Applied Mathematics, 2010, 1, 520-528
doi:10.4236/am.2010.16069 Published Online December 2010 (http://www.SciRP.org/journal/am)

Scanning and Selection Methods Using Solution Boxes of Inequality

Ferenc Kálovics
Analysis Department, University of Miskolc, Miskolc, Hungary
E-mail: *matkf@uni-miskolc.hu*
Received September 10, 2010; *revised October* 18, 2010; *accepted October* 22, 2010

Abstract

Numerical methods often reduce solving a complicated problem to a set of elementary problems. In some previous papers, the author reduced the finding of solution boxes of a system of inequalities, the computation of integral value with error bound, the approximation of global maxima to computing solution boxes of one inequality. This paper contains new and improved methods for application of solution boxes of an inequality, furthermore the computational aspects are discussed in detail.

Keywords: Inequality, Solution Box, Scanning of Set

1. Introduction

The paper [1] gives a complete description and code of a process which is able to compute solution boxes of an inequality automatically (using only the structure of the appropriate expression). This means the following. Let $g: D \subset R^m \to R$ be a continuous multivariate real function, where $D = \left((\underline{x}_1, \overline{x}_1), (\underline{x}_2, \overline{x}_2), \cdots, (\underline{x}_m, \overline{x}_m)\right)$ is an open box. Define the box $B[g,c,\alpha] \subset D$, where $c \in D, \alpha \in R$, as an open box around c, in which the relation is the same as between $g(c)$ and α (it is supposed that $g(c) \neq \alpha$). Thus, if $g(c) < \alpha$, then $g(x) < \alpha$ for all $x = (x_1, \cdots, x_m) \in B[g,c,\alpha]$, if $g(c) > \alpha$, then $g(x) > \alpha$ for all $x = (x_1, \cdots, x_m) \in B[g,c,\alpha]$. The C++ function segment void solbox (double D[][3], double G[][4], double c[], double alp, int m, int nt) of [1] can compute a box $B[g,c,\alpha]$ if the continuous multivariate real function $g: D \subset R^m \to R$ is built of the well-known (univariate real) elementary functions by the ordinary function operations, and the expression $g(x)$ is given in so-called triple form (G). This numerically coded form G is easy to learn, but also [1] gives a Maple code for its preparation. The parameter list of the segment is D[][3] $\leftrightarrow D$, G[][4] $\leftrightarrow G$, c[] $\leftrightarrow c$, alp $\leftrightarrow \alpha$, m $\leftrightarrow m$, nt $\leftrightarrow$ number of triples in G and the output parameter is $B[g,c,\alpha]$. The five numerical methods defined in the following four sections are based on automatic computation of $B[g,c,\alpha]$. Here, let us emphasize two facts about solution boxes. 1) If $g(c) < \alpha$, then $x \in B(g,c,\alpha)$ implies $-\infty < g(x) < \alpha$. Consequently, the box $B(g,c,\alpha)$ of domain D is assigned to the interval $(-\infty, \alpha)$ of function values. Similarly, if $g(c) > \alpha$, then the box $B(g,c,\alpha)$ of domain D is assigned to the interval (α, ∞) of function values. The so-called interval extension functions used in interval methods (see e.g. in [2]) are inverse type functions, they assign intervals of function values to boxes of domain. The handling and application of these two tools require a highly different mathematical and computational background. 2) The box $B(g,c,\alpha)$ is not a symmetrical box around c. Often it has a large volume, although $g(c) < \alpha$ or $g(c) > \alpha$ is only just satisfied. At the end of this section, some properties of our methods are mentioned. The notations, names, definitions and discussions (similarly to [1]) are simpler and clearer than they were in the former papers of the author. Each of our five methods has both scanning and selection features, with the names showing the more characteristic feature. The methods for computation of area and volume, for computation of integral values and for finding global maxima can give an error bound to the solution. The author is not aware of tools aside from solution boxes of inequality for such a demanding handling of these problems. The methods for finding a solution of a system of equations and for finding of global minima cannot give error bounds, they are only reliable methods (which can be an important feature in case of practical problems). The computational aspects of our methods are discussed in detail in an appendix (the last section).

2. A Scanning Method for Area and Volume

Let a section set S be given by the system of inequalities

$$f_i(x_1, x_2, \cdots, x_m) \geq 0, i = 1, 2, \cdots, n, m \geq 2, n \geq 1,$$

where the multivariate real functions $f_1, f_2, \cdots, f_n$ are continuous on the closed box I and are built from the well-known univariate real elementary functions. Our aim is to give a good approximation value with guaranted error bound for the area (the volume) of the set S. The method is based on the following four principles. 1) If the box I contains the set S, then the scanning of S gives an approximation of the volume of S and the scanning of the complementary set $I - S$ also facilitates the computation of an error bound. 2) If $B[f_1, c, 0]$ is a solution box to the inequality $f_1(x) \geq 0$ and $B[f_2, c, 0]$ is a solution box to the inequality $f_2(x) \geq 0$, then the box $B[f_1, c, 0] \cap B[f_2, c, 0]$ is a solution box to the system of the two inequalities. 3) If U and T are m-dimensional boxes, then the set $U - T$ can be divided into (at most) $2m$ boxes easily. 4) The too small boxes (the volume is too small) are filtered by the simple condition $vol(B) > \kappa$. Naturally, the value κ has a strong influence on the available error bound. The algorithmic description of the method is as follows.

(a) Call (b)-(d). Let $vol = vol + eps/2$, $eps = eps/2$. Print vol, eps and exb . Stop.

(b) Define the first element of an interval (box) sequence $\{I_k\}$ by $I_1 = I$. Let $nob = 1$, $exb = 0$, $vol = 0$, $eps = vol(I)$, where nob, exb, vol, eps denote the number of boxes in the sequence, the number of the boxes examined, the approximating value of $vol(S)$, and the error bound, respectively.

(c) Let $exb = exb + 1$. Compute the first i^* where $f_{i^*}(c) = \min f_i(c)$ if $1 \leq i \leq n$ and c is the centre of I_{nob} .

(c1) If $f_{i^*}(c) < 0$, then compute the box $B = B(f_{i^*}, c, 0) \subset \overline{S} = I - S$. (The 'worst inequality' is used here.) Let $eps = eps - vol(B)$.

(c2) If $f_{i^*}(c) \geq 0$, then $B := I_{nob}$ and $B := B \cap B(f_i, c, 0)$, $i = 1, \cdots, n$. Let $vol = vol + vol(B)$, $eps = eps - vol(B)$.

(d) Divide the set $I_{nob} - B$ into nb boxes (if the set is empty, then $nb := 0$). Filter the 'unimportant' (too small) boxes by the condition $vol\,(\text{box}) > \kappa$, where κ is a (small) given value. Place the $nb^* \leq nb$ new boxes into the box sequence $\{I_k\}$ as nob th, $(nob + 1)$ th, $\cdots$, $(nob + nb^* - 1)$ th elements and let $nob = nob + nb^* - 1$. If $nob > 0$, then go to (c). If $nob = 0$, then go to the calling point.

The C++ program uses the above 'reminding names' and $\{I_k\} \leftrightarrow Ise$, $\{c_k\} \leftrightarrow Ice$, $\kappa \leftrightarrow kap$. This algorithm does not appear in other papers of the author. Now solve the problem described by

$$16 - x_1^2 - 4x_2^2 \geq 0, x_1^2 + x_2^2 - 4 \geq 0, I = ([-5, 5], [-5, 5])$$

and illustrated in the **Figure 1**.

The exact area of 'the double moon' is $4 \cdot 2 \cdot \pi - 2^2 \cdot \pi = 4\pi \approx 12.5664$. For $\kappa = 10^{-4}$, $\kappa = 10^{-5}$, $\kappa = 10^{-6}$ the area, the error bound, the real error, the number of boxes examined and the running time (with our Visual C++ version 6.0 code on a PC of two 2.2 GHz processors) are

12.5679, 0.0902, 0.0015, 4131, 0.03 sec;

12.5671, 0.0283, 0.0007, 13051, 0.1 sec;

12.5664, 0.0089, 0.0000, 41359, 0.3 sec,

respectively. The program scans 'the double moon' S by solution boxes of an inequality system and it scans the complementary set $I - S$ by solution boxes of 'wrong inequalities'.

3. A scanning method for integrals

Let the definite integral

$$\int \cdots \int_V f(x_1, x_2, \cdots, x_{m-1})\, dx_1 dx_2 \cdots dx_{m-1}$$

be given, where the $m - 1$ dimensional point set V is described by the system of inequalities

$$f_i(x_1, x_2, \cdots, x_{m-1}) \geq 0, i = 1, 2, \cdots, n - 1, m \geq 2, n \geq 1,$$

the multivariate real functions f, f_1, $f_2, \cdots, f_{n-1}$ are continuous on the closed box $D \supset V$ and are built from the well-known univariate real elementary functions. Let us assume that we know (rough) lower and upper bounds $\underline{x}_m \leq 0$, $\overline{x}_m \geq 0$ so that

$$\underline{x}_m \leq f(x_1, x_2, \cdots, x_{m-1}) \leq \overline{x}_m, \forall (x_1, x_2, \cdots, x_{m-1}) \in D.$$

Our aim is to give a good approximation value with guaranted error bound for the integral value. The method is based on the following five principles. 1) The computation of the integral value is equivalent to the computation of the volumes of the solution sets of the two systems of inequalities (consider the geometrical meaning of simple and double integrals, furthermore the definition of definite integrals)

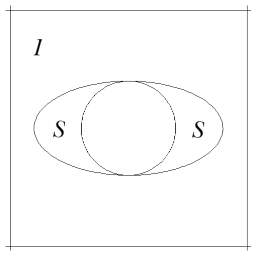


Figure 1. Section set S.

$$\left.\begin{array}{l} f_i\left(x_1,x_2,\cdots,x_{m-1}\right)\ge 0,\, i=1,2,\cdots,n-1 \\ f\left(x_1,x_2,\cdots,x_{m-1}\right)-x_m\ge 0, \end{array}\right\},$$

where

$$\left(x_1,\cdots,x_m\right)\in I=D\times\left[0,\overline{x}_m\right]=\left(\left[\underline{x}_1,\overline{x}_1\right],\cdots,\left[\underline{x}_{m-1},\overline{x}_{m-1}\right],\left[0,\overline{x}_m\right]\right),$$

and

$$\left.\begin{array}{l} f_i\left(x_1,x_2,\cdots,x_{m-1}\right)\ge 0,\quad i=1,2,\cdots,n-1 \\ -f\left(x_1,x_2,\cdots,x_{m-1}\right)-x_m\ge 0, \end{array}\right\},$$

where

$$\left(x_1,\cdots,x_m\right)\in I=D\times\left[0,-\underline{x}_m\right]=\left(\left[\underline{x}_1,\overline{x}_1\right],\cdots,\left[\underline{x}_{m-1},\overline{x}_{m-1}\right],\left[0,-\underline{x}_m\right]\right).$$

The integral value is the difference of the first and second volumes. 2) The scanning of the complementary sets also facilitates the computation of an error bound. 3) If $B\left[f_1,c,0\right]$ is a solution box to the inequality $f_1(x)\ge 0$ and $B\left[f_2,c,0\right]$ is a solution box to the inequality $f_2(x)\ge 0,$ then the box $B\left[f_1,c,0\right]\cap B\left[f_2,c,0\right]$ is a solution box to the system of the two inequalities. 4) If U and T are m-dimensional boxes, then the set $U-T$ can be divided into (at most) $2m$ boxes easily. 5) The too small boxes (the volume is too small) are filtered by the simple condition $vol(B)>\kappa$. The algorithmic description of the method is as follows.

(a) If $\underline{x}_m=0,$ then call (c)-(e) with

$$I=\left(\left[\underline{x}_1,\overline{x}_1\right],\cdots,\left[\underline{x}_{m-1},\overline{x}_{m-1}\right],\left[0,\overline{x}_m\right]\right)$$

and $f_n(x)=f(x)-x_m$.
Print *avi*=*avi*+*eps*/2, *eps*=*eps*/2, *exb* and stop.

(b) If $\underline{x}_m<0,\ \ \overline{x}_m>0,$ then call (c)-(e) with

$$I=\left(\left[\underline{x}_1,\overline{x}_1\right],\cdots,\left[\underline{x}_{m-1},\overline{x}_{m-1}\right],\left[0,\overline{x}_m\right]\right)$$

and $f_n(x)=f(x)-x_m$.
Let $avii=avi+eps/2,\ epss=eps/2,\ exbb=exb.$
Call (c)-(e) with

$$I=\left(\left[\underline{x}_1,\overline{x}_1\right],\cdots,\left[\underline{x}_{m-1},\overline{x}_{m-1}\right],\left[0,-\underline{x}_m\right]\right)$$

and $f_n(x)=-f(x)-x_m$.
Print *avii* = *avi i*− *avi* − *eps* / 2, *epss* = *epss* + *eps* / 2, *exbb* = *exbb* + *exb* and stop.

(c) Define the first element of an interval (box) sequence $\left\{I_k\right\}$ by $I_1=I.$ Let $nob=1,\ \ exb=0,\ \ avi=0,\ \ eps=vol(I),$ where nob, exb, avi, eps denote the number of boxes in the sequence, the number of the boxes examined, the approximating value of the integral value, the error bound, respectively.

(d) Let $exb=exb+1.$ Compute the first i^* where $f_{i^*}(c)=\min f_i(c)$ if $1\le i\le n$ and c is the centre of I_{nob} .

(d1) If $f_{i^*}(c)<0$, then compute the box $B=B(f_{i^*},c,0)\subset\overline{S}=I-S.$ (The 'worst inequality' is used here.) Let $eps=eps-vol(B).$

(d2) If $f_{i^*}(c)\ge 0$, then $B:=I_{nob}$ and $B:=B\cap B(f_i,c,0),\, i=1,2,\cdots,n.$
Let $avi=avi+vol(B),\ \ eps=eps-vol(B).$

(e) Divide the set $I_{nob}-B$ into nb boxes (if the set is empty, then $nb:=0$). Filter the 'unimportant' (too small) boxes by the condition $vol\,(\text{box})>\kappa,$ where κ is a (small) given value. Place the $nb^*\le nb$ new boxes into the box sequence $\left\{I_k\right\}$ as nobth, $\left(nob+1\right)$th$,\cdots,(nob+nb^*-1)$th elements and let $nob=nob+nb^*-1.$ If $nob>0,$ then go to (d). If $nob=0,$ then go to the calling point.

The C++ program uses the above 'reminding names' and $\left\{I_k\right\}\leftrightarrow Ise,\ \left\{c_k\right\}\leftrightarrow Ice,\ \kappa\leftrightarrow kap.$ This algorithm is a strongly improved version of a method in [3]. Here solve the problem

$$\iiint_V\left(x_2^2+x_3^2\right)dx_1dx_2dx_3,$$

where V is described by the inequality

$$\left(2-x_3\right)^2-4x_1^2-4x_2^2\ge 0, x_3\in\left[0,2\right].$$

(A triple integral with a cone region—the radius of the base circle is 1 unit, the altitude is 2 units—is given.) The exact value (which can be obtained by using cylinder coordinates) is $11\pi/30\approx 1.1519.$ For $\kappa=10^{-4},$ $\kappa=10^{-5},$ $\kappa=10^{-6}$ the integral value, the error bound, the real error, the number of boxes examined and the running time (with our Visual C++ version 6.0 code on a PC of two 2.2 GHz processors) are

1.1880, 0.3423, 0.0361, 9053, 0.08sec;

1.1581, 0.1684, 0.0062, 44191, 0.4sec;

1.1523, 0.0816, 0.0004, 217361, 2sec,

respectively. Observe that the ratios for the running times 0.08sec, 0.4sec, 2sec move together with the ratios for numbers of boxes examined 9053, 44191, 217361, i.e. the running time increases linearly. The method of [3] mentioned (which has not this property) can produce similar results in 0.2sec, 3.5sec, 146sec, respectively.

4. A Selection Method for System of Equations

Let the nonlinear system of equations

$$f_i\left(x_1,x_2,\cdots,x_m\right)=0, x=\left(x_1,x_2,\cdots,x_m\right)\in I\subset R^m,$$
$$i=1,2,\cdots,n;$$

or in short form

$$f\left(x\right)=0, x\in I, \text{where } f:I\subset R^m\to R^n$$

be given. We assume that the multivariate real functions f_i are continuous on the closed interval (box) I and built from the well-known real elementary functions. The aim is to find one root, i.e. to find a point z for which $\|f(z)\|_\infty < \varepsilon$. The method is based on the following four principles. 1) Select the 'most promising box' in every step. 2) Exclude a box from further examination in every step. 3) If U and T are m-dimensional boxes, then the set $U-T$ can be divided into (at most) $2m$ boxes easily. 4) The too small boxes are filtered by the simple condition $vol(B) > \kappa$. The algorithmic description of the method is as follows.

```
(a) Call (b)-(e). Print $place = z = c$, $fbest = \|f(z)\|_\infty$
    (the best norm value of $f$), $exb$ and stop.
(b) Define the first element of an interval (box) sequence $\{I_i\}$ by $I_1 = I$. Let $nob = 1$ and
    $exb = 0$, where $nob$ denotes the number of boxes in the sequence and $exb$ denotes the number of the
    boxes examined.
(c) Choose the first element $I_{i^*}$ of the box sequence $\{I_i\}$ for which $\|f(c)\|_\infty$ is the smallest value (we
    always use the 'most promising box'). Interchange the $i^*$ th and $nob$ th elements in the sequence.
(d) If $\|f(c)\|_\infty \ge \varepsilon$, where $c$ is the centre of the interval $I_{nob}$, then:
    (d1) Choose the first $f_{j^*}$ from among $f_1, f_2, \cdots, f_n$ which gives the largest value at $c$
         in absolute value (we may exclude the largest box from further examination by this $f_{j^*}$).
    (d2) Exclude the box $B = B(f_{j^*}, c, 0)$ around centre $c$. Divide the set $I_{nob} - B$ into $nb$
         boxes (if the set is empty, then $nb := 0$). Filter the 'unimportant' (too small) boxes by
         the condition $vol\,(\text{box}) > \kappa$, where $\kappa$ is a (small) given value. Place the $nb^* \le nb$ new
         boxes into the box sequence $\{I_i\}$ as $nob$th, $(nob+1)$th, $\cdots$, $(nob+nb^*-1)$th elements and
         let $nob = nob + nb^* - 1$. Go to (c).
(e) If $\|f(c)\|_\infty < \varepsilon$, where $c$ is the centre of the
    interval $I_{nob}$, then go to the calling point.
```

The C++ program uses the above 'reminding names' and $\{I_i\} \leftrightarrow Ise$, $\{c_i\} \leftrightarrow Ice$, $\kappa \leftrightarrow kap$, $\varepsilon \leftrightarrow eps$. This algorithm is a simplified and improved version of a method in [4]. Now solve the problem

$$\ln\left((x_1+1)/2\right) + x_3 - 5 = 0, |x_1 - x_3| - x_2^2 + 5 = 0,$$

$$x_3/(x_1 + x_2 + 1) + x_3 - 6 = 0,$$

$$\exp\left(x_3 - |x_4| + 2\right) - \arctan\left(|x_4| - 7\right) - 1 = 0.$$

It has one solution which will be searched in different starting intervals I. The exact root is $\hat{z} = (1,3,5,7)$. For fixed $\kappa = 10^{-12}$, $\varepsilon = 10^{-3}$ three different starting intervals are used. The interval I, the error $\|\hat{z} - z\|_\infty$, the number of examined boxes and the running time (with our Visual C++ version 6.0 code on a PC of two 2.2 GHz processors) are

$$([0,10], \cdots, [0,10]) \Rightarrow 0.0010, 179, 0.0024\,\text{sec};$$

$$([-10,10], \cdots, [-10,10]) \Rightarrow 0.0018, 284, 0.0037\,\text{sec};$$

$$([0,100], \cdots, [0,100]) \Rightarrow 0.0011, 370, 0.0047\,\text{sec},$$

respectively. Here the selection character is dominant, because of very small κ and small ε. If the aim is to produce a suitable starting vector for a fast (finishing) method (e.g. a Newton-type method) in a more complicated problem, then it is practical to utilize the scanning character by greater κ and ε (e.g. $\kappa = 10^{-4}$, $\varepsilon = 1$).

5. A Scanning Method and a Selection Method for Global Extremes

Consider the problem

maximize $f(x_1, x_2, \cdots, x_{m-1})$

subject to

$$f_i(x_1, x_2, \cdots, x_{m-1}) \ge 0, \quad i = 1, 2, \cdots, n-1, \quad m \ge 2, \quad n \ge 1;$$

$$(x_1, \cdots, x_{m-1}) \in D = \left([\underline{x}_1, \overline{x}_1], \cdots, [\underline{x}_{m-1}, \overline{x}_{m-1}]\right) \subset R^{m-1},$$

where the multivariate real functions f_i, $i = 1, \cdots, n-1$ and f are continuous on the box D and built from the well-known elementary functions. Let us assume that we know (rough) lower and upper bounds $\underline{x}_m, \overline{x}_m$, that

$$\underline{x}_m \le f(x_1, x_2, \cdots, x_{m-1}) \le \overline{x}_m, \quad \forall (x_1, x_2, \cdots, x_{m-1}) \in D.$$

Define the system of inequalities

$$f_i(x_1, x_2, \cdots, x_{m-1}) \ge 0, \quad i = 1, 2, \cdots, n-1$$

$$f(x_1, x_2, \cdots, x_{m-1}) - x_m \ge 0,$$

where

$$x = (x_1, x_2, \cdots, x_m) \in I = \left([\underline{x}_1, \overline{x}_1], \cdots, [\underline{x}_{m-1}, \overline{x}_{m-1}], [\underline{x}_m, \overline{x}_m]\right),$$

or briefly

$$f_i(x_1, x_2, \cdots, x_m) \ge 0, i = 1, 2, \cdots, n, (x_1, x_2, \cdots, x_m) \in I,$$

where

$$f_n(x_1, x_2, \cdots, x_m) = f(x_1, x_2, \cdots, x_{m-1}) - x_m.$$

The solution of our problem is a point of the solution set S of this system of inequalities with the largest mth coordinate. Our aim is to find a good approximation $obest$ of the maximum function value (belonging to the objective function f and the set A of feasible points) and to prove that the value $obest + eps$ (where eps is a supposed error bound) is an upper bound to the mth coordinate of the solution. The method is based on the following four principles. 1) If $B[f_1, c, 0]$ is a solution box to the inequality $f_1(x) \ge 0$ and $B[f_2, c, 0]$ is a solution box to the inequality $f_2(x) \ge 0$, then the box

$B[f_1,c,0]\cap B[f_2,c,0]$ is a solution box to the system of the two inequalities. 2) If U and T are m-dimensional boxes, then the set $U-T$ can be divided into (at most) $2m$ boxes easily. 3) Here it is sufficient to do a fine scanning only around the solution point, therefore a second filter is used besides the simple one seen in the integral algorithm.The too small boxes are filtered by the condition $vol(B)>\kappa$ and a second filter $\overline{x}_m > obest$ (where $\overline{x}_m$ is the maximum value of the mth coordinate in the box) saves much needless work. 4) To prove that the value $obest+eps$ is an upper bound to the maximum value it is sufficient to see (because of the continuity of f on D) that the 'narrow stripe'

$$\left([\underline{x}_1,\overline{x}_1],\cdots,[\underline{x}_{m-1},\overline{x}_{m-1}],[obest+\varepsilon,obest+2\varepsilon]\right)$$

has no common point with *S*. The algorithmic description of the method is as follows.

```
(a) Call (b)-(d) with
        $I=\left([\underline{x}_1,\overline{x}_1],\cdots,[\underline{x}_m,\overline{x}_m]\right)$
    and $\kappa>0$. Print place, obest, exb. Call (b)-(d) with
        $I=\left([\underline{x}_1,\overline{x}_1],\cdots,[\underline{x}_{m-1},\overline{x}_{m-1}],[obest+\varepsilon,obest+2\varepsilon]\right)$
    and $\kappa=0$. If $obest<\underline{x}_m$, then print upper bound
    $obest+\varepsilon$. Print exb and stop.
(b) Define the first element of an interval (box) sequence
    $\{I_k\}$ by $I_1=I$. Let $nob=1$, $exb=0$, $obest=-\infty$,
    where nob, exb, obest denote the number of boxes
    in the sequence, the number of the boxes examined,
    the approximating value of the global maximum, respectively.
(c) Let $exb=exb+1$. Compute the first $i^*$ where
    $f_{i^*}(c)=\min f_i(c)$ if $1\le i\le n$ and $c$ is the centre
    of $I_{nob}$. If $f_{i^*}(c)\ge 0$ and $f(c)=f_n(c)+c_m>obest$,
    then let $place=(c_1,\cdots,c_{m-1})$, $obest=f(c)$.
    (c1) If $f_{i^*}(c)<0$, then compute the box
            $B=B(f_{i^*},c,0)\subset\overline{S}=I-S.$
    (c2) If $f_{i^*}(c)\ge 0$, then $B:=I_{nob}$ and
            $B:=B\cap B(f_i,c,0),\quad i=1,2,\cdots,n.$
(d) Divide the set $I_{nob}-B$ into nb boxes (if the set is
    empty, then $nb:=0$). Filter the 'unimportant' boxes
    by the conditions vol (box) $>\kappa$ and $\overline{x}_m>obest$.
    Place the $nb^*\le nb$ new boxes into the box sequence $\{I_k\}$ as
    nobth, $(nob+1)$th, $\cdots$, $(nob+nb^*-1)$th elements and
    let $nob=nob+nb^*-1$. If $nob>0$, then go to (c).
    If $nob=0$, then go to the calling point.
```

The C++ program uses the above 'reminding names' and $\{I_k\}\leftrightarrow Ise$, $\{c_k\}\leftrightarrow Ice$, $\kappa\leftrightarrow kap$. This algorithm does not appear in other papers of the author referred to. Here solve the problem described by

$$\frac{2+\cos(x_1-3)\cos(x_2+2)}{1+|x_1|+4|x_2|}\to\max,\quad \left.\begin{array}{l}16-x_1^2-4x_2^2\ge 0\\ x_1^2-x_2^2-4\ge 0\end{array}\right\}$$

and illustrated, with $D=([-5,5],[-5,5])$, in the **Figures 2-3**. The exact solution is $(-2,0,(2+\cos 5\cdot\cos 2)/3)\approx(-2,0,0.6273)$. For $\kappa=10^{-5},\kappa=10^{-7},\kappa=10^{-9}$, beside fixed $\varepsilon=10^{-2}$, the solution vector, the number of examined boxes (to the solution vector + to the supposed error bound $\varepsilon=10^{-2}$) and the running time (with our Visual C++ version 6.0 code on a PC of two 2.2 GHz processors) are

$(-2.0030,0.0063,0.6206)$, $3160+317$, 0.045sec;

$(-2.0030,0.0007,0.6256)$, $14431+274$, 0.21sec;

$(-2.0004,-0.0002,0.6271)$, $49407+265$, 0.71sec,

respectively.

Observe that the 'linearity' is excellent and the proof of the supposed error bound requires insignificant work. For a practical problem (with an uncertain error bound) this work could increase considerably, therefore the second part of our examination is sometimes omitted.

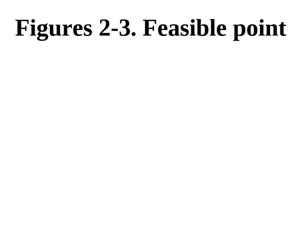
Now consider the problem

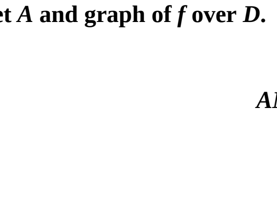
minimize $f(x_1,x_2,\cdots,x_m)$

subject to $f_i(x_1,x_2,\cdots,x_m)\ge 0, i=1,2,\cdots,n, m\ge 1, n\ge 1;$

$$(x_1,\cdots,x_m)\in D=\left([\underline{x}_1,\overline{x}_1],\cdots,[\underline{x}_m,\overline{x}_m]\right)\subset R^m,$$

where the multivariate real functions f_i, $i=1,\cdots,n$ and f are continuous on the box D and built from the well-known elementary functions, furthermore the objective function f is strictly increasing for every variable on D, i.e. the best (the minimum) value of $f(x)$ on a box $([\alpha_1,\beta_1],\cdots,[\alpha_m,\beta_m])$ appears at the 'left lower vertex' $\alpha=(\alpha_1,\cdots,\alpha_m)$. Our aim is to create a reliable method for finding the minimum function value (belonging to the objective function f and the set A of feasible points). The method is based on the following four principles. 1) Select the 'most promising box' (for which the box centre best satisfies the inequalities describing the set A of feasible points) at the beginning of the running, hereby take advantage of the speciality of the objective function in the filtering. 2) If $B[f_1,c,0]$ is a solution box to the inequality $f_1(x)\ge 0$ and $B[f_2,c,0]$ is a solu-





Figures 2-3. Feasible point set *A* and graph of *f* over *D*.

tion box to the inequality $f_2(x) \geq 0$, then the box $B[f_1,c,0] \cap B[f_2,c,0]$ is a solution box to the system of the two inequalities. 3) If U and T are m-dimensional boxes, then the set $U-T$ can be divided into (at most) $2m$ boxes easily. 4) The 'unimportant' boxes are filtered by the conditions $vol\,(\text{box}) > \kappa$ and $f(\alpha) < obest$ (α is the 'left lower vertex'). The algorithmic description of the method is as follows.

(a) Call (b)-(e) with $I = D = ([\underline{x}_1, \overline{x}_1], \cdots, [\underline{x}_m, \overline{x}_m])$ and $\kappa > 0$. Print *place*, *obest*, *exb*, *nsb*.

(b) Define the first element of an interval (box) sequence $\{I_i\}$ by $I_1 = I$. Let $nob = 1$, $exb = 0$, $nsb = 0$, $obest = \infty$, where *nob*, *exb*, *nsb*, *obest* denote the number of boxes in the sequence, the number of the boxes examined, the number of the solution boxes in A, the best discovered value of the objective function in A, respectively.

(c) If $nsb < rsb$ (the partial selection works as long as the number of solution boxes is less than the required number), then choose the first element I_{i^*} of the box sequence $\{I_i\}$ for which $\min f_j(c)$, $1 \leq j \leq n$, (c is the centre of I_i) is the largest value (we use the 'most promising box'). Interchange the i^*th and *nob*th elements in the sequence.

(d) Let $exb = exb + 1$. Take out the first j^* where $f_{j^*}(c) = \min f_j(c)$ if $1 \leq j \leq n$ and c is the centre of I_{nob}.

(d1) If $f_{j^*}(c) < 0$, then compute the box $B = B(f_{j^*}, c, 0) \subset \overline{S} = I - S$. (The 'worst inequality' is used for exclusion.)

(d2) If $f_{j^*}(c) \geq 0$, then $nsb := nsb + 1$; $B := I_{nob}$ and $B := B \cap B(f_i, c, 0)$, $i = 1, 2, \cdots, n$.
If $f(\alpha) = f_{n+1}(\alpha) < obest$, where $\alpha = (\alpha_1, \cdots, \alpha_m)$ is the 'left lower vertex' of the box B of feasible points, then $place = \alpha$, $obest = f(\alpha)$.

(e) Divide the set $I_{nob} - B$ into *nb* boxes (if it is empty, then $nb := 0$). Filter the 'unimportant' boxes by the conditions $vol\,(\text{box}) > \kappa$ and $f(\alpha) < obest$ (α is the 'left lower vertex'). Place the $nb^* \leq nb$ new boxes into the box sequence $\{I_i\}$ as *nob*th, $(nob+1)$th, $\cdots$, $(nob + nb^* - 1)$th elements and let $nob = nob + nb^* - 1$. If $nob > 0$, then go to (c). Otherwise go to the calling point.

The C++ program uses the above 'reminding names' and $\{I_i\} \leftrightarrow Ise, \{c_i\} \leftrightarrow Ice, \kappa \leftrightarrow kap$. This algorithm shows some similarity to a method in [5]. Here solve the optimal design problem described by

$$7x_1x_3 + 4x_2x_4\sqrt{1+x_5^2} \to \min$$

subject to

$$x_2x_3 - 0.1x_1x_3 - 0.01x_1^2 \geq 0,$$

$$\left(\xi + \left(\xi^2 - \frac{26732.25}{x_1^2}\right)^{1/2}\right)^{-1} - \frac{194.37}{x_1x_3x_5} \geq 0,$$

where $\xi = 0.47 + 27.80/x_1 + 13366.13/x_1^2$,

$$\left(\eta + \left(\eta^2 - 6684.70\frac{1+x_5^2}{x_2^2}\right)^{1/2}\right)^{-1} - 85.04\frac{\left(1+x_5^2\right)^{1/2}}{x_2x_4x_5} \geq 0,$$

where

$$\eta = 0.47 + 13.90\left(1+x_5^2\right)^{1/2}/x_2 + 3342.35\left(1+x_5^2\right)/x_2^2,$$

$$x_2^2x_3^3\left(1.3 - \frac{77.75}{x_2x_3x_5}\right)^2 - 2921.25x_1 \geq 0, x_2 - 0.35x_1 \geq 0,$$

$$x_1 - x_2 \geq 0, \left(1.5 - 0.1x_5\right)x_1 - x_2\left(1+x_5^2\right)^{1/2} \geq 0,$$

$$x_1 - 15x_3 \geq 0, 35x_3 - x_1 \geq 0, 3000x_4 - 100x_2 \geq 0;$$

$$(x_1, x_2, \cdots, x_5) \in D = ([100,120],[80,100],[1,11],[1,11],[1,11]),$$

which satisfies the above conditions. For $\kappa = 1$, $\kappa = 10^{-1}$, $\kappa = 10^{-2}$, beside fixed $rsb = 100$, the objective function value belongs to the best discovered place, the number of boxes examined, the number of solution boxes in the set A of feasible points and the running time (with our Visual C++ version 6.0 code on a PC of two 2.2 GHz processors) are

4849.71, 6168, 907, 0.31sec;

4836.46, 46607, 2367, 2.1sec;

4722.09, 379336, 7052, 17sec,

respectively. Similar results (obtained by much more complicated methods) can be seen in [5]. The volume of the starting box D is fairly large ($4 * 10^5$ units), therefore the use of too small κ (a too fine scanning of D) could require a long time to run (on our PC).

6. Appendix: C++ Codes for the Five Algorithms

Our Visual C++ version 6.0 programs have 5 segments for each of the five algorithms. The first 3 segments are the same in these codes. For computing the solution boxes of an inequality the function segment *solbox* is used, which handles the function

$$\text{solbox:} (D, G, c, \alpha, m, nt) \mapsto B(g, c, \alpha),$$

where D is the domain box of the multivariate real function g, G is a numerically coded form of g, $c \in D$, $\alpha \in R$, m is the number of the variables in g and nt is the number of triples in G. The complete segment *solbox* (which is the base of all five methods) is published in [1]. The function segment *fval* computes the function values from the numerically coded form, i.e. it handles the function

$$\text{fval}: (G, c, nt) \mapsto g(c),$$

where *G* is a numerically coded form of the multivariate real function *g*, *c* is a point of the domain and *nt* is the number of triples in *G*. To divide the difference of a closed *m*-dimensional box *U* and an open *m*-dimensional box *T* into closed boxes, $box_1, box_2, \cdots, box_{nb}$ which do not contain common interior points, our function segment *divi* handles the function

$$\text{divi}: (U, T, m) \mapsto (\{box_1, box_2, \cdots, box_{nb}\}, nb)$$

An algorithmic description and a two-dimensional illustration of this simple process can be seen e.g. in [3]. The functions of the fourth segments are

$$\text{scanvol}: (F, I, \kappa, m, n) \mapsto (vol, eps, exb),$$

$$\text{scanint}: (F, I, \kappa, m, n) \mapsto (avi, eps, exb),$$

$$\text{selecteqs}: (F, I, \kappa, \varepsilon, m, n) \mapsto (place, fbest, exb),$$

$$\text{scanmax}: (F, I, \kappa, m, n) \mapsto (place, obest, exb),$$

$$\text{selectmin}: (F, I, \kappa, rsb, m, n) \mapsto (place, obest, exb, nsb),$$

where *F* contains all the multivariate functions needed in triple form and the other parameters are the same as in the algorithmic descriptions. The task of the fifth (main) segments is given at the beginning of the algorithmic descriptions. A complete code of the first method is as follows.

```
#include <iostream.h>
#include <math.h>
double Ise[100000][10][3], Ice[100000][10];
/* THE OUTPUT PARAMETERS OF THE NEXT 4 FUNCTIONS */
double B[10][3];
double gc;
double boxes[20][10][3]; int nb;
double vol,eps; int exb;
/* FUNCTION: SOLUTION BOXES OF INEQUALITY */
void solbox(double D[][3],double G[][4],double c[],double alp,int m,int nt)
 {see in [1]}
/* FUNCTION: FUNCTION VALUE g(c)*/
void fval(double G[][4],double c[],int nt)
  {double fv[100],x,y,w; int i,j,k,l; const double Pi=3.14159265;
  for (i=1; i<=nt; i++)
       {j=(int)G[i][1]; k=(int)G[i][2]; l=(int)G[i][3]; w=G[i][3];
       if (k<0) x=c[-k]; else x=fv[k]; if (j>=3 && j<=5) y=fv[l];
       switch (j)
            {case 1:fv[i]=x+w;break; case 2:fv[i]=x*w;break; case 3:fv[i]=x+y;break;
            case 4:fv[i]=x/y;break; case 5:fv[i]=x*y;break; case 6:fv[i]=pow(x,w);break;
            case 7:fv[i]=exp(x);break; case 8:fv[i]=log(x);break; case 9:fv[i]=fabs(x);break;
            case 10:fv[i]=sin(x);break; case 11:fv[i]=cos(x);break; case 12:fv[i]=tan(x);break;
            case 13:fv[i]=1/tan(x);break; case 14:fv[i]=asin(x);break; case 15:fv[i]=acos(x);break;
            case 16:fv[i]=atan(x);break; case 17:fv[i]=Pi/2-atan(x);break;}}
  gc=fv[nt];}
/* FUNCTION: DIFFERENCE OF TWO m-DIMENSIONAL BOXES */
void divi(double U[][3],double T[][3],int m)
  {double ax[10][3],len[10],x; int ind[10],i,j,k;
  nb=0; for (i=1; i<=m; i++) {ax[i][1]=U[i][1]; ax[i][2]=U[i][2]; len[i]=U[i][2]-U[i][1];}
  /*Permutation of indexes by side lengths of the box U*/
  for (i=1; i<=m; i++) {j=1; for (k=1; k<=m; k++)
       {if (len[k]>len[i]) j=j+1; if (len[k]==len[i] && k<i) j=j+1;}; ind[j]=i;}
  /*Dividing the set U-T*/
  for (i=1; i<=m; i++) {j=ind[i]; x=ax[j][1];
       if (ax[j][1]<T[j][1]) {ax[j][1]=T[j][1]; nb=nb+1;
            for (k=1; k<=m; k++) {boxes[nb][k][1]=ax[k][1]; boxes[nb][k][2]=ax[k][2];}
            boxes[nb][j][1]=x; boxes[nb][j][2]=T[j][1];}
        x=ax[j][2];
        if (ax[j][2]>T[j][2]) {ax[j][2]=T[j][2]; nb=nb+1;
            for (k=1; k<=m; k++) {boxes[nb][k][1]=ax[k][1]; boxes[nb][k][2]=ax[k][2];}
            boxes[nb][j][1]=T[j][2]; boxes[nb][j][2]=x;}}}
/* FUNCTION: COMPUTING AREA AND VOLUME */
void scanvol(double F[][100][4],double I[][3],double kap,int m,int n)
  {double ce[10],ax[100][4],xx[10][3],res[10][3],minf,vo; int nob,i,j,k,iast,N;
  for (i=1; i<=m; i++) {Ise[1][i][1]=I[i][1]; Ise[1][i][2]=I[i][2]; Ice[1][i]=(I[i][1]+I[i][2])/2;}
  vo=1; for (i=1; i<=m; i++) vo=vo*(I[i][2]-I[i][1]);
  nob=1; exb=0; vol=0.; eps=vo;
  /*Using solution boxes of inequality*/
  while (nob>0)
      {exb=exb+1; for (i=1; i<=m; i++) ce[i]=Ice[nob][i]; minf=1.e10;
      for (i=1; i<=n; i++) {N=(int)F[i][0][0];
            for (j=1; j<=N; j++) for (k=1; k<=3; k++) ax[j][k]=F[i][j][k];
            fval(ax,ce,N); if (gc<minf) {minf=gc; iast=i;}}
      if (minf<0)
            {N=(int)F[iast][0][0]; for (i=1; i<=N; i++) for (j=1; j<=3; j++) ax[i][j]=F[iast][i][j];
            solbox(I,ax,ce,0.,m,N); vo=1.;
            for (i=1; i<=m; i++) {res[i][1]=B[i][1]; res[i][2]=B[i][2];
                  if (Ise[nob][i][1] > B[i][1]) B[i][1]=Ise[nob][i][1];
                  if (Ise[nob][i][2] < B[i][2]) B[i][2]=Ise[nob][i][2]; vo=vo*(B[i][2]-B[i][1]);}
            eps=eps-vo;}
      if (minf>=0)
            {for (i=1; i<=m; i++) {res[i][1]=Ise[nob][i][1]; res[i][2]=Ise[nob][i][2];}
            for (i=1; i<=n; i++)
                  {N=(int)F[i][0][0]; for (j=1; j<=N; j++) for (k=1; k<=3; k++) ax[j][k]=F[i][j][k];
                  solbox(I,ax,ce,0.,m,N);
                  for (j=1; j<=m; j++)
                        {if (B[j][1]>res[j][1]) res[j][1]=B[j][1]; if (B[j][2]<res[j][2]) res[j][2]=B[j][2];}}; vo=1;
            for (i=1; i<=m; i++) vo=vo*(res[i][2]-res[i][1]); vol=vol+vo; eps=eps-vo;}
      for (i=1; i<=m; i++) {xx[i][1]=Ise[nob][i][1]; xx[i][2]=Ise[nob][i][2];}
      nob=nob-1;
      /*Dividing the actual box*/
      divi(xx,res,m);
      for (i=1; i<=nb; i++)
            {vo=1.; for (j=1; j<=m; j++) vo=vo*(boxes[i][j][2]-boxes[i][j][1]);
            if (vo>kap) {nob=nob+1;
                  for (j=1; j<=m; j++)
                        {Ise[nob][j][1]=boxes[i][j][1]; Ise[nob][j][2]=boxes[i][j][2];
                        Ice[nob][j]=(boxes[i][j][1]+boxes[i][j][2])/2;}}}}}
/* THE CALLING SEGMENT, example 1 */
void main()
  {double F[10][100][4],I[10][3],kap; int m,n,i,j;
  double f1[6][3]= {{6,-1,2},{6,-2,2},{2,1,-1},{2,2,-4},{3,3,4},{1,5,16.}};
  double f2[4][3]= {{6,-1,2},{6,-2,2},{3,1,2},{1,3,-4}};
  m=2; n=2;
  F[1][0][0]=6; for (i=1; i<=6; i++) for (j=1; j<=3; j++) F[1][i][j]=f1[i-1][j-1];
  F[2][0][0]=4; for (i=1; i<=4; i++) for (j=1; j<=3; j++) F[2][i][j]=f2[i-1][j-1];
  I[1][1]=-5.; I[1][2]=5.; I[2][1]=-5.; I[2][2]=5.;
  kap=1.e-6; scanvol(F,I,kap,m,n); vol=vol+eps/2; eps=eps/2;
  cout <<"Volume, error bound: "<< vol <<" "<< eps << endl;
  cout <<"Examined boxes: "<< exb << endl;}
```

To avoid the dimension trouble, the two large arrays *Ise* and *Ice* (which could contain thousands of box data) are declared at the beginning of the program. The calling (main) segment uses a simple trick (push down of indexes) for the easy handling of triple forms. The codes of the further four methods are very similar to this code; the author would gladly send them to interested readers in e-mail as attached files. Finally two remarks: 1) The computation efforts (the evaluation times) belonging to $B(g,c,\alpha)$ and $g(c)$ can be characterized well enough by the formula: effort $\left(B(g,c,\alpha)\right) \approx 10 *$ effort $g(c)$. 2) We always used floating point arithmetic for computing solution boxes. Since these boxes are computed by lower estimates, we have never happened to obtain a faulty result because of the effect of rounding errors.

7. References

[1] F. Kálovics, "A New Tool: Solution Boxes of Inequality," *Journal of Software Engineering and Applications*, Vol. 3, No. 8, 2010, pp. 737-745.

[2] R. Hammer, M. Hocks, U. Kulisch and D. Ratz, "Numerical Toolbox for Verified Computing", *Springer-Verlag, Berlin*, 1993.

[3] F. Kálovics, "Zones and Integrals," *Journal of Computational and Applied Mathematics*, Vol. 182, No. 2, 2005, pp. 243-251.

[4] F. Kálovics and G. Mészáros, "Box Valued Functions in Solving Systems of Equations and Inequalities," *Numerical Algorithms*, Vol. 36, No. 1, 2004, pp. 1-12.

[5] F. Kálovics, "Solving Nonlinear Constrained Minimization Problems with a New Interval Valued Function," *Reliable Computing*, Vol. 5, No. 4, 1999, pp. 395-406.