Solving generalised Padé approximations over polynomial rings

Johan S. R. Nielsen
Ulm University
Institute of Communications Engineering
Ulm, Germany
jsrn@jsrn.dk

ABSTRACT

We consider a type of Padé approximations over polynomial rings over fields, dubbed "2D Padé approximations", which generalise both simultaneous and Hermitian Padé approximations. This form is related to but distinct from both matrix-Padé approximations as well as order bases. These approximations occur in e.g. decoding of algebraic codes.

We demonstrate how to solve such approximations using row reduction of polynomial matrices, and we review the achieved asymptotic complexity when using various fast, known methods for this.

We take a closer look at the Mulders–Storjohann for row reduction and show that its complexity is linked to the orthogonality defect of the input matrix, implying that it is fast when applied to 2D Padé approximations. We design a new algorithm, which essentially carries out the computations performed by Mulders–Storjohann in a demand-driven manner, yielding an algorithm which is often even faster. This algorithm could be considered as a heavy generalisation of the classical Berlekamp–Massey algorithm.

We also consider "weighted" variants of 2D Padé approximations, and it is shown how the considered algorithms can handle this with very little speed penalty.

Categories and Subject Descriptors

I.1.2 [**Symbolic and Algebraic Mainpulation**]: Algorithms; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

Padé approximation, order basis, Berlekamp–Massey

1. INTRODUCTION

Let $\mathbb{F}$ be a field. We consider the following "2D Padé approximation" problem: given $\boldsymbol{S} = [S_{i,j}] \in \mathbb{F}[x]^{\rho\times\sigma}$ and $G_1, \dots, G_\sigma \in \mathbb{F}[x]$ we are seeking $(\Lambda_1, \dots, \Lambda_\rho, \Omega_1, \dots, \Omega_\sigma) \in \mathbb{F}[x]^{\rho+\sigma}$ such that

$$\sum_{i=1}^{\rho} \Lambda_i S_{i,j} \equiv \Omega_j \mod G_j, \quad \text{for } j = 1, \dots, \sigma \qquad (1)$$

We are specifically interested in such vectors which have low degrees. More precisely, we will be considering two *types* of these equations: the symmetric and the asymmetric. We will consider both in a general "weighted" form, but first we discuss the simpler, unweighted case.

For the *symmetric type*, we are interested in such a vector of minimal degree, i.e. minimising $\max_{i,j}\{\deg \Lambda_i, \deg \Omega_j\}$.

For the *asymmetric type*, we likewise seek the vector of minimal degree but subject to $\max_i\{\deg \Lambda_i\} > \max_j\{\deg \Omega_j\}$.

This problem directly generalises a number of well-known approximation problems over polynomial rings, see Table 1. These occur naturally in a number of settings, e.g. control theory. In algebraic coding theory, solving the so-called "key equation" has for more than 40 years been used for minimum-distance decoding of Reed–Solomon, Goppa and other algebraic codes. It is a classical Padé approximation but it is of asymmetric type and often modulo a general polynomial (and not a power of x). Classically, this equation is solved using the Berlekamp–Massey algorithm [4], or the extended Euclidean algorithm [18]. Recently, generalisations of this equation have been identified in list-decoding of Reed–Solomon codes [15, 19] or interleaved codes [16], usually solved using ad-hoc generalisations of the Berlekamp–Massey algorithm. In the PhD thesis of the author, such usage was systematised and extended to (list-)decoding of more general algebraic-geometry codes, utilising the full generality of the definition of 2D Padé approximations [12][1].

The 2D Padé approximation is of similar expressive power as the general approximation problems matrix Padé and order bases. We compare closer with these in Section 1.2. Following work on these, we will put the 2D Padé approximation problem into a module theoretic framework and solve them using row reduction of polynomial matrices. For a given problem, the matrix to reduce will have size $(\rho+\sigma)\times(\rho+\sigma)$ and degree $\max_j\{\deg G_j\}$. We will demonstrate this in Section 2. The definition of 2D Padé approximation is here seen to naturally fit with the lattice formulation. In fact, 2D Padé approximations can be solved by any of

[1] In that work, the name "2D key equation" was used for our problem, owing to the focus on coding theory.

Submitted to the 39th International Symposium on Symbolic and Algebraic Computation, July 2014, Kobe, Japan

Some approximation problems as 2D Padé

Name	Type	Specialisations
Classical Padé [2]	S	$\rho = \sigma = 1$, $G_1 = x^d$
Hermite Padé [2]	S	$\sigma = 1$, $G_1 = x^d$
Simultaneous Padé [2]	S	$\rho = 1$, $G_j = x^{d_j}$
Key equation (KE) [4, 14, 18]	A	$\rho = \sigma = 1$,
Multi-LFSR [6, 16]	A	$\rho = 1$, $G_j = x^{d_j}$
A–G KE	A	$\rho = \sigma = 1$,
Roth–Ruckenstein KE [15]	S	$\sigma = 1$, $G_1 = x^d$
Roth–Ruckenstein KE Ext. [19]	S	$G_j = x^{d_j}$

Table 1: For type, "S" refers to symmetric and "A" asymmetric.

the aforementioned general approximation problems, but it would be at the cost of larger, more indirect lattices.

Row reduction can be carried out by a number of algorithms: the Mulders–Storjohann algorithm [10] is generally asymptotically good when not using fast multiplication techniques. It is subject to a divide & conquer speedup, given by Alekhnovich [1]. When the matrix has small dimension compared to the degrees of its entries, the asymptotically fastest is the Las Vegas algorithm by Giorgi, Jeannerod and Villard [8] (the GJV), or the deterministic algorithm by Zhou and Labahn [20], both of which also employ fast multiplication[2]. We will see that applying all these algorithms for solving 2D Padé approximations will have the complexities as given in Table 2. There, and in the rest of the paper, $\mathrm{P}(n)$ is the complexity of multiplying two polynomials of degree at most n, while $\mathrm{M}(m)$ is the complexity of multiplying two matrices in $\mathbb{F}^{m\times m}$. Also, $O^\sim$ refers to big-O with logarithmic factors omitted, and $\#\mathrm{supp}(G)$ for some $G \in \mathbb{F}[x]$ is the number of non-zero monomials in G.

The complexity of applying the GJV or Zhou–Labahn will follow generically. However, for the Mulders–Storjohann, we will show that it performs better than expected on matrices resulting from 2D Padé approximations. This is done by pointing out with a new analysis that the algorithm's complexity is directly linked to the *orthogonality defect* of the input matrix. This also carries over to Alekhnovich's algorithm.

A main result of the paper is a recasting of the Mulders–Storjohann algorithm specifically for 2D Padé approximations. The intermediate results of the Mulders–Storjohann turn out to be highly redundant for such a matrix, which means that we could instead compute most of it *on demand*. The resulting "Demand–Driven" algorithm is an asymptotic improvement for many relative sizes of the approximation problem's parameters, as well as having a very low hidden constant and low memory complexity.

In applications, e.g. in many of those from coding theory, one needs to shift the relative importance of the degrees of the Λ_i and Ω_j; for this, we introduce *weights*: $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_\rho) \in \mathbb{N}_0^\rho$ and $\boldsymbol{\mu} = (\mu_1, \ldots \mu_\sigma) \in \mathbb{N}_0^\sigma$, as well as $\nu \in \mathbb{Z}_+$. For the *symmetric* type, we now wish to minimise

$$\max_{i,j}\{\nu \deg \Lambda_i + \eta_i,\ \nu \deg \Omega_j + \mu_j\}$$

For the *asymmetric* type, we wish to minimise the same but subject to

$$\max_i\{\nu \deg \Lambda_i + \eta_i\} > \max_j\{\nu \deg \Omega_j + \mu_j\}$$

Notice that ν emulates allowing fractional weights in a highly controlled manner.

We will show in Section 5 two mappings which embed these weights into the matrix to be row reduced. The first is simple but will generically incur an overhead of at least a factor ν in the complexity. We therefore derive from it a slightly more involved mapping which embeds the weights with practically no overhead. Finally, in Section 5.1 we show how the weights can be handled by the new demand–driven algorithm.

The special case of weighted 2D Padé approximations with $\rho = 1$ were investigated by the author in [11] where also a simpler version of the Demand–Driven algorithm was presented.

[2]These two algorithms compute "order bases", which can be used for row reduction. See also Section 1.2.

Complexity for solving a weighted 2D Padé approximation

Algorithm	Field operations in big-$O^\sim$
Mulders–Storjohann	$\rho(\rho+\sigma)^2\gamma^2$
Demand–Driven	$\rho^2(\rho+\sigma)\gamma\mathrm{P}_G(\gamma)$
Alekhnovich	$\mathrm{M}(\rho+\sigma)\mathrm{P}(\rho\gamma)$
GJV or Zhou–Labahn	$\mathrm{M}(\rho+\sigma)\mathrm{P}(\gamma)$

Table 2: Here, $\gamma = \max_j\{\deg G_j\}$ in the unweighted case, and $\gamma = \max\{\eta_1/\nu, \ldots, \eta_\rho/\nu,\ \deg G_1 + \mu_1/\nu, \ldots, \deg G_\sigma + \mu_\sigma/\nu\}$ in general. $\mathrm{P}_G(\gamma)$ is the cheaper of $\mathrm{P}(\gamma)$ and $O(\gamma \max_j\{\#\mathrm{supp}(G_j)\})$.

1.1 Notation

For $\boldsymbol{v} = (v_1, \ldots, v_m) \in \mathbb{F}[x]^m$, let $\deg \boldsymbol{v} = \max_i\{\deg v_i\}$, and similarly for $\boldsymbol{V} = [v_{i,j}] \in \mathbb{F}[x]^{n\times m}$ then $\deg \boldsymbol{V} = \max_{i,j}\{\deg v_{i,j}\}$. We will also deal with row-degrees of matrices: $\mathrm{rowdeg}\, \boldsymbol{V} = \sum_{i=1}^n \deg \boldsymbol{v}_i$ where the $\boldsymbol{v}_i$ are the rows of $\boldsymbol{V}$.

By the leading position $\mathrm{LP}(\boldsymbol{v})$, we will mean the *rightmost* element of $\boldsymbol{v}$ which has degree $\deg \boldsymbol{v}$. This breaking of ties should be kept in mind; it has subtle effects in e.g. Section 5. From this we let $\mathrm{LT}(\boldsymbol{v}) = v_{\mathrm{LP}(\boldsymbol{v})}$. For a polynomial $p = \sum_{h=0}^{\deg p} p_h x^h \in \mathbb{F}[x]$, then $\mathrm{LC}(p) = p_{\deg p}$.

We will be working with free $\mathbb{F}[x]$-modules which are subsets of $\mathbb{F}[x]^m$. When we say "module" we will always mean such a module, and we will be representing bases of them by matrices whose rows are basis elements. We will refer to such matrices directly as "bases".

1.2 Relation to other approximations

There are other, already studied approximations of high generality. We briefly compare our new definition with two of these: solving 2D Padé approximations with either of these are possible but less direct as the manner we propose in Section 2.

Slightly paraphrased, a matrix Padé [3] is, given $\boldsymbol{T} \in \mathbb{F}[x]^{m\times s}$ and $N, M, \tau \in \mathbb{Z}_+$, the task of finding $\boldsymbol{A} \in \mathbb{F}[x]^{n\times m}$ and $\boldsymbol{B} \in \mathbb{F}[x]^{n\times s}$ such that $\boldsymbol{A}\boldsymbol{T} \equiv \boldsymbol{B} \mod x^\tau$ while $\deg \boldsymbol{A} \leq N$ and $\deg \boldsymbol{B} \leq M$, and A has full rank. It is not embedded in the definition that the rows of $\boldsymbol{A}$ should in any way represent the *smallest* solutions, or equivalently, that $\boldsymbol{A}$ is row reduced. Therefore, though the congruence part of 2D Padé approximations can be emulated with matrix Padé, we can not simply choose any off-the-shelf algorithm for matrix

Padé and apply.

The related "Order bases" [3] embeds this kind of minimality, but does so by requiring a complete basis of solutions: given $\boldsymbol{T} \in \mathbb{F}[x]^{m\times s}$ and an order $\tau \in \mathbb{Z}_+$, the order basis is a matrix $\boldsymbol{A} \in \mathbb{F}[x]^{n\times m}$ whose rows span the $\mathbb{F}[x]$-module of vectors $\boldsymbol{a} \in \mathbb{F}[x]^m$ satisfying $\boldsymbol{aT} \equiv \boldsymbol{0} \mod x^\tau$. Furthermore, $\boldsymbol{A}$ should have minimal row degree, or rather, it should be in weak Popov form, see Section 2. Unweighted, symmetric 2D Padé approximations where all G_j are the same power of x can be solved as an order basis: set $\boldsymbol{T} = [\boldsymbol{S}^\top \mid -\boldsymbol{I}]^\top$, and then $(\Lambda_1, \ldots, \Lambda_\rho, \ldots, \Omega_1, \ldots, \Omega_\sigma)$ will be a minimal-degree row in the order basis. For asymmetric 2D Padé, we instead find the minimal-degree row with leading position at most ρ. When the G_j are different powers of x, we can multiply the columns of $\boldsymbol{T}$ by x-powers in an appropriate way such that one value of $\tau = \max_j\{\deg G_j\}$ suffice for the congruence.

For general G_j, one would need to convert to a larger order basis problem. One approach is to set $T = [\boldsymbol{S}^\top \mid -\boldsymbol{I} \mid \operatorname{diag}(G_1, \ldots, G_\sigma)]^\top$ and set τ large enough so that the required congruence actually lifts to an equality for all rows in the found order basis. Another approach is solve the 2D Padé approximation by row reduction, using Section 2, and solve this row reduction using an order basis computation as explained in [8]. This has the advantage of immediately carrying over the solutions from this paper for handling weights. In either case the order basis problem considered will have more rows than $\rho + \sigma$ and seemingly attempts to capture more information than we are interested in. The asymptotic complexity is completely unharmed, though, so one can therefore sensibly use order bases for solving 2D Padé approximations.

2. SOLUTION BY ROW REDUCTION

Let some unweighted 2D Padé approximation problem be given; i.e we are seeking the $\boldsymbol{s} = (\Lambda_1, \ldots, \Lambda_\rho, \Omega_1, \ldots, \Omega_\sigma)$ such that the congruences (1) are satisfied, and such that $\deg \boldsymbol{s}$ is minimal. If we are solving an asymmetric approximation, we also wish $\operatorname{LP}(\boldsymbol{s}) \leq \rho$. Consider first the space $\mathcal{M}$ of all vectors $(\lambda_1, \ldots, \lambda_\rho, \omega_1, \ldots, \omega_\sigma) \in \mathbb{F}[x]^{\rho+\sigma}$ which satisfy only the congruence equations:

$$\textstyle\sum_{i=1}^{\rho} \lambda_i S_{i,j} \equiv \omega_j \mod G_j, \quad \text{for } j = 1, \ldots, \sigma \qquad (2)$$

$\mathcal{M}$ forms an $\mathbb{F}[x]$-module. In fact, we have

Proposition 1. *$\mathcal{M}$ is generated as an $\mathbb{F}[x]$-module by the rows of $\boldsymbol{M} \in \mathbb{F}[x]^{(\rho+\sigma)\times(\rho+\sigma)}$ where*

$$\boldsymbol{M} = \left(\begin{array}{c|c} \boldsymbol{I} & \boldsymbol{S} \\ \hline 0 & \operatorname{diag}(G_1, \ldots, G_\sigma) \end{array}\right)$$

Proof. Equation (2) is easily seen to be satisfied for each row of $\boldsymbol{M}$, so by $\mathbb{F}[x]$-linearity, it is satisfied for every vector in $\mathcal{M}$. Contrarily, for any vector $\boldsymbol{v} = (\lambda_1, \ldots, \lambda_\rho, \omega_1, \ldots, \omega_\sigma)$ satisfying (2) there must exist $p_1, \ldots, p_\sigma \in \mathbb{F}[x]$ such that $\sum_{i=1}^{\rho} \lambda_j S_{i,j} + p_i G_i = \omega_i$ wherefore $\boldsymbol{v} = \lambda_1\boldsymbol{m}_1 + \ldots + \lambda_\rho\boldsymbol{m}_\rho + p_1\boldsymbol{m}_{\rho+1} + \ldots + p_\sigma\boldsymbol{m}_{\rho+\sigma}$, where $\boldsymbol{m}_i$ are the rows of $\boldsymbol{M}$. □

To solve the 2D Padé approximation we can seek a vector of minimal degree in the row space of $\boldsymbol{M}$, and in the asymmetric case, also subject to having leading position at most ρ. We will see that in the weak Popov form for polynomial matrices, which was introduced in [10], such vectors will figure directly as rows:

Definition 2. *A full-rank matrix $\boldsymbol{V} \in \mathbb{F}[x]^{m\times m}$ is in* weak Popov form *if an only if the leading position of all rows are different. The* orthogonality defect *of $\boldsymbol{V}$ is $\Delta(\boldsymbol{V}) \triangleq \operatorname{rowdeg} \boldsymbol{V} - \deg\det \boldsymbol{V}$.*

We will use "a weak Popov form of $\boldsymbol{U}$" for some matrix $\boldsymbol{U}$ to mean any unimodular equivalent matrix $\boldsymbol{V}$ which is in weak Popov form. That such a $\boldsymbol{V}$ always exists for any $\boldsymbol{U}$ follows from e.g. the Mulders–Storjohann algorithm for computing it; note however that it is not canonical. That a matrix $\boldsymbol{V}$ is "row reduced" means that $\operatorname{rowdeg} \boldsymbol{V}$ is minimal over all matrices unimodular equivalent to $\boldsymbol{V}$. It follows from the lemma below that a matrix in weak Popov form is row reduced. Sometimes, "row reduced" is defined directly as the more useful definition of weak Popov form above.

The concept of orthogonality defect was introduced by Lenstra [9] for estimating the running time of his algorithm for row reduction; we will use it to a similar effect. The following lemma gives the foundations for such a use. We omit the easy proof which can be found in [12]:

Lemma 3 ([12, Proposition 2.5]). *If a matrix $\boldsymbol{V}$ over $\mathbb{F}[x]$ is in weak Popov form then $\Delta(\boldsymbol{V}) = 0$.*

Clearly we always have $\Delta(\boldsymbol{V}) \geq 0$ so since the determinant is the same for *any* basis of the module for which $\boldsymbol{V}$ is a basis, $\Delta(\boldsymbol{V})$ measures how much $\operatorname{rowdeg} \boldsymbol{V}$ is greater than the minimal degree possible.

Definition 4. *The* value of a vector $\psi : \mathbb{F}[x]^m \to \mathbb{N}_0$ *is* $\psi(\boldsymbol{v}) = m \deg \boldsymbol{v} + \operatorname{LP}(\boldsymbol{v})$

The following well-known result is key to our method:

Proposition 5. *Let $\boldsymbol{V} \in \mathbb{F}[x]^{m\times m}$ be a basis in weak Popov form of a module $\mathcal{V}$. Any non-zero $\boldsymbol{b} \in \mathcal{V}$ satisfies $\deg \boldsymbol{v} \leq \deg \boldsymbol{b}$ where $\boldsymbol{v}$ is the row of $\boldsymbol{V}$ with $\operatorname{LP}(\boldsymbol{v}) = \operatorname{LP}(\boldsymbol{b})$.*

Proof. Since $\boldsymbol{V}$ is a basis of $\mathcal{V}$, there exists $p_1, \ldots, p_m \in \mathbb{F}[x]$ such that $\boldsymbol{b} = p_1\boldsymbol{v}_1 + \ldots + p_m\boldsymbol{v}_m$ where the $\boldsymbol{v}_i$ are the rows of $\boldsymbol{V}$. But the $\boldsymbol{v}_i$ all have different leading position, so the $p_i\boldsymbol{v}_i$ must as well for those $p_i \neq 0$, which in turn means that their values are all different. Notice that when for two $\boldsymbol{u}_1, \boldsymbol{u}_2$ with different value, then the value of $\boldsymbol{u}_1 + \boldsymbol{u}_2$ is that of either $\boldsymbol{u}_1$ or $\boldsymbol{u}_2$. That implies that there is a j such that $\psi(\boldsymbol{b}) = \psi(p_j\boldsymbol{v}_j)$, and this gives $\operatorname{LP}(\boldsymbol{b}) = \operatorname{LP}(\boldsymbol{v})$ and $\deg \boldsymbol{b} = \deg p_j + \deg \boldsymbol{v}_j$. □

For a 2D Padé approximation, we therefore see that a basis $\boldsymbol{V}$ for $\mathcal{M}$ in weak Popov form must contain a row $\boldsymbol{s}$ which is a minimal solution: if the approximation is symmetric, it is simply the minimal degree row of $\boldsymbol{V}$, while if it is asymmetric, it is the minimal degree row of $\boldsymbol{V}$ which has leading position at most ρ.

Remark 6. For asymmetric 2D Padé approximations, it is easy to characterise the entire space of solutions, and not only those of minimal degree, from the weak Popov form, by extending Proposition 5: from the degrees of the coefficient polynomials p_i we can determine the leading position of any $\boldsymbol{b} \in \mathcal{M}$. This is a reflection of the deeper statement that if $\boldsymbol{V}$ is in weak Popov form, then its rows form a Gröbner basis of its row space for a certain module monomial ordering. See [12, Section 2.1.2 and Theorem 2.35].

Any algorithm which brings $\mathbb{F}[x]$-matrices to weak Popov form can thus be used to solve the approximation. There exist a number of such algorithms, and whichever is asymptotically fastest on generic, square matrices depends on the relative size of its dimension m and its degree d. The Mulders–Storjohann [10] has $O(m^3d^2)$ but with a very low hidden constant. It admits a divide & conquer variant by Alekhnovich [1] which has $O^\sim(\mathrm{M}(m)\mathrm{P}(md))$. Using the fast order basis algorithm by Giorgi, Jeannerod or Villard [8], or the one by Zhou and Labahn [20], we have a complexity of $O^\sim(\mathrm{M}(m)\mathrm{P}(d))$. Plugging the parameters of $\boldsymbol{M}$ into the last expression, we directly get the corresponding entry in Table 2 for the unweighted case. However, for the two former algorithms, we will show in the following section how their complexity estimates can be refined to depend on the orthogonality defect of the input matrix. Due to its special form, $\boldsymbol{M}$ has particularly low orthogonality defect:

LEMMA 7. $\Delta(\boldsymbol{M}) = \deg \boldsymbol{S} < \rho \max_j\{\deg G_j\}$

PROOF. Since $\boldsymbol{M}$ is upper triangular $\det(\boldsymbol{M}) = \prod_{i=1}^{\sigma} G_i(x)$ and the lemma follows. □

3. MULDERS–STORJOHANN

We will now review the Mulders–Storjohann algorithm [10] for bringing a matrix to weak Popov form. We consider only square matrices, though the algorithm also applies to non-square. This allows a new complexity analysis, tying the complexity of the algorithm to the orthogonality defect. This analysis was also presented in [13], where it was utilised in a fast multi-trial list-decoder for Reed–Solomon codes.

DEFINITION 8. *Applying a* row reduction *on a full-rank matrix over $\mathbb{F}[x]$ means to find two different rows $\boldsymbol{v}_i, \boldsymbol{v}_j$, $\deg \boldsymbol{v}_i \leq \deg \boldsymbol{v}_j$ such that $\mathrm{LP}(\boldsymbol{v}_i) = \mathrm{LP}(\boldsymbol{v}_j)$, and then replacing $\boldsymbol{v}_j$ with $\boldsymbol{v}_j - \alpha x^\theta \boldsymbol{v}_i$ where $\alpha \in \mathbb{F}$ and $\theta \in \mathbb{N}_0$ are chosen such that the leading term of the polynomial $\mathrm{LT}(\boldsymbol{v}_j)$ is cancelled.*

LEMMA 9. *If $\boldsymbol{v}'_j$ is the vector replacing $\boldsymbol{v}_j$ in a row reduction, then $\psi(\boldsymbol{v}'_j) < \psi(\boldsymbol{v}_j)$.*

PROOF. We can't have $\deg \boldsymbol{v}'_j > \deg \boldsymbol{v}_j$ since all terms of both $\boldsymbol{v}_j$ and $\alpha x^\delta \boldsymbol{v}_i$ have degree at most $\deg \boldsymbol{v}_j$. If $\deg \boldsymbol{v}'_j < \deg \boldsymbol{v}_j$ we are done since $\mathrm{LP}(\boldsymbol{v}'_j) < m$, so assume $\deg \boldsymbol{v}'_j = \deg \boldsymbol{v}_j$. Let $h = \mathrm{LP}(\boldsymbol{v}_j) = \mathrm{LP}(\boldsymbol{v}_i)$. By the definition of LP, all terms in both $\boldsymbol{v}_j$ and $\alpha x^\delta \boldsymbol{v}_i$ to the right of h must have degree less than $\deg \boldsymbol{v}_j$, and so also all terms in $\boldsymbol{v}'_j$ to the right of h satisfies this. The row reduction ensures that $\deg v'_{j,h} < \deg v_{j,h}$, so it must then be the case that $\mathrm{LP}(\boldsymbol{v}'_j) < h$. □

The elegant Mulders–Storjohann algorithm is now succinctly described as Algorithm 1.

Algorithm 1 Mulders–Storjohann

Input: $\boldsymbol{V} \in \mathbb{F}[x]^{m\times m}$, a basis of $\mathcal{V}$.
Output: A basis of $\mathcal{V}$ in weak Popov form.

1 Apply row reductions on the rows of $\boldsymbol{V}$ until no longer possible.
2 **return** $\boldsymbol{V}$.

LEMMA 10. *The Mulders–Storjohann algorithm is correct. It performs fewer than $K = m\Delta(\boldsymbol{V}) + \sum_{i=1}^{m}(\mathrm{LP}\boldsymbol{v}_i - i) \leq m\Delta(\boldsymbol{V}) + \frac{1}{2}m^2$ row reductions on input $\boldsymbol{V}$ with rows $\boldsymbol{v}_i$. It has complexity $O(mK \deg \boldsymbol{V}) \in O(m^2\Delta(\boldsymbol{V}) \deg \boldsymbol{V})$.*

PROOF. Since we can apply a row reduction on a matrix if and only if it is not in weak Popov form, the algorithm must bring $\boldsymbol{V}$ to weak Popov form in case it terminates. Since all we have then done is row operations, the resulting matrix must have the same row space as the input. Termination follows directly from Lemma 9 since the value of a row decreases each time a row reduction is performed. We will be more precise, though. Denote by $\boldsymbol{V}$ the input to the algorithm and by $\boldsymbol{U}$ the output in weak Popov form, and let $\boldsymbol{u}_i, \boldsymbol{v}_i$ denote the rows of these matrices. Then $\mathrm{rowdeg}\,(\boldsymbol{U}) = \deg\det(\boldsymbol{U}) = \deg\det(\boldsymbol{V})$ and the $\mathrm{LP}(\boldsymbol{u}_i)$ are all different. The total number of row reductions is then upper bounded by:

$$\begin{aligned}&\textstyle\sum_{i=1}^{m}\big(\psi(\boldsymbol{v}_i) - \psi(\boldsymbol{u}_i)\big)\\ &\leq m(\mathrm{rowdeg}\,(\boldsymbol{V}) - \mathrm{rowdeg}\,(\boldsymbol{U})) + \textstyle\sum_{i=1}^{m}(\mathrm{LP}(\boldsymbol{u}_i) - \mathrm{LP}(\boldsymbol{v}_i))\\ &\leq m\Delta(\boldsymbol{V}) + \textstyle\sum_{i=1}^{m}(\mathrm{LP}\boldsymbol{v}_i - i)\end{aligned}$$

For the asymptotic complexity, note that during the algorithm, no polynomial in the intermediate matrix will have larger degree than $\deg \boldsymbol{V}$. One row reduction consists of m times scaling and adding two such polynomials. □

For generic square matrices $\boldsymbol{V}$ the orthogonality defect is $O(m \deg \boldsymbol{V})$; therefore the above complexity estimate relaxes to the same as in [10]. However, we have:

COROLLARY 11. *Solving an unweighted 2D Padé approximation by applying the Mulders–Storjohann algorithm to $\boldsymbol{M}$ has complexity $O(\rho(\rho+\sigma)^2 \max_j\{\deg G_j^2\})$.*

Alekhnovich showed in [1] how to accelerate the Mulders–Storjohann with fast multiplication. His algorithm calculates the same row reductions, but they are bundled and structured in a binary tree which allows matrix multiplication techniques to reduce the quadratic dependence on $\deg \boldsymbol{V}$. The reported complexity is $O^\sim(\mathrm{M}(m)\mathrm{P}(m \deg \boldsymbol{V}))$. However, by the observations on orthogonality defect as above, this can be immediately improved to $O^\sim(\mathrm{M}(m)\Delta(\boldsymbol{V}))$. More details can be found in [11]. We therefore get the improved complexity as given in Table 2.

For $\rho \in \omega(1)$, this is still asymptotically worse than the fast algorithms GJV and Zhou–Labahn. However, $\rho \in O(1)$ is still an interesting special case, including e.g. multi-sequence linear-feedback register synthesis, and here one would need a deeper analysis and/or experimentation to determine which algorithm is fastest.

4. THE DEMAND–DRIVEN ALGORITHM

We will show how Mulders–Storjohann admits a wholly different variant, but only when applied to 2D Padé approximations. This variant, which we call the Demand–Driven algorithm, is an improvement when ρ is small and when all or most G_j are sparse, i.e. have few non-zero coefficients.

The intuitive idea is very simple: for any basis of $\mathcal{M}$, it is basically sufficient to keep track of only its ρ first columns, as the remaining columns can be computed from these. By keeping track of the leading position for each row, one then

computes the other entries only when the need arise. Formalising and proving the resulting algorithm is unfortunately somewhat technical. The result is Algorithm 2.

Algorithm 2 Demand-Driven algorithm

Input: $M = \left(\begin{array}{c|c} \boldsymbol{I} & \boldsymbol{S} \\ \hline 0 & \operatorname{diag}(G_1, \ldots, G_\sigma) \end{array}\right)$.
Let $\boldsymbol{m}_i$ be the rows of this matrix.

Output: The first ρ columns of a basis of $\mathcal{M}$ in weak Popov form.

1 $\boldsymbol{\Lambda}$ = first ρ columns of M. Denote its rows by $\boldsymbol{\lambda}_i$.
2 $\big(\theta_i, \alpha_i\big) = \big(\deg(\boldsymbol{m}_i), \mathrm{LC}(\mathrm{LT}(\boldsymbol{m}_i))\big)$ for $i = 1, \ldots, \rho + \sigma$
3 $W = \big\{(i, \mathrm{LP}(\boldsymbol{m}_i)) \mid 1 \le i \le \rho \ \wedge\ \mathrm{LP}(\boldsymbol{m}_i) \neq i\big\}$
4 **while** $W \neq \emptyset$ **do**
5 $\quad(i, h) = \mathsf{pop}(W)$
6 $\quad$**if** $\theta_i < \theta_h$ **then** swap $(\boldsymbol{\lambda}_i, \alpha_i, \theta_i)$ and $(\boldsymbol{\lambda}_h, \alpha_h, \theta_h)$
7 $\quad\boldsymbol{\lambda}_i = \boldsymbol{\lambda}_i - \frac{\alpha_i}{\alpha_h} x^{\theta_i - \theta_h} \boldsymbol{\lambda}_h$
8 $\quad$**repeat**
9 $\quad\quad(\theta_i, h) = \mathsf{previous}(\theta_i, h)$
10 $\quad\quad\alpha_i =$ coefficient to x^{θ_i} in
11 $\quad\quad\quad\begin{cases} \lambda_{i,h} & \text{if } h \le \rho \\ \sum_{j=1}^{\rho} \lambda_{i,j} S_{j,h-\rho} \mod G_{h-\rho} & \text{otherwise} \end{cases}$
12 $\quad$**until** $\alpha_i \neq 0$
13 $\quad$**if** $i \neq h$ **then**
14 $\quad\quad$**if** $(h, j) \in W$ for some j **then**
15 $\quad\quad\quad$swap $(\boldsymbol{\lambda}_i, \alpha_i, \theta_i)$ and $(\boldsymbol{\lambda}_h, \alpha_h, \theta_h)$
16 $\quad\quad\quad$replace (h, j) with (i, j) in W.
17 $\quad\quad$**else**
18 $\quad\quad\quad\mathsf{push}(W, (i, h))$
19 **return** $\boldsymbol{\Lambda}$

To begin, we will overload ψ to $\mathbb{N}_0 \times \{1, \ldots, \rho + \sigma\} \to \mathbb{N}_0$ by $\psi(\theta, h) = (\rho + \sigma)\theta + h$, i.e. for any non-zero $\boldsymbol{v} \in \mathbb{F}[x]^{\rho+\sigma}$, $\psi(\boldsymbol{v}) = \psi(\deg \boldsymbol{v}, \mathrm{LP}(\boldsymbol{v}))$. We will also use the helper function $\mathsf{previous}(\theta, h)$ which gives the degree and leading position for the "previous" possible value, that is:

$$\mathsf{previous}(\theta, h) = \begin{cases} (\theta, h-1) & \text{if } h > 1 \\ (\theta - 1, \rho + \sigma) & \text{if } h = 1 \end{cases}$$

We will prove the correctness of Algorithm 2 by showing that the computations correspond to a possible run of a slight variant of Mulders–Storjohann; first we need a technical lemma on this variant:

Lemma 12. *Consider the Mulders–Storjohann with input $\boldsymbol{M}$. Consider now a variant of the algorithm where we, when replacing some $\boldsymbol{v}_j$ with $\boldsymbol{v}'_j$ in a row reduction, instead replace it with*

$$\boldsymbol{v}''_j = (v'_{j,1}, \ldots, v'_{j,\rho}, v'_{j,\rho+1} \bmod G_1, \ldots, v'_{j,\rho+\sigma} \bmod G_\sigma)$$

This does not change correctness of the algorithm or the upper bound on the number of row reductions performed.

Proof. Correctness follows if we can show that each of the σ modulo reductions could have been achieved by a series of $\mathbb{F}[x]$ row operations on the current matrix $\boldsymbol{V}$ after the row reduction producing $\boldsymbol{v}'$, since then $\boldsymbol{V}$ would remain a basis of $\mathcal{M}$.

Consider the modulo reduction on the $(\rho + h)$th position for some h. This could be achieved by adding a multiple of the vector $\boldsymbol{g}_h = (0, \ldots, 0, G_h, 0, \ldots, 0)$, with position $\rho + h$ non-zero, to $\boldsymbol{v}'$. Since $\boldsymbol{g}_h$ is a row in $\boldsymbol{M}$, then as long as this has not yet been row reduced, the $(\rho + h)$th position reduction is allowed. Notice that if $\psi(\boldsymbol{v}') < \psi(\boldsymbol{g}_h)$ then the modulo reduction on the $(\rho + h)$th position is void.

Introduce now a loop invariant involving $J_h = \{\boldsymbol{g}_h\}$, a subset of the current rows in $\boldsymbol{V}$ having two properties: that $\boldsymbol{g}_h$ can be constructed as an $\mathbb{F}[x]$-linear combination of the rows in J_h; and that each $\boldsymbol{v} \in J_h$ has $\psi(\boldsymbol{v}) \le \psi(\boldsymbol{g}_h)$. After row reductions on rows not in J_h, the $(\rho + h)$th modulo reduction is therefore allowed, since $\boldsymbol{g}_h$ can be constructed by the rows in J_h. On the other hand, after a row reduction on a row $\boldsymbol{v} \in J_h$ by some $\boldsymbol{v}_k$ resulting in $\boldsymbol{v}'$, the hth modulo reduction has no effect since $\psi(\boldsymbol{v}') < \psi(\boldsymbol{v}) \le \psi(\boldsymbol{g}_h)$. Afterwards, J_h is updated as $J_h = J_h \setminus \{\boldsymbol{v}\} \cup \{\boldsymbol{v}', \boldsymbol{v}_k\}$ and the loop invariant is kept since $\psi(\boldsymbol{v}_k) \le \psi(\boldsymbol{v})$.

Since $\psi(\boldsymbol{v}''_j) \le \psi(\boldsymbol{v}'_j)$ the proof of Lemma 10 shows that the number of row reductions performed is not greater than in the original Mulders–Storjohann. □

We will say for a matrix $\boldsymbol{U}$ that there is a "collision on (i, j)" if one could perform a row reduction involving $\boldsymbol{u}_j$ and $\boldsymbol{u}_i$, i.e. $i \neq j$, $\mathrm{LP}(\boldsymbol{u}_i) = \mathrm{LP}(\boldsymbol{u}_j)$; we will also say that these rows are "involved in a collision". Now the proof of correctness; it essentially just establishes that W is a "worklist" containing indices of rows involved in a collision, which means we should at some point row reduce it. All indices i not in W have been "parked" such that $\mathrm{LP}(\boldsymbol{v}_i) = i$, and they will only be used further in the algorithm if there is a row named in W with leading position i.

Theorem 13. *Algorithm 2 is correct.*

Proof. Let $\boldsymbol{V}$ be the matrix continually changing in the variant of Mulders–Storjohann described in Lemma 12. For notational convenience, and simplicity in the description of Algorithm 2, we will consider a further variant of Mulders–Storjohann where we sometimes swap two rows, to be described momentarily.

Let us write $(i, \star) \notin W$ to mean $\neg\exists j.(i, j) \in W$, and similarly for $(\star, i) \notin W$. We will then demonstrate that each iteration of Algorithm 2 corresponds to exactly one row reduction on $\boldsymbol{V}$, and that the following loop invariants hold:

1. $\boldsymbol{\Lambda}$ is the first ρ columns of $\boldsymbol{V}$;
2. $\alpha_i x^{\theta_i}$ is the leading monomial of $\mathrm{LT}(\boldsymbol{v}_i)$;
3. If there is a collision (i, j) then $\exists k.(i, k) \in W \vee (j, k) \in W$;
4. If $(i, h) \in W$ then $i \neq h$ and $\mathrm{LP}(\boldsymbol{v}_i) = \mathrm{LP}(\boldsymbol{v}_h) = h$;
5. For any i, there is at most one pair in W with first position i.
6. If $(i, \star) \notin W$ then $\mathrm{LP}(\boldsymbol{v}_i) = i$;

The invariants are confirmed to be true after initialisation; assume now they are true on entry of an iteration, and we will show they are true on exit. Once Algorithm 2 terminates, by Invariant 3 then there are no collisions, so $\boldsymbol{V}$ is in weak Popov form, so by Invariant 1 the result is correct.

If $W \neq \emptyset$, then by Invariant 4 there is a collision (i, h), and the considered variant of Mulders–Storjohann could have

chosen to perform a row reduction on this. Invariant 4 implies that $(\star, i) \notin W$ and $(h, \star) \notin W$, and that Invariant 5 then implies that no other element in W contains i.

Now comes the promised swap in our Mulders–Storjohann variant: we will possibly perform a swap of rows of $\boldsymbol{V}$ such that the row to be reduced is $\boldsymbol{v}_i$; i.e. if $\deg \boldsymbol{v}_i < \deg \boldsymbol{v}_j$ we swap, which is exactly when $\theta_i < \theta_j$ in Line 6. We note that the Invariants 1 and 2 are maintained since everything involved is swapped. The remaining invariants never distinguish the two swapped rows, which means all non-violations and violations are kept intact. Thus for the remainder of the proof, we can assume that no swap was performed since this will be indistinguishable from the other case.

In Line 7 we perform the actual row reduction, and Invariant 1 is seen to be maintained, due to Invariant 2. The ensuing loop is for maintaining Invariant 2: it will go through possible degrees and leading positions of the updated row $\boldsymbol{v}_i$ in descending order of ψ until it finds one with non-zero coefficient. Note that $\psi(\boldsymbol{v}_i') < \psi(\boldsymbol{v}_i)$, so the loop initiates correctly. The calculation in Line 11 is also exactly correct for the considered variant of Mulders–Storjohann, cf. Lemma 12.

Refer to $\boldsymbol{V}'$ with rows $\boldsymbol{v}_i'$ as $\boldsymbol{V}$ after this row operation, and refer to h' as the updated value of h. Thus $\mathrm{LP}(\boldsymbol{v}_i') = h'$. The last thing is then to maintain Invariants 3–6. Note that all possible violations must involve i since only for i we have $\boldsymbol{v}_i' \neq \boldsymbol{v}_i$. We distinguish the three cases of the if-statements:

- $i = h'$: Then $\boldsymbol{v}_i'$ cannot be involved in any collision: for if there was a $j \neq i$ such that (j, i) is a collision in $\boldsymbol{V}'$, then $\mathrm{LP}(\boldsymbol{v}_j) = \mathrm{LP}(\boldsymbol{v}_j') = \mathrm{LP}(\boldsymbol{v}_i') = i$ which means that $(j, i) \notin W$ due to Invariant 4 holding on entry of the iteration (at which time $\mathrm{LP}(\boldsymbol{v}_i) \neq i$); but then $(j, \star) \notin W$ which means $\mathrm{LP}(\boldsymbol{v}_j) = j$ by Invariant 6. Thus, we must have $j = h' = i$ but this was false by assumption. Since then $\boldsymbol{v}_i'$ is involved in no collisions, the remaining invariants are seen to hold since: all possible violations to them must involve i; since i is no longer in any pair of W; and since $\mathrm{LP}(\boldsymbol{v}_i') = i$.
- $i \neq h'$ and $\exists j.(h', j) \in W$: That means $(\star, h') \notin W$ by Invariant 4 at iteration entry, and since $(\star, i) \notin W$, we do not create violations to the invariants by swapping $\boldsymbol{v}_i'$ and $\boldsymbol{v}_{h'}'$ when also replacing (h', j) with (i, j) in W. After this swap, we have $\mathrm{LP}(\boldsymbol{v}_{h'}') = h'$, and invariants involving i all hold. There now can't be any collision involving $\boldsymbol{v}_{h'}'$: for if there was a $k \neq h'$ with $\mathrm{LP}(\boldsymbol{v}_k') = \mathrm{LP}(\boldsymbol{v}_{h'}') = h'$, then also $\mathrm{LP}(\boldsymbol{v}_k) = h'$. But at iteration entry, by Invariant 6, since $k \neq h'$ then $(k, \star) \in W$, which means by Invariant 4 that $(k, h') \in W$ and $\mathrm{LP}(\boldsymbol{v}_{h'}) = h'$. This contradicts $\mathrm{LP}(\boldsymbol{v}_{h'}) = j$ from $(h', j) \in W$. Summarily, since we then have removed the only element in W involving h' and since $\mathrm{LP}(\boldsymbol{v}_{h'}') = h'$, the invariants are again maintained.
- $i \neq h'$ and $(h', \star) \notin W$: By Invariant 6, we have $\mathrm{LP}(\boldsymbol{v}_{h'}) = h'$ and so (i, h') is a collision, meaning the invariants are maintained by adding (i, h') to W.

□

The algorithm is very straight-forward to analyse, and we can even include the leading constant:

PROPOSITION 14. *Algorithm 2 performs at most*

$$2(\rho+1)^2(\rho+\sigma)\gamma \mathrm{P}_G(\gamma) + O\big(\rho(\rho+\sigma)\gamma\big)$$

field multiplications, where $\gamma = \max_j\{\deg G_j\}$, *and* $\mathrm{P}_G(\gamma)$ *is the cheaper of* $\mathrm{P}(\gamma)$ *and* $\gamma \max_j\{\#\mathrm{supp}(G_j)\}$. *It uses* $2\rho(\rho+\sigma)\gamma + O((\rho+\sigma)\gamma)$ *field elements of memory.*

PROOF. Notice that the complexity of executing Line 11 is at most $(\rho+1)\mathrm{P}_G(\gamma)$: for we can compute this one coefficient either by ρ multiplications and one division of polynomials of degree at most γ; or we can in each product $\lambda_{i,j} S_{j,h-\rho}$ compute only the $\mathrm{supp}(G_{h-\rho})$ single coefficients influencing the x^{θ_i}-coefficient of the remainder.

Each iteration through the repeat loop will decrease the upper bound on $\psi(\boldsymbol{v}_i)$, so by arguments exactly like those of the proof of Lemma 10, the total number of times through the repeat loop is at most the number of row reductions by Mulders–Storjohann, i.e. $(\rho+\sigma)(\Delta(M)+\rho)$. Apart from the Line 11, only Line 7 is not $O(1)$, and this has complexity at most $O(\rho\ \deg(M)) \subset \rho \mathrm{P}_G(\gamma)$. Multiplying the number of iterations with the cost per iteration yields the result. The main memory requirement is for $\boldsymbol{\Lambda}$ and $\boldsymbol{S}$, and otherwise only asymptotically lesser amounts. □

REMARK 15. Algorithm 2 bears a striking resemblance to the Berlekamp–Massey algorithm [4] and various generalisation, e.g. for synthesising multi-sequence LFSRs [16], or for solving Roth-Ruckenstein "key equations" [15]. The Mulders–Storjohann algorithm can be considered a generalisation of the Extended Euclidean algorithm, so this resemblance generalise the long-known correspondence between the Euclidean algorithm and the Berlekamp–Massey [5,7] to the much more general case of 2D Padé approximations.

5. ADDING WEIGHTS

We now study the weighted variant of the problem. We are therefore again seeking $\boldsymbol{s} = (\Lambda_1, \ldots, \Lambda_\rho, \Omega_1, \ldots, \Omega_\sigma) \in \mathbb{F}[x]^{\rho+\sigma}$ such that the congruence (2) holds, but now such that $\max_{i,j}\{\nu \deg \Lambda_i + \eta_i,\ \nu \deg \Omega_j + \mu_j\}$ is minimised. If we are solving an asymmetric approximation, we furthermore wish that $\max_i\{\nu \deg \Lambda_i + \eta_i\} > \max_j\{\nu \deg \Omega_j + \mu_j\}$.

We will introduce two injective mappings for matrices such that finding a weak Popov form of the image of $\boldsymbol{M}$ under either will solve the weighted approximation problem. The first is a straightforward embedding of the weights but carries a computational penalty if one can not handle it specifically within the row reduction algorithm. The second we derive from the first to mitigate this problem.

First the straightforward map $\Phi : \mathbb{F}[x]^{\rho+\sigma} \mapsto \mathbb{F}[x]^{\rho+\sigma}$:

$$\Phi\big((v_1, \ldots, v_{\rho+\sigma})\big) = \big(x^{\eta_1} v_1(x^\nu), \ldots, x^{\eta_\rho} v_\rho(x^\nu), \\ x^{\mu_1} v_{\rho+1}(x^\nu), \ldots, x^{\mu_\sigma} v_{\rho+\sigma}(x^\nu)\big)$$

Extend Φ element-wise to sets of vectors, and extend Φ row-wise to $(\rho+\sigma) \times (\rho+\sigma)$ matrices such that the ith row of $\Phi(\boldsymbol{V})$ is $\Phi(\boldsymbol{v}_i)$, where $\boldsymbol{v}_i$ are the rows of $\boldsymbol{V}$. Note that $\Phi(\mathcal{M})$ is a free $\mathbb{F}[x^\nu]$-module of dimension $\rho+\sigma$, and that any basis of it is by Φ^{-1} sent back to a basis of $\mathcal{M}$.

PROPOSITION 16. *A minimal solution to the weighted 2D Padé approximation problem is the* Φ^{-1}*-image of a vector in* $\Phi(\mathcal{M})$ *with minimal degree, and if the approximation is asymmetric, of only those vectors with* LP *at most* ρ.

PROOF. This follows immediately since for any vector $\boldsymbol{v} = (v_1, \ldots, v_{\rho+\sigma}) \in \mathbb{F}[x]^{\rho+\sigma}$, then $\deg \Phi(\boldsymbol{v}) = \max_{i,j}\{\nu \deg v_i + \eta_i,\ \nu \deg v_{\rho+j} + \mu_j\}$, and $\mathrm{LP}(\Phi(\boldsymbol{v})) \leq \rho \iff \max_i\{\nu \deg v_i + \eta_i\} > \max_j\{\nu \deg v_{\rho+j} + \mu_j\}$. □

The above proposition together with Proposition 5 therefore immediately gives a method for solving the weighted approximation problems: simply find a weak Popov form[3] of $\Phi(\boldsymbol{M})$, where $\boldsymbol{M}$ is from Proposition 1. However, since $\deg \Phi(\boldsymbol{M}) \geq \nu \deg \boldsymbol{M}$ we should expect a penalty in performance of at least a factor ν from doing this. To mitigate this problem, we now derive a second mapping. It can already be mentioned, though, since we will need it for the Demand–Driven algorithm later, that the Mulders–Storjohann can be implemented to not suffer this ν penalty, even under the direct Φ-mapping.

For notational convenience, let $\boldsymbol{w} = (w_1, \ldots, w_{\rho+\sigma}) = (\eta_1, \ldots, \eta_\rho, \mu_1, \ldots, \mu_\sigma)$. For the improved mapping, consider first the permutation π of $[1, \ldots, m]$ indirectly defined by

$$\begin{aligned}\pi(i) > \pi(j) \iff\ & (w_i \bmod \nu) > (w_j \bmod \nu) \\ & \vee \big((w_i \bmod \nu) = (w_j \bmod \nu) \wedge i > j\big)\end{aligned}$$

This is seen to be well-defined. Extend now π to elements of $\mathbb{F}[x]^{\rho+\sigma}$ such that π permutes the positions of such vectors. Our desired mapping is now Ψ:

$$\Psi\big((v_1, \ldots, v_{\rho+\sigma})\big) = \pi\big((x^{\lfloor w_1/\nu \rfloor} v_1, \ldots, x^{\lfloor w_{\rho+\sigma}/\nu \rfloor} v_{\rho+\sigma})\big)$$

Proposition 17. *For any $\boldsymbol{v} \in \mathbb{F}[x]^{\rho+\sigma}$ then*

$$(\pi^{-1} \circ \mathrm{LP} \circ \Psi)(\boldsymbol{v}) = (\mathrm{LP} \circ \Phi)(\boldsymbol{v})$$

Proof. Let v_i be the elements of $\boldsymbol{v}$, and $h = (\mathrm{LP} \circ \Phi)(\boldsymbol{v})$. We will prove that no index but $\pi(h)$ can be $(\mathrm{LP} \circ \Psi)(\boldsymbol{v})$. Consider first some $j > h$. By the definition of h then

$$\nu \deg v_h + w_h > \nu \deg v_j + w_j, \quad \text{i.e.} \qquad (3)$$
$$\deg v_h + \lfloor w_h/\nu \rfloor + \tfrac{w_h \bmod \nu}{\nu} > \deg v_j + \lfloor w_j/\nu \rfloor + \tfrac{w_j \bmod \nu}{\nu}$$

So either $\deg v_h + \lfloor w_h/\nu \rfloor > \deg v_j + \lfloor w_j/\nu \rfloor$, or they are equal and $w_h \bmod \nu > w_j \bmod \nu$. In the first case, then clearly $\pi(j)$ can't be $(\mathrm{LP} \circ \Psi)(\boldsymbol{v})$ due to degrees. In the second case the degrees of $\Psi(\boldsymbol{v})$ at positions $\pi(h)$ and $\pi(j)$ are tied, but we have $\pi(h) > \pi(j)$, which means that $\pi(j)$ can't be the LP.

Consider now some $j < h$, so we have the same inequality (3) but with $>$ replaced by $\geq$. If sharp inequality really holds, then we can continue as before, so assume instead that equality holds. That implies both $\deg v_h + \lfloor w_h/\nu \rfloor = \deg v_j + \lfloor w_j/\nu \rfloor$ and $w_h \equiv w_j \mod \nu$. So the degrees of $\Psi(\boldsymbol{v})$ at positions $\pi(h)$ and $\pi(j)$ are tied, but then since $h > j$, we have $\pi(h) > \pi(j)$. Again, $\pi(j)$ is not the LP. □

Corollary 18. *For any $\boldsymbol{V} \in \mathbb{F}[x]^{\rho+\sigma}$, then $\Phi(\boldsymbol{V})$ is in weak Popov form if and only if $\Psi(\boldsymbol{V})$ is in weak Popov form.*

The algorithm is then clear: to solve the weighted 2D Padé approximation problem, simply compute a weak Popov form of $\Psi(\boldsymbol{M})$. The row with minimal degree—and in the asymmetric case, among only those with leading position among $\pi(1), \ldots, \pi(\rho)$—is the solution. This works immediately for the general row reduction matrices. The following lemma can be used for calculating the resulting complexities:

[3]Since $\Phi(\mathcal{M})$ is an $\mathbb{F}[x^\nu]$-module, and not $\mathbb{F}[x]$, then it is not immediately obvious that computing a weak Popov form of $\Phi(\boldsymbol{M})$ will not result in a matrix with rows outside $\Phi(\mathcal{M})$. However this turns out to not be the case, which can easily be proved for Mulders–Storjohann and then generalised, see [12, Proposition 2.26]

Lemma 19.

$$\deg \Psi(\boldsymbol{M}) = \gamma \qquad \Delta(\Psi(\boldsymbol{M})) \leq \deg \Psi([\boldsymbol{I} \mid \boldsymbol{S}]) \leq \rho\gamma$$

where $\gamma = \max_{i,j}\{\lfloor \eta_i/\nu \rfloor,\ G_j + \lfloor \mu_j/\nu \rfloor\}$.

Proof. The degree is obvious. For the orthogonality defect, the determinant follows from $\Psi(\boldsymbol{M})$ having been an upper triangular matrix before the column permutations. □

5.1 Handling weights Demand–Drivenly

For Algorithm 2, however, the permutations performed by Ψ would make it messy to describe. Instead, we are going to use the more direct mapping Φ, and prove that for this algorithm, there is no performance penalty for ν.

Only certain ψ-values are possible for the vectors of $\Phi(\mathcal{M})$, so first the definition of previous needs to be generalised to still provide the degree and leading position of the previous *possible* value:

$$\begin{aligned}\mathsf{previous}(\theta, h) = \arg\max_{\theta', h'}\{\psi(\theta', h') \mid\ & \psi(\theta', h') < \psi(\theta, h) \\ & \wedge\ \theta' \equiv w_{h'} \mod \nu\}\end{aligned}$$

Just as with π, this function is easily calculable though it is cumbersome to write down a direct description.

The Mulders–Storjohann with input $\Phi(\boldsymbol{M})$ exhibits structurally the same behaviour as with input $\boldsymbol{M}$; it is therefore not surprising that we can make a demand–driven recasting for this case also. In fact, apart from the more general definition of previous, this recast is *exactly* as in Algorithm 2 except for the following details:

- Input is now $\Phi(\boldsymbol{M})$ and the $\boldsymbol{m}_i$ are its rows.
- The output of the algorithm is now the first ρ columns of a basis of $\Phi(\mathcal{M})$ in weak Popov form.
- In Line 11, the polynomial from which to choose the coefficient to x^{θ_i} for the case $h > \rho$ is now:

$$\textstyle\sum_{j=1}^{\rho} x^{-\eta_j} \lambda_{i,j} \tilde{S}_{j,h-\rho} \bmod \tilde{G}_{h-\rho},$$

$$\tilde{S}_{j,h-\rho} = x^{\mu_h} S_{j,h-\rho}(x^\nu) \text{ and } \tilde{G}_{h-\rho} = x^{\mu_h} G_{h-\rho}(x^\nu).$$

The proof that this amended algorithm works correctly is exactly like in Theorem 13: first we get a statement equivalent to Lemma 12 but for $\Phi(\boldsymbol{M})$, and then all the invariants used in Theorem 13 are again shown to be fulfilled. The change to Line 11 is correct since the $\boldsymbol{V}$ will at position (i, h) have exactly this element.

For complexity, we first need to bound the number of iterations through the loop; this is done by bounding how many row reductions Mulders–Storjohann with input $\Phi(\boldsymbol{M})$ would do. The fact that only certain ψ-values are possible implies that this number is roughly $1/\nu$'th of what the degree of $\Phi(\boldsymbol{M})$ would have us think:

Lemma 20. *The number of row reductions performed by Mulders–Storjohann on $\Phi(\boldsymbol{M})$ is less than $\rho(\rho+\sigma)(\gamma+1)$ where γ is as in Lemma 19.*

Proof. Observe that for any $\boldsymbol{v} \in \mathcal{M}$, if we let $h = \mathrm{LP}(\Phi(\boldsymbol{v}))$ we have

$$\begin{aligned}\psi(\Phi(\boldsymbol{v})) &= (\rho+\sigma)(\nu \deg v_h + w_h) + h \\ &\equiv (\rho+\sigma) w_h + h \mod (\rho+\sigma)\nu\end{aligned}$$

That is, on any given interval of size $(\rho+\sigma)\nu$, then $\psi(\Phi(\boldsymbol{v}))$ can attain at most $\rho+\sigma$ of the values, depending on the

value of $\mathrm{LP}(\Phi(\boldsymbol{v}))$. Continuing as in the proof of Lemma 10, the number of row reductions is at most $\sum_{i=1}^{\rho+\sigma} \nu^{-1}(\psi(\boldsymbol{v}_i) - \psi(\boldsymbol{u}_i))$, which simplifies to the statement. □

PROPOSITION 21. *The amended Algorithm 2 on input $\Phi(\boldsymbol{M})$ performs at most*

$$2(\rho+1)^2(\rho+\sigma)\gamma \mathrm{P}_G(\gamma) + O\big(\rho(\rho+\sigma)\gamma\big)$$

field multiplications, where γ is as in Lemma 19, and $\mathrm{P}_G(\gamma)$ is the cheaper of $\mathrm{P}(\gamma)$ and $\gamma \max_j\{\#\mathrm{supp}(G_j)\}$. It uses $2\rho(\rho+\sigma)\gamma + O((\rho+\sigma)\gamma)$ field elements of memory.

PROOF. As in the proof of Proposition 14, then Lemma 20 gives an upper bound on the number of iterations of the main loop as well as executions of Line 11. This critical line costs $\mathrm{P}_G(\gamma)$ by exactly the same arguments as in Proposition 14 combined with the observation that all the involved polynomials are sparse with only every ν coefficient possibly non-zero; therefore computations with them cost as if their degree was only $1/\nu$'th. □

6. CONCLUSION

We introduced the 2D Padé approximations, a general form of approximations for polynomials over fields which includes several earlier types as special cases. Inspired by various applications we introduce two *types* of these: the traditional "symmetric" Padé approximation, and the "asymmetric" form. We furthermore consider both types in a weighted form, where the priority of the various terms of an approximation can essentially be "shifted" by rational numbers.

Using off-the-shelf row reduction algorithms, we showed how these approximations can be solved efficiently. It speaks for the naturalness of the approximation's definition that the matrices to be row reduced have low dimension; lower than had we solved them using one of the already studied, general approximation types matrix-Padé or order bases.

The Mulders–Storjohann is such a row reduction algorithm, and we showed that one can in general describe its complexity based on the orthogonality defect of the input matrix. As a corollary, both it and its divide & conquer variant by Alekhnovich performs faster than initially expected on 2D Padé approximations.

In any basis of the lattice induced by a 2D Padé approximation problem, if it is seen as a matrix, there is a large redundant part. With this observation, we recast the Mulders–Storjohann to perform the computations on demand. This yields a "Demand–Driven" algorithm which often has lower time and memory complexity. This algorithm has been implemented in Sage [17] and is available at `http://jsrn.dk/codinglib`

For future work, it seems possible that when applying the Mulders–Storjohann for solving asymmetric 2D Padé approximation, one can carefully order the row reductions in such a way that once a solution appears, this must be minimal, and one can stop the computation. For the Demand–Driven algorithm, this would be particularly interesting, since one could then bound the complexity by the degree of the smallest solution, instead of the degree of the input matrix.

7. REFERENCES

[1] M. Alekhnovich. Linear diophantine equations over polynomials and soft decoding of Reed–Solomon codes. *IEEE Trans. Inf. Theory*, 51(7):2257–2265, July 2005.

[2] G. Baker and P. Graves-Morris. *Padé approximants*, volume 59. Cambridge Univ. Press, 1996.

[3] B. Beckermann and G. Labahn. A uniform approach for the fast computation of matrix-type padé approximants. *SIAM J. Matr. Anal. Appl.*, 15(3):804–823, 1994.

[4] E. R. Berlekamp. *Algebraic Coding Theory*. Aegean Park Press, 1968.

[5] J. Dornstetter. On the equivalence between berlekamp's and euclid's algorithms. *IEEE Trans. Inf. Theory*, 33(3):428–431, 1987.

[6] G.-L. Feng and K. K. Tzeng. A generalization of the berlekamp-massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes. *IEEE Trans. Inf. Theory*, 37(5):1274–1287, 1991.

[7] P. Fitzpatrick. On the key equation. *IEEE Trans. Inf. Theory*, 41(5):1290–1302, 1995.

[8] P. Giorgi, C. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *Proc. of ISSAC*, page 135–142, 2003.

[9] A. Lenstra. Factoring multivariate polynomials over finite fields. *J. Comp. Syst. Sc.*, 30(2):235–248, 1985.

[10] T. Mulders and A. Storjohann. On lattice reduction for polynomial matrices. *J. Symb. Comp.*, 35(4):377–401, 2003.

[11] J. S. R. Nielsen. Generalised multi-sequence shift-register synthesis using module minimisation. In *Proc. of IEEE ISIT*, 2013.

[12] J. S. R. Nielsen. *List Decoding of Algebraic Codes*. PhD thesis, Technical University of Denmark, 2013. Available at jsrn.dk.

[13] J. S. R. Nielsen and A. Zeh. Multi-trial guruswami–sudan decoding for generalised reed–solomon codes. In *Proc. of WCC*, 2013.

[14] R. Roth. *Introduction to Coding Theory*. Cambridge Univ. Press, 2006.

[15] R. Roth and G. Ruckenstein. Efficient decoding of reed–solomon codes beyond half the minimum distance. *IEEE Trans. Inf. Theory*, 46(1):246 –257, 2000.

[16] V. Sidorenko and G. Schmidt. A linear algebraic approach to multisequence shift-register synthesis. *Problems of Information Transmission*, 47(2):149–165, 2011.

[17] W. A. Stein et al. *Sage Mathematics Software*. http://www.sagemath.org.

[18] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa. A method for solving key equation for decoding goppa codes. *Information and Control*, 27(1):87–99, 1975.

[19] A. Zeh, C. Gentner, and D. Augot. An interpolation procedure for list decoding reed-solomon codes based on generalized key equations. *IEEE Trans. Inf. Theory*, 57(9):5946–5959, 2011.

[20] W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *J. Symb. Comp.*, 47(7):793–819, July 2012.