Computation of Limit Cycles for Uncertain Nonlinear Fractional-order Systems using Interval Constraint Propagation

P. S. V. Nataraj* Rambabu Kalla**

* Systems and Control Engineering Group, IIT Bombay, India (Tel: +91-22-25767887; e-mail: nataraj@sc.iitb.ac.in).
** Systems and Control Engineering Group, IIT Bombay, India (Tel: +91-22-25764884; e-mail: ram@sc.iitb.ac.in)

Abstract: The present paper proposes an algorithm to compute limit cycle points for an uncertain nonlinear fractional-order system with separable nonlinearities using interval constraint propagation. It is first shown that the problem of finding limit cycle points can be formulated as an interval constraint satisfaction problem and then solved using branch and prune algorithm. The algorithm guarantees that all the points on the limit cycle locus are computed to prescribed accuracy, i.e., the error in the computation of limit cycle point can never exceed the accuracy tolerance specified by the user. The algorithm also guarantees that all the limit cycle points in the given search domain are found. The other advantage of the method is that it does not need any kind of approximation of the fractional-order system. It must be noted that any errors resulting in the limit cycle locus due to approximate nature of describing function method cannot be avoided. The proposed algorithm is illustrated on two examples taken from the literature.

Keywords: Constraint satisfaction problems, Control system analysis, Fractional-order systems, Interval analysis, Parameter uncertainty, Reliable computation, Robust control.

1. INTRODUCTION

In the last decade, a significant amount of research has been reported for the analysis and design of linear fractional-order control systems. However, for the analysis and design of non-linear fractional-order systems, there is a sparsity of methods in the literature. Barbosa and Machado (2002) apply the fractional describing function to systems with fixed parameter values in presence of backlash and impact phenomena. Manabe (2004) proposes an improved controller for a fractional-order system with fixed parameter values, under the strong influence of saturation. Pereira et al. (2004) propose a MATLAB toolbox for analyzing fractional-order systems with fixed parameter values and hard nonlinearities. However, we find sparsity of tools for the stability analysis of uncertain nonlinear fractional-order systems. Deshpande (2006) proposes an interval arithmetic based technique for computing the limit cycle locus. In this paper we add interval constraint propagation to interval arithmetic to compute the limit cycle locus more efficiently.

The objective of this paper is to propose an efficient tool to construct the limit cycle locus of uncertain nonlinear fractional-order systems with separable nonlinearities using interval constraint propagation. The uncertainties can be in the parametric values of the linear and nonlinear elements. The proposed tool is in the form of an algorithm having several useful features: it guarantees that all points on limit cycle locus in the region of interest are found; if no limit cycle point exists in the region of interest, then this fact is established with a mathematical and computational guarantee (such guarantees are crucial in stability analysis). The algorithmic results can also be used to identify which uncertain parameters have a strong influence on the limit cycle behavior.

The rest of the paper is organised as follows: Section-2 deals with the background of limit cycles, uncertain fractional-order systems, interval arithmetic, and interval constraint processing. The methodology followed in proposed algorithm is presented in the Section-3. In Section-4, we give the proposed algorithm for the computation of limit cycle points for a nonlinear uncertain fractional-order system. Theoretical properties of the proposed algorithm are discussed in Section-5. Section-6 demonstrates the proposed algorithm on two examples taken from the literature and Section-7 gives the conclusions.

2. BACKGROUND

2.1 Limit Cycles

The basic idea of the describing function consists of applying a sinusoidal signal as input to the nonlinear element and considering only the fundamental component of the signal appearing at the output of the nonlinear system.

Consider the feedback system of Fig. 1 where $p(s,q_g)$ denotes the transfer function of the fractional-order linear element with parameter vector $q_g$, $N(a, \omega, q_n)$ denotes the describing function of the nonlinear element with
Fig. 1. Nonlinear feedback system comprising of a fractional-order plant $p$ and a nonlinear element with describing function $N$.

The parameter vector $q_n$, and $a, \omega$ denote the amplitude and frequency of the periodic input signal to nonlinear element. Let $y := (a, \omega) \in \mathcal{R}^2$, $q := (q_3, q_n) \in \mathcal{R}^l$. Then, under certain assumptions stated in Atherton (1975), the nonlinear system exhibits a limit cycle if there exists a solution $y^*$ to the characteristic equation $f(y, q) = 0$, where $f(y, q)$ is defined as

$$f(y, q) := (f_{Re}(y, q), f_{Im}(y, q))$$

$$f_{Re}(y, q) := \Re \{1 + N(a, \omega, q_n)p(s, q_3)\}$$

$$f_{Im}(y, q) := \Im \{1 + N(a, \omega, q_n)p(s, q_3)\}$$

(1)

The solution $y^*$ is called a limit cycle point, and the corresponding $a^*$ and $\omega^*$ are called the limit cycle amplitude and frequency, respectively.

Now, suppose there is parametric uncertainty in the system such that the parameter vector $q$ varies over a bounding box $q^0 \in \mathcal{I} (\mathcal{R}^l)$. The parametric uncertainty gives rise to an uncertain fractional-order nonlinear system. The problem of computing the limit cycle locus for this uncertain system is defined as

$$\mathcal{L}_C(f, q^0) := \{y \in \mathcal{R}^2 : f(y, q) = 0\}$$

(2)

for some $q \in q^0$. We assume that the transfer function $p$ and the describing function $N$ are continuously differentiable in $q_3$ and $q_n$, respectively, and in $y$.

### 2.2 Uncertain Fractional-order Systems

A fractional-order transfer function is of the form

$$P(s) = b_0 s^{\alpha_0} + ... + b_1 s^{\alpha_1} + b_0 s^{\alpha_0}$$

(3)

as given by Podlubny (1998), where $\alpha_i, \beta_j$ are positive real numbers and $a_i, b_j, i = 0, 1, ..., n, j = 0, 1, ..., m$ are arbitrary real coefficients. In the time domain, this transfer function can be represented by a fractional-order differential equation. Several researchers have shown that certain phenomena, especially those governed by non-integral order physical laws, can be described more accurately by fractional-order differential equations; for example transmission lines given by Wang (1987), electrical noises given by Mandelbrot (1967), and diffusion of heat into a semi-infinite solid given by Petras (1990), etc.

For an uncertain fractional-order system the coefficients and the powers in the transfer function need not be fixed real numbers but are intervals of real numbers.

### 2.3 Interval Arithmetic

The key idea behind interval arithmetic as given by Moore (1966) is the approximation of real numbers by intervals to quantify the errors introduced with finite precision arithmetic. In addition, interval computations provide an appropriate framework to deal with uncertain data.

A closed interval $x = [q, \bar{x}]$, with $q, \bar{x} \in \mathcal{R}$ can be regarded as the set of real numbers $\{q \leq r \leq \bar{x}\}$, or as an approximation of some real numbers lying within that set. Instead of using a single floating-point number to approximate a real, interval arithmetic encloses the real number within a closed interval having (in general) floating-point bounds. An interval vector $x = (x_1, ..., x_n)^T$ with components $x_k = [l_k, u_k]$ is called as a box. $I(x)$ is the set of all boxes contained in $x$. The general results of interval arithmetic like natural inclusion function, hull, union, projection, width of an interval $w(x)$ etc., can be found in the book by Moore (1966).

### 2.4 Interval Constraint Processing

Numerical constraint satisfaction problems accept as input only problems represented by exact numerical values and correspondingly produce only crisp solutions as output. This limitation can be removed by implementing generalized constraint propagation schemes based on interval arithmetic instead of conventional arithmetic. By using intervals instead of exact values, we may express inexact numerical constraints in a well-defined way and compute necessary conditions for consistency in inconsistent situations.

An interval constraint satisfaction problem (ICSP) as given by Hyvönen (1992) is composed of

1. A set of real valued variables, e.g., $v = \{v_1, ..., v_n\}$;

2. A set of interval domains of the variables, e.g., $x = \{x_1, ..., x_n\}$;

3. A set of constraints, e.g., $c = \{c_1, ..., c_m\}$ over the given set of variables.

The problem is to find in the initial box $x_1 \times \ldots \times x_n$ all the consistent values with respect to all constraints. A variable $v_i \leftarrow x_i$ is consistent if and only if each interpretation $v_i \leftarrow x \in x_i$ can be satisfied with respect to all constraints by some extension: $\forall x \in x_i \exists \{v_1 \leftarrow x_1 \in x_1, ..., v_i \leftarrow x_i, ..., v_m \leftarrow x_m \in x_m\}: c_1, ..., c_m$ are satisfied. The set of variables of the constraint $c_i$ is denoted by $\mathcal{V}_{c_i}$.

There are two steps in solving an ICSP, constraint propagation and constraint branching. The basic idea of constraint propagation algorithms (also called filtering or narrowing or consistency algorithms or narrowing operators) consists of removing, from the domains associated to the constraint variables, inconsistent values that can never be part of the solution. This process reduces significantly the search tree and possibly the computational effort to find a solution if one exists or to demonstrate that there is no solution. In general, the results are propagated through the whole constraint set, and the process is repeated until a stable set is obtained. Research in the area of solving interval constraint satisfaction problems (see, for example, Benhamou et al. (1999)) is devoted to finding correct and (near) optimal interval propagation techniques that can be efficiently implemented. A constraint narrowing algorithm
transforms the domains of those variables involved in it into tighter intervals such that:

(1) Resulting intervals are always included in the original ones (contractance property).

(2) All values in the original intervals verifying the associated constraint of the narrowing operator, belong to the resulting intervals (soundness or correctness).

(3) The subset interval relation is conserved by the transformation (monotonicity).

Well known examples of constraint narrowing operators are hull and box consistency by Benhamou et al. (1999) and kBConsistency operators by Lhomme (1993). In our problem, we make use of efficient implementation of hull consistency known as HC4 as the the narrowing operator. This is described below.

**HC4 filter:** The HC4 filter was proposed by Benhamou et al. (1999). Inputs to the HC4 filter are the constraint in user form (i.e. without decomposing it in several equations) and the set of interval domains (box). The algorithm efficiently computes an interval extension of the equation, narrowing intervals of the variables involved. Inside the HC4 filter the input equation is represented as an attribute in user form (i.e. without decomposing it in several equations) and terms in the equation form sub-trees rooted at nodes containing either a variable, a constant or an operation symbol.

The HC4 filter works in two phases called forward evaluation and backward propagation. The forward phase is a tree traversal going from the leaves to the root, evaluating at each node the natural interval extension of that sub-term of the constraint. The backward phase traverses the tree from the root to the leaves, projecting on each node the effects of interval narrowing already performed on its parent node. In the backward propagation phase, an interval may become empty. When this happens the constraint is inconsistent with respect to the initial domains. HC4 algorithm is explained by means of a simple example in Appendix-A. Refer to the paper by Benhamou et al. (1999) for an extended description.

Until now we have been dealing with the first step in solving the ICSP, i.e., constraint propagation. Constraint propagation algorithms alone are not sufficient for solving an ICSP, that is to say, they do not eliminate all the non-solution elements from the domains. As a consequence, it is necessary to employ some additional strategy to solve it. One complementary method is the so-called constraint branching that divides the variable domains to construct new sub-problems, i.e., branches in the search tree on which constraint propagation is reactivated. The process is also called as splitting or sub-division process.

In the present paper, we address the problem of computing limit cycle locus for nonlinear fractional-order system. The problem is formulated as an ICSP. HC4 filter is used for pruning or for domain reduction in this case.

3. METHODOLOGY

In the proposed algorithm, one first constructs an initial search box $y^0$ that contains all (if any) limit cycle locus points. Such a box can be constructed as follows. The amplitude $a$ and frequency $\omega$ are non-negative, so $y^0 \in I (\mathbb{R}^2)$. On a computer, set $y^0 \leftarrow [0, real_{\text{max}}]^2$ where $real_{\text{max}}$ is the largest machine representable number on the computer. Further, one sometimes knows the ranges in which limit cycle amplitudes and frequencies occur in a particular problem. If so, $y^0$ can be bounded to enclose these ranges. The computation of limit cycle loci is formulated as an ICSP. $v = \{y, q\}$ where $y := (a, \omega)$ and $q := (q_0, q_n)$ forms the variable set. The constraints of the ICSP are $f_{Re}(y, q) = 0$ and $f_{Im}(y, q) = 0$ which are defined in (1). The algorithm is guaranteed to find all the limit cycles in the given search domain.

4. PROPOSED ALGORITHM

The algorithm for computation of spectral set is as follows:

**Inputs:** Expression $f_{Re}(y, q) = 0$ and $f_{Im}(y, q) = 0$, the parameter vector $q$, the initial uncertain interval vector $q^0$, initial search region $[0, A]$ for $a$, $[0, \omega^0]$ for $\omega$ and accuracy tolerance $\varepsilon$.

**Output:** All limit cycles in the given search domain computed to the prescribed accuracy tolerance $\varepsilon$.

**BEGIN Algorithm**

**Initialization part**

(1) Form the variable set comprising $v = \{y, q\} = \{a, \omega, q\}$ for the ICSP.

(2) Construct the initial search domain (box) $x^0 = \{a^0, \omega^0, q^0\}$ where $a^0 = [0, A]$, $\omega^0 = [0, \omega^0]$ and $q^0$ is the interval vector of the uncertain parameters.

(3) Form the the constraint set $c = \{c_1, c_2\}$ where $c_1 = \{f_{Re}(y, q) = 0\}$ and $c_2 = \{f_{Im}(y, q) = 0\}$

(4) Initialize the solution list $L^{sol} \leftarrow \{}$ and the working list $L \leftarrow x^0$

**Iterative part**

(5) **WHILE** $L \neq \{}$ **DO**

(6) Extract $x$ from $L$

(7) $s \leftarrow c$

(8) **WHILE** $s \neq \{}$ **AND** $x \neq \{}$

(9) Extract $c_i$ from $s$

(10) Narrow the box $x$ using HC4 filter explained in the Section 2.4 to $x'$.

(11) **IF** $x \neq x'$ **THEN**

$\hspace{1cm}$ $s \leftarrow s \cup \{c_i \mid \exists x_k \in V_{c_i} \wedge x_k \neq x'_k\}$

(Add to the set $s$ all the constraints containing the variables whose search domains are narrowed)

$\hspace{1cm}$ (12)
The list $\mathcal{L}^{sol}$ contains the limit cycle points (boxes) $(a, \omega)$ computed to the prescribed accuracy in the given search domain.

5. THEORETICAL PROPERTIES

The properties of the algorithm concerning the reliability, accuracy, enclosure of all the limit cycle points, etc., readily follow from the basics of interval analysis. We refer the reader to the book by Hansen and Walster (2005) where these are detailed.

The maximum error in the computed limit cycle points cannot be more than the accuracy tolerance $\epsilon$. The procedure ensures that we do not miss out any limit cycle point in the given search domain.

It must be noted that any errors resulting in the limit cycle locus due to approximate nature of describing function method cannot be avoided.

6. ILLUSTRATIVE EXAMPLES

We demonstrate the proposed method on two examples. The examples are executed on a computer with Intel® Core™ 2 Duo 2.4GHz processor and 2GB of RAM, running Linux Fedora Core-7. The proposed algorithm is implemented using RealPaver (Granvilliers and Benhamou (2006)).

The proposed algorithm is applicable to a wide variety of nonlinearities, such as saturation, backslash etc. In the example below, we choose the nonlinearity as relay with hysteresis type, as it is a nonlinearity with memory. Moreover, it is found to readily offer good limit cycle points for illustrating the various features of the proposed algorithm.

Example 1. Consider the nonlinear feedback system in Fig. 1 where the linear element is the fractional-order transfer function of the gas turbine obtained through identification given by Deshpande (2006)

$$p(s) = \frac{b_1}{a_1s^{\alpha_1} + a_2s^{\alpha_2} + a_3}$$

and the nonlinear element is a memory type nonlinearity comprising of relay with hysteresis. The describing function of this nonlinear element is

$$N(a, \omega) = \frac{4H}{\pi a} \left[ \sqrt{1 - \left( \frac{D}{a} \right)^2} - j \left( \frac{D}{a} \right) \right]$$

The values of the parameters of the linear and nonlinear elements are

- $b_1 \in [103.9705, 110.9238]$, $a_1 \in [0.00734, 0.0130]$
- $a_2 \in [0.1356, 0.1818]$, $a_3 \in [1.0, 1.0]$
- $\alpha_1 \in [1.6062, 1.6807]$, $\alpha_2 \in [0.7089, 0.8421]$
- $H \in [4.5, 5.5]$, $D \in [1.0, 1.2]$

The nominal values for the parameters are the midpoints of the corresponding intervals except for $D$ and are as shown below.

- $b_1 = 107.44715$, $a_1 = 0.01017$, $a_2 = 0.1587$, $a_3 = 1.0$
- $\alpha_1 = 1.64345$, $\alpha_2 = 0.775494$, $H = 5$, $D = 1$

The objective is to construct the limit cycle locus for each uncertain parameter, to prescribed accuracy $\epsilon = 0.01$.

We apply the proposed algorithm to construct the limit cycle locus for each uncertain parameter. The initial search box for the limit cycle is taken as $Y^0 = ([0, 100], [0.01, 1000])$.

The results obtained from the algorithm are plotted in Fig. 2 to Fig. 9. From the limit cycle plots in Fig. 10 to Fig. 17, we can make the following observations. For each uncertain parameter, there is a single branch of the limit cycle locus corresponding to stable limit cycle points. The limit cycle frequency decreases with $a_1, \alpha_1, D$ but increases with $b_1, a_2, \alpha_2$ and $H$. The limit cycle amplitude decreases with $a_2, \alpha_2$ but increases with $b_1, a_1, \alpha_1, H, D$. Both the limit cycle frequency and amplitude increase with $b_1, H$. Note that since $a_3$ parameter has been fixed at unity and so does not vary, there is no variation in the limit cycle frequency and amplitude for this parameter. Table-1 gives the various performance metrics of the proposed algorithm in this example, in terms of the number of solution boxes generated to enclose the limit cycle locus for each uncertain parameter, the number boxes examined, and the computational time. We thus see that the proposed algorithm is readily able to compute all the limit cycle points in the given search domain in a few seconds.
Example 2. Consider again the nonlinear feedback system in Fig. 1, where the transfer function of the linear fractional-order element is \((Cois et al. (2002))\)

\[
p(s) = \frac{q_1 s^{1.648} + q_2 s^{0.824} + q_3}{q_4 s^2 + q_5 s^{1.648} + q_6 s^{0.824} + q_7}
\]

and the nonlinear element consists of relay with hysteresis whose describing function is

\[
N(a, \omega) = \frac{4H}{\pi a} \left[ \sqrt{1 - \left( \frac{D}{a} \right)^2} - j \left( \frac{D}{a} \right) \right]
\]

The nominal parameter values of the linear and nonlinear elements are

\[
q_1 = 0.054, q_2 = 0.271, q_3 = 0.466, q_4 = 1.0, q_5 = 3.15, q_6 = 2.84, q_7 = 0.367, H = 5, D = 1
\]

Parametric uncertainty of 10% around each of the above nominal values is introduced. The first objective is to construct the limit cycle locus for each uncertain parameter to a specified accuracy \(\varepsilon = 0.001\).
1. LIMIT CYCLE FREQUENCY (rad/sec)

2. LIMIT CYCLE MAGNITUDE

3. TABLE 1. PERFORMANCE METRICS OF THE ALGORITHM IN EXAMPLE 1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Solution Boxes</th>
<th>Boxes Examined</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>6065</td>
<td>27,581</td>
<td>1,680</td>
</tr>
<tr>
<td>$a_1$</td>
<td>31,180</td>
<td>157,397</td>
<td>6,520</td>
</tr>
<tr>
<td>$a_2$</td>
<td>1,141</td>
<td>2,963</td>
<td>200</td>
</tr>
<tr>
<td>$a_3$</td>
<td>1</td>
<td>41</td>
<td>1</td>
</tr>
<tr>
<td>$a_1$</td>
<td>34,595</td>
<td>160,827</td>
<td>8,540</td>
</tr>
<tr>
<td>$a_2$</td>
<td>2414</td>
<td>10,207</td>
<td>850</td>
</tr>
<tr>
<td>$H$</td>
<td>12,752</td>
<td>52,871</td>
<td>2,880</td>
</tr>
<tr>
<td>$D$</td>
<td>10171</td>
<td>57,167</td>
<td>1,780</td>
</tr>
</tbody>
</table>

We set the initial search box for the limit cycle as $y^0 = ([0, 100], [0.01, 100])$, and apply the proposed algorithm. The limit cycle loci are shown in Fig. 10 to Fig. 17. From the limit cycle plots in Fig. 10 to Fig. 17, we can make the following observations. For each uncertain parameter, there is a single branch of the limit cycle locus corresponding to stable limit cycle points. The limit cycle frequency decreases with $q_2$ and $q_6$ but increases with $q_3$, $q_7$ and $H$. There is a very slight increase of limit cycle amplitude with $q_1$, $q_4$ and $q_5$. The limit cycle amplitude increases with $q_2$, $q_3$ and $q_6$ but decreases with $q_6$ and $q_7$. Both limit cycle frequency and amplitude increase with $q_3$ and $H$ and both decrease with $q_6$. Table-2 gives the various performance metrics of the proposed algorithm in this example, in terms of the number of solution boxes generated to enclose the limit cycle locus for each uncertain parameter, the number of boxes examined, and the computational time. We thus see that the proposed algorithm is readily able to compute all the limit cycle points in the given search domain in a few seconds.

7. CONCLUSIONS

The tool of limit cycle locus is clearly useful in gaining insight into the effect of each uncertain parameter on the limit cycle behavior. The proposed algorithm computes this locus reliably and accurately, for a very wide class of
linear and nonlinear elements of fractional-order and non-linearities of the type frequency dependent/independent, with memory/without memory. It must be noted that any errors resulting in the limit cycle locus due to the approximate nature of the describing function method cannot be avoided.

REFERENCES


<table>
<thead>
<tr>
<th>Parameter</th>
<th>Solution Boxes</th>
<th>Boxes Examined</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>54</td>
<td>6,577</td>
<td>780</td>
</tr>
<tr>
<td>$q_2$</td>
<td>342</td>
<td>8,637</td>
<td>1,170</td>
</tr>
<tr>
<td>$q_3$</td>
<td>727</td>
<td>10,467</td>
<td>1,590</td>
</tr>
<tr>
<td>$q_4$</td>
<td>729</td>
<td>11,543</td>
<td>1,710</td>
</tr>
<tr>
<td>$q_5$</td>
<td>2853</td>
<td>22,919</td>
<td>3,670</td>
</tr>
<tr>
<td>$q_6$</td>
<td>2965</td>
<td>24,251</td>
<td>3,260</td>
</tr>
<tr>
<td>$q_7$</td>
<td>372</td>
<td>8,385</td>
<td>1,060</td>
</tr>
<tr>
<td>$H$</td>
<td>3281</td>
<td>34,549</td>
<td>6,140</td>
</tr>
</tbody>
</table>


Appendix A. EXAMPLE TO ILLUSTRATE HC4

Let us consider a constraint \( C : x^2 + y^2 = 1 \). Here the variable set is \( \{x, y\} \). Let the initial search domain for \( x \) and \( y \) be \([-10, 10]\). First step is to form the tree using constraint as shown in Fig. A.1. The operands and constants occupy the leaf nodes whereas the operators are be placed at the parent nodes.

Fig. A.1. Binary tree constructed using the constraint \( C : x^2 + y^2 = 1 \)

The forward evaluation is done using the left-hand and right-hand parts of the equation using interval arithmetic, saving at each node the result of the local evaluation as shown in Fig. A.2. In the backward propagation step the expression tree is swept from top to bottom (see Figs. A.3, A.4 and A.5), the domains computed during the forward evaluation are used to project the relation at each node on the remaining variables. The values evaluated during backward propagation are shown in left side of the node in \textbf{bold}. First we start at the root node and from the constraint the right hand side should be equal to left hand side, so the value at the node \( O \) should be equal to \([1, 1]\). The updated tree is shown in Fig. A.3. The old values are shown in ellipses on the right side of the node and the new values are shown in bold on the left side of the node.

New value at node \( M \) is evaluated as given below.

\[
M_{\text{new}} = M_{\text{old}} \cap (O_{\text{new}} - N_{\text{old}})
\]

\[
= [0, 100] \cap ([1, 1] - [0, 100])
\]

\[
= [0, 100] \cap [-99, 1]
\]

\[
= [0, 1]
\]

New value at the node \( N \) is evaluated using the new value of node \( M \)

\[
N_{\text{new}} = N_{\text{old}} \cap (O_{\text{new}} - M_{\text{new}})
\]

\[
= [0, 100] \cap ([1, 1] - [0, 1])
\]

\[
= [0, 100] \cap [0, 1]
\]

\[
= [0, 1]
\]

The updated tree is as shown in Fig. A.4. The new evaluation of \( x^2 \) is the new value at the node \( M \)

\[
x_{\text{new}} = x_{\text{old}} \cap (M_{\text{new}})^{0.5}
\]

\[
= [-10, 10] \cap ([0, 1])^{0.5}
\]

\[
= [-10, 10] \cap [-1, 1]
\]

\[
= [-1, 1]
\]

Similarly the new value of \( y \) is evaluated as shown below.
Fig. A.4. Backward propagation - 2

\[ y_{\text{new}} = y_{\text{old}} \cap (N_{\text{new}})^{0.5} \]
\[ = [-10, 10] \cap ([0, 1]^{0.5}) \]
\[ = [-10, 10] \cap [-1, 1] \]
\[ = [-1, 1] \]

The updated tree is as shown in Fig. A.5

Fig. A.5. Backward propagation - 3

After backward propagation is complete, the search domain contracts i.e., the inconsistent search domain will be thrown out. The updated values of both \( x \) and \( y \) are [-1, 1]. The updated tree after complete backward propagation is shown in Fig. A.6

Fig. A.6. Updated tree after complete backward propagation

So the search domain (box) \( x \times y \) shrinks from \([-10, 10] \times [-10, 10]\) to \([-1, 1] \times [-1, 1]\) after one iteration of HC4.