What's $Z-X$, when $Z = X+Y$?
Dependency tracking in interval arithmetic with bivariate sets

Ander Gray[1,*], Marco De Angelis[2,*], Scott Ferson[3,*] and Edoardo Patelli[4,*,†]
[*] *Institute for Risk and Uncertainty, University of Liverpool, UK*
[†] *University of Strathcylde, UK*

[1] *Ander.Gray@liverpool.ac.uk,* [2] *mda@liverpool.ac.uk*
[3] *Scott.Ferson@liverpool.ac.uk,* [4] *Edoardo.Patelli@strath.ac.uk*

Abstract. In this paper we propose an extension to interval arithmetic to include arithmetic with bivariate sets, which allows for an initial dependency to be propagated, as well as the tracking of complicated dependencies arising from repeated variables. These bivariate sets are represented by Boolean fields which define where two intervals jointly exist or not. We name this extended interval arithmetic *zone arithmetic*. We show how conditional sets may be constructed from bivariate intervals, and how dependent interval arithmetic may be performed with this conditioning. With conditioning we calculate the dependencies between the inputs and outputs of operations, allowing for the extra uncertainty from repeated variables to be greatly reduced. Recent work on using copulas to perform a similar calculations with p-boxes is also reviewed.

Keywords: Uncertainty Propagation, Interval Arithmetic, Imprecise Probabilities, Copulas, Probabilistic Arithmetic, Probability Bounds Analysis, Repeated Variables, Dependency Tracking

1. Introduction

Uncertain computer arithmetics have gained a lot of attention and research over the years (Williamson, 1989; Ferson and Hajagos, 2004; Rump, 1999; Tucker, 2011). They are an intrusive alternative to sampling based methods for uncertainty propagation and quantification. They relay on access to the source code of the computational model. Much like how automatic differentiation calculates the derivative after every line of code, the uncertainty arithmetics calculate the uncertainty. These intrusive methods work by replacing floating point operations with operations defined for uncertain numbers, such as intervals, distributions, fuzzy numbers or p-boxes. This allows for uncertainty to be propagated with the following features:

– Only a single model evaluation needed; albeit more expensive than an evaluations based on floating point.

– A bounded solution to the output uncertainty is achieved. That is, an outer approximation which contracts to the correct result with more computational effort. Interval calculations also allow for automatically verified computation.

– Functions of sets of probability distributions can be computed as cheaply as functions of singular distributions.

One of the main problems for this type of uncertainty propagation is known as the *dependency problem*, or sometimes called *repeated variables* or the *wrapping effect* in interval arithmetic. That when a variable appears multiple times in a sequence of operations, the uncertainty is over-estimated. By the nature of the uncertainty arithmetics, the correct solution is still bounded, however the bounds can be prohibitively large. For example consider the following sequence of binary operations:

$$\begin{array}{l|l} Z = X + Y & [-1,1] + [-1,1] = [-2,2] \\ \tilde{Y} = Z - X & [-2,2] - [-1,1] = [-3,3] \end{array} \tag{1}$$

The left shows a sequence of operations, and the right shows it evaluated with interval arithmetic. X and Y are the intervals $[-1,1]$, and are summed to create Z. X is then subtracted from Z to create $\tilde{Y}$. Clearly since the *same* X was added and then subtracted: $Y = \tilde{Y}$. However the calculation gave us $Y \subset \tilde{Y}$ (i.e. $[-1,1] \subset [-3,3]$). This extra uncertainty came from the fact that X was repeated in the sequence of operations. Operations between uncertain quantities are a function of their dependence. Since Z was created in a binary operation involving X, they are somehow dependent on one another and it was the disregard of this dependence that lead to the over-estimation. The calculation sequence itself introduces a dependence.

There are been several proposed solutions for repeated variables in interval arithmetic, including significance arithmetic (Hyman, 1982), affine arithmetic (Comba and Stolfi, 1993; Rump and Kashiwagi, 2015), zonotopes (Bogomolov et al, 2019) and Taylor models (Makino, 1998). There has also been a method proposed for probability distributions (Li and Hyman, 1998). In this paper we propose an alternative where interval dependencies are propagated and tracked through operations, allowing for an initial interval dependency to be propagated as well as capturing the dependence arising from a sequence of operations with repeated variables. A similar technique for p-boxes but using copulas recently proposed by the authors will also be reviewed.

2. Interval Dependencies

There have been a number of proposed models for dependencies amongst intervals, two which are notable from previous proceedings of the *International Workshop on Reliable Engineering*. Ferson and Kreinovich (2006) describe a dependence between two intervals as regions where the two sets are pair-wise allowed and disallowed (values where the two sets jointly exist). They describe a number of parametric families for interval dependencies, and show how arithmetic operations can be performed with these families. They include *non-interactive* (the standard interval arithmetic case), *perfect* and *opposite* dependencies as special cases. Ceberio et al (2006) similarly describe a joint interval as a set of possible allowed pairs, but allow the set to take any shape. They define a dependence between two intervals x_1 and x_2 to be any proper subset of their Cartesian product $J_{x_1,x_2} \subseteq x_1 \times x_2$. They describe how any arbitrary subset of $x_1 \times x_2$ can be represented on a computer

by a discrete *outer* approximation. They first divide the box $x_1 \times x_2$ into $n \times n$ sub-boxes; and then describe the set J_{x_1,x_2} as the union of sub-boxes which contain a possible pair. This representation of a set is the well known upper approximation used in *rough set* theory (Polkowski, 2002), where dependent interval operations are performed with interval arithmetic on each individual sub-box, much in the style of *sub-intervalisation*. In this section we expand on this, describing in more detail how multivariate intervals may be defined and manipulated. We also describe a method for performing an arithmetic with dependent intervals, which is computationally cheaper than sub-intervalisation and may be used to determine dependencies between inputs and outputs of binary and unary operations, a task which is required for dependency tracking.

2.1. Exclusion Zones

In probability theory, a copula returns a cdf value for a (u, v) pair in the unit square $[0, 1]^2$, $C : [0, 1]^2 \to [0, 1]$, and can be used to define any probabilistic dependency. Following a similar idea, we define any generic interval dependency as an indicator function $\mathbb{Z}(u, v)$ on $[0, 1]^2$ which returns a Boolean value, $\mathbb{Z} : [0, 1]^2 \to \{0, 1\}$, stating whether the pair (u, v) is contained in the bivariate set. Much like how univariate intervals make no statement of how the probability is distributed within them, these bivariate sets only state whether possible pairs are contained or not. Generally the set $\mathbb{Z}$ may be any arbitrarily complicated shape, however we construct an outer approximation of Z by a uniform grid of sub-boxes on $[0, 1]^2$, which may be stored on the computer by a $n \times n$ Boolean field. $\mathbb{Z}$ may also be extended to allow for interval inputs, in which case $\mathbb{Z}$ returns a 1 if *any* real values in the interval are included in $\mathbb{Z}$:

$$Z(u, v) = \begin{cases} 1 & \text{if } (u \times v) \cap \mathbb{Z} \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Stated another way, Z returns a 0 only if all the values in the interval are not in the original set, and in that sense Z guarantees exclusion. For this reason we label functions like Z as exclusion zones, or *zones* for short. Figure 1 shows a hypothetical interval dependency represented by a zone, with some examples of intervals which would be included and excluded. This set is the union of two rings, and is just an example of a general bivariate set picked to show the flexability of this method. Note that the zone does not need to extend to the entire $[0, 1]^2$ range, which is a modelling choice. Information about the bounds of one variable may inform the bounds of the other, and may not extend to its entire range. Also for practical reasons, any values outside $[0, 1]^2$ will be returned as 0.

Much like how a bivariate distribution may be defined in terms of two marginals and a copula, a joint interval may be defined by two marginal (or univariate) intervals x_1, x_2 and a zone $Z(u, v)$, by the following relation: $J_{x_1,x_2}(x, y) = \operatorname{supp} Z(u, v)$ with $u = (x - \underline{x_1})/(\overline{x_1} - \underline{x_1})$ and $v = (y - \underline{x_2})/(\overline{x_2} - \underline{x_2})$. This defines a grid of sub-boxes following Z by a uniform scaling by the intervals x_1 and x_2, and then taking the support of resulting indicator function. This allows dependencies to be stored and manipulated independently from marginal intervals.

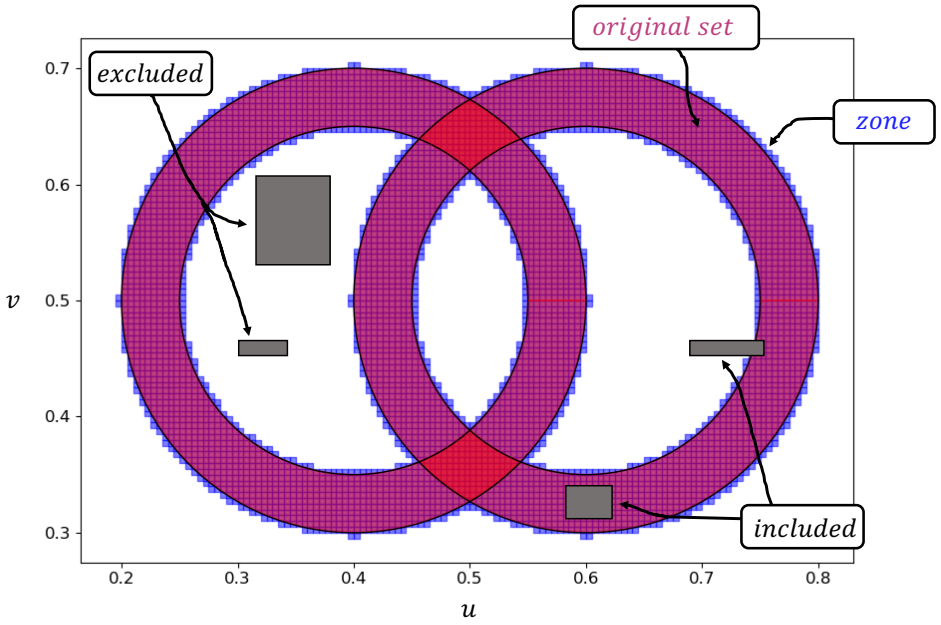

Figure 1. Shows an outer approximation of a set represented by a zone. Also shown are several intervals which would be included and excluded by the zone.

3. Zone Arithmetic

Ceberio et al (2006) suggest that binary operations may be performed with J by performing interval arithmetic on each individual sub-box, which requires of the order of $\sim n^2$ interval operations. We proposed an alternative method which only requires $\sim n$ operations. For this we use another familiar concept for probability theory: *conditioning*. Much like how a univariate distribution can be created from a bivariate distribution by conditioning on a value, it's possible to produce a conditional set from a bivariate set. A conditional set $J(x|\ y = Y)$ may be created from the bivariate interval $J(x, y)$ by intersecting the it with an interval, or real value, Y:

$$J(x|\ y = Y) = J(x, y) \cap ([-\infty, \infty] \times Y) \tag{3}$$

$J(x|\ Y)$ is a univariate set's indicator function, and may be the union of several disjoint intervals. For example, if we condition the zone in figure 1 with $v = [0.48, 52]$, we would get a set made up of 4 disjoint intervals: $[0.195, 0.255] \cup [0.395, 0.455] \cup [0.545, 0.605] \cup [0.745, 0.8]$. Figure 2 shows an illustration of this. Calculations of this sort can be performed quite efficiently in terms of zones, since it's merely a subselection of an array of Z.

We perform dependent interval operations using this conditioning. Instead of n^2 interval operations, we perform n (for each row or column) set operations on the conditionals. Since a conditional may contain more than one interval, multiple interval operations are performed per set operation, however the overall number of interval operations remains lower than n^2. The minimum number of

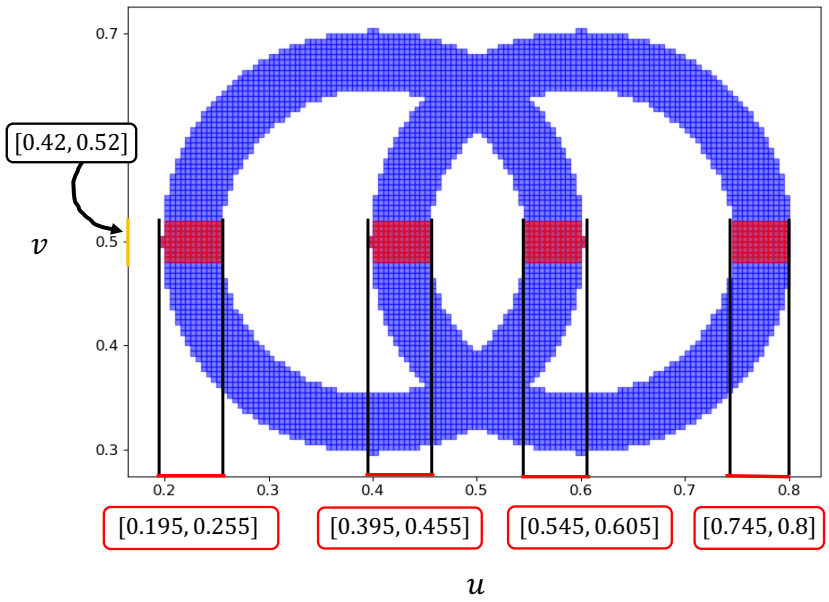

Figure 2. illustration of conditioning a zone with the interval $v = [0.48, 52]$. The resulting set is the union of the disjoint intervals $[0.195, 0.255] \cup [0.395, 0.455] \cup [0.545, 0.605] \cup [0.745, 0.8]$.

interval operations is n, corresponding to the non-interactive case (everywhere true). The maximum number of required interval operations is $\frac{n^2}{2}$, where is a very extreme case where every other element of the zone is empty.

Using conditioning, the dependencies between inputs and outputs of operations may be calculated, which is required for dependency tracking. Take for example the sequence of operations in the title of this paper: $x + y = z$, and then $z - x$, where initially the joint interval J_{XY} is known. For $z - x$ to be evaluated tightly, the joint J_{ZX}, or similarly the zone Z_{ZX}, must be calculated. For this we condition J_{XY} on some subinterval $x = X$, and evaluate the $x + y$ with the conditional set, which gives $J_{ZX}(z|x = X)$. This is then repeated for n subintervals of x, and the joint J_{ZX} can be calculated as the union of the resulting conditionals of z:

$$J_{ZX}(z|x = X) = J_{YX}(y|x = X) + X \tag{4}$$

$$J_{ZX}(z, x) = \bigcup_{i=1}^{n} J_{ZX}(z|x = X_i) \tag{5}$$

where the $+$ operator is an interval sum, and where X_i is a subinterval. Figure 3 shows an example of this, where $x + y = z$ is evaluated for $x, y = [-1, 1]$ and with J_{XY} having the complement of a ring shaped dependence shown on the left. The calculated interval $z = [-2, 2]$ as expected, however the dependence has also been propagated. If $\tilde{y} = z - x$ is evaluated using zone arithmetic, then $\tilde{y} = [-1.08, 1.08]$, which tightly encloses the correct solution of $\tilde{y} = y = [-1, 1]$, and a vast improvement of the standard interval arithmetic answer of $\tilde{y} = [-3, 3]$. The tightness of the result

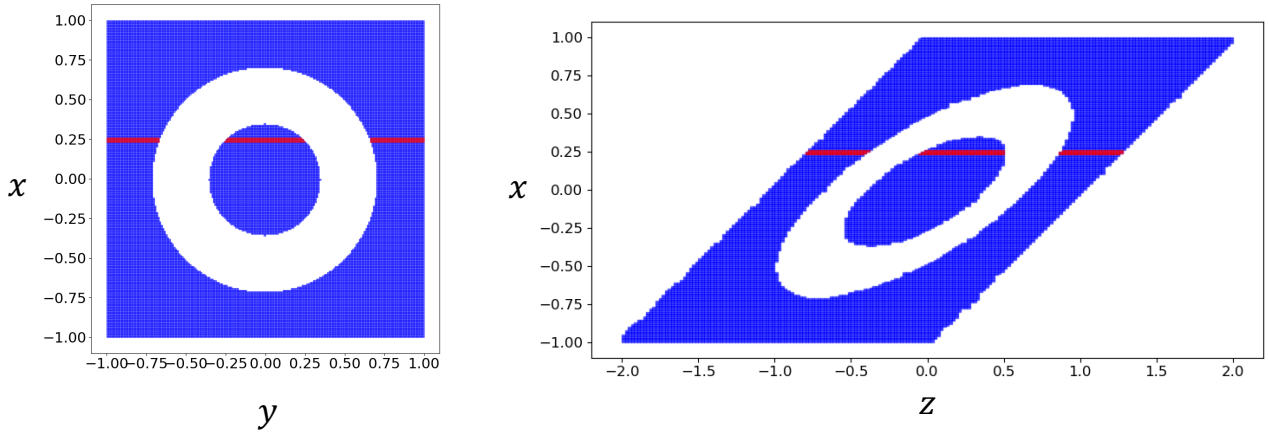

Figure 3. An example of using conditional arithmetic to propagate an interval dependency though a binary operator. Shown is $x + y = z$, with $x, y = [-1, 1]$ having the dependency shown on the left figure. Right shows the calculated bivariate set J_{ZX}. Shown in red is a conditional set, conditioned on a particular subinterval of x, which is used to construct J_{ZX}.

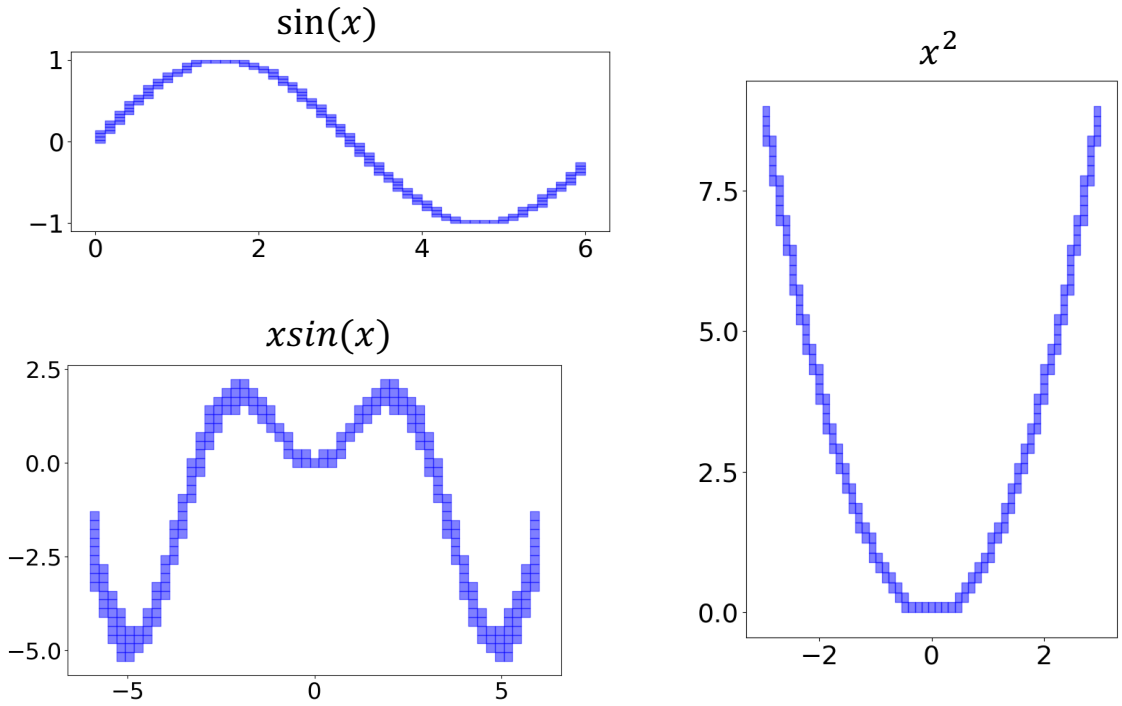

Figure 4. Examples of dependency tracking through unary functions, for discretisations 50×50. Shown is $\sin([0, 6])$ (top left) and $[-3, 3]^2$ (right). Also shown is $x \sin(x)$ for $x = [-6, 6]$ on the bottom left.

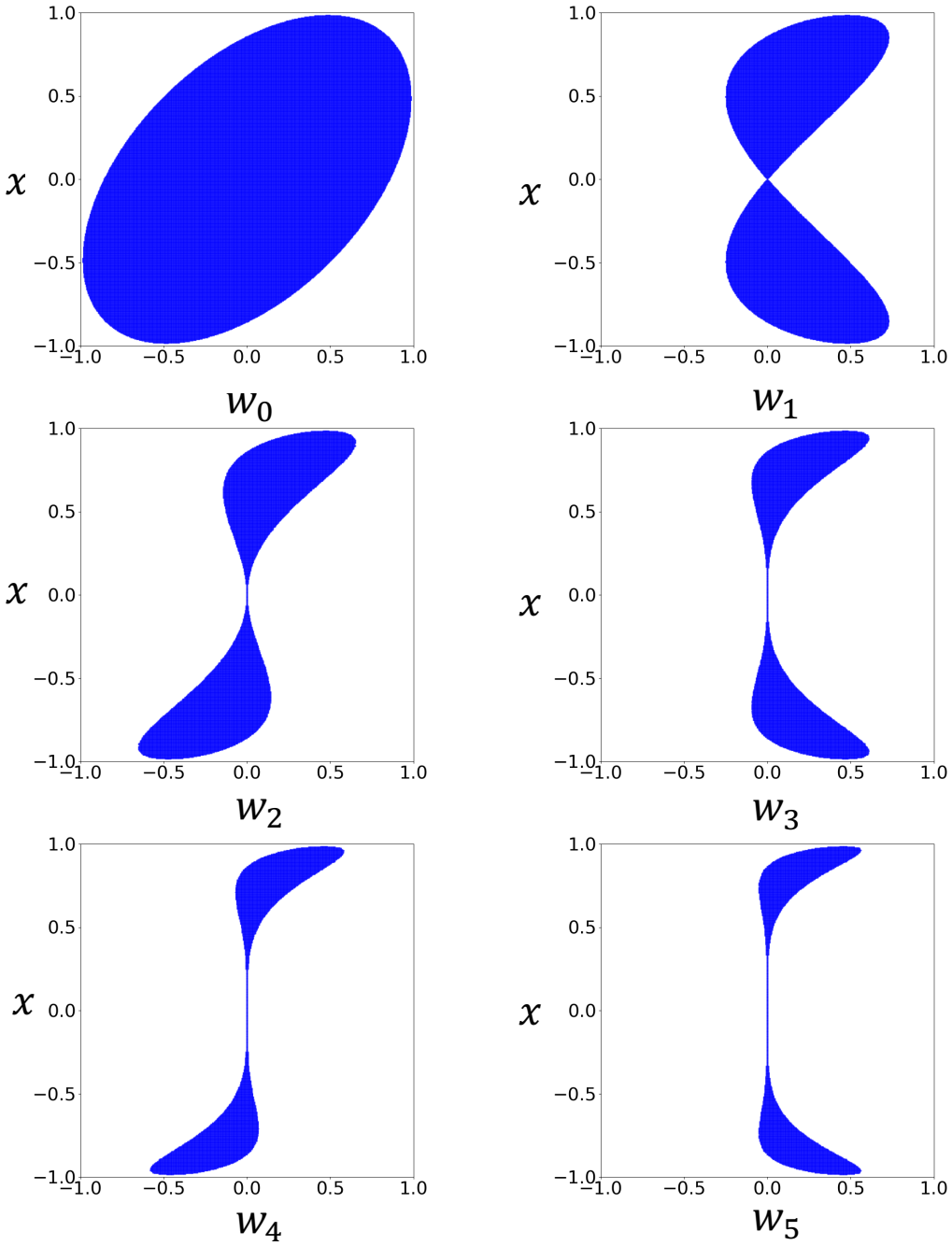

Figure 5. Shown is the evolution of an ellipse through the repeated evaluation of $w_{i+1} = w_i * x$

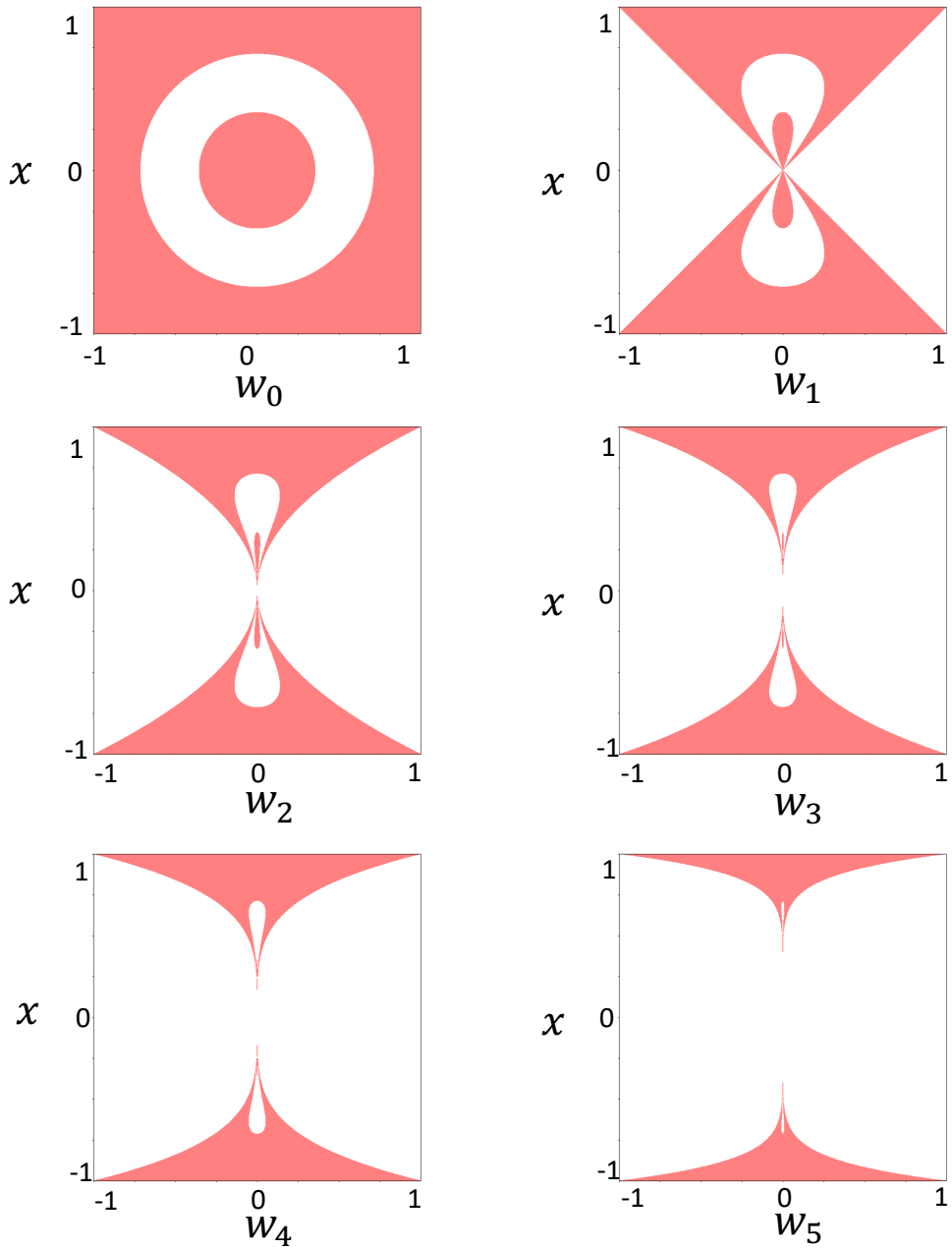

Figure 6. Shown is the evolution of an interval dependence through the repeated evaluation of $w_{i+1} = w_i * x$, starting from the complement of a ring shaped dependence in the top left.

depends on the discretisation used, and for this example a discretisation 200×200 was used. Using fewer sub-boxes will compute the result quicker and lead to a wider result, however it is still guaranteed to bound the correct dependency and marginal. A similar calculation can be performed for unary functions such as $\sin(x)$ or x^2, which is shown in figure 4. It can be seen that the shape of the unary function is enclosed. This method can also be used to propagate dependencies further, such as through $x \sin(x)$ as shown on the bottom left in figure 4. Figure 5 shows the tracking of an elliptical set through repeated evaluation of $w_{i+1} = w_i * x$ for an initial. It can be seen that the dependency does not remain an ellipse. A similar calculation is shown on figure 6 but with an initial ring shaped dependence. It can be seen that this method can tightly track the quite complex dependencies that arise from repeated variables.

4. Copulas and p-boxes

A similar method for dependency tracking with p-boxes has recently been proposed by the authors, and is available through the Julia package *ProbabilityBoundsAnalysis.jl* (Gray, 2020). Here we will review the main elements of the method.

A binary operation of two random variables X and Y with a known joint distribution can be computed by the following Lebesgue-Stieltjes integral or *convolution* (Williamson and Downs, 1990):

$$\sigma_{C,L}(F, G)(x) = \int_{L\{x\}} dC(F(u), G(v)) \tag{6}$$

which is called the σ convolution, written for any two arbitrary inputs cdfs F and G, with a copula C and a binary operation L. The integration domain is $L\{x\} = \{(u, v)|u, v \in \Re, L(u, v) < x\}$, and where $L(u, v)$ is a binary operation, for example $L(u, v) = u+v$. The σ convolution computes the cdf of a binary operation of two distribution functions with a known joint distribution, and is effectively the integral which is approximated when Monte Carlo simulation is performed. A σ convolution may also be perform with p-box inputs (Williamson, 1989). Computing a σ convolution requires a copula C to be specified, which may be available at the beginning of a sequence of operations, but will be lost as the calculation sequence is executed. This leads to the only option being a *Fréchet* convolution, which calculates a binary operation between p-boxes for all possible C. This however, as in the interval case, leads to an overestimation of the uncertainty.

This may be overcome if C is also propagated through binary operations. For some binary operation between random variables $Z = X \,\square\, Y$, where $\square$ is some binary operator (eg $\square = +$), the joint distribution of Z and X is given by:

$$F_{ZX}(z, w) = \iint_{D_{zw}} dF_{XY}(x, y) \tag{7}$$

where the area of integration D_{zw} is the region in the (x, y) plane where $x \,\square\, y < z$ and $x < w$ (Williamson, 1989). Using Sklars theorem:

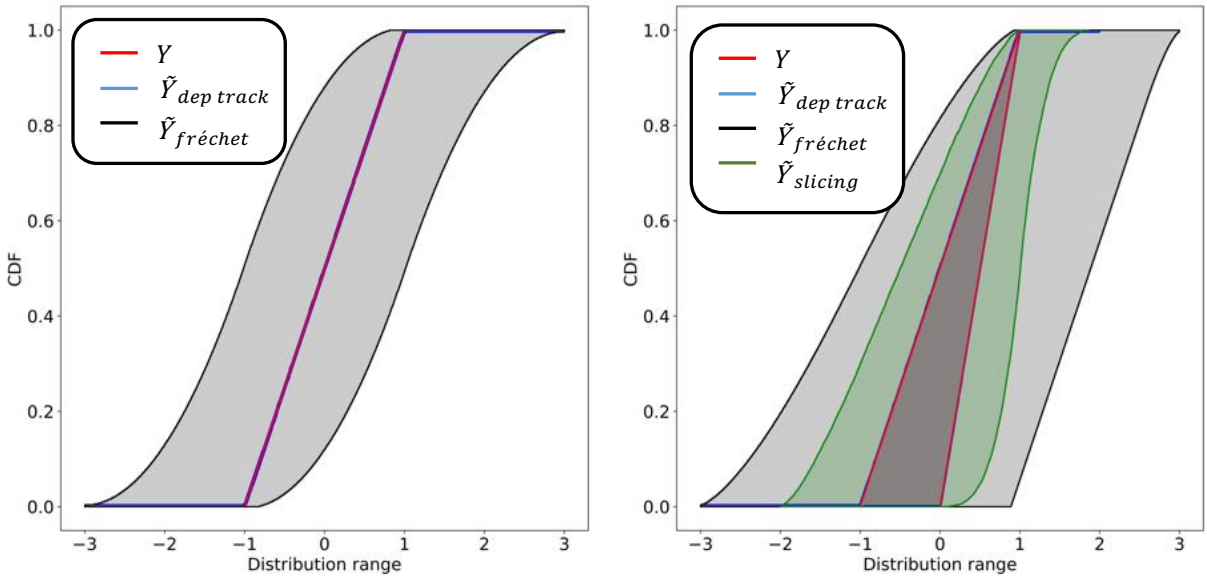

Figure 7. Left shows the calculation sequence of equation 1, with $X, Y \sim U(0,1)$ and independent. The right shows the result the sequence of binary operations but with X and Y starting as the p-boxes $U([-1,0],1)$ and with a gaussian copula with correlation $= 0.5$. The Fréchet result is plotted in both case to show the contraction in uncertainty. The green p-box is the result of random slicing and interval arithmetic.

$$C_{ZX}(u,v) = \iint\limits_{D_{F_Z^\wedge(u),F_X^\wedge(v)}} dC_{XY}(F_X(x),F_Y(y)) \tag{8}$$

where the area of integration is now where $x \,\square\, y < F_Z^\wedge(u)$ and $x < F_X^\wedge(v)$, where $F_Z^\wedge(u)$ is the inverse cdf of Z, which can be found by first computing a σ convolution. Equation 8 may be quite computationally difficult to compute due to the complex integration region. However since the copula is a cdf, it is possible to perform this calulation iteratively, reusing previously calculated values. For some step δ:

$$C_{ZX}(u+\delta,v+\delta) = C_{ZX}(u,v) - C_{ZX}(u+\delta,v) - C_{ZX}(u,v+\delta) + \iint\limits_{A} dC_{XY}(F_X(x),F_Y(y)) \tag{9}$$

with the integration region A being where $F_Z^\wedge(u) \le x \,\square\, y < F_Z^\wedge(u+\delta)$ and $F_X^\wedge(v) \le x < F_X^\wedge(v+\delta)$. Equation 9 uses the 2-d version of the *H-volume* formula (Schweizer and Sklar, 2011), which can be used to find the probability mass in some hyper-rectangle from a n-d cdf. An identical calculation can be performed for Y.

The above methodology was applied to the problem posed in the title of the paper, the sequence of binary operations 1. That is the copula C_{ZX} was calulated for $Z = X + Y$ with X and Y being independently and uniformly distributed $U(-1,1)$. Figure 7 shows the results of using this method.

F_z was first calculated with $\sigma_{\pi,+}(F_X, F_Y)$, the copula C_{ZX} was then calculated with equation 8, defining the calculation dependency between Z and X. $F_{\tilde{Y}}$ was then calulated with $\sigma_{C_{ZX},-}(F_Z, F_X)$. It can be seen that $\tilde{Y}$ and Y are the same, plus some outward directed numerical error from the calculation. The Fréchet calculation is also plotted to show the reduction in uncertainty from dependency tracking. This example was with two independent uniform distributions, however any copula may be initially used. The initial variables may even be p-boxes. The right of figure 7 shows the same calculation but with X and Y being the p-boxes $U([-1, 0], 1)$ and with an initial Gaussian copula with correlation of 0.5. Again it can be seen that the calculation with dependency tracking tightly encloses the correct result. The green p-box is the result of random slicing, where random intervals are generated from X and Y using the gaussian copula. The sequence of binary operations is then evaluated with interval arithmetic and resulting p-box is constructed from the random set. It can be see that random slicing is an improvement over Fréchet (although more expensive to evaluate), but also suffers from repeated variables, since there is no dependency tracking in the interval calculations. Note that when this calculation is performed with p-boxes, equation 8 must be evaluated twice, for the upper and lower copula bound.

5. Conclusions

In this paper an extension to interval arithmetic to include dependence was proposed, where interval dependencies were defined in terms of bivariate sets. These bivariate sets define where two intervals co-exist or not, and may be arbitrarily complex with continuous boundaries. We represent such sets on the computer with finite Boolean fields, like an $n \times n$ Boolean matrix, which defines a grid of sub-intervals. For continuous sets, we create an discrete outer approximation which is guaranteed to enclose all elements of the continuous set. Much in the style of copulas, we describe how interval dependencies may be modelled independently from univariate intervals. We describe how conditional sets are constructed by conditioning bivariate sets on real numbers or intervals, and describe a dependent interval arithmetic in terms of conditioning. Dependent interval arithmetic by conditioning requires of the order of $\sim n$ operations, which is an improvement over the standard *sub-intervalisation* which is around $\sim n^2$ interval operations. We show that when interval dependencies are propagated and calculated this way, the repeated variable problem present in standard interval arithmetic is greatly reduced, and that this method can calculate the complex dependencies that arise in computational models with repeated variables. A similar method for p-boxes using copulas was described, and we show that copulas are capable at modelling the probabilistic dependencies that arise from repeated variables in computations involving distributions and p-boxes. For this task, we calculate the copulas between the inputs and outputs of binary operators involving p-boxes, capturing the dependencies arising from the operation. Uncertainty is greatly reduced when dependency arising from the computation model is tracked when compared to Fréchet convolutions or propagation with random slicing.

Acknowledgements

The authors would like to thank the gracious support from the EPSRC iCase studentship award 15220067. We also gratefully acknowledge funding from UKRI via the EPSRC and ESRC Centre for Doctoral Training in Risk and Uncertainty Quantification and Management in Complex Systems. This research is funded by the Engineering Physical Sciences Research Council (EPSRC) with grant no. EP/R006768/1, which is greatly acknowledged for its funding and support. This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

Bezanson, J., Edelman, A., Karpinski, S., Shah, V. B., Julia: A fresh approach to numerical computing, *SIAM review* 59 (1) (2017) 65–98. https://doi.org/10.1137/141000671

Bogomolov, S., Forets, M., Frehse, G., Potomkin, K., Schilling, C., Juliareach: a toolbox for set-based reachability, in: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, 2019, pp. 39–44.

Ceberio, M., Ferson, S., Kreinovich, V., Chopra, S., Xiang, G., Murguia, A., Santillan, J., How to take into account dependence between the inputs: From interval computations to constraint-related set computations, with potential applications to nuclear safety, bio- and geosciences, RELIABLE ENGINEERING COMPUTING (2006) 126.

Comba, J. L. D., and Stolfi, J., A new arithmetic and its applications to computer graphics, In Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing) (pp. 9-18)

Ferson, S., Kreinovich, V., Modeling correlation and dependence among intervals, RELIABLE ENGINEERING COMPUTING (2006) 115.

Ferson, S. and Hajagos, J. G., Arithmetic with uncertain numbers: rigorous and (often) best possible answers. *Reliability Engineering & System Safety*, 85(1-3):135–152, 2004.

Gray, A., https://github.com/AnderGray/ProbabilityBoundsAnalysis.jl: Imprecise copulas (2020) doi: 10.5281/ZENODO.4281562

Gray, N. and De Angelis, M. and Ferson, S., *COMPUTING WITH UNCERTAINTY: INTRODUCING PUFFIN THE AUTOMATIC UNCERTAINTY COMPILER*. UNCECOMP, 2019.

Hyman, J. M., Forsig: an extension of fortran with significance arithmetic, Tech. rep., Los Alamos National Lab., NM (USA) (1982).

Li, W., Hyman, J. M., Computer arithmetic for probability distribution variables, Reliability Engineering & System Safety 85 (1-3) (2004) 191–209.

Makino, K., Rigorous analysis of nonlinear motion in particle accelerators. PhD thesis, Michigan State University, 1998.

Polkowski, L., Rough sets, Mathematical foundations. Physica Verlag, A Springer-Verlag Co., Heidelberg, Springer, 2002.

Rump, S. M., Intlab—interval laboratory, in: Developments in reliable computing, Springer, 1999, pp. 77–104.

Rump, S. M., Kashiwagi, M., Implementation and improvements of affine arithmetic, Nonlinear Theory and Its Applications, IEICE 6 (3) (2015) 341–359.

Schweizer, B. and Sklar, A. *Probabilistic metric spaces*. Courier Corporation, 2011.

Tucker, W., Validated numerics: a short introduction to rigorous computations, Princeton University Press, 2011.

Williamson, R. C., *Probabilistic arithmetic*. PhD thesis, University of Queensland Australia, 1989.

Williamson, R. C. and Downs, T. Probabilistic arithmetic. i. numerical methods for calculating convolutions and dependency bounds. *International journal of approximate reasoning*, 4(2):89–158, 1990.