Max-Planck-Institut für Mathematik in den Naturwissenschaften Leipzig

Solving p-adic Polynomial Systems via Iterative Eigenvector Algorithms.

by

Avinash Kulkarni

Preprint no.: 54 2019

SOLVING P-ADIC POLYNOMIAL SYSTEMS VIA ITERATIVE EIGENVECTOR ALGORITHMS

AVINASH KULKARNI

ABSTRACT. In this article, we describe an implementation of a polynomial system solver to compute the approximate solutions of a 0-dimensional polynomial system with finite precision p-adic arithmetic. We also describe an improvement to an algorithm of Caruso, Roe, and Vaccon for calculating the eigenvalues and eigenvectors of a p-adic matrix.

1. INTRODUCTION

Let k be a field and let $f_1, \ldots, f_m \in k[x_1, \ldots, x_n]$ be polynomials such that the variety defined by $f_1, \ldots, f_m$ has dimension 0. It is often of interest to compute the solutions of such a system, either exactly or approximately. A significant bottleneck in computing the exact solutions of such a polynomial system is the complexity of the field extension required to write down all of the solutions. A popular method to compute the solutions exactly is to compute a triangular decomposition for the ideal $\langle f_1, \ldots, f_m \rangle$ and solve for the coordinates via back-substitution. A difficulty with this approach is the explosion of the size of the coefficients, which in conjunction with taking complicated field extensions, usually renders a computer algebra system unresponsive.

Computations in an exact field (such as $\mathbb{Q}$) can often be approximated by computations in an inexact field (such as $\mathbb{R}, \mathbb{C}$), where by inexact, we mean that the computer representation of an element is only an approximation up to some precision. It is often far faster to compute an approximation to the solutions, though the trade-off is that the solutions are not known perfectly. In numerous applications, a real or complex approximation to the solutions is sufficient.

In Linear Algebra, p-adic methods go back to at least Dixon [Dix82]. One advantage pointed out by Dixon is that a linear system can be ill-conditioned with respect to an archimedean norm, but well-conditioned p-adically; general polynomial systems exhibit this behavior as well. Recently, new developments in number theory and tropical geometry highlight that non-archimedean data is not just a route to integral solutions, but is significant in and of itself.

Our polynomial system solver is based on the truncated normal form solver of [vBMT18]. To transplant their implementation to the p-adic setting, we rely on finite precision linear algebra over the p-adic field, which in analogy to the case over $\mathbb{R}$ we refer to as *pnumerical linear algebra*. It is essential for us to be able to compute the eigenvalues and eigenvectors of an approximate p-adic matrix quickly and stably. The naive algorithm to compute the eigenvalues of a matrix relies on the calculation and factorization of the characteristic polynomial. In the numerical setting (i.e, over $\mathbb{R}$) it is desirable to avoid this step due to the instability of solving for the roots of a polynomial. There is a similar loss of precision in solving for the roots of a p-adic polynomial when the roots of the polynomial are p-adically close together. Unlike the archimedean setting, the accurate calculation of the characteristic polynomial is more difficult since a division free algorithm must be used [CRV17, Introduction]. Our idea to improve on the algorithm of [CRV17] is to adapt the ideas from classical numerical linear algebra to the pnumerical setting; specifically to use an iterative scheme to compute the eigenvectors or eigenvalues. To our knowledge, this is the first appearance of a p-adic numerical algorithm based on iterating matrix multiplication to solve for eigenvalues and eigenvectors.

The significant deviation from the classical numerical setting is the preconditioning strategy. A general matrix over $\mathbb{R}$ tends to have eigenvalues of distinct absolute values, so aspects of the iteration are dominated by the unique largest eigenvalue. In the p-adic setting, it tends to be the case that all of the eigenvalues have identical absolute value, and this is true for most shifts of the form $A \mapsto A - \mu I$ as well. Our strategy is

Date: July 8, 2019.
2010 *Mathematics Subject Classification.* 15A18 (primary), 11S05 (secondary).
Key words and phrases. p-adic linear algebra, solving polynomial systems, eigenvector algorithms.

to first approximate the eigenvalues by computing the characteristic polynomial over the residue field, then refine the initial approximation via iteration. The advantage of this approach is that the computation of the characteristic polynomial of a matrix in $M_n(\mathbb{F}_q)$ is significantly faster than computing the characteristic polynomial over $\mathbb{Q}_p$ [CRV17, Sto01]. The basic Hensel-lifting based strategy applied to the system of equations $Av = \lambda v, v_i = 1$, with v_i some fixed entry of v, would require the computation of the inverse of the Jacobian matrix of the map $F\colon (v, \lambda) \mapsto (Av - \lambda v, v_i - 1)$ at each step in the iteration. Our cost per iteration step is much smaller. Unfortunately, matrices of the form $\mu I + N$, with $N \in M_n(\mathbb{Z}_p)$ topologically nilpotent, do not allow us to apply our deflation strategy as the characteristic polynomial over the residue field is unhelpful; we hope that future improvements to our strategy can be made.

The structure of the article is as follows. In Section 2 we discuss some background and terminology regarding the subject of finite precision computation over $\mathbb{Q}_p$. We review some specific results on linear algebra over the p-adic field, especially the p-adic analogues of condition-stable algorithms based on matrix factorizations. We describe our iterative strategy for the computation of the eigenvectors of a matrix over $\mathbb{Q}_p$ and compare it to the existing alternatives. In Section 3, we describe our adaptations of the algorithm of [vBMT18] to solve polynomial systems over $\mathbb{Q}_p$. Our algorithms are available as `Julia` packages, and can be obtained from:

```
https://github.com/a-kulkarn/pAdicSolver
https://github.com/a-kulkarn/Dory
```

(For technical reasons related to the `Julia` package system, we choose to divide our implementation between two packages.)

2. Linear algebra

2.1. **Precision, norms, and condition numbers.** We state some basic definitions for our discourse. We direct the reader to [CRV15, Car17, Ked10] for more details. For a matrix $A \in M_n(\mathbb{Z}_p)$, we will denote by χ_A the characteristic polynomial and denote $\chi_{A,p} := \chi_A \pmod p$. We denote the standard basis of $\mathbb{Q}_p^n$ (or $\mathbb{Z}_p^n$) by $\{e_1, \ldots, e_n\}$.

Definition 2.1. Let $v := (v_1, \ldots, v_n) \in \mathbb{Q}_p^n$ be a vector. The *norm* of v is

$$\|v\|_p := \sup_{1 \le i \le n} \{|v_i|_p\}.$$

Definition 2.2. Let $A \in \mathbb{Q}_p^{n \times m}$ be a matrix. The *operator norm* of A with respect to $\|\cdot\|_p$ is

$$\|A\|_p := \sup_{v \in \mathbb{Q}_p^m \setminus \{0\}} \frac{\|Av\|_p}{\|v\|_p}.$$

The *condition number* for an invertible square matrix is $\kappa(A) := \|A\|_p \cdot \|A^{-1}\|_p$. The condition number for a singular matrix is ∞.

We can identify a subgroup of $\mathrm{GL}_n(\mathbb{Q}_p)$ where every matrix is well-conditioned, serving the analogous role to $\mathrm{O}_n(\mathbb{R})$ in the real setting.

Lemma 2.3. *Let $A \in \mathrm{GL}_n(\mathbb{Z}_p)$. Then $\kappa(A) = 1$.*

Proof. See the comments following [Ked10, Definition 4.3.3]. □

We now come to the discussion of p-adic precision. There are many possible ways to represent a p-adic element $a \in \mathbb{Q}_p$ in a computer system [Car17]. In the `Nemo/Hecke` systems [FHHJ17], an element is represented by a series

$$a = a_{-r}p^{-r} + \ldots + a_0 + pa_1 + p^2a_2 + \ldots + a_{N-1}p^{N-1} + O(p^N)$$

where the $O(p^N)$ is the p-adic ball representing the uncertainty of the remaining digits. The *relative precision* of a is the quantity $N+r$, and the *absolute precision* is the number N. In the terminology of [Car17], we consider a system with the *zealous* (i.e, *interval*) implementation of arithmetic. The operations $-, +$ preserve the minimum of the absolute precision of the operands, and $\times, \div$ preserve the minimum relative precision of the operands. If $u \in \mathbb{Z}_p^\times$, $a \in \mathbb{Z}_p$, and $N \le N'$, then we have that $(u + O(p^{N'}))(a + O(p^N)) = ua + O(p^N)$. Multiplication by

p preserves the relative precision and increases the absolute precision by 1. The worst operation when it comes to absolute p-adic precision is dividing a small number by p. For example, the expression

$$\frac{(1+p^{99}+O(p^{100}))-(1+O(p^{100}))}{p^{100}+O(p^{200})} = p^{-1}+O(1)$$

begins with 3 numbers with an absolute and relative precision of at least 100, and ends with a result where not even the constant term is known.

In `Nemo/Hecke`, elements of a matrix store their own precision. In the terminology of [CRV17], the associated precision structure is the jagged precision lattice. However, we will not keep track of this finer precision here. Instead, in the terminology of [CRV17], we will look at the flat precision of the matrix:

Definition 2.4. Let $A, B \in M_n(\mathbb{Z}_p)$ be matrices such that $a_{i,j} = b_{i,j} + O(p^{N_{i,j}})$. Then we write $A = B + O(p^N)$, where $N := \min_{i,j} N_{i,j}$.

To refer to a matrix $A \in M_n(\mathbb{Q}_p)$ whose elements are known at an absolute precision at least N, we will simply write $A + O(p^N)$.

2.1.1. *Design choices for the implementation.* We discuss some of the design choices of the computer algebra package accompanying this article. The user we had in mind while designing our system is one who is interested in data that is inherently p-adic. More specifically, we have assumed the following hypotheses:

Assumptions.

- Input and output are inherently approximate.
- The desired output precision is roughly the input precision.
- Element-wise gains in absolute precision after $a \mapsto pa$ or $a \mapsto a^p$ are usually lost in a subsequent step.

We believe that our linear algebra package will be used in the middle of an ongoing p-adic computation, such as solving polynomial systems of equations. In such a circumstance, it does not make sense for our algorithm to "request" extra digits of precision from the input data since the input itself is approximate. We also assume that if a user inputs data at precision N, they expect output at precision roughly N, or at least N minus the condition number for the problem. Computations using relaxed p-adic numbers [Car17] allow the desired output precision to be specified. Unfortunately, relaxed arithmetic does not seem well-suited for use in an iterative scheme due to the extra memory costs (see [Car17, Section 2.4]). We also assume for a dense matrix that the entries are all given at the same flat absolute precision. We believe that this situation is fairly common in practical applications, but of course not ubiquitous. That said, the implementation of p-adic matrices in [FHHJ17] uses an element-wise precision model, so it is possible for a computation using our software to validly return a result with more accuracy than we indicate here.

In our analysis, we do not account for the extra precision on an element $a \in \mathbb{Q}_p$ gained after $a \mapsto pa$ or $a \mapsto a^p$. Experimentally, this seems to be a reasonable assumption for randomly selected dense matrices over a p-adic field, with $p \geq 40$. For a more structured scenario this assumption might not be reasonable. A nice feature of capped precision is that we do not have to account for the growth of arithmetic costs due to increased precision.

With these assumptions made, it is often useful to think of the computer as performing arithmetic in a $\mathbb{Z}/p^N\mathbb{Z}$-module for a computation with no divisions by p. It is useful to consider the image of a matrix $A \in M_n(\mathbb{Z}_p)$ in the ring $M_n(\mathbb{Z}/p^N\mathbb{Z})$. It is an essential difficulty of finite precision p-adic linear algebra that $\mathbb{Z}/p^N\mathbb{Z}$ is a local principal ideal ring, but not a domain when $N > 1$.

Remark 2.5. Our eigenvector algorithms are focused on finding only the eigenvectors defined over $\mathbb{Q}_p$. We made this choice for the following two reasons. First, in the last section of the article, we focus on finding the $\mathbb{Q}_p$-solutions of a 0-dimensional system of polynomial equations over $\mathbb{Q}_p$. Secondly, our present implementation is written using components of the OSCAR system (specifically [FHHJ17]) that are currently under development. At the time of writing the first version of our implementation, the "q-adics" interface or an interface to handle ramifed extensions of $\mathbb{Q}_p$ did not exist. Thus, our article is written with $\mathbb{Q}_p$ in mind to more closely reflect the actual software. Rapid progress is being made on the system, so we expect our implementation to evolve over time as well.

2.2. **Matrix factorizations.** In this section, we very briefly review some matrix invariants and factorizations. Matrix factorizations relating to the topological structure of $\mathbb{Q}_p$ are unsurprisingly related to the algebraic structure of the $\mathbb{Z}_p$-module spanned by the columns (or rows) of the matrix. We invite the reader to consider [Ked10, Chapter 4] for further details.

Proposition 2.6 (Iwasawa decomposition)**.** *Let k be either a finite extension of $\mathbb{Q}_p$, $\mathbb{R}$, or $\mathbb{C}$ and let $G \hookrightarrow \mathrm{GL}_n(k)$ be a linear algebraic group, let K be a maximal compact subgroup of G, and let B be a Borel subgroup. For any $A \in G$, there exists a $Q \in K$ and an $R \in B$ such that $A = QR$.*

Note that a maximal compact subgroup of $\mathrm{GL}_n(\mathbb{R})$ is the orthogonal group $\mathrm{O}_n(\mathbb{R})$, and the subgroup of invertible real upper triangular matrices is a Borel subgroup. We have as a consequence:

Corollary 2.7 (QR-factorization)**.** *Let $A \in \mathbb{R}^{n\times m}$ be a matrix. Then there exists a $Q \in \mathrm{O}_n(\mathbb{R})$ and an upper triangular matrix $R \in \mathbb{R}^{n\times m}$ such that $A = QR$.*

Of course, QR-factorization is really just a consequence of Gram-Schmidt. From the point of view of Proposition 2.6, we also have a p-adic analogue of QR-factorization. In $\mathrm{GL}_n(\mathbb{Q}_p)$, a maximal compact subgroup is $\mathrm{GL}_n(\mathbb{Z}_p)$, and the subgroup of invertible upper triangular matrices is a Borel subgroup.

Corollary 2.8 (p-adic QR-factorization)**.** *Let $A \in \mathbb{Z}_p^{n\times m}$ be a matrix. Then there exists a $Q \in \mathrm{GL}_n(\mathbb{Z}_p)$ and an upper triangular matrix $R \in \mathbb{Z}_p^{n\times m}$ such that $A = QR$.*

We decided to refer to the factorization above as a QR-factorization since many applications are similar to the real case. However, one important difference is that Q is generally not an element of $\mathrm{O}_n(\mathbb{Q}_p)$. As in the case over $\mathbb{R}$, an elementary proof of Corollary 2.8 can be given by exhibiting an algorithm to compute a QR-decomposition. The algorithm is well-known, and in fact it is just the algorithm to compute a PLU-decomposition, though with each row-pivot chosen by taking the vector with the largest p-adic norm [Ked10, Chapter 4]. We also note since $Q \in \mathrm{GL}_n(\mathbb{Z}_p)$, that R can be chosen to be the Hermite normal form of A. We summarize the discussion as:

Remark 2.9. The p-adic QR-decomposition of A is computed using a PLU-decomposition, with pivots chosen by p-adic norm.

An essential property of the QR-factorization of a real matrix A is that Q has a well-conditioned inverse. The same is true p-adically. The columns of Q in the p-adic case are also "orthogonal" in the sense of [Sch84, Section 50]. Unfortunately, this notion of orthogonality does not help to compute the inverse of Q like in the real case. As we might expect, QR-factorizations give us a way to compute the singular value decomposition of a matrix in the p-adic setting.

Proposition 2.10. *If $A \in M_n(\mathbb{Q}_p)$, then there exist $U, V \in \mathrm{GL}_n(\mathbb{Z}_p)$ and a diagonal matrix $\Sigma \in M_n(\mathbb{Q}_p)$ such that $A = U\Sigma V^T$.*

Proof. Let $A = URP'$ be a QR-decomposition with column pivoting. Now let $R^T = V\Sigma$ be a QR-decomposition. The result follows. □

The singular value decomposition is the Smith normal form of the $\mathbb{Z}_p$-module generated by columns of A [Ked10, Chapter 4]. Since the matrices U, V are well-conditioned, we can use a singular value decomposition to stabilize pnumerical computations (as in the real setting).

2.3. **The eigenvector problem: semantics.** Let $A \in M_n(\mathbb{Z}_p)$ be a matrix. In the setting of a finite p-adic precision computation, we may formulate two versions of the eigenvector problem:

Problem 1 (Eigenvector problem, Version I)**.** Let (λ, v) be an exact eigenpair for A over $\mathbb{Z}_p$. Given an approximation $\widetilde{A}$ to A such that $A = \widetilde{A} + O(p^N)$, compute a pair $(\widetilde{\lambda}, \widetilde{v})$ such that $\|\widetilde{v}\| = 1$ and $\widetilde{A}\widetilde{v} = \widetilde{\lambda}\widetilde{v} + O(p^N)$.

Problem 2 (Eigenvector problem, Version II)**.** Let (λ, v) be an exact eigenpair for A over $\mathbb{Z}_p$. Given an approximation $\widetilde{A}$ to A such that $A = \widetilde{A} + O(p^N)$, compute a triple $(\widetilde{\lambda}, \widetilde{v}, M) \in \mathbb{Z}_p \times \mathbb{Z}_p^n \times \mathbb{Z}$ such that:

(i) $\|\widetilde{v}\| = 1$, $\lambda = \widetilde{\lambda} + O(p^M)$ and $v = \widetilde{v} + O(p^M)$,

(ii) for any $B = 0 + O(p^N)$, we have that $(\tilde{A} + B)\widetilde{v} = \widetilde{\lambda}\widetilde{v} + O(p^M)$,
(iii) M is maximal among such triples.

In Version II, the second criteria ensures that the problem is well-posed.

Lemma 2.11. *Let $A \in M_n(\mathbb{Z}_p)$, let $\widetilde{A}$ be an approximation such that $A = \widetilde{A} + O(p^N)$, and let (λ, v, M) satisfy the first two criteria for a solution to Version II. Then $M \leq N$.*

Proof. If $\|(A-\widetilde{A})\widetilde{v}\|_p \geq p^{-N}$ then clearly $M \leq N$. Otherwise, choose any $B \in M_n(\mathbb{Z}_p)$ such that $\|B\widetilde{v}\|_p = 1$. Then $\|(A - \widetilde{A} + p^N B)\widetilde{v}\|_p = p^{-N}\|B\widetilde{v}\|_p = p^{-N}$. □

The following example demonstrates the difference between Version I and Version II of the problem.

Example 2.12. *Let $p = 7$, and consider the matrices*

$$A := \begin{bmatrix} p^3 & 1 \\ 0 & -p^3 \end{bmatrix}, \qquad B := \begin{bmatrix} p^3 & 1 \\ p^6 & -p^3 \end{bmatrix},$$

Both matrices are rounded to the same result with precision $N := 6$. However, the respective eigenvectors are

$$v_A := \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad v_B := \begin{bmatrix} 1 \\ p^3\sqrt{2} \end{bmatrix}.$$

Note in $\mathbb{Z}_7$ that $\sqrt{2} = \pm 4 + O(p)$. We see that the equation $Av_A + O(p^6) = p^3 v_A + O(p^6) = Bv_A + O(p^6)$. However, $v_A \neq v_B + O(p^6)$.

Version II exhibits pathological behavior whenever the characteristic polynomial of A is not square-free modulo p^M. For instance, as X ranges over all $\mathbb{Z}_p^n$ matrices, we see that the eigenvectors of $I + p^M X$ could be anything. In other words, any solution to Version II of the problem for $I + p^M X$ has $M \leq 0$. A similar difficulty arises in the eigenvector problem for real matrices. Because of this pathology, we restrict our attention to solving Version I.

Of course, we can give analogous formulations for computing a Jordan form of A or for computing a block Schur form for A. Generally, we will work with Version I of these problems as well.

2.4. **Commentary on the classical algorithm.** The naive algorithm to compute the eigenvalues of a matrix relies on the calculation and factorization of the characteristic polynomial. In the numerical setting over $\mathbb{R}$ it is desirable to avoid this step due to the instability of solving for the roots of a polynomial. Also p-adically, there is also a loss of precision in solving for the roots of a polynomial. Thus, in order to accurately compute the nullspace of $A - \lambda I$ the roots of the characteristic polynomial, and thus the characteristic polynomial itself, must be known to as much precision as possible. Presently, there are two algorithms to compute the characteristic polynomial at the maximum possible precision, both of which use more than $O(n^3)$ arithmetic operations (asymptotically):

(1) Use a division-free calculation. The asymptotic run-time is at least $O(n^4)$.
(2) Use the algorithm of [CRV17]. Compute the characteristic polynomial of a matrix with inflated precision using division, then truncate to the optimal precision computed from the precision lattice analysis. The runtime is $O(\rho n^3)$, where ρ represents the factor of performing arithmetic at the higher precision. Practically, the computation of the comatrix and the precision increase required for the precision analysis appears to impose a large cost [CRV17, Remark 5.3].

In the next subsection, we discuss how to avoid an expensive computation of the characteristic polynomial for most matrices in $M_n(\mathbb{Z}_p)$.

Remark 2.13. To clarify the term "most", if $A + O(p^N)$ is an $n \times n$-matrix whose entries are chosen with the uniform probability distribution on $[0, \ldots, p^N - 1]$, then the limit as $n \to \infty$ of the probability that χ_A is square-free is at least $\frac{1-p^{-5}}{1+p^{-3}}$ [Ful02].

2.5. **The power iteration algorithm.**

Algorithm 2.14 Eigvecs(A)
Input: $A + O(p^N)$, an $n \times n$-matrix over $\mathbb{Z}_p$.
Output: The eigenpairs (λ_i, v_i) of the matrix A.

1: **if** A is diagonal **then**
2: **return** $\{(a_{11}, e_1), \ldots, (a_{nn}, e_n)\}$.
3: **if** $A \mod p \equiv 0$ **then**
4: $\nu = \min_{i,j}\{\operatorname{ord} x_{ij}\}$
5: V = Eigvecs($p^{-v}X$)
6: Reset precision of V
7: **return** V
8: Compute $\chi_{A,p}$
9: **if** $\chi_{A,p}$ has no linear factors **then**
10: **return** Nothing.
11: **if** $\chi_{A,p} \not\equiv (x-\lambda)^n \pmod p$ **then**
12: $\{(X_1, V_1), \ldots, (X_r, V_r)\}$:= PowerIterationDecomposition(A, $\chi_{A,p}$)
13: **return** Eigvecs(X_i) for $i = 1, \ldots, r$
14: **else**
15: **return** ClassicalAlgorithm(A)

Note in step 6, we may reset the precision since the input matrix on the previous step X was a multiple of p^ν. If p is moderately large, then for most matrices the size of the Jordan blocks modulo p will be small (see Remark 2.13). The square matrix in step 13 is expected to be small, so we apply the costly method to compute the characteristic polynomial of this block.

Algorithm 2.15 PowerIterationDecomposition(A, $\chi_{A,p}$)
Input:
$A + O(p^N)$, an $n \times n$-matrix over $\mathbb{Z}_p$.
$\chi_{A,p}$, the characteristic polynomial of $A \pmod p$.

Output: A list of pairs of matrices $\{(X_i, V_i)\}_i$ such that $AV_i = V_iX_i + O(p^N)$.

1: Compute eigenpairs (λ_i, V_i) mod p
2: Set m to be the largest multiplicity of a linear factor of $\chi_{A,p}$.
3: **for** $V = V_1, \ldots, V_r$ **do**
4: Lift the eigenpair to an approximate eigenpair (λ_i, V_i) in $\mathbb{Z}_p$
5: Set $B = (A - \lambda_i I)$.
6: **Iterate** for $\log_2(mN)$ times, $B = B^2$
7: Set V := nullspace($B + O(p^N)$).
8: Solve $VX_i = AV$.
9: **return** $\{(X_1, V_1), \ldots, (X_r, V_r)\}$.

To clarify, the lifting in step 4 is just the trivial operation of interpreting the elements $\{0, \ldots, p-1\}$ as elements of $\mathbb{Z}_p$.

2.6. **Proof of correctness.** We give a quick proof of correctness of the power iteration algorithm.

Lemma 2.16. *Let λ be an approximate eigenvalue of $A \pmod p$, let m be the multiplicity of λ as a root of $\chi_{A,p}$, and let M be the nullspace of $(A - \lambda I)^{Nm}$. Then M is a free $\mathbb{Z}/p^N\mathbb{Z}$-module with trivial annihilator. Letting V be a matrix whose columns are the generators of M, there exists an $X \in M_N(\mathbb{Z}_p)$ with such that $AV = VX + O(p^N)$.*

Proof. First, note that $(A - \lambda I) \pmod p$ has 0 as an eigenvalue with multiplicity m. Ordering the singular values and eigenvalues by size, we have that $(A - \lambda I)$ satisfies

$$|\lambda_1 \dots \lambda_{n-m}| = |\sigma_1 \dots \sigma_{n-m}|_p = 1 \quad \text{and} \quad |\lambda_{n-m}|_p > |\lambda_{n-m+1}|_p .$$

Thus, by the Hodge-Newton decomposition [Ked10, Theorem 4.3.11], there is a matrix $U \in \mathrm{GL}_n(\mathbb{Z}_p)$ such that $B := U^{-1}(A - \lambda I)U$ is block upper triangular, with the lower right block B_{22} accounting for the m small eigenvalues and the upper left block B_{11} accounting for the unit eigenvalues. Now, we see that $B_{22}^m = 0 \pmod p$, so

$$(B^m)^N = \begin{bmatrix} B_{11}^{Nm} & * \\ 0 & 0 \end{bmatrix} + O(p^N).$$

But as $U \in \mathrm{GL}_n(\mathbb{Z}_p)$, we see that the Smith normal form of $(A - \lambda I)^{Nm}$ is $\begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix}$. Thus, the kernel of $(A - \lambda I)^{Nm}$ is free as a $\mathbb{Z}/p^N\mathbb{Z}$-module with trivial annihilator in $\mathbb{Z}/p^N\mathbb{Z}$. Next, note that we have

$$(A - \lambda I)^{Nm}(AM) = A(A - \lambda I)^{Nm} V = 0$$

so $AM \subseteq M$. Thus A restricted to M defines an endomorphism of M, represented by a square matrix. □

As an immediate corollary, we have:

Corollary 2.17. *Let $(V_1, X_1), \dots, (V_r, X_r)$ be the blocks returned from power iteration. With $\mathbf{V} := [V_1, \dots, V_r]$ the horizontal join, we have*

$$A\mathbf{V} = \mathbf{V} \begin{bmatrix} X_1 & & & \\ & X_2 & & \\ & & \ddots & \\ & & & X_r \end{bmatrix} + O(p^N).$$

We hope that step 15 can be improved, as we believe it is helpful in solving polynomial systems whose solutions have large valuations. Thus, we raise the question:

Question 2.18. *Given a topologically nilpotent matrix $B \in M_n(\mathbb{Z}_p)$, is there an efficient iterative strategy to determine the eigenvectors of $B + O(p^N)$?*

2.7. **The Schur form algorithm.** Solving for the nullspace at the end of the iteration is the dominant cost in the power iteration algorithm. In the classical numerical setting, we can avoid this drawback by using the QR-iteration algorithm. We also have a p-adic version of the QR-algorithm. With the p-adic QR step replacing the usual QR factorization, our algorithm is simply the original LR-algorithm [Rut58] that inspired the QR-algorithm for real matrices. The LR-algorithm fell out of favor some time in the 1960's since it is numerically less stable than the QR-algorithm, but p-adically this situation is reversed!

We now give the algorithm.

Algorithm 2.19 QR-Iteration(A, $\chi_{A,p}$)

Input:

$A + O(p^N)$, an $n \times n$-matrix over $\mathbb{Z}_p$.

$\chi_{A,p}$, the characteristic polynomial of $A \pmod p$.

Output: A (block) triangular form T for A, and matrix V such that $AV = VT + O(p^N)$.

1: Set $\lambda_1, \dots, \lambda_\ell$ to be the roots of $\chi_{A,p}$ in $\mathbb{F}_p$, lifted to $\mathbb{Z}_p$.
2: Set $m_1, \dots, m_\ell$ to be the multiplicities of the roots of $\chi_{A,p}$.
3: Compute B, V such that $AV = VB$ and B is in Hessenberg form.
4: **for** $i = 1, \dots, \ell$ **do**
5: **for** $j = 1, \dots, m_i N$ **do**
6: Factor $(B - \lambda_i I) = QR$
7: Set $B := RQ + \lambda_i I$
8: Set $V := Q^{-1}V$
9: **return** B, V.

Algorithm 2.20 BlockSchurForm(A)

Input: $A + O(p^N)$, an $n \times n$-matrix over $\mathbb{Z}_p$.

Output: A block upper triangular matrix T with $\ell + 1$ distinct blocks and matrix V such that $AV = VT + O(p^N)$. Here, ℓ is the number of linear factors of $\chi_{A,p} = \chi_A \pmod p$.

if A is diagonal **then**
  return $A, \{e_1, \ldots, e_n\}$.
if $A \mod p \equiv 0$ **then**
  $\nu = \min_{i,j}\{\mathrm{ord}_p\, x_{ij}\}$
  V = BlockSchurForm($p^{-v}X$)
  Reset precision of V
  return V

Compute $\chi_{A,p} := \chi_A \pmod p$
if $\chi_{A,p}$ has no linear factors **then**
  return $A, \{e_1, \ldots, e_n\}$

if $\chi_{A,p} \neq (x - \lambda)^n$ **then**
  return B, V := QR-Iteration(A, $\chi_{A,p}$)
else
  B, V := $A, \{e_1, \ldots, e_n\}$

for $j := 1 \ldots \ell$ **do**
  if $\chi_{B_{ii},p} \equiv (x - \lambda)^m$ for $m > 1$ **then**
    C_j, X_j := ClassicalAlgorithm(B_j)
    Update B and V
return A, V

We briefly comment on some aspects of this algorithm analogous to the archimedean setting (the proofs for the various statements are analogous as well). Note that each p-adic QR-iteration preserves the Hessenberg form, so the cost of applying the QR-step is $O(n^2)$. From the discussion of the LR-algorithm in [Wil65, Sections 5-9], we see that the subdiagonal entry $(i, i+1)$ converges to 0 at a rate of $O\left(\frac{|\lambda_{i+1}|_p^s}{|\lambda_i|_p^s}\right)$, provided that $|\lambda_{i+1}|_p < |\lambda_i|_p$. For $B \in M_n(\mathbb{Z}_p)$, if $|\lambda_{i+1}|_p < |\lambda_i|_p = 1$, then $\frac{|\lambda_{i+1}|_p}{|\lambda_i|_p} < p^{-\frac{1}{m}}$, where m is the largest multiplicity of an eigenvalue of $B \pmod p$. Additionally, if some subdiagonal entries are identified to be zero during the iteration, the standard deflation strategies can be used to accelerate the algorithm.

Remark 2.21. In our version of the iteration we use a constant shift factor, and so the iteration converges in at most mN steps [Wil65]. We believe that a p-adic analogue of the Rayleigh quotient strategy to refine the eigenvalue approximation will lead to a drastic speed-up in convergence.

Remark 2.22. It is possible to use the QR-iteration to compute the valuations of the eigenvalues iteratively. After applying a single round of the QR-iteration to a Hessenberg matrix A with no shift (i.e, $\lambda_i = 0$), the valuations of the eigenvalues can be read from the diagonal blocks in the resulting matrix. If B_{ii} is one of the diagonal blocks with no subdiagonal entry equal to 0, then all of the eigenvalues of B_{ii} must have the same valuation, as otherwise some of the subdiagonal entries of B_{ii} would have converged to 0. In fact, the valuations of the eigenvalues of A can be computed this way even if some of the eigenvalues are not defined over $\mathbb{Q}_p$.

2.8. **Complexity analysis.** We denote by $M(n)$ the cost of matrix multiplication.

Proposition 2.23. *Let N be the precision and let m be the largest size of a Jordan block of $A \pmod p$. Then the number of $\mathbb{Q}_p$-arithmetic operations performed by Algorithm 2.14 is as most*

$$O\left(\rho m^3 + \ell(n^3 + M(n)\log_2(mN))\right).$$

Proof. We tabulate the costs in a table.

Line(s)	Cost per line	
Eigvecs		
3 to 7	Negligible	
8	$O(M(n)(\log n)(\log\log n))$	Finite field operations.
12	$O(\ell n^3 + M(n)\log_2 mN)$	Power iteration
15	$O(\rho m^3)$	Classical algorithm
Total:	$O(\ell n^3 + \rho m^3 + M(n)\log_2 mN)$	

Power iteration decomposition		
1	$O(n^3)$	Finite field operations
3	ℓ iterations	
– 5	$O(M(n)\log_2 mN)$	
– 6	$O(n^3)$	
7	$O(m^3)$	
Total:	$O(\ell n^3 + M(n)\log_2 mN)$.	

□

The ρm^3 term corresponds to blocks which cannot be decomposed using power iteration, i.e, where the classical algorithm is invoked. Note that the cost of computing $\chi_A \pmod p$ is negligible, since over a finite field the characteristic polynomial can be computed in $O(M(n)(\log n)(\log\log n))$ finite field operations [Sto01].

If the linear factors occur with small multiplicities mod p, our algorithm significantly outperforms the naive strategy. If all linear factors mod p occur with multiplicity 1 the classical algorithm is never called, so the asymptotic run-time simplifies to $O(n^3 + \ell M(n)\log_2(N))$. The worst-case scenario of the algorithm is if $\chi_A \pmod p \equiv (x-\alpha)^n$, in which case the default algorithm is called immediately; the only wasted time in this case is the inexpensive computation of $\chi_A \pmod p$.

We now study the complexity of the block Schur form algorithm.

Proposition 2.24. *Let N be the precision, let m be the largest size of a Jordan block of $A \pmod p$, and let ℓ be the number of linear factors of $\chi_A \pmod p$. Then the block Schur form can be computed in $O(n^3 + \ell mNn^2 + \rho m^3)$ $\mathbb{Q}_p$-arithmetic operations.*

Proof. We tabulate the costs in a table.

Line(s)	Cost per line	
BlockSchurForm		
8 to 11	Negligible	
12	$O(n^3 + \ell mNn^2)$	QR-iteration
15	ℓ-iterations	
– 17	$O(\rho m^3)$	Classical algorithm
– 18	$O(nm^2)$	
Total:	$O(\rho m^3 + n^3 + \ell mNn^2)$	

QR-Iteration.		
1	$O(M(n)(\log n)(\log\log n))$	Finite field operations
2	$O(1)$	
3	$O(n^3)$	Hessenberg form
4	ℓ-iterations	
– 5	N-iterations	
– – 6	$O(n^2)$	
– – 7	$O(n^2)$	(parallel with step 6 row operations)
– – 8	$O(n^2)$	(parallel with step 6 row operations)
Total:	$O(n^3 + \ell mNn^2)$	

□

2.9. **p-adic Householder reflections.** We close the section with a quick observation about Householder reflections.

Lemma 2.25. *Let p be an odd prime. Let $x \in \mathbb{Z}_p^n$ be a vector with exactly one coordinate with minimal valuation r. We further assume this coordinate is not x_1. Let e_1 be the first standard basis vector. Then:*

(a) $x^T x$ is a non-zero square in $\mathbb{Z}_p$.
(b) Let $\alpha := \sqrt{x^T x} \in \mathbb{Z}_p$ be one of the square roots and let $v' := x - \alpha e_1$. Choose (the unique) $v \in \mathbb{Z}_p^n$ such that $v' = p^r v$. Then $\left|v^T v\right|_p = 1$, and the Householder transformation

$$H := I - \frac{2}{v^T v} v v^T$$

is a well-defined reflection. In particular $H \in \mathrm{O}_n(\mathbb{Z}_p)$.
(c) $Hx = \alpha e_1$.

Proof. (a) Assume x_i is the coordinate with minimal valuation. Write $x = p^r y$. We have that $x^T x = p^{2r}(y^T y) = p^{2r}(y_i^2 + O(p))$. As y_i is a unit, we see by Hensel's lemma that this is a square.

(b) Assume again x_i is the coordinate with minimal valuation. Note that $\alpha^2 = \sum_{j=1}^n x_j^2$, so as x_i is the unique coordinate with minimal valuation, we have $\alpha = x_i + O(p^{r+1})$. Now

$$\begin{aligned} p^{2r} v^T v &= (x^T x - 2\alpha x^T e_1 + \alpha^2) \\ &= 2(x^T x - (x_i + O(p^{r+1}))x_1) \\ &= 2(x_i^2 - x_i x_1) + O(p^{2r+1}) \\ &= 2x_i(x_i - x_1) + O(p^{2r+1}) \end{aligned}$$

By our assumption on x, we have $x_i - x_1 = x_i + O(p^{r+1})$. Thus, $v^T v$ is a unit and H is well-defined. That it is a reflection follows from the usual calculation, which is

$$\begin{aligned} HH &= \left(I - \frac{2}{v^T v} v v^T\right)\left(I - \frac{2}{v^T v} v v^T\right) \\ &= I - \frac{4}{v^T v} v v^T + \frac{4}{(v^T v)^2}(v v^T)(v v^T) \\ &= I - \frac{4}{v^T v} v v^T + \frac{4}{(v^T v)}(v v^T) \\ &= I. \end{aligned}$$

Finally, it is clear that all of the entries of H are integral.

(c) The result is clear from direct calculation.

□

We now offer a Householder version of the Hessenberg algorithm. Our version of the Hessenberg algorithm is more a theoretical curiosity; for our practical implementations, we use [CRV17, Algorithm 1] as it appears to be more efficient.

Algorithm 2.26 HessenbergForm(A)

1: **for** $j = 1, \ldots, (n-1)$ **do**
2: **if** The j-th subdiagonal is 0 **then**
3: **continue**
4: Swap the last (i.e, n-th) row with row i to ensure a_{nj} has the minimal valuation of the sub-diagonal elements
5: Add multiples of the last (n-th row) to the i-th row, for $j < i < n$, to ensure a_{nj} is the unique sub-diagonal element with minimal valuation
6: Apply the corresponding column operations to preserve similarity. None of these operations affect columns $1, \ldots, j$
7: Apply the Householder reflection to set the j-th subcolumn to a multiple of e_1.

3. Polynomial system solving

In this section, we solve 0-dimensional systems of polynomial equations over $\mathbb{Q}_p$. The general method is based on the truncated normal form solver of [vBMT18]. We begin by reviewing some of their terminology. Throughout, we let $R := \mathbb{Q}_p[x_1, \ldots, x_n]$ and let I be an ideal such that $d = \dim_{\mathbb{Q}_p} R/I < \infty$.

Definition 3.1. A *normal form* on R with respect to I is a linear map $\mathcal{N} : R \to B$ where $B \subseteq R$ is a $\mathbb{Q}_p$-vector subspace of dimension d such that the sequence

$$0 \longrightarrow I \longrightarrow R \xrightarrow{\mathcal{N}} B \longrightarrow 0$$

is exact, and $\mathcal{N}|_B = \mathrm{id}_B$.

Consider the canonical exact sequence

$$0 \longrightarrow I \longrightarrow R \xrightarrow{\pi} R/I \longrightarrow 0\ .$$

Let $s\colon R/I \to R$ be a section of $\mathbb{Q}_p$-vector spaces and let B be the image of R/I. We can construct a normal form by setting $\mathcal{N} := \pi \circ s$, giving the commutative diagram

$$\begin{array}{ccccccccc} & & & & B & & & & \\ & & & & \downarrow & \nwarrow^{s} & & & \\ 0 & \longrightarrow & I & \xrightarrow{\iota} & R & \xrightarrow{\pi} & R/I & \longrightarrow & 0. \end{array}$$

Vice-versa, any normal form $\mathcal{N}$ defines a section of π.

Following [vBMT18], there is no need to compute an entire normal form for our purposes, and we can often restrict to a finite dimensional subspace.

Definition 3.2. Let $B \subseteq V \subseteq R$ with B, V finite dimensional $\mathbb{Q}_p$-vector subspaces such that $x_i B \subseteq V$ for all $i = 1, \ldots, n$, and $\dim_{\mathbb{Q}_p} B = \dim_{\mathbb{Q}_p}(R/I)$. A *Truncated Normal Form (TNF)* on V with respect to I is a linear map $\mathcal{N} : V \to B$ such that N is the restriction to V of a normal form with respect to I. That is, the sequence

$$0 \longrightarrow I \cap V \xrightarrow{\iota} V \xrightarrow{\mathcal{N}} B \longrightarrow 0$$

is exact, and $\mathcal{N}|_B = \mathrm{id}_B$.

Similar to normal forms, from a commutative diagram of $\mathbb{Q}_p$-vector spaces

$$\begin{array}{ccccccccc} & & & & B & & & & \\ & & & & \downarrow & \nwarrow^{s} & & & \\ 0 & \longrightarrow & I \cap V & \xrightarrow{\iota} & V & \xrightarrow{\pi} & V/I \cap V & \longrightarrow & 0 \end{array}$$

we obtain a truncated normal form by $\mathcal{N} = s \circ \pi$, where s is a section of π and $x_i B \subseteq V$ for all i.

One way to construct a truncated normal form is via resultant matrices.

Definition 3.3. Let k be a field and let $f_1, \ldots, f_r \in k[x_1, \ldots, x_n]$ be polynomials. Let D be a positive integer and let $d_i := \deg f_i$. A *resultant matrix* is a k-linear map of the form

$$\begin{array}{rccc} \mathrm{Res}\colon & V_{D-d_1} \times \ldots \times V_{D-d_r} & \to & V_D \\ & (q_1, \ldots, q_r) & \mapsto & f_1 q_1 + \ldots + f_r q_r \end{array}$$

where V_d is the k-vector space of polynomials of degree at most d. (In particular, $V_{-n} = \{0\}$ for n a positive integer.) We denote the corresponding matrix as $[\mathrm{Res}_{ij}]$.

Note that the image of the resultant map is contained in $V_D \cap I$. If D is sufficiently large we obtain a truncated normal form. (The result below is quite old, and originally due to Macaulay.)

Proposition 3.4. *If $D = \sum_{i=1}^{r}(d_i - 1) + 1$, then $\operatorname{Im}\mathrm{Res} = I \cap V_D$. In particular, the canonical quotient $\pi\colon V_D \to V_D/(V_D \cap I)$ is represented by the matrix $(\ker[\mathrm{Res}_{ij}]^T)^T$.*

Proof. See [vBMT18, Section 4]. □

3.1. **The algorithm.** We now summarize our modifications of [vBMT18, Algorithm 1]. Our implementation is available at:

`https://github.com/a-kulkarn/pAdicSolver`

The critical theoretical underpinning of [vBMT18, Algorithm 1] is Stickelberger's Theorem [CLO15, page 621].

Theorem 3.5. *Let k be a field, let $R := k[x_1, \ldots, x_n]$, and let I be an ideal such that $\dim_k R/I$ is finite. Let $p_1, \ldots, p_\ell$ be the geometric points of the affine scheme $\mathrm{Spec}(R/I)$ with multiplicities $m_1, \ldots, m_\ell$ respectively. Let $[f]\colon R/I \to R/I$ be the endomorphism of R/I induced by multiplication by $f \in R$. Then the eigenvalues of $[f]$ and their multiplicities are $(f(p_1), m_1), \ldots, (f(p_\ell), m_\ell)$. The i-th invariant subspace of $[f]$ is $\{g \in R/I : g(p_j) = 0 \text{ for } j \neq i\}$.*

Proof. Let k^{al} be an algebraic closure for k. We may assume that $R/I \otimes_k k^{\mathrm{al}}$ is local by the Chinese Remainder Theorem. Note $R/I \otimes_k k^{\mathrm{al}}$ is Artinian, so for $\mathfrak{m}$ the maximal ideal we have $\mathfrak{m}(\mathfrak{m}^r) = \mathfrak{m}^r$ for some $r > 0$. By Nakayama's lemma we have $\mathfrak{m}^r = 0$. Thus, for any $f \in \mathfrak{m}$, $[f]$ is nilpotent and so has 0 as an eigenvalue with multiplicity $\dim_{k^{\mathrm{al}}}(R/I \otimes_k k^{\mathrm{al}})$. Since $R/I \otimes_k k^{\mathrm{al}} = k^{\mathrm{al}} \oplus \mathfrak{m}$ as a k^{al}-vector space, we now need only check the result for constant functions. For $f = 1$, the result is clear. □

Note that if k is not algebraically closed, the eigenvalues and invariant subspaces may only be defined over a finite extension of k; these correspond to points whose coordinates lie in a finite extension of k. Critically, the eigenvalues of $[x_i]$ are the i-th coordinates of the geometric points. We now give the p-adic solver algorithm:

Algorithm 3.6 SolveQpSystem($f_1, \ldots, f_m$)
Input: Polynomials $f_1, \ldots, f_m \in \mathbb{Q}_p[x_1, \ldots, x_n]$ defining a 0-dimensional polynomial system.
Output: A set of approximate solutions to the distinct $\mathbb{Q}_p$-solutions.

1: Construct the resultant map $[\mathrm{Res}_{ij}]$ with D sufficiently large
2: Compute $[\pi_{ij}] = (\ker[\mathrm{Res}_{ij}]^T)^T$
3: Extract the submatrix M of $[\pi_{ij}]$ whose columns correspond to monomials m such that $\deg m < D$.
4: Compute a square subblock of M that is pnumerically well conditioned, using the p-adic QR-factorization. Let $\mathbf{b}$ be the basis corresponding to this sub-block.
5: Construct the multiplication by x_i matrices $[x_1]_{\mathbf{b}}, \ldots, [x_n]_{\mathbf{b}}$ from the columns of $[\pi_{ij}]_{\mathbf{b}}$.
6: Compute the eigenvectors of a random linear combination via iteration.
7: **return** The eigenvalues for $[x_1]_{\mathbf{b}}, \ldots, [x_n]_{\mathbf{b}}$ indexed by the invariant subspaces.

By $[x_i]_{\mathbf{b}}$, we mean the matrix corresponding to the multiplication-by-x_i operator in $\mathbf{b}$-coordinates. The choice of $B := \mathrm{Span}(\mathbf{b})$ in step 4 defines a subspace of V_D of the same dimension as R/I, hence a section $s\colon R/I \to V_D$ and thus a truncated normal form. Since $[x_i]B \subseteq V_D$, we can easily read off the action of the operator $[x_i]\colon B \to B$ from the columns of $[\pi_{ij}]$. For explicit details see [vBMT18, Theorem 4 and following text].

The differences between the $\mathbb{Q}_p$ and $\mathbb{R}$ versions of the algorithm are precisely the adaptations in the linear algebra, i.e, steps 2, 4, and 6. Note that if k is any finite precision field for which these steps can be implemented, the general recipe of [vBMT18, Algorithm 1] allows us to compute the approximate solutions. Examples include $k = \mathbb{R}, \mathbb{C}, \mathbb{Q}_p, \mathbb{F}_q((t))$.

Finally, instead of computing a truncated normal form in steps 1 and 2, we may simply use a Gröbner basis instead. For systems where a Gröbner basis is already known or where the polynomials in the basis have small coefficients, the numerous optimizations of Gröbner basis algorithms are likely preferable from a practical standpoint. The major advantage of our implementation is the fact that the coefficient size is capped at p^N.

4. Acknowledgements

I would like to thank Yue Ren, David Roe, and Simon Telen for helpful discussions about the article. I would also like to thank Marta Panizzut, Emre Sertöz, and Bernd Sturmfels for their helpful comments.

REFERENCES

[vBMT18] Marc van Barel, Bernard Mourrain, and Simon Telen, *Solving polynomial systems via truncated normal forms*, SIAM J. Matrix Anal. Appl. **39** (2018), no. 3, 1421–1447, DOI 10.1137/17M1162433.

[Car17] Xavier Caruso, *Computations with p-adic numbers*, 2017. arXiv:1701.06794.

[CRV15] Xavier Caruso, David Roe, and Tristan Vaccon, *p-adic stability in linear algebra*, ISSAC'15—Proceedings of the 2015 ACM International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2015, pp. 101–108.

[CRV17] ———, *Characteristic polynomials of p-adic matrices*, ISSAC'17—Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2017, pp. 389–396.

[CLO15] David A. Cox, John Little, and Donal O'Shea, *Ideals, varieties, and algorithms*, 4th ed., Undergraduate Texts in Mathematics, Springer, Cham, 2015. An introduction to computational algebraic geometry and commutative algebra.

[Dix82] John D. Dixon, *Exact solution of linear equations using p-adic expansions*, Numer. Math. **40** (1982), no. 1, 137–141, DOI 10.1007/BF01459082.

[Ful02] Jason Fulman, *Random matrix theory over finite fields*, Bull. Amer. Math. Soc. (N.S.) **39** (2002), no. 1, 51–85, DOI 10.1090/S0273-0979-01-00920-X.

[FHHJ17] Claus Fieker, William Hart, Tommy Hofmann, and Fredrik Johansson, *Nemo/Hecke: computer algebra and number theory packages for the Julia programming language*, ISSAC'17—Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation, ACM, New York, 2017, pp. 157–164.

[Ked10] Kiran S. Kedlaya, *p-adic differential equations*, Cambridge Studies in Advanced Mathematics, vol. 125, Cambridge University Press, Cambridge, 2010.

[Rut58] Heinz Rutishauser, *Solution of eigenvalue problems with the LR-transformation*, Nat. Bur. Standards Appl. Math. Ser. **1958** (1958), no. 49, 47–81.

[Sch84] W. H. Schikhof, *Ultrametric calculus*, Cambridge Studies in Advanced Mathematics, vol. 4, Cambridge University Press, Cambridge, 1984. An introduction to p-adic analysis.

[Sto01] Arne Storjohann, *Deterministic computation of the Frobenius form (extended abstract)*, 42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001), IEEE Computer Soc., Los Alamitos, CA, 2001, pp. 368–377.

[Wil65] J. H. Wilkinson, *Convergence of the* LR, QR, *and related algorithms*, Comput. J. **8** (1965), 77–84, DOI 10.1093/comjnl/8.3.273.

MAX PLANCK INSTITUTE MIS LEIPZIG

E-mail address: avinash@mis.mpg.de