Effective Computation of Rational Approximants and Interpolants

Bernhard Beckermann (bbecker@ano.univ-lille1.fr)
Laboratoire d'Analyse Numérique et d'Optimisation,
Université des Sciences et Technologies de Lille,
59655 Villeneuve d'Ascq Cedex, France

George Labahn (glabahn@daisy.uwaterloo.ca)
Department of Computer Science,
University of Waterloo,
Waterloo, Ontario, Canada

Abstract. This paper considers the problem of effective algorithms for some problems having structured coefficient matrices. Examples of such problems include rational approximation and rational interpolation. The corresponding coefficient matrices include Hankel, Toeplitz and Vandermonde-like matrices. Effective implies that the algorithms studied are suitable for implementation in either a numeric environment or else a symbolic environment.

The paper includes two algorithms for the computation of rational interpolants which are both effective in symbolic environments. The algorithms use arithmetic that is free of fractions but at the same time control the growth of coefficients during intermediate computations. One algorithm is a look–around procedure which computes along a path of closest normal points to an offdiagonal path while the second computes along an arbitrary path using a look–ahead strategy. Along an antidiagonal path the look–ahead recurrence is closely related to the Subresultant PRS algorithm for polynomial GCD computation. Both algorithms are an order of magnitude faster than alternative methods which are effective in symbolic environments.

Keywords: Rational approximant, Rational interpolant, Fraction-free arithmetic

AMS(MOS): 65D05,41A21

1. Introduction

Mathematical algorithms are often described in terms of general algebraic domains having certain well defined mathematical structures, for example an integral domain or a field. Efficiency is typically determined by a count of arithmetic operations and answers are correct because all arithmetic is exact. However when it comes to the implementation of mathematical algorithms a number of new issues appear. In floating point environments the arithmetic domain ceases to have the same rules as the algebraic environment on which they are based. Arithmetic operations have errors which result in incorrect rather than correct answers. In symbolic environments encountered when implementing in

computer algebra systems such as Maple and Mathematica it is usually not enough to have a general ring or field specified as a domain. Rather, implementations typically need to deal with specific computable domains such as the integers, rational numbers, algebraic extensions of rationals, or polynomial domains such as $\mathbb{D}[a_1, \ldots, a_n]$ where $\mathbb{D}$ is computable. In these cases answers are correct since the arithmetic is exact, but the algorithms are not necessarily efficient since the basic assumption that all arithmetic operations have a constant cost is no longer valid.

The differences between general mathematical computation and algorithms for implementation in specific environments can be illustrated by some simple remarks. In the case of rational approximation, for example, there are exact algorithms which compute a Padé approximant of type (n, n) in superfast complexity of $\mathcal{O}(n \log^2 n)$ operations [15]. However, the fastest known numerically stable algorithms which include an error analysis have so far only reached a complexity of $\mathcal{O}(n^2)$ operations. In the case of exact environments one can simply point out that when Padé approximants were implemented in Maple in the middle 1980's the algorithm used was Gaussian elimination — this even though a large number of fast and superfast methods were available at that time. The reason that Gaussian elimination was preferred was that it was the only method at the time which could deal efficiently with the growth of coefficients encountered in exact arithmetic environments [25].

In this paper we study how some mathematical algorithms have been extended to consider numeric and symbolic computing environments. The particular problems of interest are problems of computing rational approximants and rational interpolants of scalar power series, problems that are defined by linear systems having structured coefficient matrices. We compute the coefficients of the polynomial numerators and denominators over a monomial basis. Other forms of such computations, such as a such as computing the coefficients of a continued fraction representing these approximants and interpolants, or, equivalently, the coefficients in some recurrence connecting three consecutive terms of these objects are not considered here.

This paper should be viewed as a review/tutorial of existing work in both numeric and symbolic environments combined with a presentation of original results in the case of symbolic environments. The review/tutorial covers some recent methods in both numeric and symbolic environments along with details for some well known methods. These details serve to illustrate some of the modifications needed for effective computation in either environments. We point out that one of the big differences between mathematical, numeric or symbolic answers

is the form of normalization used in the final results. The original work presented in this paper centers on algorithms for the exact computation of rational interpolants. The first algorithm computes, in an efficient fraction-free way, a basis of *all* rational interpolants of a given type along an offdiagonal path of computation. This method is a modification of a specialization of the fraction-free algorithms found in [9]. The second method follows the ideas of [6] and uses a look–ahead strategy to compute a given rational interpolant along an arbitrary path of computation. Along an antidiagonal path the look–ahead recurrence is closely related to the well known Subresultant PRS algorithm for polynomial GCD computation.

All the problems considered in this paper have a common theme, that of using an associated linear algebraic formulation of the given problems along with techniques for taking advantage of the special structure of the coefficient matrices of these linear systems. In the case of rational approximation or greatest common divisors (one of the topics illustrated for exact computation) the coefficient matrix is either a Hankel, Toeplitz or Sylvester matrix. In the case of rational interpolation, the corresponding linear system has a Vandermonde or paired Vandermonde form. These are all special cases of striped Krylov matrices defined using linear functionals having an associated *special rule* as given in [9]. Determining fraction-free methods for the case of this general striped Krylov matrix has already been done in [9], while we expect that the look–ahead algorithm for arbitrary paths of computation presented in this paper can also be extended to a more general setting.

The paper is organized as follows. Section 2 considers the problem of numerical computation of rational approximants and interpolants. The section is divided into two subsections, with Section 2.1 giving a review of recent numerical methods and Section 2.2 giving some added details of the workings of a particular numerical method, the Cabay–Meleshko algorithm for computing a Padé approximant. Section 3 considers computation in exact arithmetic environments. It gives a review of current work and illustrates methods for reducing coefficient growth in the case of solving linear systems and in the case of Euclidean-like GCD computations. Section 4 considers the problem of exact computation of a particular problem, that of determining a rational interpolant. The methods presented are new. Two algorithms are presented, one which finds all rational interpolants of a given type along a specific offdiagonal path of computation and the other determines solutions to the rational interpolation problem along arbitrary paths of computation. The first approach is interesting because it makes use of well known recurrences used in a non-typical way. The second approach is a look–

ahead method which uses modified Schur complements. Both methods reduce coefficient growth with a minimum of cost. The final section includes a discussion of further work and areas of research.

2. Numeric Computation

In the case where a rational approximation or interpolation problem has a domain of coefficients that is inexact, there are a number of important problems and issues, the primary one being the inability to determine when a numerical quantity is zero. In such cases there are a number of approaches that can be used. In the case of rational approximation or interpolation problems, a typical approach is to use information from an associated linear system of equations, for example the condition number of the system or a collection of subsystems. Techniques include simple heuristics such as not solving ill–conditioned subproblems. Look–ahead methods jump over ill–conditioned subproblems but encounter the problem of determining when a given problem is in fact ill–conditioned. In the case of numerical computation of rational problems one also encounters the question of providing a formal analysis of a particular method using either a forward or backward error analysis. In many cases an error analysis is only sketched.

2.1. Historical Notes

In this subsection we give a brief discussion of some of the modern approaches used for computing rational approximants and interpolants in numerical domains. The discussion is by no means exhaustive, with the purpose only being to give an idea of some of the more modern approaches used to handle this difficult domain.

In 1993, Cabay and Meleshko [19] presented a look–ahead algorithm for the computation of numerical Padé approximants along with a complete forward error analysis. The algorithm, described in the following subsection, proceeds by jumping from one stable subproblem to the next stable subproblem along a diagonal path in the Padé table. The individual subproblems are solved using Gaussian elimination with pivoting. The determination of a stable or well conditioned subproblem was done by using an estimation of the condition number of the associated Hankel linear system using inversion formulae for such matrices. The complexity of the algorithm depends on the size of the largest jump needed for the look–ahead process. In the case of small jumps the algorithm requires $\mathcal{O}(n^2)$ operations. However, in the case of very large jump sizes, the algorithm could require as many as $\mathcal{O}(n^4)$ operations.

The approach used by Cabay and Meleshko was generalized in many different directions. Cabay, Jones and Labahn [17] presented a generalization for the case of Hermite-Padé and simultaneous Padé approximation, again using an offdiagonal path in the corresponding Padé tables. Beckermann [5] used a similar look–ahead method for the computation of orthogonal polynomials, except that the small subproblems for the look–ahead steps were solved using QR decomposition rather than Gaussian elimination. The use of the QR decomposition for intermediate problems implied that the algorithm would always compute in $\mathcal{O}(n^3)$ in the worst case.

Gutknecht [33] and Gutknecht and Hochbruck [35] presented new numerical algorithms for Padé approximants which were based on Toeplitz rather than Hankel linear systems. This allowed for computation along rows of the Padé table where the singularity and stability structure was often better known. The algorithms had superfast complexity of $\mathcal{O}(n \log^2 n)$ operations but required that all points in the computational path be well conditioned. Van Barel and Bultheel [49] presented a look–ahead algorithm for the solution of block Toeplitz systems, again providing an algorithm for computation along row paths. The authors also provided a complete forward error analysis. Beckermann and Labahn [10, 11] used a look–ahead procedure based on jumping over ill conditioned subproblems of a Sylvester matrix to efficiently determine when two numeric polynomials are relatively prime.

Additional methods in numeric environments include the work of Freund and Zha [26, 27] for orthogonal polynomials, and Bojanczyk, Brent and de Hoog [13, 14] for fast QR factorization of Toeplitz matrices. One interesting method based on displacement rank matrices is given by Gohberg, Kailath and Olshevski [29] (see also [20, 30, 32]). This approach uses FFTs to converts a Toeplitz problem into a Cauchy problem. The Cauchy problem can be solved in a numerically stable way by using pivoting — essentially the ability to reorder the interpolation Fourier points. The solution to the Cauchy problem is converted into a solution to the original problem by again using FFTs. The transformation of (block) Hankel matrices via FFT was also considered by Kravanja and Van Barel [39, 40]. They recover (block) Loewner matrices and solve the corresponding system via rational interpolation on the unit circle.

Finally we mention that there is a close link between look-ahead methods for Hankel systems and look-ahead procedures for Lanzcos type algorithms. To our knowledge, there is no precise error analysis for the latter class of methods. In fact, though a polynomial language is quite suitable for describing the mechanism of the jumps, it seems that the use of the basis of monomials (or another fixed basis of polynomials

like Chebyshev polynomials) does not allow one to derive sharp error bounds for Lanzcos type methods.

2.2. Example: Cabay–Meleshko Revised

In this subsection we give some details of the Cabay and Meleshko [19] algorithm. This illustrates one method of using look–ahead to solve numeric problems. It is particularly interesting for its use of a single numerical parameter, determined by studying inversion formula for the coefficient matrices of the associated linear system.

The Cabay–Meleshko algorithm is used for computing a Padé approximant of type (n, n) for a given integer n in a numeric environment. That is, for a given power series $f(x)$ having floating point coefficients and an integer n the algorithm looks for polynomials $p_n(x)$ and $q_n(x)$ satisfying

$$\deg p_n(x) < n, \quad \deg q_n(x) \leq n, \quad f(x)q_n(x) - p_n(x) = \mathcal{O}(x^{2n})_{x\to 0}.$$

In the case of numeric computation the Cabay–Meleshko algorithm scales the input series so that $||f(x)|| \approx 1$ and also specifies the constraints $||p_n(x)|| + ||q_n(x)|| \approx 1$. Here $||\sum a_j x^j|| = \sum |a_j|$ for polynomials (we may assume that the power series input f is truncated).

The basic building block used by the Cabay–Meleshko algorithm is

$$\mathbf{U}_n(x) = \begin{bmatrix} p_n(x) & p_{n+1}(x) \\ q_n(x) & q_{n+1}(x) \end{bmatrix},$$

a pair of consecutive Padé forms in a matrix polynomial form. These matrix polynomials satisfy $\det \mathbf{U}_n(x) = \tau_n \cdot x^{2n}$ for a constant $|\tau_n| \leq 1$. A building block $\mathbf{U}_n(x)$ is called *stable* if $|\tau_n|$ is not "too small" this being specified with a user input tolerance. If $\mathbf{U}_n(x)$ is stable then the algorithm iterates by constructing the "next" stable $\mathbf{U}_{n+s}(x)$. For $s = 1, 2, ...$, candidates are obtained by forming the product $\mathbf{U}_n(x) \cdot \hat{\mathbf{U}}_s(x)$ and then normalizing so that columns have magnitude 1. Here $\hat{\mathbf{U}}_s(x)$ is a building block of index s for the residual power series $r_n(x)r_{n+1}(x)^{-1}$, where $r_n(x) := x^{-2n}[f(x)q_n(x) - p_n(x)]$. This $\hat{\mathbf{U}}_s(x)$ is computed as a solution of a "small" system of equations by some stable method (which does not take into account the structure).

The reason that the algorithm works has do to with estimation of condition numbers of the Hankel coefficient matrices

$$H_n = \begin{bmatrix} f_0 & f_1 & \cdots & f_n \\ f_1 & f_2 & \cdots & f_{n+1} \\ \vdots & \vdots & \vdots & \vdots \\ f_n & f_{n+1} & \cdots & f_{2n} \end{bmatrix}$$

of the associated linear systems. The norm of this matrix is related to the scaling of the finite section $f^{(2n)}(x) = f_0 + ... + f_{2n}x^{2n}$ of the initial power series; its inverse may be expressed in terms of the coefficients of $\mathbf{U}_n(x)$ by means of an inversion formula from Heinig and Rost [37]. This leads to the following bounds for the 1–Hölder norm [5, Theorem 1]

$$\frac{1}{4\sqrt{|\tau_n|}} \frac{||f^{(2n)}||}{\sqrt{||f^{(2n+1)}||}} \leq \mathrm{cond}_1(H_n) \leq \frac{||f^{(2n)}||}{|\tau_n|}.$$

In other words, we are constructing a solution $\mathbf{U}_n(x)$ using only well-conditioned subproblems (with sufficiently large $|\tau_n|$).

Of course, this latter argument is only heuristic if one wants to validate a numerical algorithm. To be more precise, one has to estimate the growth of the error vector η_n consisting of the coefficients of the undesired powers $x^0, ..., x^{2n-1}$ in the expression $f(x)q_n(x) - p_n(x)$, where now $(p_n(x), q_n(x))$ denotes the computed counterpart of our desired Padé form. However, for sufficiently small errors η_n, η_{n+1}, we may still estimate the condition number using numerical building blocks $\mathbf{U}_n(x)$ [5, Theorem 1]. Moreover, denoting as before the leading coefficient of $\det \mathbf{U}_n(x)$ by τ_n, Cabay and Meleshko in their forward error analysis [19] (see also [5]) show that by using only stable numerical building blocks one may bound the norms of the error vector: if $\mathbf{U}_N(x)$ is the final block and $0 = n_0 < n_1 < ... < n_S \leq N$ the indices of intermediate stable numerical building blocks, then $||\eta_N|| + ||\eta_{N+1}||$ is bounded by a sum of products of the form $||\mathbf{U}_{n_j}(z)^{-1}\mathbf{U}_N(z)||$ times some small local error occurring in the jth step. Since $||\mathbf{U}_{n_j}(z)^{-1}\mathbf{U}_N(z)|| \leq 2/|\tau_n|$ in exact arithmetic and about the same size in a numerical setting, the growth of the error vectors is kept under control.

We remark that the typical normalization of an $(n-1, n)$ Padé approximant requires that the denominator $q_n(x)$ be either monic or has 1 as the constant coefficient. In the Cabay–Meleshko algorithm a different normalization is required, namely that $||p_n(x)|| + ||q_n(x)|| = 1$. This change in normalization also comes up in Section 4 where one wishes to efficiently compute rational interpolants without forming fractions.

3. Symbolic Computation

Computer algebra systems have the basic property that they allow a user to manipulate with symbolic and numeric data. In the case of symbolic manipulation, the natural domain of computation is a quotient field $\mathbb{Q}(a_1, \ldots, a_n)$ of symbols and exact rationals or else an algebraic extension of such a domain. Clearly arithmetic in a polynomial domain

such as $\mathbb{Q}[a_1, \ldots, a_n]$ is easier than arithmetic in its quotient domain. For example, the simple operation of addition

$$\frac{a}{b} + \frac{c}{d} = \frac{a \cdot d + b \cdot c}{b \cdot d}$$

requires one to normalize by removing greatest common divisors at every step if one wants to do such operations as recognizing zero, a fundamental concern in most computational algorithms.

The greatest common divisor computation in $\mathbb{Q}[a_1, \ldots, a_n]$ required for the addition of rational functions is a hidden cost for such an operation. In order to avoid such hidden costs one typically attempts to keep all computations inside the polynomial domain, whenever possible. In the next two subsections we discuss how one overcomes such a hidden cost in the case of such classical computations as Gaussian elimination and Euclidean remainder sequences for GCD computation.

3.1. Fraction-free Gaussian Elimination

One can use classical Gaussian elimination to illustrate some of the problems and some of the alternatives that are possible when one wishes to avoid quotient arithmetic.

It is easy to see that one can use a process of cross multiplication to eliminate nonzero terms from a given column in the reduction

$$A = \begin{bmatrix} a & b & c & \cdots & \cdots \\ d & e & f & \cdots & \cdots \\ g & h & i & \cdots & \cdots \\ \vdots & \vdots & \vdots & & \end{bmatrix} \approx \begin{bmatrix} a & b & c & \cdots & \cdots \\ 0 & \tilde{e} & \tilde{f} & \cdots & \cdots \\ 0 & \tilde{h} & \tilde{i} & \cdots & \cdots \\ \vdots & \vdots & \vdots & & \end{bmatrix}.$$

While this avoids forming quotients it also potentially doubles the size of the entries in the remaining reduced matrix with each cross multiplication reduction process. This results in an exponential growth of coefficients giving a complexity of $\mathcal{O}(2^n \cdot N^2)$ in the $n \times n$ case where N is a bound for the size of the entries of the matrix.

Bareiss [3] observed that one has a common divisor after 2 steps of the reduction, namely

$$A \approx \begin{bmatrix} a & b & c & \cdots & \cdots \\ 0 & \tilde{e} & \tilde{f} & \cdots & \cdots \\ 0 & 0 & a(..) & \cdots & a(...) \\ \vdots & \vdots & \vdots & & \\ 0 & 0 & a(..) & \cdots & a(...) \end{bmatrix}.$$

Thus one can divide out by a known common factor - the pivot used in the reduction 2 steps before - and continue the elimination process.

This simple observation results in a linear rather than exponential growth of coefficients. The result is an algorithm, *fraction-free Gaussian elimination*, which triangulates a matrix in $\mathcal{O}(n^5 \cdot N^2)$ operations. In fact Bareiss also presented a multi-step version of this process (see also [28]). An important fact to point out is that fraction-free Gaussian elimination gives the Cramer solution of a linear problem, that is the solution determined by using Cramer's rule. This is fundamental to the observations of Section 4.

3.2. Euclidean-like GCD Computation

Consider now the problem of computing a greatest common divisor of a pair of polynomials in $\mathbb{D}[x]$ where $\mathbb{D}$ is an integral domain rather than a field. We again wish to compute such a greatest common divisor using only operations in $\mathbb{D}[x]$, that is, without going to a quotient field.

As an example, consider the two polynomials

$$\begin{aligned} a(x) &= x^8 + x^6 - 3x^4 - 3x^3 + 8x^2 + 2x - 5 \\ b(x) &= 3x^6 + 5x^4 - 4x^2 - 9x + 21 \end{aligned}$$

originally used as an example by Knuth [38]. Let $R_0(x) = a(x)$ and $R_1(x) = b(x)$. The five examples below illustrate various ways of computing a polynomial greatest common divisor via a "polynomial remainder sequence "process.

(i) The *Euclidean Algorithm* over $\mathbb{Q}[x]$. In this case the remainders are

$$\begin{aligned} R_2(x) &= x^4 - \tfrac{1}{5}x^2 + \tfrac{3}{5} \\ R_3(x) &= x^2 + \tfrac{25}{13}x - \tfrac{49}{13} \\ R_4(x) &= x - \tfrac{6150}{4663} \\ R_5(x) &= 1 \end{aligned}$$

(ii) A *Pseudo Euclidean Algorithm* over $\mathbb{Z}[x]$ based on pseudo-division but without removing common factors:

$$\begin{aligned} R_2(x) &= 15\,x^4 - 3\,x^2 + 9 \\ R_3(x) &= 15795\,x^2 + 30375\,x - 59535 \\ R_4(x) &= 1254542875143750\,x - 1654608338437500 \\ R_5(x) &= 12593338795500743100931141992187500 \end{aligned}$$

Pseudo-division can be viewed as a type of reduction which corresponds to the cross multiplication used in the previous section [28, 38].

(iii) The *Primitive Euclidean Algorithm* over $\mathbb{Z}[x]$ based on pseudo-division and removing common factors after every pseudo-division:

$$\begin{aligned} R_2(x) &= 5\,x^4 - x^2 + 3 \\ R_3(x) &= 13\,x^2 + 25\,x - 49 \\ R_4(x) &= 4663\,x - 6150 \\ R_5(x) &= 1 \end{aligned}$$

(iv) The *Reduced PRS Algorithm* over $\mathbb{Z}[x]$:

$$\begin{aligned} R_2(x) &= -15\,x^4 + 3\,x^2 - 9 \\ R_3(x) &= 585\,x^2 + 1125\,x - 2205 \\ R_4(x) &= -18885150\,x + 24907500 \\ R_5(x) &= 527933700 \end{aligned}$$

(v) The *Subresultant PRS Algorithm* over $\mathbb{Z}[x]$:

$$\begin{aligned} R_2(x) &= 15\,x^4 - 3\,x^2 + 9 \\ R_3(x) &= 65\,x^2 + 125\,x - 245 \\ R_4(x) &= 9326\,x - 12300 \\ R_5(x) &= 260708 \end{aligned}$$

The mathematical descriptions of these algorithms for polynomials from $\mathbb{D}[x]$ with $\mathbb{D}$ an integral domain is given using the polynomial remainder sequences defined by $R_0(x) = a(x)$, $R_1(x) = b(x)$ and

$$\alpha_i \cdot R_{i-1}(x) = Q_i(x) \cdot R_i(x) + \beta_i \cdot R_{i+1}(x) \text{ with } \alpha_i, \beta_i \in \mathbb{D}\,.$$

Let r_i denote the leading coefficient of the i-th remainder $R_i(x)$ and $\delta_i = \deg R_{i-1}(x) - \deg R_i(x)$. Then the five algorithms mentioned previously are given by

(i) Classical Euclidean: $\alpha_i = 1, \beta_i = 1$.

(ii) Pseudo-Eucidean using only pseudo-division: $\alpha_i = r_i^{\delta_i+1}, \beta_i = 1$.

(iii) Primitive Eucidean: $\alpha_i = r_i^{\delta_i+1}, \beta_i = \text{ content}(R_{i+1}(x))$ where the content of a polynomial is the greatest common divisor of the coefficients of the polynomial.

(iv) Reduced PRS algorithm: $\alpha_i = r_i^{\delta_i+1}, \beta_i = \alpha_{i-1}$ where $\beta_1 = 1$.

(v) Subresultant PRS algorithm: $\alpha_i = r_i^{\delta_i+1}, \beta_i = -r_{i-1} \cdot \phi_i^{\delta_i}$ with $\beta_1 = (-1)^{\delta_i+1}$, $\phi_1 = 1$ and $\phi_i = (-r_{i-1})^{\delta_{i-1}} \cdot \phi_{i-1}^{1-\delta_{i-1}}$ for $i \geq 2$.

Example (i) is the classical Euclidean algorithm which works over a field and hence uses rational arithmetic, something which we have wanted to avoid. Example (ii) avoids rational arithmetic but encounters exponential coefficient growth making it unacceptable. Example (iii) uses only polynomial operations and has a minimal coefficient growth. However it accomplishes this by using greatest common divisor calculations making it equivalent to the classic Euclidean algorithm for our purposes.

The remaining two methods - the Reduced PRS algorithm of Collins [21] and the Subresultant PRS algorithm of Brown and Collins [16, 21] both use only polynomial operations and have moderate coefficient growth. While the coefficient growth is not minimal it does have the advantage that the cost to reduce coefficient growth is minimal, namely a simple division by a known divisor, exactly the process followed in fraction-free Gaussian elimination. Further details on these algorithms can be found in [1, 28, 45].

In the case of fraction-free Gaussian elimination there is a simple mechanism for determining a divisor at a later stage, at least in the single step case. However, one can see by the previous examples and accompanying mathematical formulae that determining divisors of remainders is a nontrivial task. The key observation that we wish to note is that the Subresultant PRS algorithm, with each subresultant viewed as a solution of a linear system with a Trudi or Sylvster matrix of coefficients, is an iteration of Cramer solutions that ultimately result in the same solution as that obtained by fraction-free Gaussian elimination but in a manner that takes advantage of the special structure of the coefficient matrices.

In addition to the algorithms of Brown and Collins for polynomial GCD computations, other examples of note for fraction-free computations include the work of Sylvester [47] and Habicht [36] on Sturm sequences, Cabay and Kossowski [18] for Padé approximants and Akritas [1] for polynomial GCD computations using a matrix triangulation procedure that sometimes results in larger known divisors.

4. Symbolic Computation of Rational Interpolants

In this section we describe two methods for computing rational interpolants using only fraction-free computations. Let $\mathbb{D}$ be an integral domain, $\mathbb{Q}$ its quotient field and let $(x_i)_{i=0,1,\ldots}$ be a set of (distinct) interpolation points (knots) from $\mathbb{D}$.

DEFINITION 4.1. *Let f, g be functions such that $f(x_i), g(x_i) \in \mathbb{D}$ and $(f(x_i), g(x_i)) \neq (0,0)$ for all i. A rational interpolant of type (m,n)*

is a pair of polynomials $p(x), q(x)$ *from* $\mathbb{D}[x]$*, at least one nonzero, satisfying*

$$deg\ p(x) \leq m,\ deg\ q(x) \leq n \tag{1}$$

$$f(x_i) \cdot p(x_i) + g(x_i) \cdot q(x_i) = 0\ for\ i = 0, \ldots, m+n. \tag{2}$$

Recursive algorithms for computing rational interpolants along a path in the solution table with coefficients in a field have been given by a number of authors, see for instance [2, 7, 24, 31, 34, 48, 51, 52]. Our goal in this section is to describe efficient methods without leaving the domain $\mathbb{D}[x]$.

Notice that a solution of our interpolation problem always exists, but is not necessarily unique. However, as seen for instance from [22, 23] or [48], there exists always a so–called "minimal" solution. This minimal solution is unique up to a constant from $\mathbb{Q}$ and cannot be further reduced by dividing both $p(x)$ and $q(x)$ by a nontrivial polynomial from $\mathbb{Q}[x]$. In case of singularities, we will determine this particular solution.

For a given pair (m, n) define

$$K_{m,n,k} = \left[\begin{array}{ccc|ccc} f(x_0) & \cdots & x_0{}^{m-1} \cdot f(x_0) & g(x_0) & \cdots & x_0{}^{n-1} \cdot g(x_0) \\ \vdots & & \vdots & \vdots & & \vdots \\ f(x_k) & \cdots & x_k{}^{m-1} \cdot f(x_k) & g(x_k) & \cdots & x_k{}^{n-1} \cdot g(x_k) \end{array}\right] \tag{3}$$

a paired Vandermonde matrix of size $(k+1) \times (m+n)$. The rational interpolation equation (2) is equivalent to solving the linear system of equations

$$K_{m+1,n+1,m+n} \cdot [p_0, \ldots, p_m, q_0, \ldots, q_n]^T = 0. \tag{4}$$

We can obtain a fraction-free solution to the system (4) by using fraction-free Gaussian elimination. However such an approach does not take into consideration the special structure of the matrix $K_{m,n,m+n}$. Our goal is to obtain the same solution as found via fraction-free Gaussian elimination but using a more efficient procedure. This procedure will avoid fractions but take advantage of the special structure of the coefficient matrix of the linear system.

If we set $K_{m,n} = K_{m,n,m+n-1}$ and $d_{m,n} = \det K_{m,n}$, then Cramer's rule produces a solution with coefficients in $\mathbb{D}$ to the linear system

$$K_{m+1,n+1} \cdot [p_0, \ldots, p_m, q_0, \ldots, q_n]^T = [0, \ldots, 0, d_{m+1,n+1}]^T \tag{5}$$

which we call the *Cramer solution*. This solution may be written down in terms of determinants. Indeed, if

$$p_{m,n}(x) = \det \left[\begin{array}{c} K_{m+1,n+1,m+n} \\ \hline 1, \ldots \quad , x^m | 0 \quad \cdots 0 \end{array} \right], \tag{6}$$

$$q_{m,n}(x) = \det \left[\begin{array}{c} K_{m+1,n+1,m+n} \\ \hline 0 \cdots \quad 0 | 1, \quad \ldots, x^n \end{array} \right] \tag{7}$$

and

$$r_{m,n}(x) = \det \left[\begin{array}{c} K_{m+1,n+1,m+n} \\ \hline f(x) \cdots \quad x^m f(x) | g(x), \quad \ldots, x^n g(x) \end{array} \right] \tag{8}$$

then

$$f(x) \cdot p_{m,n}(x) + g(x) \cdot q_{m,n}(x) = r_{m,n}(x) \tag{9}$$

and $r_{m,n}(x_i) = 0$ for $i = 0, \ldots, m+n$ with $r_{m,n}(x_{m+n+1}) = d_{m+1,n+1}$. Thus $p_{m,n}(x)$ and $q_{m,n}(x)$ define the Cramer solution for the linear system (5). Using the determinantal representations (6) and (7) we see that the m-th coefficient of $p_{m,n}(x)$ is $(-1)^{n+1} d_{m,n+1}$ and the n-th coefficient of $q_{m,n}(x)$ is $d_{m+1,n}$.

Notice that, while computing Cramer solutions, we get for free the determinant of paired Vandermonde and thus of Toeplitz matrices [41].

REMARK 4.2. *Cramer solutions enable us to find the particular "minimal" solution [22, 23] mentioned after Definition 4.1. For suppose that the Cramer solution $p_{m,n}(x)$ and $q_{m,n}(x)$ for the linear system (5) is not identically zero. Then at least one minor of size $m+n+1$ of $K_{m+1,n+1,m+n}$ is nontrivial, showing that (up to multiplication with an element of $\mathbb{Q}$) there is a unique solution for the index (m,n). In particular, the pair $p_{m,n}(x)$, $q_{m,n}(x)$ cannot be reducible since after division of $p_{m,n}(x)$ and $q_{m,n}(x)$ by some common factor we necessarily have lost one interpolation condition.*

Consequently, for any indices m', n' with $\deg p_{m,n}(x) \leq m'$, $\deg q_{m,n}(x) \leq n'$, $m'+n' \geq m+n$, and $f(x_i) \cdot p(x_i) + g(x_i) \cdot q(x_i) = 0$, $i = 0, \ldots, m'+n'$, the couple $p_{m,n}(x)$, $q_{m,n}(x)$ constitutes a minimal solution of index (m', n').

We say that (m,n) is a *normal point* if the determinant $d_{m,n} \neq 0$. The fraction-free methods presented in this section use *Mahler Systems* [9, 6] as the basic building blocks for their recursions. These systems have been considered already by Mahler [44] in his study of Hermite-Padé approximants. However, he chooses a different normalization and supposes that all points are normal.

DEFINITION 4.3. *[9, 6] A Mahler system of type* (m,n) *is a* 2×2 *matrix polynomial*

$$\mathbf{M}_{m,n}(x) = \begin{bmatrix} (-1)^n p_{m,n-1}(x) & p_{m-1,n}(x) \\ (-1)^n q_{m,n-1}(x) & q_{m-1,n}(x) \end{bmatrix}. \tag{10}$$

One can verify using the well-known block structure of the table of rational interpolants that $\det\ \mathbf{M}_{m,n}(x) \neq 0$ iff (m,n) is normal. Thus in the sequel we will only consider Mahler systems at normal points. Clearly both columns of $\mathbf{M}_{m,n}(x)$ verify the interpolation condition at the knots $x_0, ..., x_{m+n-1}$. Furthermore, the entries of $\mathbf{M}_{m,n}(x)$ satisfy the degree constraints

$$\begin{bmatrix} = m & < m \\ < n & = n \end{bmatrix}$$

with a common leading coefficient (namely $d_{m,n}$) of the entries on the diagonal (the desire for the same leading coefficient explains the need for the sign $(-1)^n$ used in the first column). Finally, in the normal case a Mahler system is unique up to multiplication from $\mathbb{Q}$, that is, any 2×2 matrix polynomial satisfying the same interpolation conditions and degree constraints coincides with $c \cdot \mathbf{M}_{m,n}(x)$ for some $c \in \mathbb{Q}$. For further properties and proofs we refer the reader to [9, Section 5].

The aim of the following subsections is to show how to compute from a given Mahler system at a normal point (m,n)

$$\begin{bmatrix} p(x) & u(x) \\ q(x) & v(x) \end{bmatrix} = \epsilon \cdot \mathbf{M}_{m,n}(x), \quad \epsilon \in \{\pm 1\}, \tag{11}$$

to a Mahler system at a "neighboring" normal point (m^*, n^*)

$$\begin{bmatrix} p^*(x) & u^*(x) \\ q^*(x) & v^*(x) \end{bmatrix} = \epsilon^* \cdot \mathbf{M}_{m^*,n^*}(x), \quad \epsilon^* \in \{\pm 1\} \tag{12}$$

in a way that does not introduce fractions. We remark that the freedom of the sign will allow us to obtain some major simplifications. Notice that, starting from the normal point $(0,0)$ (with the identity as corresponding Mahler system), the above recursion completely specifies an algorithm following a particular path in the two-dimensional solution table.

4.1. Computation of Mahler Systems

In this section we want to follow a "ascending" path $(\vec{n}_k)_{k=0,1,\ldots}$ of normal indices, $\vec{n}_0 = (0,0)$, where two succeeding points $\vec{n}_k = (m,n)$ and $\vec{n}_{k+1} = (m^*,n^*)$ satisfy

$$m^* \geq m, \quad n^* \geq n, \quad m^* + n^* = m + n + 1.$$

THEOREM 4.4. *Let (m,n) be normal, with its corresponding Mahler system given by (11). Furthermore, let*

$$\begin{aligned} \hat{f}_0 &= f(x_{m+n}) \cdot p(x_{m+n}) + g(x_{m+n}) \cdot q(x_{m+n}), \\ \hat{g}_0 &= f(x_{m+n}) \cdot u(x_{m+n}) + g(x_{m+n}) \cdot v(x_{m+n}), \end{aligned} \tag{13}$$

and denote here (and in the sequel) $u(x) = \sum u_j x^j$ (and similarly for the other polynomials involved in (11), (12)).

(a) *We have $\hat{f}_0 \neq 0$ iff $(m^*,n^*) = (m+1,n)$ is normal. In this case, the columns in (12) with $\epsilon^* = (-1)^n \cdot \epsilon$ are computed via*

$$\begin{aligned} v_n \cdot \begin{bmatrix} u^*(x) \\ v^*(x) \end{bmatrix} &= \hat{f}_0 \cdot \begin{bmatrix} u(x) \\ v(x) \end{bmatrix} - \hat{g}_0 \cdot \begin{bmatrix} p(x) \\ q(x) \end{bmatrix} \\ v_n \cdot \begin{bmatrix} p^*(x) \\ q^*(x) \end{bmatrix} &= \hat{f}_0 \cdot (x - x_{m+n}) \cdot \begin{bmatrix} p(x) \\ q(x) \end{bmatrix} - q_{n-1} \cdot \begin{bmatrix} u^*(x) \\ v^*(x) \end{bmatrix}, \end{aligned}$$

and thus $u^*_{m^*} = v^*_{n^*} = \hat{f}_0 = \epsilon^* \cdot d_{m+1,n}$.

(b) *We have $\hat{g}_0 \neq 0$ iff $(m^*,n^*) = (m,n+1)$ is normal. In this case, the columns in (12) with $\epsilon^* = \epsilon$ are computed via*

$$\begin{aligned} v_n \cdot \begin{bmatrix} p^*(x) \\ q^*(x) \end{bmatrix} &= \hat{g}_0 \cdot \begin{bmatrix} p(x) \\ q(x) \end{bmatrix} - \hat{f}_0 \cdot \begin{bmatrix} u(x) \\ v(x) \end{bmatrix} \\ v_n \cdot \begin{bmatrix} u^*(x) \\ v^*(x) \end{bmatrix} &= \hat{g}_0 \cdot (x - x_{m+n}) \cdot \begin{bmatrix} u(x) \\ v(x) \end{bmatrix} - u_{m-1} \cdot \begin{bmatrix} p^*(x) \\ q^*(x) \end{bmatrix}, \end{aligned}$$

and thus $u^*_{m^*} = v^*_{n^*} = \hat{g}_0 = \epsilon^* \cdot d_{m,n+1}$.

Proof: See [9, Theorem 6.1]. □

In fact, the identities of Theorem 4.4 are two of the four well-known Frobenius identities in the table of rational interpolants, see [24, 51] or [7, Theorem 1]. Though often not stated in terms of determinants, a verification of the above identities is immediate by checking the degrees and interpolation constraints, and by using the above mentioned determinantal representation of $u_n, p_m, \hat{f}_0, \hat{g}_0$. Notice also that each of the first of the two coupled recurrences may be shown using Sylvester's identity — precisely as in the fraction-free Gaussian elimination procedure mentioned earlier. The second recurrence, however, depends on

the special structure of the coefficient matrix and allows for a significant speed-up.

Theorem 4.4 gives us the identities that we need to compute Mahler systems in a way that avoids fractions. Namely, in each case we first form the right hand side and then divide out by the known constant v_n. At each step we compute the components of a Mahler system. Since the entries of each Mahler system have a representation as a determinantal polynomial or equivalently as a Cramer's solution, the growth of the coefficients in the entries is bounded by Hadamard's inequality. In fact the algorithm produces a linear growth in the coefficient size of the entries. Note that this restricted growth of coefficients involves no greatest common divisor coefficient operations. We only need divide out by a known common multiple.

When all the points $(\vec{n}_k)_{k=0,1,\ldots,n}$ are normal, we can therefore apply part (a) for the case where $\vec{n}_{k+1} = \vec{n}_k + (1,0)$, and part (b) otherwise. The resulting algorithm is a special case of FFFGnormal of [9, Section 6]; we omit the details. However we recall from [9, Theorem 6.3] that the complexity of this algorithm is then $\mathcal{O}(n^4 \cdot N^2)$, where N is a bound on the coefficients of the input matrix. Thus we gain a factor n in comparison with fraction-free Gaussian elimination.

4.2. Nonnormality and Minimal Row Pivoting

In this subsection we consider the general case where some of the points on our staircase are singular. Since the interpolation problem of Definition 4.1 does not involve derivatives, we have at our disposal a permutation of the knots (which is no longer true for problems involving derivatives such as Newton-Padé or Padé type problems[1]). Such a (row) pivoting has been applied successfully by Werner [52] and others in the context of Thiele continued fractions. In fact, Graves-Morris [31] showed that a pivoting similar to partial row pivoting in Gaussian elimination leads to a forward stable numerical algorithm. Knot pivoting has also been applied successfully in [39, 40, 41].

To describe more precisely the necessary changes of the method in the preceding section, we take a closer look at $\vec{n}_{k+1} = (m^*, n^*) = \vec{n}_k + (1,0) = (m+1, n)$ (the other case is similar). Denote by $\gamma_k \geq m+n-1$ the largest index from the knots which were involved in the computation of $\mathbf{M}_{\vec{n}_k}(x)$ and its predecessors ($\gamma_0 = -1$). In order to be able to apply a relation similar to Theorem 4.4(a), we need to next treat

[1] In the more general setting of [9], one may apply row pivoting following the general lines described below exactly in the case of diagonal $\mathbf{C}$. Such a pivoting is in fact a useful tool in algorithms for matrices having a small displacement rank [29, 30].

a suitable point where the residual of the first column does not vanish. On the other hand, we only want to apply pivoting if necessary in order to keep track of the original rational interpolants corresponding to our staircase.

Therefore, let $\hat{f}(x) = f(x) \cdot p(x) + g(x) \cdot q(x)$, and denote by κ the smallest index i such that $\hat{f}(x_i) \neq 0$. We then apply Theorem 4.4(a) with

$$\begin{aligned} \hat{f}_0 &= f(x_\kappa) \cdot p(x_\kappa) + g(x_\kappa) \cdot q(x_\kappa), \\ \hat{g}_0 &= f(x_\kappa) \cdot u(x_\kappa) + g(x_\kappa) \cdot v(x_\kappa), \end{aligned}$$

instead of (13). Furthermore, $\gamma_{k+1} = \max\{\gamma_k, \kappa\}$. Such a "minimal" knot pivoting enables us to find all minimal solutions connected to our staircase. In fact, it follows from Remark 4.2 that, provided that $\gamma_{k+1} > \gamma_k$, a minimal solution for the index $\vec{n}_j - (1,1)$, $j = \gamma_k+1, \gamma_k+2, ..., \gamma_{k+1}$ is given by $(p(x), q(x))$.

4.3. Look-around and Minimal Column Pivoting

Though knot (or row) pivoting can be applied quite successfully for rational interpolation in the presence of singularities, it fails for similar approximation problems involving derivatives, for example problems such as Padé approximation. The aim of this subsection is to show that, for offdiagonal paths, we may instead apply a kind of minimal column pivoting. In order to keep track of the special structure of the corresponding linear system, this column pivoting should be done in a way that the occurring minors will all be paired Vandermonde matrices.

Roughly speaking, we make a small detour around singular blocks in the solution table while staying as close as possible to the desired offdiagonal path. This will enable us to recover the subsequence of normal points of our off-diagonal path, and to compute minimal solutions for all points of our path. For readers familiar with the block structure in the table of rational interpolants (overlapping squares with common main diagonal, see [23]), the principle of our *look–around* algorithm is probably best understood by looking at Figure 1.

In order to be more precise, we start by defining the offdiagonal path induced by a point (m', n')

$$\vec{n}_k = \begin{cases} (k, 0) & \text{for } k = 0, ..., m' - n' + 1 \text{ (if } m' \geq n'), \\ (0, k) & \text{for } k = 0, ..., n' - m' \text{ (if } m' < n'), \\ (m' - n' + j, j) & \text{for } k = m' - n' + 2j,\ j \geq \max\{0, n' - m'\}, \\ (m' - n' + j + 1, j) & \text{for } k = m' - n' + 2j + 1,\ j \geq \max\{0, n' - m'\}. \end{cases} \tag{14}$$

Then the closest normal point of the form $\vec{\nu}_k = \vec{n}_k + (t_k, -t_k)$ with an integer t_k is uniquely defined by the requirements that (compare [9,

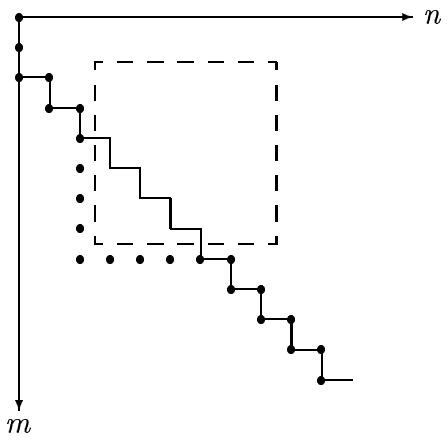

Figure 1. An example of singular rational interpolation. We have drawn the corresponding C–table of Bigradients, that is, the table having (m,n) entry the determinant $d_{m,n}$. Here the dashed square indicates a singular block of zero–entries. By a straight line we denote the offdiagonal path induced by $\vec{n}=(7,6)$, with the dots characterizing the path of computation used by our look–around method.

Theorem 7.3])

$$d_{\vec{\nu}_k} \neq 0, \text{ and } d_{\vec{n}_k+(t,-t)} = 0 \text{ for } |t| < |t_k|.$$

In the case where $|t_k| > 1$, $d_{\vec{n}_k-(t_k,-t_k)} \neq 0$ (where our offdiagonal path goes through the main diagonal of an interior block) and thus $\vec{n}_k \geq (|t_k|,|t_k|)$ we need to add the convention that $t_k > 0$ if $k-m'+n'$ is even and $t_k < 0$ if $k-m'+n'$ is odd.

Suppose that we have at our disposal the point $\vec{\nu}_k = (m,n)$ and the corresponding Mahler system $\pm\mathbf{M}_{\vec{\nu}_k}(x)$ as described in (11) (the algorithm takes the initializations $\vec{\nu}_0 = 0$, $\mathbf{M}_{\vec{\nu}_0}(x) = I_2$). Since minimal solutions along antidiagonals in singular blocks remain the same [23], the minimal solution for the point $\vec{n}_{k+1} - (1,1)$ is then given by $(p(z),q(z))$ provided that $m-n \leq m'-n'$, and by $(u(z),v(z))$ otherwise.

Define $(\hat{f}_0,\hat{g}_0)$ as in (13). It follows from the block structure (overlapping squares with common main diagonal) of the generalized C–table of determinants $d_{i,j}$, that, with $d_{m,n} \neq 0$, at least one of the quantities $d_{m+1,n} = \pm\hat{f}_0$ or $d_{m,n+1} = \pm\hat{g}_0$ is nonzero (for a formal proof see for example [4, Lemma 3.1]). Thus we may always continue our algorithm by choosing $\vec{\nu}_{k+1} = (m^*,n^*) \in \{(m+1,n),(m,n+1)\}$ and by applying

Theorem 4.4 in order to compute the new Mahler system. [2] Indeed, as shown in [9, Section 7], we have $\vec{\nu}_{k+1} = (m+1, n)$ iff

$$\hat{g}_0 = 0 \quad \text{or} \quad (\hat{f} \neq 0 \text{ and } m' - m \geq n' - n),$$

and otherwise $\vec{\nu}_{k+1} = (m, n+1)$. Roughly speaking, if $m' - m > n' - n$ then we actually are located above the offdiagonal path. Thus we would like to come closer to our offdiagonal path and thus take $\vec{\nu}_{k+1} = (m+1, n)$ if possible (i.e., if $\hat{f}_0 \neq 0$). For a concrete implementation of this look–around procedure we refer the reader to the algorithm FFFG given in [9, Section 7]. We remark here that the complexity of our procedure again is $\mathcal{O}(n^4 \cdot N^2)$.

4.4. Look-ahead and Modified Schur Complements

So far we have presented a fraction-free procedure that allows for computation of rational interpolants along a set of closest normal points to an off-diagonal path of computation. The method uses relationships between neighbors in the rational interpolation table in the normal case.

In this subsection we use a different method to obtain a second procedure, again fraction-free, for computing interpolants of a given type. The method is a look–ahead method for computing interpolants which allows for an arbitrary path $(\vec{n}_k)_{k=0,1,\ldots}$, $\vec{n}_0 = (0,0)$ of computation, with the only restriction for two succeeding points $\vec{n}_k = (m, n)$ and $\vec{n}_{k+1} = (m^*, n^*)$ being that

$$m^* \geq 0, \quad n^* \geq 0, \quad m^* + n^* \geq m + n.$$

Thus we may follow columns, rows, diagonal or anti-diagonal paths in the solution table. The method is based on the use of modified Schur complements of the associated linear system. It can be extended for general steps in the setting of [9] (for the special case of Matrix Padé approximation and diagonal paths see [6]). We can view the problem of rational interpolation as being parallel to rational approximation with the paired Vandermonde matrix being replaced by a Sylvester matrix. Using this view a computation along a path $(m+s, n+s)_{s=1,2,\ldots}$ becomes similar to a Hankel solver while along a path $(m+s, n)_{s=1,2,\ldots}$ is similar to a Toeplitz solver.

The principle of our algorithm is that, given a normal point $\vec{n}_k$, we scan the points $\vec{n}_j$, $j = k+1, k+2, \ldots$ in order to find the next normal point of the sequence. This requires us to describe two steps:

[2] The fact that $\vec{\nu}_{k+1}$ is a neighbor of $\vec{\nu}_k$ depends on the particular offdiagonal path chosen. It is no longer true for an arbitrary ascending path.

- How to decide that a point $(m^*, n^*) = \vec{n}_j$, $j > k$ is normal given that the point $(m, n) = \vec{n}_k$ is normal?

- In this case, how to compute the new Mahler system $\mathbf{M}^*(x)$ given by (12) in terms of the current Mahler system $\mathbf{M}(x)$ of (11)?

From the general principle of look–ahead methods it is not surprising that the answer for the first question is to check whether a certain "small" matrix $\hat{K}$ is nonsingular. In this case, we may apply fraction-free Gaussian elimination on $\hat{K}$ in order to find a certain 2×2 matrix polynomial

$$\hat{\mathbf{M}}(x) = \begin{bmatrix} \hat{p}(x) & \hat{u}(x) \\ \hat{q}(x) & \hat{v}(x) \end{bmatrix} \tag{15}$$

with coefficients in $\mathbb{D}$. The new Mahler system is then obtained by

$$c \cdot \mathbf{M}^*(x) = \mathbf{M}(x) \cdot \hat{\mathbf{M}}(x) \tag{16}$$

with a suitable $c \in \mathbb{D}$. Here the main difficulty is to predict in advance the common factor c.

Recurrence relations of the form (16) with coefficients in a field have been given by a number of authors, for instance see [8] for references. In the context of rational interpolation, Gutknecht showed in [34] that the columns of $\hat{\mathbf{M}}(x)$ are solutions of a multi-point Padé approximation problem.

In order to present more details, we assume that $s := n^* - n \geq m^* - m$ (the other case is covered by a permutation of f, g), and define $\tau^* := m^* + n^* - 1$, $\tau := m + n - 1$, $\sigma = s - (m^* - m) \geq 0$. Since $\tau^* \geq \tau$ by assumption, we get that $s \geq 1$, and $0 \leq \sigma \leq 2s$ ($\sigma = 0$ for a diagonal step and $\sigma = 2s$ for an antidiagonal step). We also define the residuals $\hat{f}(x) = f(x)p(x) + g(x)q(x)$ and $\hat{g}(x) = f(x)u(x) + g(x)v(x)$.

It is easy to check that finding $\mathbf{M}^*(x)$ is equivalent to solving

$$\left[\begin{array}{c|cc} K_{m+s+1,n+s+1,\tau^*} & & \\ \hline 0 \;\; I_{\sigma+1} & 0 & 0 \\ \hline 0 \quad 0 & 0 & I_1 \end{array}\right] \cdot \left[\begin{array}{cc} p_0^* & u_0^* \\ \vdots & \vdots \\ \hline p_{m+s}^* & u_{m+s}^* \\ \hline q_0^* & v_0^* \\ \vdots & \vdots \\ q_{m+s}^* & v_{m+s}^* \end{array}\right] = d_{m^*,n^*} \cdot \left[\begin{array}{cc|l} 0 & 0 & \\ \vdots & \vdots & \\ 0 & 0 & \\ 1 & 0 & \leftarrow \tau^* + 2 \\ 0 & 0 & \\ \vdots & \vdots & \\ 0 & 0 & \\ 0 & 1 & \leftarrow \tau^* + \sigma + 3 \end{array}\right]$$

Denote the square matrix of order $m + n + 2s + 2$ on the left by K^*, and define the square matrix $K = K_{m,n,\tau}$ of order $m + n$, the square

matrix $\hat{K}$ of order $2s+2$ with row blocks of size $(\tau^*-\tau, \sigma+1, 1)$ by

$$\left[\begin{array}{cccc|cccc}
\hat{f}(x_{\tau+1}) & x_{\tau+1}\cdot\hat{f}(x_{\tau+1}) & \cdots & x_{\tau+1}^s\cdot\hat{f}(x_{\tau+1}) & \hat{g}(x_{\tau+1}) & x_{\tau+1}\cdot\hat{g}(x_{\tau+1}) & \cdots & x_{\tau+1}^s\cdot\hat{g}(x_{\tau+1}) \\
\vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\
\hat{f}(x_{\tau^*}) & x_{\tau^*}\cdot\hat{f}(x_{\tau^*}) & \cdots & x_{\tau^*}^s\cdot\hat{f}(x_{\tau^*}) & \hat{g}(x_{\tau^*}) & x_{\tau^*}\cdot\hat{g}(x_{\tau^*}) & \cdots & x_{\tau^*}^s\cdot\hat{g}(x_{\tau^*}) \\
\hline
 & p_m & \cdots & p_{m-\sigma} & & u_m & \cdots & u_{m-\sigma} \\
 & & \ddots & \vdots & & & \ddots & \vdots \\
 & & & p_m & & & & u_m \\
\hline
 & & & q_n & & & & v_n
\end{array}\right]$$

as well as the square matrix of order $m+n+2s+2$ with row blocks of size $(m+s+1, n+s+1)$ and column blocks of size $(m, n, s+1, s+1)$ by

$$B = \left[\begin{array}{ccc|ccc}
 & & & p_0 & & & u_0 & & \\
I_m & 0 & & \vdots & \ddots & & \vdots & \ddots & \\
 & & & p_m & & p_0 & u_m & & u_0 \\
0 & 0 & & & \ddots & \vdots & & \ddots & \vdots \\
 & & & & & p_m & & & u_m \\
\hline
 & & & q_0 & & & v_0 & & \\
0 & I_n & & \vdots & \ddots & & \vdots & \ddots & \\
 & & & q_n & & q_0 & v_n & & v_0 \\
0 & 0 & & & \ddots & \vdots & & \ddots & \vdots \\
 & & & & & q_n & & & v_n
\end{array}\right]. \tag{17}$$

Note that $q_n = u_m = 0$ and $p_m = v_n = \pm d_{m,n}$ in the above matrices.

If $\hat{K}$ is nonsingular, we define in addition the entries of (15) by the system

$$\hat{K}\cdot\left[\begin{array}{cc}
\hat{p}_0 & \hat{u}_0 \\
\vdots & \vdots \\
\hat{p}_s & \hat{u}_s \\
\hline
\hat{q}_0 & \hat{v}_0 \\
\vdots & \vdots \\
\hat{q}_s & \hat{v}_s
\end{array}\right] = \det(\hat{K})\cdot\left[\begin{array}{cc|l}
0 & 0 & \\
\vdots & \vdots & \\
0 & 0 & \\
1 & 0 & \leftarrow \tau^*-\tau+1 \\
0 & 0 & \\
\vdots & \vdots & \\
0 & 0 & \\
0 & 1 & \leftarrow \tau^*-\tau+\sigma+2
\end{array}\right] \tag{18}$$

THEOREM 4.5. (Modified Schur Complement)
Let (m,n) be a normal point. Then

$$det(K^*)\cdot[d_{m,n}]^{2s+1} = \pm det(\hat{K}).$$

In particular, (m^*, n^*) is a normal point iff $\hat{K}$ is nonsingular. In the latter case, solving system (18) by fraction-free Gaussian elimination, we obtain $\hat{\mathbf{M}}(x)$ as described in (15), and the new Mahler system $\mathbf{M}^*(x)$ of (12) is obtained via (16) by factoring out the common factor $c = [v_n]^{2s+1}$.

Proof: We start by the observation that we may compute the determinant of B by expanding with respect to the elements p_m and v_n, leading to

$$\det(B) = \pm[p_m \cdot v_n]^{s+1} = \pm[d_{m,n}]^{2s+2}.$$

Secondly, we have by construction

$$K^* \cdot B = \begin{bmatrix} K & 0 \\ \# & \hat{K} \end{bmatrix}, \tag{19}$$

and taking determinants gives the desired determinantal identity. Writing X for the solution of (18), we also get that

$$K^* \cdot B \cdot \begin{bmatrix} 0 \\ X \end{bmatrix} = \begin{bmatrix} K & 0 \\ \# & \hat{K} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ X \end{bmatrix} = \begin{bmatrix} 0 \\ \hat{K} \cdot X \end{bmatrix}.$$

It is easily checked that on the left-hand side we multiply K^* on the right with the coefficient vector of $\mathbf{M}(x) \cdot \hat{\mathbf{M}}(x)$. Since at the right hand side we find $\det(\hat{K}) = \pm[v_n]^{2s+1} \cdot \det(K^*)$ times some columns of the identity $I_{m+n+2s+2}$, and since K^* is non-singular, we obtain the claimed identity for $\mathbf{M}^*(x)$ by uniqueness of Mahler systems at normal points. □

Equation (19) is similar to the well-known Schur complement formula. In fact, the classical Schur complement K^*/K would be obtained at the position of $\hat{K}$ by choosing a multiplier B having a slightly different structure. However, our choice of the multiplier is more appropriate for taking into account the special structure of K^* and K. We therefore may refer to $\hat{K}$ as a *modified Schur complement.*

The complexity of this approach depends on the stepsize s: if all quantities s are bounded by some modest constant, then again we obtain an $\mathcal{O}(n^4 \cdot N^2)$ algorithm.

By looking carefully at system (18), one may see that the matrix $\hat{\mathbf{M}}(x)$ already contains a common factor v_n according to the particular form of the last two rows of $\hat{K}$. Let $\widetilde{K}$ be obtained from $\hat{K}$ by dividing the last two rows through by v_n. One may easily relate the solution of (18) to that obtained by replacing $\hat{K}$ in (18) by $\widetilde{K}$.

COROLLARY 4.6. *Let (m,n) be a normal point. Then (m^*,n^*) is a normal point iff $\widetilde{K}$ is nonsingular. In the latter case, let $\widetilde{\mathbf{M}}(x)$ be obtained using fraction-free Gaussian elimination by solving system (18) with $\hat{K}$ being replaced by $\widetilde{K}$. Then the new Mahler system $\mathbf{M}^*(x)$ of (12) may be computed as*

$$(v_n)^{2s}\cdot\mathbf{M}^*(x)=\mathbf{M}(x)\cdot\widetilde{\mathbf{M}}(x),\quad \textit{if } \sigma=0, \tag{20}$$

$$\mathbf{M}^*(x)\cdot\begin{bmatrix}(v_n)^{2s-1} & 0\\ 0 & (v_n)^{2s}\end{bmatrix}=\mathbf{M}(x)\cdot\widetilde{\mathbf{M}}(x),\quad \textit{if } \sigma>0. \tag{21}$$

REMARK 4.7. *One can prove the recurrence relations of Theorem 4.4 using Theorem 4.5. For instance, part (b) follows by taking the parameters $\tau^*=\tau+1$, $s=1$, and $\sigma=0$.*

REMARK 4.8. *Following a diagonal path $\vec{n}_k=(k,k)$, our algorithm (with the new normalization of (20)) may be understood as a fast two-step fraction-free Gaussian elimination. This new normalization is in accordance with the results given in [6, Theorem 3.3] for Matrix Padé approximation on diagonal paths. Notice that, for diagonal paths, the $\widetilde{\mathbf{M}}(x)$ is also a Mahler system, but for the residual functions $\hat{f},\hat{g}$.*

REMARK 4.9. *Another interesting special case is given by the antidiagonal $\vec{n}_{k+j}=\vec{n}_k+(-j,j)$, $j\geq 1$. Here $\hat{K}$ is a Sylvester matrix (up to the last row) since, because $\tau^*=\tau$, the paired Vandermonde part in $\hat{K}$ vanishes. It is not difficult to see that the smallest index $s\geq 1$ with nonsingular $\widetilde{K}$ is given by $\deg u=m-s$, and then*

$$\widetilde{K}=\left[\begin{array}{ccc|c|ccc|c}
p_{m-s} & \cdots & p_{m+1-2s} & p_{m-2s} & u_{m-s} & \cdots & u_{m+1-2s} & u_{m-2s}\\
\vdots & & \vdots & \vdots & & \ddots & \vdots & \vdots\\
p_{m-1} & & p_{m-s} & p_{m-s-1} & & & u_{m-s} & u_{m-s-1}\\
\hline
p_m & & p_{m-s+1} & p_{m-s} & & & & u_{m-s}\\
 & \ddots & \vdots & \vdots & & & & \\
 & & p_m & p_{m-1} & & & & \\
 & & & 1 & & & & \\
\hline
 & & & & & & & 1
\end{array}\right].$$

In particular (recall that $p_m=v_n$),

$$det\,(\widetilde{K})=[p_m\cdot u_{m-s}]^s,\quad \hat{\mathbf{M}}(x)=\begin{bmatrix}0 & -p_m^{s-1}\cdot u_{m-s}^{s+1}\\ p_m^s\cdot u_{m-s}^{s-1} & \hat{v}(x)\end{bmatrix},$$

and $\hat{v}(x) = p_m^{s-1} \cdot Q(x)$, *where* $Q(x) = \mathrm{pquo}(p(x), u(x))$ *is the pseudo-quotient obtained via the pseudo-division*

$$\begin{bmatrix} u_{m-s} & \cdots & u_{m-2s} \\ & \ddots & \vdots \\ & & u_{m-s} \end{bmatrix} \cdot \begin{bmatrix} Q_0 \\ \vdots \\ Q_s \end{bmatrix} = u_{m-s}^{s+1} \begin{bmatrix} p_{m-s} \\ \vdots \\ p_m \end{bmatrix}.$$

Thus recurrence (21) takes the form

$$p_m^{2s-1} \begin{bmatrix} p^*(x) \\ q^*(x) \end{bmatrix} = p_m^s u_{m-s}^{s-1} \begin{bmatrix} u(x) \\ v(x) \end{bmatrix}, \quad p_m^{2s} \begin{bmatrix} u^*(x) \\ v^*(x) \end{bmatrix} = -p_m^{s-1} u_{m-s}^{s+1} \begin{bmatrix} p(x) \\ q(x) \end{bmatrix} + p_m^{s-1} Q(x) \begin{bmatrix} u(x) \\ v(x) \end{bmatrix},$$

which in the present case can be simplified by dividing out common factors (this is in general no longer true if $\tau^* > \tau$*).*

It is also of interest to note that there is a strong link between the above recurrence and the Subresultant PRS algorithm mentioned in Section 3. Indeed, set

$$R_{i+1}(x) = u^*(x), \quad R_i(x) = u(x), \quad r_i = u_{m-s}, \quad \phi_i = p_m, \quad \phi_{i+1} = p^*_{m-s}, \quad s = \delta_i$$

then from the first recurrence we see that $\phi_i^{\delta_i - 1} \cdot \phi_{i+1} = r_i^{\delta_i}$, *and* $\phi_i^{\delta_i - 1} p^*(x) = r_i^{\delta_i - 1} R_i(x)$, *or* $r_i p^*(x) = \phi_{i+1} R_i(x)$. *Thus we may eliminate* $p(x)$ *in the second recurrence using* $r_{i-1} p(x) = \phi_i R_{i-1}(x)$. *Dividing the resulting equation by* $p_m^{s-1} \cdot \phi_i / r_i = \phi_i^{\delta_i} / r_i$ *leads to the relation*

$$-r_i^{\delta_i + 1} R_{i-1} + (r_i/\phi_i) Q(x) R_i(x) = p_m^s r_i R_{i+1}(x) = r_i \phi_i^{\delta_i} R_{i+1}(x).$$

Since $(r_i/\phi_i) Q(x) = (r_i/\phi_i) \mathrm{pquo}(p(x), u(x)) = \mathrm{pquo}(R_{i-1}(x), R_i(x))$, *we have that*

$$\alpha_i \cdot R_{i-1}(x) = \hat{Q}(x) \cdot R_i(x) + \beta_i \cdot R_{i+1}(x)$$

with $\alpha_i =_i^{\delta_i + 1}$ *and* $\beta_i = -r_i \phi_i^{\delta_i}$ *where* $\phi_i = r_{i-1}^{\delta_{i-1}} \cdot \phi_{i-1}^{1 - \delta_{i-1}}$. *Thus we obtain (up to signs) exactly the relation of the the Subresultant PRS algorithm.*

5. Related and Future Work

In this paper we have considered the problem of computing rational approximants and interpolants for scalar power series. The intent is that the methods that we describe work at the implementation level. The arithmetic environments considered for such implementations include numeric and symbolic domains. In the numeric environment the

primary issue is to find algorithms that are both efficient and at the same time numerically stable. In the symbolic environment the main issue is to find algorithms that are efficient while controlling the size of intermediate computations to avoid exponential coefficient growth.

The intent of this paper is to show some of the modifications needed for efficient algorithms at the implementations level. The problems considered here have a common thread. In all cases the problems seek to compute coefficient representations of polynomials that satisfy certain order-like equations. The problems can all be considered as solving a linear system of equations with a special structured coefficient matrix. These structured matrices include Hankel and Toeplitz (in the case of rational approximation problems), Sylvester (in the case of GCD problems) and paired Vandermonde (in the case of rational interpolants). One can extend many of the algorithms discussed in this paper using linear functionals and a *special rule* as described in [9]. In this case the structured matrix is a striped Krylov matrix. These allow for vector and matrix versions of various rational approximation and interpolation problems — for example Hermite Padé, M-Padé or Matrix Padé Approximants.

We should point out that computing rational approximants and interpolants along with their vector and matrix generalizations has many applications in computer algebra computations. Such computations appear in such applications as the Gfun package of Salvy and Zimmerman [46] for determining recurrences relations, computation of matrix greatest common divisors [9] and the factorization of linear differential operators having rational function coefficients [50].

In all cases there is a general theme that runs through our search for efficient algorithms for such linear systems. Namely we want to solve the linear systems efficiently by taking into consideration the special structure rather than just using general techniques such as Gaussian elimination while at the same time taking care to avoid the pitfalls that are part of the computational environment. Two methods, the Cabay–Meleshko algorithm for Padé approximation and the modified Schur complement algorithm for rational interpolation are both look–ahead methods that build their solutions in an efficient way by solving a number of smaller linear systems. In both cases the methods involve combining solutions of small systems of a certain type (stable in Cabay–Meleshko and Cramer solutions in modified Schur complements) and then normalizing the results (by making the 1 norm unity in the numeric case and dividing out a predicted constant in the fraction-free case).

The *look–around* method of section 4 can be generalized to a number of important approximation and interpolation problems [9]. One of

its primary advantages is that it determines all solutions of a given problem, that is, a basis for a particular algebraic module defined by the order conditions of the problem. However, it does not appear, at least at the time of writing, to have a counterpart in a numeric settings. Indeed it seems to be a difficult task to imagine determining a closest stable point of a particular order along a specific path of computation. However, as remarked by one of the referees, a "close enough" stable path may be sufficient.

In the case of exact arithmetic environments there are a number of directions that can be pursued. A classic technique in computer algebra [28] for overcoming the problem of intermediate coefficient growth is to map the problem into a number of problems which are easier computationally (for example reducing to modular arithmetic or evaluating variables at a point), solve the individual problems and then combine the results into a solution of the original problem (for example using Chinese remaindering or polynomial interpolation). This gains an order of magnitude in the case of GCD computations. In order to design a modular algorithm one needs to have an idea of some of the properties of the final result. For example, one needs to have an idea of the potential size of the answers in order to determine the number of problems to consider. In addition, one needs to be able to ensure that a final answer has a proper normalization (the reduced domains often have extra algebraic properties which use different normalizations) [28]. We expect that both of these properties can be found using the Cramer solutions determined by our algorithms. As such we expect that we can build efficient modular algorithms for rational approximation and interpolation problems along with their vector and matrix generalizations. We also expect this can help in determining solutions using a second classical technique from computer algebra, namely Hensel lifting.

Finally, in the case of exact arithmetic there are a number of other problems which we would like to be able to compute efficiently with a fraction-free procedure. These include one sided greatest common divisors of differential operators (used in finding closed form solutions of linear ODEs) [43], minimal bases for kernels of matrix polynomials, and matrix normal forms such as Popov and Hermite normal forms. Partial results for the case of matrix normal forms and minimal bases for kernels can be found in [12].

References

1. A.G. Akritas, *Elements of Computer Algebra with Applications*,Wiley-Interscience, (1989).

2. G.A. Baker & P.R. Graves-Morris, *Padé Approximants* , second edition, Cambridge Univ. Press, Cambridge, UK (1995).
3. E. Bareiss, Sylvester's Identity and multistep integer-preserving Gaussian elimination, *Math. Comp.* **22**(103) (1968) 565-578.
4. B. Beckermann, A reliable method for computing M-Padé approximants on arbitrary staircases, *J. Comput. Appl. Math.* **40** (1992) 19-42.
5. B. Beckermann, The stable computation of formal orthogonal polynomials, *Numerical Algorithms* **11** (1996) 1-23.
6. B. Beckermann, S. Cabay & G. Labahn, Fraction–free Computation of Matrix Padé Systems, *Proceedings of ISSAC'97*, Maui, ACM Press, (1997) 125-132.
7. B. Beckermann & C. Carstensen, QD–type algorithms for the non–normal Newton–Padé Approximation Table, *Constructive Approximation* **12** (1996) 307-330.
8. B. Beckermann & G. Labahn, Recursiveness in Matrix Rational Interpolation Problems, *J. Comput. Appl. Math.* **77** (1997) 5-34.
9. B. Beckermann & G. Labahn, Fraction-free Computation of Matrix GCD's and Rational Interpolants. University of Waterloo Tech Report (1997).
10. B. Beckermann & G. Labahn, When are two numerical polynomials relatively prime? *J. of Symbolic Computation* **26** (1998) 677-689.
11. B. Beckermann & G. Labahn, A fast and numerically stable Euclidean–like algorithm for detecting relatively prime numerical polynomials. *J. of Symbolic Computation* **26** (1998) 691-714.
12. B. Beckermann, G. Labahn & G. Villard, Shifted Normal Forms of Polynomial Matrices, *Proceedings of ISSAC'99*, Vancouver (1999) 189-196.
13. A.W. Bojanczyk, R.P. Brent & F.R. de Hoog, Stability analysis of a general Toeplitz systems solver, *Numerical Algorithms* **10** (1995) 225-244.
14. A.W. Bojanczyk, R.P. Brent, F.R. de Hoog & D.R. Sweet, On the Stability of the Bareiss and related Toeplitz Factorization Algorithms, *SIAM J. Matrix Anal. Appl.* **16** (1995) 40-57.
15. R. Brent, F.G. Gustavson and D.Y.Y. Yun, Fast solution of Toeplitz systems of equations and computation of Padé approximants, *J. of Algorithms* **1** (1980) 259-295.
16. W. Brown & J.F. Traub, On Euclid's algorithm and the theory of subresultants, *J. ACM* **18** (1971) 505-514.
17. S. Cabay, A. R. Jones & G. Labahn, Computation of Numerical Padé-Hermite and Simultaneous Padé Systems II: A Weakly Stable Algorithm, *SIAM J. Matrix Anal. Appl.* **17** (1996) 268-297.
18. S. Cabay & P. Kossowski, Power Series remainder sequences and Padé fractions over an integral domain, *J. Symbolic Computation*, 10 (1990), pp. 139-163.
19. S. Cabay & R. Meleshko, A weakly stable Algorithm for Padé Approximants and the Inversion of Hankel matrices, *SIAM J. Matrix Anal. Appl.* **14** (1993) 735-765.
20. S. Chandrasekaran & A.H. Sayed, Stabilizing the generalized Schur algorithm, *SIAM J. Matrix Anal. Appl.* **14** (1996) 950-983.
21. G. Collins, Subresultant and Reduced Polynomial Remainder Sequences. *J. ACM* **14**, (1967) 128-142
22. G. Claessens: On the Newton–Padé approximation problem *J. Approx. Th.* **22** (1978) 150-160.
23. G. Claessens: On the structure of the Newton–Padé table. *J. Approx. Th.* **22** (1978) 304-319.

24. G. Claessens, Some aspects of the rational Hermite interpolation table and its applications, Ph.D. Thesis, University of Antwerp, 1979.
25. S.R. Czapor and K.O. Geddes, A comparison of algorithms for the symbolic computation of Padé approximants. *Proceedings of EUROSAM'84*, J. Fitch (ed.), Lecture Notes in Computer Science, No. 174, Springer-Verlag, Berlin, 1984, pp. 248-259.
26. R.W. Freund & H. Zha, Formally biorthogonal polynomials and a look–ahead Levinson algorithm for general Toeplitz systems, *Linear Algebra Appl.* **188/89** (1993) 255-303.
27. R.W. Freund & H. Zha, A look–ahead algorithm for the solution of general Hankel systems, *Numer. Math.* **64** (1993) 295-321.
28. K.O. Geddes, S.R. Czapor & G. Labahn, *Algorithms for Computer Algebra*, (Kluwer, Boston, MA, 1992)
29. I. Gohberg, T. Kailath & V. Olshevski, Fast Gaussian elimination with partial pivoting for matrices with displacement structure, *Math. Comp.* **64** (1995) 1557-1567.
30. G. Golub & V. Olshevski, Pivoting for structured matrices, with Applications. Manuscript (1997). `http://www-isl.stanford.edu/~olshevsk`
31. P. Graves-Morris, Efficient Reliable Rational Interpolation, *Padé Approximation and its Applications 1980*, Springer-Verlag, (1980) 28-63.
32. M. Gu, Stable and efficient algorithms for structured systems of linear equations, *SIAM J. Matrix Anal. Appl.***19**(2) (1998) 279-306.
33. M.H. Gutknecht, Stable Row Recurrences for the Padé Table and Generically Superfast Look-ahead Solvers for Non-Hermitian Toeplitz Systems, *Linear Algebra Appl.* **188/89** (1993) 351-421.
34. M.H. Gutknecht, The multipoint Padé table and general recurrences for rational interpolation, in: *Nonlinear Numerical Methods and Rational Approximation* (A. Cuyt ed.), Kluwer Academic Publishers (1994) 109-136.
35. M.H. Gutknecht & M. Hochbruck, Look-ahead Levinson and Schur algorithms for non–Hermitian Toeplitz Systems, *Numer. Math.* **70** (1995) 181-227.
36. W. Habicht, Eine Verallgemeinerung des Sturmschen Wurzelählverfahrens, *Commentarii Mathematici Helvetici* **21**, (1948) 99-116.
37. G. Heinig & K. Rost, *Algebraic methods for Toeplitz-like matrices and operators*, Operator Theory, v13, (Birkhäuser, Basel, 1984).
38. D. Knuth, The Art of Computer Programming Vol 2, Addison-Wesley, (1981).
39. P. Kravanja & M. Van Barel, A fast Hankel solver based on an inversion formula for Loewner matrices. *Linear Algebr. Appl.* **282** (1998) 275-295.
40. P. Kravanja & M. Van Barel, A fast block Hankel solver based on an inversion formula for block Loewner matrices, *CALCOLO* **33**(1996) 147-164.
41. P. Kravanja & M. Van Barel, Coupled Vandermonde matrices and the superfast computation of Toeplitz determinants. Submitted to *Numerical Algorithms*. Proceedings of the International Conference on Rational Approximation (ICRA99), Antwerp (Belgium).
42. G. Labahn, Inversion Components of Block Hankel-like Matrices, *Linear Algebra Appl.* **177** (1992) 7-48
43. Z. Li, A Subresultant Theory for Linear Differential, Linear Difference and Ore Polynomials, with Applications, PhD Thesis, Univ. Linz, Austria, 1996.
44. K. Mahler, Perfect systems, *Compos. Math.* **19** (1968) 95-166. theorems. *J. Comput. Appl. Math.* **32** (1990) 229-236.
45. B. Mishra, Algorithmic Algebra, Springer Verlag, (1993).

46. B. Salvy & P. Zimmermann, Gfun: a Maple package for the manipulation of generating and holonomic functions in one variable, *ACM Transactions on Mathematical Software (TOMS)*, 20(2) (1994) 163-177.
47. J.J. Sylvester, On a theory of the syzgetic relations of two rational integral functions, comprising an application to the theory of Sturm's functions, and that of the greatest algebraic common measure. *Philosophical Transactions* **143**, (1853) 407-548.
48. M. Van Barel & A. Bultheel, A new formal approach to the rational interpolation problem. *Numerische Mathematik* **62** (1992) 87-122.
49. M. Van Barel & A. Bultheel, A look–ahead algorithm for the solution of block Toeplitz systems, *Linear Algebra Appl.* **266** (1997) 291-335.
50. M. Van Hoeij, Factorization of Differential Operators with Rational Function Coefficients. *Journal of Symbolic Computation* (1998).
51. D.D. Warner: Hermite Interpolation with Rational Functions, Ph.D. Thesis, Univ. of California, San Diego 1974.
52. H. Werner: A reliable method for rational interpolation, in: L. Wuytack, Ed., *Padé Approximation and its Applications, Antwerp 1979*, Lecture Notes in Math. **765** (Springer, Berlin, 1979) 257-277.