J. Software Engineering & Applications, 2010, 3, 737-745
doi:10.4236/jsea.2010.38085 Published Online August 2010 (http://www.SciRP.org/journal/jsea)

A New Tool: Solution Boxes of Inequality

Ferenc Kálovics

Analysis Department, University of Miskolc, Miskolc-Egyetemváros, Hungary.
Email: matkf@uni-miskolc.hu

Received May 4th 2010; revised June 19th 2010; accepted June 23rd 2010.

ABSTRACT

Many numerical methods reduce the solving of a complicated problem to a set of elementary problems. In this way, the author reduced the finding of solution boxes of a system of inequalities, the computation of integral values with error bounds, the approximation of global maxima to computing solution boxes of one inequality. These previous papers presented the mathematical aspects of the problems first of all. In this paper the computational handling of solution boxes of an inequality (*as a new tool of computer methods*) *is discussed in detail.*

Keywords: *Inequality, Solution Box, Coded Form of Expression*

1. Introduction

Let $g : D \subset \mathrm{R}^m \to \mathrm{R}$ be a continuous multivariate real function, where $D = \left((\underline{x}_1, \overline{x}_1), (\underline{x}_2, \overline{x}_2), ..., (\underline{x}_m, \overline{x}_m)\right)$ is an open box. Define the box $B[g, c, \alpha] \subset D$, where $c \in D, \alpha \in \mathrm{R}$, 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 $x \in B[g, c, \alpha]$, if $g(c) > \alpha$, then $g(x) > \alpha$ for all $x \in B[g, c, \alpha]$. Now consider a particular case. Let $g : D \subset \mathrm{R}^2 \to \mathrm{R}, g(x_1, x_2) = \sin(x_1 x_2) + \frac{x_1^2}{\ln x_2}$, where $D = \left((0, \pi), (1, 4)\right)$ is an open box. Try to give the box $B\left[\sin(x_1 x_2) + x_1^2 / \ln x_2, (1, 2), 1\right]$.

Since $g(c) \approx 2.35 > 1$, an open box around point $(1, 2)$ in which $g(x) > 1$ is searched. Use the decomposition

$$B\left[\sin(x_1 x_2) + x_1^2 / \ln x_2, (1, 2), 1\right] =$$
$$B\left[\sin(x_1 x_2), (1, 2), \beta\right] \cap B\left[x_1^2 / \ln x_2, (1, 2), \gamma\right].$$

The spontaneous choice $\beta = \gamma = 0.5$ is suitable because $\sin(1 \cdot 2) > 0.5$ and $1^2 / \ln 2 > 0.5$, hence the boxes $B[\sin(x_1 x_2), (1, 2), 0.5]$ and $B\left[x_1^2 / \ln x_2, (1, 2), 0.5\right]$, around c, satisfy the properties $\sin(x_1 x_2) > 0.5$ and $x_1^2 / \ln x_2 > 0.5$, respectively. Since the decompositions

$$B[\sin(x_1 x_2), (1, 2), 0.5] =$$
$$B[x_1 x_2, (1, 2), \pi / 6] \cap B[x_1 x_2, (1, 2), 5\pi / 6] =$$
$$B[x_1, (1, 2), \pi / 6] \cap B[x_2, (1, 2), 1] \cap$$
$$B\left[x_1, (1, 2), \sqrt{5\pi / 12}\right] \cap B\left[x_2, (1, 2), \sqrt{5\pi / 3}\right],$$

where the first row shows that $\pi / 6 < x_1 x_2 < 5\pi / 6$, the second row shows that $x_1 > \pi / 6$, $x_2 > 1$, $x_1 < \sqrt{5\pi / 12}$, $x_2 < \sqrt{5\pi / 3}$ in the box $B[\sin(x_1 x_2), (1, 2), 0.5]$, and the decomposition

$$B\left[x_1^2 / \ln x_2, (1, 2), 0.5\right] =$$
$$B\left[x_1^2, (1, 2), 0.5\right] \cap B\left[\ln x_2, (1, 2), 1\right]$$

where the formula shows that $x_1^2 > 0.5, \ln x_2 < 1$ in the box $B\left[x_1^2 / \ln x_2, (1, 2), 0.5\right]$, are also fortunate, therefore

$$B\left[\sin(x_1 x_2) + x_1^2 / \ln x_2, (1, 2), 1\right] =$$
$$\left((\pi / 6, \pi), (1, 4)\right) \cap \left((0, \pi), (1, 4)\right) \cap$$
$$\left(\left(0, \sqrt{5\pi / 12}\right), (1, 4)\right) \cap \left((0, \pi), \left(1, \sqrt{5\pi / 3}\right)\right) \cap$$
$$\left(\left(\sqrt{0.5}, \pi\right), (1, 4)\right) \cap \left((0, \pi), (1, e)\right) \approx \left((0.71, 1.14), (1, 2.28)\right).$$

To get the result, spontaneous rules were used for sum (+), product (·), quotient (/) and composit function (∘). In the last step the boxes were computed to some elementary functions. Naturally, the decomposition can break down with any spontaneous rule, e.g. the choice

$\beta = 0.95, \gamma = 0.05$ does not work in the first step, although $\beta + \gamma = 1$. If the guaranteed rules of Theorem 1 of [1] are used, then

$$B\left[\sin(x_1 x_2) + x_1^2 / \ln x_2, (1,2), 1\right] \approx$$
$$\left((0.84, 1.21), (1, 2.41)\right).$$

Our aim is to make a C++ program which can follow the above decomposition process. Suppose that the continuous multivariate real function $g : D \subset \mathrm{R}^m \to \mathrm{R}$, where $D = \left((\underline{x}_1, \overline{x}_1), (\underline{x}_2, \overline{x}_2), ..., (\underline{x}_m, \overline{x}_m)\right)$, is built of the well-known (univariate real) elementary functions:

$$x^r,\ e^x,\ \ln x,\ |x|,\ \sin x,\ \cos x,\ \tan x, \cot x,$$
$$\arcsin x,\ \arccos x,\ \arctan x,\ \operatorname{arc}\cot x$$

by the function operations $+, *, /, \circ$. (It is supposed that r of x^r is a positive integer or negative integer or reciprocal of a positive integer. This is only a formal restriction because of $x^{k/n} = \left(x^k\right)^{1/n}, \forall k, n \in \mathrm{N}^+$.) The functions $a^x, \log_a x, \sinh x,$ etc. can be used by the connections $a^x = e^{x \ln a}$, $\log_a x = \ln x / \ln a$, $\sinh x = (e^x - e^{-x})/2,$ etc., respectively, the operation of difference can be used in form $a - b = a + (-1)b.$ Naturally, the program can be enlarged by further elementary functions, but we would like to make a program in a publishable length. The notations, names, definitions and discussions are simpler and clearer than they were in the former papers of the author (experience has slowly become knowledge). Using our new notations, the rules of Theorem 1 of [1] are as follows. (It is supposed that the functions p, q satisfy the restrictions belonging to the function g, the function e is an allowed elementary function, furthermore p/q and $e \circ q$ are well-defined on D.)

Rule of constant addition:

$B[p + \lambda, c, \alpha] = B[p, c, \beta]$, where $\lambda \in \mathrm{R}, \beta = \alpha - \lambda.$

Rule of constant multiplication:

$B[\lambda p, c, \alpha] = B[p, c, \beta]$, where $0 \neq \lambda \in \mathrm{R}, \beta = \alpha / \lambda.$

Rule of sum:

$$B[p + q, c, \alpha] = B[p, c, \beta] \cap B[q, c, \gamma],$$
where $\beta = (\alpha + p(c) - q(c))/2,\ \gamma = \alpha - \beta.$

Rule of quotient:

$$B[p/q, c, \alpha] = B[p, c, 0] \text{ if } \alpha = 0;$$
$$B[p/q, c, \alpha] = B[p, c, \beta] \cap B[q, c, \gamma], \text{ where}$$
$$\gamma = (\alpha p(c) + q(c))/(1 + \alpha^2), \beta = \alpha\gamma, \text{ if } \alpha \neq 0.$$

Rule of product:

$$B[pq, c, \alpha] = B[p, c, 0] \cap B[q, c, 0]$$
$$\text{if } \alpha = 0 \text{ or } \alpha p(c) q(c) < 0;$$
$$B[pq, c, \alpha] = B[p, c, \beta] \cap B[p, c, \gamma] \cap$$
$$B[q, c, \beta] \cap B[q, c, \gamma],$$

where $\beta = \sqrt{|\alpha|}, \gamma = -\beta,$ if $\alpha \neq 0, p(c) = 0, q(c) = 0;$

$$B[pq, c, \alpha] = B[p, c, \beta] \cap B[p, c, 0] \cap B[q, c, \gamma],$$

where $\beta = 2p(c), \gamma = \alpha/\beta,$ if $\alpha \neq 0, p(c) \neq 0, q(c) = 0;$

$$B[pq, c, \alpha] = B[p, c, \beta] \cap B[q, c, \gamma] \cap B[q, c, 0],$$

where $\gamma = 2q(c), \beta = \alpha/\gamma,$ if $\alpha \neq 0, p(c) = 0, q(c) \neq 0;$

$$B[pq, c, \alpha] = B[p, c, \beta] \cap B[p, c, 0] \cap B[q, c, \gamma],$$

where $\beta = \operatorname{sgn} p(c) \sqrt{\alpha p(c)/q(c)},$

$\gamma = \operatorname{sgn} q(c) \sqrt{\alpha q(c)/p(c)},$ if $\alpha p(c) q(c) > 0.$

Rule of composite function:

$$B[e \circ q, c, \alpha] = D \cap B\left[q, c, \beta_l^c\right] \cap B\left[q, c, \beta_r^c\right],$$

where β_l^c is the maximum value for which $e(\beta_l^c) = \alpha,\ \beta_l^c < q(c)$ and β_r^c is the minimum value for which $e(\beta_r^c) = \alpha,\ \beta_r^c > q(c).$ If β_l^c or β_r^c does not exist, then the actual member is omitted.

After the decomposition process the task is to evaluate the solution boxes belonging to elementary functions. The processes for the 12 cases (which are published now for the first time) can be obtained easily enough by the graphs of these functions. (The starting box is $D = \left((\underline{x}_1, \overline{x}_1), ..., (\underline{x}_i, \overline{x}_i), ..., (\underline{x}_m, \overline{x}_m)\right)$ in every case.)

Power function:

$B\left[x_i^r, c, \alpha\right] =$

$\left((\underline{x}_1, \overline{x}_1), ..., (\underline{x}_i, \overline{x}_i), ..., (\underline{x}_m, \overline{x}_m)\right),\ b_1 = \underline{x}_i,\ b_2 = \overline{x}_i,$

if $r \in \mathrm{Z}^+$ is odd, then $b_1 = \alpha^{1/r};$

if $r \in \mathrm{Z}^+$ is even and $\alpha \geq 0,$ then $b_1 = \alpha^{1/r}, b_2 = -b_1;$

if $r \in \mathrm{Z}^-$ is odd and $\alpha \neq 0,$ then $b_1 = \alpha^{1/r};$

if $r \in \mathrm{Z}^-$ is even and $\alpha > 0,$ then $b_1 = \alpha^{1/r}, b_2 = -b_1;$

if $r \in (0,1),\ 1/r \in \mathrm{Z}^+$ and $\alpha \geq 0,$ then $b_1 = \alpha^{1/r};$

if $b_1 \in (\underline{x}_i, c_i),$ then $\underline{x}_i = b_1,$ if $b_1 \in (c_i, \overline{x}_i),$ then $\overline{x}_i = b_1,$

if $b_2 \in (\underline{x}_i, c_i),$ then $\underline{x}_i = b_2,$ if $b_2 \in (c_i, \overline{x}_i),$ then $\overline{x}_i = b_2.$

Exponential function:

$B\left[e^{x_i},c,\alpha\right]=\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right)$, $b_1=\underline{x}_i$,

if $\alpha>0$, then $b_1=\ln\alpha$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$.

Logarithm function:

$B\left[\ln x_i,c,\alpha\right]=\left(\left(\underline{x}_1,\overline{x}_1\right),\left(\underline{x}_2,\overline{x}_2\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right)$, $b_1=e^{\alpha}$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$.

Absolute value function:

$B\left[\left|x_i\right|,c,\alpha\right]=$

$\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right),b_1=\underline{x}_i,b_2=\overline{x}_i$,

if $\alpha\geq 0$, then $b_1=\alpha$, $b_2=-\alpha$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$,

if $b_2\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_2$, if $b_2\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_2$.

Sinus function:

$B\left[\sin x_i,c,\alpha\right]=$

$\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right),b_1=\underline{x}_i,\ b_2=\overline{x}_i$,

$x=\arcsin\left|\alpha\right|$, $per=2\pi\cdot$ int $(c_i/2\pi)$, if $c_i<0$, then $per=per-2\pi$,

if $\alpha\in[0,1]$, then $b_1=x+per$, $b_2=\pi-x+per$;

if $\alpha\in[-1,0)$, then $b_1=\pi+x+per$, $b_2=2\pi-x+per$;

if $b_2<c_i$, then $y=b_1$, $b_1=b_2$, $b_2=y+2\pi$;

if $b_1>c_i$, then $y=b_2$, $b_2=b_1$, $b_1=y-2\pi$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$,

if $b_2\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_2$, if $b_2\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_2$.

Cosinus function:

$B\left[\cos x_i,c,\alpha\right]=$

$\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right)$, $b_1=\underline{x}_i$, $b_2=\overline{x}_i$,

$x=\arccos\left|\alpha\right|$, $per=2\pi\cdot$ int $(c_i/2\pi)$, if $c_i<0$, then $per=per-2\pi$,

if $\alpha\in[0,1]$, then $b_1=x+per$, $b_2=2\pi-x+per$;

if $\alpha\in[-1,0)$, then $b_1=\pi-x+per$, $b_2=\pi+x+per$;

if $b_2<c_i$, then $y=b_1$, $b_1=b_2$, $b_2=y+2\pi$;

if $b_1>c_i$, then $y=b_2$, $b_2=b_1$, $b_1=y-2\pi$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$,

if $b_2\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_2$, if $b_2\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_2$.

Tangent function:

$B\left[\tan x_i,c,\alpha\right]=$

$\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right)$, $b_1=\underline{x}_i$, $b_2=\overline{x}_i$,

$x=\arctan\left|\alpha\right|$, $per=2\pi\cdot$int $(c_i/2\pi)$, if $c_i<0$, then $per=per-2\pi$,

if $\alpha\geq 0$, then $b_1=x+per,b_2=\pi+x+per$;

if $\alpha<0$, then $b_1=\pi-x+per,b_2=2\pi-x+per$;

if $b_2<c_i$, then $b_1=b_2$, $b_2=b_1+\pi$;

if $b_1>c_i$, then $b_2=b_1$, $b_1=b_2-\pi$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$,

if $b_2\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_2$, if $b_2\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_2$.

Cotangent function:

$B\left[\cot x_i,c,\alpha\right]=$

$\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right)$, $b_1=\underline{x}_i$, $b_2=\overline{x}_i$,

$x=\operatorname{arccot}\left|\alpha\right|$, $per=2\pi\cdot$int $(c_i/2\pi)$, if $c_i<0$, then $per=per-2\pi$,

if $\alpha\geq 0$, then $b_1=x+per$, $b_2=\pi+x+per$;

if $\alpha<0$, then $b_1=\pi-x+per$, $b_2=2\pi-x+per$;

if $b_2<c_i$, then $b_1=b_2$, $b_2=b_1+\pi$;

if $b_1>c_i$, then $b_2=b_1$, $b_1=b_2-\pi$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$,

if $b_2\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_2$, if $b_2\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_2$.

Arcus sinus function:

$B\left[\arcsin x_i,c,\alpha\right]=$

$\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right)$, $b_1=\underline{x}_i$,

if $\alpha\in\left[-\pi/2,\pi/2\right]$, then $b_1=\sin\alpha$;

if $b_1\in\left(\underline{x}_i,c_i\right)$, then $\underline{x}_i=b_1$, if $b_1\in\left(c_i,\overline{x}_i\right)$, then $\overline{x}_i=b_1$.

Arcus cosinus function:

$B\left[\arccos x_i,c,\alpha\right]=$

$\left(\left(\underline{x}_1,\overline{x}_1\right),...,\left(\underline{x}_i,\overline{x}_i\right),...,\left(\underline{x}_m,\overline{x}_m\right)\right)$, $b_1=\underline{x}_i$,

if $\alpha\in\left[0,\pi\right]$, then $b_1=\cos\alpha$;

if $b_1 \in (\underline{x}_i, c_i)$, then $\underline{x}_i = b_1$, if $b_1 \in (c_i, \overline{x}_i)$, then $\overline{x}_i = b_1$.

Arcus tangent function:

$$B[\arctan x_i, c, \alpha] =$$

$$((\underline{x}_1, \overline{x}_1), ..., (\underline{x}_i, \overline{x}_i), ..., (\underline{x}_m, \overline{x}_m)),\ b_1 = \underline{x}_i,$$

if $\alpha \in (-\pi/2, \pi/2)$, then $b_1 = \tan\alpha$;

if $b_1 \in (\underline{x}_i, c_i)$, then $\underline{x}_i = b_1$, if $b_1 \in (c_i, \overline{x}_i)$, then $\overline{x}_i = b_1$.

Arcus cotangent function:

$$B[\operatorname{arc\,cot} x_i, c, \alpha] =$$

$$((\underline{x}_1, \overline{x}_1), ..., (\underline{x}_i, \overline{x}_i), ..., (\underline{x}_m, \overline{x}_m)),\ b_1 = \underline{x}_i,$$

if $\alpha \in (0, \pi)$, then $b_1 = \cot\alpha$;

if $b_1 \in (\underline{x}_i, c_i)$, then $\underline{x}_i = b_1$, if $b_1 \in (c_i, \overline{x}_i)$, then $\overline{x}_i = b_1$.

Now we have 6 rules for decomposition and 12 processes for elementary boxes. Since the application of the 6th rule (rule of composite function) requires the same investigation as the evaluation of elementary boxes, therefore the decomposition part of the program contains 17 different cases, and the evaluation part contains 12 cases. At the end of this section, let us emphasize some 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 decomposition of the expression $g(x)$, the determination of the elementary boxes, uses only the structure of the expression $g(x)$ and supposes only the continuity of g on D. 3) 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.

2. Numerically Coded Form of Expression

When the author introduced the above decomposition of expressions, the process was followed by a Maple program. This code computed a box $B(g, c, \alpha)$ very similarly as it is done in our numerical example. The program utilizes strongly the ability of Maple to give the type of an expression by the function $type(e, t)$ where e is an expression and t is a type name, furthermore to give operands from an expression by the function $op(i, e)$ where i is the position index and e is an expression. The use of our Maple program is very convenient because we must only give the function g (the expression $g(x)$) in a customary form. Unfortunately, the program is fairly slow. The conventional program languages (e.g. C++, Fortran) have no similar functions, but they can follow the decomposition, if a numerically coded form of the expression $g(x)$ is used. This form is built of triples of integer or real numbers. The first number in a triple is the so-called operation code (using the order of rules and elementary functions):

const. add. $\leftrightarrow 1$, const. mult. $\leftrightarrow 2$, $+ \leftrightarrow 3$, $/ \leftrightarrow 4$, $* \leftrightarrow 5$, power func. $\leftrightarrow 6$, exp $\leftrightarrow 7$, ln $\leftrightarrow 8$, abs. func. $\leftrightarrow 9$, sin $\leftrightarrow 10$, cos $\leftrightarrow 11$, tan $\leftrightarrow 12$, cot $\leftrightarrow 13$, arcsin $\leftrightarrow 14$, arccos $\leftrightarrow 15$, arctan $\leftrightarrow 16$, arccot $\leftrightarrow 17$.

The second number in the triples is minus i if an elementary function uses the argument x_i (the minus sign identifies the elementary funcions). If the operation uses a former triple as the first operand, then the ordinal number of this triple is the second number in the actual triple. The third number in the triples is the given constant or the exponent if the constant addition, constant multiplication or the power function is used, respectively. If the operation uses a former triple as a second operand (cases of $+, /, *$), then the ordinal number of this triple is the third number in the actual triple. If the operation is $\exp, \ln, ..., \operatorname{arccot}$ (the first number in the triple is between 7 and 17), then the third number in the actual triple is 0. To make the description of expressions by triples unique, we use the left-right rule in every choice. Exercise the description on the expression $g(x_1, x_2) = \sin(x_1 x_2) + x_1^2 / \ln x_2$. The four elementary functions used in the build-up of $g(x_1, x_2)$ (the elementary boxes belong to these functions after the decomposition) are $x_1 = (x_1)^1, x_2 = (x_2)^1, x_1^2, \ln x_2$. Their descriptions are:

$$x_1 = (6, -1, 1),\quad x_2 = (6, -2, 1),$$

$$x_1^2 = (6, -1, 2),\quad \ln x_2 = (8, -2, 0).$$

Now we have the sequence of triples:

$$(6, -1, 1), (6, -2, 1), (6, -1, 2), (8, -2, 0).$$

Using the elements of this sequence, the functions $x_1 x_2, \sin(x_1 x_2)$ and $x_1^2 / \ln x_2$ can be obtained in the form:

$$x_1x_2 = (5,1,2), \quad \sin(x_1x_2) = (10,5,0),$$
$$x_1^2 / \ln x_2 = (4,3,4).$$

Since the last operation can be written in form $(3,6,7)$, the complete description of $g(x_1,x_2)$ is:

$$\{(6,-1,1),\ (6,-2,1),\ (6,-1,2),\ (8,-2,0),$$
$$(5,1,2),\ (10,5,0),\ (4,3,4),\ (3,6,7)\}.$$

It can be seen that the preparation of numerically coded forms of expressions is easy enough, nevertheless the author has a short Maple program also for this one-off work. (But a numerically coded form can be used a thousand or a million times in an application.) The construction of our Maple program can be illustrated (on the former expression) by the **Table 1**.

The 1st column shows the index (serial number) of the expression appearing in the 2nd column. The 4th column contains the temporary triples which use the indexes of the 1st column for identification. Going downwards the temporary triples, the elementary tripes receive their new indexes, and going backwards, the other triples receive their new indexes (6th column). Using these new indexes the final triples (7th column) are obtained. Ordering them by the indexes of the 6th column the result is ready (8th column). This result differs from our previous one formally (here the left-right rule is omitted), but they are equivalent in essence. The arrays of Maple program x, com, tri, ii, res belong to vector x components of $g(x)$, triples, new indexes and result, respectively. The array v is an auxiliary vector. The complete code is as follows.

Table 1. Creation of numerically coded form

i	Components		Triples	⇓↑	ii	New triples	Result
1	$\sin(x_1x_2)+x_1^2/\ln x_2$	$\Rightarrow$	$(3,2,3)$	$\rightarrow$	8	$(3,7,6)$	$(6,-1,2)$
2	$\sin(x_1x_2)$	$\Rightarrow$	$(10,4,0)$	$\rightarrow$	7	$(10,5,0)$	$(8,-2,0)$
3	$x_1^2/\ln x_2$	$\Rightarrow$	$(4,5,6)$	$\rightarrow$	6	$(4,1,2)$	$(6,-1,1)$
4	x_1x_2	$\Rightarrow$	$(5,7,8)$	$\rightarrow$	5	$(5,3,4)$	$(6,-2,1)$
5	x_1^2	$\Rightarrow$	$(6,-1,2)$	$\Rightarrow$	1	√	$(5,3,4)$
6	$\ln x_2$	$\Rightarrow$	$(8,-2,0)$	$\Rightarrow$	2	√	$(4,1,2)$
7	x_1	$\Rightarrow$	$(6,-1,1)$	$\Rightarrow$	3	√	$(10,5,0)$
8	x_2	$\Rightarrow$	$(6,-2,1)$	$\Rightarrow$	4	√	$(3,7,6)$

```
DECLARATIONS
 > x:=array(1..10): com:=array(1..10): tri:=array(1..10,1..3):
 > ii:=array(1..10): res:=array(1..10,1..3): v:=array(1..3):
INITIAL VALUES
 > com[1]:=sin(x[1]*x[2])+x[1]^2 / ln(x[2]):
 > i:=0: imax:=1:
COMPONENTS and TRIPLES
 > while i <= imax do
 >       i:=i+1: g:=com[i]:
 >       if type(g,name) then v:=[0,op(1,g),0]: fi:
 >       if type(g,`+`) then
 >             if type(op(1,g),numeric) then v:=[1,g-op(1,g),op(1,g)]:
 >             elif type(g-op(1,g),numeric) then v:=[1,op(1,g),g-op(1,g)]:
 >             else v:=[3,op(1,g),g-op(1,g)]: fi: fi:
 >       if type(g,`*`) then
 >             if type(op(1,g),numeric) then v:=[2,g/op(1,g),op(1,g)]:
 >             elif type(g/op(1,g),numeric) then v:=[2,op(1,g),u/op(1,g)]:
 >             elif type(op(2,g),`^`) and op(2,op(2,g)) < 0 then v:=[4,op(1,g),op(2,g)^(-1)]:
 >             else v:=[5,op(1,g),g/op(1,g)]: fi: fi:
 >     if type(g,`^`) then v:=[6,op(1,g),op(2,g)]: fi:
 >     if type(g,function) then
 >             if op(0,g)=exp then v:=[7,op(1,g),0]: fi: if op(0,g)=ln then v:=[8,op(1,g),0]: fi:
 >             if op(0,g)=abs then v:=[9,op(1,g),0]: fi: if op(0,g)=sin then v:=[10,op(1,g),0]: fi:
 >             if op(0,g)=cos then v:=[11,op(1,g),0]: fi: if op(0,g)=tan then v:=[12,op(1,g),0]: fi:
 >             if op(0,g)=cot then v:=[13,op(1,g),0]: fi: if op(0,g)=arcsin then v:=[14,op(1,g),0]: fi:
 >             if op(0,g)=arccos then v:=[15,op(1,g),0]: fi:if op(0,g)=arctan then v:=[16,op(1,g),0]: fi:
 >             if op(0,g)=arccot then v:=[17,op(1,g),0]: fi: fi:
 >     if v[1]=0 then tri[i,1]:=6: tri[i,2]:=-v[2]: tri[i,3]:=1: fi:
 >     if v[1] > = 6 and v[1] < = 17 then
 >             if type(v[2],name) then    tri[i,1]:=v[1]: tri[i,2]:=-op(1,v[2]): tri[i,3]:=v[3]:
 >             else imax:=imax+1: com[imax]:=v[2]: tri[i,1]:=v[1]: tri[i,2]:=imax: tri[i,3]:=v[3]: fi: fi:
 >     if v[1]=1 or v[1]=2 then
 >             imax:=imax+1: com[imax]:=v[2]: tri[i,1]:=v[1]: tri[i,2]:=imax: tri[i,3]:=v[3]: fi:
 >     if v[1] > = 3 and v[1] < = 5 then
 >             imax:=imax+1: com[imax]:=v[2]: imax:=imax+1: com[imax]:=v[3]:
 >             tri[i,1]:=v[1]: tri[i,2]:=imax-1: tri[i,3]:=imax: fi:
```



```
 > od:
NEW INDEXES
 > ind:=1:
 > for i from 1 to imax do if tri[i,2] < 0 then ii[i]:=ind: ind:=ind+1: fi: od:
 > for i from imax by -1 to 1 do if tri[i,2] > 0 then ii[i]:=ind: ind:=ind+1: fi: od:
RESULT
 > for ind from 1 to imax do
 >     for i from 1 to imax do
 >         if ii[i]=ind then res[ind,1]:=tri[i,1]:
 >             if tri[i,2] < 0 then res[ind,2]:=tri[i,2]: res[ind,3]:=tri[i,3]: fi:
 >             if tri[i,2] > 0 then res[ind,2]:=ii[tri[i,2]]:
 >                 if tri[i,1]=3 or tri[i,1]=4 or tri[i,1]=5 then res[ind,3]:=ii[tri[i,3]]: else res[ind,3]:=tri[i,3]:
 >         fi: fi: fi:
 > od: od:
 > print(`The result`, res):
```

Now we have a suitable form of expressions to make a fast program for our original problem.

3. A Complete C++ Program for Computation of Solution Boxes

Here the function segment *solbox* is made for the computation of solution box $B[g,c,\alpha]$ where

$$g : D \subset \mathrm{R}^m \to \mathrm{R}, D = \left(\left(\underline{x}_1, \overline{x}_1\right), ..., \left(\underline{x}_m, \overline{x}_m\right)\right), c \in D, \alpha \in \mathrm{R}$$

have the former definitions. The headline of this segment contains the input parameters $D \leftrightarrow D, G \leftrightarrow g$ (in triple form), $c \leftrightarrow c, alp \leftrightarrow \alpha, m \leftrightarrow m, nt \leftrightarrow$ number of triples . The result $B\left[g,c,\alpha\right]$ (as output parameter) is placed into the array B. The segment contains 3 essential parts (great loops). In the first one the function values of components of $g(x)$ at point c are computed by using the triples in succession. These function values are placed into the array fv and will be used in the decomposition steps. In the second loop the decomposition (making elementary boxes) goes on. In the normal case ($nt > 1$), first the last triple (expression $g(x)$) and α are placed into the array of non-elementary expressions (ne). At this time the number of elements in this array is 1 ($ine = 1$). Applying the suitable rule for this element, new elements of the array ne or (and) those of the array of elementary expressions (el) are created. The values $\beta, \gamma,$ etc. appearing in the rules are denoted by $be, ga, de, ep,$ and s shows the number of new boxes. The indexes *ine* and *iel* show the numbers of elements in the arrays *ne* and *el*. The decomposition procedure is finished when there is no element for examination in the array *ne*, that is the necessary elementary expressions are all in the array *el*. In the third great loop of segment *solbox* the (*iel* pieces of) elementary boxes are evaluated and the array B is made. As it was mentioned earlier, the application of the composite function rule requires the same investigation as the evaluation of elementary boxes. Therefore the second and third loops of the segment contain similar lines (only some parameters are different) between case 6 and case 17.

```
#include <iostream.h>
#include <math.h>
double B[10][3];
/*DECLARATIONS*/
void solbox(double D[][3],double G[][4],double c[],double alp,int m,int nt)
{double fv[100],ne[100][5],el[100][5],al,be,ga,de,ep,x,y,w,alf,per;
int i,j,k,l,ii,jj,kk,ine,iel,s; const double Pi=3.14159265;
/*FUNCTION VALUES TO COMPONENTS OF EXPRESSION g(x)*/
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;}}
/*DECOMPOSITION BY USING THE 6 RULES*/
      for (i=1; i<=3; i++) ne[1][i]=G[nt][i]; ne[1][4]=alp; ine=1; iel=0; i=0;
      while (i<ine && nt>1)
            {i++; j=(int)ne[i][1]; k=(int)ne[i][2]; l=(int)ne[i][3]; w=ne[i][3]; al=ne[i][4]; s=0;
```



```
        switch (j)
            {case 1: be=al-w;s=1;break;
            case 2: be=al/w;s=1;break;
            case 3: be=(al+fv[k]-fv[l])/2;ga=al-be;s=2;break;
            case 4: if (al==0) {be=0;s=1;}; if (al !=0) {ga=(al*fv[k]+fv[l])/(1+al*al);be=al*ga;s=2;};break;
            case 5: if (al==0 || fv[k]*fv[l]*al < 0) {be=0;ga=0;s=2;}
                if (al!=0 && fv[k]==0 && fv[l]==0) {be=sqrt(fabs(al));ga=be;de=-be;ep=de;s=4;}
                if (al!=0 && fv[k] !=0 && fv[l]==0) {be=2*fv[k];ga=al/be;de=0;s=3;}
                if (al!=0 && fv[k]==0 && fv[l]!=0) {ga=2*fv[l];be=al/ga;de=be;ep=0;s=4;}
                if (fv[k]*fv[l]*al>0) {be=fv[k]/fabs(fv[k])*sqrt(al*fv[k]/fv[l]);
                    ga=fv[l]/fabs(fv[l])*sqrt(al*fv[l]/fv[k]);de=0;s=3;};break;
            case 6: if (w>0 && (l+1)/2*2==l+1) {be=pow(al,1/w); s=1;}
                if (w>0 && l/2*2==l && al>=0) {be=pow(al,1/w);ga=-be;s=2;}
                if (w<0 && (l-1)/2*2==l-1 && al !=0) {be=pow(al,1/w);s=1;}
                if (w<0 && l/2*2==l && al>0) {be=pow(al,1/w);ga=-be;s=2;}
                if (w>0 && w<1 && al>=0) {be=pow(al,1/w);s=1;};break;
            case 7: if (al > 0) {be=log(al);s=1;};break;
            case 8: be=exp(al);s=1;break;
            case 9: if (al >= 0) {be=al;ga=-al;s=2;};break;
            case 10:if (fabs(al)<=1)
                {x=asin(fabs(al));per=2*Pi*(int)(fv[k]/2/Pi); if(fv[k]<0) per=per-2*Pi;
                if (al>=0) {be=x+per;ga=Pi-x+per;}; if (al<0) {be=Pi+x+per;ga=2*Pi-x+per;}
                if (ga<fv[k]) {y=be;be=ga;ga=y+2*Pi;}; if (be>fv[k]) {y=ga;ga=be;be=y-2*Pi;}; s=2;};break;
            case 11:if (fabs(al)<=1)
                {x=acos(fabs(al));per=2*Pi*(int)(fv[k]/2/Pi); if(fv[k]<0) per=per-2*Pi;
                if (al>=0) {be=x+per;ga=2*Pi-x+per;}; if (al<0) {be=Pi-x+per;ga=Pi+x+per;}
                if (ga<fv[k]) {y=be;be=ga;ga=y+2*Pi;}; if (be>fv[k]) {y=ga;ga=be;be=y-2*Pi;}; s=2;};break;
            case 12:x=atan(fabs(al));per=2*Pi*(int)(fv[k]/2/Pi); if(fv[k]<0) per=per-2*Pi;
                if (al>=0) {be=x+per;ga=Pi+x+per;}; if (al<0) {be=Pi-x+per;ga=2*Pi-x+per;}
                if (ga<fv[k]) {be=ga;ga=be+Pi;}; if (be>fv[k]) {ga=be;be=ga-Pi;};s=2;break;
            case 13:x=Pi/2-atan(fabs(al));per=2*Pi*(int)(fv[k]/2/Pi); if(fv[k]<0) per=per-2*Pi;
                if (al>=0) {be=x+per;ga=Pi+x+per;}; if (al<0) {be=Pi-x+per;ga=2*Pi-x+per;}
                if (ga<fv[k]) {be=ga;ga=be+Pi;}; if (be>fv[k]) {ga=be;be=ga-Pi;};s=2;break;
            case 14:if (fabs(al)<=Pi/2) {be=sin(al);s=1;};break;
            case 15:if (al>=0 && al<=Pi) {be=cos(al);s=1;};break;
            case 16:if (fabs(al)<Pi/2) {be=tan(al);s=1;};break;
            case 17:if (al>0 && al<Pi) {be=1/tan(al);s=1;};break;}
        for (ii=1; ii<=s; ii++)
            {switch (ii)
                {case 1:alf=be;kk=k;break; case 2:alf=ga;if (j>=3 && j<=5) kk=l;break;
                case 3:alf=de;kk=k;break; case 4:alf=ep;if (j>=3 && j<=5) kk=l;break;}
            if (G[kk][2]<0) {iel++; for (jj=1; jj<=3; jj++) el[iel][jj]=G[kk][jj];el[iel][4]=alf;}
            if (G[kk][2]>0) {ine++; for (jj=1; jj<=3; jj++) ne[ine][jj]=G[kk][jj];ne[ine][4]=alf;}}}
    if (nt==1) {for (ii=1; ii<=4; ii++) el[1][ii]=ne[1][ii];iel=1;}
/*EVALUATING THE ELEMENTARY BOXES BY USING THE 12 PROCESSES*/
    for (i=1; i<=m; i++) {B[i][1]=D[i][1]; B[i][2]=D[i][2];}
    for (i=1; i<=iel; i++)
        {j=(int)el[i][1]; k=(int)-el[i][2]; l=(int)el[i][3]; w=el[i][3]; al=el[i][4]; s=0;
        switch (j)
            {case 6: if (w>0 && (l+1)/2*2==l+1) {be=pow(al,1/w);s=1;}
                if (w>0 && l/2*2==l && al>=0) {be=pow(al,1/w);ga=-be;s=2;}
                if (w<0 && (l-1)/2*2==l-1 && al!=0) {be=pow(al,1/w);s=1;}
                if (w<0 && l/2*2==l && al>0) {be=pow(al,1/w);ga=-be;s=2;}
                if (w>0 && w<1 && al>=0) {be=pow(al,1/w);s=1;};break;
            case 7: if (al > 0) {be=log(al);s=1;};break;
            case 8: be=exp(al);s=1;break;
            case 9: if (al >= 0) {be=al;ga=-al;s=2;};break;
            case 10:if (fabs(al)<=1)
                {x=asin(fabs(al));per=2*Pi*(int)(c[k]/2/Pi);if(c[k]<0) per=per-2*Pi;
                if (al>=0) {be=x+per;ga=Pi-x+per;}; if (al<0) {be=Pi+x+per;ga=2*Pi-x+per;}
                if (ga<c[k]) {y=be;be=ga;ga=y+2*Pi;}; if (be>c[k]) {y=ga;ga=be;be=y-2*Pi;}; s=2;};break;
            case 11:if (fabs(al)<=1)
                {x=acos(fabs(al)); per=2*Pi*(int)(c[k]/2/Pi);if(c[k]<0) per=per-2*Pi;
                if (al>=0) {be=x+per;ga=2*Pi-x+per;}; if (al<0) {be=Pi-x+per;ga=Pi+x+per;}
                if (ga<c[k]) {y=be;be=ga;ga=y+2*Pi;}; if (be>c[k]) {y=ga;ga=be;be=y-2*Pi;}; s=2;};break;
            case 12:x=atan(fabs(al));per=2*Pi*(int)(c[k]/2/Pi);if(c[k]<0) per=per-2*Pi;
                if (al>=0) {be=x+per;ga=Pi+x+per;}; if (al<0) {be=Pi-x+per;ga=2*Pi-x+per;}
                if (ga<c[k]) {be=ga;ga=be+Pi;}; if (be>c[k]) {ga=be;be=ga-Pi;};s=2;break;
            case 13:x=Pi/2-atan(fabs(al));per=2*Pi*(int)(c[k]/2/Pi);if(c[k]<0) per=per-2*Pi;
```



```
                    if (al>=0) {be=x+per;ga=Pi+x+per;}; if (al<0) {be=Pi-x+per;ga=2*Pi-x+per;}
                    if (ga<c[k]) {be=ga;ga=be+Pi;}; if (be>c[k]) {ga=be;be=ga-Pi;};s=2;break;
                case 14:if (fabs(al)<=Pi/2) {be=sin(al);s=1;};break;
                case 15:if (al>=0 && al<=Pi) {be=cos(al);s=1;};break;
                case 16:if (fabs(al)<Pi/2) {be=tan(al);s=1;};break;
                case 17:if (al>0 && al<Pi) {be=1/tan(al);s=1;};break;}
            for (ii=1; ii<=s; ii++) { alf=be; if (ii==2) alf=ga;
                if (alf < c[k] && alf > B[k][1]) B[k][1]=alf;
                if (alf > c[k] && alf < B[k][2]) B[k][2]=alf;}}}
/*THE CALLING SEGMENT WITH THE SAMPLE EXPRESSION*/
void main()
    {double D[10][3],G[100][4],c[10],alp; int m,nt,i,j;
    double g[][3]={{6,-1,1},{6,-2,1},{6,-1,2},{8,-2,0},{5,1,2},{10,5,0},{4,3,4},{3,6,7}};
    D[1][1]=0.;D[1][2]=3.141593;D[2][1]=1.;D[2][2]=4.; c[1]=1.;c[2]=2.; alp=1.; m=2; nt=8;
    for (i=1; i<=nt; i++) {for (j=1; j<=3; j++) G[i][j]=g[i-1][j-1];}; solbox(D,G,c,alp,m,nt);
    for (i=1; i<=m; i++) {cout << B[i][1] << " " << B[i][2] << endl;}}
```

The calling (main) segment uses a simple trick (push down of indexes) for the easy handling of triple forms. Using the form (produced by the Maple program)

$$\{(6,-1,2),(8,-2,0),(6,-1,1),(6,-2,1),$$
$$(5,3,4),(4,1,2),(10,5,0),(3,7,6)\}$$

and the form (produced by us)

$$\{(6,-1,1),(6,-2,1),(6,-1,2),(8,-2,0),$$
$$(5,1,2),(10,5,0),(4,3,4),(3,6,7)\}$$

of our sample expression $g(x_1,x_2)$, the result is the same:

$$B\left[\sin(x_1x_2)+x_1^2/\ln x_2,(1,2),1\right]=$$
$$\left((0.839584,1.20543),(1,2.41086)\right),$$

which was given (with rounded values) in the first section.

4. A numerical Example for Illustration

The surfaces given by formulas

$$x_3=2\sin(x_1x_2)+2,\quad x_3=x_1^2+x_2^2-2$$

have their intersection over the two dimensional interval $\left((-3,3),(-3,3)\right)$. Maple 11 can produce **Figure 1** with these data. Our aim is to give an approximating value for the volume of the section set S determined by the surfaces.

Since the set S is the solution set of the system of inequalities:

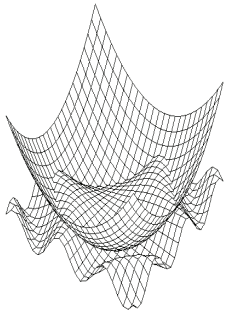

Figure 1. Intersection

$$2\sin(x_1x_2)-x_3+2\ge 0,\quad x_3-x_1^2-x_2^2+2\ge 0,$$

one of methods of [3] can be used. This method searches for disjunct open solution boxes of an inequality system and it is based on the following four principles. 1) 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. 2) If B_1 and B_2 are m-dimensional boxes, then the set B_1-B_2 can be divided into (at most) $2m$ boxes easily. In this way a sequence of solution boxes and another sequence of boxes waiting for examination can be generated. 3) A boundary region is used to prevent the appearance of too small boxes. Here this region is defined by the inequality $\left\|f(x)\right\|_\infty<10^{-3}$ and a cube with edges of 10^{-3} is excluded at a region point c. 4) The most promising box (the box with maximum volume) is chosen from boxes waiting for examination in every step. The essential input data are:

$$f_1(x)=2\sin(x_1x_2)-x_3+2=$$
$$\{(6,-1,1),(6,-2,1),...,(3,6,7),(1,8,2.)\},$$
$$f_2(x)=x_3-x_1^2-x_2^2+2=$$
$$\{(6,-1,2),(6,-2,2),...,(3,3,5),(1,6,2.)\},$$

the initial box $I=\left((-3,3),(-3,3),(-2,4)\right)$ which includes the set S, and the number N_b of solution boxes looked for, as stop criteria. For the values $N_b=10^4,\ 5\cdot10^4,\ 10^5$ the results are

$$27.0008,\quad 28.1484,\quad 28.4231,$$

respectively. The running times are (with the original Lahey Fortran 90 version 4.5 code on a PC of two 2 GHz processor):

$$2\,\text{sec},\quad 26\,\text{sec},\quad 107\,\text{sec},$$

respectively. The fast increase of running time shows that neither the 3rd nor the 4th principle work well if N_b is

large. (Only a few hundred solution boxes were searched for in [3].) Now consider another possibility. Since

$$vol(S) = \int\int\int_S 1 \, dxdydz,$$

and [4,5] contain methods for computation of integrals, perhaps better results can be obtained. The method of [5] is based on the following three principles. 1) The value of a (simple, double, triple, etc.) integral can be given by the volume of a suitable set T. If the box I contains this set, then a good scanning of T gives an approximation of the integral value and the scanning of the set $I-T$ also facilitates computation of an error bound. Here the set T can be described by the system of inequalities

$$2\sin(x_1x_2) - x_3 + 2 \geq 0,\ x_3 - x_1^2 - x_2^2 + 2 \geq 0,\ 1 - x_4 \geq 0,$$

and the scanning works by computing solution boxes. The scanning of the set $I-T$ is based on the examination of "the worst inequality" at points $c \in I-T$. 2) If B_1 and B_2 are m-dimensional boxes, then the set $B_1 - B_2$ can be divided into (at most) $2m$ boxes easily. 3) The too small boxes are filtered by the simple condition $vol(B) > \kappa$. Naturally, the value κ has a strong influence on the available error bound. The program is stopped when there is no more box waiting for examination. Now the essential input data are: the expressions $f_1(x),\ f_2(x),\ f_3(x)$ in triple form, the initial box $I = ((-3,3),(-3,3),(-2,4),(0,1))$ which includes the set T, and the filter parameter κ. For the values $\kappa = 10^{-5},\ 10^{-6},\ 10^{-7}$, the results are

$$28.9953 \pm 1.0169,\ 29.0499 \pm 0.4840,\ 29.0705 \pm 0.2294,$$

respectively. The running times are (with the original Visual C++ version 6.0 code on the former PC of two 2 GHz processor):

$$4\,\text{sec},\ 16\,\text{sec},\ 79\,\text{sec},$$

respectively. Since the difference between the third and first results is 0.0752 only, the third error bound suggests that all three of the results are good enough approximations. The extra work of the second and third cases was spent mostly on improving the error bound. Now make a goal-oriented transformation of the method of [3]. If the third principle (boundary region) and the fourth principle (most promising box) are omitted and principles of method of [5] belonging to error bound and filter parameter κ are used, then the new method will be competitive. For the values $\kappa = 10^{-5},\ 10^{-6},\ 10^{-7}$, the results are

$$29.0862 \pm 0.6765,\ 29.0829 \pm 0.3124,\ 29.0826 \pm 0.1447,$$

respectively. The running times are (with the new Lahey Fortran code):

$$1\,\text{sec},\ 5\,\text{sec},\ 24\,\text{sec},$$

respectively. These results are better from all points of view than the previous ones were. Since now the difference between the first and third results is 0.0036 only, the 3 values suggest much more accurate approximation than the guaranteed error bounds do. On the other hand, by the geometrical meaning, here the solution (section) set S (and its complementary set) was covered with cuboids, while the solution set T (and its complementary set) was covered with four-dimensional (abstract) boxes at the triple integral. It is not surprising that the computational effort increases with dimension. Finally some remarks are given.

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 $(B(g,c,\alpha)) \approx 10 *$ effort $(g(c))$, with respect to all three programming languages mentioned and problems of [1,3-5].

2) The speeds belonging to the evaluation of $B(g,c,\alpha)$ satisfy the relations: speed(C++) $\approx$ $200 *$ speed(Maple), speed(Fortran) $\approx$ $300 *$ speed(Maple) for the examples of [1]. Nevertheless, the Fortran codes are hardly faster than the C++ ones (differences of only 0-10 percent in running time) for the examples of [5].

3) We always used floating point arithmetic for computing solution boxes (zone function values by former name). Since these boxes are computed by lower estimates (see proofs of [1]), we have never happened to obtain a faulty result because of the effect of rounding errors.

4) The author would gladly send the C++, Fortran, Maple codes mentioned to interested readers in e-mail as an attached file.

REFERENCES

[1] F. Kálovics, "Creating and Handling Box Valued Functions Used in Numerical Methods," *Journal of Computational and Applied Mathematics*, Vol. 147, No. 2, 2002, pp. 333-348.

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

[3] 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.

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

[5] F. Kálovics, "Two Improved Zone Methods," *Miskolc Mathematical Notes*, Vol. 8, No. 2, 2007, pp. 169-179.

