Serdica J. Computing **4** (2010), 123–132

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

A SOLVER FOR COMPLEX-VALUED PARAMETRIC LINEAR SYSTEMS*

Evgenija Popova, Lyubomir Kolev, Walter Krämer

Abstract. This work reports on a new software for solving linear systems involving affine-linear dependencies between complex-valued interval parameters. We discuss the implementation of a parametric residual iteration for linear interval systems by advanced communication between the system *Mathematica* and the library C-XSC supporting rigorous complex interval arithmetic. An example of AC electrical circuit illustrates the use of the presented software.

1. Introduction. Scientific and engineering problems described by systems of linear algebraic equations involving uncertain model parameters include problems in engineering analysis or design [3, 5, 8], control engineering [2], etc. Significant research in this field is directed towards the use of intervals to represent the uncertain quantities in such systems.

For many years, worst-case tolerance analysis (WCTA) of linear lumped-parameter electric circuits has been the subject of numerous investigations by

ACM Computing Classification System (1998): D.2.12, J.2, G.4.

Key words: Parametric linear system, complex intervals, validated interval software, C-XSC, *Mathematica*, electrical circuits.

*This work was partly supported by the DFG grant GZ: KR1612/7-1, AOBJ: 570029.

interval methods (cf. [5, 6], [3] and the literature cited therein). Most problem formulations for linear circuits with tolerance-affected electrical parameters lead to systems of linear interval equations. Depending on the class of the electrical circuits investigated, direct current (DC) or alternating current (AC) circuits, the linear system of equations describing the circuit can be real or complex. In both cases, the system's coefficients can be either independent intervals or functions of independent interval parameters. The early efforts for WCTA of linear circuits were based on various problem formulations using methods for nonparametric interval linear systems [5]. The recent investigations focus on methods for linear systems where the elements of the matrix and the right-hand side are mainly linear functions of interval parameters [3, 6]. However, all considerations of complex-valued interval linear systems stemming from WCTA, known to us so far, try to reformulate the original system of equations into an equivalent real representation. This way, the new real interval linear system has twice as many equations and more complicated parameter dependencies. Respectively, the methods reported for these systems give only approximate solutions to the corresponding tolerance problem.

In this work we report on new software for solving linear systems where the coefficients of the matrix and the elements of the right hand side are affine-linear functions of parameters varying within given complex intervals. A general-purpose parametric fixed-point iteration is implemented by integrating the interactive symbolic-numeric environment of the system *Mathematica* [12] with the C++ library for scientific computing C-XSC, which support rigorous complex interval arithmetic [4]. In Section 2, the parametric residual iteration method for linear interval systems is introduced. Section 3 discusses the software implementation and the communication between *Mathematica* and the interval software library C-XSC. In Section 4, the new parametric solver is illustrated by an example of AC electrical circuit. The example is quite general and illustrates a wide class of AC circuit models that can be solved by the discussed parametric solver. Finally, some conclusions are given.

2. The Computing Method. Let $\mathbb{T} \in \{\mathbb{R}, \mathbb{C}\}$ be the set of real or complex numbers. By $\mathbb{T}^m, \mathbb{T}^{m\times n}$ denote the set of real/complex vectors with m components and the set of corresponding $m \times n$ matrices, respectively. A real compact interval is defined as $[a] = [\underline{a}, \overline{a}] := \{a \in \mathbb{R} \mid \underline{a} \leq a \leq \overline{a}\}$ and a rectangular complex interval $[z]$ is defined by a pair of two real intervals $[x], [y]$: $[z] = [x] + \imath[y] = \{z = x + \imath y \mid x \in [x], y \in [y]\}$. By $\mathbb{IT}^m, \mathbb{IT}^{m\times n}$ we denote the corresponding interval m-vectors and interval $m \times n$ matrices. We assume that the reader is familiar with the conventional interval arithmetic [1].

Consider a linear algebraic system

$$A(p) \cdot x \;=\; b(p), \tag{1a}$$

where the coefficients of the $n \times n$ matrix $A(p)$ and the vector $b(p)$ are functions of m parameters varying within given intervals

$$a_{ij}(p) = a_{ij}(p_1, \ldots, p_m), \qquad b_i(p) = b_i(p_1, \ldots, p_m), \qquad i, j = 1, \ldots, n, \tag{1b}$$

$$p \;\in\; [p] = ([p_1], \ldots, [p_m])^\top \in \mathbb{IT}^m. \tag{1c}$$

The set of solutions to (1a)–(1c), called the *parametric solution set*, is

$$\Sigma = \Sigma\,(A(p), b(p), [p]) \;:=\; \{x \in \mathbb{T}^n \mid \exists p \in [p] \in \mathbb{IT}^m,\; A(p) \cdot x = b(p)\}\,. \tag{2}$$

The set Σ is compact if $A(p)$ is nonsingular for every $p \in [p]$. For a nonempty bounded set $\mathcal{S} \subseteq \mathbb{T}^n$, define its interval hull by $\Box\,\mathcal{S} := [\inf \mathcal{S},\, \sup \mathcal{S}] = \cap\{[s] \in \mathbb{IT}^n \mid \mathcal{S} \subseteq [s]\}$. Since it is quite expensive to obtain Σ or $\Box\,\Sigma$, we seek an interval vector $[y]$ for which it is guaranteed that $[y] \supseteq \Box\,\Sigma \supseteq \Sigma$.

The following theorem gives a self-verified method for bounding the solution set of a parametric linear system. It is a general-purpose method since it does not assume any particular structure among the parameter dependencies. The method originates in the inclusion theory for nonparametric problems, which is discussed in many works (cf. [11] and the literature cited therein).

Theorem 1. *Consider a parametric linear system defined by* (1a)–(1c). *Let* $R \in \mathbb{T}^{n\times n}$, $[y] \in \mathbb{IT}^n$, $\tilde{x} \in \mathbb{T}^n$ *be given. Define* $[z] \in \mathbb{IT}^n$, $[C] \in \mathbb{IT}^{n\times n}$ *by*

$$\begin{aligned} [z] \;&:=\; \Box\{z(p) = R\,(b(p) - A(p)\tilde{x}) \mid p \in [p]\}, \\ [C] \;&:=\; \Box\{C(p) = I - R \cdot A(p) \mid p \in [p]\}, \end{aligned}$$

where I *denotes the identity matrix. Define* $[v] \in \mathbb{IT}^n$ *by means of the following Gauss-Seidel iteration*

$$1 \le i \le n \;:\; [v_i] := \{[z] + [C] \cdot ([v_1], \ldots, [v_{i-1}], [y_i], \ldots, [y_n])^\top\}_i.$$

If $[v] \subsetneqq [y]$, *then* R *and every matrix* $A(p)$ *with* $p \in [p]$ *are regular, and for every* $p \in [p]$ *the unique solution* $\widehat{x} = A^{-1}(p)b(p)$ *of* (1a)–(1c) *satisfies* $\widehat{x} \in \tilde{x} + [v]$.

The above theorem generalises [11, Theorem 4.8] by stipulating a sharp enclosure of $C(p) := I - R\cdot A(p)$ for $p \in [p]$, instead of using the interval extension $C([p])$. In case of affine-linear parameter dependencies

$$\begin{aligned} a_{ij}(p) := a_{ij,0} + \sum_{\mu=1}^{m} a_{ij,\mu} p_\mu, \qquad & b_i(p) := b_{i,0} + \sum_{\nu=1}^{m} b_{i,\mu} p_\mu, \\ a_{ij,\mu}, b_{i,\mu} \in \mathbb{T}, \qquad & \mu = 0, \ldots, m,\; i, j = 1, \ldots, n \end{aligned}$$

the interval functions $z(p), C(p)$ have explicit representation and

$$\begin{aligned} [z] &:= R(b^{(0)} - A^{(0)}\tilde{x}) + \sum_{\mu=1}^{m} [p_\mu](R(b^{(\mu)} - A^{(\mu)}\tilde{x})) \\ [C] &:= I - R \cdot A^{(0)} - \sum_{\mu=1}^{m} [p_\mu](R \cdot A^{(\mu)}), \end{aligned}$$

where $A^{(\mu)} := (a_{ij,\mu}) \in \mathbb{T}^{n\times n}$, $b^{(\mu)} := (b_{i,\mu}) \in \mathbb{T}^n$, $\mu = 0, \ldots, m$.

When aiming to compute a self-verified enclosure of the solution to a parametric linear system by the above inclusion method, a fixed-point iteration scheme discussed in more detail in [10] proves to be very useful.

3. Software Tools. A variety of publicly-available software for the solution of parametric interval linear systems for the *Mathematica* [7, 8] and C-XSC [10] environments has been developed. The particular solvers differ with respect to the type of the dependencies involved in the linear system to be solved and the implemented solution method. To our knowledge, no software tools for solving complex-valued parametric interval linear systems have been reported and used so far.

The basic goals of self-validating methods are to deliver rigorous results by computations in finite precision arithmetic, including the proof of existence (and possibly uniqueness) of a solution. To achieve this goal the inclusion theorems should be verifiable on computers, the latter can be done by a rigorously implemented interval arithmetic. The interactive environment of *Mathematica* [12] and its analytic computations allow flexible generation of the parametric systems stemming from WCTA of el. circuits. However *Mathematica* supports numerical intervals and the corresponding arithmetic on all numerical data types, except for complex numbers. This makes the implementation of the complex-valued parametric linear solver difficult. On the other hand, the C++ library for scientific computing C-XSC supports predefined interval arithmetic with maximum accuracy on floating-point real and complex numbers, as well as in the corresponding vector/matrix spaces [4]. The provided exact dot products (based on a long accumulator) over all numerical data types enable C-XSC users to compute interval enclosures with very high accuracy. For the sake of efficiency it will be better for the computation of matrix inversion in floating-point (the preconditioner) to be done by an external optimized software. In our implementation of the complex-valued parametric interval linear solver we have used an advanced technology of communication protocols to develop a versatile software integrating *Mathematica* with C-XSC. Namely, the symbolic-numeric system formulation

and all floating-point complex computations from the initialization step of the algorithm are done in *Mathematica*. Then, via the *MathLink* communication protocol, all the numerical data are transferred to an external C/C-XSC program module which implements in C-XSC the interval enclosures and the verification step required by the numerical method. If the iteration step is successful, the obtained solution enclosure is transferred back to *Mathematica*, otherwise an error message is triggered back.

All functions necessary for solving a complex-valued parametric interval linear system are packaged in a template file according to the *MathLink* technology. The template file together with the corresponding communication module can then be processed and compiled to binary code which can be installed and used in any *Mathematica* session. Since C-XSC uses special data types for representing intervals, the main purpose of the developed communication module is to initialize new variables having the corresponding specific data types with the incoming *Mathematica* data, and after the actual C-XSC computations to transform the computed results into variables of fundamental C data types that will be passed back to *Mathematica*. More details about the *MathLink* technology and how to use it for the interoperability between *Mathematica* and C-XSC can be found in [9] and in the implementation source code which is available at `http://www.math.bas.bg/ epopova/papers/complexParLinSolveML.zip`. End-users who do not have *Mathematica* or do not want to establish a communication with external programs can run *Mathematica* and the available parametric solvers remotely via a webComputing service framework.

Below, we briefly outline the functionality of the newly developed integrated *Mathematica*/C-XSC software and illustrate its use from within a *Mathematica* session.

4. Numerical Example. The circuit shown in Fig. 1 consists of five nodes and eleven branches. The parameters of the system have the following nominal values:

$$
\begin{aligned}
&e_1 = e_2 = 100V, \quad e_5 = e_7 = 10V,\\
&Z_j = R_j + \imath X_j \in \mathbb{C}, \quad R_j = 100\Omega, \quad X_j = \omega L_j - \frac{1}{\omega C_j}, \quad j = 1, \ldots, 11,\\
&\omega = 50, \quad X_{1,2,5,7} = \omega L_{1,2,5,7} = 20, \quad X_3 = \omega L_3 = 30,\\
&X_4 = -\frac{1}{\omega C_4} = -300, \quad X_{10} = -\frac{1}{\omega C_{10}} = -400, \quad X_{6,8,9,11} = 0.
\end{aligned}
$$

The electric parameters: resistance R_j, inductance L_j, and capacitance C_j, $j = 1, \ldots, 11$, of the branch elements are considered to be unknown but bounded

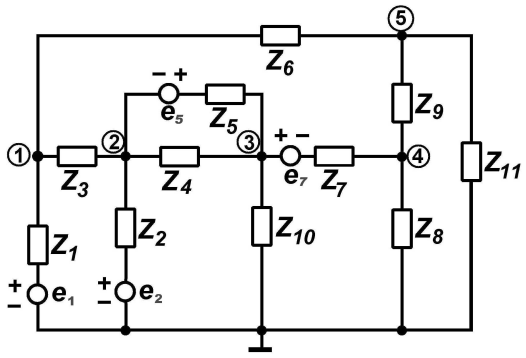

Fig. 1. Example of an electrical circuit having complex parameters

to vary within given tolerance intervals. The source voltages $e_1 = e_2$, $e_5 = e_7$ can be considered also as unknown but bounded. The tolerance analysis problem is to find bounds for all node voltages V_i, $i = 1, \ldots, 5$. This circuit has been considered in [5] for the special case of a resistive circuit where all parameters involved are pure resistors (no inductances L_j and capacitances C_j).

The method of "loop analysis", is used to set up the following system of parametric equations

$$\begin{pmatrix} \frac{1}{Z_1}+\frac{1}{Z_3}+\frac{1}{Z_6} & -\frac{1}{Z_3} & 0 & 0 & -\frac{1}{Z_6} \\ -\frac{1}{Z_3} & \frac{1}{Z_2}+\frac{1}{Z_3}+\frac{1}{Z_4}+\frac{1}{Z_5} & -\frac{1}{Z_4} & 0 & 0 \\ 0 & -\frac{1}{Z_4} & \frac{1}{Z_4}+\frac{1}{Z_5}+\frac{1}{Z_7}+\frac{1}{Z_{10}} & -\frac{1}{Z_7} & 0 \\ 0 & 0 & -\frac{1}{Z_7} & \frac{1}{Z_7}+\frac{1}{Z_8}+\frac{1}{Z_9} & -\frac{1}{Z_9} \\ -\frac{1}{Z_6} & 0 & 0 & -\frac{1}{Z_9} & \frac{1}{Z_6}+\frac{1}{Z_9}+\frac{1}{Z_{11}} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{pmatrix} = \begin{pmatrix} \frac{e_1}{Z_1} \\ \frac{e_2}{Z_2}-\frac{e_5}{Z_5} \\ \frac{e_5}{Z_5}+\frac{e_7}{Z_7} \\ -\frac{e_7}{Z_7} \\ 0 \end{pmatrix}.$$

Without loss of generality, we change the parameters in the system and substitute $p_j = 1/Z_j$, $j = 1, \ldots, 11$. This way the parametric system involves affine-linear dependencies in the matrix. The dependencies in the right-hand side vector are nonlinear. Since $e_{1,2,5,7}$ are involved only in the right-hand side and linearly, the parametric solution set will depend linearly on these parameters. Thus, the worst-case values for these parameters are at particular end-points of the corresponding tolerance intervals. This is why, when demonstrating below

our new software, we solve the above parametric system with the nominal values of the parameters $e_{1,2,5,7}$.

Suppose that the above data are available in a *Mathematica* session, say variables `cAp` and `cBpe` contain the system matrix and the right-hand side vector, respectively, as defined above. The variables `trPMid` and `trE` are set up to lists of transformation rules specifying the nominal values for the parameters p and e, respectively. A *Mathematica* function, available in the *Mathematica* demonstration notebook[1] accompanying this paper, is developed to generate complex intervals with specified tolerances from given nominal complex values. Let this function be used to generate a list `trPInt10` of transformation rules assigning 10% complex tolerance intervals to the parameters p.

We suppose that `complexParLinSolveML.tm/cpp` have been compiled to an external *MathLink*-compatible program `complexParLinSolveML`, which can be installed in any *Mathematica* session. Below, the *Mathematica* `Install` function launches the program and opens a link through which the external functions can be called from within *Mathematica*.

```
In[9] := lnk = Install["complexParLinSolveML"]
Out[9] = LinkObject[./complexParLinSolveML, 2, 2]
```

The evaluation of `Names["complexParLinSolveML‘*"]` will give the functions available in this context and `?complexParLinSolve` creates an online message specifying the syntax and the use of this function which we are interested in using. Now, we are ready to compute an outer enclosure of the solution set of the above complex-valued parametric system.

```
In[13] := res = complexParLinSolve[cAp, cBpe /. trE, trPInt10]
Out[13] = {Complex[{48.7778, 64.0264}, {-6.97476, -0.892966}],
  Complex[{40.164, 54.8626}, {-8.03546, -1.4352}],
  Complex[{12.4068, 21.7213}, {-0.939578, 6.92676}],
  Complex[{5.09736, 14.6988}, {-1.42627, 3.03358}],
  Complex[{16.1099, 28.0902}, {-2.98966, 0.90285}]}
```

Note that the two arguments of the function `Complex` are lists containing the end-points of the real and the imaginary part intervals, respectively. The output is designed this way in order to give the C-XSC computed result communicated straightforward in the internal representation. If necessary, the *Mathematica* function `Interval` can be applied to have the interval end-points outwardly rounded and for subsequent interval computations in *Mathematica*.

[1]The Mathematica demonstration notebook is involved in the archive
`http://www.math.bas.bg/ epopova/papers/complexParLinSolveML.zip`

```
In[14] := Map[Interval, res, {2}] // InputForm
Out[14] =
{Complex[Interval[{48.77781998049774, 64.0264187993989}],
         Interval[{-6.974763234039071, -0.8929661048388836}]],
 Complex[Interval[{40.16395400847202, 54.86256471848972}],
         Interval[{-8.03546458638071, -1.4351990318016539}]],
 Complex[Interval[{12.406832479931252, 21.721328446565188}],
         Interval[{-0.9395775103974154, 6.926762112296828}]],
 Complex[Interval[{5.097357562946341, 14.69883953920892}],
         Interval[{-1.4262688553317318, 3.0335782454122757}]],
 Complex[Interval[{16.10989659532433, 28.09024869869297}],
         Interval[{-2.989656508482889, 0.9028498588837516}]]}
```

The function `InpuForm` is used to display all digits in the mantissas.

The present complex parametric solver works for $A^{(\mu)} \in \mathbb{R}^{n\times n}$, $b^{(\mu)} \in \mathbb{R}^n$ but it can be easily generalized for $A^{(\mu)} \in \mathbb{T}^{n\times n}$, $b^{(\mu)} \in \mathbb{T}^n$, $\mu = 0, \ldots, m$. The results obtained above were compared to the solution of the equivalent real-valued parametric system involving $2n$ unknowns and rational parameter dependencies. Both the complex-valued parametric solver for affine-linear dependencies and the real-valued parametric solver for rational dependencies, implementing the same iteration method, produced enclosures of similar accuracy for this problem. The transparent communication of numerical data between *Mathematica* and C-XSC allows rigorous comparisons of interval results and easy debugging of new C-XSC code.

The communication module is designed in such a way that all error messages generated during the execution of the external C-XSC code are triggered back to *Mathematica*. Due to lack of space these are illustrated in the demonstration notebook. Once installed in a *Mathematica* session, the package `complexParLinSolveML`, resp. the function `complexParLinSolve`, can be used interactively with different values for the parameters and for different parametric systems. An advanced application of this software to WCTA of the above circuit will be presented in subsequent work.

5. Conclusion. In this work we have used the advanced technology of communication protocols for developing of new software, integrated between *Mathematica* and C-XSC, that solves complex-valued parametric linear systems. While complex-valued linear systems are usually solved by transforming the original systems into equivalent real systems of double size and involving more complicated parameter dependencies, the new complex parametric solver allows straightforward bounding of the solution to the original system.

The new self-verified parametric solver for complex-valued linear systems can be the basic ingredient in a general framework for the computer-assisted proof of global and local monotonicity properties of the parametric solution [8]. For the WCTA the latter means proving and identifying which vertices of the parameter set are involved in the worst case parameter set, although the circuit response is not monotonic with respect to all the uncertain circuit parameters. Based on these properties (the worst case parameter set), a guaranteed and highly accurate enclosure of the exact interval hull of the solution set (the worst cases of circuit response) can be computed.

The implementation demonstrates *MathLink* communication of complex intervals between *Mathematica* and C-XSC, as well as the embedding of higher-level C-XSC functions of complex interval arguments into *Mathematica*. It can be a sample for other future embeddings of the C-XSC complex interval arithmetic and other C-XSC functions into *Mathematica*. This approach of software interoperability is especially suitable when interval methods become part of large simulation systems or electronic-design automation tools. In this case the application benefits from both the symbolic and interactive environment (e.g. *Mathematica*) and the speed and accuracy of the external compiled language (C-XSC) software.

The presented methodology and new software are applicable in the context of problems from any other domain that require solution of complex-valued parametric linear systems.

REFERENCES

[1] Alefeld G., J. Herzberger. Introduction to Interval Computations. Academic Press, New York, 1983.

[2] Barmish B. R. New Tools for Robustness of Linear Systems. MacMillan, New York, 1994.

[3] Dreyer A. Interval Analysis of Analog Circuits with Component Tolerances. Shaker Verlag, Aachen, Germany, 2005 Doctoral thesis, TU Kaiserslautern, 2005.

[4] Hofschuster W., W. Krämer. C-XSC 2.0: A C++ Library for Extended Scientific Computing. In: Numerical Software with Result Verification (Eds R. Alt, A. Frommer, R. B Kearfott, W. Luther) Lecture Notes in Computer Science, **2991** (2004), 15–35.

[5] Kolev L. Interval Methods for Circuit Analysis. World Scientific, 1993.

[6] Kolev L. Worst-case tolerance analysis of linear DC and AC electric circuits. *IEEE Transactions on Circuits and Systems*, **49** (2002), No 12, 1–9.

[7] Popova E. Parametric Interval Linear Solver. *Numerical Algorithms*, **37** (2004), No 1–4, 345–356.

[8] Popova E. Computer-Assisted Proofs in Solving Linear Parametric Problems. In: Conference Post-Proceedings of the 12th GAMM-IMACS International Symposium on Scientific Computing, Arithmetic and Validated Numerics (SCAN 2006), Duisburg-Essen, IEEE Computer Society Press. http://doi.ieeecomputersociety.org/10.1109/SCAN.2006.12, 2007.

[9] Popova E. Mathematica Connectivity to Interval Libraries filib++ and C-XSC. In: Numerical Validation in Current Hardware Architectures (Eds A. Cuyt, W. Krämer, W. Luther, P. Markstein), Lecture Notes in Computer Science **5492**, Springer, Berlin/Heidelberg, 2009, 117–132.

[10] Popova E., W. Krämer. Inner and Outer Bounds for the Solution Set of Parametric Linear Systems. *J. of Computational and Applied Mathematics*, **199** (2007), No 2, 310–316.

[11] Rump S. Verification Methods for Dense and Sparse Systems of Equations. In: Topics in Validated Computations (Ed J. Herzberger), N. Holland, 1994, 63–135.

[12] Mathematica, Version 5.2, Champaign, IL,Wolfram Research Inc., 2005.

Evgenija Popova
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev Str., Bl. 8
1113 Sofia, Bulgaria
e-mail: epopova@bio.bas.bg

Lyubomir Kolev
Faculty of Automation
Technical University
8, Kliment Ohridski Str.
1756 Sofia, Bulgaria
e-mail: kolev.l@mail.bg

Walter Krämer
Wissenschaftliches Rechnen/Softwaretechnologie
Bergische Universität Wuppertal
Faculty of Mathematics and Natural Sciences
Gaußstr. 20
D-42097 Wuppertal, Germany
email: kraemer@math.uni-wuppertal.de

Received November 5, 2009
Final Accepted February 4, 2010