ALGORITHMS TO COMPUTE DISCRETE RESIDUES OF A RATIONAL FUNCTION

by

Hari Prasad Sitaula

APPROVED BY SUPERVISORY COMMITTEE:

Carlos Arreche, Chair

Maxim Arnold

Mieczyslaw K. Dabkowski

Nathan Williams

This dissertation is dedicated to my family.

ALGORITHMS TO COMPUTE DISCRETE RESIDUES OF A RATIONAL FUNCTION

by

HARI PRASAD SITAULA, BA, MA

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

MATHEMATICS

THE UNIVERSITY OF TEXAS AT DALLAS

August 2023

ACKNOWLEDGMENTS

First of all, I would like to take a moment to express my deepest appreciation to my PhD supervisor, Dr. Carlos Arreche. Throughout my PhD journey, Dr. Arreche has been an invaluable mentor and guide, providing me with crucial insights, expertise, and feedback that have helped me navigate the complex world of research. His dedication, passion, and unwavering support have inspired me to strive for excellence in everything I do. He has inspired me to think critically and creatively and has provided me with the tools and resources I need to succeed. His mentorship has been a cornerstone of my success, and I could not have achieved this significant milestone without him. Therefore, I would like to extend my sincerest gratitude to Dr. Arreche for his constant support, guidance, and mentorship. I feel incredibly fortunate to have had the opportunity to work with him. Thank you for everything, Dr. Arreche.

Secondly, my wife has been my biggest supporter and my most trusted advisor. Her dedicated love, encouragement, and support have sustained me through the long hours of research, writing, and analysis. Her constant belief in me and her tireless efforts to support me in every possible way have made this day possible. Without her, I would not have been able to achieve this success in my academic career. Therefore, I would like to express my heartfelt gratitude to my wife for her selflessness, her love, and her support throughout my PhD journey. I am incredibly fortunate to have her in my life, and I could not have done this without her. Thank you, my love, for always being there for me.

Finally, I would like to acknowledge the people who have been my constant source of support and inspiration throughout my academic journey - my parents and family members. Their unconditional love, encouragement, and support have been the foundation of my success, and I could not have achieved this landmark without them. My parents have always been my biggest supporters, believing in me even when I doubted myself. They have been my

role models, teaching me the value of hard work, perseverance, and dedication. My family members have also been a constant source of encouragement and inspiration, cheering me on every step of the way and providing me with support in every endeavor. Therefore, I would like to express my deepest gratitude to my parents and family members for their love, support, and encouragement throughout my academic journey. Their determined faith in me has helped me overcome every obstacle and has enabled me to reach this goal. Thank you for everything, Mom, Dad, Son, Daughter, and all my family members.

June 2023

ALGORITHMS TO COMPUTE DISCRETE RESIDUES OF A RATIONAL FUNCTION

Hari Prasad Sitaula, PhD
The University of Texas at Dallas, 2023

Supervising Professor: Carlos Arreche, Chair

The classical notion of residue, for a rational function with complex coefficients, is a powerful and ubiquitous tool, having applications in many different areas. For example: Complex Analysis, Physics, Number Theory, Differential Equations, and Combinatorics, to name a few. In the last decade several new notions of discrete residues have been developed by different researchers, all of which have in common the following obstruction-theoretic feature: a given rational function $f(x)$ is "special" (e.g., rationally integrable, or rationally summable, or rationally q-summable) if and only if all of its corresponding residues are zero. All of these notions of residue (both the classical one and also its discrete variants) are originally defined in terms of a complete partial fraction decomposition of the given rational function $f(x)$, which is too expensive to carry out in practice due to the high computational cost of finding the complete factorization of the denominator. The main contribution of this dissertation is the development of an efficient factorization-free algorithm to compute the discrete residues of a rational function.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this thesis, we explain how to use a computer algebra system to compute the discrete residues of a given univariate rational function. Before going into detail, let us first briefly introduce the notion *computer algebra*. In recent decades, the ubiquity of computers to perform mathematical computation symbolically has been enormously increased. This is because of the development of different computer systems and algorithms that can transform, combine, and compute symbolic algebraic expressions. The advancement in computer algebra systems has made it possible to use abstract symbols in computers for representing complicated mathematical expressions, rather than dealing with just numbers.

Computer algebra is the area of computer science where mathematical tools and computer software are developed for the exact solution of equations, and more generally algebraic problems [1]. But if we can find exact solutions to an equation, then why should we spend time approximating solutions of equations? The answer is that in many cases an exact solution is not possible, and in many other cases, efficiency is a real issue. For example, it is pointless to wait for a computer for ten whole days, to just get a weather forecast. However, if the problem is within the range of exact solvability, the exact solutions provided by a computer algebra system would be much better in the accuracy than any numerical approximation.

There are many attempts to define the terminology *symbolic algebraic computation* or *computer algebra*. Let us introduce a definition made by R. Loos in his Introduction to Buchberger et al. (1983):
"Computer algebra is the part of computer science which designs, analyzes, implements and applies algebraic algorithms." To learn more about computer algebra system we refer the readers to [2].

In the realm of computer algebra, our focus expands beyond just integers and real numbers found in numerical computation. We delve into the realm of symbolic computation, where we manipulate expressions representing solutions to abstract mathematical equations. This does not imply that computer algebra excludes numerical computations entirely; rather, it encompasses a broader range of objects for study. Alongside integers and real numbers, computer algebra deals with rational functions, polynomials, algebraic numbers, trigonometric expressions, and more.

This doesn't diminish the importance of numerical algorithms. Both forms of scientific computation possess their own unique strengths and advantages. Therefore, combining them within a computational environment allows for significant advancements in mathematical science. For instance, when confronted with the task of computing an approximate solution to a differential equation, one approach could involve using computer algebra to determine the first n terms of a power series solution through exact methods. Subsequently, these terms can be passed on to a numerical package for power series evaluation, yielding an approximate solution.

By merging the capabilities of computer algebra and numerical computation, we leverage the strengths of each approach, paving the way for enhanced progress in the field of mathematical science.

In this thesis, we are going to use symbolic computation, mainly using the *Maple* computer algebra software, to compute the discrete residues of a rational function. The main goal of this thesis is to develop an efficient and practical algorithm to compute the discrete residues of a rational function in the shift case. To read more about symbolic computation please see [3].

Summability of a Rational Function

The classical notion of *residues* comprises a powerful and ubiquitous tool in real and complex analysis, with applications to a variety of fields including in particular Physics, Combinatorics, Number Theory, and many more [4]. Chen and Singer introduced a theory of discrete and q-discrete residues in recent years [5]. This theory primarily focuses on investigating telescoping problems, but it has also proven to be valuable in various related problem domains [6, 7, 8, 5]. These different notions of residues are closely interconnected with problems that share a similar nature. Specifically, given a rational function $f(x) \in C(x)$, where C represents an algebraically closed field with characteristic 0, the objective is to determine whether $f(x)$ can be characterized as:

1. Rationally integrable, i.e., whether there exists a rational function $g(x) \in C(x)$ such that $f(x) = g'(x)$; or

2. Rationally summable, i.e., whether there exists a rational function $g(x) \in C(x)$ such that $f(x) = g(x+1) - g(x)$; or

3. Rationally $q-$summable, i.e., whether there exists a rational function $g(x) \in C(x)$ such that $f(x) = g(qx) - g(x)$.

As discussed by Arreche and Zhang in [9], one of the computational advantages of approaching the above problems via residues is that these questions can be decided without computing the *certificate* $g(x) \in C(x)$, whose computation in practice is often expensive and not strictly necessary. More precisely, in each of the above cases such a g exists if and only if all of the corresponding residues of f are zero. As part of this thesis, I develop practical and efficient algorithms to compute the discrete residues of any rational function, resulting in a practical approach to item 2 above.

Motivation: Classical Residues and Rational Integrability

To motivate the main ideas and strategy underpinning my doctoral work, let me first briefly recall its well-known classical analogue: classical/continuous residues and their role in deciding rational integrability. Let $f(x) = \frac{a(x)}{b(x)} \in C(x)$ be a rational function where $a, b \in C[x]$ are relatively prime polynomials with $\deg(a) < \deg(b)$. Then there exists a *complete partial fraction decomposition*

$$f(x) = \frac{a(x)}{b(x)} = \sum_{k \geq 1} \sum_{\alpha \in C} \frac{c_k(\alpha)}{(x-\alpha)^k} \tag{1.1}$$

where $c_k(\alpha) \in C$ are zero for all but finitely many α and k. We know from Calculus that there exists a rational certificate $g(x) \in C(x)$ such that $f(x) = g'(x)$ if and only if all the first-order residues of $f(x)$ vanish, i.e.,

$$\mathrm{res}(f, \alpha, 1) := c_1(\alpha) = 0 \; \forall \; \alpha \in C.$$

It would seem *a priori* that in order to decide whether $f(x)$ is rationally integrable, we should first compute its complete partial fraction decomposition (1.1) and then check whether every single $c_1(\alpha) = 0$ — in fact, this is how we teach Calculus students to integrate rational functions! However, even this classical question (of whether $f(x)$ admits a rational antiderivative) can be decided much more efficiently, and in particular without the expensive computation of its complete partial fraction decomposition (in which the bottleneck is the complete factorization of the denominator $b(x)$ into linear factors). Indeed, one can very inexpensively find a "reduced form" $\tilde{f} = \frac{\tilde{a}}{\tilde{b}}$, where $\tilde{a}$ and $\tilde{b}$ are still relatively prime with $\deg(\tilde{a}) < \deg(\tilde{b})$ such that: (1) $\tilde{f}$ has the same first-order residues as f; and (2) $\tilde{b}$ is *squarefree*, meaning that it has no repeated linear factors. Now the residues of $\tilde{f}$ are precisely the roots of the *Rothstein-Trager resultant*

$$P(z) := \mathrm{Res}_x(\tilde{a} - z\tilde{b}', \tilde{b}).$$

I refer to [10] for more details regarding this approach for computing classical residues of rational functions. This was one of the main motivations for me to embark on my doctoral

research project: is there an analogous way to compute the *discrete residues* of a rational function *without* having to first compute its complete partial fraction decomposition? I will next explain why the answer is "yes".

Discrete Residues and Summability

Let once again $f(x) = \frac{a(x)}{b(x)} \in C(x)$ be a rational function, with $a(x), b(x) \in C[x]$ relatively prime polynomials such that $\deg(a) < \deg(b)$. In order to define the *discrete residues* of f following [9], we first reorganize the complete partial fraction decomposition (1.1):

$$f(x) = \frac{a}{b} = \sum_{k \geq 1} \sum_{[\alpha] \in C/\mathbb{Z}} \sum_{n \in \mathbb{Z}} \frac{c_k(\alpha + n)}{(x - \alpha - n)^k} \tag{1.2}$$

where $\alpha \in C$ is some choice of coset representative for each $[\alpha] = \alpha + \mathbb{Z} \in C/\mathbb{Z}$. The k^{th} order discrete residue of $f(x)$ at the $\mathbb{Z}-$orbit $[\alpha] \in C/\mathbb{Z}$ is defined to be

$$\operatorname{dres}(f, [\alpha], k) := \sum_{n \in \mathbb{Z}} c_k(\alpha + n).$$

By [[5], Prop.(2.5)], f is rationally summable, i.e., there exists a rational function $g(x) \in C(x)$ such that $f(x) = g(x+1) - g(x)$, if and only if

$$\operatorname{dres}(f, [\alpha], k) = 0.$$

for every $[\alpha] \in C/\mathbb{Z}$ and every $k \in \mathbb{N}$. But here once again it would seem as though, in order to decide whether f is rationally summable (and more generally to compute the discrete residues of f, whatever they may be), it is necessary to first compute the complete partial fraction decomposition (1.1) of f, then reorganize it by $\mathbb{Z}$-orbits according to (1.2), and finally sum the classical k-order residues of f for each k within each $\mathbb{Z}$-orbit $[\alpha]$. But this approach is too (prohibitively!) expensive in practice, which is why there is a need for a practical algorithm for computing discrete residues. Please see [9] for more details about residues and summability .

Procedure for Reduced form and Discrete Residues

To reduce the computational cost of computing discrete residues of a rational function, we will avoid the full partial fraction decomposition. The strategy behind reducing the computational cost is to find the "reduced" form of a given rational function such that the discrete residue of original function f is exactly equal to the classical residue of its reduced form. The reduced version of a given rational function plays a key role in development of algorithm. To find the reduced form of given rational function f, we first decompose it using Hermite reduction as

$$f(x) = (f_1(x))' + \frac{\tilde{a}_1(x)}{\tilde{b}_1(x)} \tag{1.3}$$

where $f_1(x)$ is some rational function and $\frac{\tilde{a}_1(x)}{\tilde{b}_1(x)}$ is a rational function with a square-free denominator. Now, we apply Hermite reduction to $f_1(x)$ in (1.3) to get

$$f_1(x) = (f_2(x))' + \frac{\tilde{a}_2(x)}{\tilde{b}_2(x)} \tag{1.4}$$

where $f_2(x)$ is some rational function and $\frac{\tilde{a}_2(x)}{\tilde{b}_2(x)}$ is a rational function with a square-free denominator. Continuing Hermite reduction to "rational part" in each iteration, we get

$$f_i(x) = (f_{i+1}(x))' + \frac{\tilde{a}_{i+1}(x)}{\tilde{b}_{i+1}(x)} \tag{1.5}$$

where $f_{i+1}(x)$ is some rational function and $\frac{\tilde{a}_{i+1}(x)}{\tilde{b}_{i+1}(x)}$ is a rational function with a square-free denominator. The reduced form $\frac{\tilde{a}_{i+1}(x)}{\tilde{b}_{i+1}(x)}$ in each iteration i can be efficiently found using only gcd computations. Once this reduced form is determined, the discrete residues associated with it can be computed using the Rothstein-Trager resultant, since the discrete residues of the original rational function are equal to the classical residues of the reduced form.

Applications to Creative Telescoping and Computation of Galois Groups

Once we are able to compute the discrete residues efficiently, the immediate application of this achievement is to solve telescoping problems. Let us briefly discuss these telescoping problems, and the procedure to solve them using discrete residues. Consider a finite collection of rational functions $f_1, f_2, \ldots, f_n \in C(x)$. The problem of finding (or deciding non-existence) of a system of linear operators $\mathfrak{L}_1, \ldots, \mathfrak{L}_n \in C[x]$ such that

$$\mathfrak{L}_1(f_1) + \cdots + \mathfrak{L}_n(f_n) = g(x+1) - g(x)$$

for some $g \in C(x)$. In fact, such a g exists if and only if all the discrete residues of the left-hand side are zero, i.e.,

$$\operatorname{dres}(\sum_i (\mathfrak{L}_i(f_i)), [\alpha], k) = 0 \ \forall \ k \in \mathbb{N}$$

. If we have already computed all the discrete residues of $f_1, \ldots, f_n$, this becomes a system of linear equations in the unknown coefficients of the operators $\mathcal{L}_1, \ldots, \mathcal{L}_n$.

Being able to solve these telescoping problems is one of the main ingredients in Arreche's algorithm [11] to compute (differential) Galois groups for systems of linear difference equations with respect to the shift operator.

CHAPTER 2

PRELIMINARIES

2.1 Background on Polynomial Algebra

In this section we will present some definitions and preliminary theorems. Here and elsewhere in this thesis, unless stated otherwise, we let C to be an algebraically closed field of characteristic zero.

2.1.1 Greatest Common Divisor of Polynomials

The computation of the greatest common divisor (GCD) of two polynomials holds significant importance in algebraic manipulation. Polynomial GCD computations emerge as subproblems in numerous scenarios. For instance, they play a prominent role in polynomial factorization and symbolic integration. Furthermore, they are essential when determining inverses in finite Galois extensions and simple algebraic extension fields. Thus, the computation of polynomial GCDs holds relevance across different mathematical contexts, contributing to a wide range of applications.

Let us revisit Euclid's algorithm, which can be seen as a constructive proof of the existence of greatest common divisors (GCDs) and remains a practical tool for manual calculations. When it comes to computing the GCD of two polynomials, namely $p(x)$ and $q(x)$ with coefficients from a field $C[x]$, Euclid's algorithm can be viewed as the process of generating a sequence of remainders. Specifically, if the degree of $p(x)$ is greater than or equal to the degree of $q(x)$, Euclid's algorithm constructs a sequence of polynomial $r_0(x), \cdots, r_k(x)$ where

$r_0(x) = p(x)$ and $r_1(x) = q(x)$ and

$$\begin{aligned}
r_0(x) &= r_1(x)a_1(x) + r_2(x) \text{ with } \deg(r_2(x)) < \deg(r_1(x)).\\
r_1(x) &= r_2(x)a_2(x) + r_3(x) \text{ with } \deg(r_3(x)) < \deg(r_2(x)).\\
&\dots\\
r_{k-2}(x) &= r_{k-1}(x)a_{k-1}(x) + r_k(x) \text{ with } \deg(r_k(x)) < \deg(r_{k-1}(x)).\\
r_{k-1}(x) &= r_k(x)a_k(x).
\end{aligned}$$

Then $r_k(x)$= GCD$(p(x), q(x))$. When appropriately normalized to achieve unit normality, the extended Euclidean algorithm yields the same results as Euclid's algorithm. However, it provides additional information by computing not only the remainder sequence $r_i(x)$ but also the sequences $\{s_i(x)\}$ and $\{t_i(x)\}$. These sequences satisfy the following conditions:

$$r_i(x) = p(x) \cdot s_i(x) + q(x) \cdot t_i(x) \text{ for all } i.$$

Here

$$\begin{aligned}
s_{i+1}(x) &= s_{i-1}(x) - s_i(x) \cdot a_i(x).\\
t_{i+1}(x) &= t_{i-1}(x) - t_i(x) \cdot a_i(x).
\end{aligned}$$

The quotient $a_i(x)$ is defined by division

$$r_{i-1}(x) = r_i(x) \cdot a_i(x) + r_{i+1}(x).$$

The initial conditions for these sequences are

$$s_0(x) = t_1(x) = 1, s_1(x) = t_0(x) = 0.$$

Example 2.1.1. Let $p(x)$ and $q(x)$ be polynomials from $\mathbf{Q}[x]$, given by

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

where $\mathbf{Q}[x]$ is the field of rational function. The remainders obtained from the process discussed above are

$$\begin{aligned} r_2(x) &= -\frac{5}{9}x^4 + \frac{1}{9}x^2 - \frac{1}{3}, \\ r_3(x) &= -\frac{117}{25}x^2 - 9x + \frac{411}{25}, \\ r_4(x) &= \frac{233150}{19773}x - \frac{102500}{6591}, \\ r_5(x) &= -\frac{1288744821}{543589225} \end{aligned}$$

Consequently, $p(x)$ and $q(x)$ are relatively prime since their gratest common divisior is unit in $\mathbf{Q}[x]$

2.1.2 Square-Free Factorization

Definition 2.1.2. A non-constant polynomial $p(x) \in C[x]$ over a field C is said to be *square-free* if it has no repeated factors, i.e., there does not exist any $q(x) \in C[x]$ with $\deg(q(x)) > 1$ such that $q(x)^2 \big| p(x)$.

The square-free factorization of $p(x)$ is

$$p(x) = a \prod_{i=1}^{k} p_i(x)^i \tag{2.1}$$

where each $p_i(x)$ is monic and square-free, a is the leading coefficient of $p(x)$ and $\gcd(p_i(x), p_j(x)) = 1$ for $i \neq j$.

Example 2.1.3. $p(x) = (x^2+2)(x^2-1)^5(x^3+4x)^7$ is a square free factorization.

Square-free factorization of polynomials are not only important in polynomial factorization but also have central importance in symbolic integration. There are several ways to calculate the square-free factorization of a polynomial $p(x) \in C[x]$, we will introduce some of them in the later sections.

As discussed in [6], the following theorem gives us a criterion to check if the given polynomial is square-free or not.

Theorem 2.1.4. A polynomial $p(x) \in C[x]$ is square-free if and only if $\gcd(p(x), p^{'}(x)) = 1$.

Proof. We will prove the contrapositive statement of the theorem. Forone direction, let us assume $p(x)$ is not square-free. Then it suffices to prove $g(x) = \gcd(p(x), p^{'}(x))$ is non-trivial. Since $p(x)$ has repeated roots, then for some non-constant $q(x)$ it can be written as

$$\begin{aligned} p(x) &= q(x)^2 \cdot r(x) \\ p'(x) &= 2 \cdot q(x)q'(x) \cdot r(x) + q(x)^2 \cdot r'(x) \\ &= q(x) \cdot (2 \cdot q'(x) \cdot r(x) + q(x) \cdot r'(x)) \\ &= q(x) \cdot \hat{q}(x) \end{aligned}$$

where $\hat{q}(x) = 2 \cdot q'(x) \cdot r(x) + q(x) \cdot r'(x) \in C[x]$ and prime indicates the first derivative with respect to x. Which shows that $g(x) = \gcd(p(x), p'(x))$ is nontrivial.

Conversely, let us consider $g(x) = \gcd(p(x), p'(x))$ is nontrivial and $p(x)$ is square-free. Since $p(x)$ is square-free, it can be factored as

$$p(x) = p_1(x) \cdot p_2(x) \cdots p_k(x)$$

where each $p_i(x)$ are irreducible such that $\deg(p_i(x)) \geq 1$ and are pairwise relatively prime. Then,

$$p^{'}(x) = p_1^{'}(x) \cdot p_2(x) \cdots p_k(x) + \cdots + p_1(x) \cdot p_2(x) \cdots p_k^{'}(x).$$

Assume, $p_i(x)|g(x)$ for some i. Without loss of generality, let us assume that $p_1(x)|g(x)$ then $p_1(x)|p'(x)$ so $p_1(x)|p_1'(x) \cdot p_2(x) \ldots p_k(x)$. But p_i are pairwise relatively prime then $p_1(x)|p_1'(x)$. This is possible only if $p_1' = 0$. But in a field of characteristic 0, this is possible only if $p_1(x)$ is constant, which is a contradiction. □

2.1.3 Algorithm to Compute Square-free Factorization

Theorem (2.1.4) gives us an easy way to see if a polynomial has repeated roots. Let us now discuss a process to calculate the square-free factorization of a given polynomial. Let $p(x)$ be a polynomial with square-factorization as defined in equation (2.1). Differentiating both sides we get,

$$p'(x) = a \sum_{i=1}^{k} p_1(x) \cdots i \cdot p_i(x)^{i-1} p_i'(x) \cdots p_k(x)^k. \tag{2.2}$$

Then,

$$g(x) = \gcd(p(x), p'(x)) = a \prod_{i=2}^{k} p_i(x)^{i-1}. \tag{2.3}$$

Let

$$q(x) = \frac{p(x)}{g(x)} = p_1(x) \cdot p_2(x) \ldots p_k(x). \tag{2.4}$$

Then, $q(x)$ is product of square-free factors without their multiplicities.

$$h(x) = \gcd(g(x), q(x)). \tag{2.5}$$

So,

$$p_1(x) = \frac{q(x)}{h(x)}. \tag{2.6}$$

gives us the first square-free factor of $p(x)$. The second square-free factor of $p(x)$ is the first square-free factor of $g(x)$.

$$\gcd(c(x), c'(x)) = \prod_{i=3}^{k} p_i(x)^{i-2} = \frac{g(x)}{g_1(x)}.$$

The product of remaining square-free factors are just $h(x)$.

We give an example of the process described in section (2.1.3) to calculate the square-free factorization of a given polynomial.

Example 2.1.5. Let $p(x) = x^8 - 2x^6 + 2x^2 - 1$. As described in the section (2.1.3), let us first calculate

$$\begin{aligned} p'(x) &= 8x^7 - 12x^5 + 4x \\ g(x) &= \gcd(p(x), p'(x)) = x^4 - 2x^2 + 1 \\ q(x) &= x^4 - 1 \end{aligned}$$

Since $g(x) \neq 1$ we calculate

$$\begin{aligned} h(x) &= \gcd(g(x), q(x)) = x^2 - 1 \\ z(x) &= \frac{q(x)}{h(x)} = x^2 + 1 \\ q(x) &= g(x) = x^2 - 1 \end{aligned}$$

where number of iteration is $i = 2$.

Repeating the entire process for the second time gives,

$$\begin{aligned} h(x) &= x^2 - 1 \\ z(x) &= 1 \\ q(x) &= x^2 - 1 \\ g(x) &= 1 \end{aligned}$$

where number of iterations is $i = 3$.

We notice that the output will be unchanged after the third iteration as $g(x)$ in iteration three is 1. So we terminate the process and return the final output. Thus, the final output is product of the output in the second iteration multiplied by $q(x)^3$. So, the required square-free factorization of $p(x) = (x^2 + 1) \cdot (x^2 - 1)^3$.

The aforementioned procedure in the section (2.1.3) provides a straightforward and comprehensible approach for accurately computing the square-free factorization of a given polynomial within characteristic zero domains. However, there exist alternative efficient methods for calculating square-free factorizations of polynomials. One such method is Yun's method, as described in [6]. More detail about square-free factorizations can be found in [12].

2.1.4 Partial Fraction Decompositions

The concept of partial fraction decompositions for rational functions is a well-established topic that finds applications in various fields such as Calculus, Control Theory, Differential Equations, and other branches of mathematics. It is known that every rational function admits a *theoretical* decomposition into partial fractions, defined as follows.

$$f(x) = \frac{a(x)}{b(x)} = p(x) + \sum_{i=1}^{m} \sum_{r=1}^{j_i} \frac{\beta_{ir}(x)}{(q_i(x))^r}$$

with $\deg(\beta_{ir}) < \deg(q_i)$.

Definition 2.1.6. Let $f(x) = \frac{a(x)}{b(x)} \in C[x]$ be a rational function with b monic. Then by the fundamental theorem of algebra, we can write

$$b(x) = (x - \alpha_1)^{j_1} \cdots (x - \alpha_m)^{j_m}$$

where $\alpha_1, \cdots, \alpha_m \in C$, and $j_1, \cdots, j_m \in \mathbb{N}$. The partial fraction decomposition of $f(x)$ is

$$f(x) = \frac{a(x)}{b(x)} = p(x) + \sum_{i=1}^{m} \sum_{r=1}^{j_i} \frac{\beta_{ir}}{(x - \alpha_i)^r}$$

where $p(x)$ is a polynomial (possibly 0) and $\beta_{ir} \in C$.

The process of finding the complete partial fraction decomposition of a rational function can be quite time-consuming and computationally expensive, especially if it is done without a pre-factored denominator. However, the computational cost can be significantly reduced if

the denominator is pre-factored, as the factorization of the denominator is typically the most expensive step in this process. To reduce the computational cost of computing the discrete residues of a rational function, we will first find a pre-factorization of the denominator to find a "partial" fraction decomposition of the given rational function.

2.1.5 Algorithm for Full Symbolic PFD of Rational Function

Let $f(x) = \frac{a(x)}{b(x)} \in C(x)$ be a rational function. Then, as discussed in [6], by the fundamental theorem of algebra, $f(x)$ can be written in the form of

$$f(x) = \frac{a(x)}{b(x)} = p(x) + \sum_{b(\beta)=0} \sum_{i=1}^{n_\beta} \frac{a_{\beta,i}}{(x-\beta)^i} \ . \tag{2.7}$$

where $p(x)$ is a polynomial in $C[x], a_{\beta,i} \in C$, and n_β is the multiplicity of pole β. Since computing actual value of β is difficult, we use various partial forms of this expression. The decomposition in the right hand side of (2.7) are not necessarily of degree 1. In their paper [4], Bronstein and Salvy discuss the method to compute the $a_{\beta,i}$ in (2.7) without any factorization using operations only in C. Please see [13] for more about algorithms for symbolic partial fraction decomposition.

2.1.6 Resultant of Polynomials

When working with polynomial roots, it is often necessary to determine whether two polynomials share roots or common factors. The most straightforward method is to factor both polynomials completely and compare the sets of roots. However, this approach can be computationally expensive for high degree or multivariate polynomials.

A more efficient method is to use the Euclidean algorithm to compute the greatest common divisor of the two polynomials. However, this method is not always feasible, as it requires both polynomials to be in a Euclidean domain, which is not always the case for all polynomial rings.

To address this issue, the concept of a resultant is used to determine efficiently whether two polynomials share common roots. The resultant, which is a polynomial function of the coefficients of the two polynomials, serves as an important criterion. It equals zero if and only if the two polynomials have a common root. This approach is computationally more efficient than factoring polynomials completely or using the Euclidean algorithm to compute the greatest common divisor, especially for high degree or multivariate polynomials that are not defined over fields or Euclidean domains.

Theorem 2.1.7. Let $p_1(x), p_2(x) \in C(x)$ have degrees n and m respectively and both greater than zero. Then $p_1(x)$ and $p_2(x)$ have a non-constant common factor if and only if there exist nonzero polynomials $A(x), B(x) \in C[x]$ such that $\deg(A(x)) \leq m-1, \deg(B(x)) \leq n-1$ and $A(x)p_1(x) + B(x)p_2(x) = 0$.

Proof. Let $c(x) \in C[x]$ be a non-constant common factor of two polynomials $p_1(x)$ and $p_2(x)$. Then $p_1(x)$ and $p_2(x)$ can be written as $p_1(x) = c(x) \cdot q_1(x)$ and $p_2(x) = c(x) \cdot q_2(x)$ for some $q_1(x), q_2(x) \in C[x]$

Here, $q_2(x) \cdot p_1(x) + (-q_1(x) \cdot p_2(x)) = q_2(x) \cdot c(x) \cdot q_1(x) - q_1(x) \cdot c(x) \cdot q_2(x) = 0$. Since $c(x)$ is non-constant polynomial, $\deg(c(x)) \geq 1$ and hence $\deg(q_2(x)) \leq m-1$ and $\deg(q_1(x)(x)) \leq n-1$ and indeed these are the required polynomials $A(x)$ and $B(x)$ in the statement.

To prove the converse of the statement, we argue by contradiction. Let us assume that two polynomials $A(x)$ and $B(x)$ in the statement exist such that $p_1(x)$ and $p_2(x)$ have no non-constant common factors i.e. $\gcd(p_1(x), p_2(x)) = 1$ then there exist two polynomials

$q(x), r(x) \in C[x]$ such that $q(x)\cdot_1 (x) + r(x) \cdot p_2(x)) = 1$.Then

$$\begin{aligned} A(x) &= 1 \cdot A(x) \\ &= (q(x) \cdot p_1(x) + r(x) \cdot p_2(x))A(x) \\ &= q(x) \cdot p_1(x) \cdot A(x) + r(x) \cdot p_2(x) \cdot A(x) \\ &= q(x)(-B(x) \cdot p_2(x)) + r(x) \cdot p_2(x) \cdot A(x) \\ &= (r(x)A(x) - q(x)B(x))p_2(x) \neq 0, \end{aligned}$$

which is a contradiction to the assumption that the degree of $A(x)$ is strictly less than m. □

Now we would like to present a technique to calculate the resultant of two polynomials on the basis of above theorem: Let

$$\begin{aligned} p_1(x) &= a_n x^n + \cdots + a_1 x + a_0, \\ p_2(x) &= b_m x^m + \cdots + b_1 x + b_0, \\ A(x) &= c_{m-1} x^{m-1} + \cdots + c_1 x + c_0, \\ B(x) &= d_{n-1} x^{n-1} + \cdots + d_1 x + d_0. \end{aligned} \tag{2.8}$$

From Theorem (2.1.7) we see that

$$\begin{aligned} A(x)p_1(x) + B(x)p_2(x) &= 0 \\ \Rightarrow a_n c_{m-1} + b_m d_{n-1} &= 0 \\ a_n c_{m-2} + a_{n-1} c_{m-1} + b_m d_{n-2} + b_{m-1} d_{n-1} &= 0 \\ a_n c_{m-3} + a_{n-1} c_{m-2} + a_{n-2} c_{m-1} + b_m d_{n-3} + b_{m-1} d_{n-2} + b_{m-2} d_{n-1} &= 0 \\ &\vdots \\ a_0 c_0 + b_0 d_0 &= 0 \end{aligned} \tag{2.9}$$

The equivalent coefficient matrix of this system of equation is an $(m+n) \times (m+n)$ matrix which is known as the Sylvester matrix and is given by:

$$\mathrm{Syl}(p_1, p_2, x) = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_n & 0 & 0 & \cdots & 0 \\ 0 & a_0 & a_1 & \cdots & a_{n-1} & a_n & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_0 & a_1 & \cdots & a_{n-1} & a_n \\ b_0 & b_1 & b_2 & \cdots & b_m & 0 & 0 & \cdots & 0 \\ 0 & b_0 & b_1 & \cdots & b_{m-1} & b_m & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & b_0 & b_1 & \cdots & b_{m-1} & b_m \end{bmatrix}.$$

Example 2.1.8. Consider two polynomials $p_1(x) = x^2 + 5x + 6 = (x-2)(x-3), p_2 = (x+1)^2 + 5x + 11 = (x-3)(x-4)$. Then,

$$\mathrm{Res}(p_1, p_2, x) = \begin{vmatrix} 1 & 5 & 6 & 0 \\ 0 & 1 & 5 & 6 \\ 1 & 7 & 12 & 0 \\ 0 & 1 & 7 & 12 \end{vmatrix} = 0.$$

The resultant is 0 since p_1 and p_2 share a common root $x = -3$.

Lemma 2.1.9. Let $p_1(x) = a_n x^n + \cdots + a_1 x + a_0, p_2(x) = b_m x^m + \cdots + b_1 x + b_0$ be two polynomials with the roots α_i and β_j in $C[x]$, where $1 \leq i \leq n$ and $1 \leq j \leq m$. Then the resultant of p_1 and p_2 relative to x is is a polynomial over C and is defined as

$$\mathrm{Res}(p_1, p_2, x) := a_n^m b_m^n \prod_{i,j} (\alpha_i - \beta_j)$$

.

Note: $Res(p_1, p_2, x) = a_n^m \prod_i p_2(\alpha_i) = (-1)^{nm} b_m^n \prod_j p_1(\beta_j)$. That is, the resultant is the product of either polynomial evaluated at each root including multiplicity of the other polynomial. Let us now discuss a theorem from [14] which gives us a criterion to check whether two polynomials share a common roots or not.

Theorem 2.1.10. The determinant of the Sylvester matrix $\text{Syl}(p_1, p_2, x)$ is a polynomial in the coefficients a_i, b_j of the polynomials $p_1(x)$ and $p_2(x)$. Moreover, $\det(\text{Syl}(p_1, p_2, x)) = \text{Res}(p_1, p_2, x)$.

We refer the reader to [15] for more on resultants of polynomials.

2.1.7 Hermite Reduction

Hermite reduction, introduced by Hermite over a century ago [16], is a procedure used to obtain a reduced form of a rational function. By employing polynomial operations, Hermite reduction transforms a given rational function into a form where the denominator is square-free. The primary objective of Hermite reduction for a proper rational function, denoted as $\text{HermiteReduction}(f) = (g, h)$, is to find expressions such that $f = g' + h$, where h possesses a square-free denominator.

We apply Hermite recursively to get a reduced form of a rational function:

$$\frac{a}{b} = \left(\frac{c}{d}\right)' + \frac{p}{q} \tag{2.10}$$

where $a, b, c, d, p, q \in C(x)$, $\deg(p) < \deg(q)$ and q is monic and square-free. If we integrate equation (2.10) on both sides, $\frac{c}{d}$ is called the rational part of integral as the remaining part can be expressed only by introducing logarithmic extensions (assuming this part is non-zero).

The following is the procedure of integrating (2.10) using this method.

Let $\frac{a}{b} \in C(x)$ be normalized such that $\gcd(a, b) = 1$ and b is monic. By Euclidean division b can be written as $a = b.s + r$ where $r, s \in C[x]$ are two polynomials such that either $r = 0$ or $\deg(r) < \deg(b)$. Then,

$$\int \frac{a}{b} = \int s + \int \frac{r}{b}.$$

Integrating the polynomial s is trivial and its integral which is the polynomial part is one contribution to the term $\frac{c}{d}$ in equation (2.10). To integrate $\frac{r}{b}$, we compute the square-free factorization of the denominator

$$b = \prod_{i=1}^{k} b_i^i$$

where $b^i (1 \leq i \leq k)$ is monic and square-free such that $\gcd(b_i, b_j) = 1$ for $i \neq j$, and $\deg(b_k) > 0$. We now compute the partial fraction decomposition of integral $\frac{r}{b} \in C(x)$ in the form

$$\frac{r}{b} = \sum_{i=1}^{k} \sum_{j=1}^{i} \frac{r_{ij}}{b_i^j}$$

, where $1 \leq i \leq k$ and $1 \leq j \leq i, r_{ij} \in C[x]$ and $\deg(r_{ij}) < \deg(b_i)$ if $\deg(b_i) > 0, r_{ij} = 0$ if $b_i = 1$. The integral $\frac{r}{b}$ can be written as

$$\int \frac{r}{b} = \sum_{i=1}^{k} \sum_{j=1}^{i} \int \frac{r_{ij}}{b_i^j}. \tag{2.11}$$

We apply Hermite reduction on the integral in the right hand side of (2.11) until each integral that remains has denominator which is square-free.

Let a particular nonzero integrand $\frac{r_{ij}}{b_i^j}$. with $j > 1$. Since b_i is square-free, $\gcd(b_i, b_i') = 1$. Then there exist polynomials $s, t \in C[x]$ such that

$$sb_i + tb_i' = r_{ij}, \tag{2.12}$$

where $\deg(s) < \deg(b_i) - 1$ and $\deg(t) < \deg(b_i)$. Dividing (2.12) by b_i^j

$$\int \frac{r_{ij}}{b_i^j} = \frac{s}{b_i^{j-1}} + \frac{tb_i'}{b_i^j}$$

Integrating the second integral on the right using integration by parts we get,

$$\int \frac{tb_i'}{b_i^j} = \frac{-t}{(j-1)b_i^{j-1}} + \int \frac{t'}{(j-1)b_i^{j-1}}$$

Thus the Hermite reduction is

$$\int \frac{r_{ij}}{b_i^j} = \frac{-t(j-1)}{b_i^{j-1}} + \int \frac{s + t'(j-1)}{b_i^{j-1}}$$

This process has produced a rational function which contributes to the term $\frac{c}{d}$ in (2.10) and the remaining integral has the power reduced by 1. If the numerator of the new integrand is zero, the reduction process terminates at that step. Otherwise, if $j - 1 = 1$, this integral contributes to the logarithmic part to be considered in the next subsection, and if $j - 1 > 1$ the reduction procedure is applied again.

By repeated application of this reduction process until the denominator of all remaining integrands are square-free, we obtain the complete rational part of the integral.

CHAPTER 3

RATIONAL SUMMABILITY OF A RATIONAL FUNCTION

This chapter delves into the concept of rational summability for a rational function and explores its determination using discrete residues. We will examine the discrete residues of a rational function and their role in identifying rational summability. Additionally, we will conclude this chapter by exploring various algorithms proposed by different researchers. These algorithms aim to determine the shift set of a rational function, a crucial component in obtaining the reduced form of a rational function. By establishing this reduced form, we can effectively calculate the discrete residues and establish correlations with the classical residues of a rational function.

Definition 3.0.1. Let $\alpha \in C$. Then the subset $\alpha + \mathbb{Z}$ is called the $\mathbb{Z}$-*orbit* of α. The $\mathbb{Z}$-orbit of α will be denoted by $[\alpha]$. If r_1 is a root of a polynomial $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$, then a root r_2 is said to be in the $\mathbb{Z}$-orbit of r_1 if $r_1 = r_2 + k$ for some $k \in \mathbb{Z}$.

Definition 3.0.2. The *dispersion* of a polynomial p is defined by

$$\text{disp}(p) := \max\{|\alpha - \beta| : p(\alpha) = p(\beta) = 0 \text{ and } \alpha, \beta \in C, \ \alpha - \beta \in \mathbf{Z}\}.$$

That is, the dispersion is the largest integer difference between any two roots of p. We say p is shift-free if $\text{disp}(p) = 0$.

Definition 3.0.3. Let $f \in C$ be a univariate rational function in x. We say f is *rational summable* if there exists a rational function r such that

$$\Delta r = r(x+1) - r(x) = f(x) \tag{3.1}$$

has a solution in the field $C(x)$.

Now, let us introduce the notion of residues (discrete) of a rational function as they play key role in determination of summability of a rational function.

Definition 3.0.4. Consider a rational function $f = \frac{a}{b} \in C(x)$ such that $a, b \in C(x)$ with $\gcd(a, b) = 1, b$ monic and square-free and $\deg(a) < \deg(b)$. Then f can uniquely be written as

$$f = p + \sum_{i=1}^{m} \sum_{j=1}^{n_i} \sum_{l=0}^{d_{i,j}} \frac{\alpha_{i,j,l}}{(x - (\beta_i + l))^j} \tag{3.2}$$

where, $p \in C[x], m, n_i, d_{i,j} \in \mathbb{Z}^+, \alpha_{i,j,l} \in C$ and β_i's are in distinct $\mathbb{Z}-$orbits. Then the sum $\sum_{l=0}^{d_{i,j}} \alpha_{i,j,l}$ is called the *discrete residue* of f in the $\mathbb{Z}-$orbit of $[\beta_i]$ with respect to x with multiplicity j and is denoted by $\text{dres}_x(f, [\beta_i], j)$. We have used a variation in notations while defining the discrete residues to incorporate the following lemma and theorem in [[5], proposition 2.5].

The notion of discrete residues has been used in [5] to determine the summability of a rational function as discussed in following lemma and theorem.

Lemma 3.0.5. Let $f = \sum_{l=0}^{d} \frac{\alpha_l}{(x-(\beta+l))^j}$, where $d, s \in \mathbb{N}$ and $\alpha_l, \beta \in C$ then f is rational summable in $C(x)$ if and only if $dres_x(f, [\beta], j)$ is zero.

Now, let us discuss the rational summability of a rational function in the algebraically closed field of characteristic zero using the notion of discrete residues as discussed in [5].

Theorem 3.0.6. A rational function $f = \frac{a}{b} \in C[x]$ such that $\gcd(a, b) = 1$ then f is rational summable in $C(x)$ if and only if $\text{dres}_x(f, [\beta], j)$ is zero for every $\mathbb{Z}-$orbit $[\beta]$ with $b(\beta) = 0$ of any multiplicity $j \in \mathbb{N}$.

3.1 Algorithms to Find Reduced Form of a Rational Function

In this section we will discuss the effort made by different researchers to find the reduced form of a rational function using the concept of shift set. We will propose an efficient way to get a reduced form of rational function that has shift free and square free denominator.

The function r in equation (3.1) is called *indefinite sum of* f. The problem of indefinite summation for an arbitrary rational function can be formulated as:

$$f(x) = r(x+1) - r(x) + g(x), \tag{3.3}$$

where the rational functions r, g are called *summable* and *residual* part of f. Acquiring rational functions r and g with denominator with least degree possible is important because together with r and g in (3.3) any pair $r + k, g$ for $k \in C$ is also solution of (3.3). If f is not summable, it implies that the problem has an infinite number of solutions with distinct residues. In such cases, the summable components of the solutions can become quite extensive. It is worth noting that for any given f, there exists a solution r and g in (3.3) with the denominator of r possessing an arbitrarily large degree. For simplicity, let us call finding a pair (r, g) in (3.3) with g having minimum degree denominator as problem (1) and finding a solution of problem (1) with minimum degree of the denominator of summable part r as problem (2) respectively.

The computation of closed forms for sum expressions holds significant relevance in the realm of combinatorics, showcasing a vital application of symbolic computation. Specifically, when dealing with the indefinite summation of a rational function, various algorithms are available that can be efficiently implemented within computer algebra systems like Maple. These algorithms enable effective and streamlined calculations, facilitating the determination of closed forms for such sum expressions.

We will briefly discuss a concise overview of the shift structure of a polynomial. We will illustrate this concept with relevant examples, as it serves as a crucial step in the development of algorithms proposed by Moenck, Paule, and Abramov to address problem (1). The concept of shift set of rational function will be also used on the algorithm to find the discrete residues of a rational function. We will discuss the implementation of shifit shet in finding discrete residues of a rational function in chapter 4. Let us now discuss shift

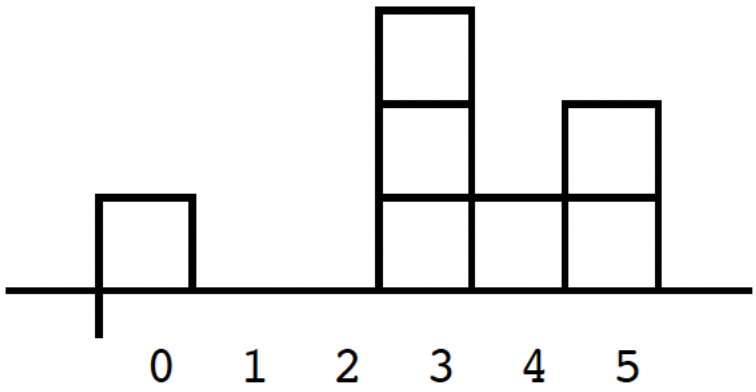

Figure 3.1. shift structure of $q(x)$

structure of the polynomial in some examples: Let

$$f(x) = \frac{p(x)}{q(x)} = \frac{x^2+1}{x^7+23x^6+218x^5+1090x^4+3033x^3+4455x^2+2700x}.$$

The factorization of q is

$$q(x) = x(x+4)(x+5)^2(x+3)^3$$

Collect all the irreducible factors $q_1(x), \ldots, q_l(x)$ of $q(x)$ which are shift equivalent i.e., for all $k \leq l$, $q_1(x) = q_k(x+i)$ for some integer i. In above example, all factors of q belong to the same class. Choosing $q_1(x) = x$,

$$q(x) = q_1(x)(q_1(x+3))^3 q_1(x+4)(q_1(x+5))^2$$

The graphical representation of this shift structure has shown in Figure 3.1 where we put m boxes on ith place on a line when $q_1(x+i)$ has multiplicity m as a factor of $q(x)$. The maximal distance of stacks of boxes in the figure gives the the *dispersion* of $q(x)$. In this example, the dispersion of q is 5. If there is more than one shift structure in the polynomial, then the dispersion is maximum among the dispersion of all the shift structures.

If a sum has a rational closed form, then the denominator of the summand must have a non-zero dispersion. Harmonic numbers, for example, do not possess a rational closed form, resulting in a summand of the form $\frac{1}{x}$ with zero dispersion. Consequently, the graph representing the shift structure of this particular type of rational function comprises a solitary box [10].

Moenck's Algorithm

Consider an example to illustrate the procedure of Moenck's algorithm: Let

$$f(x) = \frac{p(x)}{q(x)} = \frac{x^2+3}{x^2(x+2)^3(x+3)^2(x^2+2)(x^2+2x+3)^2}$$

The shift structure of the denominator q has two classes as shown in left of Figure 3.2. As the first step of the algorithm, we fill up the gap in each classes. For this, we insert boxes in each line to construct rectangles on it. This necessitates considering all factors with the highest multiplicity that emerge within that category. Following this procedure, we derive the updated denominator as illustrated on the right-hand side of Figure 3.2. As a result, the rational function is represented in the following form:

$$f(x) = \frac{x(x+1)^3(x+3)(x^2+2)(x^2+3)}{x^3(x+1)^3(x+2)^3(x+3)^3(x^2+2)^2(x^2+2x+3)^2}$$

There are two shift structures $q_1(x)$ and $q_2(x)$ of the denominator $q(x)$, where

$$q_1(x) = x^3(x+1)^3(x+2)^3(x+3)^3$$

$$q_2(x) = (x^2+2)^2(x^2+2x+3)^2$$

Since q_1 and q_2 are relatively prime, we can rewrite the rational function f in the form of

$$f(x) = \frac{p_1(x)}{q_1(x)} + \frac{p_2(x)}{q_2(x)} \tag{3.4}$$

where p_1 and p_2 are some polynomials.

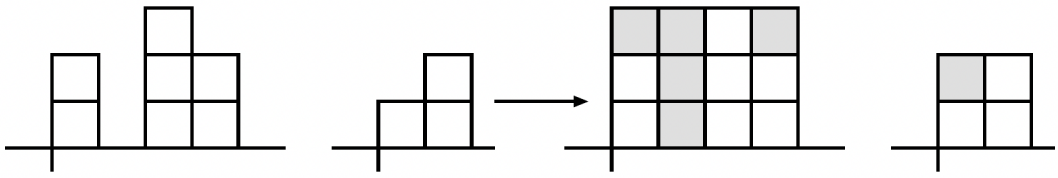

Figure 3.2. shift structure and saturation of $x^2(x+2)^3(x+3)^2(x^2+2)(x^2+2x+3)^2$

If r_1, g_1 and r_2, g_2 are solutions for first and second shift components of f then $r = r_1 + r_2$ and $g = g_1 + g_2$ will be the solution for problem (1) described above. The similar process works for the rational function which have more than two shift structures. Now, let us find the complete decomposition of the first class $\frac{p_1(x)}{q_1(x)}$.

$$\frac{p_1(x)}{q_1(x)} = \frac{p_1(x)}{x^3(x+1)^3(x+2)^3(x+3)^3} = \sum_{j=0}^{3} \frac{p_{1,j}(x)}{(x+j)^3(x+j+1)^3 \dots (x+3)^3} \tag{3.5}$$

where $\deg(p_{1,j}(x)) < \deg(x+3)^3 = 3$. We obtained a decomposition of each summand of (3.4) iteratively that has remainder with smaller dispersion after each step. Let $\alpha(x)$ and $\beta(x)$ be the solution of the equation

$$(x+j)^3\alpha(x) + ((x+3)^3 - (x+j)^3)\beta(x) = p_{1,j}(x)$$

for any summand on the right of (3.5)

Since $(x+j)^3$ and $(x+3)^3 - (x+j)^3$ are relatively prime,

$$\frac{p_{1,j}(x)}{(x+j)^3(x+j+1)^3 \dots (x+3)^3} = \frac{-\beta(x+1)}{(x+j+1)^3 \dots (x+3)^3} - \frac{-\beta(x)}{(x+j)^3(x+j+1)^3 \dots (x+3)^3} + \frac{\beta(x+1)-\beta(x)+\alpha(x)}{(x+j+1)^3 \dots (x+3)^3}$$

and one can gets a decomposition as in (3.3) for the jth summand. The iterative procedure will continue until we get either remainder zero or the remainder with zero dispersion. Finally, we sum up all the sub-results to get the solution $r(x)$ and $g(x)$ of equation (3.3).

This approach can be regarded as a discrete counterpart to Hermite's algorithm, which is utilized for the integration of rational functions.

Moenck's algorithm suffers from following shortcomings :

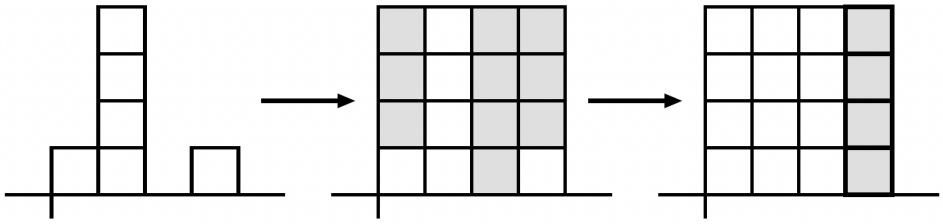

Figure 3.3. SSE and decomposition of the denominator

- While defining the shift saturation of denomination of rational function he does not make sure that the classes are relatively prime.

- He has not given any algorithm to fill up the gap of the shift structure of the denominator.

Paule's Algorithm

As in the previous section, we will consider an example to discuss Paule's algorithm: Let

$$f(x) = \frac{x^2+1}{x(x+1)^4(x+3)}$$

The shift structure of f is shown in Figure 3.3 . The denominator of rational function is said to be *shift saturated extension* (SEE) if each stack of boxes in each class has same height. We can find such saturation without actually factorizing the polynomials. Algorithms to find such saturation can be found in [17].

Now, let us reformulate equation (3.3) with $r(x) = \frac{\gamma(x)}{\delta(x)}$ and $g(x) = \frac{\epsilon(x)}{\eta(x)}$.

$$f(x) = \frac{\alpha(x)}{\beta(x)} = \frac{x^3(x+2)^4(x+3)^3(x^2+1)}{x^4(x+1)^4(x+2)^4(x+3)^4} = \frac{\gamma(x+1)}{\delta(x+1)} - \frac{\gamma(x)}{\delta(x)} + \frac{\epsilon(x)}{\eta(x)} \tag{3.6}$$

Paule proved that equation (3.6) can be solved by using an ansatz

$$\begin{aligned}\delta(x) &= \gcd(\beta(x), \beta(x-1) \\ &= x^4(x+1)^4(x+2)^4 \text{and} \\ \eta(x) &= \frac{\beta(x)}{\delta(x)} \\ &= (x+3)^4.\end{aligned}$$

As a result, for the non-summable remainder, we select only the rightmost boxes in each shift saturation class to form the denominator while the complete rectangle is utilized as the denominator for the rational part. Figure 3.3 displays the shift saturation extension of the rational function's denominator along with the decomposition into summable and non-summable components.

Substituting $\delta(x)$ and $\eta(x)$ in (3.6) we get

$$x^3(x+2)^2(x+3)^3(x^2+1) = x^4\gamma(x+1) - (x+3)^4\gamma(x) + \delta(x)\epsilon(x)$$

Solving the above equation we get

$$\gamma(x) = -\frac{x^3(128+336x+912x^2+1764x^3+2061x^4+1491x^5+661x^6+165x^7+18x^8)}{24}$$

$$\epsilon(x) = -\frac{25}{4} - 4x - \frac{3x^2}{4}$$

Thus the solution to equation (3.3) is

$$\begin{aligned}r(x) &= \frac{\gamma(x)}{\delta(x)} \\ &= -\frac{128+336x+912x^2+1764x^3+2061x^4+1491x^5+661x^6+165x^7+18x^8}{24x(x+1)^4(x+2)^4}\end{aligned}$$

and

$$\begin{aligned}g(x) &= \frac{\epsilon(x)}{\eta(x)} \\ &= -\frac{25+16x+3x^2}{4(x+3)^4}\end{aligned}$$

We also see that the solution obtained from this algorithm are not in reduced form in general.

Abramov's Algorithm

To discuss Abramov's Algorithm, we will continue with the example taken in Paul's algorithm. Let us now isolate right most boxes from the rest, i.e., $v(x) = x + 3$ and $w(x) = x(x + 1)^4$. Decomposing the rational function f in the form of

$$f(x) = \frac{x^2 + 1}{x(x + 1)(x + 3)} = \frac{5x^4 + 5x^3 + 15x^2 + x + 8}{24x(x + 1)^4} + \frac{-5}{24(x + 3)}$$

.

This decomposition can be written as

$$f(x) = u(x + 1) - u(x) + g(x) = \frac{\frac{-5}{24}}{x + 3} + \frac{\frac{5}{24}}{x + 2} + \frac{-1}{24}\left(\frac{5x^4 + 5x^3 - 9x^2 - x - 16}{x(x + 1)^4(x + 2)}\right) \quad (3.7)$$

where $u(x + 1) = \frac{-5}{24(x+3)}$.

Upon observation, we notice that the remainder g exhibits a dispersion of two, whereas the original function has a dispersion of three. To further analyze this, let us focus on the isolated boxes within the shift structure of $g(x)$ and continue this iterative process until we achieve a remainder with zero dispersion. In other words, we aim to obtain either an upper bound on $f(x)$ or elements that equal zero. The latter case signifies that the summation over $f(x)$ possesses a rational closed form.

Now, let us examine the shift structure of $g(x)$ in greater detail after each step. It comprises the boxes from the shift structure of the denominator of $f(x)$, where we remove the rightmost boxes and shift the remaining ones one place to the left. This process is illustrated in the right portion of the Figure 3.4 . The degree of the denominator in the rational part is dependent on the number of boxes that are erased.

To handle the shift structure in a more adaptable manner, a variation of this approach was proposed by Pirastu in [18]. The modification is rooted in a simple observation: instead of removing the rightmost boxes of each class to reduce dispersion, the leftmost boxes are erased, as illustrated in the left portion of Figure Figure 3.4. This modification offers increased

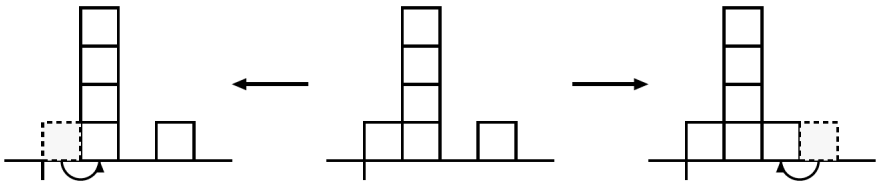

Figure 3.4. shift structure after one step in Abramov's algorithm

flexibility in handling the shift structure. Let us now decompose f with respect to stack $v(x) = x$ and to remainder $w(x) = (x+1)^4(x+3)$ such that

$$\begin{aligned} f(x) &= \frac{x^2+1}{x(x+1)^4(x+3)} \\ &= -\frac{x^4+7x^3+18x^2+19x+13}{3(x+1)^4(x+3)} + \frac{1}{3x} \\ &= u(x+1) - u(x) + g(x) \\ &= -\frac{1}{3(x+1)} + \frac{1}{3x} - \frac{9x^3+x^2+6x+10}{3(x+3)(x+1)^4} \end{aligned}$$

where $u(x) = -\frac{1}{3x}$. We see that the dispersion of the remainder $g(x)$ is two. So we can iterate again. From this we can get clear idea on deciding at which endpoints we want to erase or shift a stack of boxes at each step. In the work by Roboto [18], a proposition was made to determine the decision based on the degree of the resulting rational part's denominator. More specifically, the selection of boxes to be erased is based on the smaller degrees to which they correspond.

As a result of this modification, it is often observed that the rational part $r(x)$ yields a smaller degree for its denominator. In our example, the Abramov algorithm provides a rational part with a denominator of degree six, whereas this modification produces an output with a degree three denominator for the rational part. This discrepancy arises because Abramov's algorithm shifts four boxes at $(x+1)^4$, resulting in a remainder of x^4, while the modification only shifts the individual boxes corresponding to x, $(x+2)$, and $(x+3)$, yielding a remainder of $(x+1)^4$.

Please consider [19, 20, 21, 22] for more about algorithm for indefinite summation of rational function in Maple.

NOTE: All the figures in this chapter are extracted from [10].

CHAPTER 4

COMPUTATION OF DISCRETE RESIDUES

In this chapter, we will discuss the strategy behind our algorithms to compute the discrete residues of a rational function $f(x) \in C(x)$, based on a series of reductions producing successively "simpler" rational functions with "the same" discrete residues as those of the original $f(x)$ to compute the discrete residues practically and efficiently, without computing expensive factorizations.

4.1 Introduction

Let us explain the broad outline of our proposed algorithm to compute the discrete residues of a rational function $f(x) \in C(x)$, where C is an algebraically closed field of characteristic zero.

We know that $f(x) = p(x) + \frac{a(x)}{b(x)}$, where $p(x), a(x), b(x) \in C[x]$ are polynomials. Since the *polynomial part* $p(x)$ of $f(x)$ does not contribute to the discrete residues of $f(x)$ at all, we can systematically ignore it and make our first simplifying assumption.

Simplifying assumption 1: From now on we assume that $0 \neq f(x) = \frac{a(x)}{b(x)}$ is a proper non-zero rational function, with $\deg(a) < \deg(b)$ and $\gcd(a, b) = 1$.

We know that a proper rational function $f(x)$ admits a *theoretical* complete partial fraction decomposition

$$f(x) = \frac{a(x)}{b(x)} = \sum_{k \geq 1} \sum_{\alpha \in C} \frac{c_k(\alpha)}{(x - \alpha)^k} \tag{4.1}$$

where $c_k(\alpha) \in C$ are all zero for all but finitely many values of $k \in \mathbb{N}$ and $\alpha \in C$. In fact, $c_k(\alpha) = 0$ for every α such that $b(\alpha) \neq 0$, and for every $k > \deg(b)$.

Let us now define, for each $k \in \mathbb{N}$, the *degree* k part of the proper rational function $f(x)$

$$f_k(x) := \sum_{\alpha \in C} \frac{c_k(\alpha)}{(x - \alpha)^k} \tag{4.2}$$

where the $c_k(\alpha) \in C$ are precisely as in (4.1). Then, denoting by $g^{(n)}(x)$ the n-th derivative of any $g(x) \in C(x)$, with the convention that $g^{(0)}(x) := g(x)$, we observe that each $f_k(x) = \hat{f}_k^{(k-1)}(x)$, where

$$\hat{f}_k(x) := \frac{(-1)^{k-1}}{(k-1)!} \sum_{\alpha \in C} \frac{c_k(\alpha)}{x - \alpha}. \tag{4.3}$$

We see directly from the Definition 3.0.4 of discrete residues that, for each orbit $[\alpha] = \alpha + \mathbb{Z} \in C/\mathbb{Z}$ and for each $k \in \mathbb{N}$,

$$\operatorname{dres}(f, [\alpha], k) = \operatorname{dres}(f_k, [\alpha], k) = (-1)^{k-1}(k-1)! \cdot \operatorname{dres}(\hat{f}_k, [\alpha], 1). \tag{4.4}$$

Moreover, the $\hat{f}_k(x)$ can be computed directly from $f(x)$ without performing any factorizations, and indeed only using fast gcd computations, as follows.

Recall that the process of *Hermite reduction* takes as input a proper rational function $f(x)$ and returns a pair of rational functions $(g(x), h(x))$, such that $f(x) = g'(x) + h(x)$ and $h(x)$ has squarefree denominator. There exist several efficient implementations of Hermite reduction, and we emphasize that all of them rely only on fast factorization-free algorithms. For $f(x)$ given as in (4.1), we see that we must have

$$h(x) = \sum_{\alpha \in C} \frac{c_1(\alpha)}{x - \alpha} = f_1(x) = \hat{f}_1(x)$$

and

$$g(x) = \sum_{k \geq 2} \sum_{\alpha \in C} \frac{-1}{k-1} \cdot \frac{c_k(\alpha)}{(x-\alpha)^{k-1}} = \hat{f}_2(x) + \sum_{k \geq 3} \sum_{\alpha \in C} \frac{-1}{k-1} \cdot \frac{c_k(\alpha)}{(x-\alpha)^{k-1}},$$

with the $\hat{f}_k(x)$ defined as in (4.3).

These observations form the basis of the *Hermite list* algorithm, which takes as input a proper rational function $f(x)$ and outputs the finite list $(\hat{f}_1(x), \ldots, \hat{f}_d(x))$, where d is the largest multiplicity of any root of $b(x)$, or equivalently the largest value of k such that some $c_k(\alpha) \neq 0$. The *Hermite list* algorithm is defined recursively, by setting (initializing) $g_0(x) := f(x)$, and then setting

$$(g_k(x), \hat{f}_k(x)) := \text{HermiteReduction}(g_{k-1}(x))$$

for each $k \geq 1$. The algorithm stops when $g_k(x) = 0$, which happens to occur precisely at $k = d$.

The fact that we know how to efficiently compute the $\hat{f}_k(x)$ in (4.3) with the *Hermite list* algorithm, together with the observation (4.4), immediately leads us to our second simplifying assumption.

Symplifying assumption 2: From now on we assume that $0 \neq f(x) = \frac{a(x)}{b(x)}$ is a proper non-zero rational function, with $\deg(a) < \deg(b)$, $\gcd(a, b) = 1$, and squarefree denominator b.

For such a rational function $f(x)$, we can *theoretically* rewrite its complete partial fraction decomposition as

$$f(x) = \frac{a(x)}{b(x)} = \sum_{i=1}^{m} \sum_{n \geq 0} \frac{c_1(\alpha_i + n)}{x - n - \alpha_i}, \tag{4.5}$$

where $\alpha_1, \ldots, \alpha_m \in C$ is the set of roots of $b(x)$ which are *initial* in their orbit $[\alpha_i] = \alpha_i + \mathbb{Z}$ among all roots of $b(x)$, in the sense that $b(\alpha_i) = 0$ but $b(\alpha_i - n) \neq 0$ for every $n \geq 1$. We point out that this implies in particular the the α_i must belong to distinct orbits, that is, $\alpha_i - \alpha_j \notin \mathbb{Z}$ for $i \neq j$, or equivalently $[\alpha_i] \neq [\alpha_j]$ for $i \neq j$.

Let us now define, for $f(x)$ as in (4.5) and $n \geq 0$, the *n-shifted part* of $f(x)$ by

$$f_n(x) := \sum_{i=1}^{m} \frac{c_1(\alpha_i + n)}{x - n - \alpha_i}. \tag{4.6}$$

Let us write the reduced form of $f_n(x) = \dfrac{a_n(x)}{b_n(x)}$. We observe that

$$b_0(x) = \prod_{i=1}^{m} (x - \alpha_i); \qquad b(x) = \prod_{n \geq 0} b_n(x); \qquad \text{and}$$

$$b_n(x) = \gcd(b_0(x - n), b(x)) \quad \text{for } n \geq 0. \tag{4.7}$$

We emphasize that $b_n(x) = 1$ for all but finitely many $n \geq 0$, and that $\gcd(b_n(x), b_\ell(x)) = 1$ for $n \neq \ell$, since $b_0(x)$ is shift-free.

We now define the *shift-reduced form* of $f(x)$ by:

$$\bar{f}(x) := \sum_{n \geq 0} f_n(x+n).$$

Then we see immediately from the definition of the $f_n(x)$ in (4.6) that

$$\bar{f}(x) = \sum_{i=1}^{m} \frac{\sum_{n \geq 0} c_1(\alpha_i + n)}{x - \alpha_i} = \sum_{i=1}^{m} \frac{\operatorname{dres}(f, [\alpha_i], 1)}{x - \alpha_i}. \tag{4.8}$$

It follows from (4.8) that

$$\operatorname{dres}(f, [\alpha_i], 1) = \operatorname{res}(\bar{f}, \alpha_i, 1),$$

and therefore the problem of computing the discrete residues of $f(x)$ (again assuming that $f(x)$ is proper with squarefree denominator) is reduced to the problem of computing the classical first-order residues of the proper rational function $\bar{f}(x)$, which also has squarefree denominator $b_0(x)$.

Let us now explain how to compute the reduced form $\bar{f}(x)$ in (4.8) from $f(x) = \frac{a(x)}{b(x)}$, without performing any expensive factorizations. First, note that in order to compute $\bar{f}(x)$, it is enough to compute the pairwise relatively prime squarefree factors $b_n(x) = \text{denominator}(f_n(x))$, since the "partial decomposition" into partial fractions

$$f(x) = \sum_{n \geq 0} \frac{a_n(x)}{b_n(x)}$$

can be computed efficiently once the $b_n(x)$ are known. Moreover, we see from (4.7) that, in order to compute all the $b_n(x)$, it is sufficient to compute:

1. The *shift set* $S := \{n \in \mathbb{N} \mid b_n(x) \neq 1\}$; and

2. The *divisor of initial roots* $b_0(x)$.

Let us now explain how to compute the shift set S. We note that S can be defined equivalently as

$$S = \{n \in \mathbb{N} \mid \gcd(b(x), b(x+n)) \neq 1\}.$$

Let us consider the *resultant*

$$R(z) := \mathrm{Res}_x(b(x), b(x+z)) \in C[z],$$

where we consider z as an indeterminate. It follows from the well-known properties of the resultant that S can also be defined equivalently by

$$S = \{n \in \mathbb{N} \mid R(n) = 0\}.$$

There are many highly efficient algorithms to compute all the (positive) integer roots of a given polynomial with coefficients in C, such as $R(z)$. We note that our implementation of the computation of the shift set S exploits additional structural properties of the polynomial $R(z)$ in order to compute its positive integer roots even more efficiently.

Once we have efficiently computed the shift set S, we proceed with the computation of the divisor of initial roots $b_0(x)$. We first define, for each $n \in S$,

$$g_n(x) := \gcd(b(x), b(x-n)),$$

which clearly has the property that, for any $\alpha \in C$, $g_n(\alpha) = 0$ if and only if both $b(\alpha) = 0 = b(\alpha - n)$, i.e., the roots of $g_n(x)$ are precisely those roots α of $b(x)$ such that $\alpha - n$ is also a root of $b(x)$. Then we let

$$g(x) := \mathrm{lcm}(g_n(x) \mid n \in S),$$

which is then seen to have the property that the roots of $g(x)$ are precisely those roots α of $b(x)$ such that $\alpha - n$ is also a root of $b(x)$ for *some* $n \in \mathbb{N}$. Finally, we let

$$b_0(x) := \frac{b(x)}{g(x)},$$

which is seen to have the desired property that the roots of $b_0(x)$ are precisely those *initial roots* α_i of $b(x)$ such that $\alpha_i - n$ is *not* a root of $b(x)$ for any $n \in \mathbb{N}$.

As mentioned earlier, having computed the shift set S and the divisor of initial roots $b_0(x)$, the computation of the reduced form $\bar{f}(x)$ of $f(x)$ can be carried out efficiently. We

note that the denominator $b_0(x)$ of $\bar{f}(x)$ is shift-free by construction, which leads us to our last simplifying assumption.

Simplifying assumption 3: From now on we assume that $0 \neq f(x) = \frac{a(x)}{b(x)}$ is a proper non-zero rational function, with $\deg(a) < \deg(b)$, $\gcd(a, b) = 1$, and squarefree and shift-free denominator b.

For such a rational function $f(x)$, its discrete and classical first-order residues are the same, and it is known that the latter can be computed as the roots of the *Rothstein-Trager resultant*:

$$P(z) := \mathrm{Res}_x(a(x) - z \cdot b'(x), b(x)) \in C[z],$$

where we once again consider z as an indeterminate. Moreover, for each root $\gamma \in C$ of $P(z)$, the factor $v_\gamma(x)$ of $b(x)$, whose roots are those roots α of $b(x)$ such that

$$\mathrm{dres}(f, [\alpha], 1) = \gamma = \mathrm{res}(f, \alpha, 1),$$

is given by:

$$v_\gamma(x) := \gcd(a(x) - \gamma \cdot b'(x), b(x)).$$

We will now present the algorithms for performing sequential reduction in a rational function, with the aim of progressively simplifying it at each stage. These algorithms will enable the computation of discrete residues in each orbit, regardless of their order. The procedure described bellow for each algorithm is Maple specific.

4.2 Hermite List

The process of obtaining a sequence of proper rational functions with only simple poles through Hermite reduction applied to a given rational function is demonstrated in Algorithm (4.1). To generate this sequence of rational functions, built-in symbolic computation functions such as "ReduceHyperex" in Maple can be utilized.

Algorithm 4.1 Hermite List

Input: A proper rational function $0 \neq f(x) = \frac{a(c)}{b(x)}$ with $\deg(a) < \deg(b)$ and $\gcd(a, b) = 1$.
Output: The list $F = [\hat{f}_1, \ldots, \hat{f}_d]$, such that each $\hat{f}_k(x)$ has square free denominator and $f(x) = \sum_{k=1}^{d} \hat{f}_k^{(k-1)}$.

$k := 0$; $g := f$; # Initialize loop
while $g \neq 0$ **do:**
$(g, \hat{f}_{k+1}) :=$ HermiteReduction(g);
n := n + 1;
end do;
return $F := [\hat{f}_k;\ k = 1, \cdots, n]$

4.2.1 Procedure

The Hermite List Algorithm 4.1 takes a rational function f as input. We have taken $f(x)$ to be a non-zero proper rational function in which the numerator and denominator are already relatively prime, without loss of generality and to simplify the exposition.

The procedure then enters a loop that computes successive Hermite reductions of an auxiliary rational function g that gets modified in place until it becomes 0.

To describe in more detail the steps within the loop, let us consider the *theoretical* complete partial fraction decomposition of the rational function input

$$f = \sum_{k \geq 1} \sum_{\alpha \in C} \frac{c_k(\alpha)}{(x - \alpha)^k},$$

and define $f_k := \sum_{\alpha} \frac{c_k(\alpha)}{(x-\alpha)^k}$. Applying Hermite reduction to $g_0 := f$ we obtain $(g_1, \hat{f}_1)$, where $\hat{f}_1(x)$ has squarefree denominator and such that

$$f = g_0 = g_1^{'} + \hat{f}_1. \tag{4.9}$$

Then we see that in fact

$$g_1 = \sum_{k \geq 2} \sum_{\alpha \in C} \frac{(-1)^1 c_k(\alpha)}{(k-1)(x-\alpha)^{k-1}} \qquad \text{and} \qquad \hat{f}_1 = \sum_{\alpha \in C} \frac{c_1(\alpha)}{x - \alpha}.$$

The next step in the loop applies Hermite Reduction to g_1, obtaining $(g_2, \hat{f}_2)$ such that $\hat{f}_2$ has squarefree denominator and(4.9)

$$g_1 = g_2^{'} + \hat{f}_2. \tag{4.10}$$

Then we see that in fact

$$g_2 = \sum_{k\geq 3} \frac{(-1)^2 c_k(\alpha)}{(k-1)(k-2)(x-\alpha)^{k-2}} \qquad \text{and} \qquad \hat{f}_2 = \sum_{\alpha\in C} \frac{(-1)^1 c_2(\alpha)}{(2-1)(x-\alpha)^{2-1}}.$$

In the n-th step of the loop, we will apply Hermite reduction to g_{n-1} to obtain $(g_n, \hat{f}_n)$ such that $\hat{f}_n$ has squarefree denominator and

$$g_{n-1} = g_n^{'} + \hat{f}_n. \tag{4.11}$$

And we can see by induction that

$$g_n = \sum_{k\geq n+1} \sum_{\alpha\in C} \frac{(-1)^n (k-n-1)! c_k(\alpha)}{(k-1)!(x-\alpha)^{k-n}} \qquad \text{and} \qquad \hat{f}_n = \sum_{\alpha\in C} \frac{(-1)^{n-1} c_n(\alpha)}{(n-1)!(x-\alpha)}.$$

Since all the $c_k(\alpha)$ are zero for large enough k, we see that the loop does eventually stop, and it produces the correct output, as explained in the introduction to this chapter.

4.2.2 Example

Let us illustrate the process of Hermite reduction with examples.

Example 4.2.1.

$$f(x) = \frac{1}{x-1} + \frac{3}{(x-2)^2} + \frac{4}{(x+1)^5}$$

$$\text{HermiteList}(f) = \left[\frac{1}{x-1}, \frac{-3}{x-2}, 0, 0, \frac{1}{6(x+1)}\right]$$

Example 4.2.2.

$$f(x) = \frac{\begin{array}{l} 71744535x^{18} - 1218062772x^{17} + 7774273242x^{16} - 17196033267x^{15} - 44563254597x^{14} \\ + 341550187461x^{13} - 623240586144x^{12} - 390719940273x^{11} + 2402120809710x^{10} \\ + 2021761817337x^{9} - 21312891452782x^{8} + 42586197379426x^{7} - 35337018890316x^{6} \\ + 284910292072x^{5} + 20670566668304x^{4} - 9635315048352x^{3} - 5234354771520x^{2} \\ + 5576317246080x - 1291989753600 \end{array}}{(9x-5)(x-2)(3x-2)^2(3x-5)^3(x-3)^4(9x-14)^4(x+2)^4}$$

$$\text{HermiteList}(f) = \left[\frac{2}{(x-2)} + \frac{3}{(x-\frac{5}{9})}, -\frac{5}{x-\frac{2}{3}}, \frac{5}{2(x-\frac{5}{3})}, -\frac{(117x^2-174x-336)}{(6(9x-14)(x-3)(x+2))}\right]$$

4.3 Shift Set

The purpose of the Shift Set algorithm is to compute, for a given *squarefree* polynomial $b(x) \in C[x]$, the set S of all positive integers m for which there exists a root α of b such that $\alpha + m$ is also a root of b. The main application that we have in mind is to compute the shift set of the denominator $b(x)$ of a rational function $f(x) = \frac{a(x)}{b(x)}$ with only simple poles (i.e., with squarefree denominator $b(x)$). We emphasize that the original idea for how to compute the set S is not new, and was already proposed by Ambramov in [23]. But we explain how to carry out this computation as efficiently as possible.

4.3.1 Procedure

Let us explain why the Shift Set algorithm produces the correct output, and why it is as efficient as possible. In the first step, it checks whether $b(x)$ has at most one root: if this is the case, then the shift set is known a priori to be empty and there is nothing to compute. If $b(x)$ has degree at least 2, then we know from the properties of the resultant that the roots of $R(z) := \text{Resultant}_x(b(x), b(x+z))$ are precisely the values of $z \in C$ for which $\gcd(b(x), b(x+z))$ is non-trivial. Since we only care about the roots of this polynomial,

Algorithm 4.2 Shift Set

Input: A rational function $f = \frac{a(x)}{b(x)}$ such that $\deg(a) < \deg(b)$ and b is square-free.
Output: The S of all positive integers m such that there exists a root α of $b(x)$ such that $\alpha - m$ is also root of $b(x)$.
if $\deg(b) \leq 1$ **then** $S := \emptyset$;
else
$R(z) := \text{Resultant}_x(b(x), b(x+z))$;
$\tilde{R}(z) := \dfrac{R(z)}{z \cdot \gcd\left(R(z), \frac{dR}{dz}\right)}$;
$T(z) := \tilde{R}(z^{\frac{1}{2}})$;
$S := \{d \in \mathbb{N} \mid T(d^2) = 0\}$;
end if;
return S;

and not their multiplicities, we can divide $R(z)$ by $\gcd(R(z), R'(z))$ to obtain a squarefree polynomial with the same roots as $R(z)$. Since $z = 0$ is known ahead of time to be an uninteresting root of $R(z)$, we further divide by it to obtain the $\tilde{R}(z)$ defined in the algorithm, which has the same non-zero roots that $R(z)$ does. Not it is clear that if $z = \alpha$ is a non-zero root of $\tilde{R}(z)$, then $z = -\alpha$ is also a root, which implies that $\tilde{R}(z)$ has terms of even degree only, which implies that the algebraic function $T(z) = \tilde{R}(z^{\frac{1}{2}})$ defined in the algorithm is actually a polynomial, which has the property that α is a root of $T(z)$ if and only if $\pm\sqrt{\alpha}$ is a root of $\tilde{R}(z)$. Although we were only interested at the beginning in the positive integer roots of the polynomial $R(z)$, we see that the degree of $T(z)$ will in general be smaller than $\frac{1}{2}\deg(R)$, which will make it more efficient to find all its positive integer roots. The final step of the algorithm is to identify which positive integer roots of $T(z)$ happen to be perfect squares, and return the set S of positive integer roots of $R(z)$. We emphasize that in the implemented version of the algorithm it was more convenient to convert S into a list, but we have described the set version for simplicity.

4.3.2 Example

Example 4.3.1.

$$f(x) = \frac{1}{x-1} + \frac{3}{(x-2)^2} + \frac{4}{(x+1)^5}$$
$$\text{ShiftSet}(f) = [1, 2, 3]$$

Example 4.3.2.

$$f(x) = \frac{\begin{array}{l} 71744535x^{18} - 1218062772x^{17} + 7774273242x^{16} - 17196033267x^{15} - 44563254597x^{14} \\ + 341550187461x^{13} - 623240586144x^{12} - 390719940273x^{11} + 2402120809710x^{10} \\ + 2021761817337x^{9} - 21312891452782x^{8} + 42586197379426x^{7} - 35337018890316x^{6} \\ + 284910292072x^{5} + 20670566668304x^{4} - 9635315048352x^{3} - 5234354771520x^{2} \\ + 5576317246080x - 1291989753600 \end{array}}{(9x-5)(x-2)(3x-2)^2(3x-5)^3(x-3)^4(9x-14)^4(x+2)^4}$$
$$\text{ShiftSet}(f) = [1, 4, 5]$$

4.4 Shift Reduction for Rational Function with Simple Poles

The purpose of the following First Reduction Algorithm 4.3 is to compute, for $f(x) = \frac{a(x)}{b(x)} \in C(x)$ with only simple poles, a so-called "reduced form" $\bar{f}$ of f, such that

$$\bar{f}(x) = f(x) + g(x+1) - g(x)$$

for some $g(x) \in C(x)$ (which implies that the discrete residues of $\bar{f}$ and f are the same), and such that $\bar{f}$ has shift-free and square-free denominator, which implies that the classical residues of $\bar{f}$ are the same as its classical first-order residues.

4.4.1 Procedure

The First Reduction Algorithm 4.3 follows the procedure described in the introduction, to compute a reduced form for a proper rational function f with simple poles. It first checks

Algorithm 4.3 First Reduction

Input: A rational function $f = \frac{a(x)}{b(x)}$ such that $\deg(a) < \deg(b)$ and $b(x)$ is square-free.
Output: A reduced form $\bar{f}$ of f such that the denominator of $\bar{f}$ is square-free and shift-free.

$S := \text{ShiftSet}(f)$;
if $S = \emptyset$, **then** $\bar{f} := f$;
else for $k \in S$ **do**
$g_k := gcd(b(x), b(x-k))$;
end do;
$L := lcm(g_k;\ k \in S)$;
$b_0(x) := \frac{b(x)}{L(x)}$;
for $k \in S$ **do**
$b_k(x) := gcd(b_0(x-k), b(x))$;
end do;
$N := \{0\} \cup \{k \in S \mid \deg(b_k) \geq 1\}$;
compute the partial fraction decomposition $f(x) = \sum_{n \in N} \frac{a_n(x)}{b_n(x)}$;
$\bar{f}(x) := \sum_{n \in N} \frac{a_n(x+n)}{b_n(x+n)}$;
end if
return $\bar{f}$.

whether the denominator b of f is already shift-free: in this case, f is already reduced, so there is nothing to compute. Otherwise, we compute the set $S := \text{ShiftSet}(f)$ using Algorithm 4.2, consisting of positive integers k such that there exists a root α of $b(x)$ such that $\alpha - k$ is also a root of $b(x)$. For each $k \in S$, the rational function $g_k(x)$ defined in the algorithm has as its roots those roots $\alpha$of $b(x)$ such that $\alpha - k$ is also a root of $b(x)$ for this particular k. Then the least common multiple $L(x)$ of these $g_k(x)$ has as its roots those roots α of $b(x)$ such that $\alpha - k$ is also a root of $b(x)$ for some k. Finally the quotient $b_0(x) = (x)/L(x)$ has as its roots the remaining roots α of $b(x)$ such that no $\alpha - k$ is a root of $b(x)$ for any positive integer k. The roots of $b_0(x)$ are then seen to be the *initial* roots of $b(x)$ within each $\mathbb{Z}$-orbit. The polynomials $b_k(x) := \gcd(b_0(x-k), b(x))$ are the roots of $b(x)$ which lie at exactly k shifts from the initial root within their orbit, and therefore they are pairwise relatively prime, so the corresponding partial fraction decomposition of $f(x)$ with

respect to the factorization

$$b(x) = \prod_{n \in N} b_n(x)$$

of the denominator can be computed efficiently. It follows from their definition that $b_n(x + n) = \gcd(b_0(x), b(x + n))$, and therefore each $b_n(x + n)$ is a factor of $b_0(x)$, which is shift-free by construction. Therefore the denominator of $\bar{f}$ is a factor of $b_0(x)$, and is therefore shift-free. Finally, we observe that

$$\begin{aligned} \bar{f} := \sum_{n \in N} \frac{a_n(x+n)}{b_n(x+n)} &= \sum_{n \in N} \frac{a_n(x)}{b_n(x)} + \sum_{n \in N} \frac{a_n(x+n)}{b_n(x+n)} - \sum_{n \in N} \frac{a_n(x)}{b_n(x)} \\ &= f + \left(\sum_{n \in N} \frac{a_n(x+n)}{b_n(x+n)} - \sum_{n \in N} \frac{a_n(x)}{b_n(x)} \right). \end{aligned}$$

We conclude by observing that the rightmost term is summable for each n: trivially so for $n = 0$, and then for each $n \geq 1$ we have the following "telescoping-sum" relation

$$\frac{a_n(x+n)}{b_n(x+n)} - \frac{a_n(x)}{b_n(x)} = \sum_{k=0}^{n-1} \left(\frac{a_n(x+k+1)}{b_n(x+k+1)} - \frac{a_n(x+k)}{b_n(x+k)} \right),$$

which expresses $\bar{f} - f$ as a sum of summable rational functions.

4.4.2 Example

Example 4.4.1.

$$f(x) = \frac{1}{x-1} + \frac{3}{(x-2)^2} + \frac{4}{(x+1)^5}$$

$$\text{FirstReduction}(f) = \frac{8x^2 - 13x + 3}{(x-2)(x^2-1)}$$

Example 4.4.2.

$$f(x) = \frac{\begin{array}{l} 71744535x^{18} - 1218062772x^{17} + 7774273242x^{16} - 17196033267x^{15} - 44563254597x^{14} \\ + 341550187461x^{13} - 623240586144x^{12} - 390719940273x^{11} + 2402120809710x^{10} \\ + 2021761817337x^{9} - 21312891452782x^{8} + 42586197379426x^{7} - 35337018890316x^{6} \\ + 284910292072x^{5} + 20670566668304x^{4} - 9635315048352x^{3} - 5234354771520x^{2} \\ + 5576317246080x - 1291989753600 \end{array}}{(9x-5)(x-2)(3x-2)^2(3x-5)^3(x-3)^4(9x-14)^4(x+2)^4}$$

$$\text{FirstReduction}(f) = \frac{\begin{array}{l} 885735x^{10} - 17793432x^{9} + 158769639x^{8} - 827630298x^{7} + 2788080156x^{6} \\ - 6327302499x^{5} + 9743014206x^{4} - 9936219178x^{3} + 6271070715x^{2} \\ - 2093395185x + 235573485 \end{array}}{(x-3)^4(9x-14)^4(3x-5)^3}$$

4.5 Discrete Residues for Rational Functions

The purpose of the following PreDiscreteResidues Algorithm 4.4 is to compute the discrete residues of a proper rational function f which has shift-free and square-free denominator. For such an f, its discrete residues are the same as its classical residues. However, within the broader algorithm we wish to compute the discrete residues of the entries in the HermiteList Algorithm 4.1, which need to be modified by a multiplicative factor corresponding to the degree.

It is clear that the PreDiscreteResidues Algorithm 4.4 above simply computes the classical residues of a rational function f with square-free denominator, and then modifies them by the coresponding factor introduced by the HermiteList Algorithm 4.1, according to the second input k. Next we explain how to finally compute the discrete residues of a general rational function $f(x) \in C(x)$, this time with no serious simplifying assumptions at all (except for

Algorithm 4.4 PreDiscreteResidues

Input: A pair (f, k), consisting of a rational function $f = \frac{a(x)}{b(x)}$ such that $\deg(b) > \deg(a)$ and b is square-free and shift-free, and a positive integer k.
Output: A set D of pairs $((-1)^{k-1}(k-1)!\gamma, v_\gamma(x))$, where the γ are the first-order residues of f and $v_\gamma(x)$ is the factor of $b(x)$ of roots at which the first order residue is γ.
$R(z) := \text{Resultant}_x(a(x) - z \cdot \frac{db}{dx}(x), b(x))$;
$G := \{\gamma \in C \mid R(\gamma) = 0\}$;
for $\gamma \in G$ **do:**
$v_\gamma(x) := \gcd(a(x) - \gamma \cdot \frac{db}{dx}(x), b(x))$;
end do;
$D := \left\{ \left((-1)^{k-1}(k-1)!\gamma, v_\gamma(x)\right) \;\middle|\; \gamma \in G \right\}$;
return D

the simple assumptions that the rational function is proper and that its numerator and denominator are relatively prime), by simply applying the previous described algorithms.

Algorithm 4.5 DiscreteResidues

Input: A rational function $f(x) = \frac{a(x)}{b(x)}$ with $\deg(a) < \deg(b)$ and $\gcd(a, b) = 1$.
Output: A list of sets D_k for $k = 1, \ldots, d$, where D_k consists of pairs $(\gamma, v_\gamma(x))$, such that, for each $(\gamma, v_\gamma) \in D_k$, $\text{dres}(f, [\alpha], k) = \gamma$ for each root α of v_γ.
$F := HermiteList(f) = (\hat{f}_1, \ldots, \hat{f}_d)$;
$L := (FirstReduction(\hat{f}_k);\ k = 1, \ldots, d) = (\bar{f}_1, \ldots, \bar{f}_d)$;
$D_k :=$ PreDiscreteResidues$(\bar{f}_k, k)$
return $(D_1, \ldots, D_d)$

4.5.1 Procedure

Algorithm 4.4 takes a rational function $f(x) = \frac{a(x)}{b(x)}$ as input where $a(x)$ and $b(x)$ are polynomials with $\deg(b(x)) > \deg(a(x))$ and $b(x)$ is square-free.

The output of this algorithm is a list of pairs, where each pair contains a residue c_i and the factor of $b(x)$ whose roots are the poles at which $f(x)$ has residues c_i.

Step 1: Calculate the resultant of $a_r - z\frac{db_r}{dx}$ and b_r, where a_r is the numerator polynomial of $f(x)$ and b_r is the denominator polynomial. The resultant is a polynomial in the variable z that has a zero at the values of z for which $a_r - z\frac{db_r}{dx}$ and b_r has a common root.

Step 2: Find the roots of the resultant. The roots correspond to the poles of $f(x)$ with a residue c_i.

Step 3: For each root, calculate the residue c_i using the formula $(-1)^{(i-1)}(i-1)!c_i$, where i is the order of the pole and c_i is the residue.

Step 4: For each root, calculate the greatest common divisor of $a_r - D_R\frac{db_r}{dx}$ and b_r, where D_R is the set of roots of the resultant. This gives the factor of $b(x)$ whose roots are the poles at which $f(x)$ has residues c_i.

Step 5: Return a list of pairs, where each pair consists of a residue c_i and the factor of $b(x)$ whose roots are the poles at which $f(x)$ has residues c_i. This gives us a complete description of the residues and their corresponding factors of b(x).

4.5.2 Example

Example 4.5.1.

$$f(x) = \frac{1}{x-1} + \frac{3}{(x-2)^2} + \frac{4}{(x+1)^5}$$

$$\text{PreDiscreteResidues}(f, 2) = [[3, x-2]]$$

Example 4.5.2.

$$f(x) = \frac{\begin{aligned}&71744535x^{18} - 1218062772x^{17} + 7774273242x^{16} - 17196033267x^{15} - 44563254597x^{14}\\ &+ 341550187461x^{13} - 623240586144x^{12} - 390719940273x^{11} + 2402120809710x^{10}\\ &+ 2021761817337x^9 - 21312891452782x^8 + 42586197379426x^7 - 35337018890316x^6\\ &+ 284910292072x^5 + 20670566668304x^4 - 9635315048352x^3 - 5234354771520x^2\\ &+ 5576317246080x - 1291989753600\end{aligned}}{(9x-5)(x-2)(3x-2)^2(3x-5)^3(x-3)^4(9x-14)^4(x+2)^4}$$

$$\text{PreDiscreteResidues}(f, 3) = [[4, x-2], [6, 9x-5]]$$

4.6 Discrete Residues

4.6.1 Theoretical Overview

The main idea underlying behind the algorithm (4.5) is to make use of preceding algorithms sequentially to compute the discrete residues of a rational function. To achieve this goal, first we find the list of rational functions that has only simple poles foe an arbitrary rational function f by applying HermiteReduction algorithm. Once we get the sequence of rational function as the output of HermiteList algorithm, we apply FirstReduction algorithm to each rational function obtained from HermiteReduction.Finally, we apply DiscreteResidues alogorithm to each rational function obtained from PreDiscreteResidues to get the discrete residues of a rational function in any orbit and of any order.

4.6.2 Procedure

The algorithm (4.5) computes the first-order residues of a list of rational functions using the Hermite reduction and the PreDiscreteResidues algorithms.

The input of the algorithm is a rational function $f(x)$.

The first step is to compute the Hermite list H of $f(x)$ using the HermiteList algorithm. The Hermite list is a list of rational functions that are the Hermite reductions of $f(x)$, each having a smaller degree than the previous one.

Then, for each rational function in the Hermite list, the algorithm computes its reduced form using the FirstReduction algorithm. The reduced form is a rational function with a square-free and shift-free denominator.

Finally, for each reduced rational function, the algorithm computes its first-order residues using the PreDiscreteResidues algorithm. The first-order residues are the residues of the rational function at its simple poles.

The output of the algorithm is a list of pairs (c_i, b_i), where c_i is the first-order residue of the i^{th} reduced rational function at its simple poles and b_i is the factor of the denominator whose roots are the simple poles.

In summary, the algorithm works by first computing the Hermite list of the input rational function, then reducing each rational function in the list to its square-free and shift-free form, and finally computing the first-order residues of each reduced rational function at its simple poles. The output is a list of pairs that represent the first-order residues and the factors of the denominators whose roots are the simple poles of each reduced rational function.

4.6.3 Example

Example 4.6.1.

$$f(x) = \frac{1}{x-1} + \frac{3}{(x-2)^2} + \frac{4}{(x+1)^5}$$

$$\text{DiscreteResidues}(f) = [[1, x-1], [3, x-2][0, 0], [0, 0], [4, x+1]]$$

Example 4.6.2.

$$f(x) = \frac{\begin{aligned} &71744535x^{18} - 1218062772x^{17} + 7774273242x^{16} - 17196033267x^{15} - 44563254597x^{14} \\ &+ 341550187461x^{13} - 623240586144x^{12} - 390719940273x^{11} + 2402120809710x^{10} \\ &+ 2021761817337x^9 - 21312891452782x^8 + 42586197379426x^7 - 35337018890316x^6 \\ &+ 284910292072x^5 + 20670566668304x^4 - 9635315048352x^3 - 5234354771520x^2 \\ &+ 5576317246080x - 1291989753600 \end{aligned}}{(9x-5)(x-2)(3x-2)^2(3x-5)^3(x-3)^4(9x-14)^4(x+2)^4}$$

$$\text{DiscreteResidues}(f) = [[[2, x-2], [3, 9x-5]], [[5, 3x-2]], [[5, 6x-10]], [[6, x-3], [7, x-14/9]]]$$

CHAPTER 5

CONCLUSION AND FUTURE WORK

From the definition, computing discrete residues of rational function appears to require a complete partial fraction decomposition of it. However, this decomposition can be costly. To circumvent this issue, we have developed a series of algorithms that can effectively and practically compute the discrete residues for the shift case.

The strategy behind computation of discrete residues of a rational function is to perfom a squence of reduction and making it easier to compute discrete residues by reducing given rational function that has only simple pole such that its denominator is shift-free. For such a rational function $f(x)$, its discrete and classical first-order residues are the same, and it is known that the latter can be computed as the roots of the *Rothstein-Trager resultant*:

$$P(z) := \mathrm{Res}_x(a(x) - z \cdot b'(x), b(x)) \in C[z],$$

where we once again consider z as an indeterminate. Moreover, for each root $\gamma \in C$ of $P(z)$, the factor $v_\gamma(x)$ of $b(x)$, whose roots are those roots α of $b(x)$ such that

$$\mathrm{dres}(f, [\alpha], 1) = \gamma = \mathrm{res}(f, \alpha, 1),$$

is given by:

$$v_\gamma(x) := \gcd(a(x) - \gamma \cdot b'(x), b(x)).$$

Although our present implementation proceeds by finding the set of discrete residues $\gamma \in C$, by finding all the roots of $P(z)$, and then returning a list of pairs $(\gamma, v_\gamma(x))$, we point out that this last step is still very computationally expensive in general.

A potentially better way to proceed is following ideas of Salvy and Bronstein: for $f(x)$ satisfying the simplifying assumption 3 in the introduction of Chapter 4, one can compute a polynomial $H_f(x) \in C[x]$ with $\deg(H_f(x)) < \deg(b(x))$ having the property that

$$H_f(\alpha_i) = \mathrm{dres}(f, [\alpha_i], 1) = \mathrm{res}(f, \alpha_i, 1) \tag{5.1}$$

for each root α_i of $b(x)$. For most of the envisioned applications of discrete residues (telescoping problems and Galois theory), one does not actually need to know what they are (which is what our main goal was, and what our algorithm accomplishes). Rather, most of these applications call for finding or deciding whether some C-linear combinations of the discrete residues of some finite set of rational functions is zero – for this kind of application, the knowledge of the polynomials $H_f(x)$ should be sufficient. In future work, we plan to pursue this line of research further.

For now, let us explain how to compute the polynomial $H_f(x)$ as in (5.1). For $f(x)$ satisfying our simplifying assumption 3, let $\alpha_1, \ldots, \alpha_m$ be the set of roots of $b(x)$. One can show (cf. Bronstein-Salvy) that actually

$$f(x) = \sum_{i=1}^{m} \frac{a(\alpha_i)/b'(\alpha_i)}{x - \alpha_i}.$$

Therefore the desired polynomial $H_f(x) \in C[x]$ needs to have the property that

$$H_f(\alpha_i) = \frac{a(\alpha_i)}{b'(\alpha_i)} \qquad \text{for } i = 1, \ldots, m. \tag{5.2}$$

Finding such an $H_f(x)$ with degree smaller than that of $b(x)$ is now a basic computation in polynomial algebra: since $b(x)$ is squarefree, $\gcd(b(x), b'(x)) = 1$, and therefore there exist polynomials $p(x), q(x) \in C[x]$ such that

$$p(x) \cdot b(x) + q(x) \cdot b'(x) = 1. \tag{5.3}$$

These polynomials $p(x)$ and $q(x)$ are called *Bézout coefficients*, and they are found using the *extended gcd algorithm*. We then see from (5.3) that

$$q(\alpha_i) = \frac{1}{b'(\alpha_i)} \quad \Longrightarrow \quad a(\alpha_i) \cdot q(\alpha_i) = \frac{a(\alpha_i)}{b'(\alpha_i)}$$

for every root α_i of $b(x)$. Therefore the polynomial $a(x) \cdot q(x) \in C[x]$ already satisfies the desired property (5.2) that we want for $H_f(x)$, except that its degree might be larger than

that of $b(x)$. To conclude, we carry out polynomial division of $a(x) \cdot q(x)$ by $b(x)$, and let $H_f(x)$ be the associated remainder, so that now $\deg(H_f(x)) < \deg(b(x))$ and $H_f(x)$ still has property (5.2), because $a(x) \cdot q(x)$ has this property already and

$$a(x) \cdot q(x) = r(x) \cdot b(x) + H_f(x)$$

for some $r(x) \in C[x]$.

Once the discrete residues of a rational function have been computed for the shift case, the next step would be to calculate the discrete residues for more complex and extended cases, such as the q-dilation case and the Mahler case.

5.0.1 q-dilation case

Let us just introduce the notion of q-discrete residues of rational function.
Fix $q \in C^\times := C - \{0\}$ not a root of unity, and write $f(x) \in C(x)$:

$$f(x) = f_L(x) + \sum_{k \geq 1} \sum_{[\alpha]_q \in C^\times/q^{\mathbb{Z}}} \sum_{n \in \mathbb{Z}} \frac{c_k(q^n \alpha)}{(x - q^n \alpha)^k}$$

where $f_L(x) = \sum_{j \in \mathbb{Z}} r_j x^j \in C[x, x^{-1}]$ is a Laurent polynomial and $\alpha \in C^\times$ denote some choice of coset representatives for $[\alpha]_q := \alpha \cdot q^{\mathbb{Z}} \in C^\times/q^{\mathbb{Z}}$.

The q-discrete residue of $f(x)$ at the $q^{\mathbb{Z}}$-orbit $[\alpha]_q \in C^\times/q^{\mathbb{Z}}$ of order k (resp., at infinity) is defined as

$$q\text{-dres}(f, [\alpha]_q, k) := \sum_{n \in \mathbb{Z}} q^{-kn} c_k(q^n \alpha) \quad (\text{resp., } \; q\text{-dres}(f, \infty) := r_0).$$

Proposition 5.0.1 (Chen-Singer 2012)**.** $f(x)$ is rationally q-summable, i.e., there exists $g(x) \in C(x)$ such that $f(x) = g(qx) - g(x)$, if and only if q-dres$(f, \infty) = 0$ and q-dres$(f, [\alpha]_q, k) = 0$ for each $[\alpha]_q \in C^\times/q^{\mathbb{Z}}$ and $k \in \mathbb{N}$.

An open problem is to carry out a similar algorithm to compute q-discrete residues.

REFERENCES

[1] James H. Davenport, Yvon Siret, and Évelyne Tournier. *Computer algebra systems and algorithms for algebraic computation*. Academic Press Professional, Inc., 1993.

[2] Maurice Mignotte. *Mathematics for computer algebra*. Springer Science & Business Media, 2012.

[3] Joel S. Cohen. *Computer algebra and symbolic computation: Mathematical methods*. CRC Press, 2003.

[4] Manuel Bronstein and Bruno Salvy. Full partial fraction decomposition of rational functions. In *Proceedings of the 1993 international symposium on Symbolic and algebraic computation*, pages 157–160, 1993.

[5] Shaoshi Chen and Michael F Singer. Residues and telescopers for bivariate rational functions. *Advances in Applied Mathematics*, 49(2):111–133, 2012.

[6] K.O. Geddes, S.R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Springer US, 2007.

[7] Laura F. Matusevich. Rational summation of rational functions. *Beiträge Algebra Geom*, 41(2):531–536, 2000.

[8] Peter Paule. Greatest factorial factorization and symbolic summation. *Journal of symbolic computation*, 20(3):235–268, 1995.

[9] Carlos E. Arreche and Yi Zhang. Mahler discrete residues and summability for rational functions. In *Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation*, pages 525–533, 2022.

[10] Roberto Pirastu. Algorithms for indefinite summation of rational functions in maple. *way*, 15:4, 1995.

[11] Carlos E. Arreche. Computation of the difference-differential galois group and differential relations among solutions for a second-order linear difference equation. *Communications in Contemporary Mathematics*, 19(06):1650056, 2017.

[12] Kaltofen E. Factorization of polynomials. *Computer Algebra: Symbolic and Algebraic Computation*, pages 95–113, 1982.

[13] Manuel Bronstein and Bruno Salvy. Full partial fraction decomposition of rational functions. In *Proceedings of the 1993 international symposium on Symbolic and algebraic computation*, pages 157–160, 1993.

[14] Henry Woody. Polynomial resultants. *GNU operating system*, 2016.

[15] Griffiths H. Cayley's version of the resultant of two polynomials. *The American Mathematical Monthly*, 88(5):328–338, 1981.

[16] Jamil Baddoura. Integration in finite terms and simplification with dilogarithms: a progress report. In *Computers and Mathematics*, pages 166–171. Springer, 1989.

[17] Peter Paule. Greatest factorial factorization and symbolic summation. *Journal of symbolic computation*, 20(3):235–268, 1995.

[18] Roberto Pirastu. Algorithms for indefinite summation of rational functions in maple. *way*, 15:4, 1995.

[19] R. William Gosper Jr. Decision procedure for indefinite hypergeometric summation. *Proceedings of the National Academy of Sciences*, 75(1):40–42, 1978.

[20] Sergei A. Abramov. The rational component of the solution of a first order linear recurrence relation with rational right hand side. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 15(4):1035–1039, 1975.

[21] Robert Moenck. On computing closed forms for summations. In *Proceedings of MACSYMA users' conference, Berkley*, pages 225–236, 1977.

[22] Anu Pathria. Indefinite summation with maple. *The Maple Technical Newsletter*, 5:24–31, 1991.

[23] SA Abramov. On the summation of rational funcilons. 1971.

BIOGRAPHICAL SKETCH

Hari Prasad Sitaula was born in Panchthar, Nepal, in 1985. He earned his bachelor's degree in mathematics from Ratna Rajya Laxmi Campus in 2012 and went on to complete his master's in mathematics from Tribhuvan University, Nepal, in 2015.

In 2018, Hari was accepted into the PhD program in Mathematical Science at The University of Texas at Dallas (UTD). His doctoral research focused on developing algorithms to solve real-world problems, specifically in computing discrete residues of rational functions. His dissertation, titled "Algorithms for Efficiently Computing Discrete Residues of Rational Functions," involved the development of a sequence of algorithms that progressively reduce a rational function to a simpler form, making it easier to compute discrete residues using the Rothstein-Trager resultant.

In addition to his research, Hari was an active member of the UTD community. He served as a teaching assistant for several undergraduate courses, volunteered for community outreach programs, and participated in various student organizations.

Upon completing his PhD, Hari plans to continue his research in the field of Computer Algebra, with a particular interest in symbolic computation and data science. Starting August 2023, he will be joining Montana Tech University as an Assistant Professor of Mathematics.

CURRICULUM VITAE

Hari Prasad Sitaula

EDUCATION

- **PhD in Mathematics – The University of Texas at Dallas** 2023
  Department of Mathematical Sciences
- **MA in Mathematics – Tribhuvan University, Nepal** 2015
- **BA in Mathematics and Population – Tribhuvan University, Nepal** 2012

TEACHING EXPERIENCE

- **Graduate Teaching & Research Assistant** 2018 - 2023
  Department of Mathematical Sciences, UT Dallas
- **Mathematics Faculty Member** 2016 - 2018
  South Western State College

RESEARCH INTERESTS

Applied Mathematics, Artificial Intelligence, Data Science

LANGUAGES

English, Nepali, Hindi

PROFESSIONAL MEMBERSHIPS

- Association of Nepalese Mathematicians in America (ANMA) Life Member 2018
- Nepal Mathematical Society (NMS), Life Member 2016