Abstract – A new method of interval computations using fuzzy sets and stochastic processes is proposed. It is concerned with the development of a practical technology that can be used by engineers and researchers to facilitate the process of decision making under uncertainty, in a state of “bounded rationality”. The methodology called Interval Resolution considers a model and interval values specified by the researcher for each model parameter of interest, and returns the feasible interval ranges for all model’s parameters. The process continues with the researcher refocusing on different subsets of parameters as needed and can end up with collapsing some intervals. Fuzzy sets and Monte Carlo methods are used to compute the feasible ranges. A modeling software, Interval Solver, and a descriptive language are proposed and implemented along with the IR Integrated Development Environment. A number of applications have been considered. Theoretical and experimental analysis of computational efficiency is under way.

Keywords: Interval computations, models with uncertainty, fuzzy sets, decision making, bounded rationality

1 Introduction

In general field of modeling and engineering, researchers and designers always deal with uncertainties in setting model parameters. Making decisions about the proper parameters and factors requires interval analysis. Researchers in the mathematical modeling and engineering design community have identified interval analysis as a potentially powerful design tool, and called for their further development [1, 2, 3]. However, efficient practical methods are still to be devised that can handle the complexities of models used in many engineering applications.

Here we present a technology called Interval Solver that can facilitate finding the relations between the model’s characteristics, with uncertainties given as conditions of the model. Using Interval Solver, one can define the model in the form of implicit interval equations. Initial estimates of the intervals are given by the modeler. By changing the intervals of uncertainties for a subset of the variables, one can find out how the other intervals are affected. New feasible intervals are computed by Solver. Now, the modeler can decide on what intervals are to be controlled next and make the appropriate changes. A new set of feasible intervals for all variables are recalculated by Solver. The process continues until the intervals (maybe, completely collapsed) are found that are considered as the most agreeable with the objectives of the engineering design or the model validation.

The underlying concept of this methodology is in compliance with the principle of bounded rationality developed by Herbert Simon [4] who identified the importance of the fact that in the presence of uncertainty, humans will attempt to isolate a closed subsystem that contains only a limited number of parameters and a limited range of consequences. Modeling tools should assist and improve the effectiveness of this human response to bounded rationality.

The Interval Solver methodology can be applied to various modeling situations. As just a single example, consider chemical and biological modeling using kinetic equations. The experimental results are often used to validate the model; however, the observed concentrations of the reagents and other measurable quantities leave the researcher with a significant uncertainty in the ways of how to interpret the results. In [5], certain chemical kinetics equilibrium problems have been solved correctly with interval techniques applied to analysis of sensitivity of such characteristics as the dimensionless numbers, some scalars, such as the max/min concentrations, or max/min spatial and temporal derivatives, or concentrations (spatial/temporal derivatives) at the boundaries of the chemical reactors, and others.

In engineering, designers often use bounded parameters during the preliminary design phase to provide flexibility, and/or to express uncertainty about what the design
component “should look like”. Resolving intervals of uncertainty helps to devise a proper set of parameters adjusted ad hoc.

The main difficulty of finding intervals satisfying the model is in solving the inverse model, namely, in finding the feasible ranges of parameters and input variables of the model given the intervals of its output values. By changing an interval for one of the variables, any other intervals become affected and should automatically change to comply with the new intervals.

The methods of interval and soft computing seem to be appropriate for this problem. There is a significant body of research in this field (a comprehensive analysis of the state of art in the interval and soft computing field can be found in [6]).

The approach implemented in Interval Solver is based on combining the fuzzy sets presentation of intervals and stochastic modeling that work well with interval computations.

Use of fuzzy sets and statistical method of resolving intervals makes it a robust method capable of handling “difficult” model characteristics such as nonlinearity, discontinuity, non-algebraic functions, etc [7].

2 Interval Resolution

We consider a model with \( N \) degrees of freedom represented by variables \( x_i, i \in [1, N] \). Each variable is given with its interval, \( X_i = [x_{i_{\min}}, x_{i_{\max}}] \). There is an operator, \( \mathcal{R} \), applied to a \( k \)-th set of intervals \( \{X_i\}^k \) to produce an updated set of intervals \( \{X_i\}^{k+1} = \mathcal{R}(\{X_i\}^k) \) such that any repeated applications of \( \mathcal{R} \) to the same set of intervals does not change them thus approaching the fixed point in interval resolution. We call intervals corresponding to fixed point balanced or feasible.

It is assumed that the model the interval analysis is applied to is a mathematical description of a problem which can be presented as a function \( F(V) \) that takes some inputs as a mathematical vector \( V \) to produce an output vector \( W \):

\[
W = F(V).
\]

Figure 1 depicts this relationship.

This is considered to be a model of a problem for which an appropriate set of values \( V \) should be found to satisfy a certain constraints for output \( W \). The typical process is based on the What If paradigm. What happens to \( W \) if a value of \( V \) is somehow changed? Again, more typically, in such analysis one keeps the output \( W \) within a specific interval of values while trying different combinations of input values \( V \).

\[
\Phi(X) = 0
\]

Figure 1. A typical process of modeling and sensitivity analysis

With the interval method, the decision making is the process where both input \( V \) and output \( W \) are treated similarly. Both input and output variables belong to the same set of variables \( X = V \cup W \) thus allowing to use input and output variables interchangeably: \( \Phi(X) = 0 \) where implicit function \( \Phi(X) = F(V) - W \).

Figure 2. The interval based sensitivity analysis

As Figure 2 shows all variables can be used interchangeably either as input or output, so it is possible to use either \( V \) or \( W \) to see how they affect each other.

Each variable is taken from a given interval. The proposed decision making process finds all possible combinations of \( V \) and \( W \) that satisfy the equation \( \Phi(X) = 0 \), thus producing all feasible subintervals defining all possible solutions.

Example. Given

\[
y - ax^2 = 0 \ (x, y \in X)
\]

and intervals from which \( x \) and \( y \) can be taken, find all possible pairs \( (x, y) \) satisfying the given equation.

Let \( a = 1 \), given intervals for \( x = [-3,3] \) and for \( y = [1,4] \), the interval resolution produces the following feasible intervals for \( x \):
[-2,-1], [1,2] (the original interval is split onto two subintervals)

and for y:

[1,4] (this interval is not changed by this process).

This result can be interpreted as application of the operator ạ to given intervals for x and y:

\{ [-2,-1], [1,2], [1,4] \} \mapsto \{[-3,3], [1,4]\}

One can consider any values in the resulted intervals for both x and y as productive in the sense that there is at least one solution trajectory that passes through the point represented by this value. Now, any desirable changes can be made in the obtained intervals and applying the operator ạ again, thus producing a new set of intervals until the desirable combination of x and y can be found.

The process involved in the single ạ application converges to the completely balanced set of intervals.

The proposed interval analysis uses Monte Carlo method as a general approach to any problem having N degrees of freedom represented by variables \(x_i, i \in [1, N]\). Each variable is given with its interval, \(X_i = [x_{i_{\text{min}}}, x_{i_{\text{max}}}]\). Randomly selected values from the \(X_i\) intervals are processed to calculate the left hand side function in the \(\Phi(X) = 0\) equation. The selected values from the \(X_i\) intervals are given the weights from 0 to 1 depending on how close the calculated function approaches 0. This represents another important feature of the method – the use of fuzzy sets.

The membership function, \(\mu\), for the implicit non-linear model of \(M\) equations:

\[
\Phi_k(x_1, x_2, \ldots, x_N) = 0, \quad i = 1, 2, \ldots, M
\]

represents the "soft" intersection of the fuzzy sets for each equation (the product of their membership functions) and is calculated as a single multidimensional bell curve:

\[
\mu = \prod_{k=1}^{M} e^{-\sum_{i=1}^{N} F_k^2(x_{1}, x_{2}, \ldots, x_{N})/\sigma^2} = e^{-\sum_{i=1}^{N} F_k^2(x_{1}, x_{2}, \ldots, x_{N})/\sigma^2}
\]

To estimate the performance of the method let’s start with finding a specific point \(x_0\) with and resolution \(\Delta x\) in an interval of \(x\) having the unit length.

If the uniform distribution of probability is used then the probability of finding the point \(x_0\) is

\[
\Pr[x = x_0] = \Delta x.
\]

Next, consider a two variable problem, such as in Example above. The probability of finding a point in the feasible interval will be given as a ratio of the total length of feasible intervals to the length of the original interval. In the case of \(x\), it will be \(\Pr[x \in [-2,-1] \cap [1,2]] = 4/6\).

The probability of finding a bound for a variable \(x_0\) of a feasible interval in a system having \(n\) degrees of freedom is given as the product of probabilities \(\Pr_i\) to find a point in \(n-1\) feasible intervals times the probability \(\Pr_0\) to find a single point in the given interval. Assuming that all original intervals are normalized to the unit length then the probability to find a bound is:

\[
\Pr[x_0 = \text{bound}] = \Delta x \prod_{i=1}^{n-1} \Delta l_i
\]

where \(\Delta l_i\) is a feasible interval for \(x_i, (0 < \Delta l_i \leq 1\).

Assuming \(\Delta l_i = \frac{1}{2}\) and \(m = 1/\Delta x\), the previous formula becomes

\[
\Pr[x_0 = \text{bound}] = 2^{n-1}/m.
\]

Some numerical examples illustrate this analysis as shown further. The major advantage of using the Monte Carlo method applied to the fuzzy set of a non-linear model is the fact that it seems working (1) with any type of mathematical models, (2) converges much faster than other deterministic methods and, at the same time, (3) provides the full scope of the feasible intervals even if the original intervals become split onto many subintervals after applying operator ạ.

### 3 Graphical Interface

Graphical user interface is a significant part of Interval Solver technology. The intervals both original and feasible are produced under user’s control. Given initial intervals for variables \(x\) and \(y\)

\((x \in [-1,1], y \in [-2,2])\)

in the above quadratic model, the feasible intervals for the both variables can be depicted in many graphical forms. The linear diagrams as shown in Figure 3(a, b) are one of the possible graphical methods.

The feasible intervals found as \((x \in [-1,1], y \in [-0,1])\) are depicted as thick lines while the thin lines represent the original intervals (Figure 3, a).
Figure 3. Interval resolution for a quadratic equation

The user can try other intervals. For intervals $[-2; 1.2]$ the solution $(x \in [-1.41, -1] \cap [1,1.41], y \in [1,2])$ obtained is shown in Figure 3, b.

Another simple example concerns the relationship between the intervals of variables in the dimensionless expression for diffusivity in chemical reactions:

$$D = \frac{D_0 \varepsilon}{uLT}$$

with six variables,

$D, D_0, \varepsilon, u, L, T$.

A set of initial (arbitrary) intervals can be set. An example of the interval distribution, both initial and feasible is shown in Figure 4.

This small model can be resulted from the decomposition of a larger chemical model including differential equations of chemical kinetics. Each part of the model can be resolved separately and then iteratively matched with other submodels.

Figure 4. Interval resolution for diffusivity equation

4 Interval Solver

Interval Solver is a technology developed to facilitate the Interval Resolution methodology. Several algorithms were investigated as potential candidates for use by the technology.

Simple Monte Carlo Algorithm

This basic implementation of the Interval Solver method conducts $N$ random walks. Each walk takes as input a system of implicit equations, i.e. $\Phi(X) = 0$, and randomly generated values for each parameter of the model, i.e. the solution vector $X$. The values are selected so that they are within the corresponding user prescribed intervals for each parameter in $X$. The values in $X$ are applied to each equation of the model. The results of this application are then evaluated using the soft membership function according to [7]. This results in a fuzzy membership value expressing how “close” $X$ is to being an exact solution (i.e. generating $\Phi(X) = 0$). The solution, and its corresponding fuzzy membership value are stored in a data structure for graphical presentation after the $N$-th random walk. Below is a diagram of the algorithm:

Bubble Monte Carlo Algorithm

The bubble Monte Carlo approach improves on the simple Monte Carlo algorithm efficiency by guiding the random walk process over successive iterations, or epochs. In the beginning, $K$ number of solutions are generated. The minimum and maximum bounds for each variable are tested during this first iteration. Therefore, it is a requirement that $K > 2V$, where $V$ is the number of parameters in the model.

The solution vectors for these boundary tests are randomly generated and saved for the boundary test variable which is assigned the maximum or minimum bound. The remaining $K-2V$ solution vectors are generated at random, using the prescribed intervals. The solution vectors are evaluated and fuzzy membership estimated in the same manner as the simple Monte Carlo algorithm, and the algorithm proceeds through $N$ epochs.

Each epoch has a certain threshold value, $\mu_T$, which is assigned using a function of the current epoch.
solution vectors are evaluated and fuzzy membership estimated, the solution vectors for the next epoch are assigned using $\mu_T$ and the solution vectors from the current and previous epochs. More specifically, say $\mu_{k,n}$ is the fuzzy membership value for the $k^{th}$ solution vector of the $n$-th epoch ($n$ being the current epoch). Then, assigning values for the $k$-th solution vector of the $n+1$ epoch uses the following decision structure:

- If $\mu_{k,n-1} < \mu_{k,n}$ and $\mu_{k,n} \geq \mu_T$
  Then the current epoch improved on the previous epoch’s solution, and is above $\mu_T$. So permute each parameter of the $k^{th}$ vector of the current epoch ($n$) by a random amount. This permutation is a function of $\mu_T$ so the permutation amount decreases as the epochs increase.

- If $\mu_{k,n-1} \geq \mu_{k,n}$ and $\mu_{k,n} \geq \mu_T$
  Then the current epoch did not improve on the previous epoch’s solution and is above $\mu_T$. So permute each parameter of the $k^{th}$ vector of the previous epoch ($n-1$) by a random amount.

- If $\mu_{k,n} < \mu_T$
  Then the current epoch solution vector has membership below $\mu_T$. So randomly assign each parameter of the $k^{th}$ vector to be between the initial minimum and maximum bounds.

After assignment, the vectors are evaluated using the model, and fuzzy membership is estimated. The solution vector and its membership are stored in a data structure for graphical display after the last epoch, and then the next iteration proceeds.

The algorithm has a higher efficiency than the simple algorithm. This is due to the fact that the permutation function and solution vector acceptance are both functions of $\mu_T$. More specifically, large changes in the solution vectors are allowed in the first epochs, while $\mu_T$ is low; thereby allowing a comprehensive scan of the entire range. As epochs increase, and $\mu_T$ increases towards 1, the permutation amount decreases; thereby allowing for greater exploration of the feasible solution intervals, and less exploration of the infeasible regions.

As described in the next section, this method was found to be more efficient than the basic Monte Carlo approach.

### 5 Experimental results

The properties of the above Interval Solver algorithms were examined using various statistical methods. Some of them are discussed here for the same quadratic model as above (Figure 3 a).

### Analysis of Simple Monte Carlo Algorithm

The “Simple Monte Carlo” algorithm was used as baseline for a general analysis of the method, and benchmark for the more advanced algorithms.

The random walk is a significant part of the Interval Solver algorithms. A random vector $X = [x, y]$ and the corresponding fuzzy set’s membership function, $\mu$, are evaluated.

The graphs of the envelopes of $\mu(x)$ are shown in Figure 6. The peaks are the points where the variables produce the best fit for the given model. The peaks equal to 1.0 represent the points belonging to the sought feasible interval. The peaks lower than 1.0 belongs to the fuzzy interval of solutions.

![Figure 6. The $\mu(x)$ envelopes with number of samples 1000 and 10,000](image)

In the case of 10,000 samples, statistical accuracy of funding the interval boundaries = 0.003/4=0.1%. Exhaustive search with the same number of steps (100 by 100) has the accuracy 4/100=4%. (The boundaries are presented as data tips on the graph.) This result agrees with the theoretical analysis. In order to improve on the simple algorithm’s efficiency, we noted the following observations and relationships:

1. As demonstrated in the table below, reducing the exact solution membership value increases the probability that the random walk will encounter an exact solution (as defined by the lowered membership value). The probability of finding at least one exact solution, with reduced accuracy, belonging to the feasible intervals is estimated as a deviation $\Delta\mu = 1.0 - \mu$:

<table>
<thead>
<tr>
<th>$\Delta\mu$</th>
<th>0.001</th>
<th>0.1</th>
<th>0.2</th>
<th>0.4</th>
<th>0.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pr(\Delta\mu)$</td>
<td>0.0128</td>
<td>0.135</td>
<td>0.2</td>
<td>0.3</td>
<td>0.42</td>
</tr>
</tbody>
</table>

Table 1: Reduced Accuracy ($\Delta\mu$) versus exact solution encounter probability ($\Pr(\Delta\mu)$)
Numerous experiments confirm the linear character of the probability of finding a single solution as a function of its accuracy.

Corollary to this then is the notion that increasing the exact solution membership value reduces the probability that the random walk will encounter non-exact solutions.

2. The probability of finding, at least, a single (exact) solution vs. the model’s space dimension, \( N \), (number of degrees of freedom) is confirmed to be the following function:

\[
\Pr(\Delta \mu = 0.001) = a / 2^{N-2}.
\]

(In the case of the above quadratic model, \( a = 0.0128 \).)

3. Since we deal with the one dimensional search space the probability of finding at least one point with \( \mu \) close to 1 is high:

\[
\Pr[ x_0 = \text{bound} ] = \Delta x \prod_{i=1}^{n-1} \Delta I_i.
\]

Analysis of the Bubble Monte Carlo Algorithm

These results imply that it is possible to develop an algorithm which identifies and expands the feasible intervals by slowly increasing \( \mu \) (decreasing \( \Delta \mu \)) so that the length of the subinterval(s) traversed by the random walks are iteratively decreased until only the final exact solutions remain. The previously described bubble Monte Carlo algorithm implements these findings.

The following table summarizes the efficiency of the various algorithms in identifying the feasible intervals for the same quadratic model as above (Figure 3 a). For the purposes of this experiment, we found the number of solution points required to “identify” the feasible interval length to within 98% of the analytically derived length of 2.0 for \( X \), and 1.0 for \( Y \).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Solutions Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Monte Carlo</td>
<td>3500</td>
</tr>
<tr>
<td>Bubble Monte Carlo</td>
<td>1500</td>
</tr>
</tbody>
</table>

Table 2: Efficiency of two algorithms described above

6 Interval Solver (software)

A software implementing the Interval Solver methodology was developed to analyze the performance of the various Interval Solver algorithms, and to demonstrate how the proposed method might be practically implemented. Figure 7 shows an example of the Interval Solver GUI with a model’s run.

7 Conclusion

We presented here a new method, Interval Solver, that provides an efficient, robust methodology of interval analysis through its novel use of fuzzy sets in the Monte Carlo simulation settings.

8 References


