ИНСТИТУТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

INSTITUTE OF MATHEMATICS AND INFORMATICS

Секция Биоматематика
Department Biomathematics

БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ

BULGARIAN ACADEMY OF SCIENCES

Решаване на линейни системи с полиномиални зависимости между параметрите

Ю. Гарлоф, Е. Попова, А. Смит

Solving Linear Systems with Polynomial Parameter Dependency

J. Garloff, E. Popova, A. Smith

PREPRINT № 1/2009

Sofia
January 2009

Solving Linear Systems with Polynomial Parameter Dependency

J. Garloff* E. Popova A. P. Smith

University of Applied Sciences / HTWG Konstanz
Department of Computer Science
Postfach 100543, D-78405 Konstanz, Germany
e-mail: garloff@htwg-konstanz.de

Institute of Mathematics & Informatics
Bulgarian Academy of Sciences
Acad. G. Bonchev str., bldg. 8, 1113 Sofia, Bulgaria
e-mail: epopova@bio.bas.bg

University of Applied Sciences / HTWG Konstanz
Institute of Applied Research
Postfach 100543, D-78405 Konstanz, Germany
e-mail: smith@htwg-konstanz.de

Abstract. A wide range of scientific and engineering problems can be described by systems of linear algebraic equations involving uncertain model parameters. We report on new software tools for solving linear systems where the coefficients of the matrix and the right hand side are multivariate polynomials or rational functions of parameters varying within given intervals. A general-purpose parametric fixed-point iteration is combined with efficient tools for range enclosure based on the Bernstein expansion of multivariate polynomials. A C++ software package for constructing the Bernstein enclosure of polynomial ranges, based on the interval library `filib++`, is integrated into a *Mathematica* package for solving parametric systems via the *MathLink* communication protocol. We discuss an advanced application of the general-purpose parametric method to linear systems obtained by standard FEM analysis of mechanical structures and illustrate the efficiency of the new parametric solver.

Keywords: parametric linear system, interval parameter, polynomial range, Bernstein expansion, mechanical structure.

AMS subject classification: 65G20, 15A06, 74S05

1 Introduction

Scientific and engineering problems described by systems of linear algebraic equations involving uncertain model parameters include problems in engineering analysis or design [5, 6, 20, 29], con-

*corresponding author

trol engineering [2, 3, 35], etc. Causes of uncertainty in the model parameters are measurement imprecision, round-off errors, and various other kinds of inexact knowledge.

Significant research in this field is directed towards the use of intervals to represent the uncertain quantities in such systems. When uncertain parameters are modelled by bounded intervals, the problem can be formulated as an interval linear system. Dependencies between such interval parameters may be linear or nonlinear in nature, with the former, simpler, case having been more extensively studied. In the latter case there may be highly nontrivial dependencies between the parameters.

One of the earliest papers on the solution of linear systems with nonlinear parameter dependencies is [8], cf. [9]. Later works focus on the solution of systems of linear equations whose coefficient matrices enjoy a special structure, e.g. circulant [10], Toeplitz [12], symmetric, and skew-symmetric matrices [13].

A standard method for solving problems in structural mechanics, such as linear static problems, is the finite element method (FEM). The method leads to a system of algebraic equations, which in case of uncertain (interval) physical parameters becomes a linear system involving interval parameters. An overview of developments in techniques for the handling of uncertainty using the finite element method and applications in structural engineering mechanics can be found in [20]. Here, the authors combine an element-by-element (EBE) formulation, where the elements are kept disassembled, with a penalty method for imposing the necessary constrains for compatibility and equilibrium, in order to reduce the overestimation in the solution intervals. This approach should be applied simultaneously with FEM and affects the construction of the global stiffness matrix and the right-hand side vector, making them larger. A non-parametric fixed-point iteration is then used to solve the parametric interval linear system. While special construction methods are applied in [20], the parametric system obtained by standard FEM applied to a structural steel frame with partially constrained connections is solved by a sequence of interval-based (but not parametric) methods [5]. In [29], a parametric residual iteration [34], generalised in [23], is applied to bounding the response of structural engineering systems involving rational dependencies between the model parameters. Corresponding software tools with result verification, implemented in the *Mathematica* environment [39], have been developed. This general-purpose interval approach imposes no restrictions on how the parametric system is generated and can be applied to linear parametric problems for which special methods have not yet been designed.

The last method requires an enclosure of the range of nonlinear functions over the domain of the parameters. When the parameter dependencies are polynomial, tight bounds for the polynomial ranges can be obtained by the expansion of a multivariate polynomial into Bernstein polynomials [11, 38]. The goal of our work is to combine the generalised parametric residual iteration with range enclosure, based on the Bernstein expansion of polynomials, into a more efficient parametric linear system solver. For the sake of rapid development, run-time efficiency, and for exploiting the advantages of modern general-purpose software environments, such as *Mathematica*, our implementation is based on an advanced connectivity between *Mathematica* and an external C++ software via the *MathLink* communication protocol. The present parametric solver is illustrated by numerical solutions to three problems from structural mechanics which have been modelled by standard FEM and involve interval uncertainty in all material and load parameters. A discussion on the comparison between the present parametric solver, based on Bernstein polynomial ranges, and the former one is provided.

The paper is organised as follows. In Section 2, the parametric residual iteration method for linear interval systems is introduced, followed by an introduction to the Bernstein expansion and

the implicit Bernstein form. Section 3 discusses the software implementation of these methods and the interface between *Mathematica*, `C++`, and the interval software library `filib++`. In Section 4, the new parametric solvers and software tools are illustrated by three examples of one- and two-bay steel frames. Finally, some conclusions are given.

2 Methodology

2.1 The Iteration Method

Consider a linear system

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

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

$$a_{ij}(x) \;=\; a_{ij}(x_1, \ldots, x_n), \quad i, j = 1, \ldots, m, \tag{1b}$$

$$x \;\in\; [x] = ([x_1], \ldots, [x_n])^\top, \tag{1c}$$

and similarly for b.
The set of solutions to (1a–1c), called the *parametric solution set*, is

$$\Sigma = \Sigma\,(A(x), b(x), [x]) \;:=\; \{s \in \mathbb{R}^m \mid A(x) \cdot s = b(x) \text{ for some } x \in [x]\}\,. \tag{2}$$

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

We use the following notation: $\mathbb{R}^m, \mathbb{R}^{m\times n}$ denote the set of real vectors with m components and the set of real $m \times n$ matrices, respectively. A real compact interval is defined as $[a] = [\underline{a}, \overline{a}] := \{a \in \mathbb{R} \mid \underline{a} \le a \le \overline{a}\}$. By $\mathbb{IR}^m, \mathbb{IR}^{m\times n}$ we denote interval m-vectors and interval $m \times n$ matrices. Operations on interval values yield the smallest interval value containing the corresponding result when power set operations are used. We assume that the reader is familiar with the conventional interval arithmetic [1, 19].

In this section we consider a self-verified method for bounding the solution set of a parametric linear system. This 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. [34] and the literature cited therein). Historically, the basic idea of combining the Krawczyk-operator [14] and the existence test by Moore [18] is further elaborated by S. Rump [33] who proposes several improvements leading to inclusion theorems for the solution set of a nonparametric system of linear interval equations $[A] \cdot s = [b]$. In [34, Theorem 4.8] S. Rump gives a straightforward generalization to affine-linear dependencies in the matrix and the right hand side. With obvious modifications, the corresponding theorems can also be applied directly to linear systems involving nonlinear dependencies between the parameters in $A(x)$ and $b(x)$. This is demonstrated in [25, 29]. The following theorem is a general formulation of the enclosure method for linear systems involving arbitrary parametric dependencies.

Theorem 2.1. *Consider a parametric linear system defined by (1a–1c). Let* $R \in \mathbb{R}^{m\times m}$, $[y] \in \mathbb{IR}^m$, $\tilde{s} \in \mathbb{R}^m$ *be given and define* $[z] \in \mathbb{IR}^m$, $[C] \in \mathbb{IR}^{m\times m}$ *by*

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

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

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

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

The above theorem generalises [34, Theorem 4.8] by stipulating a sharp enclosure of $C(x) := I - R\cdot A(x)$ for $x \in [x]$, instead of using the interval extension $C([x])$, cf. [23]. A sharp enclosure of the iteration matrix $C(x)$ is also required by other authors (who do not refer to [34]) [6, 20], without addressing the issue of rounding errors. However, the generalization of [34, Theorem 4.8] is first proven in [22, 23]. Examples demonstrating the expanded scope of application of the generalized inclusion theorem can be found in [22, 23, 31].

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 is proven to be very useful. A detailed presentation of the computational algorithm can be found in [25, 33].

In case of arbitrary nonlinear dependencies between the uncertain parameters, computing $[z]$ and $[C]$ in Theorem 2.1 requires a sharp range enclosure of nonlinear functions. This is a key problem in interval analysis and there exists a huge number of methods and techniques devoted to this problem, with no one method being universal. In this work we restrict ourselves to linear systems where the elements of $A(x)$ and $b(x)$ are rational functions of the uncertain parameters. In this case the elements of $z(x)$ and $C(x)$ are also rational functions of x. The quality of the range enclosure of $z(x)$ will determine the sharpness of the parametric solution set enclosure. In [25] the above inclusion theorem is combined with a simple interval arithmetic technique providing inner and outer bounds for the range of monotone rational functions. The arithmetic of generalised (proper and improper) intervals is considered as an intermediate computational tool for eliminating the dependency problem in range computation and for obtaining inner estimations by outwardly rounded interval arithmetic. Since this methodology is not efficient in the general case of non-monotone rational functions, in this work we combine the parametric fixed-point iteration with range enclosing tools based on the Bernstein expansion of multivariate polynomials. Other approaches are presented in [21].

2.2 Bernstein Enclosure of Polynomial Ranges

In this section we recall some properties of the Bernstein expansion which are fundamental to our approach, cf. [4, 11, 38].

Firstly, some notation is introduced. We define multiindices $i = (i_1, \ldots, i_n)^T$ as vectors, where the n components are nonnegative integers. The vector 0 denotes the multiindex with all components equal to 0, which should not cause ambiguity. Comparisons are used entrywise. Also the arithmetic operators on multiindices are defined componentwise such that $i \odot l := (i_1 \odot l_1, \ldots, i_n \odot l_n)^T$, for $\odot = +, -, \times$, and $/$ (with $l > 0$). For instance, i/l, $0 \le i \le l$, defines the Greville abscissae. For $x \in \mathbb{R}^n$ its multipowers are

$$x^i := \prod_{\mu=1}^{n} x_\mu^{i_\mu}. \tag{3}$$

For the n-fold sum we use the notation

$$\sum_{i=0}^{l} := \sum_{i_1=0}^{l_1} \cdots \sum_{i_n=0}^{l_n} . \tag{4}$$

The generalised binomial coefficient is defined by

$$\binom{l}{i} := \prod_{\mu=1}^{n} \binom{l_\mu}{i_\mu}. \tag{5}$$

For reasons of familiarity, the Bernstein coefficients are denoted by b_i; this should not be confused with components of the right hand side vector b of (1a). Hereafter, a reference to the latter will be made explicit.

An n-variate polynomial p,

$$p(x) = \sum_{i=0}^{l} a_i x^i, \quad x = (x_1, \ldots, x_n), \tag{6}$$

can be represented over $U = [0,1]^n$ as

$$p(x) = \sum_{i=0}^{l} b_i B_i(x), \tag{7}$$

where B_i is the *i-th Bernstein polynomial of degree l*

$$B_i(x) = \binom{l}{i} x^i (1-x)^{l-i} \tag{8}$$

and the so-called *Bernstein coefficients* b_i are given by

$$b_i = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{l}{j}} a_j, \quad 0 \le i \le l. \tag{9}$$

Although the case of the unit box U may be considered without loss of generality, since any nonempty box in $\mathbb{R}^n$ can be mapped affinely thereupon, we need here the general case. The Bernstein coefficients b_i of degree $l = (l_1, \ldots, l_n)$ over a box

$$\begin{aligned} [x] &:= [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n], \\ \underline{x} &= (\underline{x}_1, \ldots, \underline{x}_n),\ \overline{x} = (\overline{x}_1, \ldots, \overline{x}_n), \end{aligned} \tag{10}$$

are given by

$$b_i = \sum_{j=0}^{i} \frac{\binom{i}{j}}{\binom{l}{j}} (\overline{x} - \underline{x})^j \sum_{\kappa=j}^{l} \binom{\kappa}{j} \underline{x}^{\kappa-j} a_\kappa, \quad 0 \le i \le l. \tag{11}$$

The essential property of the Bernstein expansion is the *range enclosing property*, namely that the range of p over $[x]$ is contained within the interval spanned by the minimum and maximum Bernstein coefficients:

$$\min_i \{b_i\} \le p(x) \le \max_i \{b_i\},\ x \in [x]. \tag{12}$$

It is also worth noting that the values attained by the polynomial at the vertices of $[x]$ are identical to the corresponding vertex Bernstein coefficients, for example $b_0 = p(\underline{x})$ and $b_l = p(\overline{x})$. The *sharpness property* states that the lower (resp. upper) bound provided by the minimum (resp. maximum) Bernstein coefficient is sharp, i.e. there is no underestimation (resp. overestimation), if and only if this coefficient occurs at a vertex of $[x]$.

The traditional approach (see, for example, [11, 38]) assumes that all of the Bernstein coefficients are computed, and their minimum and maximum is determined. By use of an algorithm (cf. [11, 38]) which is similar to de Casteljau's algorithm (see, for example, [32]), this computation can be made efficient, with time complexity $O(n\hat{l}^{n+1})$ and space complexity (equal to the number of Bernstein coefficients) $O((\hat{l}+1)^n)$, where $\hat{l} = \max_{i=1}^{n} l_i$. This exponential complexity is a drawback of the traditional approach, rendering it infeasible for polynomials with moderately many (typically, 10 or more) variables.

In [37] a new method for the representation and computation of the Bernstein coefficients is presented, which is especially well suited to sparse polynomials. With this method the computational complexity typically becomes nearly linear with respect to the number of the terms in the polynomial, instead of exponential with respect to the number of variables. This improvement is obtained from the results surveyed in the following subsections. For details and examples the reader is referred to [37].

2.2.1 Bernstein Coefficients of Monomials

Let $q(x) = x^r$, $x = (x_1, \ldots, x_n)$, for some $0 \leq r \leq l$. Then the Bernstein coefficients of q (of degree l) over $[x]$ (10) are given by

$$b_i = \prod_{m=1}^{n} b_{i_m}^{(m)}, \tag{13}$$

where $b_{i_m}^{(m)}$ is the i_mth Bernstein coefficient (of degree l_m) of the univariate monomial x^{r_m} over $[\underline{x}_m, \overline{x}_m]$. If the box $[x]$ is restricted to a single orthant of $\mathbb{R}^n$ then the Bernstein coefficients of q over $[x]$ are monotone with respect to each variable x_j, $j = 1, \ldots, n$.

With this property, for a single-orthant box, the minimum and maximum Bernstein coefficients must occur at a vertex of the array of Bernstein coefficients. This also implies that the bounds provided by these coefficients are sharp; see the aforementioned sharpness property. Finding the minimum and maximum Bernstein coefficients is therefore straightforward; it is not necessary to explicitly compute the whole set of Bernstein coefficients. Computing the component univariate Bernstein coefficients for a multivariate monomial has time complexity $O(n(\hat{l}+1)^2)$. Given the exponent r and the orthant in question, one can determine whether the monomial (and its Bernstein coefficients) is increasing or decreasing with respect to each coordinate direction, and one then merely needs to evaluate the monomial at these two vertices.

Without the single orthant assumption, monotonicity does not necessarily hold, and the problem of determining the minimum and maximum Bernstein coefficients is more complicated. For boxes which intersect two or more orthants of $\mathbb{R}^n$, the box can be bisected, and the Bernstein coefficients of each single-orthant sub-box can be computed separately.

2.2.2 The Implicit Bernstein Form

Firstly, we can observe that since the Bernstein form is linear, if a polynomial p consists of t terms, as follows,

$$p(x) = \sum_{j=1}^{t} a_{i_j} x^{i_j}, \quad 0 \leq i_j \leq l, \quad x = (x_1, \ldots, x_n), \tag{14}$$

then each Bernstein coefficient is equal to the sum of the corresponding Bernstein coefficients of each term, as follows:

$$b_i = \sum_{j=1}^{t} b_i^{(j)}, \quad 0 \leq i \leq l, \tag{15}$$

where $b_i^{(j)}$ are the Bernstein coefficients of the jth term of p. (Hereafter, a superscript in brackets specifies a particular term of the polynomial. The use of this notation to indicate a particular coordinate direction, as in the previous subsection, is no longer required.)

Therefore one may implicitly store the Bernstein coefficients of each term, and compute the Bernstein coefficients as a sum of t products, only as needed. The implicit Bernstein form thus consists of computing and storing the n sets of univariate Bernstein coefficients (one set for each component univariate monomial) for each of t terms. Computing this form has time complexity $\mathrm{O}(nt(\hat{l}+1)^2)$ and space complexity $\mathrm{O}(nt(\hat{l}+1))$, as opposed to $\mathrm{O}((\hat{l}+1)^n)$ for the explicit form. Computing a single Bernstein coefficient from the implicit form requires $(n+1)t-1$ arithmetic operations.

2.2.3 Determination of the Bernstein Enclosure for Polynomials

We consider the determination of the minimum Bernstein coefficient; the determination of the maximum Bernstein coefficient is analogous. For simplicity we assume that $[x]$ is restricted to a single orthant.

We wish to determine the value of the multiindex of the minimum Bernstein coefficient in each direction. In order to reduce the search space (among the $(\hat{l}+1)^n$ Bernstein coefficients) we can exploit the monotonicity of the Bernstein coefficients of monomials and employ uniqueness, monotonicity, and dominance tests, cf. [37] for details. As the examples in [37] show, it is often possible in practice to dramatically reduce the number of Bernstein coefficients that have to be computed.

3 Software Tools

Publically-available software for the solution of parametric interval linear systems, for the *Mathematica* [24, 25] and C-XSC [31] environments has been developed. The *Mathematica* package `IntervalComputations` `` `LinearSystems` `` contains a variety of functions for computing guaranteed inclusions for the solution set of an interval linear system [24]. The particular solvers differ with respect to the type of the linear system to be solved and the implemented solution method. Recently, these parametric linear solvers were upgraded to handle linear systems involving arbitrary rational dependencies [25, 29]. The enclosures of $z(p)$ and $C(p)$ from Theorem 2.1 were computed by a technique based on generalised intervals, which provides sharp range enclosures for monotone rational functions. The goal of this work is to further upgrade the parametric solvers for systems involving polynomial and/or arbitrary rational dependencies, by integrating

more powerful and efficient tools for range computation into the corresponding *Mathematica* functions.

3.1 filib++ Software for Polynomial Ranges

Given a polynomial p (6) and a box $[x]$ (10), we wish to compute a guaranteed tight enclosure for $p([x])$. The existing C++ software routines of the last author, which implement the aforementioned implicit Bernstein form, are utilised. Interval arithmetic is used extensively throughout, for which the C++ interval library `filib++` [16, 17] is employed.

Polynomials are passed to the program in a sparse representation, consisting of a one-dimensional array of non-zero terms. The ordering of the terms in the polynomial is unimportant. Each term thus consists of a coefficient with an array of associated variable exponents. Coefficients are stored as intervals; they are passed as point values and converted to intervals of machine-precision width. Implicit in this data structure are n, the number of variables, and l, the vector comprising the degrees in each variable. The principal data construct created by the program, designed as a "workspace" for applications, is an aggregate structure comprising a polynomial, a box, and the associated Bernstein coefficients. Firstly, given a polynomial and a box as input, the workspace is initialised and the corresponding Bernstein coefficients in implicit form are computed and stored. The range computation routine then takes such a workspace as input and returns a tight outer estimation for the range of the polynomial over the box. This range is equal to the Bernstein enclosure, i.e. the range spanned by the minimum and maximum Bernstein coefficients. In many cases (often for small boxes and/or where the polynomial is monotonic over the box), the range is provided without overestimation (except for the outward rounding which is inherent in the interval arithmetic). Due to the use of interval arithmetic, this range is a guaranteed outer estimation. A routine for box subdivision for further tightening of the bounds exists, but is not required for the present application.

3.2 New Parametric Solvers

Parametric linear systems involving linear dependencies and systems with particular fixed data dependencies allow for an entirely numerical data representation and for an efficient algorithm implementation. Linear systems involving nonlinear parameter dependencies can be declared and processed more easily and the solution-enclosing methods can be more readily investigated in a symbolic programming environment. For example, by means of suitable algebraic manipulations the expression of a general rational function can be transformed into the form of a quotient of two polynomials. On the other hand, the implementation of the polynomial range enclosing tools, presented in Sections 2.2 and 3.1, is based on the C++ interval library `filib++`, for efficiency reasons. Since it is a high-quality specialized software exhibiting good performance, there is no reason for its re-implementation in *Mathematica*. In order to shorten the development time and to preserve the beneficial properties of both implementation environments, the authors have connected the generalized parametric fixed-point iteration and the Bernstein enclosure of polynomial ranges into a new parametric solver via the *MathLink* communication protocol.

MathLink [39] allows the external `filib++` function for polynomial range computation to be called from within *Mathematica* as required. Following the *MathLink* technology [28, 39], the following steps constitute the development process:

1. Create a template file whose main purpose is to establish a correspondence between the function for range computation that will be called from *Mathematica* and the external C++ function that will call the `filib++` range computation function. In the developed template

file we have included as pre-evaluated expressions the source of the whole *Mathematica* package involving the desired parametric solvers. Thus, when installing the *MathLink* connection in Step 4 below, all the parametric solvers involved in the package are ready for use.

2. A corresponding communication module should be written in C++ to combine the template file with the external `filib++` software. A C++ function, specified in the template file, reads the *Mathematica* generated data that numerically define a multivariate polynomial and a box, initializes new variables whose data types are specific for the `filib++` range computation function and after the actual computations (calling the `filib++` range computation function) transforms the computed result into variables of fundamental C++ data types that are passed back to *Mathematica*. The communication module also contains a function for communicating error messages, and a standard main function.

3. Process the *MathLink* template information and compile all the source code.

4. Install the binary in the current *Mathematica* session.

More details about *MathLink* technology and the connectivity between *Mathematica* and external `filib++` based interval programs can be found in [28].

Below we briefly outline the functionality of the newly developed *Mathematica* package `ParametricPolySolvers` based on *MathLink* connection to the external `filib++` software for the enclosure of a polynomial range. The usage of the package is illustrated in Section 4.

The solver `polyParametricSolve[Ax, bx, parLst]` computes guaranteed outer bounds for the solution set of a parametric linear system, where the matrix `Ax` and/or the right hand side vector `bx` involve polynomial dependencies between uncertain parameters. The latter and their interval values are specified by a list of transformation rules[1] `parLst`. The general parametric residual iteration, cf. Theorem 2.1, implemented in this function, uses some algebraic manipulations and *MathLink* communication with the `filib++` software for bounding the ranges of multivariate polynomials involved in the computation of $z(x)$ and $C(x)$. The solver `polyParametricSolve` can take a fourth optional argument `Refinement -> True` which determines whether an iterative refinement procedure is applied to the computed outer solution enclosure. The default setting is `False`.

A specific feature of the `polyParametricSolve` function is that the function input arguments `Ax`, `bx` can be represented as

$$Ax = A(x) + [A], \qquad bx = b(x) + [b], \tag{16}$$

where the elements of $A(x), b(x)$ are multivariate polynomials of the parameters $x_1, \ldots, x_n$, while $[A] \in \mathbb{IR}^{n\times n}$, $[b] \in \mathbb{IR}^n$. This way, `polyParametricSolve` could be used for solving parametric linear systems involving rational dependencies as well as intervals which represent bounds on the remainder terms of the Taylor expansion applied to non-rational dependencies.

The `filib++` software described in Section 3.1 for bounding the range of a multivariate polynomial over a box can also be used for bounding the range of a function involving arbitrary rational dependencies between its variables, if we represent the rational function as a quotient of two multivariate polynomials which are to be bounded separately. This motivates the development of another function `polyRationalSolve[Ax, bx, parLst]` applicable to linear systems involving arbitrary rational parameter dependencies. The solver `polyRationalSolve` has the

[1] *Mathematica* transformation rules have the form `name -> value`.

same optional argument as `polyParametricSolve`. It can also be applied to systems with polynomial dependency but without function input arguments of the form (16).

For the sake of comparison, the package contains the function `ParametricSolve[Ax, bx, parLst]` with the same usage as the previous solvers but applying the former range computation method based on generalised interval arithmetic.

3.3 Accessibility

The new software tools described above can be obtained from the authors. The library `filib++` is free software [17]. This library and *Mathematica* should be installed on the user machine. Then the developed template file and the communication module `ParametricPolySolvers.tm/cpp` should be compiled together with the range computation `filib++` software following the *MathLink* technology. The end-users, who do not have *Mathematica* or do not want to establish a communication with external programs, can run *Mathematica* and the parametric solvers remotely via the webComputing service framework [27].

4 Numerical Examples

In this section we illustrate the usage of the new parametric solvers based on bounding polynomial ranges by Bernstein expansion. The improved efficiency of the new polynomial solvers is demonstrated by comparing both the computing time and the quality of the solution enclosure for the new solvers and the former one. The examples were run on a PC with AMD Athlon-64 3GHz processor. Below we present the *Mathematica* commands and the corresponding output results in a session. Once the external `filib++` program and the developed *MathLink* compatible files have been processed and compiled to an executable file, called `ParametricPolySolvers`, the latter can be installed in a *Mathematica* session.

```
In[1]:= lnk = Install["ParametricPolySolvers"]
Out[1]= LinkObject[./ParametricPolySolvers, 3, 2]
```

The `Install` function opens a link through which the external range computing functions can be called. The program also makes all definitions and the *Mathematica* code of the parametric solves described in Section 3.2 above visible for the current *Mathematica* session.

```
In[2]:= Names["ParametricPolySolvers`*"]
Out[2]= {ParametricSolve, polyParametricSolve, polyRationalSolve, Refinement}
```

4.1 One-Bay Steel Frame

Consider a simple one-bay structural steel frame, as shown in Figure 1, which was initially proposed and analyzed in [5]. Following standard practice, the authors have assembled a parametric linear system of order eight and involving eight uncertain parameters. The typical nominal parameter values and the corresponding worst case uncertainties, as proposed in [5], are shown in Table 1. The explicit analytic form of the given system involving polynomial parameter dependencies can be found in [5, 29].

As in [5, 29], we solved the system with parameter uncertainties which are 1% of the values presented in the last column of Table 1. Let us assume that all input data for the given parametric system are stored in the *Mathematica* variables `Ax, bx, tr`. We call the parametric solver `polyParametricSolve` and measure the absolute time for execution of the main steps.

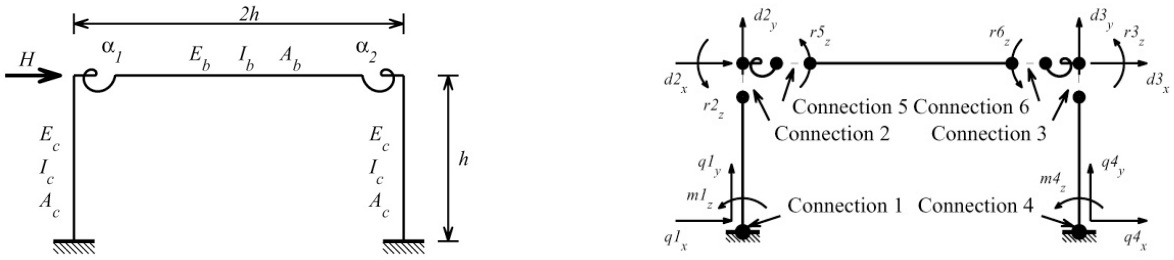

Figure 1: One-bay structural steel frame [5].

Table 1: Parameters involved in the steel frame example, their nominal values, and worst case uncertainties.

parameter		nominal value	uncertainty
Young modulus	E_b	$29 * 10^6$ lbs/in^2	$\pm 348 * 10^4$
	E_c	$29 * 10^6$ lbs/in^2	$\pm 348 * 10^4$
Second moment	I_b	510 in^4	± 51
	I_c	272 in^4	± 27.2
Area	A_b	10.3 in^2	± 1.03
	A_c	14.4 in^2	± 1.44
External force	H	5305.5 lbs	± 2203.5
Joint stiffness	α	$2.77461 * 10^9$ lb-in/rad	$\pm 1.26504 * 10^9$
Length	L_c	144 in, L_b 288 in	

```
In[7]:=AbsoluteTiming[polyRes=polyParametricSolve[Ax,bx,tr,Refinement->True];]
  absolute time for C = 0.024074 Second
  absolute time for z = 0.007313 Second
  {0.045983 Second, Null}
```

Now, running the previous parametric solver we get the following time measurements

```
In[9]:= AbsoluteTiming[paramRes=ParametricSolve[Ax,bx,tr,Refinement->True];]
   absolute time for C = 0.162854 Second
   absolute time for z = 0.161104 Second
Out[9] = {0.342757 Second, Null}
```

showing that the compiled external code for range computation was considerably faster than the interpretative *Mathematica* code. For this example, the quality of the solution set enclosures, provided by both solvers, was comparable. As shown in [25, 29], the solution enclosure obtained by the parametric solver is by more than one order of magnitude better than the solution enclosure obtained in [5].

Based on the runtime efficiency of the new parametric solver, we next attempt to solve the same parametric linear system for the worst case parameter uncertainties in Table 1 ranging between about 10% and 45.6%. Firstly, we notice that the parametric solution depends linearly on the parameter H, so that we can obtain a better solution enclosure if we solve two parametric systems with the corresponding end-points for H. Secondly, enclosures of the hull of the solution set are obtained by subdivision of the worst case parameter intervals $(E_b, E_c, I_b, I_c, A_b, A_c, \alpha)^\top$

into $(2,2,2,2,1,1,6)^\top$ subintervals of equal width, respectively. We use more subdivision with respect to α since α is subject to the greatest uncertainty. The solution enclosure, obtained within 11 sec., is given in Table 2.

Table 2: One-bay steel frame example with worst-case parameter uncertainties (Table 1). Solution enclosure found by dividing the parameter intervals $(E_b, E_c, I_b, I_c, A_b, A_c, \alpha)^\top$ into $(2,2,2,2,1,1,6)^\top$ subintervals of equal width, respectively. All numerical quantities are multiplied by 10^5.

$d2_x$:	[5454.706610, 24708.395582]	$r6_z$:	[-105.9680866, -17.64526946]
$d2_y$:	[11.5445278769, 84.761351898]	$d3_x$:	[5325.027833, 24285.634925]
$r2_z$:	[-129.02427835, -22.381136355]	$d3_y$:	[-148.1290977, -16.38968649]
$r5_z$:	[-113.21398401, -17.95789860]	$r3_z$:	[-122.3361772, -21.69878778]

The quality of the solution set enclosure is presented in Table 3, where $\mathcal{O}_\omega$ is defined by

$$\mathcal{O}_w([a],[b]) := 100(1-\omega([a])/\omega([b])), \text{ for } [a] \subseteq [b]$$

and ω is the width of the interval. The *Mathematica* function `Overestimation[a, b]`, used below, implements $\mathcal{O}_w([a],[b])$. The combinatorial solution, obtained as the convex hull of the solutions to point linear systems where the parameters take all possible combinations of the interval end-points, is used as an inner estimation of the solution set hull.

Table 3: One-bay steel frame example with worst-case parameter uncertainties (Table 1). Solution enclosure found by dividing the parameter intervals $(E_b, E_c, I_b, I_c, A_b, A_c, \alpha)^\top$ into $(2,2,2,2,1,1,6)^\top$ subintervals of equal width, respectively. The obtained enclosure $[u]$ of the solution set hull is compared to the combinatorial solution $[\tilde{h}]$.

	$d2_x$	$d2_y$	$r2_z$	$r5_z$	$r6_z$	$d3_x$	$d3_y$	$r3_z$
$\mathcal{O}_\omega([\tilde{h}],[u])$	12.5	8.0	23.7	25.6	25.0	12.7	13.2	23.5

These results show that by means of a minimal number of subdivisions the new parametric solver provides a good solution enclosure very quickly for the difficult problem of worst-case parameter uncertainties. Note that sharper bounds, close to the exact hull, can be obtained by proving the monotonicity properties of the parametric solution [26].

4.2 Two-Bay Two-Story Frame Model with 13 Parameters

Consider a two-bay two-story steel frame with IPE 400 beams and HE 280 B columns, as shown in Figure 2, after [29]. The frame is subjected to lateral static forces and vertical uniform loads. Beam-to-column connections are considered to be semi-rigid and they are modelled by single rotational spring elements. Applying conventional methods for the analysis of frame structures, a system of 18 linear equations is obtained, where the elements of the stiffness matrix and of the right hand side vector are rational functions of the model parameters. We consider the parametric system resulting from a finite element model involving the following 13 uncertain parameters: A_c, I_c, E_c, A_b, I_b, E_b, c, $w_1, \ldots, w_4$, f_1, f_2. Their nominal values, taken according to

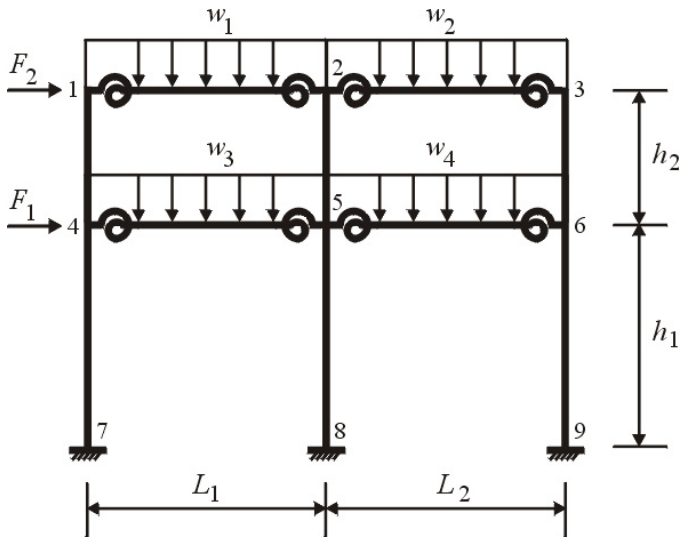

Figure 2: Two-bay two-story steel frame [29].

the European Standard Eurocode3 [7], are given in Table 4. The explicit analytic form of the given parametric system can be found in [30].

The parametric system is solved for the element material properties $(A_c, \ldots E_b)$, which are taken to vary within a tolerance of 1% (that is $[x-x/200,\ x+x/200]$, where x is the corresponding parameter nominal value from Table 4) while the spring stiffness and all applied loadings are taken to vary within 10% tolerance intervals.

parameter	Columns (HE 280 B)	Beams (IPE 400)
Cross-sectional area	$A_c = 0.01314$ m^2	$A_b = 0.008446$m^2
Moment of inertia	$I_c = 19270 * 10^{-8}$ m^4	$I_b = 23130 * 10^{-8}$ m^4
Modulus of elasticity	$E_c = 2.1 * 10^8$kN/m^2	$E_b = 2.1 * 10^8$ kN/m^2
Length	$L_c = 3$ m	$L_b = 2L_c$ m
Rotational spring stiffness	$c = 10^8$ kN	
Uniform vertical load	$w_1 = \ldots = w_4 = 30$ kN/m	
Concentrated lateral forces	$f_1 = f_2 = 100$ kN	

Table 4: Parameters involved in the two-bay two-story frame example with their nominal values.

It is assumed that all input data for the given parametric system are contained in the *Mathematica* variables `Ax, bx, tr`. The parametric solver `polyRationalSolve` is called and the absolute time for execution of the main steps is measured. Then the interval enclosure of the solution set is obtained.

```
In[7]:= AbsoluteTiming[polyRes= polyRationalSolve[Ax,bx,tr,Refinement->True];]
   absolute time for C = 0.585924 Second
   absolute time for z = 0.693990 Second
Out[7] = {1.318729 Second, Null}
```

Now, the former parametric solver is run.

```
In[9]:= AbsoluteTiming[paramRes= ParametricSolve[Ax,bx,tr,Refinement->True];]
   absolute time for C = 3.144183 Second
   absolute time for z = 4.142891 Second
Out[9] = {7.355345 Second, Null}
```

The new parametric solver is about six times faster than the previous one. Comparing the quality of the solution enclosures we obtain the following result.

```
In[10]:= MapThread[Overestimation, {polyRes, paramRes}]
Out[10] = {64.38, 91.79, 66.15, 64.89, 87.45, 61.94, 64.8, 92.61,
59.34, 53.46, 92.1, 57.43, 54.66, 88.1, 61.02, 55.08, 92.92, 56.61}
```

An algebraic simplification, applied to functional expressions in computer algebra environments, may reduce the occurrence of interval variables which could result in a sharper range enclosure. Such an algebraic simplification is expensive and when applied to complicated rational expressions usually does not result in a sharper range enclosure. For the sake of comparison, we have run the former parametric solver in two ways: applying intermediate simplification during the range computation, and without any algebraic simplification. The above results were obtained when the range computation in `ParametricSolve` does not use any algebraic simplification. When the range computation of the previous solver uses intermediate algebraic simplification, we obtain the following results.

```
In[11]:= AbsoluteTiming[paramRes13=ParametricSolve[Ax,bx,tr,Refinement->True];]
   absolute time for C = 5.785307 Second
   absolute time for z = 8.588179 Second
Out[11] = {14.402811 Second, Null}

In[12]:= MapThread[Overestimation, {paramRes13, polyRes}]
Out[12]= {18.95, 27.09, 37.06, 18.91, 27.09, 36.97, 18.97, 27.08, 37.07,
18.66, 27.10, 37.02, 18.62, 27.099, 36.97, 18.68, 27.09, 37.05}
```

In this case `ParametricSolve` was much slower but provided a tighter enclosure of the solution set than the rational solver, based on polynomial ranges, which did not account for all the parameter dependencies.

4.3 Two-Bay Two-Story Frame Model with 37 Parameters

As a larger problem of a parametric system involving rational parameter dependencies, we consider the finite element model of the two-bay two-story steel frame from Section 4.2, where each structural element has properties varying independently within 1% tolerance intervals. This does not change the order of the system but it now depends on 37 interval parameters. The explicit analytic form of the given parametric system can be found in [30]. Here the right hand side vector is given to illustrate the dependencies.

$$\Bigg(f_2,\ -\frac{1}{2}w_1Lb_1,\ -\frac{w_1Lb_1^2}{12(1+\frac{2Eb_1Ib_1}{cLb_1})},\ 0,\ -\frac{w_1Lb_1}{2}-\frac{w_2Lb_2}{2},\ \frac{w_1Lb_1^2}{12(1+\frac{2Eb_1Ib_1}{cLb_1})}-\frac{w_2Lb_2^2}{12(1+\frac{2Eb_2Ib_2}{cLb_2})},$$
$$0,\ -\frac{w_2Lb_2}{2},\ \frac{w_2Lb_2^2}{12(1+\frac{2Eb_2Ib_2}{cLb_2})},\ f_1,\ -\frac{1}{2w_3Lb_3},\ -\frac{w_3Lb_3^2}{12(1+\frac{2Eb_3Ib_3}{cLb_3})},$$
$$0,\ -\frac{w_3Lb_3}{2}-\frac{w_4Lb_4}{2},\ \frac{w_3Lb_3^2}{12(1+\frac{2Eb_3Ib_3}{cLb_3})}-\frac{w_4Lb_4^2}{12(1+\frac{2Eb_4Ib_4}{cLb_4})},\ 0,\ -\frac{w_4Lb_4}{2},\ \frac{w_4Lb_4^2}{12(1+\frac{2Eb_4Ib_4}{cLb_4})}\Bigg)^\top.$$

First, the polynomial solver is run, and one observes a considerable increase in the computing time compared to the time needed for the 13 parameters example, caused by the larger number of parameters.

```
In[15]:=AbsoluteTiming[polyRes = polyRationalSolve[Ax,bx,tr,Refinement->True];]
   absolute time for C = 1.446245 Second
   absolute time for z = 243.311070 Second
Out[15] = {244.789783 Second, Null}
```

The former parametric solver, based on range computation without algebraic simplification, exhibits approximately three times slower performance than the new one.

```
In[17]:= AbsoluteTiming[paramRes=ParametricSolve[Ax,bx,tr,Refinement->True];]
   absolute time for C = 11.506395 Second
   absolute time for z = 743.446813 Second
Out[17] = {754.986023 Second, Null}
```

The quality of the solution enclosure, provided by the new polynomial solver, is also much better than the solution enclosure provided by the former solver.

```
In[18]:= MapThread[Overestimation, {polyRes, paramRes}]
Out[18] = {35.54, 64.19, 48.02, 35.96, 77.74, 39.02, 36.01, 66.67,
35.51, 28.4, 64.85, 39.6, 29.32, 78.45, 40.03, 30.14, 66.96, 95.46}
```

Note that when the previous range computation uses algebraic simplification, `ParametricSolve` is much slower. However, the quality of the solution enclosure does not improve by more than $5.44 * 10^{-13}$, probably due to the more complicated parameter dependencies. This demonstrates the merit of the general-purpose parametric iteration, combined with Bernstein enclosure of polynomial ranges, for solving parametric systems involving complicated dependencies between many parameters.

5 Conclusions

In this paper, we demonstrated the advanced application of a general-purpose parametric method, combined with the Bernstein enclosure of polynomial ranges, to linear systems obtained by standard FEM analysis of mechanical structures, and illustrated the efficiency of the new parametric solver.

New software tools for the enclosure of the solution set of a system of linear equations with polynomial or rational parameter dependencies are described. It is demonstrated that powerful techniques for range enclosure are necessary to provide tight bounds on the solution set, in particular when the parameters of the system are subject to large uncertainties and the dependencies are complicated.

The new self-verified parametric solvers can be incorporated into a general framework for the computer-assisted proof of global and local monotonicity properties of the parametric solution. Based on these properties, a guaranteed and highly accurate enclosure of the interval hull of the solution set can be computed [26]. The parametric solvers for square systems facilitate the guaranteed enclosures of the solution sets to over- and underdetermined parametric linear systems.

Being the only general-purpose parametric linear solver, the presented methodology and software tools are applicable in the context of any problem (stemming, e.g., from fuzzy set theory

[36], control engineering [35], robust Monte Carlo simulation [15], or others) that requires the solution of linear systems whose input data depend on uncertain (interval) parameters.

References

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

[2] Ackermann, L., A. Bartlett, D. Kesbauer, W. Sienel, and R. Steinhauser, *Robust Control: Systems with Uncertain Physical Parameters*, Springer-Verlag, Berlin, 1994.

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

[4] Cargo G. T. and Shisha O. (1966), "The Bernstein form of a polynomial," *J. Res. Nat. Bur. Standards Vol. 70B*, 79–81.

[5] Corliss, G., Foley, C. and Kearfott. R. B., "Formulation for Reliable Analysis of Structural Frames", in R. L. Muhanna and R. L. Mullen, editors, *Proceedings of NSF workshop on Reliable Engineering Computing*, Savannah, Georgia, September 2004, USA.

[6] Dessombz O. *et al.*, Analysis of Mechanical Systems Using Interval Computations Applied to Finite Element Methods, *Journal of Sound and Vibration* 239(5):949–968, 2001.

[7] European Standard. Eurocode 3: Design of Steel Structures. European Committee for Standardization, Ref.No. prEN 1993-1-1:2003 E, Brussels, 2003.

[8] Franzen, R., Die intervallanalytische Behandlung parameterabhängiger Gleichungssysteme, Berichte der GMD Vol. 47, Bonn, 1971.

[9] Franzen, R., Die Konstruktion eines Approximationspolynoms für die Lösungen parameterabhängiger Gleichungssysteme, *Z. Angew. Math. Mech.* 52, T202–T204, 1972.

[10] Garloff J., Zur intervallmäßigen Durchführung der schnellen Fourier-Transformation, *Z. Angew. Math. Mech.* 60, T291–T292, 1980.

[11] Garloff J. (1986), "Convergent bounds for the range of multivariate polynomials," *Interval Mathematics 1985,* K. Nickel, editor, *Lecture Notes in Computer Science Vol. 212*, Springer, Berlin, 37–56.

[12] Garloff J., Solution of linear equations having a Toeplitz interval matrix as coefficient matrix, *Opuscula Math.* 2, 33–45, 1986.

[13] Jansson, C., Interval linear systems with symmetric matrices, skew-symmetric matrices and dependencies in the right hand side, *Computing* 46, 265–274, 1991.

[14] Krawczyk, R.: Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken, *Computing* **4** (1969), 187–201.

[15] C. M. Lagoa, B. R. Barmish, Distributionally Robust Monte Carlo Simulation: A Tutorial Survey, in: Proceedings of the 15th IFAC World Congress, 2002, pp. 1327–1338. Available under http://www.ece.lsu.edu/mcu/lawss/add_materials/BRossBarmishTutorial.pdf

[16] Lerch M., Tischler G. and Wolff von Gudenberg J. (2001), "filib++ - Interval library specification and reference manual," Technical Report 279, University of Würzburg.

[17] Lerch, M., Tischler, G., Wolff von Gudenberg, J., Hofschuster, W; Krämer, W.: `filib++`, a Fast Interval Library Supporting Containment Computations, *ACM TOMS* 32(2):299–324, 2006. Library download: http://www.math.uni-wuppertal.de/org/WRST/software/filib.html

[18] Moore, R. E.: A Test for Existence of Solutions to Nonlinear Systems, *SIAM J. Numer. Anal.* **14** (1977), 611–615.

[19] Moore, R. E.: *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[20] Muhanna, R. L., Mullen, R.L., Zhang, H.: Penalty-Based Solution for the Interval Finite-Element Methods. *J. Eng. Mech.* 131(10):1102–1111, 2005.

[21] Neumaier, A.: *Improving Interval Enclosures*, manuscript, 2008. http://www.mat.univie.ac.at/~neum/papers.html♯encl

[22] Popova, E. D.: Strong Regularity of Parametric Interval Matrices. In Mathematics and Education in Mathematics, Proc. of the 33rd Spring Conference of the Union of Bulgarian Mathematicians, Sofia, 2004, 446–451. (http://www.math.bas.bg/~epopova/papers/04smbEP.pdf)

[23] Popova, E. D.: Generalizing the Parametric Fixed-Point Iteration, *Proceedings in Applied Mathematics & Mechanics* (PAMM) 4(1):680–681, 2004.

[24] Popova, E. D.: Parametric Interval Linear Solver, *Numerical Algorithms* 37(1–4):345–356, 2004.

[25] Popova, E. D.: Solving Linear Systems whose Input Data are Rational Functions of Interval Parameters. In T. Boyanov et al. (Eds): NMA 2006, Springer LNCS 4310, 2007, 345–352. Expanded version in Preprint 3/2005, Institute of Mathematics and Informatics, BAS, Sofia, 2005. (http://www.math.bas.bg/~epopova/papers/05PreprintEP.pdf)

[26] 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), IEEE Computer Society Press, July 2007, Library of Congress Number 2007929345, p. 35.

[27] Popova, E.: WebComputing Service Framework. *Int. Journal Information Theories and Applications* 13(3):246–254, 2006. Accessible at http://cose.math.bas.bg/webComputing

[28] E. Popova: Mathematica Connectivity to Interval Libraries filib++ and C-XSC. To appear in: A. Cuyt, W. Kramer, W. Luther, P. Markstein (Eds.), Numerical validation in current hardware architectures — From embedded system to high-end computational grids, Springer LNCS.

[29] Popova, E., R. Iankov, Z. Bonev: Bounding the Response of Mechanical Structures with Uncertainties in All the Parameters. In R.Muhannah, R.Mullen (Eds): Proceedings of the NSF Workshop on Reliable Engineering Computing, Svannah, 2006, 245–265.

[30] Popova, E., R. Iankov, Z. Bonev: FEM Model of a Two-Bay Two-Story Steel Frame — 2 Benchmark Examples. (http://www.math.bas.bg/~epopova/papers/2bay2storyProblems.pdf)

[31] Popova, E., and W. Krämer: Inner and Outer Bounds for the Solution Set of Parametric Linear Systems. *J. of Computational and Applied Mathematics* 199(2):310–316, 2007.

[32] Prautzsch H., Boehm W. and Paluszny M. (2002), "Bezier and B-Spline Techniques," Springer, Berlin, Heidelberg.

[33] Rump, S.: New Results on Verified Inclusions. In Miranker, W. L., R. Toupin (Eds): *Accurate Scientific Computations*. Springer LNCS **235** (1986), 31–69.

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

[35] Savov S., I. Popchev, Generalized Lyapunov Function for Stability Analysis of Uncertain Systems, in Proceedings of 16th IFAC World Congress, Prague, Czech Republic, July 4-8, 2005. http://www.nt.ntnu.no/users/skoge/prost/proceedings/ifac2005/Fullpapers/02330.pdf

[36] I. Skalna, Parametric Fuzzy Linear Systems, in: O. Castillo et al. (Eds.): Theoretical Advances and Applications of Fuzzy Logic and Soft Computing, Advances in Soft Computing vol. 42, Springer Berlin, Heidelberg, pp. 556–564, 2007.

[37] Smith A. P., "Fast construction of constant bound functions for sparse polynomials", to appear in J. Global Optimization.

[38] Zettler M. and Garloff J. (1998), "Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion," *IEEE Trans. Automat. Contr. Vol. 43*, 425–431.

[39] Wolfram Research Inc.: *Mathematica*, Version 5.2, Champaign, IL, 2005.