# A Verified MATLAB Toolbox for the Dempster-Shafer Theory

Ekaterina Auer, Wolfram Luther, Gabor Rebner, Philipp Limbourg
Computer and Cognitive Sciences (INKO)
University of Duisburg-Essen
Duisburg, Germany
Email: {auer, luther, rebner}@inf.uni-due.de, p.limbourg@uni-due.de

***Abstract*—The importance of the Dempster-Shafer theory (DST) for modeling and propagating uncertainty has grown in the recent past. An obstacle for wider application of this theory in industrial practice is the lack of software support for analysts. The few tools available depend on floating point arithmetic and do not consider the inherently interval-based nature of the DST to the full extent. Therefore, the obvious next step is to combine the DST ideas with those from interval arithmetic. An additional advantage of employing interval methods is the guarantee that the results obtained on a computer are mathematically correct.**

**In this paper, we introduce a new verified DST implementation for MATLAB based on the previously developed IPP TOOLBOX. It extends this software using interval arithmetic and simultaneously takes care of the rounding errors. After giving a short overview of the Dempster-Shafer theory and interval methods, we describe the main features of the new toolbox and show its potential using several examples.**

**Keywords: Dempster-Shafer, interval arithmetic, MATLAB, INTLAB.**

## I. INTRODUCTION

The experience of the last decades shows that while the design process in many application fields becomes shorter due to time-to-market pressure, the requirements on numerical accuracy and performance grow stricter. However, engineers lack precise knowledge regarding the process and its input data in early design stages. Therefore, to assess how reliable a system is, they have to deal with uncertainty. That is the reason why methods to propagate uncertainties through the system gain more and more importance.

The overall imprecision in the outcome can be specified by providing upper and lower bounds on all possible results using interval or other verified methods. As further options, probability theory, the Dempster-Shafer theory or the Bayes theory can be used. In this paper, we concentrate on the use of the Dempster-Shafer theory (DST), the significance of which for modeling and propagating uncertainty has grown recently [1]. However, the few existing DST implementations [1], [2], [3] rely on floating point arithmetic and do not exploit to the full extent the inherently interval based nature of the theory.

A method is called *verified* if it guarantees the correctness of its output. In this context, interval arithmetic [4] is a widely used approach to verifying results obtained on a computer. It provides a (multidimensional) box described in terms of floating point arithmetic which is guaranteed to contain the exact result. Besides allowing for uncertainty in parameters, interval arithmetic helps to generate more realistic mathematical models or to take into account measurement errors.

In this paper, we describe a new DST implementation that makes use of the advantages of interval methods. The DSI TOOLBOX (Dempster-Shafer with intervals) is a MATLAB package and based on the software IPP TOOLBOX [1] available in MATLAB [5] and R [6]. The new implementation is designed to define, aggregate and evaluate precise Dempster-Shafer structures by using directed rounding and verified methods made accessible in MATLAB by the INTLAB library [7].

This paper is structured as follows. First, we give a brief introduction to the fundamentals of the DST and the interval methods. In the next section, we describe the main features of the DSI TOOLBOX and discuss several basic usage examples after a short overview of the IPP and INTLAB libraries. Over the course of section IV, we exemplify the applicability of the new software using tasks from fault tree analysis and non-monotonous uncertainty propagation. We conclude by recapitulating the main results and providing a perspective for future research.

## II. FUNDAMENTALS

In this section, we describe the fundamentals of the Dempster-Shafer theory and interval methods briefly.

### A. The Dempster-Shafer Theory

The Dempster-Shafer theory [8] allows us to combine evidence from different experts or other sources and provides a measure of confidence that a given event occurs. A special feature of this theory is the possibility to characterize uncertainties arising because of the lack of knowledge as discrete probability assignments associated with the power set of values $X$. Due to the presence of imprecision, it is only possible to compute a lower and an upper bound (belief and plausibility) of the probability of a subset of $X$.

The DST equivalent of a random variable is the belief variable, which is characterized by its basic probability assignment (BPA) $m$. If $A_1, \ldots, A_n$ are the sets of interest where each $A_i \in 2^X$, then

$$m : 2^X \to [0,1], \quad \sum_{i=1}^{n} m(A_i) = 1, \quad m(\emptyset) = 0. \tag{1}$$

In the continuous case, we restrict the sets $A_i$ to intervals of the form $([\underline{x}, \overline{x}])$, $\underline{x} \leq \overline{x}$, where $\underline{x}$ denotes the lower bound or infimum and $\overline{x}$ the upper bound or supremum, for computational simplicity. Such intervals can be considered as evidence by an expert, for example, [10,20] hours lifetime for a sensor. The restriction in (1) concerning the sum of masses shows the necessity to normalize real life evidence because experts tend to provide BPAs for which it does not hold.

The plausibility ("worst case") and belief ("best case") functions can be defined with the help of the BPAs for all $i = 1 \dots n$ and $Y \subseteq X$as

$$PL(Y) := \sum_{A_i \cap Y \neq \emptyset} m(A_i), \quad BEL(Y) := \sum_{A_i \subseteq Y} m(A_i). \quad (2)$$

Every element $A$ with a mass unequal zero is known as a focal element. All focal elements including their masses are called a random set. If two or more experts provide different estimations in the same areas, the BPAs have to be aggregated. There exist several methods for this purpose [8], of which Dempster's rule and mixing based on arithmetic averaging are those used in this paper.

### B. Interval arithmetic

Interval arithmetic ( [4] and the references therein) is a well developed field of mathematics with applications in many areas of engineering, medical science, (bio)mechanics and others. It belongs to the group of the verified methods, that is, methods that guarantee the correctness of the outcome of a simulation using mathematically exact proofs.

An interval $[\underline{x}, \overline{x}]$, where $\underline{x}$ is the lower, $\overline{x}$ the upper bound, is defined as $[\underline{x}, \overline{x}] = \{x \in \mathbb{R} | \underline{x} \leq x \leq \overline{x}\}$. For any operation $\circ = \{+, -, \cdot, /\}$ and intervals $[\underline{x}, \overline{x}]$, $[\underline{y}, \overline{y}]$, the corresponding interval operation can be defined as $[\underline{x}, \overline{x}] \boxed{\circ} [\underline{y}, \overline{y}] = [\min(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y}), \max(\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y})]$. It can be shown that the result of an interval operation is also an interval. Every possible combination $x \circ y$ with $x \in [\underline{x}; \overline{x}]$ and $y \in [\underline{y}; \overline{y}]$ lies inside this interval. (For division of intervals, usually $0 \notin [\underline{y}, \overline{y}]$ is assumed.)

To be able to work with this definition on a computer using a finite precision arithmetic, the concept of machine intervals is necessary. They are represented by floating point numbers for the lower and upper bounds. To obtain the corresponding machine interval for the real interval $[\underline{x}, \overline{x}]$, the lower bound is rounded down ($\bigtriangledown$) to the largest representable machine number equal or less than $\underline{x}$, and the upper bound is rounded up ($\bigtriangleup$) to the smallest machine number equal or greater than $\overline{x}$. These notions can be extended to define interval vectors and matrices.

There is a number of software libraries implementing this theory in different programming languages such as C++ or FORTRAN and computer algebra packages such as MAPLE or MATLAB.

## III. DSI TOOLBOX — THE DEMPSTER-SHAFER THEORY WITH INTERVAL ARITHMETIC

The main focus of this section is the newly implemented DSI TOOLBOX which combines the DST approach with rigorous interval methods. This toolbox is implemented in MATLAB and uses INTLAB for basic interval functionalities. After a short overview of the IPP TOOLBOX from which the DSI software originated, we outline the functionalities of INTLAB used by the DSI and finally describe the new toolbox in detail and with usage examples.

### A. IPP TOOLBOX

The Imprecise Probability Propagation (IPP) Toolbox [1] is a collection of methods for uncertainty quantification and propagation in the framework of the Dempster-Shafer theory and imprecise probabilities. This library uses floating point arithmetic as a computational basis. It is available as a MATLAB version [5] and an extended R package [6]. The IPP TOOLBOX contains a broad range of methods for practical application of the DST such as construction of belief functions from bounds on distributions, computation of empirical BPAs from data, evaluation of BPA fits using Kolmogorov-Smirnov tests, various aggregation methods, propagation through arbitrary monotonous and non-monotonous system functions and computation of statistical properties. The toolbox was applied in several case studies, such as [9], [10].

The feature we would like to concentrate on is the propagation of uncertainties through non-monotonous system functions. This could be a major obstacle for using DST in practice. In most applications, the analysts either employ a Monte Carlo sampling approach or simply propagate all focal elements of (small) discrete BPAs through the system. In each case, a large amount of intervals need to be propagated through the system function $F(x)$. In the conventional DST, this is formally done by solving two optimization problems for each (multidimensional) interval $[\underline{x}, \overline{x}]$:

$$[\underline{F}, \overline{F}] = F([\underline{x}, \overline{x}]) = [\min_{x \in [\underline{x}, \overline{x}]} F(x), \max_{x \in [\underline{x}, \overline{x}]} F(x)].$$

The amount of computing power required in this step varies widely with the complexity of the function representing the system model. While monotonous functions do not increase it substantially, complex, non-monotonous ones render the propagation task very time consuming.

The IPP supports three different propagation algorithms:
**Monotonous:** A fast algorithm for monotonous functions (increasing/decreasing/mixed). Only the two point values $F(\underline{x})$, $F(\overline{x})$ have to be evaluated because of the monotony property.
**Regular:** An approximation algorithm that evaluates the function value in all corners of the focal element. The assumption is, that the extrema of $F([\underline{x}, \overline{x}])$ are located on the boundary of $[\underline{x}, \overline{x}]$.
**Optimization:** An algorithm compatible with arbitrary non-monotonous functions, which uses gradient descent optimization to propagate focal elements.

In case of non-monotonous functions, the *regular* algorithm is fast but provides only a coarse approximation of the true propagation result, whereas the *optimization* algorithm can be very time consuming, as one gradient descent run is carried out for each interval propagated. In section IV-B, we show that such computations can be performed more effectively using interval methods in the newly developed DSI-TOOLBOX.

### *B.* INTLAB — *Interval Laboratory*

INTLAB [7] is a MATLAB library implementing the basics of (multiple precision) interval arithmetic for real and complex numbers (cf. section II-B) along with providing linear algebra methods for intervals. Besides, it features automatic differentiation up to the second order, rigorous real and complex interval standard functions, accurate summation, dot product and matrix-vector residuals.

We can use either the infimum-supremum notation $[\underline{x}, \overline{x}]$ or midpoint-radius notation $\left\langle \frac{\underline{x}+\overline{x}}{2}, \frac{\underline{x}-\overline{x}}{2} \right\rangle$ for intervals. Point intervals in INTLAB can be defined by using the function `intval(x)`, where `x` is a real number. A mathematically correct interval enclosure of the real number $x$ that is not simultaneously a floating point number is given by `intval('x')`. By specifying the number as a string, we can ensure that the result is enclosed between the greatest machine number smaller and the smallest machine number greater than $x$.

One important feature of this library is directed rounding. With the help of the INTLAB function `setround(par)`, it is possible to take influence on the current rounding mode. If `par` is equal to one the rounding mode is set to positive infinity (or upwards), for `par=-1` to the negative infinity (or downwards) and for `par=0` the mode is set to round-to-nearest. This function can be used to generate an exact enclosure of a piece of evidence.

INTLAB supports a number of interval standard functions such as sine or cosine. Using them, it is easy to evaluate an interval enclosure of an arbitrary function given as a combination of standard ones and interval operations $\{\boxed{+}, \boxed{-}, \boxed{\cdot}, \boxed{/}\}$.

The current version of INTLAB is written completely in MATLAB to assure the ability to run identical code on different machines. The requirement for such portability is that the considered architecture uses IEEE754 arithmetic and can switch the rounding mode permanently. It is important to keep in mind that interval vector and matrix operations are fast in INTLAB due to its extensive employment of BLAS (basic linear algebra subsystems) routines. Rump shows in [11] that the unrestricted use of the midpoint-radius interval notation and BLAS type three leads to very fast algorithms. However, loops and nonlinear tasks slow down the computations which is characteristic of MATLAB as a whole.

### *C. Main Features of the* DSI-TOOLBOX

The IPP TOOLBOX described in section III-A uses floating point arithmetic and therefore does not exploit to the full extent the inherently interval nature of the DST. With this software as a basis, we developed a new verified implementation called DSI TOOLBOX (Dempster Shafer with intervals) for MATLAB to work with rigorous DST structures that rely on interval calculus and directed rounding. DSI contains both functions from the IPP TOOLBOX, which were rewritten to take into account all rounding errors and adjusted to intervals, and newly designed functions.

The main task of the new toolbox is to guarantee correctness of the solution. For that purpose, we take care of all rounding errors that might occur during the computation by enclosing real numbers in their corresponding machine intervals. Note that we do not take into account the modeling error present in DST or other probability based methods. A further goal is to provide enclosures with minimal overestimation, that is, with the minimum degree of conservativeness or pessimism. That is why we use sharp matrix multiplication provided by INTLAB which, however, needs slightly more CPU time.

Using DSI, we can define DST structures either directly by their focal elements with masses (routine `dsistruct`) or by cumulative distribution functions such as the triangular or the Weibull distribution (`dsitriangleinv`, `dsiweibullinv`). In the first case, the masses of the resulting structure have to be normalized according to Eq. (1).

The normalization is not trivial in our case because we represent masses as point or very tight intervals to provide verified solutions. However, it is not our goal at the moment to introduce interval masses with diameters considerably greater than a couple of ulps because it is difficult to think of a practical interpretation in this case. Our approach is as follows. If a random set consists of $n$ focal elements $A_1, \ldots, A_n$ with interval masses $m(A_i) = [\underline{m}(A_i), \overline{m}(A_i)]$, then the normalized masses $m^{new}(A_i)$ for $i = 1 \ldots n$ are computed as a hull of the two intervals

$$\underline{m}(A_i) \boxed{/} \left( \triangle \sum_{j=1}^{n} \underline{m}(A_j) \right), \quad \overline{m}(A_i) \boxed{/} \left( \triangledown \sum_{j=1}^{n} \overline{m}(A_j) \right).$$

To optimize the CPU time, we compute all steps using vector-matrix operations.

Let us consider further functionalities of the DSI TOOLBOX using the following example. Two experts give estimations about a robot failure. The first expert provides an assessment in form of a triangular distribution function. The important feature which the DSI TOOLBOX offers in this case is the possibility to define it with an uncertain mode and lower/upper bounds. To obtain the corresponding BPA, the user has to specify the number of samples to be computed. For a subdivision consisting of floating point numbers, which means a subdivision of purely point intervals in our framework, we recommend using $2^y$ samples with an arbitrary positive integer $y$. In Fig. 1, the solution space of the triangular distribution with lower bound $[1, 2]$, upper bound $[10, 12]$, mode $[4, 6]$ and $2^{12}$ samples is shown.

This space lies between the belief and the plausibility function defined in Eq. (2). To compute plausibility, all upper bounds of the masses are sorted in the ascending order and stored as point intervals in a matrix. To compute function

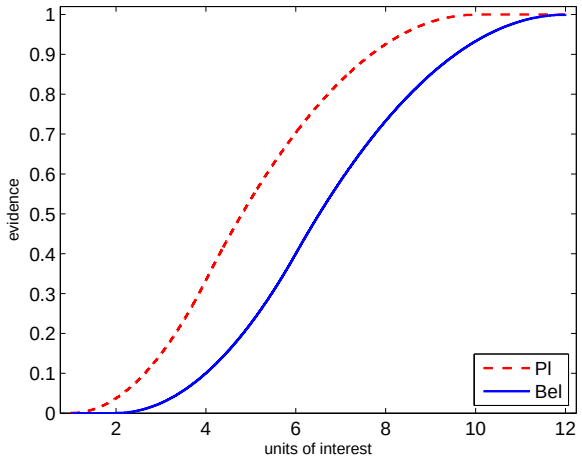


Figure 1. Triangular distribution with lb=$[1, 2]$, ub=$[10, 12]$ and mode=$[4, 6]$

values, we iterate from one to the length of the matrix adding the current element to the result of the last iteration. The solution array is sorted in the ascending order with the last element equal to one. The belief function can be computed in parallel to the plausibility by using the lower bounds.

The second expert provides an assessment in form of a BPA directly. Using the DSI TOOLBOX, we define the BPA by the routine

```
dsistruct(
   [infsup(1,3),2/6;infsup(1.5,6),1/6;
   infsup(5,15),3/6]).
```

Here, `infsup(x,y)` is the standard INTLAB function to define an interval in infimum-supremum notation. In this example, the first focal element $[1, 3]$ has the mass $2/6$, the second ($[1.5, 6]$) $1/6$ and the third ($[5, 15]$) $3/6$. In Fig. 2, the solution space of this BPA is shown.

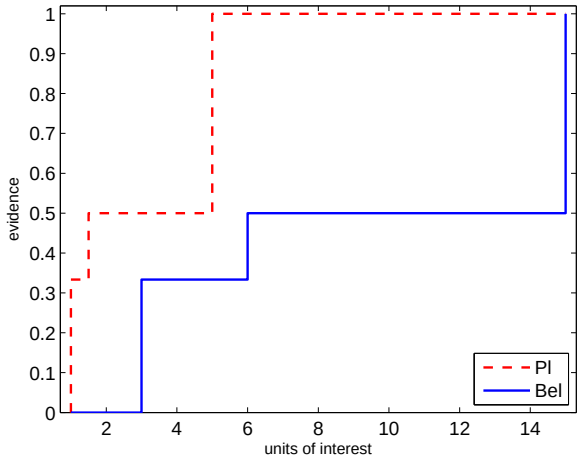


Figure 2. Solution space for expert two

To aggregate these two structures, we use Dempster's rule and mixing. DSI provides the routines `dsidempstersrule(x)` and `dsiwmixing(x,y)` where `x` is an array containing all BPAs (two in our example) and `y` is an array of corresponding weights.

The main issue with these two functions is the CPU time. It is acceptable for the weighted mixing because only matrix multiplication is in use. In contrast, Dempster's rule needs to be computed for every element in the first BPA giving a complexity of $O(n^2)$. In contrast to IPP TOOLBOX, we evaluate necessary intersections using matrices and the fast INTLAB function `intersect(x,y)` to accelerate computations.

In Fig. 3, the results of the application of Dempster's rule and unweighted mixing for the two BPAs from our example are shown. The BPAs and their aggregation by Dempster's rule are computed in 6.932 seconds on an Intel Core 2 DUO @ 2.1 GHz platform with 2 GB RAM. The overall CPU time for computing the BPAs and their unweighted mixing is 0.0925 seconds only on the same platform. Fig. 3 shows in addition that Dempster's rule aggregates only for intersecting focal elements while the mixing takes into account all of them.

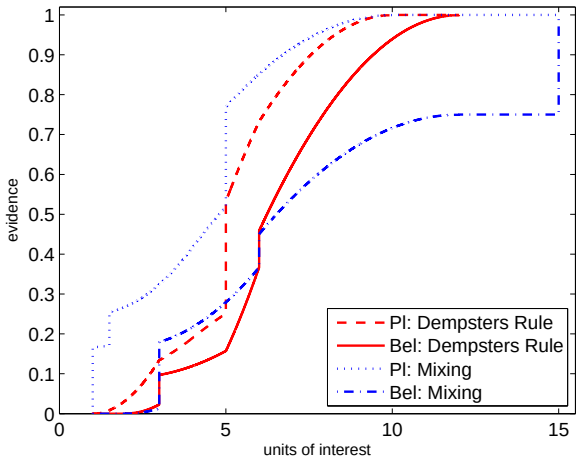


Figure 3. Aggregation by Dempster's rule and unweighted mixing

## IV. EXAMPLES

In this section, we first show briefly how fault tree analysis can be performed using the DSI TOOLBOX and then consider a simple example of uncertainty propagation through a non-monotonous system function. In each case, we compare obtained results with those from the IPP TOOLBOX.

### A. A Simple Example of Fault Tree Analysis

The Dempster-Shafer theory can be used in fault tree analysis [9]. Consider a robot arm consisting of three rigid links $L_1, L_2, L_3$ and three joints $J_1, J_2, J_3$ shown in Fig. 4. Each joint is driven by an autonomous motor. At the end of the arm, a utility is mounted which can only be manipulated by modifying the angles of the three joints.

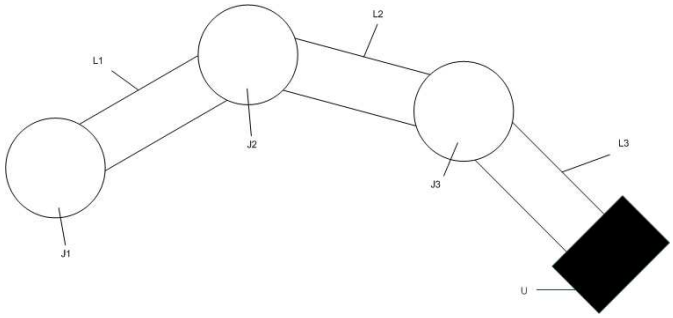


Figure 4. Robot with three joints

The distribution on the time to failure for each motor is assessed by three experts. The first expert estimates it for $J_1$ and $J_2$ to be in intervals $[700, 1400]$ and $[1000, 1500]$, respectively. The second expert provides an opinion in form of cumulative distribution functions for motor two and three. The failure distribution for motor two follows a triangular distribution between 700 and 1200 hours, with an unsure mode of [800,900], for motor three the same distribution in the interval [900,1500] hours with mode [1300,1400]. The third expert estimates motor three only and states that its time to failure is between 1000 and 1200 hours of work with 80 percent confidence and between 1250 and 1700 hours with 20 percent confidence. We suppose that the robot is out of order only if all motors fail at the same time which corresponds to the fault tree in Fig. 5.

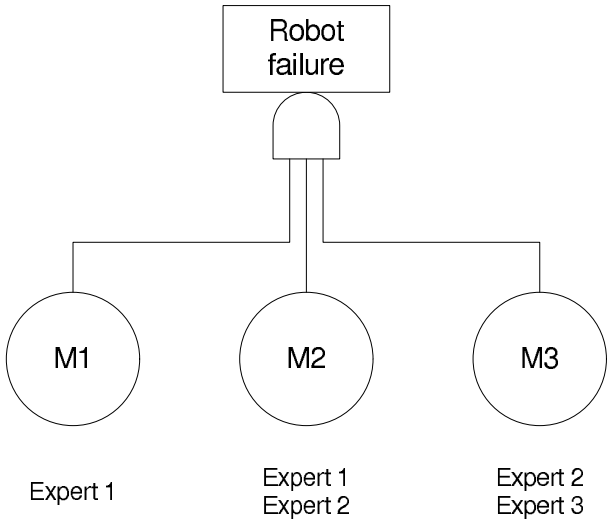


Figure 5. A fault tree for the robot

In an analogy to the definition from [12], we can compute the failure probability at the AND gate by using Dempster's rule.

We aggregate the evidence for the motors with the help of the unweighted mixing because every expert has the same confidence level. We prefer mixing to Dempster's rule in this case because the latter ignores focal elements without intersection which leads to information loss. After that, we obtain the belief and plausibility for the root element from the fault tree, that is the overall failure probability for the robot shown in Fig. 6.

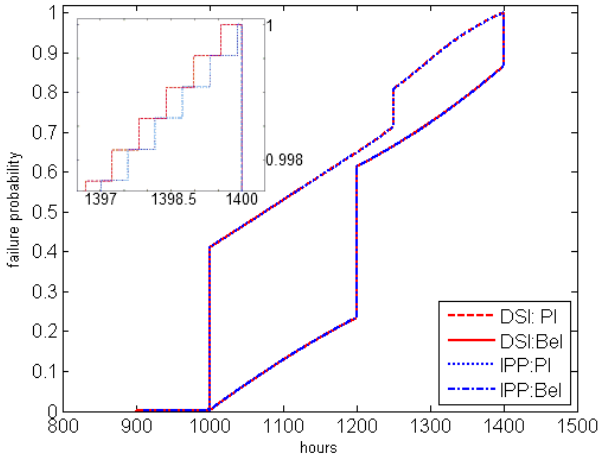


Figure 6. Solution space for robot failure with $2^{10}$ samples

The probability of a complete fault is zero for less than 900 hours of work. The robot fails with the certainty of 100 percent after 1400 hours. Note that the experts and their estimations are hypothetical in this example. The results are therefore not objective. The plausibility and belief values for failure probabilities between 900 and 1400 hours can be computed by the function `dsigetprob(x)`.

To provide a comparison between the IPP (the R version) and DSI toolboxes, we start a benchmark with 200, 1024 and 2048 samples. IPP supplies two sampling techniques called `dsodf` and `dsadf` [1], [13], the former of which is an outer discretization generating small and sharp intervals while the latter is more conservative and computes larger intervals. In Tab. I, we show results obtained with both methods.

Table I
COMPARISON OF CPU TIMES FOR THE FAULT TREE

| samples | 200 | 1024 | 2048 |
|---|---|---|---|
| IPP (dsadf) | 0.4641 s | 2.9110 s | 11.3253 s |
| IPP (dsodf) | 0.5226 s | 4.0561 s | 11.3214 s |
| DSI | 0.3368 s | 1.6899 s | 4.5552 s |

Each cell of the table contains the corresponding CPU time at the same platform as in section III. The DSI toolbox is faster for this example than the IPP.

### *B. Uncertainty Propagation for Non-monotonous Functions*

In this example, we demonstrate the potential of using interval methods and the DSI TOOLBOX for propagation of uncertainties through non-monotonous system models. We consider the function $\sin x^2$, which is simple, but highly non-monotonous. The argument $x$ follows a normal distribution

with the uncertain lower and upper bounds lying in intervals $[0, 0.1]$ and $[99.9, 100]$, respectively. We use the Monte Carlo sampling approach to propagate this uncertainty through a hypothetical system described by $\sin x^2$ in DSI and IPP (the R version). In the latter toolbox, the *monotonous* and *regular* methods described in section III-A produce wrong results. The only possible propagation algorithm for IPP is therefore the *optimization* based one, which we used for the comparison. The results are shown in Tab. II and in Fig. 7 (for 1024 samples with `dsodf` sampling method in IPP).

As demonstrated in the Table, the DSI TOOLBOX is considerably faster than the IPP. The reason for this is that the DSI exploits the ability of interval methods (and INTLAB in particular) to compute interval enclosures of functions over interval arguments directly rather via optimization problems. 'Directly' means that enclosures of functions consisting of combinations of $\{+, -, \cdot, /\}$ and standard functions such as trigonometric ones as well as their compositions are evaluated by substituting their interval counterparts for them on an operation-by-operation basis (the so called natural interval extension). That is, the optimization problem for the complex original function does not have to be solved. For such interval extensions, the relation $F_{Opt} \subseteq F_{Nat}$ always holds, where $F_{Opt}$ is the enclosure obtained via optimization, $F_{Nat}$ the natural one.

Table II
COMPARISON OF CPU TIMES FOR THE NON-MONOTONOUS PROPAGATION

| samples | 100 | 1000 | 1000 |
|---|---|---|---|
| IPP (dsadf) | 0.2850 s | 2.7190 s | 47.518 s |
| IPP (dsodf) | 0.2870 s | 2.6920 s | 48.407 s |
| DSI | 0.0585 s | 0.8237 s | 8.1557 s |

Note that $F_{Opt}$ obtained by a floating point based method is not *proved* to contain *all* possible solutions whereas $F_{Nat}$ is. However, $F_{Nat}$ might become too conservative if the function is evaluated for interval arguments with large widths (cf. Fig. 7). The results obtained with IPP lie inside the verified bounds provided by DSI. In this case, the reason is not only the conservativeness of enclosures because the arguments are tight intervals. The difference might also result from inaccuracies of the floating point based algorithm.

## V. CONCLUSIONS AND OUTLOOK

In this paper, we presented a new toolbox for MATLAB implementing a combination of the Dempster-Shafer theory with rigorous interval arithmetic. With its help, it was possible to work with DST structures in a natural way and take into account all rounding errors. We demonstrated the applicability of the box using several examples and shown that it worked faster than the floating point based IPP TOOLBOX from which it originated. Besides, we demonstrated that it was more advantageous to use DSI for uncertainty propagation through systems described by non-monotonous functions, both in terms of CPU time and correctness of results.

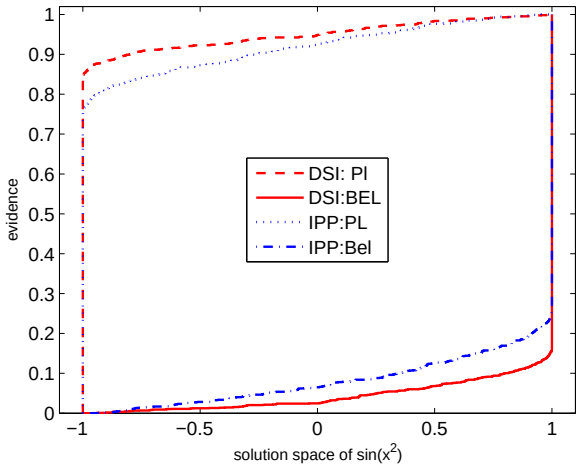


Figure 7. Propagation of non-monotonous functions

Our future work will consist of combining the verified DST with the fault tree analysis and Markov chains.

## REFERENCES


[1] P. Limbourg, "Imprecise probabilities for predicting dependability of mechatronic systems in early design stages," Ph.D. dissertation, University of Dusburg-Essen, 2007.

[2] A. Martin, "Implementing general belief function framework with a practical codification for low complexity," in *Advances and Applications of DSmT for*. American Research Press, 2009, vol. Collected Works, Vol. 3, pp. 217–273.

[3] G. Nassreddine, F. Abdallah, and T. Denoeux, "A state estimation method for multiple model systems using belief function theory," in *Information Fusion, 2009. 12th International Conference on Information Fusion (FUSION '09)*, 2009, pp. 506–513.

[4] E. Moore, B. Kearfott, and M. Cloud, *Introduction to Interval Analysis*. Society for Industrial Mathematics, 2009, vol. 1.

[5] "The mathworks deutschland - matlab - the language of technical computing," 2009. [Online]. Available: http://www.mathworks.de/products/matlab/

[6] "The r project for statistical computing." [Online]. Available: http://www.r-project.org/

[7] S. Rump, "Intlab - interval laboratory," *Developments in Reliable Computing*, pp. 77–104, 1999.

[8] S. Ferson, V. Kreinovich, L. Ginzburg, D. Myers, and K. Sentz, "Constructing probability boxes and dempster-shafer structures," no. SAND2002-4015, 2003.

[9] P. Limbourg, R. Savic, J. Petersen, and H. D. Kochs, "Modelling uncertainty in fault tree analyses using evidence theory," *Journal of Risk and Reliability*, vol. 222, pp. 291–302, 2008.

[10] E. de Rocquigny, N. Devictor, and S. Tarantola, *Uncertainty in Industrial Practice - A guide to quantitative uncertainty management*. United Kingdom: John Wiley & Sons, 2008, vol. 1.

[11] S. Rump, "Fast and parallel interval arithmetic," *BIT*, vol. 39(3), pp. 539–560, 1999. [Online]. Available: http://www.ti3.tu-harburg.de/paper/rump/Ru99b.pdf

[12] H. Traczinski, "Integration von algorithmen und datentypen zur validierten mehrkörpersimulation in mobile," Ph.D. dissertation, Universität Duisburg-Essen, 2006.

[13] F. Tonon, "Using random set theory to propagate epistemic uncertainty through a mechanical system," *Reliability Engineering and System Safety*, vol. 85, no. 1-3, pp. 169–181, 2004.