1, 1) * (1, 1) = (1, 2)$. The result is then $f(2) = 1$, and $f'(2) = 2$.

Automatic Differentiation – complexity

Computational- (L) and memory complexity (S) of some differentiating problems over the set of operations $\{+, -, *, /, \sqrt{}, \log, \exp, \sin, \cos\}$:

Problem	Algorithm	
	forward	backward
$L(f, \nabla f)$	$\leq 4nL(f)$	$\leq 4L(f)$
$L(f, \nabla f, H)$	$\mathcal{O}(n^2 L(f))$	$\leq (10n + 4)L(f)$
$L(\mathbf{f}, J)$	$\mathcal{O}(nL(\mathbf{f}))$	$\leq (3m + 1)L(\mathbf{f})$
$S(f, \nabla f)$	$\mathcal{O}(S(f))$	$\mathcal{O}(S(f) + L(f))$
$S(f, \nabla f, H)$	$\mathcal{O}(S(f))$	$\mathcal{O}(S(f) + L(f))$
$S(\mathbf{f}, J)$	$\mathcal{O}(S(\mathbf{f}))$	$\mathcal{O}(S(\mathbf{f}) + L(\mathbf{f}))$

Here f is an n-dimensional function, $\mathbf{f}$ is a vector of length m containing n-dimensional functions, ∇f is the gradient of f, H is the Hesse matrix of f, and J is the Jacobi matrix of $\mathbf{f}$.

Penalty functions

One of the usual transformations of constraint optimization problems to unconstrained ones is made by using penalty functions.

Penalty function is an added term, with which we complete the objective function of $\min f(x), g_i(x) \leq 0), i = 1, 2, \ldots, m$ to get the solution of local search methods into the set of feasibility:

$\hat{f}(x) = f(x)$, if $g_i(x) \leq 0$, and $\hat{f}(x) = f(x) + C_1 + C_2 d(x, X)$, where C_1 and C_2 are suitable positive constants, and $d(x, X)$ is the distance of the point x to the set of feasibility X.

C_1 should be larger than the supremum value of $f(x)$ on the set of feasibility.

For interval methods, we apply the Hausdorff metric to describe the distance of a subinterval to the set of feasibility:

$D_H(Y, X) = \sup_{y \in Y} \inf_{x \in X} d(x, y)$, where $d(x, y)$ is an underlying point distance, e.g. the Eucledian metric.

Barrier functions

Barrier functions aim to keep the smoothness of the objective function, and build a strict obstacle for the local search methods, not to cross the border of the set of feasibility.

For example, if the variable x_i should be kept below b_i, then add to the objective function $-\log(b_i - x_i)$ multiplied with a proper small positive constant.

To ensure that the global minimizer point that is inside of the set of feasibility can be found, the multiplicative coefficients of the barrier functions should go to zero as we iterate the local search steps.

Questions and problems

1. How can we transform maximization problems to minimization ones? What happens with the extremum value and with the solution points?
2. What kind of functions $f(x)$, $g_i(x)$, and $h_j(x)$ turn the defined global optimization problem into a linear one?
3. Show such an optimization problem that cannot be formulated as a global optimization problem!
4. Prove that the equality constraints can be formulated by inequality constraints!
5. Why we do not use greater or equal relations?
6. Prove that all sets in $\mathbb{R}^n$ can be given by the mentioned constraints!
7. Give those constraints that allow only a single point as feasible!
8. Show a set of constraints that is redundant (at least one of them can be deleted without changing the set of feasibility)! (Fourier method...)

Questions and problems 2

1. Show such a set of constraints that is not redundant (i.e. no one can be left out without changing the set of feasibility)!
2. How can you use the equality constraints for decreasing the problem dimension? Give an example!
3. Show such a nonlinear optimization problem that has a minimizer point where the first derivative, or the gradient is not zero!
4. Give the definitions of global maximum, and global maximizer point!
5. Determine that funny class of problems, for which all global minimizer point is a global maximizer point as well!
6. Determine the optimum of the line fitting problem on the basis of the zero gradient! (The line fitting problem finds those coefficients of a linear function, with which the summed squared differences in the measured points are minimal.)

Questions and problems 3

1. Under which circumstances has the above problem an infinity of solutions?
2. What happens with the solution of the linear regression problem, when we change the distance function in it?
3. Determine the best fitting circle accordingly! Define first the distance of a point to a circle!
4. Try to fit an ellipsoid! (open)
5. How many minima (maxima) can an unconstrained convex (concave) problem have? Cf. $f(x) = 1/x$, $x \geq 0$.
6. Show a function that is continuously differentiable but not smooth!

Extension of the basic operations for intervals

Set theoretical definition:

$$A \circ B = \{a \circ b \mid a \in A \text{ and } b \in B\}, \; A, B \in \mathbb{I},$$

($\mathbb{I}$ is the set of compact intervals (i, j), where $i, j \in \mathbb{R}$, and $i \leq j$.)

Arithmetic definition:

$$[a, b] + [c, d] = [a + c, b + d],$$

$$[a, b] - [c, d] = [a - d, b - c],$$

$$[a, b] \cdot [c, d] = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)],$$

$$[a, b]/[c, d] = [a, b] \cdot [1/d, 1/c] \;\text{ if }\; 0 \notin [c, d].$$

The two definitions are equivalent.

An example

The inclusion of the function

$$f(x) = x^2 - x$$

obtained for the interval $[0, 1]$ is $[-1, 1]$,

while the range of it is here just $[-0.25, 0.0]$.

This eight times wider enclosure indicates that interval calculations should be useless. Still by using more sophisticated techniques the problem of the too loose enclosure can be overcome – at the cost of higher computational times.

Good bounds can easily be calculated also for the standard functions like sin, log, exp etc. too.

Inclusion functions

A function $F : \mathbb{I}^n \to \mathbb{I}$ is an *inclusion function* of the real function f if $f(y) \in F(Y)$ holds for $\forall Y \in \mathbb{I}^n$ and $\forall y \in Y$, where $\mathbb{I}$ stands for the set of closed real intervals.

The simplest inclusion function is the *naive interval extension*: we substitute each real variable, operation and standard function by its interval equivalent.

More sophisticated inclusion functions are the *central forms* $(F(X) = F(c) + \nabla F(X)(X - c))$, the *slopes*, and the *Taylor models*.

Properties of inclusion functions

F is said to be an *isotone inclusion function* over X if
$\forall Y, Z \in \mathbb{I}(X), \quad Y \subseteq Z$ implies $F(Y) \subseteq F(Z)$.

We say that the inclusion function F has the *zero convergence property*, if $w(F(Z_i)) \to 0$ holds for all the $\{Z_i\}$ interval sequences for which $Z_i \subseteq X$ for all $i = 1, 2, \ldots$ and $w(Z_i) \to 0$.

We call the inclusion function F an *α-convergent inclusion function* over X if $\forall Y \in \mathbb{I}(X) \;\; w(F(Y)) - w(f(Y)) \leq Cw^{\alpha}(Y)$ holds, where α and C are positive constants.

Properties of inclusion functions 2

The naive interval extension is inclusion isotone, zero convergent, and linearly convergent ($\alpha = 1$).

The central forms are inclusion isotone if $c \in X$, always zero convergent, and quadratically convergent.

The intersection of inclusion function values keeps the above properties of the involved inclusion functions.

The algebraic properties of interval arithmetic are not the same as for real arithmetic (e.g. the distributivity does not hold, in general there is no inverse for addition etc.).

Implementation issues

In a floating point environment (most cases) the *outward rounding* is important to have a conservative inclusion that is a must in computer supported mathematical proofs.

Outward rounding means that the bounds of the calculated result intervals are rounded in such a way that all result points are within the given bounds.

In other words, the lower bound is always rounded toward $-\infty$, and the upper bound toward ∞. This can easily be realized applying the four rounding modes of the IEEE 754 standard (available on most programming languages and computers).

Implementation issues 2

Several programming languages and packages support interval arithmetic based inclusion function generation: C-XSC, FORTRAN-XSC, PASCAL-XSC, and PROFIL/BIAS.

Interval packages are available in several numerical languages such as Maple, Mathematica, Matlab. The latter one, Intlab is especially easy to use.

By automatic differentiation the inclusion of first and second derivatives are easy to obtain, as well as multiple precision evaluations to achieve higher accuracy when necessary.

The interval function evaluations require $4 - 35$ times more computation and time.

Interval B&B algorithm

The Moore-Skelboe algorithm will give lower and upper bounds on the minimum value of a nonlinear function over a multidimensional interval X. It locates such a subinterval $X' \subseteq X$ that $F(X')$ contains the global minimum value f^*, and the width of the interval $F(X')$ is less than a preset positive tolerance value ε.

The algorithm will provide a list of subintervals that may still contain global minimizer points, hence only those points will not be covered by the subintervals of the result list, that were rejected by the algorithm, since they surely cannot contain global optimizer points.

The algorithm is described on the next slide.

Interval B&B algorithm 2

1. Set $Y := X$ and $y := \min F(X)$. Initialize the working list $L = ((Y, y))$.
2. Select such a k coordinate, along which $Y = Y_1 \times \cdots \times Y_n$ has a maximal width edge.
3. Bisect Y along the direction of k to obtain boxes V_1 and V_2, for which $Y = V_1 \cup V_2$.
4. Calculate $F(V_1)$ and $F(V_2)$, and let $v_i = \min F(V_i)$ for $i = 1,\ 2$.
5. Delete (Y, y) from the list L.
    1. Monotonicity test: delete pair (V_i, v_i) if $0 \notin F'_j(V_i)$ for a j $(1 \leq j \leq n)$, and $i = 1,\ 2$.
    2. Cut-off test: delete pair (V_i, v_i), if $v_i > \tilde{f}$ (where $\tilde{f}$ is an algorithm parameter, in general the best known upper bound of the global minimum) and $i = 1,\ 2$.
    3. Interval Newton step:

$$V_i := \left(c(V_i) - H^{-1}(V_i)\nabla f(c(V_i))\right) \cap V_i,$$

6. Insert the remaining ones from the pairs (V_1, v_1) and (V_2, v_2) to the working list. If L is empty STOP.
7. Denote that pair of the list L with (Y, y) that has the smallest second member y.
8. If the width of $F(Y)$ is less than ε, then print $F(Y)$ and Y, and STOP.
9. Continue the execution of the algorithm at Step 2.

Interval Newton Step

Consider the one-dimensional problem of finding a zero of a function $f(x) = 0$. We assume that $f'(x)$ is continuous on the interval $[a, b]$, and

$$0 \notin \{f'(x), x \in [a, b]\} \text{ and } f(a)f(b) < 0.$$

When an inclusion X_k of the zero of $f(x)$ is known, then a better X_{k+1} inclusion can be obtained by

$$X_{k+1} := \left(c(X_k) - \frac{f(c(X_k))}{F'(X_k)} \right) \cap X_k.$$

Here $c(X)$ is an inner point of the interval X (e.g. the center of it). Consider the function $f(x) = \sqrt{x} + (x + 1)\cos(x)$ on the interval $[2, 3]$.

Interval Newton Step 2

The obtained iteration sequence together with the widths $w(X_k)$ of the intervals:

k	X_k		$w(X_k)$
1	[2,0,	3,0]	1,0
2	[2,0,	2,3]	0,3
3	[2,05,	2,07]	0,02
4	[2,05903,	2,05906]	0,00003
5	[2,059045253413,	2,059045253417]	0,000000000004

For optimization problems, obviously we look for the zeros of the derivative function. Then the iteration expression is

$$X_{k+1} := \left(c(X_k) - \frac{f'(c(X_k))}{F''(X_k)} \right) \cap X_k.$$

Interval Newton Step 3

When the objective function is multidimensional, then the respective iteration is

$$X_{k+1} := \big(c(X_k) - H^{-1}(X_k)\nabla f(c(X_k))\big) \cap X_k,$$

where $H(X_k)$ is the inclusion of the Hesse matrix of $f(x)$ for the argument interval X_k. Notice that the iteration does not depend directly on the objective function. It is in accordance with the fact that we expect the same iteration sequence for a shifted version $f(x) + c)$ of the original objective function.

Excel Solver for nonlinear optimization

The Excel program is better for routine calculations on large sets of data that may change regularly. Still an Add-in procedure of it, Solver is capable to solve nonlinear optimization problems.

Implementation:

Take Excel version 2007, push the colourful button in the upper left corner. Ask for the Settings and inside of it for the managing of Add-in procedures. Check Solver, push the jump button, and check Solver again in the pop up window. Accept the selection with OK. Now Solver is ready for use. We can find it in the Data tag in the upper right corner, in the analysis box.

An Example

Consider the simple bounded nonlinear optimization problem $\min x^2,\ x \geq 1$. To solve it with Solver, write the value 2 into the cell A1, and fill in cell B1 by entering =A1*A1 into the editing line. We can also show the cell A1 with the mouse, in stead of typing in the name A1. When everything was set properly, then the content of B1 will be 4.

Now select the Data tag, and the Solver. A new dialog window appears. The objective cell should contain our function to be optimized. Hence either write here B1, or show it with the mouse (first the coloured button must be pushed to the right of the respective input row). By pushing ENTER, or the button on the right side we can return to the basic dialog window of Solver.

Solver can minimize, maximize, or set an objective value for a given function. Select minimization by pushing the radio button in the second row.

An Example (continued)

In the third line of the Solver dialog window, select A1 as the changing cell. In case of a multiple variables all of these can be shown by the mouse, or specified accordingly.

Finally set the constraint of our problem. It should come into the lower left subwindow of the dialog window. To specify it, push the button Add to the right of it. In the appearing new dialog box set cell A1 containing our variable x in the left place. Change the relation sign to $\geq$, and write to the right place the number 1. Now if we push OK, then this constraint will be added to the set of constraints, and we return to the basic Solver dialog window. When we have many constraints, use the Add button after specifying one of them.

An Example (continued again)

Now all specifications are set, we are ready to solve the problem. Pushing the Solution button in the upper right corner in the dialog window, Solver reports that a solution has been found. It asks whether the old table can be modified by it. After accepting the modification we can see the correct result: the minimum and the minimizer point are both equal to one. □

Finding a requested value for the objective function is not the subject of our course, but can be useful in practical application. If we obtained a worse than expected solution, we can change the default settings of Solver by opening the Settings dialog box.

The related keywords for a German language Excel 2010 program are: Datei / Optionen / Add-Ins / Solver / Gehe zu... Then Daten / Analyse / Solver / Ziel Festlegen / Lösen / Hinzufügen

Exercises

1. Determine the minimizer point of the Rosenbrock problem: $f(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$ starting from the point (-1, 1)! Try some settings for the solution method!
2. Find the extreme value and the solution point of the function $f(x) = x^6(\sin(1/x) + 2)$ starting from the point 1! Try to find a better approximation.
3. Try to prove the Fermat conjecture by minimizing the function $(a^n + b^n - c^n)^2 + \sin^2(a\pi) + \sin^2(b\pi) + \sin^2(c\pi) + \sin^2(n\pi)$ for $n \geq 2$! Argue that global optimization problems are difficult to solve.
4. How would you use Solver for the solution of multimodal problems (that have several different local minimizers with substantial size regions of attraction)?

Matlab as a development platform

Matlab is a professional program for numerical calculation. Its strength is linear algebra with large and/or sparse matrices. It is interactive, the user types the commands at a prompt, but the resulting algorithms can be turned to compilable high level language program codes. Matlab itself is handling the branch-forecasting of modern processors, hence the pipeline of the processors can be optimally fed. In this sense it is more efficient, than a user written average code can be (cf. BLAS).

Some optimization related Matlab commands (not of a package):

```
>> fzero('-2*(1-x)+200*(x^2-x)*(2*x-1)',0)
>> fminbnd('(1-x)^2+100*(x^2-x)^2',0,2)
>> fminsearch('(1-x(1))^2+100*(x(1)^2-x(2))^2',[0 0])
```

```
>> fminsearch(@Rosenbrock,[0 0])
```

if the subroutine `Rosenbrock.m` contains the respective test function.

Implementing the Intlab package for Matlab

Intlab is a package of S.M. Rump for Matlab that supports both interval calculation with proper outward rounding, and automatic differentiation.

The compressed archive can be downloaded from the web page `http://www.ti3.tu-harburg.de/rump/intlab/`.

Find the source code for your operating system (Windows or Linux), and decompress the archive into a directory *different from the Matlab home*.

Set the path for Matlab in the left subwindow, so that Matlab can recognize where it should find Intlab.

Type the command `>> startintlab` at the prompt.

If everything was fine, you have no error message during the implementation and testing phase, then Intlab reports that it is ready for use. Type `>> infsup(1,2)` and `>> ans+ans`.

Implementing an interval optimization package

Register yourself at the web page

`www.inf.u-szeged.hu/∼csendes/Reg/regform.php`

Decompress the obtained file with Winzip or the 7zip utility of Linux (`>> 7z x filename`). The keyword (`s2L5c4b7aT`) comes in an email one day later.

To use the interval global optimization code, set first the main search path of Matlab (the one highlighted in the top middle position of its window) to the one, where Intlab is. Start Intlab with the command `startintlab`.

Then change the search path to the directory, where the interval optimization code is, and type `>> MainTester`.

Implementing an interval optimization package 2

What happens now, is a test run of the interval global optimization program on the test function Shekel-5. The respective two description files are in the directory TestFunctions.

To test a new test problem, the easiest way is to modify (and rename, respectively) the two files s5.m and s5.bnd. The s5.m file contains the subroutine that calculates the Shekel-5 test function.

The s5.bnd file has the function name in its first line. Then the problem dimension is in the next line, followed by the lower and upper bounds for each variable to be optimized. The last row gives the tolerance value for the stopping condition.

For many other standard global optimization test functions the two description files are in the directory Tests.

An exaple run

```
>> MainTester
Function name:  S5
The set of global minimizers is located in the union of
the following boxes:
c1:  [ 4.00003713662883, 4.00003718945147 ]
     [ 4.00013323800906, 4.00013329348396 ]
     [ 4.00003713910016, 4.00003717168197 ]
     [ 4.00013326916774, 4.00013328566954 ]
The global minimum is enclosed in:
[ -10.15319967905869400, -10.15319967905819700 ]
Statistics:
Iter Feval Geval Heval MLL CPUt(sec)
 16   126    86    7    10    5.05
```

The subroutine calculating the function Shekel-5

```
function y = sh5(x)
m = 5;
a = ones(10,4);
a(1,:)=4.0*a(1,:); a(2,:)=1.0*a(2,:); a(3,:)=8.0*a(3,:); a(4,:)=6.0*a(4,:);
for j = 1:2;
  a(5,2*j-1) = 3.0; a(5,2*j) = 7.0; a(6,2*j-1) = 2.0; a(6,2*j) = 9.0;
  a(7,j) = 5.0; a(7,j+2) = 3.0; a(8,2*j-1) = 8.0; a(8,2*j) = 1.0;
  a(9,2*j-1) = 6.0; a(9,2*j) = 2.0; a(10,2*j-1)= 7.0; a(10,2*j)= 3.6;
end
c(1) = 0.1; c(2) = 0.2; c(3) = 0.2; c(4) = 0.4; c(5) = 0.4;
c(6) = 0.6; c(7) = 0.3; c(8) = 0.7; c(9) = 0.5; c(10)= 0.5;
s = 0.0;
for j = 1:m;
  p = 0.0;
    for i = 1:4
    p = p+(x(i)-a(j,i))^2;
  end
  s = s+1.0/(p+c(j));
end
y = -s;
```

The file `s5.bnd` providing details on the test problem Shekel-5

```
S5
4
0    10
0    10
0    10
0    10
1e-8
```

Here the function name is given (identical with the file name), the dimension of the problem, the lower and upper bounds and the tolerance parameter of the stopping rule.

Exercises for the interval optimization package

It is worth to see what results can be obtained for the following problems:

$\min x^2, x \in [-1, 1]$

$\min x^6 * (\sin(1/x) + 2), x \in [-1, 1]$

$\min 1, x \in [-1, 1]$

$\min(x(1)^2 + x(2)^2 - x(3)^2)^2 + \sin(x(1) * pi)^2 + \sin(x(2) * pi)^2 + \sin(x(3) * pi)^2$
$x \in ([2, 10], [2, 10], [2, 10])$

In the latter case we expect to see some Euclidean triplets. It is worth to play with the tolerance value, and with the starting interval as well.

A clustering multistart stochastic global optimization method: GLOBAL

It is a multistart procedure, that applies local search methods for finding local minimizer points.

Two local search algorithms can be selected: the first is UNIRANDI, a direct search technique. It is a random walk method, that does not assume differentiability of the objective function, and requires only a subroutine for the calculation of the objective function value.

The other method is BFGS, that is readily available in Matlab (even without the optimization packages). This is assumes a smooth objective function, although requires again only a subroutine for the calculation of the objective function value.

The framework multistart algorithm assumes that the relative size of the region of attraction of the global minimizer point(s) is not negligible, i.e. say larger than 0.00001.

The algorithm

Step 1: Draw N points with uniform distribution in X, and add them to the current cumulative sample C. Construct the transformed sample T by taking the γ percent of the points in C with the lowest function value.

Step 2: Apply the clustering procedure to T one by one. If all points of T can be assigned to an existing cluster, go to Step 4

Step 3: Apply the local search procedure to the points in T not yet clustered. Repeat Step 3 until every point has been assigned to a cluster.

Step 4: If a new local minimizer has been found, go to Step 1.

Step 5: Determine the smallest local minimum value found, and stop.

Download GLOBAL for Matlab

The GLOBAL algorithm for Matlab is in the same package, that contains the interval global optimization program.

After decompressing the archive, we can find the code in the directory named `Global_Matlab`.

The equivalent C and FORTRAN language implementations are in the directory `Global_C_Fortran`.

To use the Matlab version, it is enough to add the directory `Global_Matlab` to the search path of Matlab.

How to run GLOBAL inside Matlab

The main function is GLOBAL.m which can be started by the sequence:

```
[X0,F0,NC,NFE] = GLOBAL(FUN, LB, UB, OPTS);
```

The function can be called without the `OPTS` parameter. In this case the default parameters will be used:

```
[X0,F0,NC,NFE] = GLOBAL(FUN, LB, UB);
```

The input parameters are the following:

- `FUN`: Function handler to a function that we want to minimize
- `LB`: Lower bounds on the variables given in the form of a vector
- `UB`: Upper bounds on the variables in the form of a vector
- `OPTS`: An optional MATLAB structure argument to customize the options for the program

Control parameters of GLOBAL

- `OPTS.N100`: Number of sample points to be drawn uniformly in one cycle (`20<=N100<=100000`), the default value is `10*nvars`
- `OPTS.NG0`: Number of best points selected from the actual sample (`1<=NG0<=20`), default is min(`2*nvars,20`)
- `OPTS.NSIG`: Convergence criterion, the number of significant digits in the result (suggested value = 6,7,8),
  default is 6
- `OPTS.MAXFN`: Maximum number of function evaluations for local search, the default is 1000
- `OPTS.MAXNLM`: Maximum number of local minima (clusters), the default is 20
- `OPTS.METHOD`: Local method used (`'bfgs' or 'unirandi'`), the default is `'unirandi'`
- `OPTS.DISPLAY`: Display intermediary or final results (`'on','final','off'`) the default is `'final'`

A sample run of GLOBAL

```
>> LB =[-2; -2];
>> UB =[ 2; 2];
>> OPTS.N100   = 50;
>> OPTS.NG0    = 2;
>> OPTS.NSIG   = 6;
>> OPTS.METHOD = 'bfgs';
>> FUN = @ros2;
>> [X0,F0,NC,NFE] = GLOBAL(FUN, LB, UB, OPTS);
```

where `ros2` is the MATLAB function

```
function y = ros2(x)
    y = 100*(x(1).^2-x(2)).^2+(x(1)-1).^2;
end
```

A typical result of GLOBAL for the Rosenbrock test problem

```
NORMAL TERMINATION AFTER 307 FUNCTION EVALUATIONS
LOCAL MINIMUM FOUND: 1
F0 =
  1.927661340730457e-011
X0 =
  0.999995637462118
  0.999991225458257
GLOBAL MINIMUM VALUE: 0.000000000019277
GLOBAL MINIMUM:
  0.999995637462118
  0.999991225458257
```

Exercises

Check first what GLOBAL can achieve on the test problem Shekel-5.

It is worth again to see what results can be obtained for the following problems:

$\min x^6 * (\sin(1/x) + 2), x \in [-1, 1]$

Notice that UNIRANDI is more effective for this problem than BFGS due to its robust behaiour, and its nondifferentiable model. Check starting intervals zooming in to the origin.

$\min(x(1)^2 + x(2)^2 - x(3)^2)^2 + \sin(x(1) * pi) + \sin(x(2) * pi) + \sin(x(3) * pi)$
$x \in ([2, 10], [2, 10], [2, 10])$

In the latter case we expect to see some Euclidean triplets in the interval $[3, 20]^4$. It is worth to play with the tolerance value, with the starting interval, and also with the two local search methods as well.

The motivating problem

Consider the parameter estimation problem given by the sum-of-squares form objective function:

$$F(R_{aw}, I_{aw}, B, \tau) = \left[\frac{1}{m}\sum_{i=1}^{m}\left|Z_L(\omega_i) - Z_L'(\omega_i)\right|^2\right]^{1/2},$$

where $Z_L(\omega_i) \in \mathbb{C}$ is the measured impedance value, $Z_L'(\omega_i)$ is the modeled impedance at frequencies ω_i for $i = 1, 2, \ldots, m$ and R_{aw}, I_{aw}, B, and τ are model parameters.
The original nonlinear model function is based on obvious physical parameters:

$$Z_L'(\omega) = R_{aw} + \frac{B\pi}{4.6\omega} - \imath\left(I_{aw}\omega + \frac{B\log(\gamma\tau\omega)}{\omega}\right),$$

where $\gamma = 10^{1/4}$ and $\imath$ is the imaginary unit.

The motivating problem 2

The original nonlinear model function is based on obvious physical parameters:

$$Z_L'(\omega) = R_{aw} + \frac{B\pi}{4.6\omega} - \imath\left(I_{aw}\omega + \frac{B\log(\gamma\tau\omega)}{\omega}\right),$$

where $\gamma = 10^{1/4}$ and $\imath$ is the imaginary unit.

The symbolic algorithm was motivated by the existence of a simplified and equivalent model function, that is linear in the model parameters:

$$Z_L'(\omega) = R_{aw} + \frac{B\pi}{4.6\omega} - \imath\left(I_{aw}\omega + \frac{A + 0.25B + B\log(\omega)}{\omega}\right).$$

Some theoretical results

Theorem

If $h(x)$ is smooth and strictly monotonic in x_i, then the corresponding transformation simplifies the function in the sense that each occurrence of $h(x)$ in the expression of $f(x)$ is padded by a variable in the transformed function $g(y)$, while every local minimizer (or maximizer) point of $f(x)$ is transformed to a local minimizer (maximizer) point of the function $g(y)$.

Theorem

If $h(x)$ is smooth, strictly monotonic as a function of x_i, and its range is equal to $\mathbb{R}$, then for every local minimizer (or maximizer) point y^* of the transformed function $g(y)$ there exists an x^* such that y^* is the transform of x^*, and x^* is a local minimizer (maximizer) point of $f(x)$.

More theoretical results

Assertion

If a variable x_i appears everywhere in the expression of a smooth function $f(x)$ in a term $h(x)$, then the partial derivative $\partial f(x)/\partial x_i$ can be written in the form $(\partial h(x)/\partial x_i)\,p(x)$, where $p(x)$ is continuously differentiable.

Assertion

If the variables x_i and x_j appear everywhere in the expression of a smooth function $f(x)$ in a term $h(x)$, then the partial derivatives $\partial f(x)/\partial x_i$ and $\partial f(x)/\partial x_j$ can be factorized in the forms $(\partial h(x)/\partial x_i)\,p(x)$ and $(\partial h(x)/\partial x_j)\,q(x)$, respectively, and $p(x) = q(x)$.

Algorithm

1. compute the gradient of the original function,
2. factorize the partial derivatives,
3. determine the substitutable subexpressions and substitute them:
    1. if factorization was successful, then explore the subexpressions that can be obtained by integration of the factors,
    2. if factorization was not possible, then explore the subexpressions that are linear in the related variables,
4. solve the simplified problem if possible, and give the solution of the original problem by transformation, and
5. verify the obtained results.

A successful example

The objective function of the Rosenbrock problem is:

$$f(x) = 100\left(x_1^2 - x_2\right)^2 + (1 - x_1)^2 .$$

We run the simplifier algorithm with the procedure call:

$$\text{symbsimp([x2, x1]}, 100 * (x1^2 - x2)^2 + (1 - x1)^2);$$

In the first step, the algorithm determines the partial differentials:

$$\text{dx}(1) = -200x_1^2 + 200x_2$$

$$\text{dx}(2) = 400(x_1^2 - x_2)x_1 - 2 + 2x_1$$

Then the factorized forms of the partial derivatives are computed:

$$\text{factor}(\text{dx}(1)) = -200x_1^2 + 200x_2,$$

$$\text{factor}(\text{dx}(2)) = 400x_1^3 - 400x_1x_2 - 2 + 2x_1.$$

A successful example 2

The list of the subexpressions of f ordered by the complexity in x_2 is the following:

$$\{100(x_1^2 - x_2)^2, (x_1^2 - x_2)^2, x_1^2 - x_2, -x_2, x_2, (1 - x_1)^2, x_1^2, 100, 2, -1\}.$$

The transformed function at this point of the algorithm is $g = 100y_1^2 + (1 - x_1)^2$.
Now compute again the partial derivatives and its factorization:

$$\text{factor}(\text{dx}(1)) = \text{dx}(1) = 200y_1,$$

$$\text{factor}(\text{dx}(2)) = \text{dx}(2) = -2 + 2x_1.$$

The final simplified function, what our automatic simplifier method produced is

$$g = 100y_1^2 + y_2^2.$$

Notions on the quality of the results

A: simplifying transformations are possible according to the presented theory,

B: simplifying transformations are possible with the extension of the presented theory,

C: some useful transformations could be possible with the extension of the presented theory, but they not necessarily simplify the problem at all points (e.g. since they increase the dimensionality),

D: we do not expect any useful transformation.

Our program produced ...

1: proper substitutions,

2: no substitutions,

3: incorrect substitutions.

The mistake is due to the incomplete ...

a: algebraic substitution,

b: range calculation.

Results for the problems in the original article

ID	Function f	Function g	Substitutions	Result type
Cos	$\cos(e^{x_1}+x_2)+\cos(x_2)$	$\cos(y_1)+\cos(y_2)$	$y_1=e^{x_1}+x_2, y_2=x_2$	A1
ParamEst1	$[\frac{1}{3}\sum_{i=1}^{3}\lvert Z_L(\omega_i)-Z_L'(\omega_i)\rvert^2]^{1/2}$	g_1	$y_1=\omega, y_2=-R_{aw}, y_3=I_{aw}, y_4=B, y_5=\tau$	A2a
ParamEst2	$[\frac{1}{3}\sum_{i=1}^{3}\lvert Z_L(\omega_i)-Z_L''(\omega_i)\rvert^2]^{1/2}$	$.5773502693y_5^{1/2}$	$y_1=\omega, y_2=-R_{aw}, y_3=I_{aw}, y_4=B, y_5$	A3ab
ParamEst3	$[\frac{1}{3}\sum_{i=1}^{3}\lvert Z_L(\omega_i)-Z_L'''(\omega_i)\rvert^2]^{1/2}$	$.5773502693y_5^{1/2}$	$y_1=\omega, y_2=-R_{aw}, y_3=I_{aw}, y_4=B, y_5$	A3b
Otis	$(\lvert Z_L(s)-Z_m(s)\rvert^2)^{1/2}$	$(\lvert -Z_L[1]+1.\frac{y_2}{y_4}\rvert^2)^{1/2}$	$y_1=s, y_2=I_C(R_1+R_2)C_1C_2y_1^3+(I_C(C_1+C_2)+(R_C(R_1+R_2)+R_1R_2)C_1C_2)y_1^2+(R_C(C_1+C_2)+R_1C_1+R_2C_2)y_1+1, y_4=(R_1+R_2)C_1C_2y_1^2+(C_1+C_2)y_1$	B3

Results for standard global optimization problems

ID	Function g	Substitutions	Result type
Rosenbrock	$100y_2^2 + (1 - y_1)^2$	$y_1 = x_1, y_2 = y_1^2 - x_2$	A1
Shekel-5	memory error	none	D2
Hartman-3	none	none	D2
Hartman-6	none	none	D2
Goldstein-Prize	none	none	D2
RCOS	$y_2^2 + 10(1 - 1/8/\pi) * \cos(y_1) + 10$	$y_1 = x_1, y_2 = 5/\pi y_1 - 1.275000000y_1^2/\pi^2 + x_2 - 6$	A1
Six-Hump-Camel-Back	none	none	D2

Other often used global optimization test functions

ID	Function g	Substitutions	Result type
Levy-1	none	none	D2
Levy-2	none	none	D2
Levy-3	none	none	D2
Booth	none	none	C2
Beale	none	none	C2
Powell	$(y_1+10y_2)^2+5(y_3+y_4)^2+(y_2-2y_3)^4+10(y_1+y_4)^4$	$y_1 = x_1, y_2 = x_2, y_3 = x_3, y_4 = -x_4$	D2
Matyas	none	none	D2
Schwefel $(n = 2)$	none	none	C2
Schwefel-227	$y_2^2 + .25y_1$	$y_1 = x_1, y_2 = y_1^2+x_2^2-2y_1$	A1
Schwefel-31 $(n = 5)$	none	none	D2
Schwefel32 $(n = 2)$	none	none	D2
Rastrigin $(n = 2)$	none	none	C2
Ratz-4	none	none	C2
Easom	none	none	D2
Griewank-5	none	none	D2

Questions of an earlier exam test

1. Give such an optimization problem that has a non-separated local minimizer point, that is not a global minimizer, but still a local maximizer point! 4 points
2. Show such a two dimensional optimization problem, for which the size of the region of attraction of the global minimizer point is one quarter of that of the search domain. 4 points
3. Is it true that there exists such an optimization procedure that uses only the subroutine calculating the objective function value, and which is capable to determine the solution of all nonlinear optimization problems? Argue why. 4 points
4. Which learned global optimization method you would apply for the solution of such a minimization problem, that has only simple bounds for the variables, only the subroutines calculating the objective function and its gradient are available, and the objective function is smooth? 4 points

Questions of an earlier exam test 2

5. Describe in detail one of the learned global optimization procedures, give that problem class, for the solution of which it is well suited, and provide its advantages and drawbacks! 9 points
6. Determine the inclusion function of $f(x) = x\sin(2\pi x) + 2$ with natural interval extension, and calculate its value at the interval $X = [0, 1]$! 3 points
7. Calculate the derivative value of $f(x) = (x - 1)(x - 2)$ at the point 3 with automatic differentiation! 3 points
8. Find the simplifying transformation with the technique learned for the objective function $f(x_1, x_2) = 10^{(2x_1+3x_2)^2}$, and discuss the result! 3 points

Evaluation: 27 points and above: excellent, 24 points: good, 21 points: satisfactory, 17 points: fair.

Literature

Bazaraa, M.S., H.D. Sherali, and C.M. Shetty: Nonlinear Programming — Theory and Algorithms, Wiley, 1993.

Bomze, I.M., T. Csendes, R. Horst, and P.M. Pardalos (eds.): Developments in Global Optimization. Kluwer, Dordrecht, 1997.

Csendes, T.: Introduction to Global Optimization. Lecture notes in Hungarian, in preparation, `www.inf.u-szeged.hu/∼csendes/go.pdf`. English language slides at `...goslides.pdf`

Dixon, L.C.W., G.P. Szegő (eds.): Towards Global Optimisation. North-Holland, Amsterdam, 1974.

Dixon, L.C.W., G.P. Szegő (eds.): Towards Global Optimisation 2. North-Holland, Amsterdam, 1978.

Literature 2

Gill, P. E., W. Murray, M.H. Wright: Practical Optimization. Academic Press, London, 1981.

Hansen, E.: Global Optimization Using Interval Analysis. Marcel Dekker, New York, 1992.

Horst, R., P.M. Pardalos, and N.V. Thoai: Introduction to Global Optimization, Kluwer, Dordrecht, 1995.

R. Baker Kearfott: Rigorous Global search: Continuous Problems. Kluwer, Dordrecht, 1996

The site of Arnold Neumaier on global optimization:

`http://solon.cma.univie.ac.at/glopt.html`

Literature 3

Pintér, J.D.: Global Optimization in Action, Kluwer, Dordrecht, 1996.

Ratschek, H., J. Rokne: Computer Methods for the Range of Functions. Ellis Horwood, Chichester, 1984.

Ratschek, H., J. Rokne: New Computer Methods for Global Optimization. Ellis Horwood, Chichester, 1988.

Törn, A., A. Žilinskas: Global Optimization. (Lecture Notes in Computer Science No. 350, G. Goos and J. Hartmanis, Eds.) Springer, Berlin, 1989.