KaucherPy: interval package for Kaucher arithmetic

Dirceu Maraschin Jr[1], Lucas M. Tortelli[1], Alice F. Finger[1] and Aline B. Loret[2]

[1] *CDTec - Universidade Federal de Pelotas (UFPEL)*

[2] *Universidade Federal de Santa Maria (UFSM)*

Abstract: The paper presents the KaucherPy interval package developed in the Python language following the concepts of Kaucher's interval arithmetic extension. Kaucher arithmetic introduces the concept of interval generalization allowing the use of improper or direct intervals. Few interval packages allow direct use of Kaucher arithmetic. Therefore, KaucherPy introduces Kaucher concepts with an open source package for scientific programming. All operations defined in Kaucher interval arithmetic and qualitative metrics for quality evaluation of interval result are available. KaucherPy was developed under IEEE guidelines allowing easy installation and guaranteeing results with high precision by the principles of interval arithmetic. The paper presents practical examples computed to demonstrate the operability and quality of the results offered by KaucherPy, also demonstrated the limitations of Kaucher arithmetic.

Keywords: Interval Arithmetic, Interval package, Kaucher, KaucherPy, Python, Scientific Computing.

1 Introduction

Floating-point arithmetic has limitations to discretize continuous values, since its representation is given by an approximation of a conventional values subset. This approximation occurs because rounding and truncation operations, which add errors that contribute to the numerical values uncertainty, so that its accuracy can not be guaranteed. Interval arithmetic consists of a powerful numerical computation problem solving tool, providing a rigorous and automatic error control.

The commonly known definition for interval arithmetic is Moore's Standard Interval Arithmetic [13]. SIA is widely used in the scientific area, but contains insufficiencies. Therefore, it does not supply all the modeling mathematical needs. With a linear representation of space □ℝ, Moore's interval definition does not guarantee important properties such as duality and isotonicity, because ranges with negative width are not accepted.

Thus, extensions were developed in order to suppress these limitations and expand the set of solvable problems by applying intervals instead of point values giving rise to new interval arithmetic, such as:

- □ℝ space interval extension from proper to improper intervals developed by Kaucher [10] showing a new definition for operations involving intervals, represented by □;
- Affine IA [7] expands the concepts of interval arithmetic, seek to adjust the SIA variables dependence problems;
- Constrained IA (CIA) defined by Lodwick [11] satisfies the inverse additive, inverse multiplicative and distributivity law properties representing an interval value through a three parameter function:

  $X^I(\underline{x},\bar{x},\alpha_x)=\{x \mid x=(1-\alpha_x)\underline{x}+\alpha_x\bar{x}, 0\le\alpha_x\le 1\}$;
- The multidimensional arithmetic RDM (RDM-IA) development by Piegat and Landowski [15] [16] is an arithmetic that determines the universal algebraic solution of a problem in form of a multidimensional set. RDM-IA represents an interval with one uncertainty variable α as parameter:

  $[x]=\{x : x=\underline{x}+\alpha_x(\bar{x}-\underline{x}), \alpha_x\in[0,1]\}$.

Interval development environments are usually associated with a programming language, depending on its use for application development. Some packages for interval development are: Moore [5], JAVA-XSC [1], C-XSC [4], IntPy [6] and INTLAB [3]. Although there is the development of others interval arithmetic packages for computational systems, but cannot be directly used. Incompatibilities or problems with the language in which they were developed are some of the reasons, for example, the development of Kaucher's interval arithmetic extension by Popova [17] in the Pascal language.

The present developed package in this paper contains the interval extension concepts developed by Kaucher [10], named KaucherPy. This package aims to provide quality of results in numerical problems allowing, in addition to the automated generated intrinsic error control, to extend the solvable problems domain using interval arithmetic. Its development in the Python language is given because it presents a large community, documentation and libraries available.

Python contains a package that make modifications to conventional arithmetic operations, adding higher accuracy to the results. Thus, KaucherPy unifies with these numerical definitions to introduce them at intervals. All the carried out development complies with the IEEE [2] directives, providing all production, exception handling and portability systems developed in conventional arithmetic for the KaucherPy solution.

2 Interval Extension of Kaucher

In Kaucher's arithmetic, a continuous real values function is extended, following the definition of generalized interval: $[x] = \{x \in \mathrm{R} \mid \underline{x} \le x \le \overline{x},$ if $\underline{x} \le \overline{x}; \overline{x} \le x \le \underline{x},$ if $\underline{x} > \overline{x}\}$, being an actual values subset complemented by a direction [17].

In Kaucher, the set of all finite intervals $\mathrm{IR} = \{[\underline{x}, \overline{x}] \mid \underline{x}, \overline{x} \in \mathrm{R}, \underline{x} \le \overline{x}\}$ is extended for a set $H = \{[\underline{x}, \overline{x}] \mid \underline{x}, \overline{x} \in \mathrm{R}\} \cong \mathrm{R}^2$ of ordered pairs of finite real numbers. The generalized interval of Kaucher is also known by different names, including modal interval, directed interval, Kaucher's interval and generalized interval.

A continuous real function $f : \mathrm{R}^n \to \mathrm{R}$, its generalization to intervals can be expressed by $f^{\mathrm{KR}} : \mathrm{KR}^n \to \mathrm{KR}$. Considering the interval $[t] \in \mathrm{IR}$ such that $f([t]) = \{f(t) \mid t \in [t]\}$, the f monotonicity type determines its "direction" wherein the range $f([t])$ is traced.

A direct interval $[x] = [\underline{x}, \overline{x}] \in H$ is proper if $\underline{x} \le \overline{x}$, improper if $\underline{x} > \overline{x}$, or generated when $\underline{x} = \overline{x}$. The signal σ of $[x]$ defined by (1).

$$\sigma(A) = \begin{cases} +, & if \quad 0 \le \underline{x} \quad and \quad 0 \le \overline{x} \\ -, & if \quad \underline{x} < 0 \quad and \quad \overline{x} < 0, \end{cases} \tag{1}$$

and your direction can be defined by:

$$\tau(A) = \begin{cases} +, & if \quad \underline{x} \le \overline{x}, \\ -, & otherwise. \end{cases} \tag{2}$$

Considering a function $f(x) = f_1(x) \circ f_2(x)$, such that $\circ \in \{+, -, \times, /\}$, the function f image with know intervals $f_1([x])$, $f_2([x])$. When f_1 and f_2 are continuous in ℝ, the values of $f_1([x])$ and $f_2([x])$ are interval-value, so the image function is $f([x]) = \{f_1(x) \circ f_2(x) \mid x \in [x]\} \subseteq f_1([x]) \circ f_2([x])$. If both functions are monotonic in $[x]$, then the equality relation has been verified [17].

The operations below are defined by Kaucher and be will present considering $[x] = [\underline{x}, \overline{x}]$ and $[y] = [\underline{y}, \overline{y}]$.

- **Equivalence:** $[x] = [y] \Leftrightarrow \underline{x} = \underline{y} \wedge \overline{x} = \overline{y}.$

- **Addition:** $[x] + [y] = [\underline{x} + \underline{y}, \overline{x} + \overline{y}].$

- **Difference:** $[x] - [y] = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] = [x] + [-y], \quad where \quad -[y] = [-\overline{y}, -\underline{y}].$

- **Division:** $[x]/[y] = [x] \times [1/\overline{y}, 1/\underline{y}]; if \quad 0 \notin [y].$

- **Union:** $[x] \cap [y] = [\underline{x} \grave{o} \underline{y}, \overline{x} \acute{o} \overline{y}].$

- **Intersection:** $[x]\cup[y]=[\underline{x}ó\underline{y},\bar{x}ò\bar{y}].$
- **Inverse Addition:** $inv[\underline{x},\bar{x}]=[-\underline{x},-\bar{x}].$
- **Square Root:** $\sqrt{}([\underline{x},\bar{x}])=[\sqrt{}(\underline{x}),\sqrt{}(\bar{x})],\quad defined \quad if \quad \underline{x}\geq 0\wedge\bar{x}\geq 0.$
- **Minimum:** $min([\underline{x},\bar{x}],[\underline{y},\bar{y}])=[min(\underline{x},\underline{y}),min(\bar{x},\bar{y})].$
- **Maximum:** $max([\underline{x},\bar{x}],[\underline{y},\bar{y}])=[max(\underline{x},\underline{y}),max(\bar{x},\bar{y})].$

$$x\in[\underline{y},\bar{y}]\Leftrightarrow\begin{cases}\underline{y}\leq x\leq\bar{y}, & if\,[\underline{y},\bar{y}]\in\mathsf{IR}\\ \bar{y}\leq x\leq\underline{y}, & otherwise.\end{cases}$$

- **Association:**

In Table 1 is demonstrated the multiplication operation defined by Kaucher, where: $\mathsf{P}=\{[X]\in\mathsf{KR}\,|\,\underline{x}\geq 0,\bar{x}\geq 0\}$, $\mathsf{Z}=\{[X]\in\mathsf{KR}\,|\,\underline{x}\leq 0\leq\bar{x}\}$, $-\mathsf{P}=\{[X]\in\mathsf{KR}\,|\,-x\in\mathsf{P}\}$, $dual\mathsf{Z}=\{[X]\in\mathsf{KR}\,|\,dualZ\in\mathsf{Z}\}$.

Table 1: Multiplication in Kaucher Interval Arithmetic

$[X]*[Y]$	$[Y]\in\mathcal{P}$	$[Y]\in\mathcal{Z}$	$[Y]\in-\mathcal{P}$	$[Y]\in dual\ \mathcal{Z}$
$[X]\in\mathcal{P}$	$[\underline{x}\underline{y},\bar{x}\bar{y}]$	$[\bar{x}\underline{y},\bar{x}\bar{y}]$	$[\bar{x}\underline{y},\underline{x}\bar{y}]$	$[\underline{x}\underline{y},\underline{x}\bar{y}]$
$[X]\in\mathcal{Z}$	$[\underline{x}\bar{y},\bar{x}\bar{y}]$	$[min(\underline{x}\bar{y},\bar{x}\underline{y}),$ $max(\underline{x}\underline{y},\bar{x}\bar{y})]$	$[\bar{x}\underline{y},\underline{x}\underline{y}]$	0
$[X]\in-\mathcal{P}$	$[\underline{x}\bar{y},\bar{x}\underline{y}]$	$[\underline{x}\bar{y},\underline{x}\underline{y}]$	$[\bar{x}\bar{y},\underline{x}\underline{y}]$	$[\bar{x}\bar{y},\bar{x}\underline{y}]$
$[X]\in dual\ \mathcal{Z}$	$[\underline{x}\underline{y},\bar{x}\underline{y}]$	0	$[\bar{x}\bar{y},\bar{x}\underline{y}]$	$[max(\underline{x}\underline{y},\bar{x}\bar{y})),$ $min(\underline{x}\bar{y},\bar{x}\underline{y})]$

In these operations the operators are real values pair. However, in a computer system they are subsets of real values represented in floating point. This approximation occurs through a special rounding, called directed rounding [18] $\lozenge:\mathsf{IR}\rightarrow\mathsf{IF}$ where $\lozenge[\underline{x},\bar{x}]=[\nabla\underline{x},\Delta\bar{x}]$. In directed rounding, the first and second element are rounded down and up respectively.

The rounded operation about two interval-values $[x]=[\underline{x},\bar{x}]$ and $[y]=[\underline{y},\bar{y}]$ is presented below:

$$[\underline{x},\bar{x}]\lozenge[\underline{y},\bar{y}]\Rightarrow\lozenge([\underline{x},\bar{x}]\circ[\underline{y},\bar{y}]),\ for\circ\in\{+,-,*,/\},\ with\ 0\notin[\underline{y},\bar{y}]\ for\circ=/.$$

This maintains the system integrity, but also conforms to the arithmetic interval definition over space □ℝ.

3 KaucherPy

KaucherPy consists of a robust, easy to use and install package, providing high accuracy methods for numeric processing with interval data using Python programming language.

This section provides framework specifications that were developed, demonstrating the functionalities of each available subpackage. Intended for scientific processing, all steps were carefully developed using the arithmetic implementation report IEEE 1788 [2] and the GPL - General Public License - to provide an open . It presents all the concepts that define KaucherPy: a Kaucher number; package implementation design, arithmetic functions, and exception handling coverage.

The KaucherPy package is organized in such a way as to split the application into standard operations and auxiliary operations. A Kaucher interval represented through the KaucherPy consists of the core package. A number represents a Kaucher object in which its limits are displayed as *np.float64* which consists of the real numeric type provided by NumPy [14]. This occurs to represent numerical data with greater accuracy and flexibility, since NumPy represents numeric values maintaining the Python language operations structure. In this way KaucherPy becomes even more flexible and easily adaptable to applications.

3.1 KaucherPy Design

The Python language was used because it is an interpreted, open source language, with easy extension and cross-platform portability. For these and other reasons, Python has been adopted by the scientific community for different applications, besides containing a wide functions availability and extensive documentation.

A special package has great importance to the KaucherPy operation: the NumPy. It is responsible for supporting mathematical operations with maximum accuracy and for providing data structures as multidimensional vectors. The NumPy use makes KaucherPy compatible with a wide applications variety developed only with real value support, and can be easily adapted to enter ranges provided by KaucherPy. The development of KaucherPy was conducted by the software diagram shown in Figure 1.

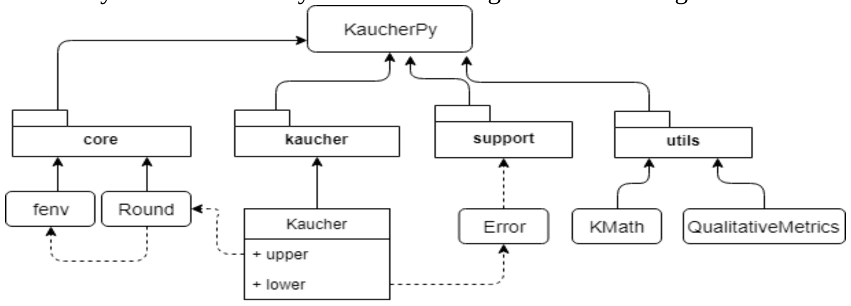


Figure 1: KaucherPy's Architecture

The class *Kaucher* consists of the smallest existing information unity in the package. It also composes the core of all defined arithmetic operations. Therefore, more complex functions can be defined. KaucherPy is linked directly to classes *Round* and *Error*:

• **Round:** behaves as a facilitator to access rounding functions provided by the subpackage *fenv*. These operations are important once introduce the flexibility to introduce the directed rounding to the system. These hardware operations for numerical adjustment should be explored when interval numbers are used. In KaucherPy, the direct rounding concept is maintained as a restriction of the interval arithmetic;

• **Error:** provides all the exception treatments existing in the KaucherPy package to ensure the operations integrity and applications developed with Kaucher numbers.

3.2 Operations in KaucherPy

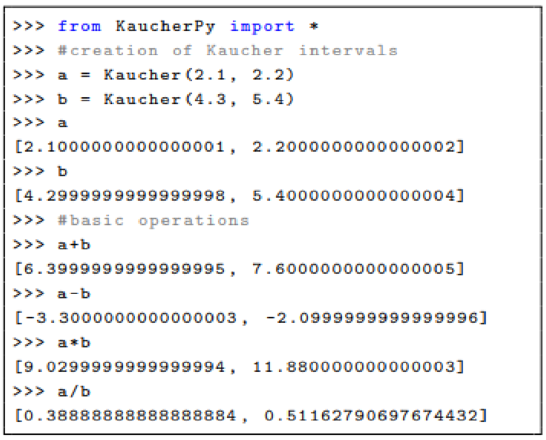
The feasibility of the *KaucherPy* package is guaranteed through the available functionalities. The operations made available for use are related to the definitions of the Kaucher interval extension, thus providing basic arithmetic operations, trigonometric operations, and discrete and continuous qualitative methods. The Table 2 summarizes the operations available for usage with the method signature.

Table 2: Mathematical operations available on KaucherPy

Operation	**Description**
$Kaucher()$	Creating a interval
$+$	Sum
$-$	Subtraction or inversion
$*$	Multiplication
$/$	Division
$\sim$	Dual operation
$sqrt, abs, exp$	Mathematical methods
$Sin, Cos, Tan, Sec, Cot, Cos$	Trigonometric operations
Not, and, or	Logic operations
$=$, $!=$, $<$, $>$, $\geq$, $\leq$, $\subset$, $\subseteq$	Comparative operations

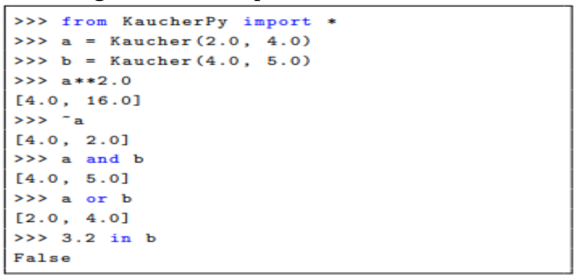
The Figure 2 shows the KaucherPy package request, enabling the use of all its defined operations.

Then, two intervals $[a]$ and $[b]$ are manipulated over basic arithmetic operations. Completely, in the Figure 3 the other operations are presented.



```
>>> from KaucherPy import *
>>> #creation of Kaucher intervals
>>> a = Kaucher(2.1, 2.2)
>>> b = Kaucher(4.3, 5.4)
>>> a
[2.1000000000000001, 2.2000000000000002]
>>> b
[4.2999999999999998, 5.4000000000000004]
>>> #basic operations
>>> a+b
[6.3999999999999995, 7.6000000000000005]
>>> a-b
[-3.3000000000000003, -2.0999999999999996]
>>> a*b
[9.0299999999999994, 11.880000000000003]
>>> a/b
[0.38888888888888884, 0.51162790697674432]
```

Figure 2: Basic operations from Kaucher



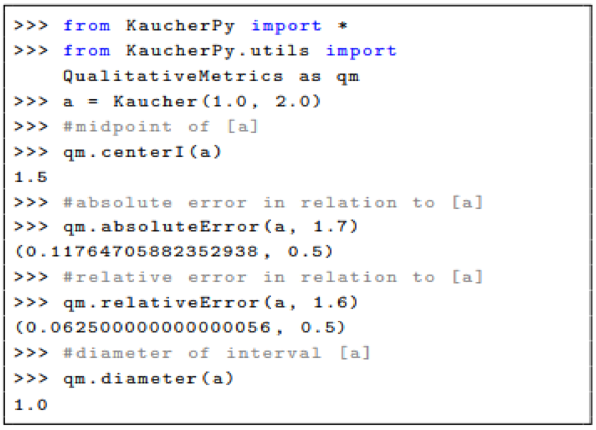
```
>>> from KaucherPy import *
>>> a = Kaucher(2.0, 4.0)
>>> b = Kaucher(4.0, 5.0)
>>> a**2.0
[4.0, 16.0]
>>> ~a
[4.0, 2.0]
>>> a and b
[4.0, 5.0]
>>> a or b
[2.0, 4.0]
>>> 3.2 in b
False
```

Figure 3: Basic operations from KaucherPy

The Table 3 presents the signature for the qualitative interval methods implemented in KaucherPy. Presents the metrics of absolute error, relative error, and diameter intervals. The operation to calculate the range midpoint is also displayed. To use these KaucherPy operations, a call to the *KaucherPy. utils* sub package is required. The use of these methods is illustrated in Figure 4.

Table 3: Qualitative metrics available in *KaucherPy. utils*

Method	**Description**
absoluteError()	Absolute error calculation
relativeError()	Relative error calculation
diameter ()	Diameter of interval calculation
centerI ()	Midpoint calculation



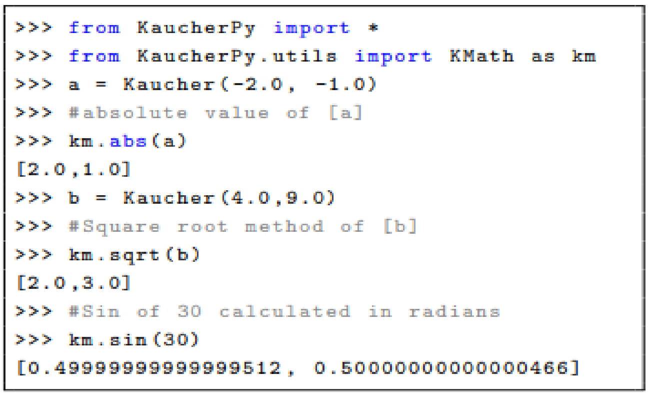
```
>>> from KaucherPy import *
>>> from KaucherPy.utils import
    QualitativeMetrics as qm
>>> a = Kaucher(1.0, 2.0)
>>> #midpoint of [a]
>>> qm.centerI(a)
1.5
>>> #absolute error in relation to [a]
>>> qm.absoluteError(a, 1.7)
(0.11764705882352938, 0.5)
>>> #relative error in relation to [a]
>>> qm.relativeError(a, 1.6)
(0.062500000000000056, 0.5)
>>> #diameter of interval [a]
>>> qm.diameter(a)
1.0
```

Figure 4: Metrics of quality in KaucherPy

Auxiliary mathematical operations such as trigonometric, exponential, absolute value, square root and logarithmic also present on the sub package *KaucherPy. utils*. The Table 4 presents these signature methods, while the Figure 5 presents these operations with use examples.

Table 4: Auxiliary mathematical methods available in *KaucherPy. utils*

Method	**Description**
sin()	Sin function calculation
cos()	Cosine function calculation
tan()	Tangent function calculation
sec()	Secant function calculation
csc()	Cosecant function calculation
cot()	Cotangent function calculation
abs()	Absolute value calculation
exp()	Exponenciation function calculation
log()	Logarithmic function calculation
sqrt()	Square root calculation



```
>>> from KaucherPy import *
>>> from KaucherPy.utils import KMath as km
>>> a = Kaucher(-2.0, -1.0)
>>> #absolute value of [a]
>>> km.abs(a)
[2.0,1.0]
>>> b = Kaucher(4.0,9.0)
>>> #Square root method of [b]
>>> km.sqrt(b)
[2.0,3.0]
>>> #Sin of 30 calculated in radians
>>> km.sin(30)
[0.49999999999999512, 0.50000000000000466]
```

Figure 5: Qualitative metrics of KaucherPy

Some features compatible with the structure provided by the Python language were also compatibilized with KaucherPy, such as union, intersection, minimum and maximum.

3.3 Development plan

The KaucherPy package developed and presented in this paper contains feasibility in relation to its exercise plan, since it was guided through the IEEE definitions and the concepts developed by Kaucher. Some objectives were prioritized for the final project contextualization:

- high numerical accuracy with □ℝ space extension;
- cross-platform interoperability (Unix and Windows);
- compatibility with SciPy and NumPy packages;
- exception handling;
- providing qualitative methods for evaluating results;
- flexibility for compatibility with existing applications.

For exception handling, a simplified schema has been devised so that exceptions already available in Python can be used and extended to the interval domain. The Figure 5 demonstrates the hierarchical exceptions structure in KaucherPy, then, each of them is described.

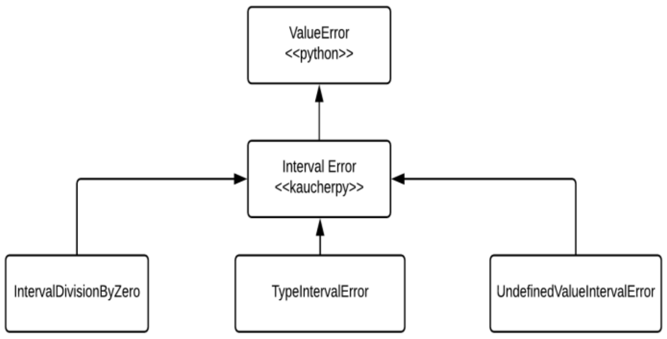


Figure 6: Exception handling diagram in KaucherPy

1. *Interval Error*: generic exceptions class in Kaucher intervals;
2. *TypeInterval Error*: non-numeric types are disabled to work with Kaucher intervals, except compound types such as lists, dictionaries, tuples etc.;
3. *Interval Division By Zero*: type division impossibility $[x]/[y]$ where $0 \in [y]$;
4. *Undefined Valu eInterval Error*: indefinite values are being used in operations.

4 Comparison with interval environments

The numerical tests performed demonstrate KaucherPy applicability and quality in the search for solutions to problems. The purpose of this analysis is to compare KaucherPy with other interval modules, maintaining interest in those that are compatible with the Python language.

Currently there are only two modules compatible with improper intervals according to Kaucher's interval arithmetic: JInterval [9] developed as a package for the Java language and the Pascal-XSC module [17], developed in Pascal. Both implementations are not available in Python. For the latter, its implementation and availability were discontinued.

Among the public available interval-type environments, IntPy [6] represents the use of intervals in Python language. This allows not only an effective available methods comparison, but also the numerical and run-time efficiency in relation to KaucherPy.

To comparative test between the different interval calculation models will use global minimization problems available in the Hedar [8] work. These problems contain interesting features for efficiency analysis, such as: different complexities in each function, convergence under different approaches and existence of local minimums.

The Tables 5 and 6 demonstrate the results achieved in the test scenario elaborated for this work. The Table 5 represents the base execution with real arithmetic demonstrating the minimum solution found for the problem (t_{min}), the amount of iterations required (N_{it}), the dimension of the problem (Dim) and the computational time (t_{cpu}) in seconds.

The y_{min} value presented in each case is calculated by equation 3, which calculates the difference between the actual solution y and the interval result $\hat{y}$ where ε is interpreted as residual operation value.

$$\varepsilon = y - \hat{y} \qquad (3)$$

Table 5: Continuous global optimization problems computed with real arithmetic

	Real			
Problem	**Dim**	**N** $_{It}$	**y** $_{min}$	**t** $_{cpu}$
Ackley (**ACK**)	2	52	4.4408^{-16}	0.002
Dixonprice (**DIX**)	2	476	0.1881	0.014
Ellipse (**ELL**)	2	53	0.0	0.002
Griewank (**GRI**)	2	51	0.0	0.001
Levy (**LEV**)	2	61	1.4997^{-32}	0.002
Michalewicz (**MIC**)	8	532	-5.0324	0.015

Nesterov (**NES**)	2	53	0.0	0.001
Perm (**PER**)	2	52	7.8607^{-5}	0.004
Powell (**POW**)	2	55	0.0	0.003
Rastrigin (**RAS**)	2	51	0.0	0.001
Rosenbrock (**ROS**)	2	73	0.0	0.002
Schwefel (**SCH**)	2	621	2.5709^{-5}	0.012
Sphere (**SPH**)	2	51	0.0	0.001
Sum2 (**SUM**)	2	51	0.0	0.001
Trid (**TRI**)	8	65	-56.0	0.001
Zakharov (**ZAK**)	2	52	0.0	0.001

Table 6: Continuous global optimization problems computed with KaucherPy and IntPy packages

	KaucherPy				**IntPy**			
Pr.	**Dim**	$\mathbf{N}_{It}$	ε	$\mathbf{t}_{cpu}$	**Dim**	$\mathbf{N}_{It}$	ε	$\mathbf{t}_{cpu}$
ACK	2	52	1.776^{-15}	0.354	2	52	1.776^{-15}	0.110
DIX	2	476	1.731^{5}	0.099	−	−	−	−
ELL	2	53	0.0	0.013	2	53	0.0	0.005
GRI	2	51	0.0	0.343	−	−	−	−
LEV	2	61	2.515^{-17}	0.945	−	−	−	−
MIC	8	50	0.106	2.778	−	−	−	−
NES	2	53	0.0	0.008	2	57	0.374	0.006
PER	2	52	2.527^{-10}	0.025	−	−	−	−
PO_1	2	55	0.0	0.018	−	−	−	−
RAS	2	51	0.0	0.344	2	51	0.0	0.106
ROS	2	73	0.0	0.011	−	−	−	−
SCH	2	50	0.0	0.334	−	−	−	−
SPH	2	51	0.0	0.003	2	51	0.0	0.016
SUM	2	51	0.0	0.006	2	51	0.0	0.003
TRI	8	65	0.0	0.041	8	65	0.0	0.016
ZAK	2	52	0.0	0.010	−	−	−	−

The IntPy package, which introduces SIA in the Python language, does not contain some operations required to be considered a scientific package for high accuracy calculations. Complex operations such as trigonometric, hyperbolic, module, square root and exponentiation, and properties such as associativity, commutativity, and variable dependence are not correctly analyzed. In this way, incorrect results can be generated or can not be calculated. The Table 6 demonstrates this limitation found in IntPy, since some of the optimization problems addressed could not be calculated.

In short, KaucherPy's operations remained the same as actual results. Problems like MIC, SCH and TRI converged more quickly into the solution. However, when analyzing the computational effort required, actual arithmetic has shorter execution times. What was expected, since its operations are performed with only a single floating point number. Because KaucherPy uses numbers defined in the *numpy* package and its operations contains definition other than SIA, its runtime is affected.

Comparing the results achieved by the interval packets is notable a shorter execution time in problems solved by IntPy. This is related to their operation definitions, which differ from the way is executed in KaucherPy. For example, the multiplication operation performs a greater number of evaluations on KaucherPy than in IntPy package.

All possible operations to be performed by real arithmetic have also been solved integrally by KaucherPy with similar results and convergence. Presented as an alternative to real arithmetic and providing high accuracy as well as metrics to evaluate the result quality.

5 Other applications for numerical testing

5.1 Simple example of temperature control

An case study developed by Sainz *et. al* [12] consisting of a temperature control model will be used. The computation occur exclusively through the KaucherPy package usage.

The model has a house as the scenario where the central temperature control system is allocated. Considering the forms of heat transmission: radiation, conduction and convection, heat losses can be estimated by equation X, where the energy transmitted in the heat form from inside to outside is calculated by unit time.

$$p_{io} = \frac{t_{in} - t_{out}}{r_{th}}, \tag{4}$$

where:

- t_{in} is the inside temperature;
- t_{out} is the outside temperature;
- r_{th} is the thermal resistance of the house.

The temperature varies in function of the difference between the thermal power produced by the central heating/cooling system and the power lost. This can be measured by:

$$\frac{dt_{in}}{dt} = \frac{p_{h/c} - p_{io}}{\sum_i m_i c_i}, \tag{5}$$

where:

- m_i represent the mass of each element inside of house;
- c_i is the specify heat;
- $p_{h/c}$ corresponds to the heating/cooling power of the system.

The discrete model that allows computing the power required for the system is given by equation 6.

$$p_{h/c}(k) = \frac{t_{in}(k) - t_{out}(k)}{r_{th}} + (t_{in}(k+1) - t_{in}(k)) \frac{\sum_i m_i c_i}{\Delta t} \tag{6}$$

Suppose that r_{th} and $\sum_i m_i c_i$ are parameters of uncertainty measured by determined intervals R_{th} and MC and that $t_{in}(k)$ and $t_{out}(k)$ are variables measured with values contained in the intervals $T_{in}(k)$ and $T_{out}(k)$.

The problem focuss to find, in a single step, a control range $P_{h/c}(k)$ capable of providing sufficient power to reach any temperature $t_{in}(k)$ within a range $T_{in}(k)$ at any temperature $t_{in}(k+1)$ within a range $T_{in}(k+1)$ when the outside temperature is any value in interval $T_{out}(k)$ and the parameter values are within their respective ranges R_{th} and MC.

To calculate a practical example, let us consider some typical values for the parameters of the model:

- $r_{th} \in R_{th} = [0.001, 0.01] \frac{K}{W}$,
- $\sum_i m_i c_i \in MC = [5 \cdot 10^6, 6 \cdot 10^6] \frac{J}{K}$,

- $\Delta t = 120s$.

In the problem solution 6, the values of R_{th}, MC, Δt, $T_{in}(k)$ and $T_{out}(k)$ are proper intervals $\underline{x} < \overline{x}$. The initial temperature $t_{in}(k) = [18.0,18.1]$ and for $T_{in}(k+1)$ an improper interval will be given as input $T_{in}(k+1) = [18.7,18.2]$. We have as a result of 6:

$$p_{h/c}(k) = [26300.0,24100.0]W.$$

Given that one of the input values is an improper interval, KaucherPy is able to solve the problem. The result is also compound an improper interval where $\underline{x} > \overline{x}$ to the required power value applied to the temperature control system. This result is interesting because it has guarantees regarding the uncertainties contained in the values and the desired temperature value given the power applied. In terms of meaning for this inadequate interval mode solution, it is apparently not that significant. But if we think that this result represents only a partial solution of a problem, its application will have effects on the system.

5.2 Linear equations system solution

Using Cramerâ€™s Rule we will seek for the solution to a system of simple linear equations. Cramerâ€™s Rule is an efficient method for finding equation system solutions with an arbitrary number of unknowns compatible with the number of equations using determinant. In the conventional arithmetic the method returns an solution only, if it exists. However, if the system of equations has no solution or has infinite solutions, it is indicates for the determinant equals zero.

In this case no type of interval solution was applied, only the direct use of interval values. We will consider the 2×2 system represented in 7 and 8.

$$[1,3]x_1 + [4,6]x_2 = y_1 = [17,21] \quad (7)$$

$$[3,5]x_1 + [-2,0]x_2 = y_2 = [5,9] \quad (8)$$

Generalizing a system of two linear equations into two variables will have something like:

$$a_1x_1 + b_1x_2 = y_1$$

$$a_2x_1 + b_2x_2 = y_2$$

The solution for Cramerâ€™s Rule is given using determinant as:

$$x_1 = \frac{D_{x_1}}{D} = \frac{\begin{bmatrix} y_1 & b_1 \\ y_2 & b_2 \end{bmatrix}}{\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}}, D \neq 0 \quad (9)$$

$$x_2 = \frac{D_{x_2}}{D} = \frac{\begin{bmatrix} a_1 & y_1 \\ a_2 & y_2 \end{bmatrix}}{\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}}, D \neq 0 \quad (10)$$

Applying the 9 and 10 models on the coefficients of 7 and 8 we will have something like:

$$x_1 = \frac{D_{x_1}}{D} = \frac{\begin{bmatrix} [17,21] & [4,6] \\ ] & [-2,0]] \end{bmatrix}}{\begin{bmatrix} [1,3] & [4,6] \\ ] & [-2,0] \end{bmatrix}} = [8.0,0.5555555555555556] \quad (11)$$

$$x_2 = \frac{D_{x_2}}{D} = \frac{\begin{bmatrix} [1,3] & [17,21] \\ ] & [5,9] \end{bmatrix}}{\begin{bmatrix} [1,3] & [4,6] \\ & [-2,0] \end{bmatrix}} = [8.333333333333332, 0.6666666666666667] \qquad (12)$$

At this point we can already see the presence of improper intervals in the values found in 11 and 12. Which is not a problem since we are operating with the generalization of Kaucher's intervals in KaucherPy.

Even already calculated at 12, we can test the possibility of finding the value of unknown x_2 by substitution. So, using the obtained value at 11 back in the equation 7 we have:

$$[1,2][8.0, 0.5555555555555556] + [4,6]x_2 = [17,21]$$

Performing the algebraic manipulations we will reach the value for $x_2 = [2.5555555555555554, 3.25]$. Obviously the result found is different from that obtained in 12. Therefore, we use the values in 11 and 12 to try to reach the same y_1 and y_2 at 7 and 8 respectively. So we have something like:

$$[1,3][8.0, 0.5555555555555556] +$$

$$[4,6][8.333333333333332, 0.6666666666666667] = y_1;$$

$$[3,5][8.0, 0.5555555555555556] +$$

$$[-2,0][8.333333333333332, 0.6666666666666667] = y_2.$$

With results for $y_1 = [18.22222222222222, 21.166666666666668]$ and $y_2 = [22.666666666666664, 2.7777777777777778]$ visibly we didn't reach the expected solutions. Moreover, associated with the uncertainty as to whether or not there is a solution to the system of linear equations used in this example, another mathematical insufficiency of the present arithmetic is added: when solving an equation of type $[a]+[x]=[c]$ where $[x]$ is the unknown, $[x]$ not the algebraic solution and when apply the found value for $[x]$ again in the equation, we see that the equation is no longer valid.

For example, for $[2,3]+[x]=[3,7]$ the solution is given by $[x]=[3,7]-[2,3]=[0,5]$. However, when $[x]$ returns to the equation: $[2,3]+[0,5]=[2,8]\neq[3,7]$, not corresponding to the original right-hand-side of the equation. There are points when intervals form a subgroup rather than a group, making the solution process infeasible.

This test was useful for demonstrate the arithmetics limitations at Kaucher's arithmetic, some of them inherited from SIA. These incorrect solutions are not a limitation or problem of the KaucherPy package.

Additionally, lack of invertibility is highlighted, where $[x]-[x]\neq 0$ unless $[x]$ is a degenerated interval ($[2,4]-[2,4]=[-2,2]\neq[0,0]$ e.g.). In this sense the intervals have a dual relation, ie $[2,4]=dual[4,2]$ with os similar to the introduction of negative values in conventional arithmetic, R. Thus, with the "*dual*" operation, zero can be acquired in the interval generalization of Kaucher arithmetic: $[2,4]-dual[2,4]=[0,0]$. In the KaucherPy package we get this operation by means of "$:[x]$" signature.

6 Conclusions

Kaucher's arithmetic has in his published theory an overview of theoretical proofs of its validity because of the deficiencies of Moore's arithmetic, SIA. Therefore, this paper presents the interval package named KaucherPy, developed in Python with GPL-licensed, providing high-accuracy solutions for Kaucher interval arithmetic. Its subpackages architecture facilitates its use in applications involving the construction of numerical solutions, as it makes use of numeric representation of *numpy* package.

All methods were implemented from the theoretical definitions of Kaucher's interval extension keeping the operational integrity of the package. Development following the guidelines IEEE-1788 make KaucherPy an open source arithmetic package for scientific computation. The implemented methods were explicitly presented in order to facilitate their use. The operability of the package was demonstrated through of computation of numerical examples, beginning with a set of global optimization problems.

Finding the solution to these objective functions shows the ability to generate solutions to more complex problems and the quality of results that can be obtained using the KaucherPy package. The simple example of temperature control demonstrated the possibility of improper intervals being generated where only Kaucher arithmetic is able to operate.

Then an seemingly simple example of linear equations system make way for further explanation about the possibilities and barriers that can be faced in numerical calculations involving used. In this example it could be seen that at some points in the process of searching for a solution improper intervals can be generated and Kaucher's arithmetic is able to handle. However, there is no guarantee that the best solution will be obtained because it has inherited deficiencies of SIA.

Although Kaucher's interval extension is from the 1980s, recent researches have been developed using his concepts. The aims to development of KaucherPy's package promote to support and facilitate the make solution of problems around of numerical computations with intervals. The package is open source and can easily be found in Python repositories.

References

[1]. J. E. Dutra, "Java-xsc: uma biblioteca java para computações intervalares," *M. SC. diss. DIMAp/UFRN*, 2000.

[2]. I. S. 1788-2015, "Ieee std 1788-2015 - ieee standard for interval arithmetic," 2016.

[3]. S. M. Rump, "Intlabâ€"interval laboratory," in *Developments in reliable computing*, pp. 77–104, Springer, 1999.

[4]. R. Klatte, U. Kulisch, A. Wiethoff, and M. Rauch, *C-XSC: A C++ class library for extended scientific computing*. Springer Science & Business Media, 2012.

[5]. W. F. Mascarenhas, "Moore: Interval arithmetic in modern c++," *arXiv preprint arXiv:1611.09567*, 2016.

[6]. F. VarjÃ£o, "Computaã§ã£o cientã-fica auto validã¡vel em python.," Master's thesis, Universidade Federal de Pelotas, 2011.

[7]. L. H. De Figueiredo and J. Stolfi, "Affine arithmetic: concepts and applications," *Numerical Algorithms*, vol. 37, no. 1-4, pp. 147–158, 2004.

[8]. A.-r. Hedar and A. Ahmed, "Studies on metaheuristics for continuous global optimization problems," 2004.

[9]. D. Y. Nadezhin and S. I. Zhilin, "Jinterval library: Principles, development, and perspectives.," *Reliable Computing*, vol. 19, no. 3, pp. 229–247, 2013.

[10]. E. Kaucher, "Interval analysis in the extended interval space ir," in *Fundamentals of Numerical Computation (Computer-Oriented Numerical Analysis)*, pp. 33–49, Springer, 1980.

[11]. W. A. Lodwick, *Constrained interval arithmetic*. Citeseer, 1999.

[12]. M. A. Sainz, J. Armengol, and R. C. i Puig, *Modal Interval Analysis: New Tools for Numerical Information*. Springer, 2014.

[13]. R. E. Moore, "Interval analysis, vol. 4," 1966.

[14]. N. Developers, "Numpy," *NumPy Numpy. Scipy Developers*, 2013.

[15]. A. Piegat and M. Landowski, "Is the conventional interval-arithmetic correct?," *Journal of Theoretical and Applied Computer Science*, vol. 6, no. 2, pp. 27–44, 2012.

[16]. A. Piegat and M. Landowski, "Two interpretations of multidimensional rdm interval arithmetic: Multiplication and division," *International Journal of Fuzzy Systems*, vol. 15, no. 4, pp. 486–496, 2013.

[17]. E. D. Popova, "Extended interval arithmetic in ieee floating-point environment," *Interval Computations*, vol. 4, pp. 100–129, 1994.

[18]. H. Ratschek and J. Rokne, *New Computer Methods for Global Optimization. Chichester*. Horwood Chichester, 1988.