Interval Analysis and Verification of Mathematical Models

Tibor Csendes

University of Szeged, Institute of Informatics, csendes@inf.u-szeged.hu

Summary. The paper provides an introduction to interval arithmetic based techniques for the verification of mathematical models. Illustrative examples are described from the fields of circle packing, chaotic behaviour dynamical systems, and process network synthesis.

1 Interval arithmetic

Model verification, optimization, and especially global optimization is sensitive on the reliability of the numerical computations. There exist practical problems where good approximative solutions are more or less accepted as the true solutions. Still there remain important application fields where the guaranteed reliability of the provided solution is of ample importance. The uncertainties are mostly caused by the rounding errors. These are necessarily part of the calculations when the algorithms are coded with floating point arithmetic – which allows quick computation. To provide a remedy for these problems, we shall apply interval arithmetic based inclusion functions [AH83, RR84]. These offer a theoretically reliable and computationally tractable means of locating a feasible suboptimal interval.

Denote the real numbers by $x, y, \ldots$, the set of compact intervals by $\mathbb{I} := \{[a,b] \mid a \leq b,\ a,b \in \mathbb{R}\}$, and the set of n-dimensional intervals (also called simply intervals or boxes) by $\mathbb{I}^n$. Capital letters will be used for intervals. For real vectors and interval vectors the notations

$$x = (x_i), \quad x_i \in \mathbb{R}, \qquad \text{and} \qquad X = (X_i), \quad X_i \in \mathbb{I}$$

are applied, respectively.

Definition 1. *A function $F : \mathbb{I}^n \to \mathbb{I}$ is an* inclusion function *of the real function f if for $\forall Y \in \mathbb{I}^n$ and $\forall y \in Y \;\; f(y) \in F(Y)$.*

In other words, $f(Y) \subseteq F(Y)$ where $f(Y)$ is the range of f over Y. We assume that f is continuous, then $f(Y)$ is an interval. The lower and upper bounds of an interval $Y \in \mathbb{I}^n$ are denoted by $\underline{Y}$ and $\overline{Y}$, respectively, and the inclusion function of the gradient of $f(x)$ by $\nabla F(X)$. The width of an interval is $w(Y) = \overline{Y} - \underline{Y}$, and $w(Y) = \max_i(\overline{Y_i} - \underline{Y_i})$ if $Y \in \mathbb{I}^n$ is an n-dimensional interval vector (also called a box). The midpoint of the interval X is defined by $m(X) = (\underline{X} + \overline{X})/2$ if $X \in \mathbb{I}$, and $m(X) = (m(X_i))$, if $X \in \mathbb{I}^n$. $\mathbb{I}(X)$ stands for all intervals in X. Three important possible properties of inclusion functions are:

Definition 2. *F is said to be an* isotone inclusion function *over $X \subseteq \mathbb{R}^n$ if $\forall Y, Z \in \mathbb{I}(X)$, $Y \subseteq Z$ implies $F(Y) \subseteq F(Z)$.*

Definition 3. *We call the inclusion function F an* α-convergent inclusion function *over X if for $\forall Y \in \mathbb{I}(X)$ $w(F(Y)) - w(f(Y)) \leq Cw^{\alpha}(Y)$ holds, where α and C are positive constants.*

Definition 4. *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 $w(Z_i) \to 0$.*

An inclusion function obviously provides more information over an interval than could be conveyed with independent real function evaluations. The inclusion function gives upper and lower bounds on the objective function over the specified interval.

There are several ways to build an inclusion function (e.g. by using the Lipschitz constant, if it is known). Interval arithmetic [AH83, HH93, H92, RR88] is a convenient tool for constructing the inclusion functions. This can be done for almost all functions that can be calculated by a finite algorithm (i.e., not only for given expressions).

The idea of interval calculations is to extend the basic operations and the elementary functions from the real numbers to intervals. Finding the range for a function over an n-dimensional interval has in general the same complexity as an optimization problem, because we have to find the extreme values of the function over the interval. By using interval arithmetic it is possible to find bounds on the function values more efficiently. The interval operations can be carried out using only real operations. For the argument intervals $[a, b]$ and $[c, d]$ the following expressions hold:

$$\begin{aligned}
[a,b] + [c,d] &= [a+c, b+d] \\
[a,b] - [c,d] &= [a-d, b-c] \\
[a,b] \ * [c,d] &= [\min\{ac, ad, bc, bd\}, \max\{ac, ad, bc, bd\}] \\
[a,b] \ / \ [c,d] &= [a,b] * [1/d, 1/c] \quad \text{if} \quad 0 \notin [c,d].
\end{aligned}$$

As an example, consider the range of $x - x^2$: it is $[-2, 0.25]$ on the interval of $[0, 2]$. In contrast to that, the above interval arithmetic will provide the inclusion of $[-4, 2]$, which is much wider. This too conservative estimation can be improved at the cost of more computation with sophisticated numerical techniques.

If outwardly directed rounding is also applied, then the interval calculated by a computer contains every real number that can be a result of the given function on real numbers inside the original intervals. The technique of producing an inclusion function by replacing the real variables and operations by their interval equivalent is called *natural interval extension* [AH83, RR84].

Natural interval extension provides an isotone inclusion function, that is α-convergent with $\alpha = 1$, and hence it has also the zero convergence property. More sophisticated inclusion functions, such as the centered forms provide quadratic convergence as interval width approaches zero to tight bounds, but they are not necessarily isotone inclusion functions. The computational cost of the inclusions for higher derivatives, Taylor models, or slopes is certainly high, but this can pay off when solving difficult optimization problems. In the present work we apply natural interval extension, and inclusions of the gradient and the Hessian for the monotonicity and concavity tests, and also within the interval Newton step.

Applying automatic differentiation or differentiation arithmetic in connection with interval arithmetic [HH93], we are also able to compute the inclusion function for the gradient. Automatic differentiation combines the advantages of symbolic and numerical differentiation and handles numbers instead of symbolic formulas. The computation of the gradient is done automatically together with the computation of the function value. The main advantage of this process is that only the algorithm or formula for the function is required. No explicit formulas for the gradient are needed.

Many programming languages are now available that support interval datatypes and automatic differentiation with the corresponding operations and intrinsic functions [BR87, J92, KK92, KK93]. Matlab also has an interval extension package called Intlab [R99]. These programming environments provide a convenient access to inclusion functions (with automatic outward rounding), but one can also simulate interval operations and functions by subroutines in any algorithmic language. We used the natural interval extension to calculate the inclusion functions. For more information about inclusion functions and interval arithmetic, see [AH83, RR84].

An often heard question is, how can we characterize the maximal size or difficulty of problems that still can be solved by interval inclusion function based methods. The short answer is that the dimension of a problem is a wrong measure, since low dimensional problems can be hopeless, and larger dimensional ones can be solved in a short time. For interval techniques the most dangerous is the large excess width, a bad estimation of the range of the related function on the studied intervals. It is most affected by the dependency problem, that is caused by multiple appearances of the same variable in a complex expression. According to this, the rule of thumb says that in case all

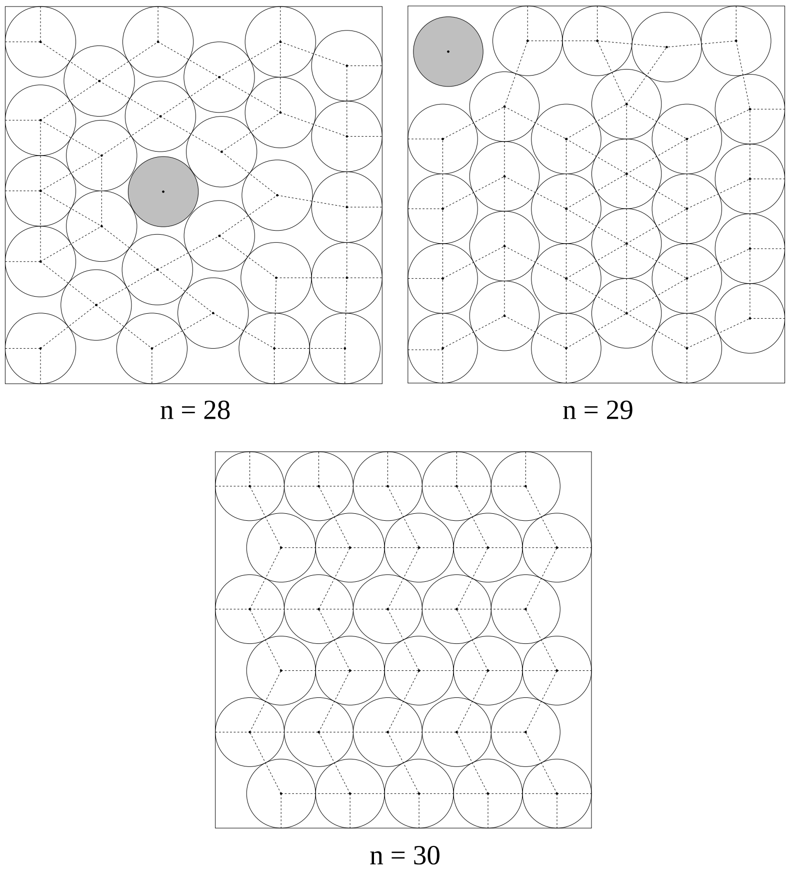

Fig. 1. The proven optimal circle packings into the unit square for 28, 29, and 30 circles.

involved variables appear only a few times in the expression of the objective and constraint functions, then the overestimation will be small, and the optimization algorithms can be successful even for larger dimensional problems (the number of variables can be up to 100).

A telling example for the capabilities of interval optimization methods is the results on circle packing problems. Mihály Csaba Markót [M04, MC05] was able to solve the problem cases of $n = 28, 29$, and 30, i.e. to find the configuration of n congruent nonoverlapping maximal circles fitting into the unit square. These problems were held before as hopeless, since the expected CPU time necessary for their solutions with the last available techniques were estimated to be around decades. The problems have 56, 58, and 60 variables, respectively, and hundreds of nonlinear constraints. The difficulty of the problems is highlighted by the facts that (due to obvious geometrical reasons) there are an astronomical number of equivalent, symmetric optimal solutions, and in the cases of $n = 28, 29$ there exist positive measure sets of global optimizer points.

Standard interval optimization algorithms could not solve the problems. A careful problem decomposition, and custom made built in acceleration devices based on the understanding of the problem structure enabled the successful solution. The running times were below 3 days, and ca. one million subintervals were stored during the solution process. The uncertainty in the position of the circles have been decreased by more than 700 orders of magnitude in each case. Due to the controlled outward rounding mode applied, the obtained results are reliable even in the sense of a rigorous mathematical theorem.

To demonstrate the capabilities of interval based computational methods, the following sections discuss example application in the fields of tolerance optimization and chaos verification for dynamic systems.

2 Tolerances in optimization and constraint satisfaction

Consider the nonlinear optimization problem (P)

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to } g_j(x) \leq 0 & \qquad j = 1, 2, \ldots, m, \end{array}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and the constraint functions $g_j(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ are continuous nonlinear functions, and n is the dimension of the problem. Let us denote the set of feasible points by A, that is $A := \{x \in \mathbb{R}^n : g_j(x) \leq 0 \text{ for each } j = 1, 2, \ldots, m\}$. Also let x^* be an optimal solution for problem (P). Notice that we restrict now our investigations to continuous functions.

It may happen that the optimal solution x^*, or an approximation of it, is known, yet this result is not suitable for practical use. For example, consider an engineering design problem which is formulated as a constrained global optimization problem, see for example those studied in [KF93], and [ZG92]. It is possible that the optimal design cannot be reproduced exactly with current manufacturing processes, thus each variable has a specific manufacturing tolerance, $\delta > 0$.

Since the optimal solution x^* may be on one or more active constraints, then the n-dimensional interval $[x_i^* - \delta, x_i^* + \delta]$ for $i = 1, 2, \ldots, n$ is not feasible. From a practical point of view, it is preferable to find a feasible suboptimal box instead of a single optimal point. Thus we seek a feasible n-dimensional interval X^* for which $g_j(x) \leq 0$, $j = 1, 2, \ldots, m$, for all $x \in X^*$. It is also desirable to have this feasible box as close to the optimum as possible. Thus we also impose the constraint $f(x) \leq f(x^*) + \varepsilon$ for some $\varepsilon > 0$ for all $x \in X^*$. Such an interval would also reflect the sensitivity of the objective function [F76] because the size of the feasible box may vary as ε varies.

We restate our problem: find an n-dimensional interval X^* such that for all $x \in X^*$

$$f(x) \leq f_\varepsilon \equiv f(x^*) + \varepsilon, \quad \text{and} \tag{1}$$
$$g_j(x) \leq 0 \qquad \text{for } \; j = 1, 2 \ldots, m. \tag{2}$$

Methods discussed in earlier papers [C89, C90, CP93] study similar problems. In [C89, C90] an interval method was introduced to find a bounding interval of the level set of an unconstrained nonlinear optimization problem. The algorithm converges to the smallest n-dimensional interval containing the specified level set. Another technique [CP93] locates the boundary of a level set in a given direction. Kearfott discussed an interval branch and bound method for bound constrained global optimization [K92]. In the study [RR93] the solution of a difficult nonlinear optimization problem (in an alternate form) was reported by using a customized interval subdivision scheme. These methods provide a guaranteed reliable solution, although they are computationally tractable [C98]. They are based on inclusion functions and interval arithmetic, which was shortly reviewed in the previous section.

2.1 The algorithm and its convergence

The suggested algorithm iteratively grows a box about a given seed point. A seed point x^{seed}, which lies interior to the region of feasibility and in the ε-level set must be provided to start the algorithm. Thus the seed point, x^{seed}, must satisfy the following conditions:

$$f(x^{\text{seed}}) < f_\varepsilon \;\; \text{and} \tag{3}$$
$$g_j(x^{\text{seed}}) < 0 \;\; \text{for each } \; j = 1, 2, \ldots, m. \tag{4}$$

This will imply that there exists a feasible box with a positive volume containing the seed point. It is possible to construct nonlinear optimization problems for which no proper seed point exists which would satisfy (3) and (4). In such cases the search for feasible suboptimal boxes makes no sense.

Seed points can be obtained in several ways. One is to find an approximation of x^*, and if it is interior to the feasible region, use it as x^{seed}. If it lies on an active constraint, search along the normal of the active constraint to generate x^{seed}. Another way is to sample randomly (a normal distribution may be appropriate) around the optimal point until a feasible interior point is found. This may also be used as x^{seed}. A slight variation would be to sample according to a uniform distribution in the interval hull of the feasible region intersected with the ε-level set. The first feasible point with objective function value less than f_ε would then be used as x^{seed}.

We will call an interval X a *feasible interval around* x^{seed}, if $x^{\text{seed}} \in X$ and equations (1) and (2) are satisfied for all $x \in X$. An interval will be called *maximal regarding* x^{seed}, if it is a feasible interval around x^{seed}, and there is no other feasible interval that contains it. Note that there may exist many maximal feasible boxes around a seed point. Also note that two different seed points may be contained in the same maximal feasible box.

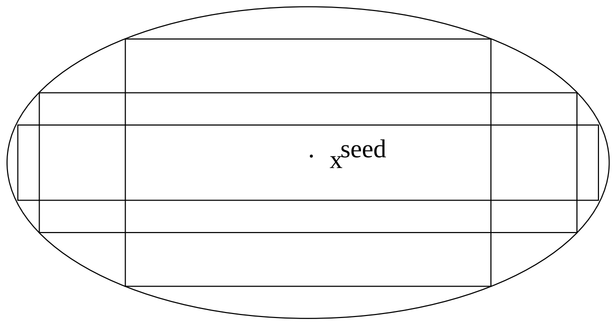

Fig. 2. Illustration of multiple maximal feasible boxes within the ε-level set (without constraints).

It may appear disturbing that there is no one-to-one relationship between x^{seed} and a maximal feasible box around x^{seed}. For an example, suppose the set of feasible points with objective function values less than or equal to f_ε is an ellipse. Suppose x^{seed} is interior to the ellipse. Then there is an infinite number of feasible boxes around x^{seed} which do not contain one another (see Fig. 2). At some point, it may be interesting to find the maximal feasible box around x^{seed} that has the largest volume. However at this point the algorithm does not find the largest volume box around a seed point, but simply a maximal feasible box. In our application to manufacturing tolerances this is not a disadvantage. It is more desirable to compare trade-offs between coordinate lengths of various maximal boxes, than it is to know the box with largest volume, as this has no direct meaning for the application.

We call an interval Y *strongly feasible*, if $f(y) < f_\varepsilon$, and $g_j(y) < 0$ for all $y \in Y$ $(j = 1, 2, \ldots, m)$.

The main algorithm, Algorithm 1 is presented below. The algorithm uses parameters $d(i,1)$ and $d(i,2)$ for $i = 1, 2, \ldots, n$ and η, which are set at the beginning to positive reals. To start, $d(i,1)$ and $d(i,2)$ must be larger than η. The stopping criterion indicates that the algorithm should stop increasing the size of the actual box X, when the change along each coordinate is less than the threshold η in all directions.

The core of the algorithm is the checking procedure, Algorithm 2 called in Steps 3 and 5 of Algorithm 1. This is a version of the interval subdivision method modified to check whether the actual box Y lies entirely in the region satisfying equations (1) and (2). The parameter θ is set to a small positive real value. It is better when the relation $d(i,j) \geq \theta$ holds (else the interval Y can be quickly rejected in Step 1 of Algorithm 2). The checking procedure is defined in detail in Algorithm 2.

If the checking routine indicates, that the checked interval is not strongly feasible, this means more precisely that a very small not strongly feasible

Algorithm 1 Main algorithm

```
0. Initialize interval vector $X_i = [x_i^{\text{seed}}, x_i^{\text{seed}}]$, and $d(i,j) \geq \eta > 0$ for all $i = 1, 2, \ldots, n$ and $j = 1, 2$.
1. For $i = 1$ to $n$ do:
2. Set $Y_j = X_j$ for $j = 1, 2, \ldots, n$; $j \neq i$, and
   $$Y_i = [(\underline{X}_i) - d(i,1), (\underline{X}_i)].$$
3. Use the checking routine to check whether $f(y) < f_\varepsilon$ and $g_j(y) < 0$ $(j = 1, 2, \ldots, m)$ for each $y \in Y$. If the answer is yes, then set $X = X \cup Y$. Otherwise $d(i,1) = ((\underline{X}_i) - (\overline{Z}_i))/2$, where $Z$ is the interval passed back by the checking routine as not strongly feasible.
4. Set $Y_j = X_j$ for $j = 1, 2, \ldots, n$; $j \neq i$, and
   $$Y_i = [(\overline{X}_i), (\overline{X}_i) + d(i,2)].$$
5. Use the checking routine to check whether $f(y) < f_\varepsilon$ and $g_j(y) < 0$ $(j = 1, 2, \ldots, m)$ for each $y \in Y$. If the answer is yes, then set $X = X \cup Y$. Otherwise $d(i,2) = ((\underline{Z}_i) - (\overline{X}_i))/2$, where $Z$ is the interval passed back by the checking routine as not strongly feasible.
6. End of i-loop
7. Stopping criterion: if the number of inclusion function calls is less than 100,000, and there is an $i = 1, 2, \ldots, n$ such that either $d(i,1) \geq \eta$ or $d(i,2) \geq \eta$ then go to **Step 1**.
8. Print $X$, and STOP.
```

Algorithm 2 Checking routine

```
0. Initialize the list $L$ to be empty.
1. If the width of $Y$ is less than $\theta$, then go to **Step 7**.
2. Evaluate the inclusion functions $F(Y)$ and $G_j(Y)$ for each $j = 1, 2, \ldots, m$.
3. If $\overline{F}(Y) \geq f_\varepsilon$ or $\overline{G}_j(Y) \geq 0$ for any $j = 1, 2, \ldots, m$, then go to **Step 5**.
4. If the list $L$ is empty, then go to **Step 6**, else put the last item of the list $L$ into $Y$, delete this item from the list, and go to **Step 1**.
5. Subdivide $Y$ into subintervals $U$ and $V$, set $Y = U$, put $V$ into the list $L$ as the last member, and go to **Step 1**. The subdivision should be made, such that the largest side of $Y$ is halved.
6. RETURN that the checked interval was strongly feasible.
7. RETURN $Z = Y$, and the message that it could not been proved that the checked interval is strongly feasible.
```

subinterval was found. By properly setting θ, the place where the strong feasibility is violated can be located.

Convergence results

In this subsection the convergence properties of the algorithm introduced are characterized to provide theoretical background for the numerical implementations. The proofs of the statements can be found in [CK95] and [C07]. First the checking routine is studied.

Lemma 1. *1. If the checking routine accepts an interval Y as strongly feasible, then $f(x) < f_\varepsilon$, and $g_j(x) < 0$ for each $x \in Y$, and $j = 1, 2, \ldots, m$.*

2. If the checking routine rejects an interval Y as not strongly feasible, then there exists a nested set of intervals, $Y = Y^1 \supset Y^2 \supset \ldots \supset Y^p$ generated by the routine, with the smallest interval having width less than θ, such that for each Y^i in the nested set of intervals one of the conditions max $F(Y^i) < f_\varepsilon$ or max $G_j(Y^i) < 0$ was violated, where $i = 1, 2, \ldots, p$ and $j = 1, 2, \ldots, m$.

Only the inclusion property of F and G_j was utilized in the proof of Lemma 1, and no further requirement (like isotonicity or convergence order of the inclusion functions involved) was necessary.

Lemma 2. *Assume that*

$$w(F(X)) \to 0 \text{ as } w(X) \to 0, \text{ and} \tag{5}$$

$$w(G_j(X)) \to 0 \text{ as } w(X) \to 0 \tag{6}$$

for all $j = 1, 2, \ldots, m$, i.e., F and G_j are zero convergent.

1. If $f(x) < f_\varepsilon$ and $g_j(x) < 0$ for every $x \in Y$ and $j = 1, 2, \ldots, m$, then there exists a threshold value $\theta^{\mathrm{T}} > 0$ such that for all θ: $0 < \theta < \theta^{\mathrm{T}}$ the checking routine stops after a finite number of iteration steps and it states that Y is strongly feasible.

2. If $\theta > 0$ and there is a point $x \in Y$ such that $f(x) \geq f_\varepsilon$ or $g_j(x) \geq 0$ for any $j = 1, 2, \ldots, m$, then the checking routine will stop after a finite number of iteration steps and it states that Y is not strongly feasible.

Notice that Lemma 2 ensures that a not strongly feasible interval Y will always be detected in a finite number of steps. However, it is possible that a strongly feasible interval will be mistaken as not strongly feasible if θ is too large. Thus it is important that θ be chosen with care.

Consider, again, a fixed constrained nonlinear optimization problem (P) as given at the beginning of the present section. Denote the result box calculated with the algorithm parameters θ and η by $X^*_{\theta,\eta}$, and the level set belonging to the function value f_ε by S_{f_ε}. Denote the vector of the G_j functions by G. The following theorems characterize the convergence properties of our main algorithm, Algorithm 1 for the obvious case when the stopping criterion for the number of function calls is deleted.

Theorem 1. *If the set $S_{f_\varepsilon} \cap A$ is bounded, the seed point x^{seed} fulfils the conditions (3) and (4), and the properties (5) and (6) hold for the inclusion functions $F(X)$ and $G(X)$, then there exist suitable $d(i,j) > 0$ ($i = 1,2,\ldots,n; j = 1,2$) values and threshold values $\theta^{\mathrm{T}} > 0$ and $\eta^{\mathrm{T}} > 0$ such that for all θ: $0 < \theta < \theta^{\mathrm{T}}$ and η: $0 < \eta < \eta^{\mathrm{T}}$*

1. *the algorithm stops after a finite number of steps,*
2. *the result box $X^*_{\theta,\eta}$ has a positive measure, and*
3. *the result interval $X^*_{\theta,\eta}$ is strongly feasible, $X^*_{\theta,\eta} \subset S_{f_\varepsilon} \cap A$.*

The strong feasibility of the accepted intervals was utilized only in proving the positive volume of the result intervals. With the exception of this, the convergence results remain valid if the checking routine accepts feasible intervals.

Theorem 2 describes the limit of the result boxes when the algorithm parameters θ and η are equal and converge together to zero.

Theorem 2. *Let the $d(i,j)$ positive values be fixed. If the conditions of Theorem 1 are fulfilled, then each accumulation interval X^* of the interval sequence $\{X^*_{\theta,\theta}\}_{\lim_{\theta\to 0}}$ is maximal in the sense that for every box X' the relations $X^* \subseteq X'$ and $X' \subseteq S_{f_\varepsilon} \cap A$ imply $X' = X^*$.*

The limiting interval X^* is not necessarily strongly feasible. For example, if $S_{f_\varepsilon} \cap A$ is an n-dimensional interval, then this not strongly feasible interval may be a limiting interval of a sequence of strongly feasible result intervals.

Theorem 1 suggests that for a problem satisfying its conditions, sufficiently small positive θ and η values ensure a positive measure result interval in a finite number of iteration steps, i.e., after a finite number of objective and constraint function calls. Theorem 2 gives the basis that with θ and η values close to the machine precision one may obtain a closely maximal result box. It has to be stressed that beyond the given algorithm many others can be given for the same problem, and that it is a very difficult problem to find a maximal volume feasible interval (equivalent to a global optimization problem cf. [C90]). In general, the location of a maximal volume feasible interval can only be solved with a certain kind of backtracking.

2.2 Numerical testing and examples

Consider the following simple constrained quadratic problem to illustrate how the algorithm discussed above proceeds. Let

$$f(x) = x_1^2 + x_2^2,$$

$$g_1(x) = (3 - x_1)^2 + (3 - x_2)^2 - 18, \text{ and}$$

$$g_2(x) = 1 - (2 - x_1)^2 - (2 - x_2)^2.$$

The set of feasible points A is now the circle C_1 with center at $(3,3)$ and with a radius of $3\sqrt{2}$ with the exceptions of the points of the circle C_2 with

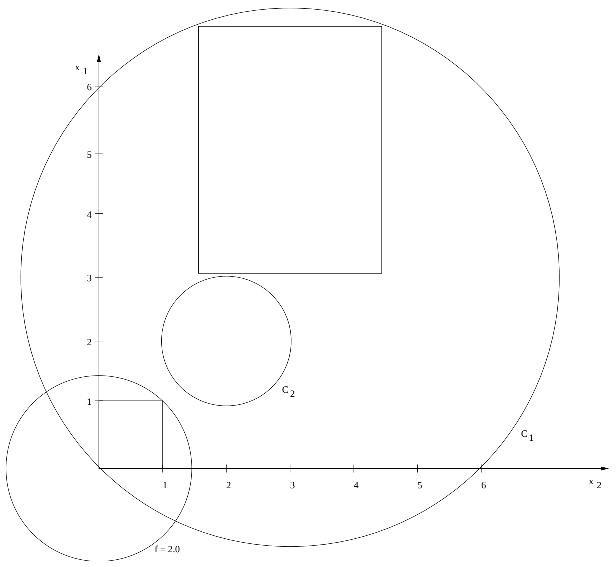

Fig. 3. Test problem and result intervals of the first and last lines of Table 1.

center $(2, 2)$, and radius 1. The only global optimal point is at the origin, and the optimal function value is $f^* = f(0, 0) = 0$. The level sets S_{f_ε} are circles around the origin with radii of $\sqrt{f_\varepsilon}$, respectively. The constraint $g_1(x) \leq 0$ is active at the global minimum, and its normal is parallel to the line $x_1 = x_2$. The problem is illustrated on Fig. 3.

The inclusion functions are generated by natural interval extension:

$$F(X) = X_1^2 + X_2^2,$$

$$G_1(X) = (3 - X_1)^2 + (3 - X_2)^2 - 18, \text{ and}$$
$$G_2(X) = 1 - (2 - X_1)^2 - (2 - X_2)^2.$$

The capital letters denote again intervals with the subscript indicating coordinate direction. These inclusion functions are exact in the sense that the so-called *excess width* (defined by $w(F(X)) - w(f(X))$) is zero for every argument interval. It is unfortunately not typical for interval calculations, yet it makes the demonstration of the working of the algorithm more transparent.

Assuming exact arithmetic

Set the seed point to $x^{\text{seed}} = (0.5, 0.5)^T$. The conditions (3) and (4) are now fulfilled for each $f_\varepsilon > 0.5$:

$$g_1(x^{\text{seed}}) = -5.5 < 0,$$

$$g_2(x^{\text{seed}}) = -3.5 < 0,$$
$$f(x^{\text{seed}}) = 0.5 < f_\varepsilon.$$

Choose the algorithm parameters $d(i,1) = d(i,2) = 0.1$ for $i = 1, 2$ and $\eta = 0.01$. For $f_\varepsilon = 2.0$ the intersection set of feasible points and the level set S_{f_ε} is the intersection of the circles with centres (0, 0) and (3, 3), and with radii $\sqrt{2}$ and $3\sqrt{2}$, respectively. It is a convex set, and the maximal volume inscribed box is $X_1^* = [0.0, 1.0]$, $X_2^* = [0.0, 1.0]$.

The starting interval is set to $X_1 = [0.5, 0.5]$ and $X_2 = [0.5, 0.5]$. The first check is made on the interval $Y_1 = [0.4, 0.5]$, $Y_2 = [0.5, 0.5]$. The corresponding inclusion function values are

$$F(Y) = [0.4, 0.5]^2 + [0.5, 0.5]^2 = [0.16, 0.25] + [0.25, 0.25] = [0.41, 0.5],$$

$$G_1(Y) = [6.25, 6.76] + [6.25, 6.25] - 18 = [-5.5, -4.99],$$

and

$$G_2(Y) = 1 - [2.25, 2.56] - [2.25, 2.25] = [-3.81, -3.5].$$

The checking routine returns thus that Y is strongly feasible, and X is set in **Step 3** of the main algorithm to $([0.4, 0.5], [0.5, 0.5])^T$.

The next check is then made in **Step 5** on the interval $Y_1 = [0.5, 0.6]$, $Y_2 = [0.5, 0.5]$. The corresponding inclusion function values are $F(Y) = [0.5, 0.61]$, $G_1(Y) = [-5.99, -5.5]$ and $G_2(Y) = [-3.5, -3.21]$. The actual interval is then updated to $X_1 = [0.4, 0.6]$, $X_2 = [0.5, 0.5]$.

The actual interval is modified for $i = 2$ to $([0.4, 0.6], [0.4, 0.5])^T$, and then to $([0.4, 0.6], [0.4, 0.6])^T$. The sequence of subsequent actual intervals is as follows:

$$X = ([0.3, 0.7], [0.3, 0.7])^T,$$
$$X = ([0.2, 0.8], [0.2, 0.8])^T,$$
$$X = ([0.1, 0.9], [0.1, 0.9])^T.$$

The final interval X was obtained after 48 inclusion function evaluations. The calculation of inclusion functions involves on the average ca. two times more computation than the corresponding real functions do. Until this point was reached, the checking routine accepted all the extension intervals Y immediately, without subdivision. Thus the value of the algorithm parameter θ had no effect on this part of the result. In the next iteration the checked intervals and the inclusion function values $F(Y)$ are as follows:

$$Y = ([0.0, 0.1], [0.1, 0.9])^T, \text{ and } F(Y) = [0.01, 0.82],$$
$$Y = ([0.9, 1.0], [0.1, 0.9])^T, \text{ and } F(Y) = [0.82, 1.81],$$
$$Y = ([0.0, 1.0], [0.0, 0.1])^T, \text{ and } F(Y) = [0.00, 1.01],$$

Table 1. The role of the seed point in locating maximal feasible boxes.

x^{seed}	X^{res}	$\text{vol}(X^{\text{res}})$	NFE
$(0.50, 0.50)^T$	$([-0.000028, 1.046680], [0.000098, 0.951000])^T$	0.99532	1822
$(0.10, 0.10)^T$	$([-0.000056, 1.000066], [0.000069, 0.999902])^T$	0.99996	1945
$(0.01, 0.01)^T$	$([-0.003131, 1.034968], [0.003153, 0.963763])^T$	0.99721	2065
$(0.90, 0.90)^T$	$([-0.000028, 1.000160], [0.000098, 0.999805])^T$	0.99989	2118
$(0.10, 0.90)^T$	$([-0.400226, 0.556738], [0.462630, 1.300000])^T$	0.80133	1610
$(0.00, 1.00)^T$	$([-0.425529, 0.522607], [0.496856, 1.314084])^T$	0.77484	1669
$(-0.01, 0.10)^T$	$([-0.072507, 0.990052], [0.074328, 1.009826])^T$	0.99402	1996
$(4.0, 4.0)^T$	$([2.662546, 6.048340], [2.749031, 5.9508304)^T$	10.841	3015
$(5.0, 5.0)^T$	$([2.600000, 6.048340], [2.800000, 5.9508246)^T$	10.865	2677
$(3.0, 6.0)^T$	$([1.732192, 4.267773], [3.000000, 7.0487793)^T$	10.266	2801

$$Y = ([0.0, 1.0], [0.9, 1.0])^T, \text{ and } F(Y) = [0.81, 2.00].$$

The last interval is not strongly feasible, and a new $d(2, 2) < 0.05$ is determined (depending on the value of θ). And in this way the next $X = ([0.0, 1.0], [0.0, 0.9])^T$.

With further calculations this actual interval may be refined to obtain a maximal box X^*. We have a computational proof that each point x of the result interval is feasible, and $f(x) < f_\varepsilon$.

Computer implementation with outward rounding

The main difference between the results of the last section and those obtained by the computer program is that the latter is produced by operations with outward rounding. For example, $Y = ([0.0, 1.0], [0.9, 1.0])^T$ would be found feasible (but not strongly feasible) calculating with exact arithmetic (since then $\overline{F}(Y) = 2.0$), while $\overline{F}(Y) > 2.0$ if it is evaluated with outward rounding. This is the reason why the results in the first line of Table 1 may be slightly different from those discussed in the earlier subsection. It is worth mentioning that if the stopping condition would be based on the difference $((\underline{X}_i) - d(i, 1)) - (\underline{X}_i)$ then this value could also attain zero because of the computer representation of floating point numbers.

Table 1 contains details of the results on the numerical test that examines how the place of the seed point affects the result box constructed by the program. In the following, we use only one initial value for all $d(i, j)$ step sizes ($i = 1, 2$; $j = 1, 2$). All of the problem and algorithm parameters were constant during this test ($d(i, j) = 0.1$ for $i = 1, 2$ and $j = 1, 2$, $\eta = 0.0001$ and $\theta = 0.0001$), only x^{seed} and f_ε was changed. The latter was 2.0 for the first 7 lines and 72.0 for the last three. For the problem specified by $f_\varepsilon = 2.0$, the maximal volume feasible box is $X^* = [0.0, 1.0]^2$. The result interval calculated by the program is denoted by X^{res}, and $\text{vol}(X^{\text{res}})$ is its volume. The latter is found close to one (it is, of course, not greater than $\mathrm{vol}(X^*) = 1$). NFE stands for the number of function ($F(X)$ and $G_j(X)$ for $j = 1, 2, \ldots, m$) evaluations.

The test results presented in Table 1 suggest that the seed point may be chosen close to the normal of the active constraint at x^*, even if it is outside of X^*. It is interesting that the center of the maximal volume inscribed box is not an optimal seed point. It is also worth mentioning that in the first 7 lines (where $S_{f_\varepsilon} \cap A$ is symmetric for the $x_1 = x_2$ line) the first component of the result interval is always wider than the second one. The explanation for it is that the actual interval is always enlarged first along the first coordinate direction. Two result boxes are shown on Fig. 3 together with the constraint functions and a corresponding levels.

The presented algorithm was also applied to a practical engineering design problem to construct manufacturing tolerances for an optimal design of composite materials [ZG92] that motivated our study. The numerical experiences on this composite laminate design problem are reported in the papers [KZ93, KZ96]. The methodology was also applied in civil engineering, for providing optimal designs with manufacturing tolerances for building and construction problems [CC04]. All these computational studies confirmed that the suggested algorithm is capable to provide applicable size suboptimal feasible tolerance intervals for a wide set of problems using an acceptable amount of computation.

In addition to our above mentioned applications, the related papers [CK95, KZ93, KZ96] were cited by several scientific publications reporting on the successful use of the introduced algorithms in several diverse application fields. The tolerance optimization approach was also used to describe the set of Hénon mapping parameters that allow chaotic behaviour [BC07, CG06].

3 Chaos verification in mathematical models of dynamical systems

The last section is devoted to an optimization model and to the related algorithms to locate chaotic regions of dynamic systems [CG06]. Computer-assisted proofs for the existence of chaos are important for the understanding of dynamic properties of the solutions of differential equations. These techniques have been intensively investigated recently, see e.g. [GZ01, NR93, RN94, Z97, Z03].

We study verified computational methods to check and locate regions the points of which fulfill the conditions of chaotic behaviour. The investigated Hénon mapping is $\mathcal{H}(x, y) = (1 + y - Ax^2, Bx)$. The paper [Z97] considered the $A = 1.4$ and $B = 0.3$ values and some regions of the two dimensional Euclidean space: $E = E_1 \cup E_2 = \{(x, y) \mid x \geq 0.4,\ y \geq 0.28\} \cup \{(x, y) \mid x \leq 0.64,\ |y| \leq 0.01\}$, $\mathcal{O}_1 = \{(x, y) \mid x < 0.4,\ y > 0.01\}$, $\mathcal{O}_2 = \{(x, y) \mid y < 0\}$.

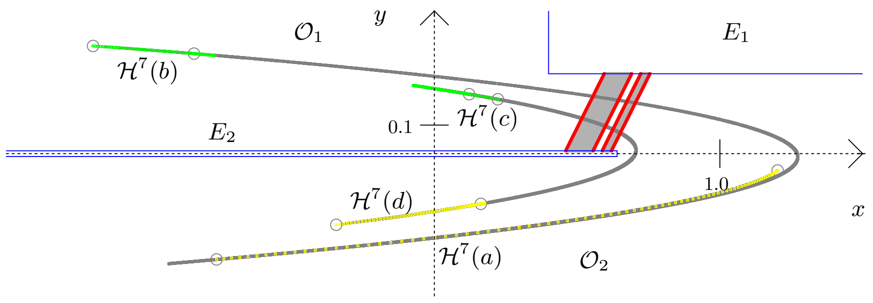

Fig. 4. Illustration of the H^7 transformation for the classic Hénon parameters $A = 1.4$ and $B = 0.3$ together with the chaotic region of two parallelograms. The a, b, c, and d sides of the parallelograms are depicted on the upper left picture of Figure 5.

According to [Z97] Theorem 3 below ensures the chaotic behaviour for the points of the parallelograms Q_0 and Q_1 with parallel sides with the x axis (for $y_0 = 0.01$ and $y_1 = 0.28$, respectively), with the common tangent of 2, and x coordinates of the lower vertices are $x_a = 0.460$, $x_b = 0.556$; and $x_c = 0.558$, $x_d = 0.620$, respectively. The mapping and the problem details (such as the transformed sides of the parallelograms, $H^7(a)$, $H^7(b)$, $H^7(c)$, and $H^7(d)$) are illustrated on Figure 4.

Theorem 3. *Assume that the following relations hold for the given particular Hénon mapping:*

$$\mathcal{H}^7(a \cup d) \subset \mathcal{O}_2, \tag{7}$$

$$\mathcal{H}^7(b \cup c) \subset \mathcal{O}_1, \tag{8}$$

$$\mathcal{H}^7(Q_0 \cup Q_1) \subset \mathbb{R}^2 \setminus E, \tag{9}$$

then chaotic trajectories belong to the starting points of the regions Q_0 and Q_1.

The present section provides a method to verify chaos for certain mappings and regions. We discuss first how to check the set theoretical conditions of the above theorem in a reliable way by computer programs. Then we introduce optimization problems that provide a model to locate chaotic regions. We check the correctness of the earlier published chaotic region, the correctness of the underlying checking algorithms, and prove the optimization model. We also give new chaotic places located by the new technique. The papers [BC07, CB07] provide additional new chaotic regions located by the present method.

The main difficulty of checking conditions (7) to (9) is that one has to prove these for a continuum of points. In [Z97] the author calculated the Lipschitz constant, gave an upper bound for the rounding error committed and thus reduced the whole task to investigating a finite number of points of a dense enough grid. This method works only with human interaction. To search chaotic regions an automated checking routine is more appropriate. The technique we applied combines interval arithmetic and adaptive branch-and-bound subdivision of the region of interest. It is basically a customized version of the technique introduced in Section 2.

This algorithm first encloses the sets Q_0 and Q_1 in a 2-dimensional closed interval I, the starting interval. Then to prove subset relations an adaptive branch-and-bound technique generates such a subdivision of the starting interval that either:

- for all subintervals the given conditions of chaos hold – in case they contain points of the respective sets, or
- it is shown that a small subinterval (of a user set size) exists that contains at least one point of the respective set, and it contradicts at least one of the relations.

Now the sets $\mathcal{O}_1$, $\mathcal{O}_2$, and $R^2 \setminus E$ in the conditions (7) to (9) are all open sets, and the union of a finite number of closed sets is closed. It is why the algorithm should check whether the transformed subintervals are subsets of the respective sets.

Our algorithm is capable of recognizing that a region satisfies the conditions of chaos. We have proven the correctness of the procedure in [CG06].

3.1 A global optimization model for locating chaotic regions

Once we have a reliable computer procedure to check the conditions of chaotic behavior of a mapping it is straightforward to set up an optimization model that transforms the original chaos location problem to a global optimization problem.

The chaotic regions have several parameters that identify them. In the early phase of our investigation we have restricted the search to locate two parallelograms similar to that used in the article [Z97]: we are allowed to change the vertical and horizontal positions and also the common tangent, but the parallelograms always had two sides parallel to the x axis. It is also possible to find fitting parameter values for the Hénon mapping, i.e., for the mapping parameters A and B, and furthermore also for parameters of the aimed sets of the underlying theorem, e.g. the border coordinates of the set E.

The search for a chaotic region was modelled as a constrained global optimization problem, subsequently the constraints were represented by a penalty function approach. The original objective function was constant, still the possibility exists to extend it to a more complex form that expresses further aims, e.g. to locate a second chaotic region, different from the known one.

The key question for the successful application of a global optimization algorithm is how to compose the penalty functions. On the basis of earlier experiences collected solving similar constrained problems, we have decided to add a nonnegative value proportional to how much the given condition was violated, plus a fixed penalty term in case at least one of the constraints was not satisfied.

As an example, consider the case when one of the conditions for the transformed region was hurt, e.g. when (8), i.e., the relation $\mathcal{H}^k(b \cup c) \subset \mathcal{O}_1$ does not hold for a given kth iterate, and for a region of two parallelograms. For such a case the checking routine will provide a subinterval I that contains at least one point of the investigated region, and which contradicts the given condition. Then we have calculated the Hausdorff distance of the transformed subinterval $H^k(I)$ to the set $\mathcal{O}_1$ of the right side of the condition,

$$\max_{z \in H^k(I)} \inf_{y \in \mathcal{O}_1} d(z, y),$$

where $d(z, y)$ is a given metric, a distance between two two-dimensional points. Notice that the use of maximum in the expression is crucial, with minimization instead our optimization approach could provide (and has provided) result regions that do not fulfill the given conditions of chaotic behaviour. On the other hand, the minimal distance according to points of the aimed set (this time $\mathcal{O}_1$) is satisfactory, since it enables the technique to push the search into proper directions. In cases when the checking routine answered that the investigated subinterval has fulfilled the given condition, we have not changed the objective function.

Summing it up, we have considered the following bound constrained problem for the T inclusion function of the mapping $\mathcal{T}$:

$$\min_{x \in X} g(x), \tag{10}$$

where

$$g(x) = f(x) + p\left(\sum_{i=1}^{m} \max_{z \in T(I(x))} \inf_{y \in S_i} d(z, y)\right),$$

X is the n-dimensional interval of admissible values for the parameters x to be optimized, $f(x)$ is the original, nonnegative objective function, and $p(y) = y + C$ if y is positive, and $p(y) = 0$ otherwise. C is a positive constant, larger than $f(x)$ for all the feasible x points, m is the number of conditions to be fulfilled, and S_i is the aimed set for the i-th condition. In this discussion $I(x)$ is the subinterval returned by the checking routine (or the empty set). The interval $I(x)$ depends implicitly on the parameter x to be optimized.

For more complicated cases the fixed sets given in Theorem 3 should also be changed subject to certain structural constraints, e.g. the x_a, x_b, x_c, and x_d coordinates of the parallelograms have to follow this order. These new conditions can also be represented in a similar way, following the penalty function approach of (10).

We have proved for our optimization model fits the chaos location problem, and the suggested global optimization method is capable to find chaotic places [CG06]. The interval arithmetic based checking routine provides a computational proof for the existence of the chaos there.

3.2 Numerical results

For the computational experiments we have applied the C-XSC programming language [KK93] supporting interval arithmetic. The results were obtained both in Linux and in the Cygwin environment, on an average personal computer. In the present subsection we just provide some demonstrative examples for the functioning of the introduced technique. First we have checked the reported chaotic region [Z97] by our checking routine.

1. We have investigated the seventh iterate of the Hénon mapping with the classic parameters of $A = 1.4$ and $B = 0.3$. The checked region consists of two parallelograms with sides parallel to the x-axis, the first coordinates of the lower corner points were 0.460, 0.556, 0.588, and 0.620, while the second coordinates were the same, 0.01. The common y coordinate for the upper corner points was 0.28, and the tangent of the sides was 2. We have set the ε threshold value for the checking routine to be 10^{-10}.

First the algorithm determined the starting interval that contains the region to be checked:

$$[0.46000, 0.75500] \times [0.01000, 0.28000].$$

Then the three conditions were checked one after the other. All of these proved to be valid – as expected. The number of function evaluations (for the transformation, i.e., for the seventh iterate of the Hénon mapping in each case) were 273, 523, and 1,613, respectively. The algorithm stores those subintervals for which it was impossible to prove directly whether the given condition holds; these required further subdivision to achieve a conclusion. The depth of the stack necessary for the checking was only 11, 13, and 14, respectively. The CPU time used was negligible, a few seconds.

The results are demonstrated in Figure 5 (together with the parallelograms). The density of the subintervals indicates that in the related subregion the given condition was just fulfilled, the overestimation involved in the interval calculations required much refinement.

Summarizing the results, we were able to prove with an acceptable amount of computation and human overhead that the published system is chaotic in the given, known regions. This confirms the result of Zgliczynski.

2. As a second step, randomly chosen A and B values were checked close to the classical parameters. The following ones ensured chaos for the $\mathcal{H}^7$ Hénon system with unchanged other region and algorithm parameters:

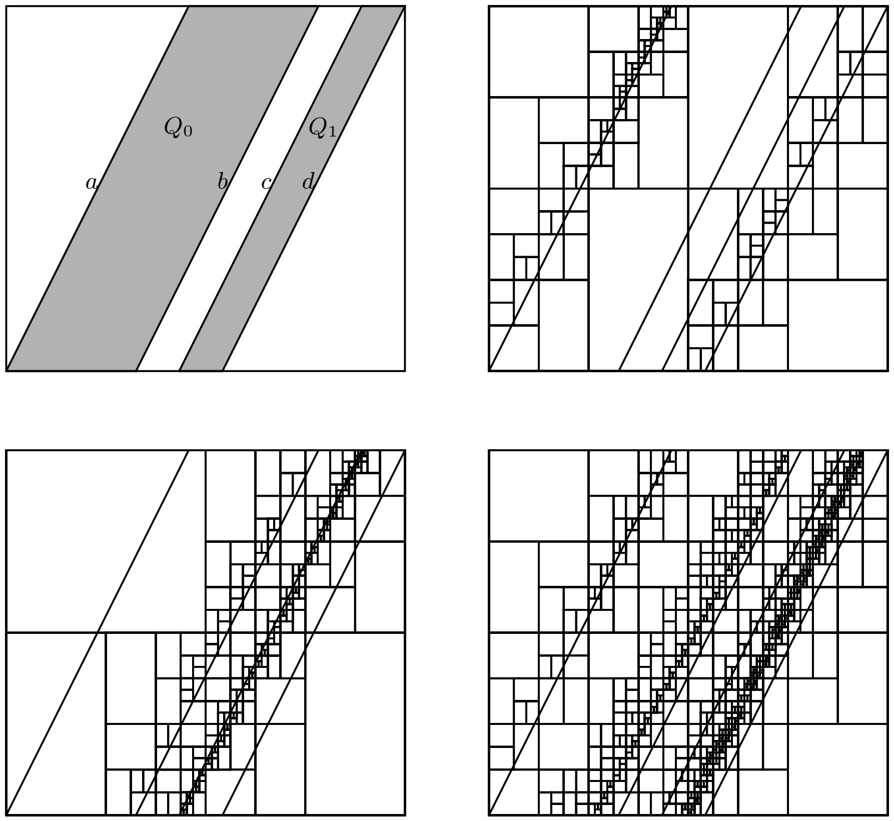

Fig. 5. The parallelograms and the starting interval covered by the verified subintervals for which the given condition holds (in the order of mentioning in Theorem 3).

```
         A                    B
1.3555400848181643, 0.32668379383472889
1.3465721096594685, 0.32450555140362324
1.4403201855906845, 0.22585009468060412
1.4136297518450903, 0.26880306437090162
1.3702743902664050, 0.30756016043366862
```

Notice that without our automatic checking of the conditions for chaos it could have been very difficult when not even impossible to arrive at the above results, since the human interaction and insight necessary plus the required overhead could be prohibitive.

3. As a third way of applying the checking routine, we have determined parameter intervals around $A = 1.4$ and $B = 0.3$ for which mapping $\mathcal{H}^7$ still

has chaos on the same pair of parallelograms. The obtained intervals were $A \in [1.377599, 1.401300]$ and $B \in [0.277700, 0.310301]$. Notice that these intervals do not contain all the A, B pairs given on the previous page.

The technique with which this result was obtained is the one discussed in Section 2. The key feature necessary for this algorithm is that the checking routine can accept interval valued parameters for the calculated mapping. More solved chaotic region location problems are reported with technical details in [BC07].

Acknowledgement. The presented work was supported by the Grants OTKA T 016413, T 017241, OMFB D-30/2000, OMFB E-24/2001, OTKA T 034350, T 048377, and T 046822.

References

[AH83] G. Alefeld and J. Herzberger: Introduction to Interval Calculations. Academic Press, New York (1983)

[BC07] B. Bánhelyi, T. Csendes, and B.M. Garay: Optimization and the Miranda approach in detecting horseshoe-type chaos by computer. Int. J. Bifurcation and Chaos, **17**, 735–748, (2007)

[BR87] J.H. Bleher, S.M. Rump, U. Kulisch, M. Metzger, Ch. Ullrich, and W. Walter: FORTRAN-SC. A study of a FORTRAN extension for engineering/scientific computation with access to ACRITH, Computing, **39**, 93–110 (1987)

[C89] T. Csendes: An interval method for bounding level sets of parameter estimation problems. Computing, **41**, 75–86, (1989)

[C90] T. Csendes: Interval method for bounding level sets: revisited and tested with global optimization problems. BIT, **30**, 650–657, (1990)

[C98] T. Csendes: Optimization methods for process network synthesis – a case study. In: C. Carlsson and I. Eriksson (eds.) Global & multiple criteria optimization and information systems quality. Abo Academy, Turku, 1998, pp. 113-132

[C07] T. Csendes: Reliable optimization, methods and applications. DSc Dissertation, Hungarian Academy of Sciences, Budapest (2007)

[CB07] T. Csendes, B. Bánhelyi, and L. Hatvani: Towards a computer-assisted proof for chaos in a forced damped pendulum equation. J. Computational and Applied Mathematics, **199**, 378-383, (2007)

[CC04] A.E. Csallner, T. Csendes, and A.B. Kocsis: Reliable numerical computation in civil engineering. Numerical Algorithms, **37**, 85–91, (2004)

[CG06] T. Csendes, B.M. Garay, B. Bánhelyi: A verified optimization technique to locate chaotic regions of a Hénon system. J. of Global Optimization, **35**, 145–160, (2006)

[CK95] T. Csendes, Z.B. Zabinsky, and B.P. Kristinsdottir: Constructing large feasible suboptimal intervals for constrained nonlinear optimization. Annals of Operations Research, **58**, 279–293, (1995)

[CP93] T. Csendes, and J. Pintér: A new interval method for locating the boundary of level sets. Int. J. of Computer Mathematics, **49**, 53–59, (1993)

[F76] A.V. Fiacco: Sensitivity analysis for nonlinear programming using penalty methods. Mathematical Programming, **12**, 287–311, (1976)

[GZ01] Z. Galias and P. Zgliczynski: Abundance of homoclinic and heteroclinic orbits and rigorous bounds for the topological entropy for the Hénon map. Nonlinearity, **14**, 909–932, (2001)

[H92] E. Hansen: Global optimization using interval analysis, Marcel Dekker, New York (1992)

[HH93] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz: Numerical Toolbox for Verified Computing I., Springer, Berlin (1993)

[J92] H.P. Jüllig: BIBINS/2.0 – C++ Bibliotheken für Vektoren und Matrizen über beliebigem skalaren Datentyp unter Berücksichtigung spezieller spärlicher Strukturen sowie Intervalldatentypen. Bericht 92.6, Technical University of Hamburg-Harburg (1992)

[K92] R.B. Kearfott: An interval branch and bound algorithm for bound constrained optimization problems. J. Global Optimization, **2**, 259–280, (1992)

[KK92] R. Klatte, U. Kulisch, M. Neaga, D. Ratz, and Ch. Ullrich: PASCAL-XSC – Language Reference with Examples. Springer-Verlag, Berlin (1992)

[KK93] R. Klatte, U. Kulisch, A. Wiethoff, C. Lawo, and M. Rauch: C-XSC – A C++ Class Library for Extended Scientific Computing, Springer, Heidelberg (1993)

[KF93] Z. Kovács, F. Friedler, and L.T. Fan: Recycling in a separation process structure, AIChE J. **39**, 1087–1089, (1993)

[KZ93] B.P. Kristinsdottir, Z.B. Zabinsky, T. Csendes, and M.E. Tuttle: Methodologies for tolerance intervals. Interval Computations, **3**, 133–147, (1993)

[KZ96] B.P. Kristinsdottir, Z.B. Zabinsky, M.E. Tuttle, and T. Csendes: Incorporating manufacturing tolerances in optimal design of composite structures, Engineering Optimization, **26**, 1–23, (1996)

[M04] M.Cs. Markót: Optimal Packing of 28 Equal Circles in a Unit Square – the First Reliable Solution. Numerical Algorithms, **37**, 253–261 (2004)

[MC05] M.C. Markót and T. Csendes: A new verified optimization technique for the "packing circles in a unit square" problems. SIAM J. on Optimization, **16**, 193–219 (2005)

[NR93] A. Neumaier, A. and T. Rage: Rigorous chaos verification in discrete dynamical systems, Physica D, **67**, 327–346, (1993)

[RN94] T. Rage, A. Neumaier, and C. Schlier: Rigorous verification of chaos in a molecular model, Phys. Rev. E, **50**, 2682–2688, (1994)

[RR84] H. Ratschek and J. Rokne: Computer Methods for the Range of Functions, Ellis Horwood, Chichester (1984)

[RR88] H. Ratschek and J. Rokne: New Computer Methods for Global Optimization. Ellis Horwood, Chichester (1988)

[RR93] H. Ratschek and J. Rokne: Experiments using interval analysis for solving a circuit design problem. J. Global Optimization, **3**, 501–518, (1993)

[R99] S.M. Rump: INTLAB – INTerval LABoratory. In T. Csendes (ed.): Developements in Reliable Computing, Kluwer, Dordrecht, 1999, pp. 77–104

[SM07] P.G. Szabó, M.Cs. Markót, T. Csendes, E. Specht, L.G. Casado, and I. García: New Approaches to Circle Packing in a Square – With Program Codes. Springer, Berlin (2007)

[ZG92] Z.B. Zabinsky, D.L. Graesser, M.E. Tuttle, and G.I. Kim: Global optimization of composite laminates using Improving Hit-and-Run, in: Recent

Advances in Global Optimization, Princeton University Press, Princeton (1992), pp. 343–368.

[Z97] P. Zgliczynski: Computer assisted proof of the horseshoe dynamics in the Hénon map. Random & Computational Dynamics, **5**, 1–17, (1997)

[Z03] P. Zgliczynski: On smooth dependence on initial conditions for dissipative PDEs, an ODE-type approach. J. Differential Equations, **195**, 271–283, (2003)