Robustness Analysis of Polynomials with Polynomial Parameter Dependency Using Bernstein Expansion

M. Zettler and J. Garloff†

Fachhochschule Konstanz
Fachbereich Informatik
Postfach 10 05 43
D-78405 Konstanz, Germany

1 Introduction

Stability of a polynomial plays an important part in the analysis and design of control systems. A standard approach to robustness analysis of linear dynamic systems is to examine the characteristic polynomial in the presence of parametric uncertainties. So far, the main attention has been paid to the case of affine and multiaffine parameter dependency of the coefficients of the characteristic polynomial, see, e.g., [2, 6, 20] and the references therein. However, these cases do not cover most real-life problems. Therefore, we are concerned here with the far more general case of *polynomial* dependency:

The Robust Stability Problem Let the parameter set Q be an l-dimensional box, i.e.

$$Q = [\underline{q}_1, \overline{q}_1] \times \cdots \times [\underline{q}_l, \overline{q}_l].$$

Let a family of polynomials be given by

$$p(s, \mathbf{q}) = a_0(\mathbf{q})s^m + \ldots + a_{m-1}(\mathbf{q})s + a_m(\mathbf{q}), \qquad (1)$$

where the coefficients are depending polynomially on parameters q_i, $i = 1, \ldots, l$, $\mathbf{q} = (q_1, \ldots, q_l)$, i.e. for $k = 0, \ldots, m$

$$a_k(\mathbf{q}) = \sum_{i_1,\ldots,i_l=0}^{d} a^{(k)}_{i_1\ldots i_l} q_1^{i_1} \ldots q_l^{i_l}. \qquad (2)$$

Question: *Is the family of polynomials (robustly) stable for Q, i.e. are the polynomials $p(\mathbf{q})$ stable for all $\mathbf{q} \in Q$?*

Here stability is meant in the sense of Hurwitz or asymptotical stability, i.e. we want to show that $p(s, \mathbf{q}) \neq 0$ for all $s \in \mathbf{C}$ with $Re\, s \geq 0$, $\mathbf{q} \in Q$. To avoid dropping in degree we assume for simplicity throughout this paper that $a_0(\mathbf{q}) > 0$ for all $\mathbf{q} \in Q$.

Unfortunately, most of the methods known from literature, e.g. [4, 5, 14, 15, 16, 18, 23, 32, 34, 35, 36], can only treat problems with polynomial dependency with only a few parameters and/or polynomials of lower degree. The genetic algorithm [25] appears to be an exception. However, this algorithm seems not be fully tested for large control problems and gives no guarantee for finding the global solution. In Ex. 4 in Ch. 5 we present an example in which this algorithm fails to give the correct solution.

A possible approach is to consider the Hurwitz determinant associated with the family of polynomials, e.g. [14, 16, 18, 23, 34, 35]. In principle, by space and time limitations this approach is restricted to problems with a moderate number of parameters and to lower degree polynomials.

The first algorithm we present in this paper adopts this approach and is based on the expansion of the Hurwitz determinant into Bernstein polynomials. This leads to a fast algorithm. Focusing on larger control problems we develop then a second algorithm which avoids the blowing up of the problem caused by using the Hurwitz determinant. The underlying idea of the algorithm is to watch for zero crossing over the imaginary axis by inspecting the so-called value set. Here we profit again from the convex hull property of the Bernstein expansion.

The organization of the paper is as follows: In Ch. 2 we first introduce the Bernstein expansion of a polynomial and explain then the sweep procedure which is fundamental for subsequent developments of both algorithms. Our first algorithm is presented in Ch. 3, where also the selection rules are introduced which have led to a considerable speeding up of the algorithm. The second algorithm which is designed for larger control problems is introduced in Ch. 4. To demonstrate the efficiency of both algorithms we present in Ch. 5 numerical results to real-world problems like that of the Fiat Dedra engine studied in [6]. Brief conclusions and directions for future research are given in Ch. 6.

The results of this paper are presented in greater detail in the report [37] which is available upon request. We note that the approach the first algorithm is based on can be applied to other stability regions as well as to matrix stability using the determinantal criteria listed in [30], e.g. Ch. 17 in [6], however often at the expense of an increase of dimensionality. For the related problem of Schur stability and the problem of computing the stability margin see [28].

†Author to whom all correspondence should be addressed.

2 Bernstein Expansion

For compactness, we define a *multi-index* I as an ordered l-tupel of nonnegative integers $(i_1, \ldots, i_l)$. We will use multi-indices e.g. to shorten power products: For $\mathbf{x} = (x_1, \ldots, x_l) \in \mathbf{R}^l$ we set $\mathbf{x}^I = x_1^{i_1} x_2^{i_2} \cdot \ldots \cdot x_l^{i_l}$. For simplicity, we suppress sometimes the brackets in the notation of multi-indices.

We write $I \leq N$ if $N = (n_1, \ldots, n_l)$ and if $0 \leq i_k \leq n_k$, $k = 1, \ldots, l$. Further, let $S = \{I : I \leq N\}$. Then we can write an l-variate polynomial p in the form

$$p(\mathbf{x}) = \sum_{I \in S} a_I \mathbf{x}^I, \quad \mathbf{x} \in \mathbf{R}^l, \tag{3}$$

and refer to N as the *degree* of p. We define the *total degree* of polynomial (3) as

$$\hat{n} = \max\{n_i : i = 1, \ldots, l\}. \tag{4}$$

Also, we write I/N for $(i_1/n_1, \ldots, i_l/n_l)$ and $\binom{N}{I}$ for $\binom{n_1}{i_1} \cdot \ldots \cdot \binom{n_l}{i_l}$.

2.1 Bernstein Transformation of a Polynomial

In this section we expand a given multivariate polynomial (3) into Bernstein polynomials to obtain bounds for its range over an l-dimensional box. This approach was used in the univariate case for the first time in [7] and subsequently in a series of papers, e.g. [11, 22, 27, 29]. Generalizations to the multivariate case were given in [13, 14, 21, 23]. Without loss of generality we consider the unit box $U = [0,1]^l$ since any nonempty box of $\mathbf{R}^l$ can be mapped affinely[1] onto this box.

The ith *Bernstein polynomial* of degree n is defined as

$$b_{n,i}(x) = \binom{n}{i} x^i (1-x)^{n-i}, \quad 0 \leq i \leq n,$$

for an arbitrary $x \in \mathbf{R}$. In the multivariate case, the Ith Bernstein polynomial of degree N is defined by

$$B_{N,I}(\mathbf{x}) = b_{n_1,i_1}(x_1) b_{n_2,i_2}(x_2) \cdot \ldots \cdot b_{n_l,i_l}(x_l), \tag{5}$$

where $\mathbf{x} = (x_1, \ldots, x_l) \in \mathbf{R}^l$.

The transformation of a polynomial from its power form (3) into its *Bernstein form* results in

$$p(\mathbf{x}) = \sum_{I \in S} b_I(U) B_{N,I}(\mathbf{x}), \tag{6}$$

where the *Bernstein coefficients* $b_I(U)$ of p over U are given by

$$b_I(U) = \sum_{J \leq I} \frac{\binom{I}{J}}{\binom{N}{J}} a_J, \quad I \in S. \tag{7}$$

We collect the Bernstein coefficients in an array $B(U)$, i.e. $B(U) = (b_I(U))_{I \in S}$. A similar notation will be employed for other sets of related coefficients. In [13] a method was presented for calculating the Bernstein coefficients efficiently by a difference table scheme (which is similar to the sweep procedure, cf. Sect. 2.2) that avoids the binomial coefficients and products in (7).

In the following, we will use a special subset of the index set S comprising those indices which correspond to the indices of the vertices of the array $B(U)$, i.e.

$$S_0 = \{0, n_1\} \times \cdots \times \{0, n_l\}.$$

We list some useful properties of the Bernstein coefficients, e.g. [9, 13, 27]. As usual, we denote the convex hull of a set A by Conv$A$.

Lemma: *Let p be a polynomial (3) of degree N. Then the following properties hold for its Bernstein coefficients $b_I(U)$ (7):*

i) Sharpness of special coefficients:

$$\forall I \in S_0 : b_I(U) = p(I/N) \tag{8}$$

ii) Convex hull property:

$$\begin{aligned} &\mathrm{Conv}\{(\mathbf{x}, p(\mathbf{x})) : \mathbf{x} \in U\} \\ &\qquad \subseteq \mathrm{Conv}\{(I/N, b_I(U)) : I \in S\}. \end{aligned} \tag{9}$$

Formula (8) follows immediately from (7). The property (9) relies on two fundamental properties of the Bernstein polynomials, viz.

$$\forall \mathbf{x} \in U : B_{N,I}(\mathbf{x}) \geq 0, \quad I \in S \tag{10}$$

and

$$\forall \mathbf{x} \in \mathbf{R}^l : \sum_{I \in S} B_{N,I}(\mathbf{x}) = 1. \tag{11}$$

This shows that the polynomial p in (6) is represented as a convex combination of Bernstein polynomials.

2.2 Sweep Procedure

The exposition in this section is based on the preliminary results in [13, 14]. We define a sweep in rth direction ($1 \leq r \leq l$) similarly to de Casteljau's algorithm in CAGD, e.g. [9], as recursively applied linear interpolation. Let D be any subbox of U generated by sweep operations (at the beginning we have $D = U$, then subsequently D is obtained by successively dividing). Starting with $B^{(0)}(D) = B(D)$ we set for $k = 1, \ldots, n_r$

[1] The corresponding shift of the polynomial can be performed efficiently by the multidimensional Horner scheme, cf. [8].

$$b^{(k)}_{i_1,\ldots,i_r,\ldots,i_l}(D) = \begin{cases} b^{(k-1)}_{i_1,\ldots,i_r,\ldots,i_l}(D) : i_r = 0,\ldots,k-1 \\ (1-\lambda)b^{(k-1)}_{i_1,\ldots,i_r-1,\ldots,i_l}(D)+ \\ \lambda b^{(k-1)}_{i_1,\ldots,i_r,\ldots,i_l}(D) : i_r = k,\ldots,n_r. \end{cases} \quad (12)$$

So, if $\lambda = 1/2$, we recursively form the arithmetic mean of two neighbouring subarrays of $B^{(k-1)}(D)$ having only in the rth coordinate full dimension, whereas some subarrays remain unchanged. To obtain the new coefficients, we apply formula (12) for $i_j = 0,\ldots,n_j$, $j = 1,\ldots,r-1,r+1,\ldots,l$.

Then the Bernstein coefficients on D_0, where the subbox D_0 is given by

$$D_0 = [\underline{d}_1,\overline{d}_1] \times \cdots \times [\underline{d}_r, \underline{d}_r + \lambda(\overline{d}_r - \underline{d}_r)] \times \cdots \times [\underline{d}_l,\overline{d}_l],$$

are obtained as $B(D_0) = B^{(n_r)}(D)$. At no extra cost we get as intermediate values the Bernstein coefficients $B(D_1)$ on the neighbouring subbox D_1

$$D_1 = [\underline{d}_1,\overline{d}_1] \times \cdots \times [\underline{d}_r + \lambda(\overline{d}_r - \underline{d}_r), \overline{d}_r] \times \cdots \times [\underline{d}_l,\overline{d}_l]$$

since for $k = 0,\ldots,n_r$ the following relation holds [14]

$$b_{i_1,\ldots,n_r-k,\ldots,i_l}(D_1) = b^{(k)}_{i_1,\ldots,n_r,\ldots,i_l}(D).$$

In analogy to CAGD we call the arrays of Bernstein coefficients $B(D_0)$ and $B(D_1)$ *patches*. It is important to note that by the sweep procedure the explicit transformation of the subboxes generated by the sweeps back to U is avoided. Fig. 1 illustrates the sweeping process for $l = 2$ and $\lambda = 1/2$.

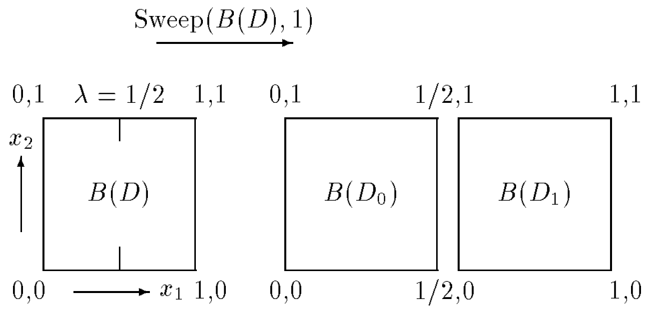

Fig. 1 Domain-splitting by the sweep procedure gives two new patches.

The algorithms presented in this paper work with the fixed value $\lambda = 1/2$ for the splitting-point. Then the multiplications required in (12) simplify to binary shifts. Let $\hat{n}$ denote the total degree (4) of polynomial (3). Since we have to perform formula (12) $n_r(n_r+1)/2$ times we need altogether $O(\hat{n}^{l+1})$ additions and binary shifts.

3 The Improved Bernstein Algorithm

The basic Bernstein Algorithm [14] was designed to check positivity of a multivariate polynomial over a box. To show robust stability of a family of polynomials (1) we may make use of the Boundary-Crossing-Theorem [12], cf. Sect. 4.3 in [2]. Then we have to check the *Hurwitz determinant* associated with the family $\tilde{p}(\mathbf{q}) = \det H(p(\mathbf{q}))$ for positivity over the box Q which can be assumed without loss of generality to be the unit box U. The underlying m-by-m matrix $H(p) = (h_{i,k}(p))$ is defined by $h_{i,k}(p) = a_{2k-i}(\mathbf{q})$, $i,k = 1,\ldots,m$, where by convention $a_n(\mathbf{q}) = 0$ if $n < 0$ or $n > m$. Since $\tilde{p}(\mathbf{q})$ is an l-variate polynomial in $\mathbf{q} = (q_1,\ldots,q_l)$ we are able to apply the Bernstein expansion provided in Sect. 2.1. If the minimum of the Bernstein coefficients $b_I(U)$, $I \in S$, is positive it follows by (9) that $p(\mathbf{q})$ is stable for Q. If there exists a nonpositive sharp Bernstein coefficient $b_{I_0}(U)$, $I_0 \in S_0$, the family of polynomials (1) is not stable for all $\mathbf{q} \in Q$ since the Hurwitz determinant associated with the family assumes nonpositive values on U. If both cases do not apply the sweep procedure is performed splitting U to obtain two new patches on which we proceed as before.

The basic Bernstein Algorithm uses subdivisions which we can express as a sequence of 2^{l-1} sweeps. The disadvantage of the subdivision based algorithm is, that we can test positivity only after a complete subdivision step. But, since properties like the convex hull property remain true for the patches generated by one sweep, we can perform the positivity test after each sweep. This saves a lot of computational work.

Further improvements concern the selection of the sweep direction and the patch selection of the depth first strategy (see below). We show in Ex. 1 in Ch. 5 that the improvements have led to a significant speeding up of the algorithm.

3.1 Basic Procedures

Selection of the Sweep Direction The definition of the sweep procedure shows that we are free in choosing the sweep direction. Our selection rules are based on the observation that in many cases it may be advantageous to sweep in a particular coordinate direction to increase the probability for finding a nonpositive sharp Bernstein coefficient thus proving that the polynomial is not positive. Our selection rules profit from the easy calculation of the partial derivatives of a polynomial in Bernstein form, e.g. [9, 10].

To shorten some expressions in the sequel we associate with an index $I = (i_1,\ldots,i_r,\ldots,i_l)$ the index $I_{r,k} = (i_1,\ldots,i_r+k,\ldots,i_l)$, where $0 \le k + i_r \le n_r$. Then the μth partial derivative with respect to x_r of (6) is given by the following formula ($1 \le r \le l$):

$$\frac{\partial^\mu p}{\partial x_r^\mu}(\mathbf{x})=\frac{n_r!}{(n_r-\mu)!}\sum_{I\leq N_{r,-\mu}}\triangle_r^{(\mu)}b_I(D)B_{N_{r,-\mu},I}(\mathbf{x}), \qquad (13)$$

where the difference operator $\triangle_r$ is defined recursively by

$$\begin{aligned}&\triangle_r^{(0)}b_I(D)=b_I(D) \quad \text{and for } k=1,\dots,\mu\\ &\triangle_r^{(k)}b_I(D)=\triangle_r^{(k-1)}b_{I_{r,1}}(D)-\triangle_r^{(k-1)}b_I(D).\end{aligned}$$

To decide which sweep direction to choose we estimate

$$\max_{\mathbf{x}\in D}\left|\frac{\partial^\mu p}{\partial x_r^\mu}(\mathbf{x})\right| \qquad (14)$$

neglecting the factorials from above by

$$\tilde{I}_r^{(\mu)}=\max_{I\leq N_{r,-\mu}}|\triangle_r^{(\mu)}b_I(D)|. \qquad (15)$$

Here we have used the triangle inequality and properties (10), (11).

For $\mu\in\{1,2\}$ we choose that r_0 with maximum value

$$\tilde{I}_{r_0}^{(\mu)}=\max_{j=1,\dots,l}\tilde{I}_j^{(\mu)}. \qquad (16)$$

We define a function

$$SelectSweepDirection(B(D),\mu)$$

which returns the value for r_0 for a fixed $\mu\in\{1,2\}$.

It should be noted that more refined measures for the curvature on the basis of (14) can be established – at the cost of higher computational effort. However, Ex. 1 in Ch. 5 shows that the rule based on the second partial derivative (16) is sufficient for practical purposes.

Depth First Strategy In case of an unstable family of polynomials we have to find a short path through a given binary tree that leads to a nonpositive sharp Bernstein coefficient. Since we do not want to visit too many nodes of the tree the choice of an appropriate path (without the need of walking back) is of great importance. An easily implemented rule is the *selection rule*: *Choose the patch with minimum Bernstein coefficient.* Of course, we are considering here only patches with at least one nonpositive Bernstein coefficient which is not sharp because otherwise local positivity or global nonpositivity is shown. Similar to the procedure for selecting the sweep direction we define a function

$$PatchSelection(B(D_0),B(D_1))$$

that returns the index value for which the minimum Bernstein coefficient is attained.

Test Procedure By the test procedure a patch is checked for local positivity or sharp nonpositive coefficients. If the coefficients are neither locally positive nor exists a sharp nonpositive coefficient the test procedure returns *undecidable.*

$$Test(B(D))=\begin{cases}\text{US} & : \ \exists I_0\in S_0: b_{I_0}(D)\leq 0\\ \text{ST} & : \ \forall I\in S: b_I(D)>0\\ \text{UD} & : \ \text{otherwise}\end{cases}.$$

3.2 The Algorithm

Our algorithm works with a stack and we assume the standard operations *MakeStack*, *Push*, *Pop*, and *Isempty* to be implemented, e.g. [19, 31]. A stack element is defined as a pointer to the Bernstein coefficients $B(U)$. Then the Improved Bernstein Algorithm reads as follows:

Procedure Bernstein(B(D),μ)

begin S=MakeStack();
t_0=t_1=Test(B(D));
if (t_0=UD) **then** Push(S,B(D));
while ($t_0\neq$ US $\wedge t_1\neq$ US $\wedge\ \neg$Isempty(S)) **do**

    begin
    B(D)=Pop(S);
    r$_0$=SelectSweepDirection(B(D),μ);
    $\{B(D_0),B(D_1)\}$ = Sweep(B(D),r$_0$);
    t_0=Test(B(D_0)); t_1=Test(B(D_1));
    if (t_0=UD $\wedge$ t_1=UD)
        if (PatchSelection(B(D_0),B(D_1))=0)
            Push(S,B(D_1)); Push(S,B(D_0));
        else
            Push(S,B(D_0)); Push(S,B(D_1));
    else if (t_0=UD) Push(S,B(D_0));
    else if (t_1=UD) Push(S,B(D_1));
    end

if (t_0=ST $\wedge$ t_1=ST) **return** RS;
else return $\neg$RS;
end

Algorithm 1

The algorithm returns robustly stable (RS) if the stack is empty and the last two patches are locally positive. Otherwise, if a nonpositive sharp coefficient is found it returns not robustly stable ($\neg$RS).

4 The Convex Hull Bernstein Algorithm

The numerical results in [28] show that already the basic Bernstein Algorithm [14] applied to robust stability problems with a moderate number of parameters is fast. A considerable speeding up of the algorithm was achieved by its improvements explained in Ch. 3. But a serious drawback of the approach is that the Hurwitz determinant has to be computed (which however can be done by a symbolic manipulation package if the number of parameters is moderate) and that the algorithm works with this determinant. This causes a considerable blowing up of the original problem. E.g. one array of Bernstein coefficients has a space complexity of $O((md)^l)$, for d see (2). In contrast, the algorithm to be introduced in the sequel has a space complexity of $O(\max\{m, d\}^{l+1})$ and that is in the most cases considerably less.

The new algorithm explores the value set of the family of polynomials (1)

$$\mathcal{P}(\Omega) = \{p(j\omega, \mathbf{q}) : \omega \in \Omega \wedge \mathbf{q} \in Q\}, \quad \Omega \subset \mathbf{R}.$$

Having found a stable member of the family it suffices by the continuous dependency of the zeros of a polynomial from its coefficients to test $0 \notin \mathcal{P}(\mathbf{R})$. In [37] we discuss how one can find a tight compact interval $\Omega \subset [0, \infty)$ such that from $0 \notin \mathcal{P}(\Omega)$ we can conclude the robust stability of the family (1).

We split the polynomial $p(j\omega, \mathbf{q})$ into its even and odd parts

$$p(j\omega, \mathbf{q}) = p_e(\omega^2, \mathbf{q}) + j\omega p_o(\omega^2, \mathbf{q}), \tag{17}$$

where

$$\begin{aligned} p_e(\omega^2, \mathbf{q}) &= \quad a_m(\mathbf{q}) - a_{m-2}(\mathbf{q})\omega^2 + a_{m-4}(\mathbf{q})\omega^4 - + \ldots \\ p_o(\omega^2, \mathbf{q}) &= a_{m-1}(\mathbf{q}) - a_{m-3}(\mathbf{q})\omega^2 + a_{m-5}(\mathbf{q})\omega^4 - + \ldots . \end{aligned}$$

It should be noted that an improvement of the order of complexity can be obtained by substituting $\sigma = \omega^2$ and by considering the pair $(p_e(\sigma), p_o(\sigma))$ rather than (17). However, to keep the presentation simpler we use the form (17).

By running some of the earlier developed procedures twice – simultaneously for the real part $p_e(\mathbf{q})$ and for the imaginary part $\omega p_o(\mathbf{q})$ – we obtain a set of points in the complex plane. Then we compute their convex hull what can be done in optimal time using $O(\nu \log \nu)$ operations, e.g. [24, 26], where ν denotes the number of points, and check whether the origin of the complex plane is contained in this convex hull. If it is outside and if there exists a stable member, the family of polynomials is robustly stable. Otherwise (if the origin is inside the convex hull) we perform an inclusion test for the value set (see below). If it fails, i.e. it can not be verified that the origin is in the value set, we apply our sweep procedure splitting the domain to obtain two new patches on which we proceed as before. If no patch remains and all inclusion tests have failed the family of polynomials is robustly stable (again under assumption that there is a stable member). Otherwise, if an inclusion test is successful the algorithm aborts immediately because we have found an unstable polynomial.

In order to use our previous notation we assume $\dim Q = l-1$. So we write $(x_1, \ldots, x_l) = (\omega, q_1, \ldots, q_{l-1})$. Then we map $\Omega \times Q$ affinely onto the unit cube U. Transforming the real and imaginary part of (17) into Bernstein form yields complex valued Bernstein coefficients $b_I(D)$, $I \in S$, where D is any subbox of U generated by sweeps. In the sequel we suppress the explicit reference to D and write $\mathrm{Conv}B$ for $\mathrm{Conv}\{b_I : I \in S\}$.

In [27] it is shown that the convex hull property also holds for univariate polynomials having complex coefficients. By (10) and (11) it is easy to see that

$$\mathrm{Conv}\{p(\mathbf{x}) : \mathbf{x} \in D\} \subseteq \mathrm{Conv}B.$$

By any standard convex hull algorithm, cf. [24, 26], we can check whether the origin belongs to $\mathrm{Conv}B$. If it is inside we apply the sweep procedure to get a better approximation of the value set. An approach based merely on the sweep procedure and a convex hull algorithm is only sufficient to verify robust stability. However, to show that a family of polynomials is *not* robustly stable we need warranty that the origin is contained in the value set.

4.1 Geometric Inclusion Test

In case of an unstable family of polynomials the sweeping process has to be terminated by a criterion which gives a guarantee that the origin belongs to the value set $\mathcal{P}(U)$. Otherwise, if no inclusion test is applied, the algorithm would produce sequences of convex hulls converging to zeros of polynomial (17).

By (10), (11), and the definition of the Bernstein polynomials $B_{N,I}(\mathbf{x})$ (5) we immediately obtain the following local convex hull result.

Edge-Lemma: *Let an edge of U be parameterized as*

$$w_\lambda = I_0/N + \lambda(I_1/N - I_0/N), \quad \lambda \in [0, 1],$$

with $I_0 = (i_1, \ldots, 0, \ldots, i_l)$, $I_1 = (i_1, \ldots, n_r, \ldots, i_l) \in S_0$.

Then $\mathrm{Conv}\{p(w_\lambda) : \lambda \in [0, 1]\}$
$\subseteq \mathrm{Conv}\{b_{i_1,\ldots,i_r,\ldots,i_l} : i_r = 0, 1, \ldots, n_r\}$.

The lemma tells that the convex hull of the image of an edge of U under p is contained in the convex hull of the Bernstein coefficients associated with this edge.

Now we consider four elements of S_0, where we manipulate the rth and sth index,

$$\begin{aligned} I_{0,0} &= (i_1, \ldots, \ 0\ , \ldots, \ 0\ , \ldots, i_l) \\ I_{1,0} &= (i_1, \ldots, n_r, \ldots, \ 0\ , \ldots, i_l) \\ I_{1,1} &= (i_1, \ldots, n_r, \ldots, n_s, \ldots, i_l) \\ I_{0,1} &= (i_1, \ldots, \ 0\ , \ldots, n_s, \ldots, i_l). \end{aligned}$$

To state our inclusion condition we denote by R a region in the complex plane circumscribed by the edges ($t \in [0,1]$)

$$\begin{array}{l} tb_{I_{1,0}} + (1-t)b_{I_{0,0}} \\ tb_{I_{1,1}} + (1-t)b_{I_{1,0}} \\ tb_{I_{0,1}} + (1-t)b_{I_{1,1}} \\ tb_{I_{0,0}} + (1-t)b_{I_{0,1}}. \end{array} \tag{18}$$

Accordingly, we define the four sets of Bernstein coefficients associated with the edges

$$\begin{array}{l} B^1 = \{b_{i_1,\ldots,i_r,\ldots,0,\ldots,i_l} : i_r = 0,\ldots,n_r\} \\ B^2 = \{b_{i_1,\ldots,n_r,\ldots,i_s,\ldots,i_l} : i_s = 0,\ldots,n_s\} \\ B^3 = \{b_{i_1,\ldots,i_r,\ldots,n_s,\ldots,i_l} : i_r = 0,\ldots,n_r\} \\ B^4 = \{b_{i_1,\ldots,0,\ldots,i_s,\ldots,i_l} : i_s = 0,\ldots,n_s\}. \end{array}$$

For a set A, A° and $\overline{A}$ denotes as usual its (topological) interior and its closure, respectively. We set

$$C = \bigcup_{i=1}^{4} (\text{Conv}\, B^i)^\circ.$$

Now we are in the position to state our inclusion test based on the four edges, cf. Fig. 2.

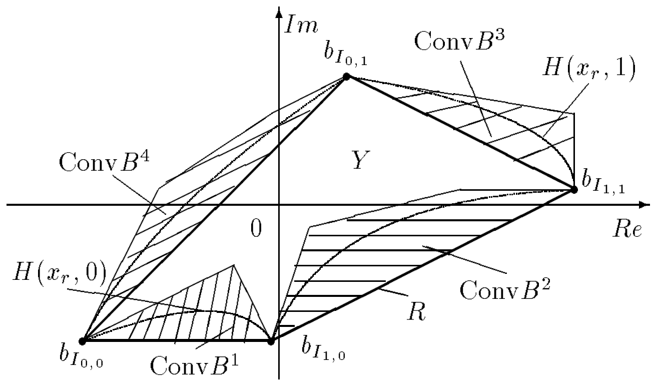

Fig. 2 Geometry of the inclusion condition.

Four-Edges-Test: *Let $Y = \overline{R^\circ \backslash C}$ and assume that $Y \neq \emptyset$. Then the origin is contained in $\mathcal{P}(U)$ if it is contained in Y.*

Proof Without loss of generality we may assume that the edges of R (18) have no proper intersection points than the Bernstein coefficients and that these points are pairwise different. Otherwise R degenerates or can be split into two triangles and we may proceed similarly. Let the function H be defined by

$$H(x_r, x_s) = p(i_1/n_1, \ldots, x_r, \ldots, x_s, \ldots, i_l/n_l).$$

By the Edge-Lemma it follows that the paths generated by $H(x_r, 0)$ and $H(x_r, 1)$, $x_r \in [0,1]$, are contained in Conv B^1 and Conv B^3, respectively. Since p is continous we have found a continous deformation of $H(x_r, 0)$ into $H(x_r, 1)$ such that the set Y is completely covered by function values of the map $p(\mathbf{x})$ with $\mathbf{x} \in U$. This proves the statement. $\square$

Time Complexity of the Inclusion Test Each Bernstein patch has $2^l \binom{l}{0}$ sharp coefficients, $2^{l-1}\binom{l}{1}$ edges and $2^{l-2}\binom{l}{2}$ (twodimensional) faces. In worst case we have to check $O(2^l\, l^2)$ faces by the inclusion test. For each face we test first if the origin is inside of R and if yes, we check the four convex hulls associated with the edges for zero inclusion. To test R costs a number of operations which is independent of l and the total degree $\hat{n}$ (4), i.e. $O(1)$ operations. The construction of the four convex hulls of the edges costs $O(\hat{n} \log \hat{n})$, and the zero inclusion tests for the convex hulls require additional $O(\log \hat{n})$ operations, e.g. [26]. So we have a total time complexity for the inclusion test of $O(2^l\, l^2\, (\hat{n} \log \hat{n} + \log \hat{n})) = O(2^l\, l^2\, \hat{n} \log \hat{n})$.

4.2 Basic Selection Procedures

Selection of the Sweep Direction The choice of the sweep direction is based on the partial derivatives of the complex valued function p similarly to the selection rule (16).

Patch Selection After sweeping along an axis we have still the choice between two patches. Let $p_i^{(w)}$, $i = 1, \ldots, \mu_w$, be the etreme points of Conv$B(D_w)$, i.e.

$$\text{Conv}B(D_w) = \text{Conv}\{p_i^{(w)} : i = 1, \ldots, \mu_w\}, \quad w = 0, 1.$$

We take that patch $B(D_w)$ for which the Euclidean distance of the center of gravity of the equally weighted points $p_i^{(w)}$ to the origin, i.e.

$$\frac{1}{\mu_w} \left| \sum_{i=1}^{\mu_w} p_i^{(w)} \right|, \quad w = 0, 1,$$

is minimal. The procedure

$$PatchSelection(B(D_0), B(D_1))$$

again returns the value from $\{0,1\}$ identifying the patch to be swept first.

Test Procedure The test procedure is based on the Four-Edges-Test and is somehow different to that introduced for the first algorithm. As convex hull algorithm we have implemented Graham's algorithm [26]. We further need a procedure for testing whether a point is located inside a convex hull or not, cf. [26].

$$Test(B) = \begin{cases} \text{US} & : \quad 0 \in Y \\ \text{ST} & : \quad 0 \notin \text{Conv}B \\ \text{UD} & : \quad \text{otherwise} \end{cases}.$$

The CHB Algorithm The body of the Convex Hull Bernstein Algorithm (shortly CHBA) is nearly identical to the improved Bernstein procedure. We exchange only the sweep direction procedure, the patch selection, the test procedure and we have to run the sweep procedure twice: For the real and simultaneously for the imaginary part of the polynomial.

5 Examples

For each example we give the number of sweeps required to obtain the result and the *sweep depth* that describes the maximum number of sweeps performed on one patch. Furthermore, we present the running times (on a Hewlett Packard workstation 9000/755) and in case of value set analysis we give the interval on the imaginary axis the analysis can be restricted to.

5.1 Positivity Tests

We use the Improved Bernstein Algorithm for testing the Hurwitz determinant for positivity. For the selection of the sweep direction we work with the second partial derivative of the polynomial, cf. Sect. 3.1.

Ex. 1 [1] We consider the following polynomial p of third degree with coefficients depending multiaffinely on l parameters q_i, where $q_i \in [0,3]$ for $i = 1, \ldots, l$:

$$p(s, \mathbf{q}) = s^3 + a_1(\mathbf{q})s^2 + a_2(\mathbf{q})s + a_3(\mathbf{q}),$$

where

$$a_1(\mathbf{q}) = a_2(\mathbf{q}) = l + \sum_{i=1}^{l} q_i$$

and

$$a_3(\mathbf{q}) = l(l-1) + \tau^2 + 2(l+1)\sum_{i=1}^{l} q_i + 2\sum_{i=1}^{l-1}\sum_{j=i+1}^{l} q_i q_j.$$

We have run positivity tests for $\tau = 10^{-3}$ and different values of l. Tab. 1 shows the characteristics of each test.

l	sweep depth	sweeps	t/sec.
6	69	69	0.5
7	81	81	2.4
8	94	94	22.4
9	106	106	131.8
10	117	118	663.5
11	129	129	3789.0

Tab. 1 Results for the l parameter polynomial.

The program reports that the family of polynomials is not robustly stable. In fact, the family has an unstable sphere of radius τ centered around $\mathbf{q}^* = (1, \ldots, 1)$ [1].

For comparison we give the results for $l = 6, 7$ for the subdivision based Bernstein Algorithm [28], cf. the remarks in Ch. 3. The computing times for $l = 6$ were 36.6 sec. and for $l = 7$ approximately 19 min.

Now we would like to demonstrate the efficiency of the sweep direction selection: We vary the intervals for the parameters for fixed $l = 6$, viz. we set $\underline{q}_i = 0$ for $i = 1, \ldots, 6$ and vary the upper bounds $\overline{\mathbf{q}} = (\overline{q}_1, \ldots, \overline{q}_6)$.

$\overline{\mathbf{q}}$	sweep depth	sweeps	t/sec.
(3, 3, 3, 3, 3, 6)	70	70	0.46
(3, 3, 3, 3, 6, 6)	71	71	0.46
(3, 3, 3, 6, 6, 6)	72	72	0.45
(3, 3, 6, 6, 6, 6)	73	73	0.41
(3, 6, 6, 6, 6, 6)	74	74	0.47
(6, 6, 6, 6, 6, 6)	75	75	0.46
(3,6,12,24,48,96)	84	84	0.51

Tab. 2 Variation of the upper bounds.

Tab. 2 shows that if we double the parameter box the algorithm needs exactly one more sweep. We see that the number of sweeps and the sweep depth in Tabs. 1,2 are nearly identical. This documents the efficiency of our selection rules.

5.2 Value Set Based Tests

We give some numerical results of the CHBA, where the algorithm uses the first partial derivative for selecting the sweep direction, cf. Sect. 4.2.

Ex. 2 We consider now the characteristic polynomial of a control problem associated with the Daimler-Benz citybus [33], p. 46.

$p(s, v, m) = 4.49876 \cdot 10^{14}v^2 + 3.6215 \cdot 10^{15}vs + 5.24855 \cdot 10^{14}v^2s + 4.22509 \cdot 10^{15}vs^2 + 1.12469 \cdot 10^{14}v^2s^2 + 5.69531 \cdot 10^{9}mv^2s^2 + 4.19884 \cdot 10^{15}s^3 + 9.05376 \cdot 10^{14}vs^3 + 6.90414 \cdot 10^{9}mv^2s^3 + 3.35907 \cdot 10^{14}s^4 + 1.67933 \cdot 10^{10}mvs^4 + 1.4446 \cdot 10^{9}mv^2s^4 + 1.34363 \cdot 10^{13}s^5 + 1.34347 \cdot 10^{9}mvs^5 + 8.30756 \cdot 10^{5}mv^2s^5 + 1.5625 \cdot 10^{4}m^2v^2s^5 + 2.68726 \cdot 10^{11}s^6 + 5.37386 \cdot 10^{7}mvs^6 + 1.66151 \cdot 10^{4}mv^2s^6 + 1.25 \cdot 10^{3}m^2v^2s^6 + 1.07477 \cdot 10^{6}mvs^7 + 50m^2v^2s^7 + m^2v^2s^8$

We have inspected the family of polynomials for $\omega \in [0.18, 751]$, where the intervals for the parameters are given by

$$v \in [1, 20] \quad m \in [9950, 32000].$$

After 0.05 sec. we have the result that this family is robustly stable. The CHBA reports 50 sweeps with a sweep depth of 21. Fig. 3 shows an approximation of the value set.

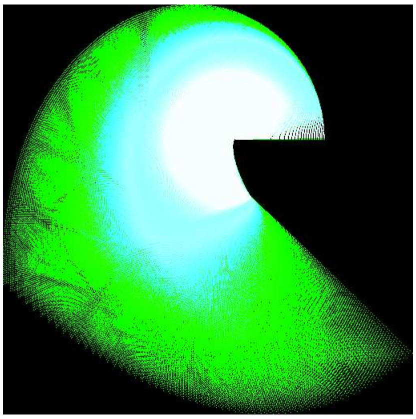

Fig. 3 Subset of the value set associated with the citybus.

Ex. 3 Control of the Fiat Dedra engine [6], pp. 29-36, 154-159, 358-360. The characteristic polynomial is of seventh degree with seven parameters entering quadratically into the transfer function. The frequency and the parameters vary inside the intervals

$$\begin{array}{ll} \omega \in [0.0058, 2.1738] & q_1 \in [2.1608, 3.4329] \\ q_2 \in [0.1027, 0.1627] & q_3 \in [0.0357, 0.1139] \\ q_4 \in [0.2539, 0.5607] & q_5 \in [0.0100, 0.0208] \\ q_6 \in [2.0247, 4.4962] & q_7 \in [1.0000, 10.000]. \end{array}$$

The CHBA reports after 10.6 sec. that this family is robustly stable, cf. Fig. 4, where 83 sweeps with a sweep depth of 17 are required.

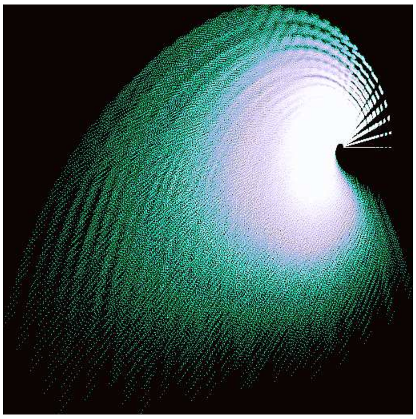

Fig. 4 Subset of the value set associated with the Fiat Dedra engine.

Ex. 4 Our next example has a multiaffine parameter dependency, where thirteen parameters are involved [3].

$$\begin{aligned} p_1(s, c_1, d_1, m_1) &= m_1 s^2 + d_1 s + c_1 + c_{12} \\ p_2(s, c_2, d_2, m_2) &= m_2 s^2 + d_2 s + c_2 + c_{12} \end{aligned}$$

$$\begin{aligned} N_c(s, \mathbf{b}) &= b_3 s^3 + b_2 s^2 + b_1 s + b_0 \\ D_c(s, \mathbf{a}) &= s^3 + a_2 s^2 + a_1 s + a_0 \end{aligned}$$

For fixed $c_{12} = 1$ the family of polynomials to be checked for zero exclusion is given by $(p_1 p_2 - 1) D_c + N_c$. As in [3, 25] we choose the following parameter intervals and analyze the value set for $\omega \in [0, 70]$.

$$\begin{array}{lll} m_1 \in [1, 3] & d_1 \in [0.5, 2] & c_1 \in [1, 2] \\ m_2 \in [2, 5] & d_2 \in [0.5, 2] & c_2 \in [2, 4] \end{array}$$

$$\begin{array}{rl} a_0 \in [17100, 20900] & b_0 \in [212062.5, 259187.5] \\ a_1 \in [1305, 1595] & b_1 \in [805837.5, 984912.5] \\ a_2 \in [55.8, 68.2] & b_2 \in [721012.5, 881237.5] \\ & b_3 \in [424125.0, 518375.0]. \end{array}$$

The CHBA finds after 9 sweeps with a sweep depth of 9 in 37.8 sec. that this family must have unstable members. This stands in contradiction to [25], where this family was checked by a genetic algorithm, and [3], where the value set was analyzed using tree structured decomposition[2].

For further analysis we have tested a subfamily choosing $m_1 \in [1, 1.1]$ and the other parameters as corners of their respective parameter intervals: $m_2 = 2$, $d_1 = 0.5$, $d_2 = 0.5$, $c_1 = 1$, $c_2 = 2$, $a_0 = 17100$, $a_1 = 1305$, $a_2 = 68.2$, $b_0 = 212062.5$, $b_1 = 805837.5$, $b_2 = 881237.5$, $b_3 = 518375$. Then the following polynomial

$$p(s, m_1) = 2m_1 s^7 + 136.9 m_1 s^6 + s^6 + 2647.1 m_1 s^5 + 72.45 s^5 + 35057.1 m_1 s^4 + 1597.35 s^4 + 12465 m_1 s^3 + 541196.75 s^3 + 51300 m_1 s^2 + 957516 s^2 + 855112.5 s + 297562.5$$

with $\omega \in [1, 25]$ results. The CHBA verifies after 5 sweeps with a sweep depth of 4 in less then 0.01 sec. that this subfamily is *not* robustly stable. In Fig. 5 the origin is overlapped by the value set of this subfamily confirming our conclusion.

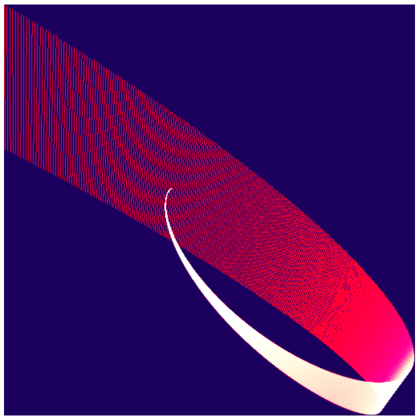

Fig. 5 Part of the value set of the subfamily.

[2] We were informed that the result in [3] was obtained by choosing the interval for ω to tight.

6 Conclusions

We have presented two algorithms for checking robust stability of a family of polynomials with coefficients depending polynomially on parameters. The improved Bernstein Algorithm is based on a positivity test of the Hurwitz determinant. In case of polynomials with a moderate number of parameters and/or of not too high degree it is very fast. But in case of larger problems it is impossible to calculate the Hurwitz determinant because of space limitations. In such cases the new algorithm based on the value set analysis is advantageous because the memory requirements do not increase as fast as they would if we would work with the Hurwitz determinant.

We have seen that the Convex Hull Bernstein Algorithm solves larger problems. But in case of unstable polynomials the algorithm could be faster if the patch selection and the selection of the sweep direction would be more efficient by preventing the algorithm to follow inefficient paths. Further research should address also the question of reducing the storage needed since the application of the algorithms to very large problems is complicated by their excessive memory requirements. Future research should be directed to the investigation of the convergence properties of the algorithm. In [23] it was shown that the basic Bernstein Algorithm with breadth first strategy converges quadratically. There is some evidence that the convergence of the Convex Hull Bernstein Algorithm is also quadratic.

An advantage of the Convex Hull Bernstein Algorithm is the capability to visualize the value set. This could facilitate the design of dynamic systems.

Acknowledgements

This work was supported by the Ministry of Science and Research Baden-Württemberg. We would like to thank further Dipl.-Inf.(FH) Tobias Link for the implementation of the modules for calculating bounds on polynomials.

References

[1] J. Ackermann, "Does it suffice to check a subset of multilinear parameters in robustness analysis?," *IEEE Trans. Automat. Contr.*, vol. 37, pp. 487-488, 1992.

[2] J. Ackermann, *Robust Control, Systems with Uncertain Physical Parameters.* London: Springer, 1993.

[3] J. Ackermann and W. Sienel, "What is a "large" number of parameters in robust systems?," in *Proc. 29th Conf. Decision Contr.*, Honolulu, Hawai, 1990, pp. 3496-3497.

[4] V. Balakrishnan, S. Boyd, and S. Balemi, "Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems," *J. Robust and Nonlinear Contr.*, vol. 1, pp. 295-317, 1991.

[5] V. Balakrishnan and S. Boyd, "Global optimization in control system analysis and design," in *Control and Dynamic Systems, vol. 53: High Performance Systems Techniques and Applications*, C.T. Leondes, Ed., San Diego: Academic Press, 1992.

[6] B.R. Barmish, *New Tools for Robustness of Linear Systems.* New York: Macmillan, 1994.

[7] G.T. Cargo and O. Shisha, "The Bernstein form of a polynomial," *J. Res. Nat. Bur. Standards Sect. B*, vol. 70B, pp. 79-81, 1966.

[8] M.L. Dowling, "A fast parallel Horner algorithm," *SIAM J. Comput.*, vol. 19, pp. 133-142, 1990.

[9] G. Farin, *Curves and Surfaces for CAGD. A Practical Guide*, 3rd Ed., New York: Academic Press, 1993.

[10] R.T. Farouki and V.T. Rajan, "Algorithms for polynomials in Bernstein form," *Computer Aided Geometric Design*, vol. 5, pp. 1-26, 1988.

[11] H.C. Fischer, "Range computations and applications," in *Contributions to Computer Arithmetic and Self-Validating Numerical Methods.* C. Ullrich, Ed., J.C. Baltzer, pp. 197-211, 1990.

[12] R.A. Frazer and W.J. Duncan, "On the criteria for the stability of small motions," *Proc. Royal Soc. A*, vol. 124, pp. 642-654, 1929.

[13] J. Garloff, "Convergent bounds for the range of multivariate polynomials," in *Interval Mathematics 1985.* K. Nickel, Ed., Lecture Notes in Computer Science, vol. 212, Berlin: Springer, pp. 37-56, 1986.

[14] J. Garloff, "The Bernstein Algorithm," *Interval Computations*, vol. 2, pp. 154-168, 1993.

[15] R.R.E. de Gaston and M.G. Safonov, "Exact calculation of the multiloop stability margin," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 156-171, 1988.

[16] D. Kaesbauer, "On robust stability of polynomials with polynomial parameter dependency: two/three parameter cases," *Automatica*, vol. 29, pp. 215-217, 1993.

[17] E.A. Kalinina and A.Yu. Uteshev, "Determination of the number of roots of a polynomial lying in a given algebraic domain," *Linear Algebra and Appl.*, vol. 185, pp. 61-81, 1993.

[18] J.C. Kantor and B.F. Spencer Jr., "On real parameter stability margins and their computations," *Int. J. Contr.*, vol. 57, pp. 453-462, 1993.

[19] D.E. Knuth, *The Art of Computer Programming*, vol. 1. 2nd Ed., Reading: Addison-Wesley, 1973.

[20] J. Kogan, *Robust Stability and Convexity. An Introduction.* Lect. Notes in Contr. and Inf. Sci., vol. 201. London: Springer, 1995.

[21] J.M. Lane and R.F. Riesenfeld, "A theoretical development for the computer generation and display of piecewise polynomial surfaces," *Trans. Pattern Anal. Mach. Intel.*, vol. 2, pp. 35-46, 1980.

[22] J.M. Lane and R.F. Riesenfeld, "Bounds on a polynomial," *BIT*, vol. 21, pp. 112-117, 1981.

[23] S. Malan, M. Milanese, M. Taragna, and J. Garloff, "B^3 algorithm for robust performances analysis in presence of mixed parametric and dynamic pertubations," in *Proc. 31st Conf. Decision Contr.*, Tucson, Arizona, 1992, pp. 128-133.

[24] K. Mulmuley, *Computational Geometry, An Introduction through Randomized Algorithms.* Englewood-Cliffs: Prentice Hall, 1994.

[25] T.M. Murdock, W.E. Schmitendorf, and S. Forrest, "Use of a genetic algorithm to analyze robust stability problems," in *Proc. Amer. Control Conf.*, Boston, 1991, pp. 886-889.

[26] F.P. Preparata and M.I. Shamos, *Computational Geometry: An Introduction.* 3rd Ed., New York: Springer, 1990.

[27] T.J. Rivlin, "Bounds on a polynomial," *J. Res. Nat. Bur. Standards Sect. B*, vol. 74B, pp. 47-54, 1970.

[28] C. Roguet and J. Garloff, "Computational experiences with the Bernstein algorithm," Konstanz: Fachhochschule Konstanz, Fachbereich Informatik, Tech. Rep. No. 9403, 1994.

[29] J. Rokne, "Bounds for an interval polynomial," *Computing*, vol. 18, pp. 225-240, 1977.

[30] L. Saydy, A.L. Tits, and E.H. Abed, Guardian maps and the generalized stability of parameterized families of matrices and polynomials," *Mathematics of Control, Signals, and Systems*, vol. 3, pp. 345-371, 1990.

[31] R. Sedgewick, *Algorithms.* Reading: Addison-Wesley, 1988.

[32] A. Sideris and R.S. Sánchez Peña, "Fast computation of the multivariable stability margin for real interrelated uncertain parameters," *IEEE Trans. Automat. Contr.*, vol. 34, pp. 1272-1276, 1989.

[33] W. Sienel, *Analyse und Entwurf von robusten Regelungssystemen durch Konstruktion von komplexen Wertemengen.* Dissertation, Fortschr.-Ber. VDI Reihe 8 No. 476. Düsseldorf: VDI-Verlag, 1995.

[34] A. Vicino, A. Tesi, and M. Milanese, "Computation of nonconservative stability perturbations bounds for systems with nonlinearly correlated uncertainties," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 835-841, 1990.

[35] E. Walter and L. Jaulin, "Guaranteed characterization of stability domains via set inversion," *IEEE Trans. Automat. Contr.*, vol. 39, pp. 886-889, 1994.

[36] E. Zeheb, "Necessary and sufficient conditions for robust stability of a continuous system - The continuous dependency case illustrated via multilinear dependency," *IEEE Trans. Circuits and Systems*, vol. 37, pp. 47-53, 1990.

[37] M. Zettler and J. Garloff, "Robustness analysis of polynomials with polynomial parameter dependency using Bernstein expansion", Konstanz: Fachhochschule Konstanz, Fachbereich Informatik, Tech. Rep. No. 9601, 1996.