Title
A Local Construction of the Smith Normal Form of a Matrix Polynomial, and Time-periodic Gravity-driven Water Waves

Author
Yu, Jia

Publication Date
2010

Peer reviewed|Thesis/dissertation

A Local Construction of the Smith Normal Form of a Matrix Polynomial, and Time-periodic Gravity-driven Water Waves

by

Jia Yu

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Applied Mathematics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jon A. Wilkening, Chair
Professor Alexandre J. Chorin
Professor John A. Strain
Professor Francisco Armero

Spring 2010

A Local Construction of the Smith Normal Form of a Matrix Polynomial, and Time-periodic Gravity-driven Water Waves

Abstract

A Local Construction of the Smith Normal Form of a Matrix Polynomial, and Time-periodic Gravity-driven Water Waves

by

Jia Yu

Doctor of Philosophy in Applied Mathematics

University of California, Berkeley

Professor Jon A. Wilkening, Chair

This dissertation consists of two separate chapters. In the first chapter, we present an algorithm for computing a Smith normal form with multipliers of a regular matrix polynomial over a field. This algorithm differs from previous ones in that it computes a local Smith form for each irreducible factor in the determinant separately and combines them into a global Smith form, whereas other algorithms apply a sequence of unimodular operations to the original matrix row by row (or column by column) to obtain the Smith normal form. The performance of the algorithm in exact arithmetic is reported for several test cases.

The second chapter is devoted to a numerical method for computing nontrivial time-periodic, gravity-driven water waves with or without surface tension. This method is essentially a shooting method formulated as a minimization problem. The objective function depends on the initial conditions and the proposed period, and measures deviation from time-periodicity. We adapt an adjoint-based optimal control method to rapidly compute the gradient of the functional. The main technical challenge involves handling the nonlocal Dirichlet to Neumann operator of the water wave equations in the adjoint formulation. Several families of traveling waves and symmetric breathers are simulated. In the latter case, we observe disconnections in the bifurcation curves due to nonlinear resonances at critical bifurcation parameters.

Contents

List of Figures

Acknowledgments

This dissertation consists of joint work with my advisor Jon Wilkening, and I would like to thank him for allowing me to use this work as the basis of my dissertation. I feel very lucky to have had chances to share his mathematical insights and ideas in last five years. His guidance, patience and great generosity, both mathematical and otherwise, have been invaluable to me.

I am very grateful to my family and Xinwen, for their love, support and encouragement, and to all my friends at Berkeley and beyond, for their friendship and giving me a home away from home.

I would like to thank Alexandre Chorin for his courses and for his guidance as my course advisor. I would also like to thank the other members of my committee, John Strain and Francisco Armero, for the mathematics they have taught me. I also thank the Applied Math community at University of California at Berkeley and Lawrence Berkeley National Laboratory, which has too many great mathematicians for me to acknowledge all of them here.

Chapter 1

A Local Construction of the Smith Normal Form of a Matrix Polynomial

1.1 Introduction

Canonical forms are useful tools for classifying matrices, identifying their key properties, and reducing complicated systems of equations to the de-coupled, scalar case, e.g. the Jordan form of a matrix over a field. When working with matrix polynomials over a field K, one fundamental canonical form, the Smith form, is defined. It is a diagonalization

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda) \tag{1.1}$$

of a given matrix $A(\lambda)$ by unimodular matrices $E(\lambda)$ and $F(\lambda)$ such that the diagonal entries $d_i(\lambda)$ of $D(\lambda)$ are monic polynomials and $d_i(\lambda)$ is divisible by $d_{i-1}(\lambda)$ for $i \geq 2$.

This factorization has various applications. The most common one involves solving the system of differential equations [34]

$$A^{(q)}\frac{d^q x}{dt^q} + \cdots + A^{(1)}\frac{dx}{dt} + A^{(0)}x = f(t), \tag{1.2}$$

where $A^{(0)}$, $\ldots$, $A^{(q)}$ are $n \times n$ matrices over $\mathbb{C}$. For brevity, we denote this system by $A(d/dt)x = f$, where $A(\lambda) = A^{(0)} + A^{(1)}\lambda + \cdots + A^{(q)}\lambda^q$. Assume for simplicity that $A(\lambda)$ is regular, i.e. $\det[A(\lambda)]$ is not identically zero, and that (1.1) is a Smith form of $A(\lambda)$. The system (1.2) is then equivalent to

$$\begin{pmatrix} d_1(\frac{d}{dt}) & & \\ & \ddots & \\ & & d_n(\frac{d}{dt}) \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix},$$

where $y = F(d/dt)x(t)$ and $g = E^{-1}(d/dt)f(t)$. Note that $E^{-1}(\lambda)$ is also a matrix polynomial over $\mathbb{C}$ due to the unimodularity of $E(\lambda)$. The system splits into n independent scalar

ordinary differential equations

$$d_i\Big(\frac{d}{dt}\Big)y_i(t) = g_i(t), \qquad 1 \le i \le n,$$

which can be solved for $y_i(t)$ separately. The solution of (1.2) is then given by $x = F^{-1}(d/dt)y$, where $F^{-1}(\lambda)$ is also a matrix polynomial over $\mathbb{C}$.

Another important application of the Smith form concerns the study of the algebraic structural properties of systems in linear control theory [43]. A close variant of the Smith form, the Smith-McMillan form of rational function matrices, plays an important role in linear systems in the sense that it reveals the finite eigenvalue structure, namely, the positions and multiplicities of its poles and zeros, of the system.

Smith forms of linear matrix polynomials can also be used to determine similarity of matrices. A fundamental theorem in matrix theory states that two square matrices A and B over a field K are similar if and only if their characteristic matrix polynomials $\lambda I - A$ and $\lambda I - B$ have the same Smith form $D(\lambda)$ [31, 34].

Other applications of this canonical form include finding the Frobenius form of a matrix A over a field by computing the invariant factors of the linear matrix polynomial $\lambda I - A$ [75, 77].

The computation of the Smith form of matrix polynomials over K with $K = \mathbb{Q}$ is a widely studied topic. Kannan [46] provides a polynomial-time algorithm for computing the reduced row-echelon form (Hermite form) of a matrix polynomial with unimodular row operations. Kaltofen, Krishnamoorthy and Saunders [44] gave the first polynomial-time algorithm for the Smith form without multipliers (also called transformations) using the Chinese remainder theorem. A new class of probabilistic algorithms (the Monte Carlo algorithms) were proposed by Kaltofen, Krishnamoorthy and Saunders [44, 45]. They showed that by pre-multiplying the given matrix polynomial by a randomly generated constant matrix on the right, the Smith form with multipliers is obtained with high probability by two steps of computation of the Hermite form. A Las Vegas algorithm given by Storjohann and Labahn [72, 73] significantly improved the complexity by rapidly checking the correctness of the result of the KKS algorithm. Villard [74, 76] established the first deterministic polynomial-time method to obtain the Smith form with multipliers by explicitly computing a good-conditioning matrix that replaces the random constant matrix in the Las Vegas algorithm. He also applied the method developed by Marlin, Labhalla and Lombardi [51] to obtain useful complexity bounds for the algorithm. Each of these methods is based on performing elementary row and column operations.

We propose a new deterministic algorithm for computing the Smith form of a matrix polynomial over a field K. Our approach differs from previous methods in the sense that we begin by constructing local diagonal forms and then combine them to obtain a (global) post-multiplier which generates the Smith form. This chapter is organized as follows: In Section 1.2, we begin by reviewing the definitions and theorems we will use. In Section 1.3, we describe our algorithm for computing a Smith form with multipliers of a matrix polynomial.

Since we do not discuss complexity bounds in this paper, we compare the performance of our algorithm to Villard's method with good conditioning [74, 76] in Section 1.4. We also include some discussion on the improvement in speed in Section 1.4. In Appendix A, we present a parallel theory in algebra that connects this work to [79], and give a variant of the algorithm in which all operations are done in the field K rather than manipulating polynomials as such.

1.2 Preliminaries

In this section, we describe the theory of the Smith form of matrix polynomials over a field K, following the definition in [34] over $\mathbb{C}$. In practice, K will be $\mathbb{Q}$, $\mathbb{Q}+i\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$, but it is convenient to deal with all these cases simultaneously. We also give a brief review of the theory of Jordan chains as well as Bézout's identity, which plays an important role in our algorithm for computing the Smith form of matrix polynomials.

1.2.1 Smith Form

Suppose $A(\lambda) = \sum_{k=0}^{q} A^{(k)}\lambda^k$ is an $n \times n$ matrix polynomial, where $A^{(k)}$ are $n \times n$ matrices whose entries are in a field K. In this paper we assume that $A(\lambda)$ is *regular*, i.e. the determinant of $A(\lambda)$ is not identically zero. The following theorem is proved in [34] (for $K = \mathbb{C}$).

Theorem 1. *There exist matrix polynomials $E(\lambda)$ and $F(\lambda)$ over K of size $n \times n$, with constant nonzero determinants, such that*

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda), \qquad D(\lambda) = \operatorname{diag}[d_1(\lambda), \ldots, d_n(\lambda)], \tag{1.3}$$

where $D(\lambda)$ is a diagonal matrix with monic scalar polynomials $d_i(\lambda)$ over K such that $d_i(\lambda)$ is divisible by $d_{i-1}(\lambda)$ for $i = 2, \ldots, n$.

Since $E(\lambda)$ and $F(\lambda)$ have constant nonzero determinants, (1.3) is equivalent to

$$U(\lambda)A(\lambda)V(\lambda) = D(\lambda), \tag{1.4}$$

where $U(\lambda) := (E(\lambda))^{-1}$ and $V := (F(\lambda))^{-1}$ are also matrix polynomials over K.

Definition 2. The representation in (1.3) or (1.4), or often $D(\lambda)$ alone, is called a *Smith form* (or *Smith normal form) of $A(\lambda)$. Square matrix polynomials with constant nonzero determinants like $E(\lambda)$ and $F(\lambda)$ are called* unimodular.

To distinguish, the term "Smith form with multipliers" is commonly used when referring to representation (1.3) or (1.4). In this paper, by "Smith form" we always mean the representation (1.3) or (1.4). The diagonal matrix $D(\lambda)$ in the Smith form is unique, while the representation (1.3) is not.

Example 1. *Consider*

$$A = \begin{pmatrix} 2\lambda^4+6\lambda^2+4 & \lambda^8+4\lambda^6-\lambda^5+6\lambda^4-5\lambda^3+7\lambda^2-6\lambda+6 \\ 2\lambda^3-2\lambda^2+4\lambda-4 & \lambda^7-\lambda^6+3\lambda^5-4\lambda^4+4\lambda^3-5\lambda^2+4\lambda-2 \end{pmatrix}$$

as a matrix polynomial over $\mathbb{Q}$. *We have*

$$A = \begin{pmatrix} \lambda^2+1 & \lambda+1 \\ \lambda-1 & 1 \end{pmatrix} \begin{pmatrix} \lambda^2+2 & 0 \\ 0 & (\lambda-1)^2(\lambda^2+2) \end{pmatrix} \begin{pmatrix} 2 & \lambda^4+\lambda^2-2\lambda+2 \\ 0 & 1 \end{pmatrix} \tag{1.5}$$

and also

$$A = \begin{pmatrix} \lambda^2+1 & \lambda^2+\lambda+2 \\ \lambda-1 & \lambda \end{pmatrix} \begin{pmatrix} \lambda^2+2 & 0 \\ 0 & (\lambda-1)^2(\lambda^2+2) \end{pmatrix} \begin{pmatrix} 2 & \lambda^4+1 \\ 0 & 1 \end{pmatrix}.$$

Both of them are Smith forms of A. We will use this matrix polynomial throughout the remainder of this chapter to explain the concepts and to demonstrate the algorithm.

Suppose that the determinant

$$\Delta(\lambda) := \det\left[A(\lambda)\right] \tag{1.6}$$

can be decomposed into prime elements $p_1(\lambda), \ldots, p_l(\lambda)$ in the principal ideal domain $K[\lambda]$, that is, $\Delta(\lambda) = c\prod_{j=1}^{l} p_j(\lambda)^{\kappa_j}$ where $c \neq 0$ is in the field K, $p_j(\lambda)$ is monic and irreducible, and κ_j are positive integers for $j = 1, \ldots, l$. Then the $d_i(\lambda)$ are given by

$$d_i(\lambda) = \prod_{j=1}^{l} p_j(\lambda)^{\kappa_{ji}}, \qquad (1 \leq i \leq n)$$

for some integers $0 \leq \kappa_{j1} \leq \cdots \leq \kappa_{jn}$ satisfying $\sum_{i=1}^{n} \kappa_{ji} = \kappa_j$ for $j = 1, \ldots, l$.

We now define a *local Smith form* of $A(\lambda)$ at $p(\lambda)$. Let $p(\lambda) = p_j(\lambda)$ be one of the irreducible factors of $\Delta(\lambda)$ and define $\alpha_i = \kappa_{ji}$, $\mu = \kappa_j$. Generalizing the case that $p(\lambda) = \lambda - \lambda_j$, we call μ the algebraic multiplicity of $p(\lambda)$.

Theorem 3. *Suppose A is an* $n \times n$ *matrix polynomial over a field K and* $p(\lambda)$ *is an irreducible factor of* $\Delta(\lambda)$ *in the principal ideal domain* $K[\lambda]$. *There exist* $n \times n$ *matrix polynomials* $E(\lambda)$ *and* $F(\lambda)$ *such that*

$$A(\lambda) = E(\lambda)D(\lambda)F(\lambda), \qquad D(\lambda) = \begin{pmatrix} p(\lambda)^{\alpha_1} & & 0 \\ & \ddots & \\ 0 & & p(\lambda)^{\alpha_n} \end{pmatrix}, \tag{1.7}$$

where $0 \leq \alpha_1 \leq \cdots \leq \alpha_n$ *are nonnegative integers and* $p(\lambda)$ *does not divide* $\det[E(\lambda)]$ *or* $\det[F(\lambda)]$.

We call (1.7) a *local Smith form* (with multipliers) of $A(\lambda)$ at $p(\lambda)$. Note that $D(\lambda)$, $E(\lambda)$ and $F(\lambda)$ in (1.7) are different from those in (1.3). $D(\lambda)$ is uniquely determined in a local Smith form, while $E(\lambda)$ and $F(\lambda)$ are not. In particular, we can impose the additional requirement that $F(\lambda)$ be unimodular by absorbing the missing parts $\prod_{s\neq j,s=1}^{l} p_j(\lambda)^{\kappa_{ji}}$ of $D(\lambda)$ into $E(\lambda)$ in a global Smith form (1.3). Then the local Smith form of $A(\lambda)$ at $p(\lambda)$ is given by

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \tag{1.8}$$

where $V(\lambda) := F(\lambda)^{-1}$ is a matrix polynomial.

Example 2. *Define the matrix $A(\lambda)$ as in Example 1. It is easy to check*

$$\Delta(\lambda) = \det\left[A(\lambda)\right] = (\lambda-1)^2(\lambda^2+2)^2.$$

From the (global) Smith form (1.5)*, we obtain*

$$A\begin{pmatrix} \frac{1}{2} & -\frac{1}{2}(\lambda^4+\lambda^2-2\lambda+2) \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda^4+3\lambda^2+2 & \lambda^3+\lambda^2+2\lambda+2 \\ \lambda^3-\lambda^2+2\lambda-2 & \lambda^2+2 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & (\lambda-1)^2 \end{pmatrix}$$

as a local Smith form of A about $(\lambda-1)$ and

$$A\begin{pmatrix} \frac{1}{2} & -\frac{1}{2}(\lambda^4+\lambda^2-2\lambda+2) \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda^2+1 & \lambda^3-\lambda^2-\lambda+1 \\ \lambda-1 & \lambda^2-2\lambda+1 \end{pmatrix}\begin{pmatrix} \lambda^2+2 & 0 \\ 0 & \lambda^2+2 \end{pmatrix}$$

as a local Smith form of A about (λ^2+2).

1.2.2 Multiplication and division in R/pR

We define $R = K[\lambda]$ and $M = R^n$. Note that R is a principal ideal domain and M is a free R-module of rank n. Suppose p is a prime element in R. Since p is irreducible, R/pR is a field and M/pM is a vector space over this field.

We use $\mathrm{quo}(\cdot,\cdot)$ and $\mathrm{rem}(\cdot,\cdot)$ to denote the quotient and remainder of polynomials:

$$g = \mathrm{quo}(f,p), \quad r = \mathrm{rem}(f,p) \qquad \Leftrightarrow \qquad f = gp + r, \quad \deg r < \deg p. \tag{1.9}$$

Multiplication and division in R/pR are easily carried out using the companion matrix of p. If we define

$$\gamma : K^s \to R/pR, \qquad \gamma\big(x^{(0)};\ldots;x^{(s-1)}\big) = x^{(0)} + \cdots + \lambda^{s-1}x^{(s-1)} + pR, \tag{1.10}$$

where $s := \deg p$, we can pull back the field structure of R/pR to K^s to obtain

$$\begin{aligned} xy = \gamma(x)(S)y = [x^{(0)}I + x^{(1)}S + \cdots + x^{(s-1)}S^{s-1}]y = [y, Sy, \ldots, S^{s-1}y]x, \\ \mathrm{quo}(x,y) = [y, Sy, \ldots, S^{s-1}y]^{-1}x, \qquad x = \big(x^{(0)};\ldots;x^{(s-1)}\big) \in K^s, \end{aligned} \tag{1.11}$$

where

$$S = \begin{pmatrix} 0 & \dots & 0 & -a_0 \\ 1 & \ddots & \vdots & \vdots \\ & \ddots & 0 & -a_{s-2} \\ 0 & & 1 & -a_{s-1} \end{pmatrix}, \quad p(\lambda) = a_0 + a_1\lambda + \cdots + a_{s-1}\lambda^{s-1} + \lambda^s \tag{1.12}$$

is the companion matrix of p, and represents multiplication by λ in R/pR. The matrix $[y, Sy, \dots, S^{s-1}y]$ is invertible when $y \neq 0$ since a non-trivial vector x in its kernel would lead to non-zero polynomials $\gamma(x), \gamma(y) \in R/pR$ whose product is zero (mod p), which is impossible as p is irreducible.

1.2.3 Jordan Chains

An irreducible factor $p(\lambda)$ in $\mathbb{C}[\lambda]$ takes the form $p(\lambda) = \lambda - \lambda_0$. Finding a local Smith form with multipliers of a matrix polynomial over $\mathbb{C}$ at $p(\lambda)$ is equivalent to finding a canonical system of Jordan chains [33, 79] for $A(\lambda)$ at λ_0. We now generalize the notion of Jordan chain to the case of an irreducible polynomial over a field K.

Definition 4. Suppose $A(\lambda)$ is an $n \times n$ matrix polynomial over a field K and $p(\lambda)$ is irreducible in $K[\lambda]$. A vector polynomial $x(\lambda) \in K[\lambda]^n$ of the form

$$x(\lambda) = x^{(0)}(\lambda) + p(\lambda)x^{(1)}(\lambda) + \cdots + p(\lambda)^{\alpha-1}x^{(\alpha-1)}(\lambda) \tag{1.13}$$

with $\alpha \geq 1$ and $\deg x^{(k)}(\lambda) < s := \deg p(\lambda)$ for $k = 0, \dots, \alpha - 1$, is called a *Jordan chain of length* α for $A(\lambda)$ at $p(\lambda)$ if

$$A(\lambda)x(\lambda) = O(p(\lambda)^\alpha) \tag{1.14}$$

and $x^{(0)}(\lambda) \not\equiv 0$. The meaning of (1.14) is that each component of $A(\lambda)x(\lambda)$ is divisible by $p(\lambda)^\alpha$. Any vector polynomial $x(\lambda)$ satisfying (1.14) such that $p(\lambda) \nmid x(\lambda)$ is called a *root function of order* α for $A(\lambda)$ at $p(\lambda)$.

We generally truncate or zero-pad $x(\lambda)$ to have α terms in an expansion in powers of $p(\lambda)$ when referring to it as a Jordan chain. If K can be embedded in $\mathbb{C}$, (1.14) implies that over $\mathbb{C}$, $x(\lambda)$ is a root function of $A(\lambda)$ of order α at each root λ_j of $p(\lambda)$ simultaneously.

Definition 5. Several vector polynomials $\{x_j(\lambda)\}_{j=1}^\nu$ form a *system of root functions* at $p(\lambda)$ if

$$\begin{aligned} &A(\lambda)x_j(\lambda) = O(p(\lambda)^{\alpha_j}), \qquad (\alpha_j \geq 1, \quad 1 \leq j \leq \nu) \\ &\text{the set } \{\dot{x}_j(\lambda)\}_{j=1}^\nu \text{ is linearly independent in } M/pM \text{ over } R/pR, \\ &\qquad\qquad \text{where } R = K[\lambda],\ M = R^n,\ \dot{x}_j = x_j + pM. \end{aligned} \tag{1.15}$$

It is called *canonical* if (1) $\nu = \dim \ker \dot{A}$, where $\dot{A}$ is the linear operator on M/pM induced by $A(\lambda)$; (2) $x_1(\lambda)$ is a root function of maximal order α_1; and (3) for $i > 1$, $x_i(\lambda)$ has

maximal order α_i among all root functions $x(\lambda) \in M$ such that $\dot{x}$ is linearly independent of $\dot{x}_1, \ldots, \dot{x}_{i-1}$ in M/pM. The integers $\alpha_1 \geq \cdots \geq \alpha_\nu$ are uniquely determined by $A(\lambda)$. We call ν the *geometric multiplicity* of $p(\lambda)$.

Definition 6. An *extended system of root functions* $x_1(\lambda),\ldots,x_n(\lambda)$ is a collection of vector polynomials satisfying (1.15) with ν replaced by n and α_j allowed to be zero. The extended system is said to be *canonical* if, as before, the orders α_j are chosen to be maximal among root functions not in the span of previous root functions in M/pM; the resulting sequence of numbers $\alpha_1 \geq \cdots \geq \alpha_\nu \geq \alpha_{\nu+1} = \cdots = \alpha_n = 0$ is uniquely determined by $A(\lambda)$.

Given an extended system of root functions (not necessarily canonical), we define the matrices

$$V(\lambda) = [x_1(\lambda), \ldots, x_n(\lambda)], \tag{1.16}$$

$$D(\lambda) = \operatorname{diag}[p(\lambda)^{\alpha_1}, \ldots, p(\lambda)^{\alpha_n}], \tag{1.17}$$

$$E(\lambda) = A(\lambda)V(\lambda)D(\lambda)^{-1}. \tag{1.18}$$

$E(\lambda)$ is a polynomial since column j of $A(\lambda)V(\lambda)$ is divisible by $p(\lambda)^{\alpha_j}$. The following theorem shows that aside from a reversal of the convention for ordering the α_j, finding a local Smith form is equivalent to finding an extended canonical system of root functions:

Theorem 7. *Suppose $A(\lambda)$, $V(\lambda) = [x_1(\lambda), \ldots, x_n(\lambda)]$, $D(\lambda) = \operatorname{diag}[p(\lambda)^{\alpha_1}, \ldots, p(\lambda)^{\alpha_n}]$ and $E(\lambda)$ are regular $n \times n$ matrix polynomials over a field K such that $A(\lambda)V(\lambda) = E(\lambda)D(\lambda)$, where $p(\lambda)$ is irreducible in $K[\lambda]$, $\alpha_1 \geq \cdots \geq \alpha_n \geq 0$, and the set $\{\dot{x}_j(\lambda)\}_{j=1}^n$ is linearly independent in M/pM over R/pR ($R = K[\lambda]$, $M = R^n$, $\dot{x}_j = x_j + pM$). The following three conditions are equivalent:*

(1) the columns $x_j(\lambda)$ of $V(\lambda)$ form an extended canonical system of root functions for $A(\lambda)$ at $p(\lambda)$.

(2) $p(\lambda) \nmid \det[E(\lambda)]$.

(3) $\sum_{j=1}^n \alpha_j = \mu$, where μ is the algebraic multiplicity of $p(\lambda)$ in $\Delta(\lambda)$.

This theorem is proved e.g. in [33] for the case that $K = \mathbb{C}$. The proof over a general field K is identical, except that the following lemma is used in place of invertibility of $E(\lambda_0)$. This lemma also plays a fundamental role in our construction of Jordan chains and local Smith forms.

Lemma 8. *Suppose K is a field, p is an irreducible polynomial in $R = K[\lambda]$, and $E = [y_1, \ldots, y_n]$ is an $n \times n$ matrix with columns $y_j \in M = R^n$. Then $p \nmid \det E \Leftrightarrow \{\dot{y}_1, \ldots, \dot{y}_n\}$ are linearly independent in M/pM over R/pR.*

Proof. The $\dot{y}_j$ are linearly independent iff the determinant of $\dot{E}$ (considered as an $n \times n$ matrix with entries in the field R/pR) is non-zero. But

$$\det \dot{E} = \det E + pR, \tag{1.19}$$

where $\det E$ is computed over R. The result follows. □

Example 3. *Define the matrix polynomial $A(\lambda)$ as in Example 1. From (1.5) we see that both $(1/2, 0)^T$ and $(-(\lambda^4+\lambda^2-2\lambda+2)/2, 1)^T$ are root functions of order 1 of $A(\lambda)$ at $p(\lambda) = \lambda^2+2$. Truncating higher order terms and keeping the zeroth order term in an expansion in powers of $p(\lambda)$, we obtain two Jordan chains of length 1 for $A(\lambda)$ at p: $(1/2, 0)^T$ and $(\lambda - 2, 1)^T$. They form a canonical system of root functions of $A(\lambda)$ at $p(\lambda)$, which provides us another local Smith form at $p(\lambda) = \lambda^2 + 2$:*

$$A \begin{pmatrix} \frac{1}{2} & \lambda - 2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda^2 + 1 & \lambda^6 + 2\lambda^4 + \lambda^3 - 2\lambda^2 - \lambda - 1 \\ \lambda - 1 & \lambda^5 - \lambda^4 + \lambda^3 - 4\lambda + 3 \end{pmatrix} \begin{pmatrix} \lambda^2 + 2 & 0 \\ 0 & \lambda^2 + 2 \end{pmatrix}.$$

The determinant of the first matrix on the right-hand side is $2(\lambda-1)^2$, which is not divisible by $p(\lambda)$.

1.2.4 Bézout's Identity

As $K[\lambda]$ is a principal ideal domain, Bézout's Identity holds, which is our main tool for combining local Smith forms into a single global Smith form. We define the notation $\gcd(f_1, \ldots, f_l)$ to be 0 if each f_j is zero, and the monic greatest common divisor (GCD) of $f_1, \ldots, f_l$ over $K[\lambda]$, otherwise.

Theorem 9. (Bézout's Identity) *For any two polynomials f_1 and f_2 in $K[\lambda]$, where K is a field, there exist polynomials g_1 and g_2 in $K[\lambda]$ such that*

$$g_1 f_1 + g_2 f_2 = \gcd(f_1, f_2). \tag{1.20}$$

Bézout's Identity can be extended to combinations of more than two polynomials:

Theorem 10. (Generalized Bézout's Identity) *For any scalar polynomials $f_1, \ldots, f_l$ in $K[\lambda]$, there exist polynomials $g_1, \ldots, g_l$ in $K[\lambda]$ such that*

$$\sum_{j=1}^{l} g_j f_j = \gcd(f_1, \ldots, f_l).$$

The polynomials g_j are called Bézout's coefficients *of $\{f_1, \ldots, f_l\}$.*

In particular, suppose we have l distinct prime elements $\{p_1, \ldots, p_l\}$ in $K[\lambda]$, and f_j is given by $f_j = \prod_{k \neq j}^{l} p_k^{\beta_k}$ $(j = 1, \ldots, l)$, where $\beta_1, \ldots, \beta_l$ are given positive integers and the notation $\prod_{k \neq j}^{l}$ indicates a product over all indices $k = 1, \ldots, l$ except $k = j$. Then $\gcd(f_1, \ldots, f_l) = 1$, and we can find $g_1, \ldots, g_l$ in $K[\lambda]$ such that

$$\sum_{j=1}^{l} g_j f_j = 1. \tag{1.21}$$

In this case, the Bézout's coefficients g_j are uniquely determined by requiring $\deg(g_j) < s_j \beta_j$, where $s_j = \deg(p_j)$. The formula (1.21) modulo p_k shows that g_k is not divisible by p_k.

Example 4. *Suppose we have two prime elements $p_1 = \lambda - 1$ and $p_2 = \lambda^2 + 2$ in $\mathbb{Q}[\lambda]$. Define $f_1 = \lambda^2 + 2$ and $f_2 = (\lambda - 1)^2$. We find that $g_1 = (-2\lambda + 5)/9$ and $g_2 = (2\lambda - 1)/9$ satisfy $g_1 f_1 + g_2 f_2 = 1$. g_1 and g_2 are uniquely determined if we require $deg(g_1) < s_1\beta_1 = 2$ and $deg(g_2) < s_2\beta_2 = 2$.*

1.3 An Algorithm for Computing a (Global) Smith Form

In this section, we describe an algorithm for computing a Smith form of a regular $n \times n$ matrix polynomial $A(\lambda)$ over a field K. We have in mind the case where $K = \mathbb{C}$, $\mathbb{R}$, $\mathbb{Q}$ or $\mathbb{Q} + i\mathbb{Q} \subset \mathbb{C}$, but the construction works for any field. The basic procedure follows several steps, which will be explained further below:

- Step 0. Compute $\Delta(\lambda) = \det[A(\lambda)]$ and decompose it into irreducible monic factors in $K[\lambda]$
$$\Delta(\lambda) = \text{const} \cdot p_1(\lambda)^{\kappa_1} \ldots p_l(\lambda)^{\kappa_l}. \tag{1.22}$$

- Step 1. Compute a local Smith form
$$A(\lambda)V_j(\lambda) = E_j(\lambda) \begin{pmatrix} p_j(\lambda)^{\kappa_{j1}} & & 0 \\ & \ddots & \\ 0 & & p_j(\lambda)^{\kappa_{jn}} \end{pmatrix} \tag{1.23}$$
for each factor $p_j(\lambda)$ of $\Delta(\lambda)$.

- Step 2. Find a linear combination $B_n(\lambda) = \sum_{j=1}^{l} g_j(\lambda) f_j(\lambda) V_j(\lambda)$ using Bézout's coefficients of $f_j(\lambda) = \prod_{k \neq j}^{l} p_j(\lambda)^{\kappa_{kn}}$ so that the columns of $B_n(\lambda)$ form an extended canonical system of root functions for $A(\lambda)$ with respect to each $p_j(\lambda)$.

- Step 3. Eliminate extraneous zeros from $\det\left[A(\lambda)B_n(\lambda)\right]$ by finding a unimodular matrix $V(\lambda)$ such that $B_1(\lambda) = V(\lambda)^{-1}B_n(\lambda)$ is lower triangular. We will show that $A(\lambda)V(\lambda)$ is then of the form $E(\lambda)D(\lambda)$ with $E(\lambda)$ unimodular and $D(\lambda)$ as in (1.3).

Note that the diagonal entries in the matrix polynomial $D(\lambda)$ are given by

$$d_i(\lambda) = \prod_{j=1}^{l} p_j(\lambda)^{\kappa_{ji}}, \quad i = 1, \ldots, n$$

once we know the local Smith forms. This allows us to order the columns once and for all in Step 2.

1.3.1 A Local Smith Form Algorithm (Step 1)

In this section, we show how to generalize the construction in [79] (for finding a canonical system of Jordan chains for an analytic matrix function $A(\lambda)$ over $\mathbb{C}$ at $\lambda_0 = 0$) to finding a local Smith form of a matrix polynomial $A(\lambda)$ with respect to an irreducible factor $p(\lambda)$ of $\Delta(\lambda) = \det[A(\lambda)]$. The new algorithm reduces to the "exact arithmetic" version of the previous algorithm when $p(\lambda) = \lambda$. In Appendix A, we present a variant of the algorithm that is easier to implement than the current approach, and is closer in spirit to the construction in [79], but is less efficient by a factor of $s = \deg p$.

Our goal is to find matrices $V(\lambda)$ and $E(\lambda)$ such that $p(\lambda)$ does not divide $\det[V(\lambda)]$ or $\det[E(\lambda)]$, and such that

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \qquad D(\lambda) = \operatorname{diag}[p(\lambda)^{\alpha_1}, \ldots, p(\lambda)^{\alpha_n}], \tag{1.24}$$

where $0 \leq \alpha_1 \leq \cdots \leq \alpha_n$. In our construction, $V(\lambda)$ will be unimodular, which reduces the work in Step 3 of the high level algorithm, the step in which extraneous zeros are removed from the determinant of the combined local Smith forms.

We start with $V(\lambda) = I_{n\times n}$ and perform a sequence of column operations on $V(\lambda)$ that preserve its determinant (up to a sign) and systematically increase the orders α_i in $D(\lambda)$ in (1.24) until $\det[E(\lambda)]$ no longer contains a factor of $p(\lambda)$. This can be considered a "breadth first" construction of a canonical system of Jordan chains, in contrast to the "depth first" procedure described in Definition 5.

The basic algorithm is presented in Figure 1.1. The idea of the algorithm is to run through the columns of V in turn and "accept" columns whenever the leading term of the residual $A(\lambda)x_i(\lambda)$ is linearly independent of its predecessors; otherwise we find a linear combination of previously accepted columns to cancel this leading term and cyclically rotate the column to the end for further processing. Note that for each k, we cycle through each unaccepted column exactly once: after rotating a column to the end, it will not become active again until k has increased by one.

At the start of the *while* loop, we have the invariants

Algorithm 1. (Local smith form, preliminary version)

$k = 0,\ \ i = 1,\ \ V = [x_1, \ldots, x_n] = I_{n\times n}$
while $i \le n$
  $r_{k-1} = n + 1 - i$ $r_{k-1} :=$ *dim. of space of J. chains of length* $\ge k$
  for $j = 1, \ldots, r_{k-1}$
    $y_i = \mathrm{rem}(\mathrm{quo}(Ax_i, p^k), p)$ *define* y_i *so* $Ax_i = p^k y_i + O(p^{k+1})$
    if the set $\{\dot{y}_1, \ldots, \dot{y}_i\}$ is linearly independent in M/pM over R/pR
      $\alpha_i = k, \quad i = i + 1$ *accept* x_i *and* y_i*, define* α_i
    else
      find $\dot{a}_1, \ldots, \dot{a}_{i-1} \in R/pR$ so that $\dot{y}_i - \sum_{m=1}^{i-1} \dot{a}_m \dot{y}_m = \dot{0}$
    $\star$ $x_i^{(new)} = x_i^{(old)} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m x_m$
      $\mathrm{tmp} = x_i, \quad x_m = x_{m+1},\ (m = i, \ldots, n-1), \quad x_n = \mathrm{tmp}$
    end if
  end for j
  $k = k + 1$
end while
$\beta = k - 1, \quad r_\beta = 0$ $\beta := \alpha_n =$ *maximal Jordan chain length*

Figure 1.1: Algorithm for computing a local Smith form.

(1) Ax_m is divisible by p^k, $(i \le m \le n)$.
(2) $Ax_m = p^{\alpha_m} y_m + O(p^{\alpha_m+1})$, $(1 \le m < i)$.
(3) if $i \ge 2$ then $\{\dot{y}_m\}_{m=1}^{i-1}$ is linearly independent in M/pM over R/pR.

The third property is guaranteed by the *if* statement in the algorithm in Figure 1.1, and the second property follows from the first due to the definition of α_i and y_i in the algorithm. The first property is obviously true when $k = 0$; it continues to hold each time k is incremented due to step $\star$, after which $Ax_i^{(new)}$ is divisible by p^{k+1}:

$$Ax_i^{(old)} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m Ax_m = p^k y_i + O(p^{k+1}) - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m \Big(p^{\alpha_m} y_m + O(p^{\alpha_m+1}) \Big)$$
$$= p^k \Big(y_i - \sum_{m=1}^{i-1} a_m y_m \Big) + O(p^{k+1}) = O(p^{k+1}).$$

This equation is independent of which polynomials $a_m \in R$ are chosen to represent $\dot{a}_m \in R/pR$, but different choices will lead to different (equally valid) Smith forms; in practice, we

choose the unique representatives such that $\deg a_m < s$, where

$$s = \deg p. \tag{1.25}$$

This choice of the a_m leads to two additional invariants at the start of the *while* loop, namely

(4) $\deg x_m \le \max(sk-1, 0)$, $\quad (i \le m \le n)$,
(5) $\deg x_m \le \max(s\alpha_m - 1, 0)$, $\quad (1 \le m < i)$,

which are easily proved inductively by noting that

$$\deg(p^{k-\alpha_m} a_m x_m) \le s(k-\alpha_m) + (s-1) + \deg(x_m) \le s(k+1) - 1. \tag{1.26}$$

The *while* loop eventually terminates, for at the end of each loop (after k has been incremented) we have produced a unimodular matrix $V(\lambda)$ such that

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \qquad D = \operatorname{diag}[p^{\alpha_1}, \ldots, p^{\alpha_{i-1}}, \underbrace{p^k, \ldots, p^k}_{r_{k-1} \text{ times}}]. \tag{1.27}$$

Hence, the algorithm must terminate before k exceeds the algebraic multiplicity μ of $p(\lambda)$ in $\Delta(\lambda)$:

$$k \le \left(\textstyle\sum_{m=1}^{i=1} \alpha_i\right) + (n+1-i)k \le \mu, \qquad \Delta(\lambda) = f(\lambda)p(\lambda)^\mu, \quad p \nmid f. \tag{1.28}$$

In fact, we can avoid the last iteration of the *while* loop if we change the test to

$$\textbf{while } \left[\left(\textstyle\sum_{m=1}^{i-1} \alpha_i\right) + (n+1-i)k\right] < \mu$$

and change the last line to

$$\beta = k, \qquad \alpha_m = k, \quad (i \le m \le n), \qquad r_{\beta-1} = n+1-i, \qquad r_\beta = 0.$$

We know the remaining columns of V will be accepted without having to compute the remaining y_i or check them for linear independence. When the algorithm terminates, we will have found a unimodular matrix $V(\lambda)$ satisfying (1.24) such that the columns of

$$\dot{E}(\lambda) = [\dot{y}_1(\lambda), \ldots, \dot{y}_n(\lambda)]$$

are linearly independent in M/pM over R/pR. By Lemma 8, $p(\lambda) \nmid \det[E(\lambda)]$, as required.

To implement the algorithm, we must find an efficient way to compute y_i, test for linear independence in M/pM, find the coefficients a_m to cancel the leading term of the residual, and update x_i. Motivated by the construction in [79], we interpret the loop over j in Algorithm 1 as a single nullspace calculation.

Let us define $R_l = \{a \in R \,:\, \deg a < l\}$ and $M_l = R_l^n$, both viewed as vector spaces over K. Then we have an isomorphism Λ of vector spaces over K

$$\begin{aligned} \Lambda : (M_s)^k &\to M_{sk}, \\ \Lambda(x^{(0)}; \ldots; x^{(k-1)}) &= x^{(0)} + px^{(1)} + \cdots + p^{k-1}x^{(k-1)}. \end{aligned} \tag{1.29}$$

At times it will be convenient to identify R_{ls} with R/p^lR and M_{ls} with M/p^lM to obtain ring and module structures for these spaces. We also expand

$$A = A^{(0)} + pA^{(1)} + \cdots + p^qA^{(q)}, \tag{1.30}$$

where $A^{(j)}$ is an $n \times n$ matrix with entries in R_s for $j = 1, \ldots, q$. By invariants (4) and (5) of the *while* loop, we may write $x_i = \Lambda(x_i^{(0)}; \ldots; x_i^{(\alpha)})$ with $\alpha = \max(k-1, 0)$. Since Ax_i is divisible by p^k in Algorithm 1, we have

$$y_i = \operatorname{rem}(\operatorname{quo}(Ax_i, p^k), p) = \sum_{j=0}^{k} \operatorname{rem}(A^{(k-j)}x_i^{(j)}, p) + \sum_{j=0}^{k-1} \operatorname{quo}(A^{(k-1-j)}x_i^{(j)}, p). \tag{1.31}$$

The matrix-vector multiplications $A^{(k-j)}x_i^{(j)}$ are done in the ring R (leading to vector polynomials of degree $\leq 2s-2$) before the quotient and remainder are taken. When $k = 0$, the second sum should be omitted, and when $k \geq 1$, the $j = k$ term in the first sum can be dropped since $x_i^{(k)} = 0$ in the algorithm.

In practice, we test all the active columns $\dot{y}_i, \ldots, \dot{y}_n \in M/pM$ for linear independence of their predecessors simultaneously using Algorithm 2 in Figure 1.2. If $k = 0$ we have

$$[y_1, \ldots, y_n] = A^{(0)}. \tag{1.32}$$

Otherwise $k \geq 1$ and we have computed the matrix X_{k-1} with columns $\left(x_m^{(0)}; \ldots; x_m^{(k-1)}\right)$ for $i \leq m \leq n$ such that $\Lambda(X_{k-1})$ (acting column by column) represents the last r_{k-1} columns of $V(\lambda)$ at the start of the *while* loop in Algorithm 1. Then by (1.31),

$$[y_i, \ldots, y_n] = \operatorname{rem}([A^{(k)}, \ldots, A^{(1)}]X_{k-1}, p) + \operatorname{quo}([A^{(k-1)}, \ldots, A^{(0)}]X_{k-1}, p). \tag{1.33}$$

As before, the matrix multiplications are done in the ring R before the quotient and remainder are computed to obtain the components of y_m, which belong to R_s. To test for linear independence, define the auxiliary matrices

$$\mathcal{A}_k = \begin{cases} A^{(0)}, & k = 0, \\ \left[\mathcal{A}_{k-1}\ ,\ [y_i, \ldots, y_n]\right], & 1 \leq k \leq n. \end{cases} \tag{1.34}$$

and compute the reduced row-echelon form of $\dot{\mathcal{A}}_k$ using Gauss-Jordan elimination over the field R/pR.

Algorithm 2. (Local smith form, final version)

$k = 0$
$\mathcal{A}_0 = A^{(0)}$
$X_0 = \mathbb{X}_0 = \text{null}(\dot{\mathcal{A}}_0)$
$r_0 = R_0 = \text{num\_cols}(\mathbb{X}_0)$ (number of columns)
$\widetilde{X}_{-1} = [e_{j_1}, \ldots, e_{j_{n-r_0}}]$, (columns j_i of rref($\dot{\mathcal{A}}_0$) start new rows)
while $R_k < \mu$ (μ = algebraic multiplicity of p)
$k = k + 1$
- $\mathcal{A}_k = \big(\mathcal{A}_{k-1}\ ,\ \text{rem}(\big[A^{(k)}, \ldots, A^{(1)}\big] X_{k-1}, p) + \text{quo}(\big[A^{(k-1)}, \ldots, A^{(0)}\big] X_{k-1}, p) \big)$
- $[Y_k; U_k] =$ new columns of null($\dot{\mathcal{A}}_k$) beyond those of null($\dot{\mathcal{A}}_{k-1}$)

$r_k = \text{num\_cols}(U_k)$, ($U_k$ is $R_{k-1} \times r_k$)
$R_k = R_{k-1} + r_k$
$X_k = [\text{rem}(\mathbb{X}_{k-1} U_k, p); Y_k] + \text{quo}(\iota(\mathbb{X}_{k-1} U_k), p)$ (X_k is $n(k+1) \times r_k$)
$\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k]$ ($\mathbb{X}_k$ is $n(k+1) \times R_k$)
$\widetilde{X}_{k-1} = X_{k-1}(:, [j_1, \ldots, j_{r_{k-1}-r_k}])$, (columns $n + R_{k-2} + j_i$ of rref($\dot{\mathcal{A}}_k$) start new rows)
end while
$\beta = k + 1$ (maximal Jordan chain length)
$\widetilde{X}_{\beta-1} = X_{\beta-1}$
$V(\lambda) = \big[\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{\beta-1})\big]$

Figure 1.2: Algorithm for computing a unimodular local Smith form.

The reduced row-echelon form of $\dot{\mathcal{A}}_k$ can be interpreted as a tableau telling which columns of $\dot{\mathcal{A}}_k$ are linearly independent of their predecessors (the accepted columns), and also giving the linear combination of previously accepted columns that will annihilate a linearly dependent column. On the first iteration (with $k = 0$), step $\star$ in Algorithm 1 will build up the matrix

$$X_0 = \text{null}(\dot{\mathcal{A}}_0), \tag{1.35}$$

where null($\cdot$) is the standard algorithm for computing a basis for the nullspace of a matrix from the reduced row-echelon form (followed by a truncation to replace the elements in R/pR of this nullspace matrix with their representatives in R_s). But rather than rotating these columns to the end as in Algorithm 1, we now *append* the corresponding y_i to the end of $\mathcal{A}_{k-1}$ to form $\mathcal{A}_k$ for $k \geq 1$. The "dead" columns left behind (not accepted, not active) serve only as placeholders, causing the resulting matrices $\mathcal{A}_k$ to be nested. We use rref($\cdot$) to denote the reduced row-echelon form of a matrix polynomial. The leading columns of

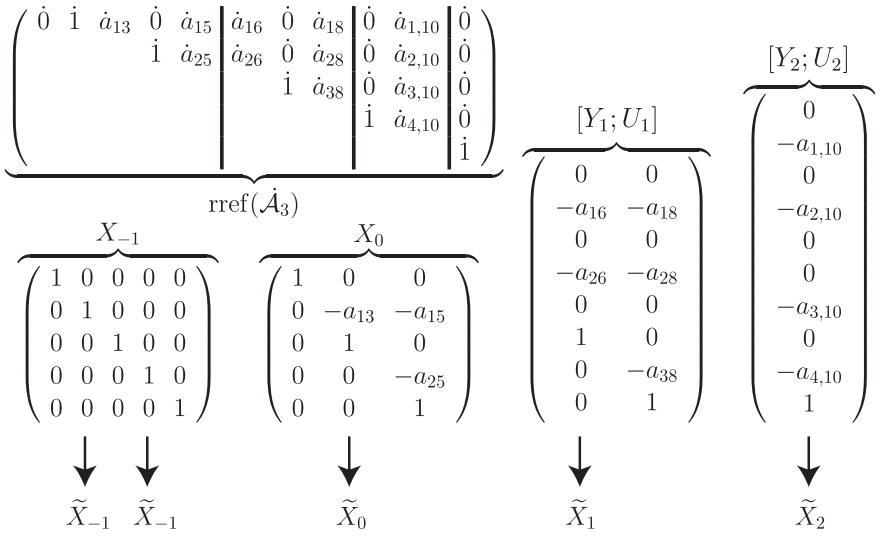

Figure 1.3: The reduced row-echelon form of $\dot{\mathcal{A}}_\beta$ contains all the information necessary to construct $V(\lambda) = [\Lambda(\widetilde{X}_{-1}), \dots, \Lambda(\widetilde{X}_{s-1})]$. An arrow from a column $[v;u]$ of $[Y_k;U_k]$ indicates that the vector $\big([\mathrm{rem}(\mathbb{X}_{k-1}u, p); v] + \mathrm{quo}(\iota(\mathbb{X}_{k-1}u), p)\big)$ should be added to $\widetilde{X}_k$.

$\mathrm{rref}(\dot{\mathcal{A}}_k)$ will then coincide with $\mathrm{rref}(\dot{\mathcal{A}}_{k-1})$, and the nullspace matrices will also be nested:

$$\begin{pmatrix} X_0 & Y_1 & \cdots & Y_{k-1} & Y_k \\ 0 & [U_1;0] & \cdots & [U_{k-1};0] & U_k \end{pmatrix} := \mathrm{null}(\dot{\mathcal{A}}_k). \tag{1.36}$$

Note that $\mathcal{A}_k$ is $n \times (n + R_{k-1})$, where

$$R_{-1} = 0, \qquad R_k = r_0 + \cdots + r_k = \dim\ker \dot{\mathcal{A}}_k, \qquad (k \geq 0). \tag{1.37}$$

We also see that X_0 is $n \times r_0$, Y_k is $n \times r_k$, and U_k is $r_{k-1} \times r_k$. Since the dimension of the kernel cannot increase by more than the number of columns added,

$$r_k \leq r_{k-1}, \qquad (k \geq 0). \tag{1.38}$$

If column i of $\dot{\mathcal{A}}_k$ is linearly dependent on its predecessors, the coefficients a_m used in step $\star$ of Algorithm 1 are precisely the (truncations of the) coefficients that appear in column i of $\mathrm{rref}(\dot{\mathcal{A}}_k)$. The corresponding null vector (i.e. column of $[Y_k;U_k]$) contains the negatives of these coefficients in the rows corresponding to the previously accepted columns of $\dot{\mathcal{A}}_k$, followed by a 1 in row i; see Figure 1.3. Thus, in step $\star$, if $k \geq 1$ and we write $x_m = \Lambda\big(x_m^{(0)}; \dots; x_m^{(\alpha)}\big)$ with $\alpha = \max(\alpha_m - 1, 0)$, the update

$$x_i^{(new)} = x_i^{(old)} - \sum_{m=1}^{i-1} p^{k-\alpha_m} a_m x_m, \qquad \operatorname{rem}(a_m x_m, p^{\alpha_m+1}) = \Lambda\big(z^{(0)}; \ldots; z^{(\alpha_m)}\big),$$

$$z^{(j)} = \begin{cases} \operatorname{rem}(a_m x_m^{(0)}, p), & j = 0, \\ \operatorname{rem}(a_m x_m^{(j)}, p) + \operatorname{quo}(a_m x_m^{(j-1)}, p), & 1 \le j < \alpha_m, \\ \operatorname{quo}(a_m x_m^{(j-1)}, p), & j = \alpha_m \text{ and } \alpha_m > 0, \end{cases}$$

is equivalent to

$$\begin{aligned} X_k = \operatorname{rem}\Bigg(\Big[\iota^k(X_{-1})\,,\, \iota^{k-1}\rho(X_0)\,,\, \ldots,\, \iota^0\rho(X_{k-1})\Big] \begin{pmatrix} Y_k \\ U_k \end{pmatrix}, p \Bigg) \\ + \operatorname{quo}\Big(\big[\iota^k(X_0)\,,\, \ldots\,,\, \iota^1(X_{k-1})\big] U_k, p \Big), \end{aligned} \tag{1.39}$$

where $\iota, \rho : (M_s)^l \to (M_s)^{l+1}$ act column by column, padding them with zeros:

$$\iota(x) = (0; x), \qquad \rho(x) = (x; 0), \qquad x \in (M_s)^l, \quad 0 \in M_s. \tag{1.40}$$

Here $\Lambda\iota\Lambda^{-1}$ is multiplication by p, which embeds $M_{ls} \cong M/p^l M$ in $M_{(l+1)s} \cong M/p^{l+1}M$ as a module over R, while ρ is an embedding of vector spaces over K (but not an R-module morphism). If we define the matrices $\mathbb{X}_0 = X_0$ and

$$\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k] = \left[\begin{pmatrix} 0_{nk \times r_0} \\ X_0 \end{pmatrix}, \begin{pmatrix} 0_{n(k-1) \times r_1} \\ X_1 \end{pmatrix}, \ldots, \begin{pmatrix} X_k \end{pmatrix} \right], \quad (k \ge 1), \tag{1.41}$$

then (1.39) simply becomes

$$X_k = [\operatorname{rem}(\mathbb{X}_{k-1} U_k, p); Y_k] + \operatorname{quo}(\iota(\mathbb{X}_{k-1} U_k), p). \tag{1.42}$$

As in (1.33) above, the matrix multiplications are done in the ring R before the quotient and remainder are computed to obtain X_k. Finally, we line up the columns of X_{k-1} with the last r_{k-1} columns of $\dot{\mathcal{A}}_k$ and extract (i.e. accept) columns of X_{k-1} that correspond to new, linearly independent columns of $\dot{\mathcal{A}}_k$. We denote the matrix of extracted columns by $\widetilde{X}_{k-1}$. At the completion of the algorithm, the unimodular matrix $V(\lambda)$ that puts $A(\lambda)$ in local Smith form is given by

$$V(\lambda) = \big[\Lambda(\widetilde{X}_{-1}), \ldots, \Lambda(\widetilde{X}_{\beta-1})\big]. \tag{1.43}$$

The final algorithm is presented in Figure 1.2. In the steps marked $\bullet$, we can avoid re-computing the reduced row-echelon form of the first $n + R_{k-2}$ columns of $\dot{\mathcal{A}}_k$ by storing the sequence of Gauss-Jordan transformations [35] that reduced $\dot{\mathcal{A}}_{k-1}$ to row-echelon form. To compute $[Y_k; U_k]$, we need only apply these transformations to the new columns of $\dot{\mathcal{A}}_k$ and then proceed with the row-reduction algorithm on these final columns. Also, if A_0 is

large and sparse, rather than reducing to row-echelon form, one could find kernels using an LU factorization designed to handle singular matrices. This would allow the use of graph theory (clique analysis) to choose pivots in the Gaussian elimination procedure to minimize fill-in. We also note that if $\Delta(\lambda)$ contains only one irreducible factor, the local Smith form is a (global) Smith form of $A(\lambda)$; steps 2 and 3 can be skipped in that case.

Example 5. *Let $A(\lambda)$ be the matrix polynomial defined in Example 1. Expanding about $p_1(\lambda) = \lambda - 1$, we have*

$$A(\lambda) = \begin{pmatrix} 12 & 12 \\ 0 & 0 \end{pmatrix} + p_1 \begin{pmatrix} 20 & 44 \\ 6 & 6 \end{pmatrix} + O(p_1^2), \quad R_{-1} = 0,$$

$$\mathcal{A}_0 = \begin{pmatrix} 12 & 12 \\ 0 & 0 \end{pmatrix}, \quad \mathbb{X}_0 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad R_0 = 1, \quad \tilde{X}_{-1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

$$\mathcal{A}_1 = \left(\begin{array}{cc|c} 12 & 12 & 24 \\ 0 & 0 & 0 \end{array}\right), \quad \begin{pmatrix} Y_1 \\ U_1 \end{pmatrix} = \left(\begin{array}{c} -2 \\ 0 \\ \hline 1 \end{array}\right), \quad \mathbb{X}_1 = \begin{pmatrix} 0 & -1 \\ 0 & 1 \\ -1 & -2 \\ 1 & 0 \end{pmatrix}, \quad R_1 = 2,$$

$$\beta = 2, \quad \tilde{X}_1 = \begin{pmatrix} -1 \\ 1 \\ -2 \\ 0 \end{pmatrix}, \quad V_1 = \begin{pmatrix} 1 & 1-2\lambda \\ 0 & 1 \end{pmatrix}, D_1 = \begin{pmatrix} 1 & 0 \\ 0 & (\lambda-1)^2 \end{pmatrix},$$

$$E_1 = AV_1D_1^{-1} = \begin{pmatrix} 2\lambda^4 + 6\lambda^2 + 4 & \lambda^6 + 2\lambda^5 + 7\lambda^4 + 7\lambda^3 + 15\lambda^2 + 6\lambda + 10 \\ 2\lambda^3 - 2\lambda^2 + 4\lambda - 4 & \lambda^5 + \lambda^4 + 4\lambda^3 - \lambda^2 + 4\lambda - 6 \end{pmatrix}.$$

Expanding about $p_2(\lambda) = \lambda^2 + 2$, we obtain

$$A(\lambda) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + O(p_2), \quad R_{-1} = 0,$$

$$\mathcal{A}_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbb{X}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R_0 = 2, \quad \tilde{X}_{-1} = \emptyset,$$

$$\beta = 1, \quad \tilde{X}_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad V_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, D_2 = \begin{pmatrix} \lambda^2+2 & 0 \\ 0 & \lambda^2+2 \end{pmatrix},$$

$$E_2 = AV_2D_2^{-1} = \begin{pmatrix} 2\lambda^2 + 2 & \lambda^6 + 2\lambda^4 - \lambda^3 + 2\lambda^2 - 3\lambda + 3 \\ 2\lambda - 2 & \lambda^5 - \lambda^4 + \lambda^3 - 2\lambda^2 + 2\lambda - 1 \end{pmatrix}.$$

1.3.2 Algorithms for the Extended GCD Problem

The extended Euclidean algorithm is widely applied to solve the extended GCD problem for two polynomials in $K[\lambda]$, e.g. gcdex in Maple. The algorithm requires $O(P(d)\log d)$ arithmetic operations in K to solve the problem, where f_1 and f_2 both have degrees no greater than d and $P(d)$ is the number of field operations in K required to multiply two polynomials of degree $d-1$ in $K[\lambda]$. Using standard polynomial multiplication, we have $P(d) = d^2$, while a fast algorithm [72] uses $P(d) = d(\log d)(\log\log d)$.

The extended GCD problem (1.21) for more than two polynomials can be solved by applying the extended Euclidean algorithm to all the polynomials simultaneously; see below. A variant of this approach is to focus on the two lowest degree polynomials until one is reduced to zero; we then repeat until all but one is zero. In practice, we use the function 'MatrixPolynomialAlgebra[HermiteForm]' in Maple.

Suppose we have n polynomials $f_1, \ldots, f_n$ in $K[\lambda]$. We find $f_i \neq 0$ with the lowest degree. Each f_j with $j \neq i$ can then be written as

$$f_j - q_j f_i = r_j,$$

where $q_j := \mathrm{quo}(f_j, f_i)$, $r_j := \mathrm{rem}(f_j, f_i)$. Hence, we have

$$\begin{pmatrix} 1 & & -q_1 & & \\ & \ddots & \vdots & & \\ & & 1 & & \\ & & \vdots & \ddots & \\ & & -q_n & & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_i \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} r_1 \\ \vdots \\ f_i \\ \vdots \\ r_n \end{pmatrix}.$$

Denote the matrix by Q_1. We repeat this procedure on $(r_1; \ldots; f_i; \ldots; r_n)$ until there is only one nonzero entry in the vector, and we obtain

$$Q_k Q_{k-1} \ldots Q_1 \begin{pmatrix} f_1 \\ \\ \vdots \\ \\ f_n \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ r \\ \vdots \\ 0 \end{pmatrix}. \tag{1.44}$$

The row of $Q := Q_k Q_{k-1} \ldots Q_1$ with the same index as r divided by the leading coefficient of r gives a solution of the extended GCD problem.

Example 6. *Define $f_1 = \lambda^2 + 2$ and $f_2 = (\lambda - 1)^2$ as in Example 4. Following the procedure described above, we obtain*

$$\begin{pmatrix} 1 & 0 \\ \frac{8}{9}\lambda + \frac{4}{9} & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2}\lambda - \frac{1}{4} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{8}{9}\lambda + \frac{4}{9} & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2}\lambda - \frac{1}{4} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda^2 + 2 \\ -2\lambda - 1 \end{pmatrix}$$
$$= \begin{pmatrix} 1 & 0 \\ \frac{8}{9}\lambda + \frac{4}{9} & 1 \end{pmatrix} \begin{pmatrix} \frac{9}{4} \\ -2\lambda - 1 \end{pmatrix} = \begin{pmatrix} \frac{9}{4} \\ 0 \end{pmatrix}.$$

With

$$Q = \begin{pmatrix} 1 & 0 \\ \frac{8}{9}\lambda + \frac{4}{9} & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2}\lambda - \frac{1}{4} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}\lambda + \frac{5}{4} & \frac{1}{2}\lambda - \frac{1}{4} \\ -\frac{4}{9}\lambda^2 + \frac{8}{9}\lambda - \frac{4}{9} & \frac{4}{9}\lambda^2 + \frac{8}{9} \end{pmatrix},$$

we have $g_1 = -\frac{2}{9}\lambda + \frac{5}{9}$ and $g_2 = \frac{2}{9}\lambda - \frac{1}{9}$ as in Example 4.

1.3.3 From Local to Global (Step 2)

Now that we have a local Smith form (1.23) for every irreducible factor $p_j(\lambda)$ of $\Delta(\lambda)$, we can apply the algorithm in Section 1.3.2 to obtain a family of polynomials $\{g_j(\lambda)\}_{j=1}^{l}$ with $\deg(g_j(\lambda)) < s_j\kappa_{jn}$, where $s_j = \deg(p_j)$, such that

$$\sum_{j=1}^{l}\left[g_j(\lambda)\prod_{k=1,k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right] = 1, \tag{1.45}$$

where $p_j(\lambda)^{\kappa_{jn}}$ is the last entry in the diagonal matrix of the local Smith form at $p_j(\lambda)$. The integers κ_{jn} are positive. We define a matrix polynomial $B_n(\lambda)$ via

$$B_n(\lambda) = \sum_{j=1}^{l}\left[g_j(\lambda)V_j(\lambda)\prod_{k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right]. \tag{1.46}$$

The main result of this section is stated as follows.

Proposition 11. *The matrix polynomial $B_n(\lambda)$ in (1.46) has two key properties:*

1. *Let $b_{ni}(\lambda)$ be the ith column of $B_n(\lambda)$. Then $A(\lambda)b_{ni}(\lambda)$ is divisible by $d_i(\lambda)$, where $d_i(\lambda) = \prod_{j=1}^{l}p_j(\lambda)^{\kappa_{ji}}$ is the ith diagonal entry in $D(\lambda)$ of the Smith form.*
2. *$\det[B_n(\lambda)]$ is not divisible by $p_j(\lambda)$ for $j = 1,\ldots,l$.*

Proof. 1. Let $v_{ji}(\lambda)$ be the ith column of $V_j(\lambda)$. Then $A(\lambda)v_{ji}(\lambda)$ is divisible by $p_j(\lambda)^{\kappa_{ji}}$ and

$$b_{ni}(\lambda) = \sum_{j=1}^{l}\left[\prod_{k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right]g_j(\lambda)v_{ji}(\lambda).$$

Since $\kappa_{jn} \geq \kappa_{ji}$ for $1 \leq i \leq n$ and $1 \leq j \leq l$,

$$A(\lambda)b_{ni}(\lambda) = \sum_{j=1}^{l}\left[(A(\lambda)v_{ji}(\lambda))\prod_{k\neq j}^{l}p_k(\lambda)^{\kappa_{kn}}\right]g_j(\lambda)$$

is divisible by $d_i(\lambda)$.

2. The local Smith form construction ensures that $p_j(\lambda) \nmid \det[V_j(\lambda)]$ for each $1 \leq j \leq l$. Equation (1.45) modulo $p_j(\lambda)$ shows that $p_j(\lambda) \nmid g_j(\lambda)$. By definition,

$$\begin{aligned}\det[B_n(\lambda)] &= \det\left(\left[b_{n1}(\lambda)\,,\,\ldots\,,\,b_{nn}(\lambda)\right]\right) = \det\left(\left[b_{ni}(\lambda)\right]_{i=1}^{n}\right)\\ &= \det\left(\left[\sum_{j'=1}^{l}\left(\prod_{k\neq j'}^{l}p_k(\lambda)^{\kappa_{kn}}\right)g_{j'}(\lambda)v_{j'i}(\lambda)\right]_{i=1}^{n}\right).\end{aligned}$$

Each term in the sum is divisible by $p_j(\lambda)$ except $j' = j$. Thus, by multi-linearity,

$$\operatorname{rem}(\det[B_n(\lambda)], p_j(\lambda)) = \operatorname{rem}\left(\left[\prod_{k \neq j}^{l} p_k(\lambda)^{\kappa_{kn}} \right]^n \left[g_j(\lambda) \right]^n \det\left[V_j(\lambda) \right], p_j(\lambda) \right) \neq 0,$$

as claimed. □

Remark 12. It is possible for $\det[B_n(\lambda)]$ to be non-constant; however, its irreducible factors will be distinct from $p_1(\lambda), \ldots, p_l(\lambda)$.

Remark 13. Rather than building $B_n(\lambda)$ as a linear combination (1.46), we may form $B_n(\lambda)$ with columns

$$b_{ni}(\lambda) = \sum_{j=1}^{l} \left[\prod_{k \neq j}^{l} p_k(\lambda)^{\max(\kappa_{ki}, 1)} \right] g_{ij}(\lambda) v_{ji}(\lambda), \qquad (1 \leq i \leq n), \tag{1.47}$$

where $\{g_{ij}\}_{j=1}^{l}$ solves the extended GCD problem

$$\sum_{j=1}^{l} \left[g_{ij}(\lambda) \prod_{k \neq j}^{l} p_k(\lambda)^{\max(\kappa_{ki}, 1)} \right] = 1.$$

The two properties proved above also hold for this definition of $B_n(\lambda)$. The reason for increasing the power of $p_k(\lambda)$ to 1 when $\kappa_{ki} = 0$ is to ensure that $p_j(\lambda) \nmid g_{ij}(\lambda)$ is satisfied for any i, j. This modification can significantly reduce the polynomial degree and coefficient size of $B_n(\lambda)$ when there is a wide range of Jordan chain lengths.

Example 7. *With $A(\lambda)$ defined as in Example 1, the matrix polynomial $B_n(\lambda)$ is given by*

$$\begin{aligned} B_n(\lambda) &= g_1(\lambda) V_1(\lambda) p_2(\lambda) + g_2(\lambda) V_2(\lambda) p_1(\lambda)^2 \\ &= \left(-\frac{2}{9}\lambda + \frac{5}{9} \right) \left(\lambda^2 + 2 \right) \begin{pmatrix} 1 & 1 - 2\lambda \\ 0 & 1 \end{pmatrix} + \left(\frac{2}{9}\lambda - \frac{1}{9} \right) (\lambda - 1)^2 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & \frac{4}{9}\lambda^4 - \frac{4}{3}\lambda^3 + \frac{13}{9}\lambda^2 - \frac{8}{3}\lambda + \frac{10}{9} \\ 0 & 1 \end{pmatrix}, \end{aligned} \tag{1.48}$$

where $n = 2$ in this case. Alternatively, we can construct $B_2(\lambda)$ with the modified formulation (1.47) *in Remark 13. We find that $(g_{11}, g_{12}) = (\frac{1}{3}, -\frac{1}{3}\lambda - \frac{1}{3})$ solves the extended GCD problem $g_{11} p_2 + g_{12} p_1 = 1$ with $\deg g_{11} < 1$ and $\deg g_{12} < 2$, and $(g_{21}, g_{22}) = (g_1, g_2)$ solves the extended GCD problem $g_{21} p_2 + g_{22} p_1^2 = 1$. We obtain*

$$B_2(\lambda) = \begin{pmatrix} b_{21} & b_{22} \end{pmatrix}, \tag{1.49}$$

where

$$\begin{aligned}
b_{21} &= g_{11}(\lambda)v_{11}(\lambda)p_2(\lambda) + g_{12}(\lambda)v_{21}(\lambda)p_1(\lambda) \\
&= \frac{1}{3}\left(\lambda^2+2\right)\begin{pmatrix}1\\0\end{pmatrix} + \left(-\frac{1}{3}\lambda - \frac{1}{3}\right)(\lambda-1)\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix}, \\
b_{22} &= g_{21}(\lambda)v_{12}(\lambda)p_2(\lambda) + g_{22}(\lambda)v_{22}(\lambda)p_1(\lambda)^2 \\
&= \left(-\frac{2}{9}\lambda + \frac{5}{9}\right)\left(\lambda^2+2\right)\begin{pmatrix}1-2\lambda\\1\end{pmatrix} + \left(\frac{2}{9}\lambda - \frac{1}{9}\right)(\lambda-1)^2\begin{pmatrix}0\\1\end{pmatrix} \\
&= \begin{pmatrix}\frac{4}{9}\lambda^4 - \frac{4}{3}\lambda^3 + \frac{13}{9}\lambda^2 - \frac{8}{3}\lambda + \frac{10}{9}\\ 1\end{pmatrix}.
\end{aligned}$$

Note that this matrix polynomial in (1.49) *coincides with that in* (1.48)*. In general this is not true. It happened here because the first column of* $V_1(\lambda)$ *and that of* $V_2(\lambda)$ *are both equal to* $(1,0)^T$*.*

1.3.4 Construction of Unimodular Matrix Polynomials (Step 3)

Given a vector polynomial $[f_1(\lambda);\ldots;f_n(\lambda)] \in K[\lambda]^n$, we can use the extended GCD algorithm to find a unimodular matrix $Q(\lambda)$ such that $Q(\lambda)f(\lambda) = [0;\ldots;0;r(\lambda)]$, where $r = \gcd(f_1,\ldots,f_n)$. Explicitly, we apply one additional transformation Q_{k+1} to (1.44) to swap row i with row n and scale this row to make r monic. We then define $Q = Q_{k+1}Q_k\cdots Q_1$, which is unimodular. We apply this procedure to the last column of $B_n(\lambda)$ and define $V_n(\lambda) = Q(\lambda)^{-1}$. The resulting matrix

$$B_{n-1}(\lambda) := V_n(\lambda)^{-1}B_n(\lambda)$$

is zero above the main diagonal in column n. We then apply this procedure to the first $n-1$ components of column $n-1$ of $B_{n-1}(\lambda)$ to get a new $Q(\lambda)$, and define

$$V_{n-1}(\lambda) = \left(\begin{array}{ccc:c} & & & 0\\ & Q(\lambda)^{-1} & & \vdots\\ & & & 0\\ \hdashline 0 & \cdots & 0 & 1\end{array}\right). \tag{1.50}$$

It follows that $B_{n-2}(\lambda) := V_{n-1}(\lambda)^{-1}B_{n-1}(\lambda)$ is zero above the main diagonal in columns $n-1$ and n. Continuing in this fashion, we obtain unimodular matrices $V_n(\lambda)$, …, $V_2(\lambda)$ such that

$$A(\lambda)B_n(\lambda) = A(\lambda)\underbrace{V_n(\lambda)\cdots V_2(\lambda)}_{V(\lambda)}V_2(\lambda)^{-1}\cdots\underbrace{V_n(\lambda)^{-1}B_n(\lambda)}_{B_{n-1}(\lambda)} = A(\lambda)V(\lambda)B_1(\lambda),$$

where $V(\lambda)$ is unimodular, $B_1(\lambda)$ is lower triangular, and

$$\det[B_1(\lambda)] = \text{const} \cdot \det[B_n(\lambda)]. \tag{1.51}$$

The matrix $V(\lambda)$ puts $A(\lambda)$ in Smith form:

Proposition 14. *There is a unimodular matrix polynomial $E(\lambda)$ such that*

$$A(\lambda)V(\lambda) = E(\lambda)D(\lambda), \tag{1.52}$$

where $D(\lambda)$ is of the form (1.3).

Proof. Let r_{mi} denote the entry of $B_1(\lambda)$ in the mth row and ith column. Define $y_i(\lambda)$ and $z_i(\lambda)$ to be the ith columns of $A(\lambda)V(\lambda)$ and $A(\lambda)V(\lambda)B_1(\lambda)$, respectively, so that

$$z_i(\lambda) = y_i(\lambda)r_{ii}(\lambda) + \sum_{m=i+1}^{n} y_m(\lambda)r_{mi}(\lambda), \qquad (1 \le i \le n). \tag{1.53}$$

By Proposition 11, $z_i(\lambda)$ is divisible by $d_i(\lambda)$ for $1 \le i \le n$ and $p_j(\lambda) \nmid \det[B_1(\lambda)]$ for $1 \le j \le l$. Since $B_1(\lambda)$ is lower triangular, we have $\det[B_1(\lambda)] = \prod_{i=1}^{n} r_{ii}$. It follows that the diagonal entries $r_{ii}(\lambda)$ of $B_1(\lambda)$ are relatively prime to each of the $d_k(\lambda)$ $(k = 1, \ldots, n)$. As $d_n(\lambda)$ divides $y_n(\lambda)r_{nn}(\lambda)$ and is relatively prime to $r_{nn}(\lambda)$, it divides $y_n(\lambda)$ alone. Now suppose $1 \le i < n$ and we have shown that $d_m(\lambda)$ divides $y_m(\lambda)$ for $i < m \le n$. Then since $d_i(\lambda)$ divides $d_m(\lambda)$ for $m > i$ and $r_{ii}(\lambda)$ is relatively prime to $d_i(\lambda)$, we conclude from (1.53) that $d_i(\lambda)$ divides $y_i(\lambda)$. By induction, $d_i(\lambda)$ divides $y_i(\lambda)$ for $1 \le i \le n$. Thus, there is a matrix polynomial $E(\lambda)$ such that (1.52) holds. Because $V(\lambda)$ is unimodular and $\det[A(\lambda)] = \text{const} \cdot \det[D(\lambda)]$, it follows that $E(\lambda)$ is also unimodular, as claimed. □

Remark 15. $V(\lambda)$ constructed as described above puts $A(\lambda)$ in a global Smith form whether we build $B_n(\lambda)$ as a linear combination (1.46) or as in Remark 13.

Remark 16. We can stop the loop before reaching (V_2, B_2) by adding a test

$$\textbf{while } d_k \neq 1$$

and defining $V(\lambda) = V_n(\lambda) \cdots V_{k+1}(\lambda)$ after the *while* loop terminates. In the remaining of this section k refers to this index. It is the largest integer for which

$$d_1(\lambda) = \cdots = d_k(\lambda) = 1.$$

We know k from the local Smith form calculations. As a matter of fact, once we obtain $V_n(\lambda), \ldots, V_{k+1}(\lambda)$, the ith column of the matrix polynomial $V = V_n(\lambda) \cdots V_{k+1}(\lambda)$ is accepted for $i = k+1, \ldots, n$ and will not change in the remaining iterations of the *for* loop. The last $n-k$ columns of $V_n \cdots V_{k+1}$ are the same as those of $V_n \cdots V_2$, and therefore contain identical Jordan chains.

We find that a slight modification can significantly reduce the degree of the polynomials and the size of the coefficients in the computation. In this variant, rather than applying the extended GCD algorithm on $b_{nn}(\lambda)$ to find a unimodular matrix polynomial $Q(\lambda)$ so that $Qb_{nn}(\lambda)$ has the form $[0;\ldots;0;r(\lambda)]$, we compute $Q(\lambda)$ that puts $\mathrm{rem}(b_{nn}(\lambda),d_n(\lambda))$ into the desired form. That is, we replace the last column of $B_n(\lambda)$ with $\mathrm{rem}(b_{nn}(\lambda),d_n(\lambda))$ and then find $Q(\lambda)$ that puts $B_n(\lambda)$ in the desired form.To distinguish, we denote this new definition of $V_n(\lambda)=Q(\lambda)^{-1}$ by $\tilde{V}_n(\lambda)$ and the resulting $B_{n-1}(\lambda)$ by $\tilde{B}_{n-1}(\lambda)$. Continuing in this manner, we find unimodular matrix polynomials $\tilde{V}_n(\lambda),\ldots,\tilde{V}_{k+1}(\lambda)$ by applying the procedure on $\mathrm{rem}(\tilde{b}_{ii}(\lambda),d_i(\lambda))$ for $i=n,\ldots,k+1$, where $\tilde{b}_{ii}(\lambda)$ has the first i components of column i of $\tilde{B}_i(\lambda)$. We also define $\bar{B}_i=\tilde{V}_{i+1}(\lambda)^{-1}\cdots\tilde{V}_n(\lambda)^{-1}B_n(\lambda)$ for $k\le i\le n-1$. Note that $\bar{B}_i(\lambda)\neq\tilde{B}_i(\lambda)$. It remains to show that this definition of $\tilde{V}(\lambda)=\tilde{V}_n(\lambda)\ldots\tilde{V}_{k+1}(\lambda)$ such that

$$A(\lambda)B_n(\lambda)=A(\lambda)\underbrace{\tilde{V}_n(\lambda)\cdots\tilde{V}_{k+1}(\lambda)}_{\tilde{V}(\lambda)}\tilde{V}_{k+1}(\lambda)^{-1}\cdots\underbrace{\tilde{V}_n(\lambda)^{-1}B_n(\lambda)}_{\bar{B}_{n-1}(\lambda)}=A(\lambda)\tilde{V}(\lambda)\bar{B}_k(\lambda)$$

also puts $A(\lambda)$ in Smith form:

Proposition 17. *There is a unimodular matrix polynomial $\tilde{E}(\lambda)$ such that*

$$A(\lambda)\tilde{V}(\lambda)=\tilde{E}(\lambda)D(\lambda), \tag{1.54}$$

where $D(\lambda)$ is of the form (1.3).

Proof. Define $\tilde{q}_i(\lambda)=\left[\mathrm{quo}(\tilde{b}_{ii}(\lambda),d_i(\lambda));0\right]\in M=R^n$ for $i=k+1,\ldots,n$, where $0\in R^{n-i}$, $\tilde{b}_{ii}(\lambda)$ has the first i components of column i of $\tilde{B}_i(\lambda)$, and $\tilde{B}_n(\lambda):=B_n(\lambda)$. Then we have

$$\begin{aligned}\tilde{B}_{n-1}(\lambda)&=\tilde{V}_n(\lambda)^{-1}\left(B_n(\lambda)-\left[\begin{array}{c|c}0_{n\times(n-1)} & d_n(\lambda)\tilde{q}_n(\lambda)\end{array}\right]\right)\\&=\bar{B}_{n-1}(\lambda)-\left[\begin{array}{c|c}0_{n\times(n-1)} & d_n(\lambda)\tilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\end{array}\right].\end{aligned}$$

The first $n-1$ columns of $\tilde{B}_n(\lambda)$ are the same as those of $\bar{B}_n(\lambda)$. Continuing to find $\tilde{B}_{n-2}(\lambda)$, we have

$$\begin{aligned}\tilde{B}_{n-2}(\lambda)&=\tilde{V}_{n-1}(\lambda)^{-1}\left(\tilde{B}_{n-1}(\lambda)-\left[\begin{array}{c|c|c}0_{n\times(n-2)} & d_{n-1}(\lambda)\tilde{q}_{n-1}(\lambda) & 0_{n\times1}\end{array}\right]\right)\\&=\tilde{V}_{n-1}(\lambda)^{-1}\left(\bar{B}_{n-1}(\lambda)-\left[\begin{array}{c|c|c}0_{n\times(n-2)} & d_{n-1}(\lambda)\tilde{q}_{n-1}(\lambda) & d_n(\lambda)\tilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\end{array}\right]\right)\\&=\bar{B}_{n-2}(\lambda)-\left[\begin{array}{c|c|c}0_{n\times(n-2)} & d_{n-1}(\lambda)\tilde{V}_{n-1}(\lambda)^{-1}\tilde{q}_{n-1}(\lambda) & d_n(\lambda)\tilde{V}_{n-1}(\lambda)^{-1}\tilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\end{array}\right].\end{aligned}$$

It follows by induction that

$$\begin{aligned}\tilde{B}_k(\lambda)&=\tilde{V}_{k+1}(\lambda)^{-1}\left(\tilde{B}_{n+1}(\lambda)-\left[\begin{array}{c|c|c}0_{n\times k} & d_{k+1}(\lambda)\tilde{q}_{k+1}(\lambda) & 0_{n\times(n-k-1)}\end{array}\right]\right)\\&=\bar{B}_k(\lambda)-\left[\begin{array}{c|c|c|c}0_{n\times k} & d_{k+1}(\lambda)\tilde{V}_{k+1}(\lambda)^{-1}\tilde{q}_{k+1}(\lambda) & \cdots & d_n(\lambda)\tilde{V}_{k+1}(\lambda)^{-1}\cdots\tilde{V}_n(\lambda)^{-1}\tilde{q}_n(\lambda)\end{array}\right].\end{aligned}$$

$\tilde{B}_k(\lambda)$ is zero above the main diagonal in columns $k+1$ to n. Define

$$u_i(\lambda) := \tilde{V}_{k+1}(\lambda)^{-1} \cdots \tilde{V}_i(\lambda)^{-1} \tilde{q}_i(\lambda).$$

Then the ith column of the difference $\bar{B}_k(\lambda) - \tilde{B}_k(\lambda)$ is given by $d_i(\lambda)u_i(\lambda)$ for $k+1 \le i \le n$.

Let $\tilde{r}_{mi}(\lambda)$ denote the entry of $\tilde{B}_k(\lambda)$ in the mth row and ith column. Define $\tilde{y}_i(\lambda)$ and $z_i(\lambda)$ to be the ith columns of $A(\lambda)\tilde{V}(\lambda)$ and $A(\lambda)\tilde{V}(\lambda)\bar{B}_k(\lambda)$, respectively, so that

$$z_i(\lambda) = \tilde{y}_i(\lambda)\tilde{r}_{ii}(\lambda) + \sum_{m=i+1}^{n} \tilde{y}_m(\lambda)\tilde{r}_{mi}(\lambda) + d_i(\lambda)A(\lambda)\tilde{V}(\lambda)u_i(\lambda).$$

By Proposition 11, $z_i(\lambda)$ is divisible by $d_i(\lambda)$ and $p_j(\lambda) \nmid \det[B_n(\lambda)] = \text{const} \cdot \det[\bar{B}_i(\lambda)]$ for $1 \le j \le l$, where const $\neq 0$. Since d_i is divisible by d_m for $i \le m \le n$, $\det[\bar{B}_i(\lambda)] - \det[\tilde{B}_i(\lambda)]$ is divisible by $d_{i+1}(\lambda)$ due to the structure of $\bar{B}_i(\lambda) - \tilde{B}_i(\lambda)$ and multi-linearity of determinants. We also know that $\det[\tilde{B}_i(\lambda)]$ is divisible by $\tilde{r}_{ii}(\lambda)$. Proof by contradiction shows that $\tilde{r}_{ii}(\lambda)$ is relatively prime to $d_i(\lambda)$ $(i = k+1, \ldots, n)$. Then we argue by induction as in the proof of Proposition 14 to conclude that $d_i(\lambda)$ divides $y_i(\lambda)$ for $k+1 \le i \le n$. It holds trivially for $1 \le i \le k$ as $d_1 = \cdots = d_k = 1$. Thus, there is a matrix polynomial $\tilde{E}(\lambda)$ such that (1.54) holds. Because $\tilde{V}(\lambda)$ is unimodular and $\det[A(\lambda)] = \text{const} \cdot \det[D(\lambda)]$, it follows that $\tilde{E}(\lambda)$ is also unimodular. □

Example 8. *To continue our example, we use the extended GCD algorithm to find*

$$Q = \begin{pmatrix} 1 & -\frac{4}{9}\lambda^4 + \frac{4}{3}\lambda^3 - \frac{13}{9}\lambda^2 + \frac{8}{3}\lambda - \frac{10}{9} \\ 0 & 1 \end{pmatrix}$$

such that $Q(\lambda)b_{22}(\lambda) = [0; 1]$. *We obtain the unimodular matrix polynomial*

$$V = V_2 = Q^{-1} = \begin{pmatrix} 1 & \frac{4}{9}\lambda^4 - \frac{4}{3}\lambda^3 + \frac{13}{9}\lambda^2 - \frac{8}{3}\lambda + \frac{10}{9} \\ 0 & 1 \end{pmatrix}.$$

The unimodular matrix polynomial $E(\lambda)$ *is given by*

$$E = \begin{pmatrix} 2\lambda^2 + 2 & \frac{17}{9}\lambda^4 + \frac{10}{9}\lambda^3 + \frac{55}{9}\lambda^2 + \frac{19}{9}\lambda + \frac{47}{9} \\ 2\lambda - 2 & \frac{17}{9}\lambda^3 - \frac{7}{9}\lambda^2 + \frac{28}{9}\lambda - \frac{29}{9} \end{pmatrix}.$$

Alternatively, we apply the extended GCD algorithm on $\text{rem}(b_{22}(\lambda), d_2(\lambda)) = [-\frac{4}{9}\lambda^3 + \frac{1}{9}\lambda^2 - \frac{8}{9}\lambda + \frac{2}{9}; 1]$ *to obtain*

$$\tilde{Q} = \begin{pmatrix} 1 & \frac{4}{9}\lambda^3 - \frac{1}{9}\lambda^2 + \frac{8}{9}\lambda - \frac{2}{9} \\ 0 & 1 \end{pmatrix},$$

$$\tilde{V} = \tilde{V}_2 = \tilde{Q}^{-1} = \begin{pmatrix} 1 & -\frac{4}{9}\lambda^3 + \frac{1}{9}\lambda^2 - \frac{8}{9}\lambda + \frac{2}{9} \\ 0 & 1 \end{pmatrix}$$

and

$$\tilde{E} = \begin{pmatrix} 2\lambda^2 + 2 & \lambda^4 + \frac{10}{9}\lambda^3 + \frac{31}{9}\lambda^2 + \frac{19}{9}\lambda + \frac{31}{9} \\ 2\lambda - 2 & \lambda^3 + \frac{1}{9}\lambda^2 + \frac{4}{3}\lambda - \frac{13}{9} \end{pmatrix}.$$

We obtain Smith forms that are different from those given in Example 1.

1.4 Performance Comparison

In this section, we compare our algorithm to Villard's method with good conditioning [76], which is also a deterministic sequential method for computing Smith forms with multipliers, and to 'MatrixPolynomialAlgebra[SmithForm]' in Maple. All the algorithms are implemented in exact arithmetic using Maple 13 so that the maximum allowable setting of digits that Maple uses (given by 'kernelopts(maxdigits)') is 38654705646. However, limitations of available memory may set the limit on the largest integer number much lower than this. We use the variant of Algorithm 2 given in Appendix A to compute local Smith forms.

To evaluate the performance of these methods, we generate several groups of diagonal matrices $D(\lambda)$ over $\mathbb{Q}$ and multiply them on each side by unimodular matrices of the form $L(\lambda)U(\lambda)$ and then by permutation matrices, where L is unit lower triangular, and U is unit upper triangular, both with off diagonal entries of the form $\lambda - i$ with $i \in \{-10, \ldots, 10\}$ a random integer. We find that row permutation on the test matrices does not affect the running time of the algorithms, while column permutation increases the size of rational numbers in the computation and therefore the running time of both our algorithm and Villard's method. We compare the results in two extreme cases: without column permutation on test matrices and with columns reversed. Each process is repeated five times for each $D(\lambda)$ and the median running time is recorded.

We use several parameters in the comparison, including the size n of the square matrix $A(\lambda)$, the bound d of the degrees of the entries in $A(\lambda)$, the number l of irreducible factors in $\det[A(\lambda)]$, and the largest order κ_{jn}. Note that when we compute a local Smith form at $p_j(\lambda)$, only coefficients up to $A^{(\kappa_{jn}-1)}(\lambda)$ in the expansion of $A(\lambda)$ in power of $p_j(\lambda)$ enter into calculation. Due to this truncation and operation of remainder for computing V from B_n, it is the degrees of the diagonal entries $d_i(\lambda)$ (depending on l, $s_j = \deg p_j(\lambda)$ and κ_{ji}) rather than the degree bound d of the entries in $A(\lambda)$ that directly affect the computational cost of our algorithm. Therefore, the running time mostly depends on n, l and κ_{jn}. The cost of Villard's method with good condition is a function of n and d, as well as the size of the coefficients, which we do not discuss here.

Our first group of test matrices $D_n(\lambda)$ are of the form

$$D_n(\lambda) = \mathrm{diag}[1, \ldots, 1, \lambda, \lambda(\lambda-1), \lambda^2(\lambda-1), \lambda^2(\lambda-1)^2],$$

where the matrix size n increases starting from 4. Hence, we have $d = 8$, $l = 2$, and $\kappa_{1n} = \kappa_{2n} = 2$ all fixed. (The unimodular matrices in the construction of $A(\lambda)$ each have degree 2.) A comparison of the cpu time of the various algorithms is plotted in Figure 1.4. We did not implement the re-use strategy for computing the reduced row-echelon form of $\mathcal{A}_k$ by storing the Gauss-Jordan transformations used to obtain $\mathrm{rref}(\mathcal{A}_{k-1})$, and then continuing with only the new columns of $\mathcal{A}_k$. This is because the built-in function LinearAlgebra[ReducedRowEchelonForm] is much faster than can be achieved by a user defined maple code for the same purpose; therefore, our running times could be reduced by a factor of

about 2 in this test if we had access to the internal LinearAlgebra[ReducedRowEchelonForm] code.

For the second test, we use test matrices $D_l(\lambda)$ of size 9×9, where

$$D_l(\lambda) = \mathrm{diag}[1, \ldots, 1, \prod_{j=1}^{l}(\lambda - j)]$$

with the number of roots of $\det[A(\lambda)]$ equal to $l = 1, 2, \cdots$. In other words, $n = 9$, $d = l + 4$ and $\kappa_{jn} = 1$ for $1 \le j \le l$. The running time of Villard's method does not directly rely on l but increases as d and the size of coefficients of entries in $A(\lambda)$ grows.

In the third test, we use 9×9 test matrices $D_k(\lambda)$ of the form

$$D_k(\lambda) = \mathrm{diag}[1, \ldots, 1, \lambda^k], \qquad (k = 1, 2, \ldots),$$

with $n = 9$, $l = 1$, $\kappa_{1n} = k$ and $d = k + 4$. The results are shown in Figure 1.6. As in the second test, Villard's method has an increasing cost as d grows. But the slope is almost flat for test matrices without column permutation, because multiplication by λ^k with a larger k will not increase the numbers of terms or the coefficient size of the polynomials in the test matrices $A(\lambda)$. We add a more general case in Figure 1.7 where

$$D_k(\lambda) = \mathrm{diag}[1, \ldots, 1, (\lambda - 1)^k], \qquad (k = 1, 2, \ldots).$$

If we had access to the internal LinearAlgebra[ReducedRowEchelonForm] code, the re-use strategy for computing $\mathrm{rref}(\mathcal{A}_k)$ from $\mathrm{rref}(\mathcal{A}_{k-1})$ would decrease the running time of our algorithm by a factor of nearly κ_{1n}, i.e. the slope of our algorithm in Figure 1.6 would decrease by one.

As the fourth test, we use matrices $D_n(\lambda)$ similar to those in the first test, but with irreducible polynomials of higher degree. Specifically, we define

$$D_n(\lambda) = \mathrm{diag}[1, \ldots, 1, p_1, p_1 p_2, p_1^2 p_2, p_1^2 p_2^2],$$

where $p_1 = \lambda^2 + \lambda + 1$, $p_2 = \lambda^4 + \lambda^3 + \lambda^2 + 1$, $\kappa_{1n} = 2$, $\kappa_{2n} = 2$, $d = 16$ and n increases, starting at 4. The cpu time of the various algorithms is plotted in Figure 1.8. It looks analogous to Figure 1.4.

In addition to these basic tests, we include two more complicated cases. In the fifth test, we use 9×9 test matrices $D_k(\lambda)$ of the form

$$D_k(\lambda) = \mathrm{diag}[1, \ldots, 1, \prod_{j=1}^{k}(\lambda^2 + j), \prod_{j=1}^{k}(\lambda^2 + j)^2, \prod_{j=1}^{k}(\lambda^2 + j)^k], \qquad (k = 2, 3, \ldots),$$

with $n = 9$, $l = k$, $\kappa_{jn} = k$ and $d = 2k^2 + 4$.The cpu time of the various algorithms is plotted in Figure 1.9.

As a final test, we define $n \times n$ matrices

$$D_n(\lambda) = \mathrm{diag}[1, 1, (\lambda^2+1), (\lambda^2+1)^2(\lambda^2+2), \ldots, \prod_{j=1}^{n-2}(\lambda^2+j)^{n-1-j}]$$

with $n = 3, 4, \ldots$, so that all the parameters n, $l = n-2$, $\kappa_{jn} = n-1-j$ and $d = (n-1)(n-2)+4$ increases simultaneously. The results are shown in Figure 1.10.

1.4.1 Discussion

The key idea in our algorithm is that it is much less expensive to compute local Smith forms than global Smith forms through a sequence of unimodular row and column operations. This is because (1) row reduction over R/pR in Algorithm 2 (or over K in the variant of Appendix A) is less expensive than computing Bézout's coefficients over R; (2) the size of the rational numbers that occur in the algorithm remain smaller (as we only deal with the leading terms of A in an expansion in powers of p rather than with all of A); and (3) each column of $V(\lambda)$ in a local Smith form only has to be processed once for each power of p in the corresponding diagonal entry of $D(\lambda)$. Once the local Smith forms are known, we combine them to form a (global) multiplier $V(\lambda)$ for $A(\lambda)$. This last step does involve triangularization of $B_n(\lambda)$ via the extended GCD algorithm, but this is less time consuming in most cases than performing elementary row and column operations on $A(\lambda)$ to obtain $D(\lambda)$. This is because (1) we only have to apply row operations to $B_n(\lambda)$ (as the columns are already correctly ordered); (2) with the diagonal entries $d_i(\lambda)$ obtained from local Smith forms, we keep the degree of polynomials (and therefore the number of terms) in the algorithm small with the operation $\mathrm{rem}(\cdot, d_i)$; and (3) the leading columns of $B_n(\lambda)$ tend to be sparse (as they consist of a superposition of local Smith forms, whose initial columns X_{-1} are a subset of the columns of the identity matrix). Sparsity is not used explicitly in our code, but it does reduce the work required to compute the Bézout's coefficients of a column.

Figure 1.11 and Figure 1.12 show the running time for each step of our algorithm (e.g. finding irreducible factors of $\det[A(\lambda)]$, computing local Smith forms, constructing $V(\lambda)$, and computing $E(\lambda)$) for the last two tests in Section 1.4. The obvious drawback of our algorithm is that we have to compute a local Smith form for each irreducible factor of $\Delta(\lambda)$ separately, while much of the work in deciding whether to accept a column in Algorithm 1 can be done for all the irreducible factors simultaneously by using extended GCDs. In our numerical experiments, it appears that in most cases, the benefit of computing local Smith forms outweighs the fact that there are several of them to compute.

We note that there is no additional computational cost to obtain $F(\lambda)$ to write the Smith form in terms of $A(\lambda) = E(\lambda)D(\lambda)F(\lambda)$. However, if the goal is to write $A(\lambda)$ in the Smith form $U(\lambda)A(\lambda)V(\lambda) = D(\lambda)$ as traditionally used, the computation of a matrix inverse is needed to obtain the left multiplier $U(\lambda) = E(\lambda)^{-1}$. We include Figure 1.13 and 1.14 to show how the performance of our algorithm is affected with U computed.

Although it is not implemented in this dissertation, it is easy to see that the local Smith form construction in Step 1 is easy to parallelize.

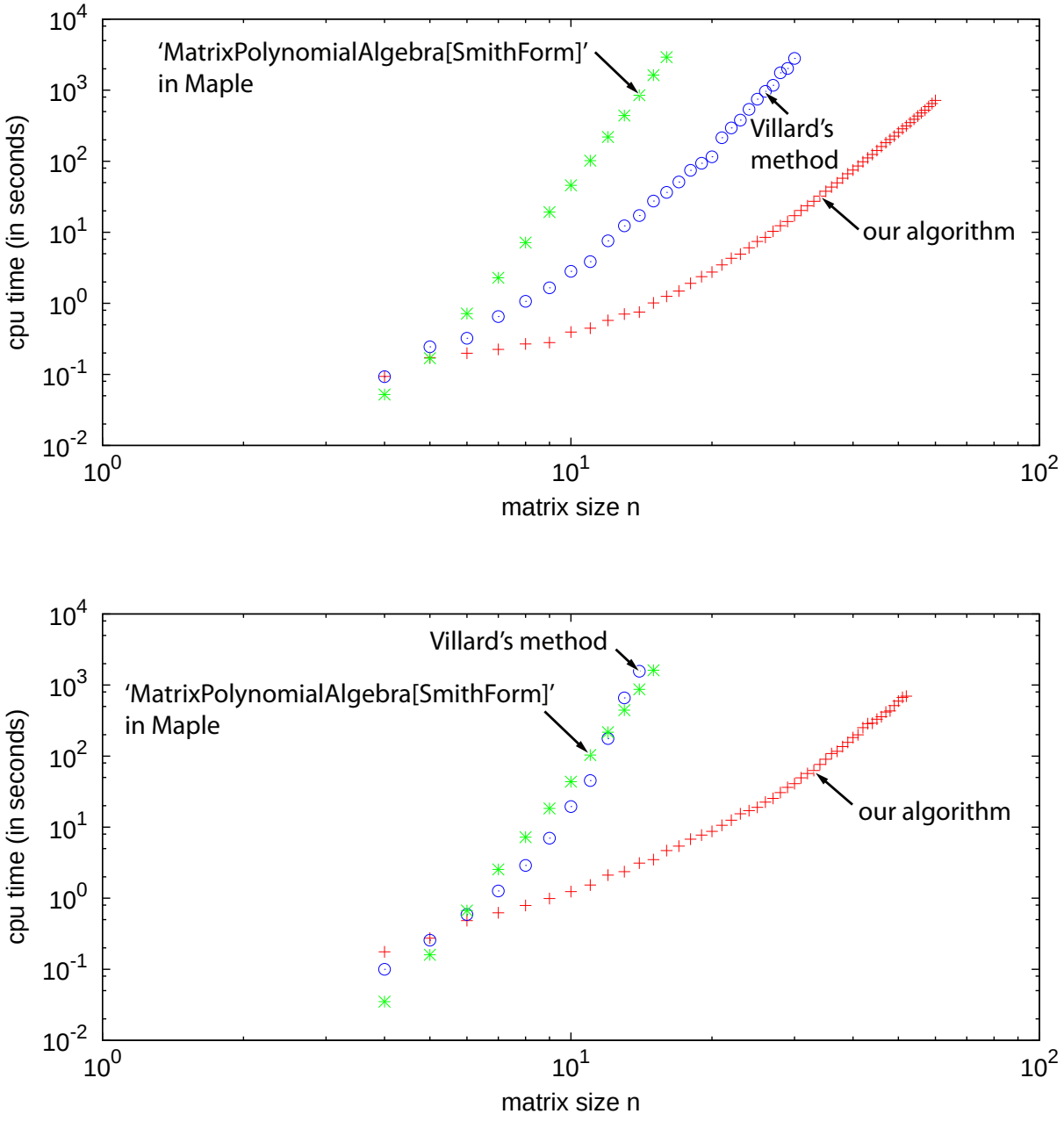

Figure 1.4: Running time vs. matrix size n for the first test, without column permutation (top) and with columns reversed (bottom).

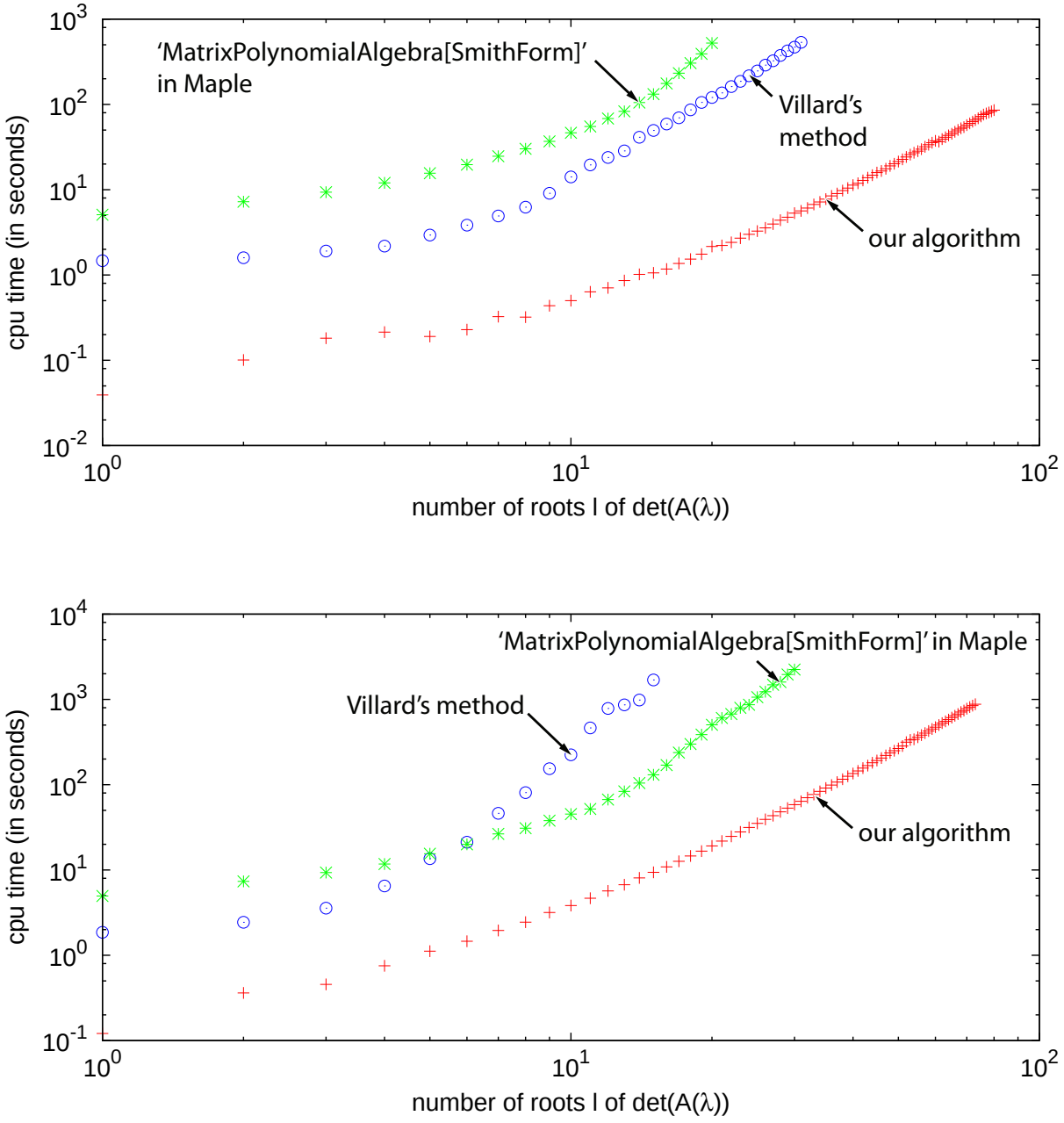

Figure 1.5: Running time vs. number of roots l of $\det[A(\lambda)]$ for the second test, without column permutation (top) and with columns reversed (bottom).

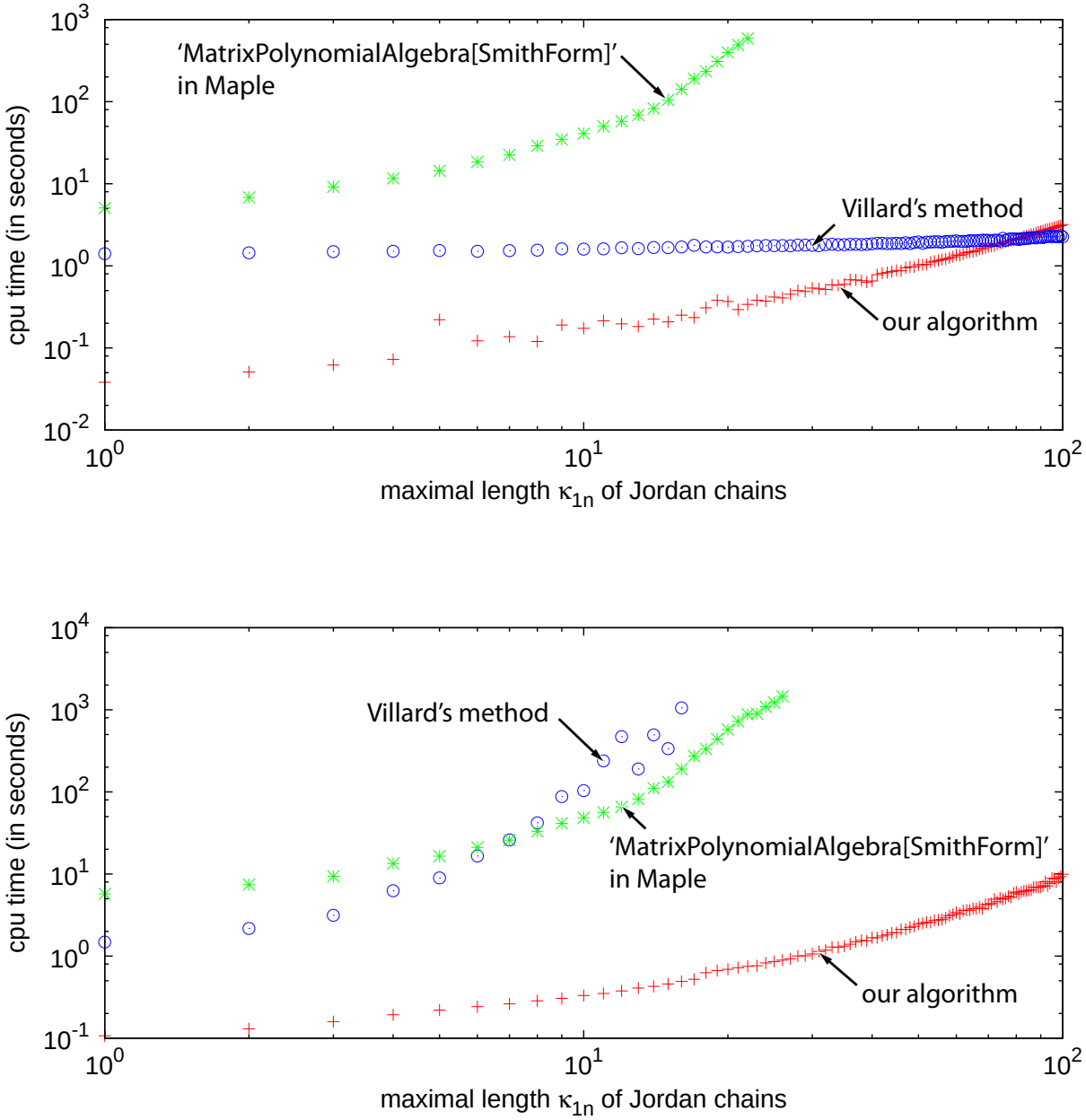

Figure 1.6: Running time vs. κ_{1n}, the maximal Jordan chain length, for the third test, without column permutation on test matrices (top) and with columns reversed (bottom).

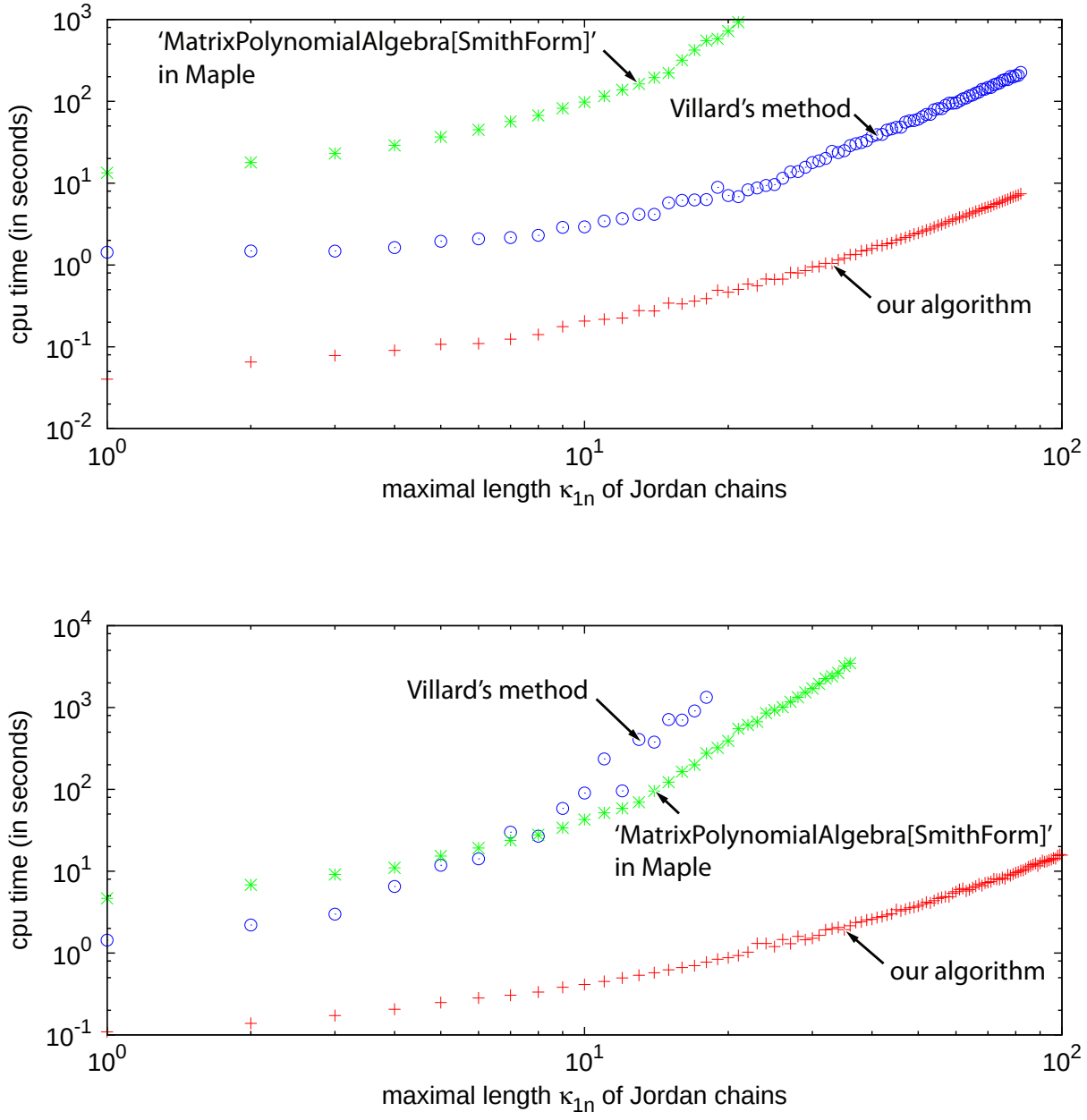

Figure 1.7: Running time vs. κ_{1n}, the maximal Jordan chain length, on a variant of the third test, without column permutation (top) and with columns reversed (bottom).

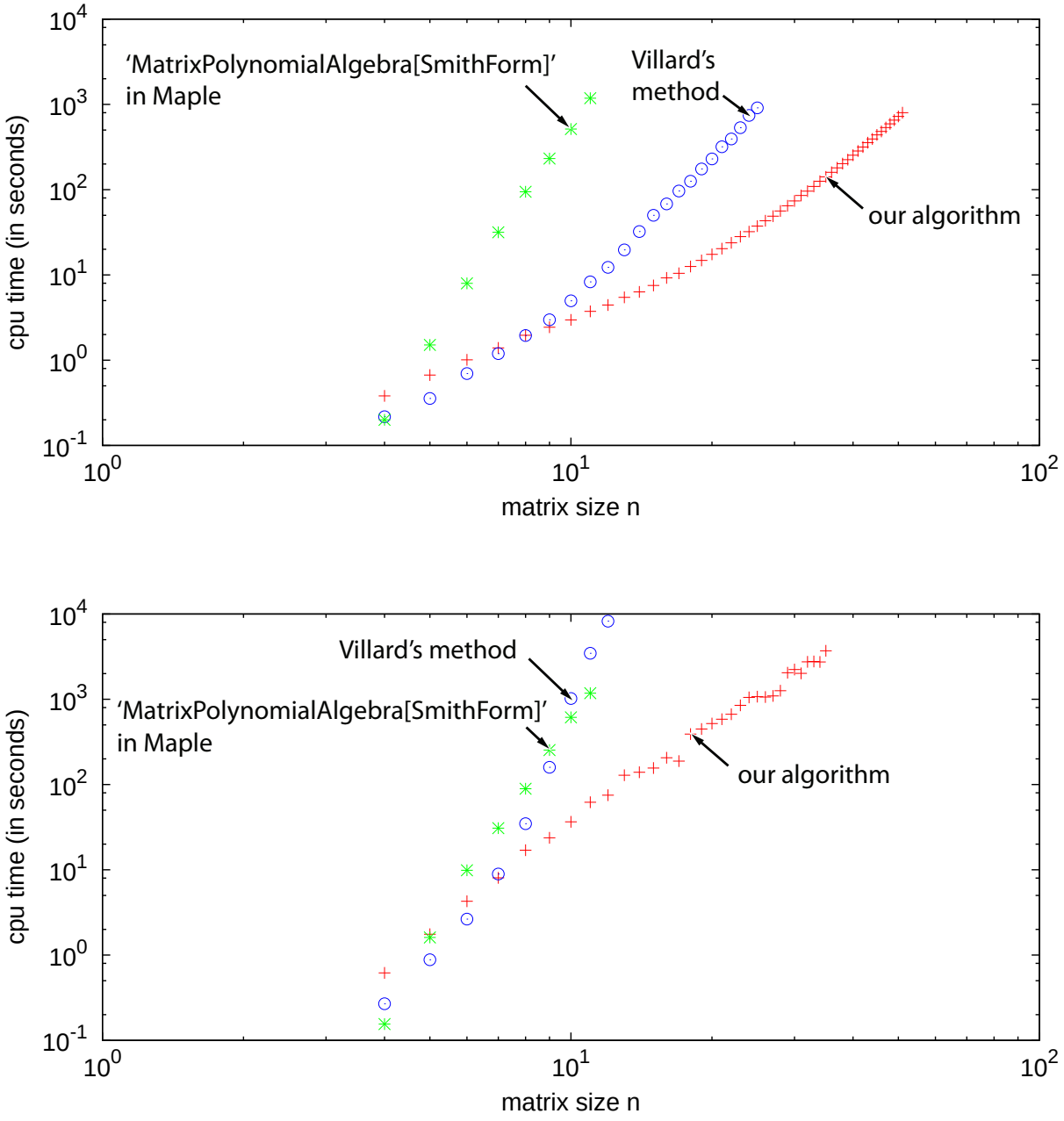

Figure 1.8: Running time vs. matrix size n for the fourth test, without column permutation (top) and with columns reversed (bottom).

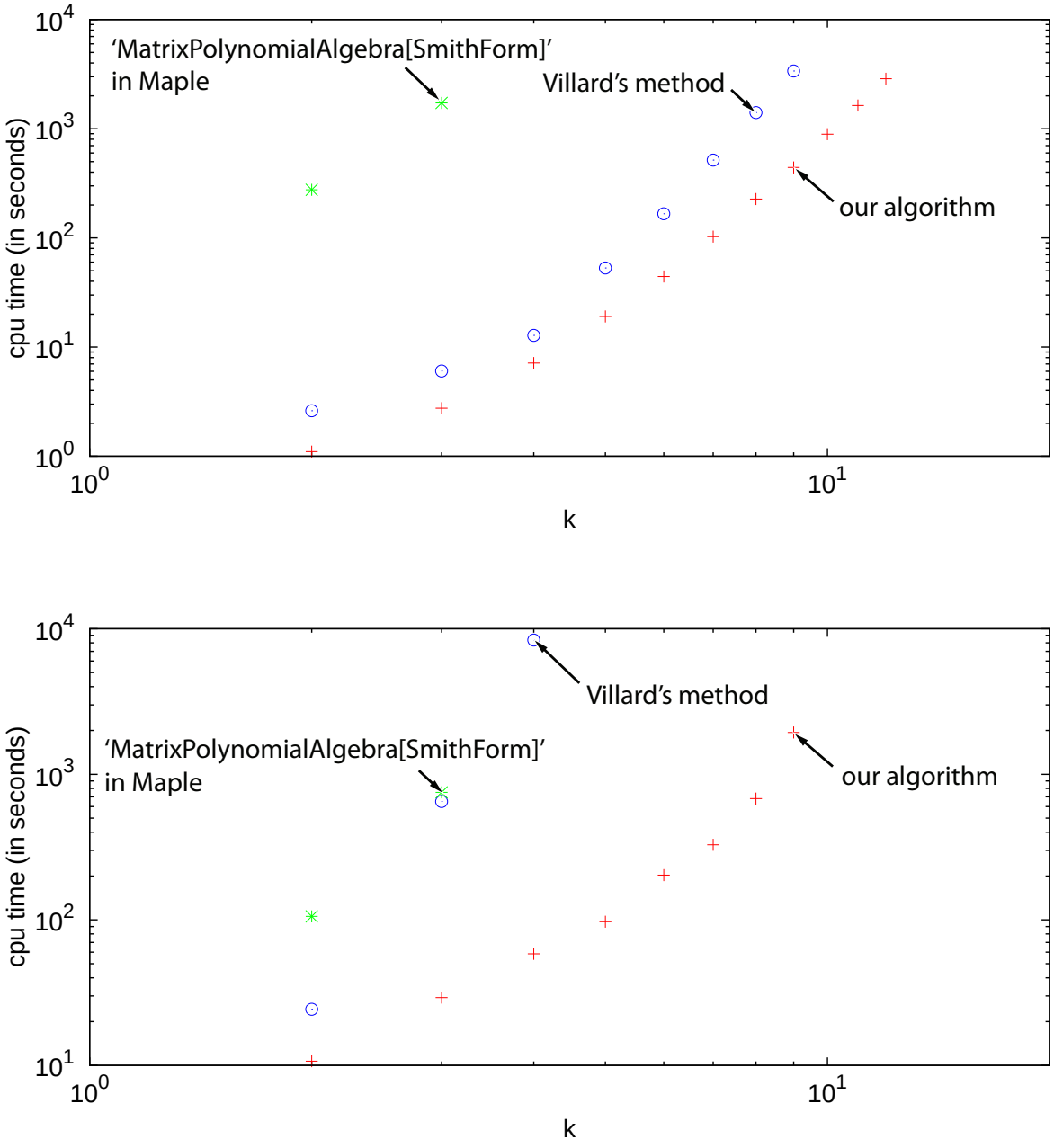

Figure 1.9: Running time vs. k for the fifth test, without column permutation (top) and with columns reversed (bottom).

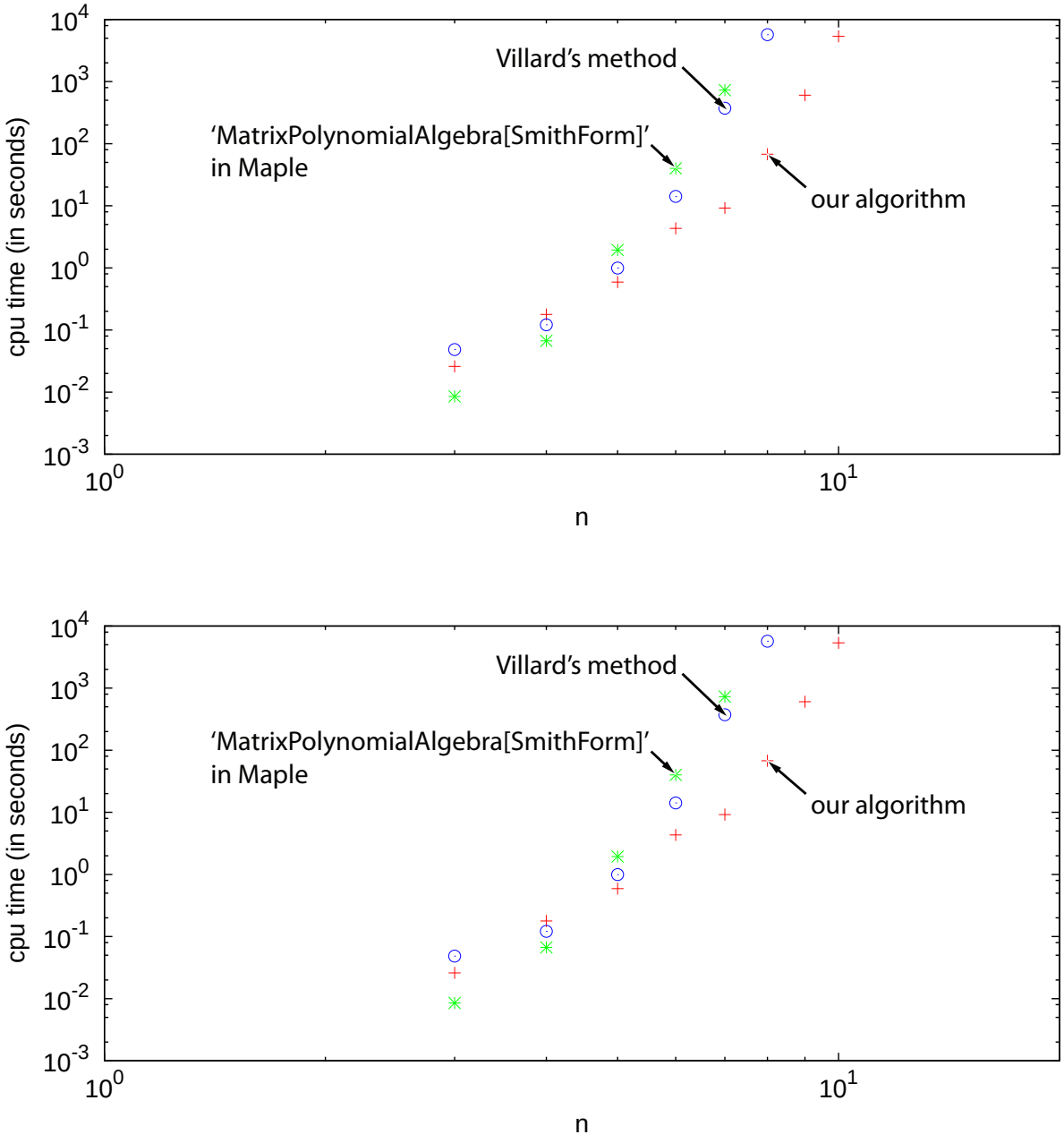

Figure 1.10: Running time vs. n for the sixth test, without column permutation (top) and with columns reversed (bottom).

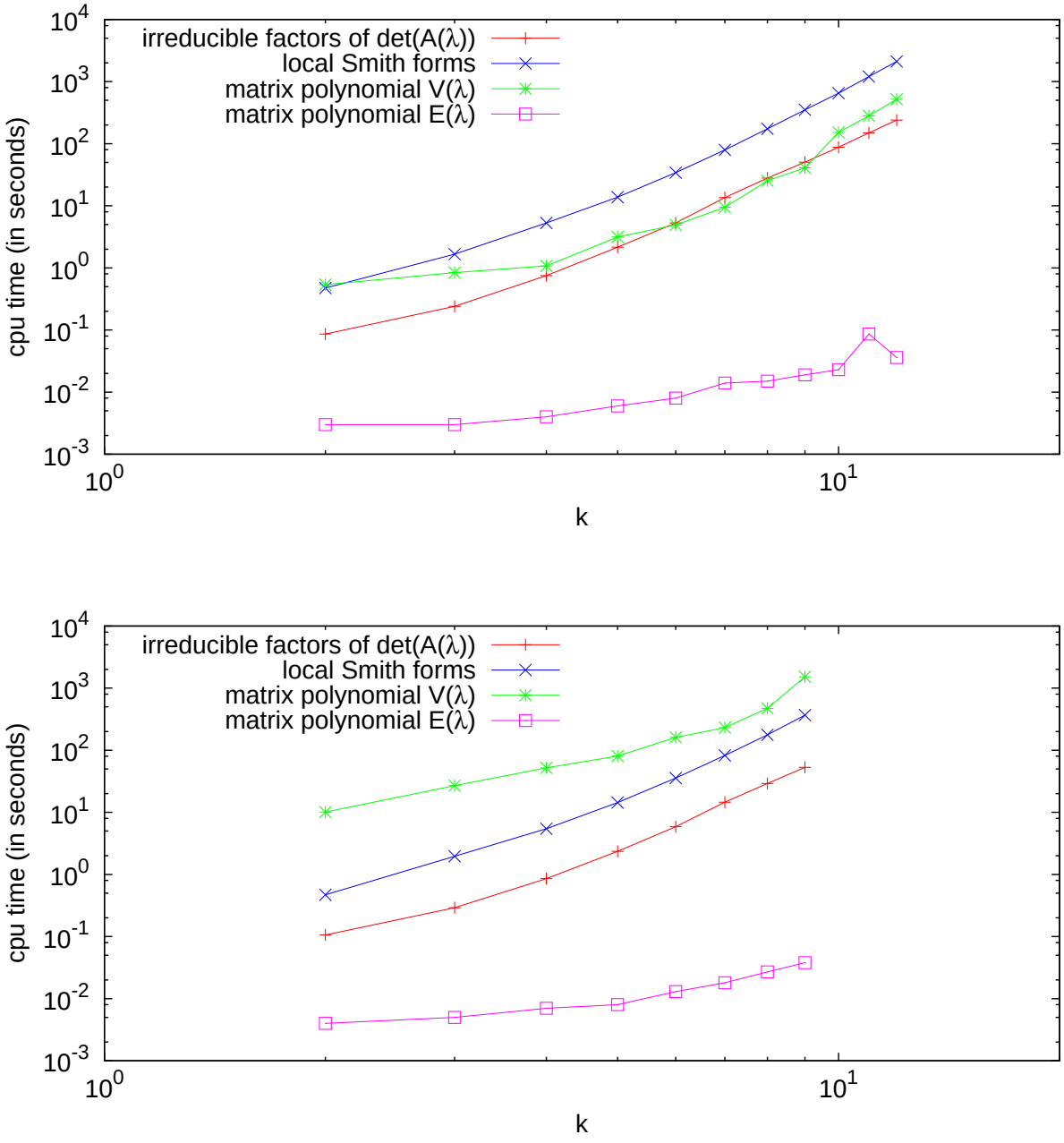

Figure 1.11: Running time of each step of our algorithm vs. k for the fifth test, without column permutation (top) and with columns reversed (bottom).

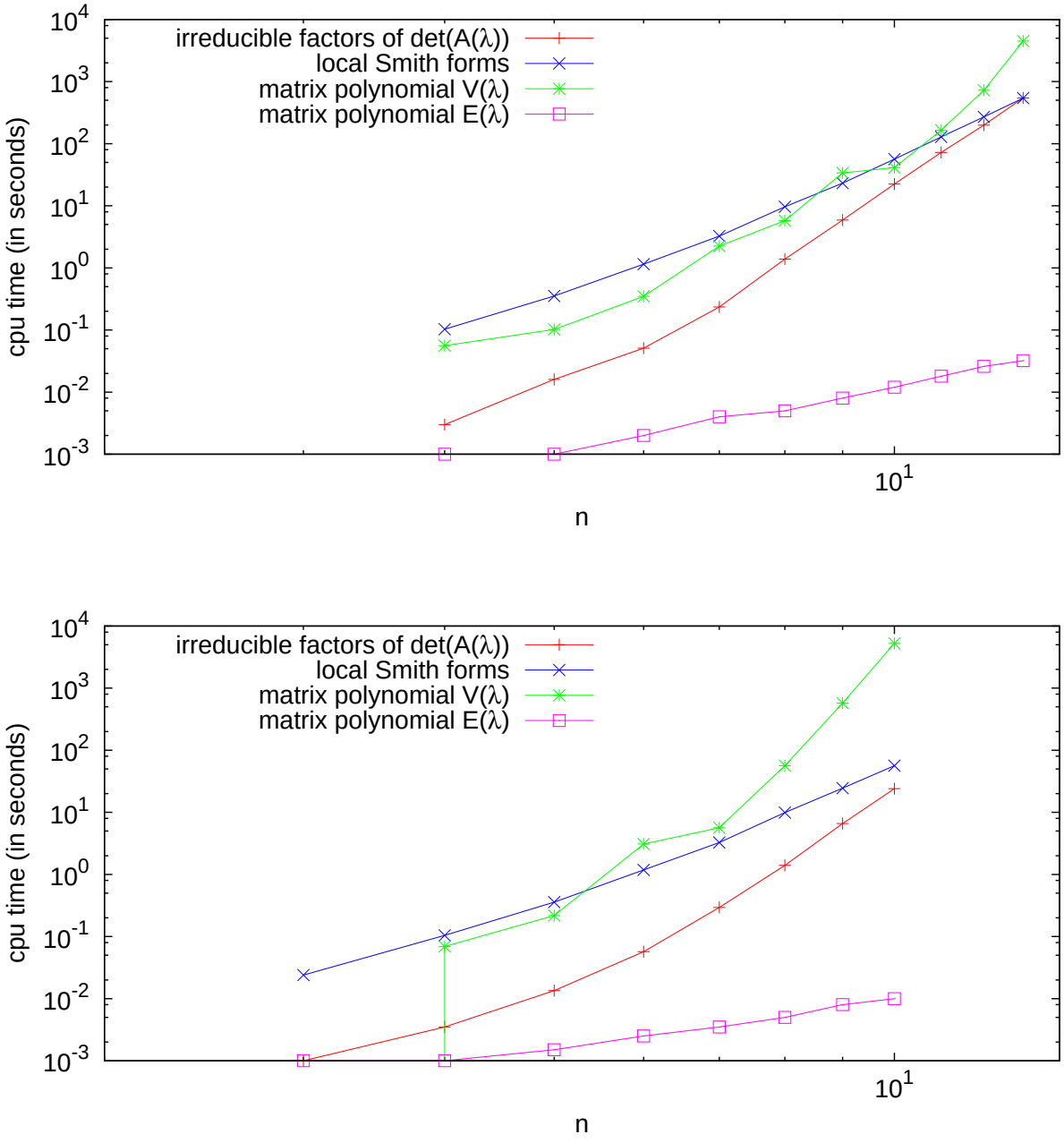

Figure 1.12: Running time of each step of our algorithm vs. n for the sixth test, without column permutation (top) and with columns reversed (bottom).

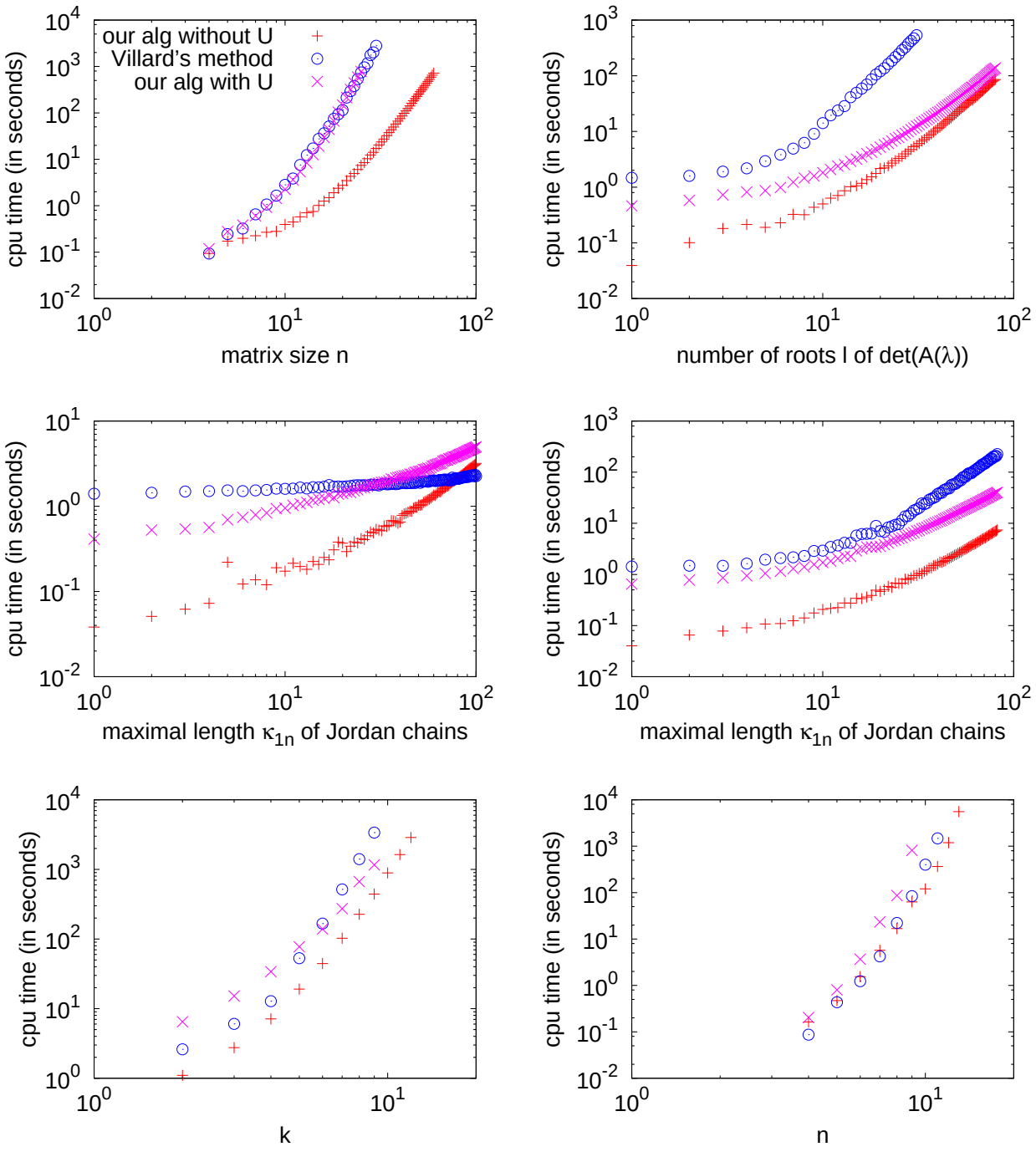

Figure 1.13: Running time of our algorithm with and without $U(\lambda)$ computed compared to Villard's method on Test 1, 2, 3-1, 3-2, 5 and 6, without column permutation.

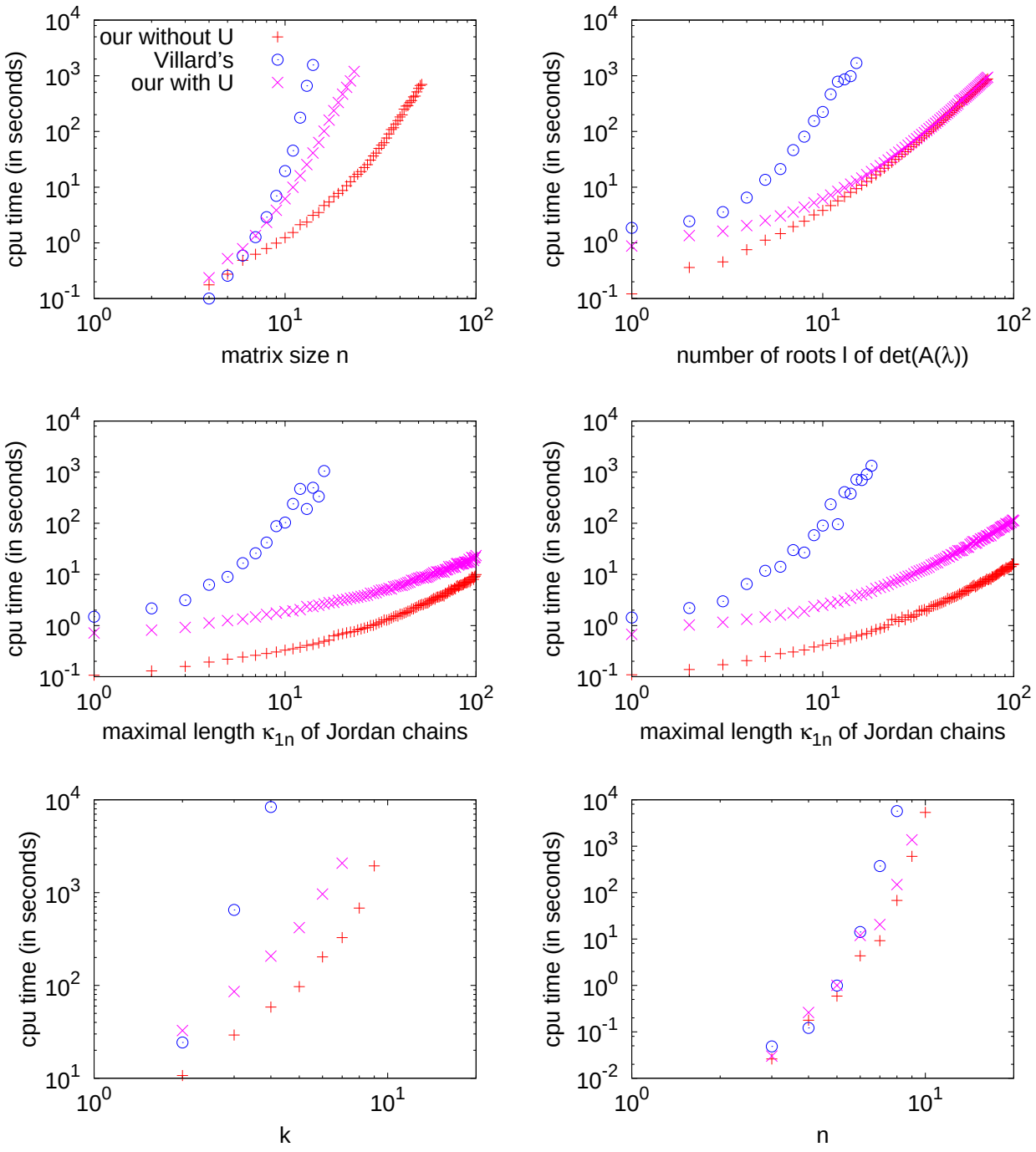

Figure 1.14: Running time of our algorithm with and without $U(\lambda)$ computed compared to Villard's method on Test 1, 2, 3-1, 3-2, 5 and 6, with columns reversed.

Chapter 2

Time-periodic Gravity-driven Water Waves with or without Surface Tension

2.1 Introduction

The study of time-periodic waves is an interesting topic in fluid mechanics. However, general numerical methods for computing time periodic solutions were designed with ordinary differential equations in mind, and are prohibitively expensive for partial differential equations. For example, "direct" methods like orthogonal collocation, as implemented in a popular software package AUTO, require solving a large nonlinear system involving all the degrees of freedom from discretizing in space at every time step; "indirect" methods such as shooting and multi-shooting rely on the computation of Jacobian matrices with respect to the variation of the initial conditions, which is also expensive. We adapt an adjoint-based optimal control algorithm developed by Ambrose and Wilkening [7, 4, 6] for solving general nonlinear two-point boundary value problems to perform a computational study of time-periodic gravity-driven water waves with or without surface tension.

The water wave problem is one of the oldest and the most classical problems in fluid mechanics [21, 22, 78]. After Newton's work [58], Euler [27, 28, 29], Laplace [55], Lagrange [52, 53, 54], Gerstner [32], Poisson [65], Cauchy [16], Russell [67], and Airy [1] in the eighteenth and first half of the nineteenth century all made significant contributions to describe the propagation of traveling waves before Stokes' most famous paper on water waves [71]. Since then numerous scientists such as Boussinesq [11], Rayleigh [66], Korteweg and de-Vries [49] have contributed substantially to study various types of waves (solitary, cnoidal, long, gravity, capillary, gravity-capillary), and a number of models were proposed in various asymptotic limits. Since the nineties, authors such as Crannell [23], Chen and Iooss [17], Iooss, Plotnikov and Toland [41, 64], and Cabral and Rosa [15] have worked on the existence

of time-periodic solutions of water wave models.

Our numerical method is a variant of the one developed by Ambrose and Wilkening to compute families of time-periodic solutions of the Benjamin-Ono equation [7, 4], and of the vortex sheet with surface tension [6]. The idea of the method is to minimize a nonlinear functional of the initial condition and supposed period that is positive unless the solution is period, in which case it is zero. We adapt an adjoint-based optimal control method [12, 13, 42, 56] to compute the gradient of the functional with respect to the initial condition, and use quasi-Newton line search algorithm BFGS [63] for minimization. Compared to standard methods for solving two point boundary value problems, namely orthogonal collocation [26] and shooting [70], this reduces the computational cost tremendously, especially when we use approximate Hessian information from the previous solution in the continuation algorithm.

A new feature of the water wave problem is that the evolution equations involve a Dirichlet to Neumann map relating the potential to its partial derivatives on the free surface. Numerical methods based on a boundary integral formulation were proposed by Ambrose and Wilkening to deal with this Dirichlet to Neumann map [5]. We start with two separate codes written by Ambrose and Wilkening, one for time-stepping the 2D water wave equations [5], and the other for computing time-periodic solutions of the vortex sheet with surface tension [6]. We combine these codes and add to them the capability of solving the adjoint system for the water wave problem, which we derive in Section 2.3 below. The adjoint system is linear but non-autonomous, and also involves a Dirichlet to Neumann operator.

This chapter is organized as follows: In Section 2.2, we describe the derivation of the model equations. In Section 2.3, we present our numerical methods for computing time-periodic solutions. We also propose techniques for finding symmetric breathers in this section. In Section 2.4, we give some numerical results. In Section 2.5, we discuss possible future work.

2.2 Equations of Motion

Following [2, 3, 40, 62, 64], we consider a two-dimensional inviscid, irrotational, incompressible fluid bounded below by a flat wall $y = -H_0$ where $H_0 > 0$ and above by an evolving free surface $\eta(x,t)$. This is a reasonable model of water waves in a large scale system such as the ocean in which inertia forces dominate viscous forces [25]. We assume that η is 2π-periodic in x.

For an inviscid fluid, the Navier-Stokes equation governing the flow is simplified to the Euler equation

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \boldsymbol{g}, \tag{2.1}$$

where $\mathbf{u}$ is the velocity field, ρ is the density, p is the pressure and $\boldsymbol{g} = -g\hat{\mathbf{y}}$ is the force due to gravity. Incompressibility implies that

$$\nabla\cdot\mathbf{u} = 0. \tag{2.2}$$

For irrotational flows, we have a velocity potential ϕ such that $\mathbf{u} = \nabla\phi$. Equation (2.1) can be rewritten in terms of ϕ

$$\nabla\phi_t + \frac{1}{2}\nabla(\nabla\phi)^2 = -\frac{1}{\rho}\nabla p + \boldsymbol{g}. \tag{2.3}$$

Integrating, we obtain Bernoulli's equation

$$\phi_t + \frac{1}{2}(\nabla\phi)^2 = -\frac{p}{\rho} - gy + c(t), \tag{2.4}$$

where $g > 0$ is the acceleration of gravity and $c(t)$ is a constant function in space. We assume the pressure above the free surface is a constant p_0 and has a jump proportional to the curvature κ across the free surface due to surface tension

$$p_0 - p\big|_{y=\eta} = \tau\kappa = \tau\partial_x\left(\frac{\eta_x}{\sqrt{1+\eta_x^2}}\right). \tag{2.5}$$

Note that $c(t)$ can be any function in t in the sense that the fluid velocity $\mathbf{u} = \nabla\phi$ will not change if $\phi(x, y, t)$ changes by a constant function in space. At $y = \eta$, (2.4) and (2.25) become

$$\varphi_t - \phi_y\eta_t + \frac{1}{2}(\nabla\phi)^2 = -\frac{p_0}{\rho} + \frac{\tau}{\rho}\partial_x\left(\frac{\eta_x}{\sqrt{1+\eta_x^2}}\right) - g\eta + c(t), \tag{2.6}$$

where $\varphi(x, t) = \phi(x, \eta(x, t), t)$ is the restriction of ϕ to the free surface, and satisfies $\phi_t = \varphi_t - \phi_y\eta_t$. The potential ϕ also satisfies the Laplace equation

$$\Delta\phi = 0 \tag{2.7}$$

by (2.2).

Another boundary condition on the free surface requires that particles on the free surface stay on it

$$\eta_t + \eta_x\phi_x = \phi_y. \tag{2.8}$$

The velocity at the bottom wall is tangential. It implies that $\phi_y = 0$ at $y = -H_0$.

Therefore, we pose the Cauchy problem as that of finding $\eta(x, t)$ and $\phi(x, y, t)$ with restriction $\varphi(x, t) = \phi(x, \eta(x, t), t)$ to the free surface such that

$$\eta(x, 0) = \eta_0(x), \quad \varphi(x, 0) = \varphi_0(x), \qquad t = 0, \tag{2.9a}$$

$$\phi_{xx} + \phi_{yy} = 0, \qquad -H_0 < y < \eta, \tag{2.9b}$$

$$\phi_y = 0, \qquad y = -H_0, \tag{2.9c}$$

$$\phi = \varphi, \qquad y = \eta, \tag{2.9d}$$

$$\eta_t + \eta_x\phi_x = \phi_y, \qquad y = \eta, \tag{2.9e}$$

$$\varphi_t = P\left[-\eta_x\phi_x\phi_y - \frac{1}{2}\phi_x^2 + \frac{1}{2}\phi_y^2 - g\eta + \tau\partial_x\left(\frac{\eta_x}{\sqrt{1+\eta_x^2}}\right)\right], \qquad y = \eta, \tag{2.9f}$$

where the density ρ has been set to be 1, and

$$P_0 f = \frac{1}{2\pi}\int_0^{2\pi} f(x)dx, \quad P = \mathrm{id} - P_0 \tag{2.10}$$

are orthogonal projections onto the mean and onto the space of zero-mean functions, respectively. Note that we have chosen

$$c(t) = \frac{p_0}{\rho} + P_0\left[\eta_x\phi_x\phi_y + \frac{1}{2}\phi_x^2 - \frac{1}{2}\phi_y^2 + g\eta - \tau\partial_x\left(\frac{\eta_x}{\sqrt{1+\eta_x^2}}\right)\right] \tag{2.11}$$

to eliminate the terms causing the mean of φ to drift as it evolves forward. That is, if we set the initial $\varphi(x,0)$ to have zero mean, it will remain true for all time. The advantage of this construction is that $\mathbf{u}$ is time-periodic with period T if and only if φ is periodic with the same period. (With arbitrary $c(t)$, φ only needs to be periodic up to a constant function in space for $\mathbf{u}$ to be periodic.)

We denote 2π-periodic functions in x by their Fourier series in numerical computation. A hat denote a Fourier coefficient $f(x) = \sum_k \hat{f}_k e^{ikx}$. Note that $\hat{f}_k$ and $\hat{f}_{-k}$ must be complex conjugates for $f(x)$ to be real for any x.

2.3 Numerical Methods

2.3.1 Boundary Integral Formulation

Following Ambrose and Wilkening [5], we explain the method for evaluating the Dirichlet to Neumann map in this section. For simplicity, we use $y = 0$ (instead of $y = -H_0$) as the bottom wall in this section to describe the boundary integral formulation of (2.9). We represent ϕ using a double-layer potential [30, 47, 50, 80] on the free surface, suppressing t in the notation when convenient:

$$\phi(\boldsymbol{x}) = \int_{\partial\Omega} -\frac{\partial N}{\partial n_{\boldsymbol{\zeta}}}(\boldsymbol{x},\boldsymbol{\zeta})\tilde{\mu}(\boldsymbol{\zeta})ds_{\boldsymbol{\zeta}}, \quad N(\boldsymbol{x},\boldsymbol{\zeta}) = -\frac{1}{2\pi}\log|\boldsymbol{x}-\boldsymbol{\zeta}|, \tag{2.12}$$

where $\tilde{\mu}$ is the dipole distribution. Rather than use a single layer potential on the bottom wall, which doubles the number of unknowns, we extend the function evenly in y and use an identical double-layer potential on the mirror image surface, $y = -\eta(x)$. The boundary condition $\phi_y = 0$ at $y = 0$ is automatically satisfied by the even extension. We parametrize the free and mirror surfaces by $\boldsymbol{\zeta} = (\alpha, \pm\eta(\alpha))$ so that

$$-\frac{\partial N}{\partial n_{\boldsymbol{\zeta}}}ds_{\boldsymbol{\zeta}} = \frac{1}{2\pi}\frac{(\alpha-x)(-\eta'(\alpha)) + (\pm\eta(\alpha)-y)(\pm 1)}{(\alpha-x)^2 + (\pm\eta(\alpha)-y)^2}d\alpha.$$

Next we use $\frac{1}{2}\cot\frac{x+iy}{2} = PV\sum_k \frac{(x+2\pi k)-iy}{(x+2\pi k)^2+y^2}$ to sum over periodic images, which gives

$$\phi(x,y) = \frac{1}{2\pi}\int_0^{2\pi} \left[K_1(x,y,\alpha) + K_2(x,y,\alpha)\right]\mu(\alpha)d\alpha, \tag{2.13}$$

where $\mu(\alpha) = \tilde{\mu}(\alpha,\eta(\alpha))$, and

$$K_1(x,y,\alpha) = \mathrm{Im}\left\{\frac{1}{2}\cot\left(\frac{(\alpha-x)-i(\eta(\alpha)-y)}{2}\right)\left[1-i\eta'(\alpha)\right]\right\}, \tag{2.14}$$

$$K_2(x,y,\alpha) = \mathrm{Im}\left\{\frac{1}{2}\cot\left(\frac{(\alpha-x)+i(\eta(\alpha)+y)}{2}\right)\left[-1-i\eta'(\alpha)\right]\right\}. \tag{2.15}$$

Conjugating and changing signs, we obtain

$$\begin{aligned} K_1(z,\alpha) &= \mathrm{Im}\left\{-\frac{1}{2}\cot\left(\frac{(\alpha-x)+i(\eta(\alpha)-y)}{2}\right)\left[1+i\eta'(\alpha)\right]\right\} \\ &= \mathrm{Im}\left\{\frac{\zeta'(\alpha)}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right)\right\}, \end{aligned} \tag{2.16}$$

$$\begin{aligned} K_2(z,\alpha) &= \mathrm{Im}\left\{-\frac{1}{2}\cot\left(\frac{(\alpha-x)-i(\eta(\alpha)+y)}{2}\right)\left[-1+i\eta'(\alpha)\right]\right\} \\ &= \mathrm{Im}\left\{-\frac{\bar{\zeta}'(\alpha)}{2}\cot\left(\frac{z-\bar{\zeta}(\alpha)}{2}\right)\right\}, \end{aligned} \tag{2.17}$$

with $z = x+iy$ and $\zeta(\alpha) = \alpha + i\eta(\alpha)$. As z approaches the boundary, the Plemelj formulas [57] show that the double-layer kernel approaches a delta-function plus a principal value integral, which we regularize via

$$\phi(x,\eta(x)^-) = \frac{\mu(x)}{2} + \frac{1}{2\pi}\int_0^{2\pi}\left[\tilde{K}_1(x,\alpha) + K_2(x,\eta(x),\alpha)\right]\mu(\alpha)d\alpha, \tag{2.18}$$

$$\tilde{K}_1(x,\alpha) = \mathrm{Im}\left\{\frac{\zeta'(\alpha)}{2}\cot\left(\frac{\zeta(x)-\zeta(\alpha)}{2}\right) - \frac{1}{2}\cot\left(\frac{x-\alpha}{2}\right)\right\}. \tag{2.19}$$

Note that $\tilde{K}_1$ is continuous if we define $\tilde{K}_1(x,x) = \frac{-\eta''(x)}{2(1+\eta'(x)^2)}$. Equation (2.18) is a second kind Fredholm integral equation which may be solved for $\mu(x)$ in terms of $\varphi(x) = \phi(x,\eta(x)^-)$. In the numerical simulations, η, φ and ζ are represented by their values at M equally spaced points $x_k = 2\pi k/M$, $0 \le k < M$. The x-derivative of η are computed using the FFT.

Once $\mu(x)$ is known, we may compute the normal derivative $\frac{\partial\phi}{\partial n}(x)$ as follows. First, for a field point z in the fluid away from the boundary, we may differentiate (2.13) and integrate

by parts to obtain

$$\begin{aligned}\phi_x - i\phi_y &= \frac{1}{2\pi}\int_0^{2\pi} -i\partial_z \left[\frac{\zeta'(\alpha)}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right) - \frac{\bar{\zeta}'(\alpha)}{2}\cot\left(\frac{z-\bar{\zeta}(\alpha)}{2}\right)\right]\mu(\alpha)d\alpha \\ &= \frac{1}{2\pi}\int_0^{2\pi} \frac{i}{2}\partial_\alpha \left[\cot\left(\frac{z-\zeta(\alpha)}{2}\right) - \cot\left(\frac{z-\bar{\zeta}(\alpha)}{2}\right)\right]\mu(\alpha)d\alpha \\ &= \frac{1}{2\pi i}\int_0^{2\pi} \left[\frac{1}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right) - \frac{1}{2}\cot\left(\frac{z-\bar{\zeta}(\alpha)}{2}\right)\right]\mu'(\alpha)d\alpha.\end{aligned} \tag{2.20}$$

If z approaches $(x,\eta(x))$ along the normal direction from the interior of the domain, we find that

$$\begin{aligned}\sqrt{1+\eta'(x)^2}\frac{\partial\phi}{\partial n} &= \mathrm{Re}\left[(-\eta'(x)+i)(\phi_x(z)-i\phi_y(z))\right] \\ &= \frac{1}{2\pi}\mathrm{Re}\int_0^{2\pi}(1+i\eta'(x))\left[\frac{1}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right) - \frac{1}{2}\cot\left(\frac{z-\bar{\zeta}(\alpha)}{2}\right)\right]\mu'(\alpha)d\alpha \\ &= \frac{1}{2\pi}\mathrm{Re}\int_0^{2\pi}\left[\frac{\zeta'(x)}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right) - \frac{\zeta'(x)}{2}\cot\left(\frac{z-\bar{\zeta}(\alpha)}{2}\right)\right]\mu'(\alpha)d\alpha.\end{aligned} \tag{2.21}$$

To evaluate the limit as $z\to(x,\eta(x))$, we subtract and add $\frac{\zeta'(\alpha)}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right)\mu'(\alpha)$ from the first term of the integrand and write it as a sum of two terms

$$\left[\frac{\zeta'(x)}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right) - \frac{\zeta'(\alpha)}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right)\right]\mu'(\alpha) + \frac{\zeta'(\alpha)}{2}\cot\left(\frac{z-\zeta(\alpha)}{2}\right)\mu'(\alpha). \tag{2.22}$$

Let $z=\zeta(x)-i\epsilon$, and let $\epsilon\to 0$. The Plemelj formula [57] tells us that the second term in (2.22) gives a jump equal to $\frac{i}{2}\mu'(x)$, and a principal value integral. The term in brackets is continuous at $\epsilon=0$. Therefore, the first term in (2.22) remains a Riemann integral in the limit as $\epsilon\to 0$. We can evaluate Riemann integrals as principal value integrals. Since the second term inside brackets makes sense as a principal value integral alone, and the difference is also defined as a principal value integral, we conclude that the first term has a limit as a principal value integral. Splitting these terms apart, we obtain three principal value integrals (two of which cancel) plus a term $\frac{i}{2}\mu'(x)$, which disappears when the real part is taken. Regularizing the remaining principal value integral, we find that as $z\to(x,\eta(x))$,

$$\begin{aligned}(2.21)\to &\frac{1}{2\pi}\mathrm{Re}\int_0^{2\pi}\left[\frac{\zeta'(x)}{2}\cot\left(\frac{\zeta(x)-\zeta(\alpha)}{2}\right) - \frac{1}{2}\cot\left(\frac{x-\alpha}{2}\right)\right]\mu'(\alpha)d\alpha \\ &+ \frac{1}{2}H\left[\mu'\right](x) - \frac{1}{2\pi}\mathrm{Re}\int_0^{2\pi}\frac{\zeta'(x)}{2}\cot\left(\frac{\zeta(x)-\bar{\zeta}(\alpha)}{2}\right)\mu'(\alpha)d\alpha,\end{aligned} \tag{2.23}$$

where $Hf(x) = \frac{1}{\pi} PV \int_{-\infty}^{\infty} \frac{f(\alpha)}{x-\alpha} = PV \int_0^{2\pi} \frac{f(\alpha)}{2\pi} \cot\left(\frac{x-\alpha}{2}\right) d\alpha$ is the Hilbert transform, which has symbol $\hat{H}_k = -i\mathrm{sgn}(k)$. The limiting value of the bracketed term in the first integrand of (2.23) as $\alpha \to x$ is $\frac{i\eta''(x)}{2(1+i\eta'(x))}$. Equation (2.23) is a variant of the Birkhoff-Rott integral [68].

From (2.23), we may compute ϕ_x and ϕ_y on the boundary using the formula

$$\begin{pmatrix} \phi_x \\ \phi_y \end{pmatrix} = \frac{1}{\sqrt{1+\eta'(x)^2}} \begin{pmatrix} 1 & -\eta'(x) \\ \eta'(x) & 1 \end{pmatrix} \begin{pmatrix} \varphi'(x)/\sqrt{(1+\eta'(x)^2)} \\ \frac{\partial \phi}{\partial n}(x, \eta(x)) \end{pmatrix}, \tag{2.24}$$

which allows the formulas (2.9) for η_t and φ_t to be evaluated. As the water wave is not stiff unless the surface tension is large, we use the 5th order explicit Runge-Kutta method DOPRI5 [36] to evolve η and φ from time $t = 0$ to $t = T$.

After the equations are solved, the velocity and pressure at any point in the fluid can be computed using (2.20) for velocity and

$$p = p_0 + P_0 \left[-\phi_y \eta_t + \frac{1}{2}\phi_x^2 + \frac{1}{2}\phi_y^2 + g\eta - \tau \partial_x \left(\eta_x / \sqrt{1+\eta_x^2} \right) \right] - \phi_t - \frac{1}{2}(u^2+v^2) - gy \tag{2.25}$$

for pressure. Formulas for ϕ_t in (2.25) are derived in terms of the known quantities η_t and φ_t in [5] and implemented in Ambrose and Wilkening's code for time-stepping the water wave problem.

2.3.2 Time-periodic Solutions

We now present our algorithm for computing time-periodic solutions of the gravity-driven water wave with or without surface tension. Let $q = (\eta, \varphi)$ and denote the system (2.9) abstractly by

$$q_t = f(q), \quad q(x,0) = q_0(x). \tag{2.26}$$

We define the inner product

$$\langle q_1, q_2 \rangle = \frac{1}{2\pi} \int_0^{2\pi} \left[\eta_1(x)\eta_2(x) + \varphi_1(x)\varphi_2(x) \right] dx. \tag{2.27}$$

Next we define a functional G of the initial conditions and supposed period via

$$G(q_0, T) = \frac{1}{2}\|q(\cdot, T) - q_0\|^2 = \frac{1}{2\pi} \int_0^{2\pi} \frac{1}{2} \left[\eta(x,T) - \eta_0(x) \right]^2 + \frac{1}{2} \left[\varphi(x,T) - \varphi_0(x) \right]^2 dx, \tag{2.28}$$

where $q = (\eta, \varphi)$ solves (2.9). $G(q_0, T)$ is zero if and only if the solution is time-periodic with period T.

To find time-periodic solutions, we minimize G, hoping to obtain $G = 0$. We use a templated C++ version of the limited memory BFGS algorithm [14, 63] developed by Wilkening

for the minimization. BFGS is a quasi-Newton line search algorithm that builds an approximate (inverse) Hessian matrix from the sequence of gradient vectors it encounters during the course of the line searches. In our continuation algorithm, we initialize the approximate Hessian with that of the previous minimization step (rather than the identity matrix), which leads to a tremendous reduction in the number of iterations required to converge (by factors of 10-20 in many cases). The initial conditions η_0 and φ_0 are represented in the BFGS algorithm by their Fourier coefficients

$$\eta(x,0) = \hat{\eta}_0(0) + \sum_{k=1}^{\infty} \left(\hat{\eta}_k(0) e^{ikx} + \overline{\hat{\eta}_k(0)} e^{-ikx} \right), \tag{2.29}$$

$$\varphi(x,0) = \hat{\varphi}_0(0) + \sum_{k=1}^{\infty} \left(\hat{\varphi}_k(0) e^{ikx} + \overline{\hat{\varphi}_k(0)} e^{-ikx} \right), \tag{2.30}$$

where $\hat{\eta}_0(0)$ and $\hat{\varphi}_0(0)$ are real numbers. Given a set of these Fourier coefficients and a proposed period T, we must supply BFGS with the function value G as well as its gradient ∇G.

The T-derivative of G is easily obtained by evaluating

$$\frac{\partial G}{\partial T} = \frac{1}{2\pi} \int_0^{2\pi} \left[\eta(x,T) - \eta_0(x) \right] \eta_t(x,T) + \left[\varphi(x,T) - \varphi_0(x) \right] \varphi_t(x,T) dx \tag{2.31}$$

using the trapezoidal rule. All qualities $\eta(\cdot,T)$, $\eta_t(\cdot,T)$, $\varphi(\cdot,T)$ and $\varphi_t(\cdot,T)$ are already known by solving (2.9). One way to compute the other components of the gradient of G, namely $\partial G/\partial \mathrm{Re}(\hat{\eta}_k(0))$, $\partial G/\partial \mathrm{Im}(\hat{\eta}_k(0))$, $\partial G/\partial \mathrm{Re}(\hat{\varphi}_k(0))$ and $\partial G/\partial \mathrm{Im}(\hat{\varphi}_k(0))$ would be to solve the variational equation

$$\dot{q}_t = Df(q(\cdot,t))\dot{q} \tag{2.32}$$

with initial condition $\dot{q}(\cdot,0) = \dot{q}_0$ to obtain $\dot{q}(\cdot,T) = (\dot{\eta}(\cdot,T), \dot{\varphi}(\cdot,T))$ in

$$\dot{G} = \left. \frac{d}{d\epsilon} \right|_{\epsilon=0} G(q_0 + \epsilon \dot{q}_0, T) = \left\langle q(\cdot,T) - q_0, \dot{q}(\cdot,T) - \dot{q}_0 \right\rangle \tag{2.33}$$

for $\dot{q}_0 = (e^{ikx} + e^{-ikx}, 0)$, $\dot{q}_0 = (ie^{ikx} - ie^{-ikx}, 0)$, $\dot{q}_0 = (0, e^{ikx} + e^{-ikx})$ and $\dot{q}_0 = (0, ie^{ikx} - ie^{-ikx})$, respectively. To avoid the expense of solving (2.32) repeatedly for each value of k and for both components of q, we solve a single adjoint PDE to find $\delta G/\delta q_0$ such that

$$\dot{G} = \left\langle \underbrace{q(\cdot,T) - q_0}_{\tilde{q}_0}, \dot{q}(\cdot,T) - \dot{q}_0 \right\rangle = \left\langle \underbrace{\tilde{q}(\cdot,T) - \tilde{q}_0}_{\delta G/\delta q_0}, \dot{q}_0 \right\rangle. \tag{2.34}$$

Here $\tilde{q}$ solves the adjoint problem

$$\tilde{q}_s = Df(q(\cdot,T-s))^* \tilde{q} \tag{2.35}$$

backward in time with initial condition $\tilde{q}_0 = q(\cdot, T) - q_0$. This definition of $\tilde{q}$ ensures that $\langle \tilde{q}(\cdot, T-t), \dot{q}(\cdot, t)\rangle$ is constant, which justifies $\langle \tilde{q}_0, \dot{q}(\cdot, T)\rangle = \langle \tilde{q}(\cdot, T), \dot{q}_0\rangle$ in the final equality of (2.34). Like the variational equation (2.32), the adjoint equation (2.35) is linear and non-autonomous due to the presence of the solution $q(t)$ of (2.9) in the equation. Note that (2.35) only needs to be solved once to obtain all the derivatives $\partial G/\partial \mathrm{Re}(\hat{\eta}_k(0))$, $\partial G/\partial \mathrm{Im}(\hat{\eta}_k(0))$, $\partial G/\partial \mathrm{Re}(\hat{\varphi}_k(0))$ and $\partial G/\partial \mathrm{Im}(\hat{\varphi}_k(0))$ simultaneously, and therefore ∇G can be computed in approximately the same amount of time as G.

$Df(q)$ and $Df(q)^*$ can be computed explicitly. The equation (2.32) is simply

$$\dot{\eta}(x,0) = \dot{\eta}_0(x), \quad \dot{\varphi}(x,0) = \dot{\varphi}_0(x), \qquad t = 0, \tag{2.36a}$$

$$\dot{\phi}_{xx} + \dot{\phi}_{yy} = 0, \qquad -H_0 < y < \eta, \tag{2.36b}$$

$$\dot{\phi}_y = 0, \qquad y = -H_0, \tag{2.36c}$$

$$\dot{\phi} + \phi_y \dot{\eta} = \dot{\varphi}, \qquad y = \eta, \tag{2.36d}$$

$$\dot{\eta}_t + \dot{\eta}_x \phi_x + \eta_x \dot{\phi}_x + \eta_x \phi_{xy} \dot{\eta} = \dot{\phi}_y + \phi_{yy} \dot{\eta}, \qquad y = \eta, \tag{2.36e}$$

$$\dot{\varphi}_t = P\left[-(\dot{\eta}\phi_x\phi_y)' - (\star) - g\dot{\eta} + \tau \partial_x \left(\frac{\dot{\eta}_x}{(1+\eta_x^2)^{3/2}} \right) \right], \qquad y = \eta, \tag{2.36f}$$

where $\star = (\phi_x, -\phi_y) \begin{pmatrix} 1 & \eta_x \\ -\eta_x & 1 \end{pmatrix} \begin{pmatrix} \dot{\phi}_x \\ \dot{\phi}_y \end{pmatrix}$, and a prime indicates an x-derivative along the free surface, e.g. $f' := \frac{d}{dx} f(x, \eta(x), t) = f_x + \eta_x f_y$. In (2.36d) and (2.36e), we account for the fact that ϕ, ϕ_x and ϕ_y in (2.9d) and (2.9e) are evaluated at the free surface $y = \eta$, which itself varies when the initial condition is perturbed. In (2.36f), $(\dot{\eta}\phi_x\phi_y)'$ is the sum of five terms

$$\begin{aligned} (\dot{\eta}\phi_x\phi_y)' &= \dot{\eta}_x\phi_x\phi_y + \dot{\eta}\phi_{xx}\phi_y + \dot{\eta}\eta_x\phi_{xy}\phi_y + \dot{\eta}\phi_x\phi_{xy} + \dot{\eta}\eta_x\phi_x\phi_{yy} \\ &= \dot{\eta}_x\phi_x\phi_y - \dot{\eta}\phi_{yy}\phi_y + \dot{\eta}\eta_x\phi_{xy}\phi_y + \dot{\eta}\phi_x\phi_{xy} + \dot{\eta}\eta_x\phi_x\phi_{yy}. \end{aligned} \tag{2.37}$$

Combined with $(\star)$, this gives $\left[\eta_x\phi_x\phi_y + \frac{1}{2}\phi_x^2 - \frac{1}{2}\phi_y^2\right]^{\cdot}$ in (2.9f). The equation $\tilde{q}_s = Df(q)^*\tilde{q}$ is obtained from

$$\begin{aligned} \langle \dot{q}, \tilde{q}_s \rangle = \langle \dot{q}_t, \tilde{q} \rangle = \frac{1}{2\pi} \int_0^{2\pi} &\left[(\dot{\phi}_y - \eta_x \dot{\phi}_x) + \phi_{yy}\dot{\eta} - \dot{\eta}_x\phi_x - \eta_x\phi_{xy}\dot{\eta} \right] \tilde{\eta} dx \\ + \frac{1}{2\pi} \int_0^{2\pi} &P\left[-(\dot{\eta}\phi_x\phi_y)' - (\star) - g\dot{\eta} + \tau\partial_x \left(\frac{\dot{\eta}_x}{(1+\eta_x^2)^{3/2}} \right) \right] \tilde{\varphi}\, dx. \end{aligned} \tag{2.38}$$

We move P from the left bracketed term to the right by self-adjointness to obtain

$$\begin{aligned} \langle \dot{q}, \tilde{q}_s \rangle = \frac{1}{2\pi} \int_0^{2\pi} &\left[\underline{(\dot{\phi}_y - \eta_x \dot{\phi}_x)} + \phi_{yy}\dot{\eta} - \dot{\eta}_x\phi_x - \eta_x\phi_{xy}\dot{\eta} \right] \tilde{\eta} dx \\ + \frac{1}{2\pi} \int_0^{2\pi} &\left[-(\dot{\eta}\phi_x\phi_y)' - \phi_x\dot{\phi}' + \underline{\phi_y(\dot{\phi}_y - \eta_x\dot{\phi}_x)} - g\dot{\eta} + \tau\partial_x \left(\frac{\dot{\eta}_x}{(1+\eta_x^2)^{3/2}} \right) \right] P\tilde{\varphi}\, dx. \end{aligned} \tag{2.39}$$

The difficult terms are underlined. These are the terms in which the partial derivatives of $\dot{\phi}$ are evaluated on the free surface. They cannot simply be integrated by parts as functions of x. Instead, we construct an auxiliary PDE such that integration by parts in the interior of the domain yields these terms as boundary terms. We combine all the underlined terms and define χ to solve Laplaces equation in the interior of the domain with homogeneous Neumann boundary conditions on the bottom wall and the Dirichlet condition $\chi = \tilde{\eta} + \phi_y P\tilde{\varphi}$ at the free surface. By Greens identity, we have

$$0 = \iint (\chi\Delta\dot{\phi} - \dot{\phi}\Delta\chi)dA = \int \chi\frac{\partial\dot{\phi}}{\partial n} - \dot{\phi}\frac{\partial\chi}{\partial n}ds = \int \chi(\dot{\phi}_y - \eta_x\dot{\phi}_x)\,dx - \int \dot{\phi}(\chi_y - \eta_x\chi_x)\,dx. \tag{2.40}$$

The first integral on the right hand side accounts for the underlined terms in (2.39) while the second has the desired form after expanding $\dot{\phi} = \dot{\varphi} - \phi_y\dot{\eta}$ on the free surface. Note that $\int_0^{2\pi} f'(x)P\tilde{\varphi}\,dx = \int_0^{2\pi} f'(x)(\mathrm{id} - P_0)\tilde{\varphi}\,dx = \int_0^{2\pi} f'(x)\tilde{\varphi}\,dx$ as $P_0\tilde{\varphi}$ is independent of x. Integrating by parts and simplifying, we obtain

$$\begin{aligned}
\langle \dot{q}, \tilde{q}_s\rangle &= \frac{1}{2\pi}\int_0^{2\pi} (\dot{\varphi} - \phi_y\dot{\eta})(\chi_y - \eta_x\chi_x)dx + \frac{1}{2\pi}\int_0^{2\pi} \left[\phi_{yy}\dot{\eta} - \dot{\eta}_x\phi_x - \eta_x\phi_{xy}\dot{\eta}\right]\tilde{\eta}dx \\
&\quad + \frac{1}{2\pi}\int_0^{2\pi} \left[-\phi_x\dot{\phi}' - g\dot{\eta}\right] P\tilde{\varphi}\,dx + \frac{1}{2\pi}\int_0^{2\pi} \left[-(\dot{\eta}\phi_x\phi_y)' + \tau\partial_x\left(\frac{\dot{\eta}_x}{(1+\eta_x^2)^{3/2}}\right)\right]\tilde{\varphi}\,dx \\
&= \frac{1}{2\pi}\int_0^{2\pi} \dot{\varphi}\left[\chi_y - \eta_x\chi_x + (\phi_x P\tilde{\varphi})'\right]\,dx + \frac{1}{2\pi}\int_0^{2\pi} \dot{\eta}\left[-\phi_y(\chi_y - \eta_x\chi_x) + \tilde{\eta}\phi_{yy} + (\phi_x\tilde{\eta})'\right. \\
&\quad \left. -\eta_x\phi_{xy}\tilde{\eta} + \phi_x\phi_y\tilde{\varphi}_x - \phi_y(\phi_x P\tilde{\varphi})' - gP\tilde{\varphi} + \tau\partial_x\left(\frac{\tilde{\varphi}_x}{(1+\eta_x^2)^{\frac{3}{2}}}\right)\right]\,dx \\
&= \frac{1}{2\pi}\int_0^{2\pi} \dot{\varphi}\left[\chi_y - \eta_x\chi_x + (\phi_x P\tilde{\varphi})'\right]\,dx + \frac{1}{2\pi}\int_0^{2\pi} \dot{\eta}\left[-\phi_y(\chi_y - \eta_x\chi_x) + \phi_x\tilde{\eta}_x\right. \\
&\quad \left. -\phi_y(\phi_x)'P\tilde{\varphi} - gP\tilde{\varphi} + \tau\partial_x\left(\frac{\tilde{\varphi}_x}{(1+\eta_x^2)^{\frac{3}{2}}}\right)\right]\,dx.
\end{aligned} \tag{2.41}$$

Here the last equality holds because $(P\varphi)_x = \varphi_x - (P_0\varphi)_x = \varphi_x$. Thus, the adjoint system

is given by

$$\tilde{\eta}(x,0) = \eta(x,T) - \eta_0(x), \quad \tilde{\varphi}(x,0) = \varphi(x,T) - \varphi_0(x), \qquad s = 0, \tag{2.42a}$$

$$\chi_{xx} + \chi_{yy} = 0, \qquad -H_0 < y < \eta, \tag{2.42b}$$

$$\chi_y = 0, \qquad y = -H_0, \tag{2.42c}$$

$$\chi = \tilde{\eta} + \phi_y P\tilde{\varphi}, \qquad y = \eta, \tag{2.42d}$$

$$\tilde{\varphi}_s = \chi_y - \eta_x\chi_x + (\phi_x P\tilde{\varphi})', \qquad y = \eta, \tag{2.42e}$$

$$\tilde{\eta}_s = -\phi_y(\diamond) + \phi_x\tilde{\eta}_x - gP\tilde{\varphi} + \tau\partial_x\left(\frac{\tilde{\varphi}_x}{(1+\eta_x^2)^{\frac{3}{2}}}\right), \qquad y = \eta, \tag{2.42f}$$

where $\diamond = \phi_x' P\tilde{\varphi} + \chi_y - \eta_x\chi_x$. Note that the adjoint problem has the same structure as the forward problem, with a Dirichlet to Neumann map involved in the evolution of $\tilde{\eta}(x,t)$ and $\tilde{\varphi}(x,t)$, which is also evaluated with the boundary integral formulation presented in [5].

To solve the adjoint equation numerically in the Runge-Kutta framework, the values of $q(\cdot, T-s)$ are needed between timesteps. We use the fourth order dense output formula in [69, 36] to compute q at these intermediate times, having stored q and q_t at each timestep when (2.9) was solved. This is enough to achieve fifth order accuracy in the adjoint problem.

2.3.3 Breather Solutions

In this section, we describe a technique we use specifically for breather solutions of (2.9) to achieve a factor of 4 improvement in speed. We always set the mean of φ_0 to zero in our numerical computation.

If at any moment $\eta(x,t)$ and $\varphi(x,t)$ are both even functions of x, then η_t and φ_t will also be even. We assume that, in addition to being even, $\varphi(x,0)$ changes sign upon translation by π, while $\eta(x,0)$ remains the same, that is, we look for breather solutions with initial conditions of the form

$$\hat{\eta}_k(0) = \begin{cases} 0 & k \text{ odd}, \\ c_{|k|} & k \text{ even}, \end{cases} \qquad \hat{\varphi}_k(0) = \begin{cases} d_{|k|} & k \text{ odd}, \\ 0 & k \text{ even}, \end{cases} \tag{2.43}$$

where c_k $(k = 0, 2, 4, \dots)$ and d_k $(k = 1, 3, 5, \dots)$ are real numbers.

If at some time $T/4$ the solution with initial conditions (2.43) evolves to a state in which $\varphi = 0$, a time-reversal argument shows that the solution will evolve back to the initial state at $T/2$ with the sign of φ and therefore the sign of the velocity reversed, that is, the condition $\varphi(x, T/4) = 0$ implies that $\eta(x, T/2) = \eta(x, 0)$ and $\varphi(x, T/2) = -\varphi(x, 0)$. Then if we can show that the symmetry upon translation by π at $t = 0$ implies that $\eta(x + \pi, T/2 + t) = \eta(x, t), \varphi(x + \pi, T/2 + t) = \varphi(x, t)$, we have $\eta(x, T) = \eta(x - \pi, T/2) = \eta(x - \pi, 0) = \eta(x, 0)$ and $\varphi(x, T) = \varphi(x - \pi, T/2) = -\varphi(x - \pi, 0) = \varphi(x, 0)$, that is, the solution is periodic with period T. As a matter of fact, $q_1 = (\eta_1, \varphi_1)$ defined by $\eta_1(x,t) = \eta(x + \pi, T/2 + t)$,

$\varphi_1(x,t) = \varphi(x+\pi, T/2+t)$ is also a solution of (2.9). In addition, the initial conditions satisfy

$$\eta_1(x,0) = \eta(x+\pi, T/2) = \eta(x+\pi, 0) = \eta(x,0),$$
$$\varphi_1(x,0) = \varphi(x+\pi, T/2) = -\varphi(x+\pi, 0) = \varphi(x,0).$$

Therefore, q_1 is equal to q. The conclusion follows. The symmetry upon translation by π ensures that η and φ evolve as a mirror image (about $x = \frac{\pi}{2}$ or $x = \frac{3\pi}{2}$) from $T/2$ to T as they did from 0 to $T/2$, ending in the original state.

We can improve our numerical methods in Section 2.3.2 when computing breather solutions based on the above analysis. We look for c_k, d_k and T such that the solution of the Cauchy problem (2.9) satisfies $\varphi(\cdot, T/4) = 0$. Rather than defining G as in (2.28), we define

$$G(q_0, T) = \frac{1}{2\pi}\int_0^{2\pi} \frac{1}{2}\varphi(x,T)^2\,dx, \tag{2.44}$$

where $q = (\eta, \varphi)$ solves (2.9) with initial conditions $q(\cdot, 0) = q_0$, and T is 1/4 of the period, which is our convention in this section to compute breather solutions only.

It remains to explain how to compute ∇G in this case. The T-derivative is simply

$$\frac{\partial G}{\partial T} = \frac{1}{2\pi}\int_0^{2\pi} \varphi(x,T)\varphi_t(x,T)\,dx \tag{2.45}$$

We define $\dot{q}$ and $\tilde{q}$ just as before, with $\dot{q}$ solving the linearized equation (2.32) and $\tilde{q}$ solveing the adjoint equation (2.35) backward in time, which ensures that $\langle \tilde{q}(\cdot, T-t), \dot{q}(\cdot, t)\rangle$ is constant. But we must change the initial condition of the adjoint problem to fit the new functional G. Specifically, to put

$$\dot{G} = \left.\frac{d}{d\epsilon}\right|_{\epsilon=0} G(q_0 + \epsilon\dot{q}_0, T) = \frac{1}{2\pi}\int_0^{2\pi} \varphi(x,T)\dot{\varphi}(x,T)\,dx \tag{2.46}$$

in the desired form

$$\dot{G} = \big\langle \tilde{q}_0, \dot{q}(\cdot,T)\big\rangle = \big\langle \underbrace{\tilde{q}(\cdot,T)}_{\delta G/\delta q_0}, \dot{q}_0\big\rangle, \tag{2.47}$$

we set the initial conditions of the adjoint equation to be

$$\tilde{\eta}_0(x) = 0, \qquad \tilde{\varphi}_0(x) = \varphi(x,T). \tag{2.48}$$

We only need to solve the adjoint problem once to obtain ∇G.

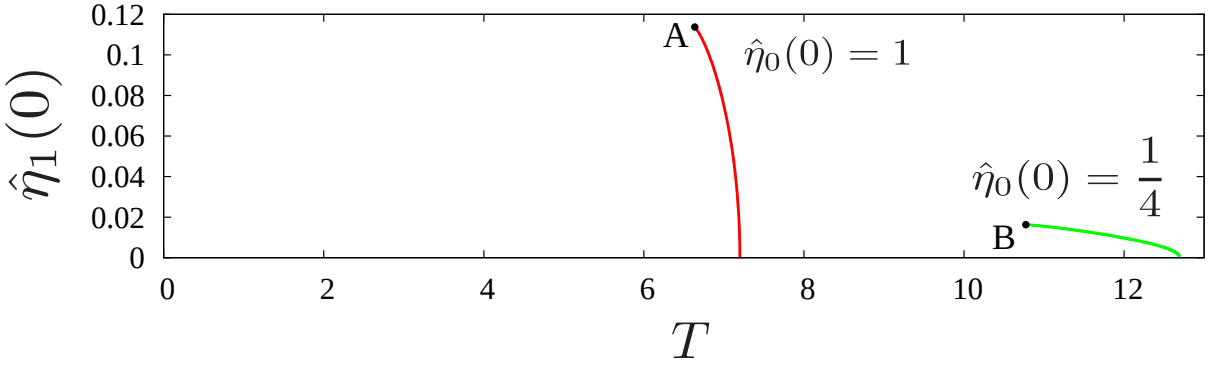

Figure 2.1: Bifurcation from the flat state to two families of traveling waves, using $\hat{\eta}_1(0)$ as the bifurcation parameter.

2.4 Numerical Results

2.4.1 Traveling Waves

Although other methods exist for computing traveling waves [2, 3, 19, 20, 59, 61], they serve as a useful initial test case for our algorithm. We use a sinusoidal solution of the linearized problem [78] for the initial guess for our optimal control algorithm to find a solution of the nonlinear problem near the flat rest state. We then use linear extrapolation (or the result of the previous iteration) for the initial guess in subsequent calculations as we vary the bifurcation parameter.

In Figure 2.1, we show the result of two families of traveling waves with the mean of η equal to 1 and 0.25 when varying $\hat{\eta}_1(0)$ from 0 (the flat state). The surface tension is set to be 0. In Figure 2.2, we give a snapshot of the solutions with largest amplitude along each path. They are labeled A and B on the bifurcation diagram.

2.4.2 Symmetric Breathers

We show the bifurcation of varying $\hat{\varphi}_k(0)$ for $k = 1, 9, 15, 21, 23, 37$ of a family of symmetric breathers with $\hat{\eta}_0(0) = 1$ without surface tension in Figure 2.3. We use a small amplitude starting guess built by hand by trial and error on the breathers. Higher Fourier modes are used to reveal a complete picture of this family of time-periodic solutions. When we plot the kth Fourier mode of the initial free surface potential with $k = 9, 15, 21, 23, 37$, the graph consists of two disconnected branches. (However, the disconnection near $T = 7.237$ in $\hat{\varphi}_{37}(0)$ is due to mesh refinement.) In Figure 2.4, we plot time-elapsed snapshots over a quarter-period of the solutions labeled C and D in Figure 2.3. The bifurcation diagram contains 697 time-periodic solutions, each computed down to G ranging from 10^{-27} to 10^{-30}, with the

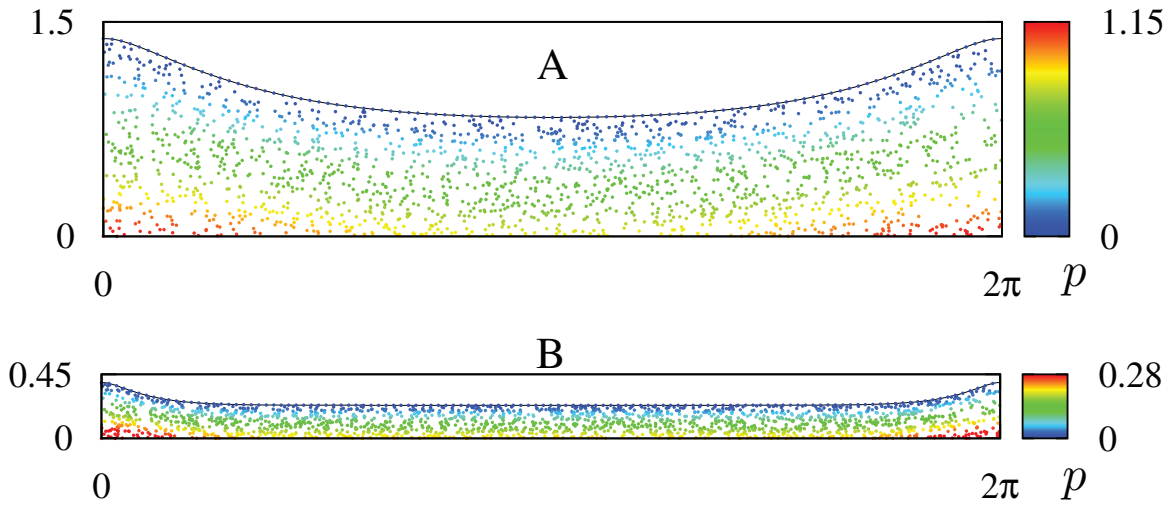

Figure 2.2: snapshot of traveling waves labeled A and B in Figure 2.1.

number of Fourier modes, M, ranging from 64 to 512. G grows as large as 2×10^{-23} as we approached point C in Figure 2.3 before refining the mesh from 256 to 512 Fourier modes. It took a week running in parallel with OpenMP using 16 threads on an 8 core 2.93 GHz Intel Nehalem architecture to compute this family of solutions.

We also show the bifurcation diagrams of two other families of symmetric breathers without surface tension with the mean of η equal to 0.25 and 0.05 in Figure 2.5 and Figure 2.7. Each of the graphs consists of two disconnected branches. Some interesting solutions from these two families are plotted in Figure 2.6 and Figure 2.8. Most visibly in Figure 2.8, two small localized waves form in the solution. They collide at $T/4$, and recover their exact shapes after collision. Small amplitude, non-localized, higher frequency standing waves are also visible in the solution of Figure 2.8.

To see the effect of surface tension on the water waves when gravity and surface tension are of similar magnitude, we give time-elapsed snapshots over a quarter-period of breathers with $\tau = 1$ in Figure 2.9. When surface tension is present, we do not observe localized waves. Instead, we find large amplitude sloshing modes spread over the whole domain.

2.5 Future Work

Concerning future work, there are many interesting problems to investigate. First, we see in Figure 2.3 that the bifurcation digram of the ninth Fourier mode reveals a disconnection not visible in that of the first mode. Similar behavior is seen in Figures 2.5 and 2.7. For the vortex sheet with surface tension [6], it was found that additional disconnections appear

on finer and finer scales for higher and higher Fourier modes. It was conjectured that time-periodic solutions of the vortex sheet do not occur in smooth families, but are parametrized by totally disconnected Cantor sets. Further investigation is needed to determine if the same situation occurs for the water wave. Next, since traveling and steady rotational water waves have been studied in [18] and [48], we hope to extend our numerical methods to compute time-periodic water waves with vorticity. This would require adopting a vorticity-stream formulation of the equations of motion rather than assuming potential flow. It would also be worthwhile to extend our work to three dimensions. The double layer potential in (2.12) is valid in the 3D case with $N(\boldsymbol{x}, \boldsymbol{\zeta}) = \frac{1}{4\pi|\boldsymbol{x}-\boldsymbol{\zeta}|}$. But it is not possible to use tools in complex analysis to sum over periodic images in the same way as in 2D. One option is to use finite element methods to compute the Dirichlet to Neumann map. Even within 2D, it would be interesting to explore the effect of a non-flat bottom boundary, or to search for time-periodic solutions that overturn, using the angle-arclength formulation developed by Hou, Lowengrub and Shelley for the vortex sheet [37, 38] and extended to the water wave case by Ambrose and Wilkening [5]. Finally, much is known about the stability of traveling water waves [3, 8, 9, 10, 24, 61, 60, 81]. We hope to extend this work to determine if stable, genuinely time-periodic solutions of the water wave problem exist.

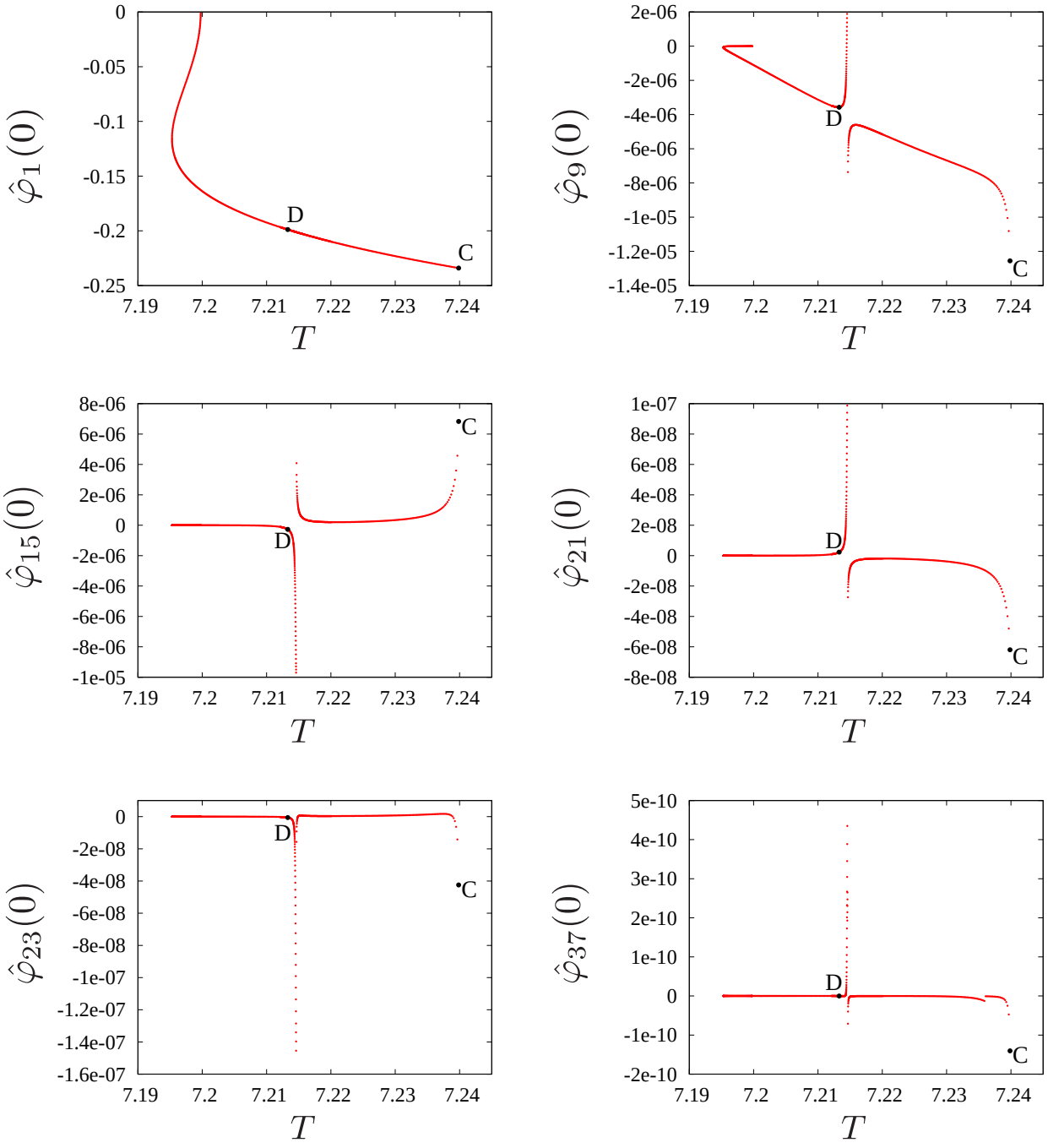

Figure 2.3: Bifurcation diagrams of a family of symmetric breathers with the mean of η equal to 1 without surface tension, using d_1, d_9, d_{15}, d_{21}, d_{23} and d_{37} ($d_k = \hat{\varphi}_k(0)$) as the bifurcation parameter respectively.

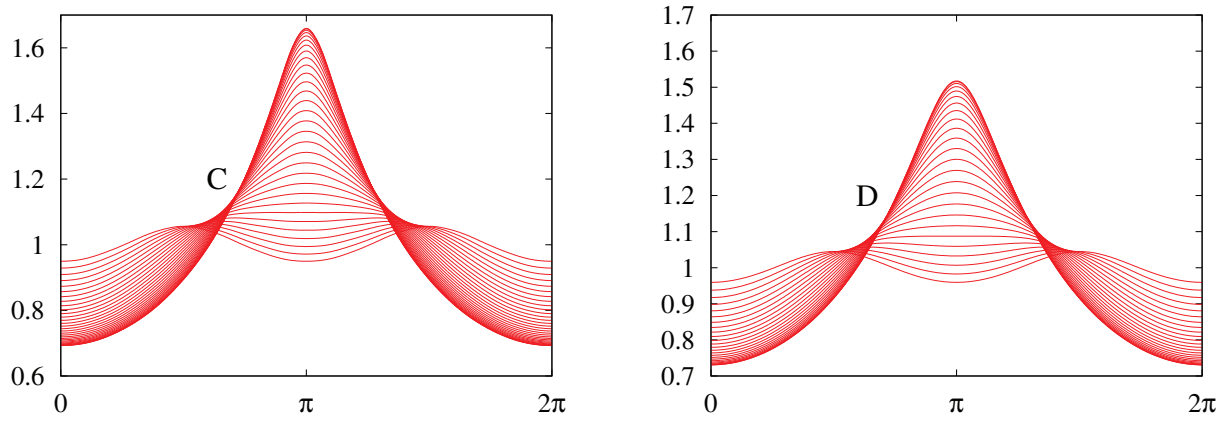

Figure 2.4: Time-elapsed snapshots over a quarter-period of the solutions labeled C and D in Figure 2.3.

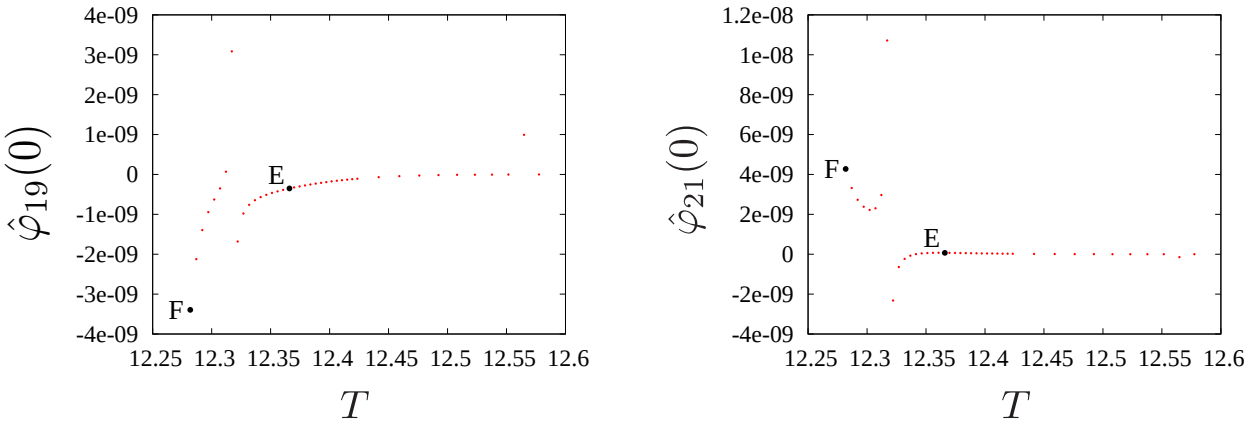

Figure 2.5: Bifurcation diagrams of a family of symmetric breathers with the mean of η equal to 0.25 without surface tension, using d_{19}, d_{21} as the bifurcation parameter respectively.

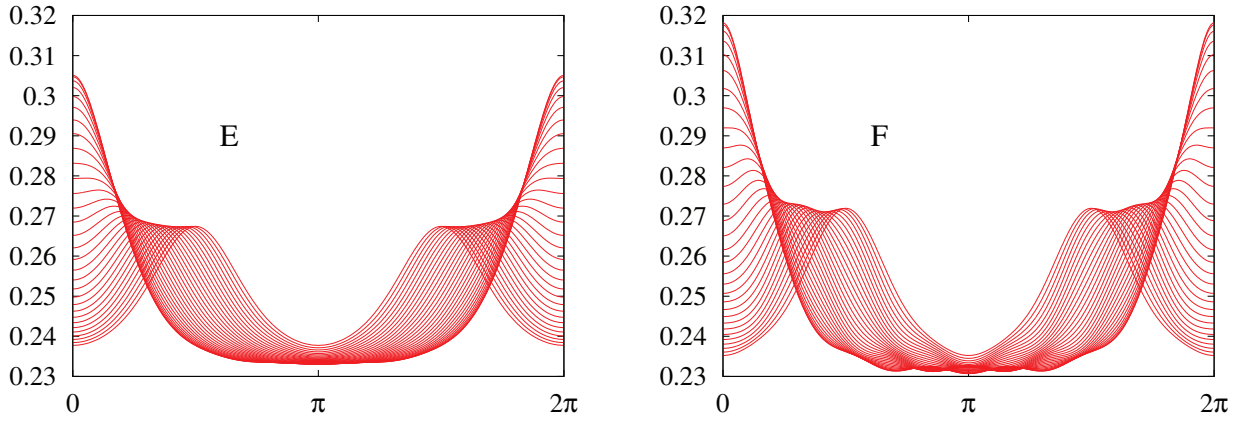

Figure 2.6: Time-elapsed snapshots over a quarter-period of the solutions labeled E and F in Figure 2.5.

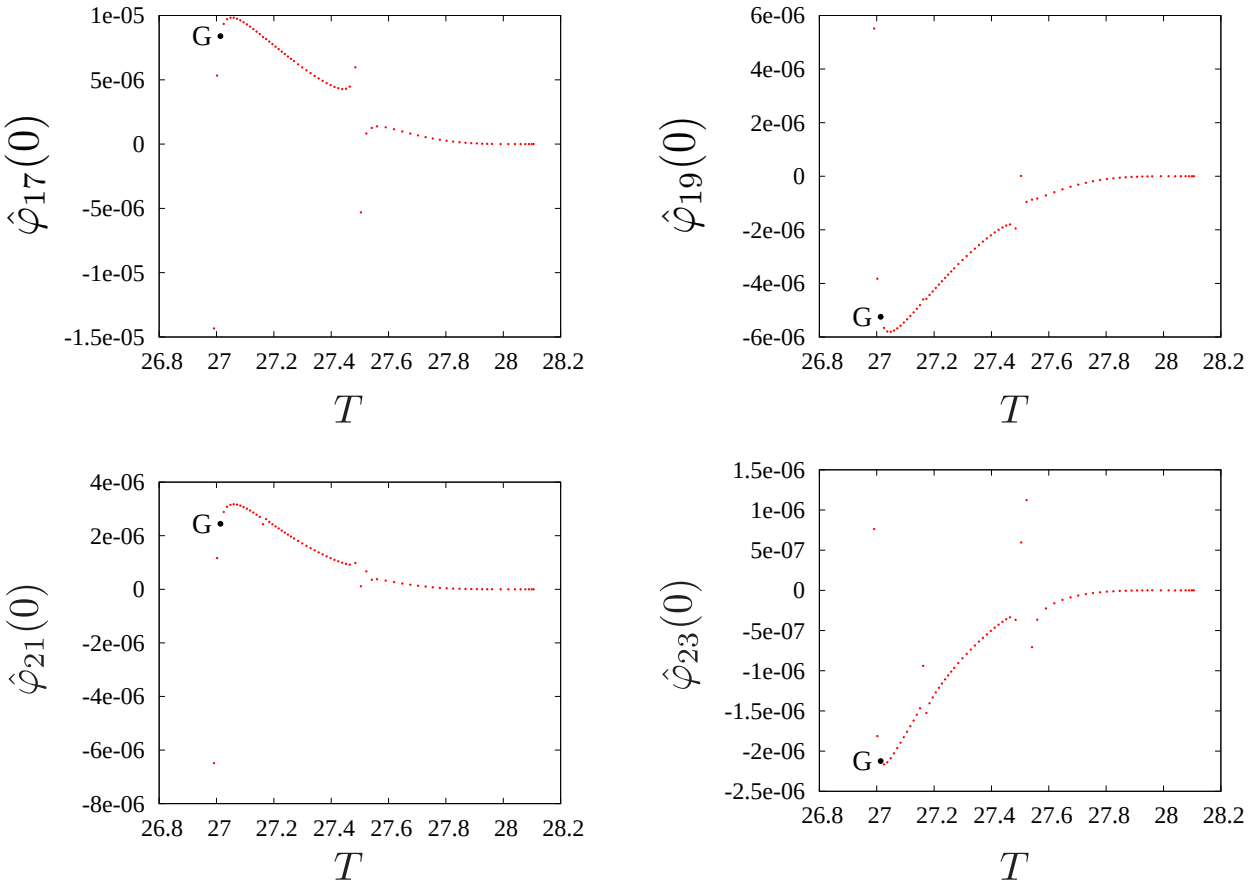

Figure 2.7: Bifurcation diagrams of a family of symmetric breathers with the mean of η equal to 0.05 without surface tension, using d_{17}, d_{19}, d_{21}, d_{23} as the bifurcation parameter respectively.

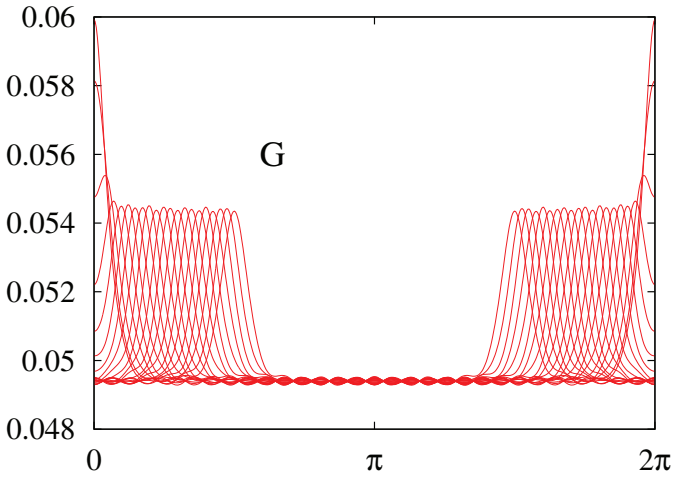

Figure 2.8: Time-elapsed snapshots over a quarter-period of the solution labeled G in Figure 2.7.

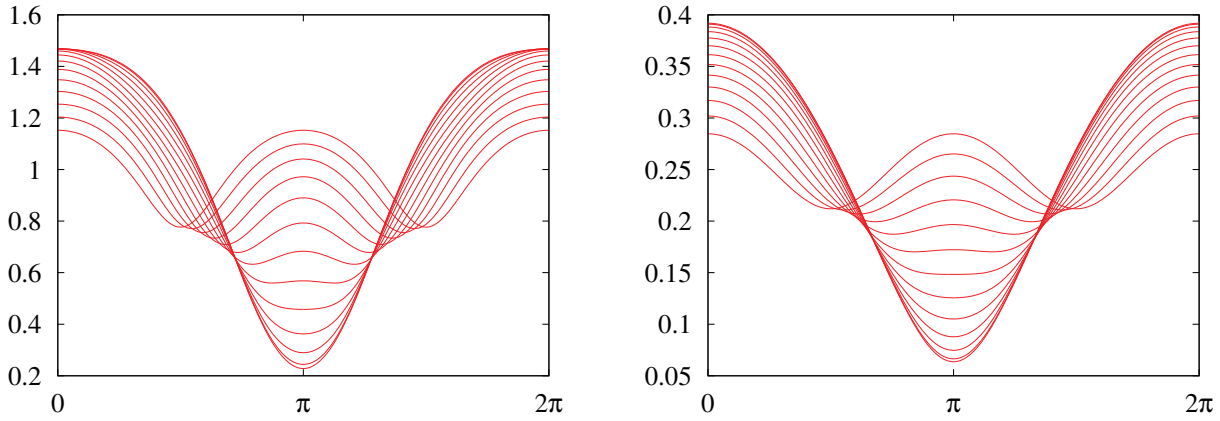

Figure 2.9: Time-elapsed snapshots over a quarter-period of breathers with surface tension ($\tau = 1$), where the mean of η is equal to 1 (left) and 0.25 (right).

Appendix A

Alternative Version of Algorithm 2 for Computing Local Smith Forms

In this section we present an algebraic framework for local Smith forms of matrix polynomials that shows the connection between Algorithm 2 and the construction of canonical systems of Jordan chains presented in [79]. This leads to a variant of the algorithm in which row-reduction is done in the field K rather than in R/pR.

Suppose R is a principal ideal domain and p is a prime in R. M defined via $M = R^n$ is a free R-module with a free basis $\{(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)\}$. Suppose $A : M \to M$ is a R-module morphism. We define submodules

$$N_k = \left\{x \in M : Ax \text{ is divisible by } p^{k+1}\right\}, \qquad (k \geq -1). \tag{A.1}$$

Then N_k is a free submodule of M by the structure theorem [39] for finitely generated modules over a principal ideal domain, which states that if M is a free module over a principal ideal domain R, then every submodule of M is free. The rank of N_k is also n, as $p^{k+1}M \subset N_k \subset M$. Note that $N_{-1} = M$ and

$$N_k \subset N_{k-1}, \qquad (k \geq 0), \tag{A.2}$$

$$N_k \cap pM = pN_{k-1}, \qquad (k \geq 0). \tag{A.3}$$

Next we define the spaces W_k via

$$W_k = N_k/pN_{k-1}, \qquad (k \geq -1), \tag{A.4}$$

where $N_{-2} := M$ so that $W_{-1} = M/pM$. By (A.3), the action of R/pR on W_k is well-defined, i.e. W_k is a vector space over this field. Let us denote the canonical projection $M \to M/pM$ by π. Note that $\pi(pN_{k-1}) = 0$, so π is well-defined from W_k to M/pM for $k \geq -1$. It is also injective as $xp \in N_k \Rightarrow x \in N_{k-1}$, by (A.3). Thus, cosets $\{\dot{x}_1, \ldots, \dot{x}_m\}$ are linearly independent in W_k iff $\{\pi(x_1), \ldots, \pi(x_m)\}$ are linearly independent in M/pM. We define the integers

$$r_k = \text{dimension of } W_k \text{ over } R/pR, \qquad (k \geq -1) \tag{A.5}$$

and note that $r_{-1} = n$. We also observe that the truncation operator

$$id : W_{k+1} \to W_k : (x + pN_k) \mapsto (x + pN_{k-1}), \qquad (k \geq -1) \tag{A.6}$$

is well-defined ($pN_k \subset pN_{k-1}$) and injective ($x \in N_{k+1}$ and $x \in pN_{k-1} \Rightarrow x \in pN_k$, due to (A.3)). We may therefore consider W_{k+1} to be a subspace of W_k for $k \geq -1$, and have the inequalities

$$r_{k+1} \leq r_k, \qquad (k \geq -1). \tag{A.7}$$

The case $r_0 = 0$ is not interesting (as $N_k = p^{k+1}M$ for $k \geq -1$), so we assume that $r_0 > 0$. When $R = K[\lambda]$, which we assume from now on, this is equivalent to assuming that $\det[A(\lambda)]$ is divisible by $p(\lambda)$. We also assume that r_k eventually decreases to zero, say

$$r_k = 0 \quad \Leftrightarrow \quad k \geq \beta, \qquad \beta := \text{maximal Jordan chain length.} \tag{A.8}$$

This is equivalent to assuming $\det[A(\lambda)]$ is not identically zero.

The *while* loop of the algorithm in Algorithm 1 is a systematic procedure for computing a basis

$$\{x_j + pN_{k-2}\}_{j=n-r_{k-1}+1}^{n-r_k} \qquad (k \geq 0) \tag{A.9}$$

for a complement $\widetilde{W}_{k-1}$ of W_k in W_{k-1}:

$$\widetilde{W}_{k-1} \oplus W_k = W_{k-1}. \tag{A.10}$$

Any basis for any complement $\widetilde{W}_{k-1}$ would also lead to a local Smith form; however, the one we chose is particularly easy to compute and has the added benefit of yielding a unimodular multiplier V. The interpretation of the r_k as dimensions of the spaces W_k shows that the integers

$$\alpha_j = k, \qquad (n - r_{k-1} < j \leq n - r_k) \tag{A.11}$$

in the local Smith form are independent of the choices of complements $\widetilde{W}_{k-1}$ and bases (A.9) for $\widetilde{W}_{k-1}$. Using induction on k, it is easy to show that for any such choices, the vectors

$$\{\text{quo}(Ax_j, p^{\alpha_j}) + pM\}_{j=1}^{n-r_k} \tag{A.12}$$

are linearly independent in M/pM; otherwise, a linear combination of the form $\star$ in Algorithm 1 can be found which belongs to $\widetilde{W}_{k-1} \cap W_k$, a contradiction.

We now wish to find a convenient representation for these spaces suitable for computation. Since $p^{k+1}M \subset pN_{k-1}$, we have the isomorphism

$$N_k/pN_{k-1} \cong (N_k/p^{k+1}M)/(pN_{k-1}/p^{k+1}M), \tag{A.13}$$

i.e.

$$W_k \cong \mathbb{W}_k/p\mathbb{W}_{k-1}, \quad (k \geq 0), \qquad \mathbb{W}_k := N_k/p^{k+1}M, \quad (k \geq -1). \tag{A.14}$$

Although the quotient $\mathbb{W}_k/p\mathbb{W}_{k-1}$ is a vector space over R/pR, the spaces $\mathbb{W}_k$ and $M/p^{k+1}M$ are only modules over $R/p^{k+1}R$. They are, however, vector spaces over K, which is useful for developing a variant of Algorithm 2 in which row reduction is done over K instead of R/pR. Treating $M/p^{k+1}M$ as a vector space over K, $A(\lambda)$ induces a linear operator $\mathbb{A}_k$ on $M/p^{k+1}M$ with kernel

$$\mathbb{W}_k = \ker \mathbb{A}_k, \qquad (k \geq -1). \tag{A.15}$$

We also define

$$R_k = \frac{\text{dimension of } \mathbb{W}_k \text{ over } K}{s}, \qquad (k \geq -1, \quad s = \deg p) \tag{A.16}$$

so that $R_{-1} = 0$ and

$$R_k = r_0 + \cdots + r_k, \qquad (k \geq 0), \tag{A.17}$$

where we used $\mathbb{W}_0 = W_0$ together with (A.14) and the fact that as a vector space over K, $\dim W_k = sr_k$. By (A.11), $r_{k-1} - r_k = \#\{j \ : \ \alpha_j = k\}$, so

$$\begin{aligned} R_{\beta-1} &= r_0 + \cdots + r_{s-1} = (r_{-1} - r_0)0 + (r_0 - r_1)1 + \cdots + (r_{\beta-1} - r_\beta)\beta \\ &= \alpha_1 + \cdots + \alpha_n = \mu = \text{algebraic multiplicity of } p, \end{aligned} \tag{A.18}$$

where we used Theorem 7 in the last step. We also note that $\nu := r_0 = R_0 = s^{-1} \dim \ker(\mathbb{A}_0)$ can be interpreted as the geometric multiplicity of p.

Equations (A.14) and (A.15) reduce the problem of computing Jordan chains to that of finding kernels of the linear operators $\mathbb{A}_k$ over K. If we use the vector space isomorphism $\Lambda : K^{sn(k+1)} \to M/p^{k+1}M$ given by

$$\Lambda(x^{(0)}; \ldots; x^{(k)}) = \gamma(x^{(0)}) + p\gamma(x^{(1)}) + \cdots + p^k\gamma(x^{(k)}) + p^{k+1}M,$$

$$\gamma(x^{(j,1,0)}; x^{(j,1,1)}; \ldots; x^{(j,n,s-1)}) = \begin{pmatrix} x^{(j,1,0)} + \lambda x^{(j,1,1)} + \cdots + \lambda^{s-1}x^{(j,1,s-1)} \\ \cdots \\ x^{(j,n,0)} + \lambda x^{(j,n,1)} + \cdots + \lambda^{s-1}x^{(j,n,s-1)} \end{pmatrix}$$

to represent elements of $M/p^{k+1}M$, then multiplication by λ in $M/p^{k+1}M$ viewed as a module becomes the following linear operator on $K^{sn(k+1)}$ viewed as a vector space over K:

$$\mathbb{S}_k = \begin{pmatrix} I \otimes S & 0 & 0 & 0 \\ I \otimes Z & I \otimes S & 0 & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & I \otimes Z & I \otimes S \end{pmatrix}, \quad S \text{ as in (1.12)}, \quad Z = \begin{pmatrix} 0 & 0 & 1 \\ & \ddots & 0 \\ 0 & & 0 \end{pmatrix}. \tag{A.19}$$

Here $\mathbb{S}_k$ is a $(k+1) \times (k+1)$ block matrix, $I \otimes S$ is a Kronecker product of matrices, S and Z are $s \times s$ matrices, and I is $n \times n$. Multiplication by λ^m is represented by $\mathbb{S}_k^m$, which has a similar block structure to $\mathbb{S}_k$, but with S replaced by S^m and Z replaced by

$$T_m = \begin{cases} 0, & m = 0, \\ \sum_{l=0}^{m-1} S^l Z S^{m-1-l}, & 1 \leq m \leq s-1. \end{cases} \tag{A.20}$$

Thus, if we expand

$$A(\lambda) = A^{(0)} + pA^{(1)} + \cdots + p^q A^{(q)}, \qquad A^{(j)} = A^{(j,0)} + \cdots + \lambda^{s-1} A^{(j,s-1)}, \tag{A.21}$$

the matrix $\mathbb{A}_k$ representing $A(\lambda)$ is given by

$$\mathbb{A}_k = \begin{pmatrix} A_0 & 0 & \cdots & 0 \\ A_1 & A_0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_k & A_{k-1} & \cdots & A_0 \end{pmatrix}, \tag{A.22}$$

where

$$A_j = \begin{cases} \sum_{m=0}^{s-1} A^{(0,m)} \otimes S^m, & j = 0, \\ \sum_{m=0}^{s-1} \left[A^{(j,m)} \otimes S^m + A^{(j-1,m)} \otimes T_m \right], & j \geq 1. \end{cases} \tag{A.23}$$

The terms $\sum_m A^{(j,m)} \otimes S^m$ and $\sum_m A^{(j-1,m)} \otimes T_m$ compute the remainder and quotient in (1.31) if we set $y = \mathbb{A}_k \cdot (x^{(0)}; \ldots; x^{(k)})$ and compare $y^{(k)}$ to the formula for $y_i = \mathrm{rem}(\mathrm{quo}(Ax_i, p^k), p)$ in (1.31).

Next we seek an efficient method of computing a basis matrix $\mathbb{X}_k$ for the nullspace $\mathbb{W}_k = \ker \mathbb{A}_k$. Note that we are now working over the field K rather than R/pR as we did in Section 1.3. Suppose $k \geq 1$ and we have computed $\mathbb{X}_{k-1}$. The first k blocks of equations in $\mathbb{A}_k \mathbb{X}_k = 0$ imply there are matrices $\mathbb{U}_k$ and $\mathbb{Y}_k$ such that $\mathbb{X}_k = [\mathbb{X}_{k-1}\mathbb{U}_k; \mathbb{Y}_k]$, while the last block of equations is

$$\overbrace{\begin{pmatrix} A_k & \ldots & A_0 \end{pmatrix}}^{\mathcal{A}_k} \underbrace{\begin{pmatrix} 0 & \mathbb{X}_{k-1} \\ I_{sn \times sn} & 0 \end{pmatrix} \begin{pmatrix} \mathbb{Y}_k \\ \mathbb{U}_k \end{pmatrix}}_{\mathbb{X}_k} = \begin{pmatrix} 0_{sn \times sR_k} \end{pmatrix}. \tag{A.24}$$

Thus, we can build up the matrices $\mathbb{X}_k$ recursively via $\mathbb{X}_0 = \mathrm{null}(A_0)$ and

$$\mathcal{A}_k = \begin{pmatrix} A_0 \ , \ [A_k, \ldots, A_1]\mathbb{X}_{k-1} \end{pmatrix}, \qquad [\mathbb{Y}_k; \mathbb{U}_k] = \mathrm{null}(\mathcal{A}_k), \qquad \mathbb{X}_k = [\mathbb{X}_{k-1}\mathbb{U}_k; \mathbb{Y}_k].$$

Here $\mathrm{null}(\cdot)$ is the standard algorithm for computing a basis for the nullspace of a matrix by reducing it to row-echelon. As the first sn columns of $\mathcal{A}_1$ coincide with $\mathcal{A}_0 := A_0$, and since $\mathbb{X}_0 = \mathrm{null}(\mathcal{A}_0)$, there are matrices Y_1 and U_1 such that

$$\begin{pmatrix} \mathbb{Y}_1 \\ \mathbb{U}_1 \end{pmatrix} = \begin{pmatrix} \mathbb{X}_0 & Y_1 \\ 0 & U_1 \end{pmatrix}, \qquad \mathbb{X}_1 = \left[\iota(\mathbb{X}_0), X_1 \right], \qquad X_1 = [\mathbb{X}_0 U_1; Y_1], \tag{A.25}$$

where $\iota : K^{snl} \to K^{sn(l+1)}$ represents multiplication by p from $M/p^l M$ to $M/p^{l+1}M$:

$$\iota(x^{(0)}; \ldots; x^{(l-1)}) = (0; x^{(0)}; \ldots; x^{(l-1)}), \qquad x^{(j)}, 0 \in K^{sn}. \tag{A.26}$$

But now the first $s(n+R_0)$ columns of $\mathcal{A}_2$ coincide with $\mathcal{A}_1$, so there are matrices Y_2 and U_2 such that

$$\begin{pmatrix}\mathbb{Y}_2\\ \mathbb{U}_2\end{pmatrix} = \begin{pmatrix}\mathbb{Y}_1 & Y_2\\ [\mathbb{U}_1;0] & U_2\end{pmatrix}, \qquad \mathbb{X}_2 = \big[\iota(\mathbb{X}_1), X_2\big], \qquad X_2 = [\mathbb{X}_1 U_2; Y_2]. \tag{A.27}$$

Continuing in this fashion, we find that the first $s(n+R_{k-2})$ columns of $\mathcal{A}_k$ coincide with $\mathcal{A}_{k-1}$, and therefore $\mathbb{Y}_k$, $\mathbb{U}_k$ and $\mathbb{X}_k$ have the form

$$\begin{aligned}\begin{pmatrix}\mathbb{Y}_k\\ \mathbb{U}_k\end{pmatrix} &= \begin{pmatrix}X_0 & Y_1 & \cdots & Y_{k-1} & Y_k\\ 0 & [U_1;0] & \cdots & [U_{k-1};0] & U_k\end{pmatrix}, && X_0 = \mathbb{X}_0,\\ \mathbb{X}_k &= \left[\begin{pmatrix}0_{snk\times sr_0}\\ X_0\end{pmatrix}, \begin{pmatrix}0_{sn(k-1)\times sr_1}\\ X_1\end{pmatrix}, \ldots, \begin{pmatrix}\\ X_k\end{pmatrix}\right], && X_k = [\mathbb{X}_{k-1}U_k; Y_k].\end{aligned} \tag{A.28}$$

By construction, $\mathbb{X}_k = [\iota(\mathbb{X}_{k-1}), X_k]$ is a basis for $\mathbb{W}_k$ when $k \geq 1$; it follows that $X_k + \iota(\mathbb{W}_{k-1})$ is a basis for W_k when W_k is viewed as a vector space over K. We define $X_0 = \mathbb{X}_0$ and $X_{-1} = I_{sn\times sn}$ to obtain bases for W_0 and W_{-1} as well.

But we actually want a basis for W_k viewed as a vector space over R/pR rather than K. Fortunately, all the matrices in this construction are manipulated in $s\times s$ blocks; everywhere we have an entry in R/pR in the construction of Section 1.3, we now have an $s\times s$ block with entries in K. This is because the matrix $\mathbb{A}_k$ commutes with $\mathbb{S}_k$ (since $A(\lambda)$ commutes with multiplication by λ). So if we find a vector $x \in \ker \mathbb{A}_k$, the vectors

$$\{x\,,\, \mathbb{S}_k x\,,\, \ldots\,,\, \mathbb{S}_k^{s-1} x\} \tag{A.29}$$

will also belong to $\ker \mathbb{A}_k$. These vectors are guaranteed to be linearly independent, for otherwise the vectors

$$\{x^{(j,i)}\,,\, Sx^{(j,i)}\,,\, \ldots\,,\, S^{s-1}x^{(j,i)}\} \tag{A.30}$$

would be linearly dependent in K^s, where $x^{(j,i,m)}$ is the first non-zero entry of x. This is impossible since p is irreducible; see (1.11) above. As a consequence, in the row-reduction processes, if we partition $\mathcal{A}_k$ into groups of s columns, either all s columns will be linearly dependent on their predecessors, or each of them will start a new row in $\mathrm{rref}(\mathcal{A}_k)$; together they contribute a block identity matrix $I_{s\times s}$ to $\mathrm{rref}(\mathcal{A}_k)$. It follows that the block nullspace matrices $[\mathbb{Y}_k; \mathbb{U}_k]$ will have supercolumns (groups of s columns) that terminate in identity matrices $I_{s\times s}$, and that the submatrix $X_k^{(0)}$ consisting of the first ns rows of X_k also has supercolumns that terminate with $I_{s\times s}$. The structure is entirely analogous to the one in Figure 1.3 above, with $s\times s$ blocks replacing the entries of the matrices shown. The following four conditions then ensure that successive columns in a supercolumn of X_k are related to each other via (A.29): (1) the terminal identity blocks in $X_k^{(0)}$ contain the only non-zero entries of their respective rows; (2) if x is a column of X_k and $x^{(0,i)} \in K^s$ is one of the columns of a terminal identity block, the $(\mathbb{S}_k x)^{(0,i)} = Sx^{(0,i)}$ is the next column of $I_{s\times s}$ (with the 1 shifted down a slot). (3) the columns of $\mathbb{X}_k$ are a basis for $\ker \mathbb{A}_k$; (4) if x is a column

of $\mathbb{X}_k$, then $\mathbb{S}_k x$ also belongs to $\ker \mathbb{A}_k$. Thus, to extract a basis for W_k over R/pR, we simply extract the first column of each supercolumn of X_k. The result is identical to that of Algorithm 2. In practice, this version of the algorithm (working over K) is easier to implement, but the other version (over R/pR) should be about s times faster as the cost of multiplying two elements of R/pR is $O(s^2)$ while the cost of multiplying two $s \times s$ matrices is $O(s^3)$. The results in Figure 1.8 to 1.10 were computed as described in this appendix (over $K = \mathbb{Q}$).

Example 9. *Define the matrix polynomial $A(\lambda)$ as in Example 1. We use the algorithm presented in this appendix to find a local Smith form of $A(\lambda)$ at $p_2(\lambda) = \lambda^2 + 2$. First, we expand*

$$\begin{aligned} A(\lambda) =& p_2 \left[\begin{pmatrix} -2 & -1 \\ -2 & -1 \end{pmatrix} + \lambda \begin{pmatrix} 0 & -1 \\ 2 & 4 \end{pmatrix} \right] + p_2^2 \left[\begin{pmatrix} 2 & 6 \\ 0 & 2 \end{pmatrix} + \lambda \begin{pmatrix} 0 & -1 \\ 0 & -3 \end{pmatrix} \right] \\ &+ p_2^3 \left[\begin{pmatrix} 0 & -4 \\ 0 & -1 \end{pmatrix} + \lambda \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right] + p_2^4 \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. \end{aligned}$$

We have the matrix $\mathbb{A}_0 = A_0 = 0_{4\times 4}$ and a basis matrix $\mathbb{X}_0 = I_4$ for the null space $\mathbb{W}_0 = \ker \mathbb{A}_0$. In practice, the while *loop terminates here so that we obtain two jordan chains of length 1, because we already have $R_0 = \mu = 2$. To show how the algorithm works, we continue to compute*

$$\mathcal{A}_1 = (\, A_0 \,,\, A_1 \mathbb{X}_0 \,),$$

where

$$A_1 = A^{(1,0)} \otimes I_2 + A^{(0,0)} \otimes T_0 + A^{(1,1)} \otimes S + A^{(0,1)} \otimes T_1 = \begin{pmatrix} -2 & 0 & -1 & 2 \\ 0 & -2 & -1 & -1 \\ -2 & -4 & -1 & -8 \\ 2 & -2 & 4 & -1 \end{pmatrix}.$$

It follows that $\mathbb{X}_1 = \iota(\mathbb{X}_0)$, which verifies that this is the last iteration in the loop. We view $\mathbb{X}_0$ in 2×2 blocks to obtain the local Smith form as in Example 5 at $p_2(\lambda) = \lambda^2 + 2$.

Bibliography

[1] G. B. Airy. Tides and waves. *Encyclopaedia Metropolitana (1817-1845), Mixed Sciences*, 3, 1841.

[2] B. Akers. On model equations for gravity-capillary waves. PhD Thesis, University of Wisconsin - Madison, 2008.

[3] B. Akers and D. P. Nicholls. Traveling waves in deep water with gravity and surface tension. (preprint), 2009.

[4] D. M. Ambrose and J. Wilkening. Global paths of time-periodic solutions of the benjamin-ono equation connecting pairs of traveling waves. *Comm. App. Math. and Comp. Sci.*, 4/1:177–215, 2009.

[5] D. M. Ambrose and J. Wilkening. A boundary integral method for time-stepping the 2d water wave. in preparation, 2010.

[6] D. M. Ambrose and J. Wilkening. Computation of symmetric, time-periodic solutions of the vortex sheet with surface tension. *Proc. Natl. Acad. Sci.*, 2010.

[7] D. M. Ambrose and J. Wilkening. Computation of time-periodic solutions of the benjamin-ono equation. *J. Nonlinear Sci.*, 2010.

[8] T. B. Benjamin and J. E. Feir. The disintegration of wave trains on deep water. part i. theory. *J. Fluid Mech.*, 27:417430, 1967.

[9] T. B. Benjamin and K. Hasselmann. Instability of periodic wavetrains in nonlinear dispersive systems. *Proc. R. Soc. Lond. A*, 299(1456):59–76, 1967.

[10] N. Bottman and B. Deconinck. Kdv cnoidal waves are spectrally stable. *Discrete and Continuous Dynamical Systems*, 25(4):11631180, 2009.

[11] J. V. Boussinesq. Théorie de l'intumescence liquide appelée onde solitaire ou de translation, se propageant dans un canal rectangulaire. *C. R. Acad. Sci. Paris*, 72:755–759, 1871.

[12] M. O. Bristeau, R. Glowinski, and J. Périaux. Controllability methods for the computation of time-periodic solutions; application to scattering. *J. Comput. Phys.*, 147:265–292, 1998.

[13] M. O. Bristeau, O. Pironneau, R. Glowinsky, J. Périaux, and P. Perrier. On the numerical solution of nonlinear problems in fluid dynamics by least squares and finite element methods. i least square formulations and conjugate gradient solution of the continuous problems. *Comput. Meth. Appl. Mech. Engng.*, 17:619–657, 1979.

[14] C. G. Broyden. The convergence of a class of double-rank minimization algorithms, parts i and ii. *J. Inst. Maths. Applics.*, 6:7690, 222231, 1970.

[15] M. Cabral and R. Rosa. Chaos for a damped and forced kdv equation. *Physica D*, 192:265–278, 2004.

[16] A-L Cauchy. Mémoire sur la théorie de la propagation des ondes 'a la surface dun fluide pesant dune profondeur indéfinie. *Mém. Présentés Divers Savans Acad. R. Sci. Inst. France*, I:3–312, 1827.

[17] M. Chen and G. Iooss. Standing waves for a two-way model system for water waves. *European J. Mech. B/Fluids*, 24:113–124, 2005.

[18] A. Constantin and W. Strauss. Exact periodic traveling water waves with vorticity. *C. R. Math. Acad. Sci. PAris*, 335:797–800, 2002.

[19] W. Craig and D. P. Nicholls. Traveling two and three dimensional capillary gravity water waves. *SIAM Journal on Mathematical Analysis*, 32:323–359, 2000.

[20] W. Craig and D. P. Nicholls. Traveling gravity water waves in two and three dimensions. *Eur. Jour. Mech. B/ Fluids*, 21:615–641, 2002.

[21] A. D. D. Craik. The origins of water wave theory. *Ann. Rev. Fluid Mech.*, 36:1–28, 2004.

[22] A. D. D. Craik. George gabriel stokes on water wave theory. *Ann. Rev. Fluid Mech.*, 37:23–42, 2005.

[23] A. Crannell. The existence of many periodic non-travelling solutions to the boussinesq equation. *J. Differential Equations*, 126:169–183, 1996.

[24] B. Deconinck, D. E. Pelinovsky, and J. D. Carter. Transverse instabilities of deep-water solitary waves. *Proc. R. Soc. A*, 462(2071), 2006.

[25] N. DeDontney and J. R. Rice. Effect of splay fault rupture on open ocean tsunamis with application to the 2004 indian ocean event. submitted to Journal of Geophysical Research - Solid Earth, 2010.

[26] E. J. Doedel, H. B. Keller, and J. P. Kernévez. Numerical analysis and control of bifurcation problems: (ii) bifurcation in infinite dimensions. *Int. J. Bifurcation and Chaos*, 1:745–772, 1991.

[27] L. Euler. Continuation des recherches sur la théorie du mouvement des fluides. *Mém. Acad. Sci. Berlin*, 11:361–361, 1757.

[28] L. Euler. Principes géneraux du mouvement des fluides. *Mém. Acad. Sci. Berlin*, 11:271–315, 1757.

[29] L. Euler. Principia motus fluidorum. *Novi Commentarii Acad. Sci. Petropolitanae*, 6:271–311, 1761.

[30] G. B. Folland. *Introduction to Partial Differential Equations*. Princeton University Press, Princeton, 1995.

[31] F.R. Gantmacher. *Matrix Theory*, volume 1. Chelsea Publishing Company, 1960.

[32] F. J. von Gerstner. Theorie der wellen. *Abhand. Kön. Böhmischen Gesel. Wiss.*, 1802.

[33] I. Gohberg, M. A. Kaashoek, and F. van Schagen. On the local theory of regular analytic matrix functions. *Linear Algebra Appl.*, 182:9–25, 1993.

[34] I. Gohberg, P. Lancaster, and L. Rodman. *Matrix Polynomials*. Academic Press, New York, 1982.

[35] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 1996.

[36] E. Hairer, S. P. Norsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff Problems, 2nd Edition*. Springer, Berlin, 2000.

[37] T. Y. Hou, J. S. Lowengrub, and M. J. Shelley. Removing the stiffness from interfacial flows with surface tension. *J. Comput. Phys.*, 114:312338, 1994.

[38] T. Y. Hou, J. S. Lowengrub, and M. J. Shelley. The long-time motion of vortex sheets with surface tension. *Phys. Fluids*, 9:19331954, 1997.

[39] Thomas W. Hungerford. *Algebra*. Springer, New York, 1996.

[40] G. Iooss, P. Plotnikov, and J. Toland. Standing waves on an infinitely deep perfect fluid under gravity. *Arch. Rat. Mech. Anal.*, 177:367–478, 2005.

[41] G. Iooss, P. I. Plotnikov, and J. F. Toland. Standing waves on an infinitely deep perfect fluid under gravity. *Arch. Rat. Mech. Anal.*, 177:367–478, 2005.

[42] A. Jameson. Aerodynamic design via control theory. *J Sci. Comput.*, 3:233–260, 1988.

[43] T. Kailath. *Linear Systems*. Prentice Hall, Englewood Cliffs, 1980.

[44] E. Kaltofen, M.S. Krishnamoorthy, and B.D. Saunders. Fast parallel computation of Hermite and Smith forms of polynomial matrices. *SIAM J. Alg. Disc. Meth.*, 8(4):683–690, 1987.

[45] E. Kaltofen, M.S. Krishnamoorthy, and B.D. Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra and its Applications*, 136:189–208, 1990.

[46] R. Kannan. Solving systems of linear equations over polynomials. *Theoretical Computer Science*, 39:69–88, 1985.

[47] O. D. Kellogg. *Foundations of Potential Theory*. Dover, New York, 1954.

[48] J. Ko and W. Strauss. Large-amplitude steady rotational water waves. *Eur. J. Mech. B/Fluids*, 27:96–109, 2008.

[49] D. J. Korteweg and G. de Vries. On the change of form of long waves advancing in a rectangular canal, and on a new type of long stationary waves. *Philos. Mag.*, 39(5):422–443, 1895.

[50] B. Kress. *Linear Integral Equations*. Springer-Verlag, New York, 1989.

[51] S. Labhalla, H. Lombardi, and R. Marlin. Algorithmes de calcul de la réduction de hermite d'une matrice à coefficients polynomiaux. *Theoretical Computer Science*, 161:69–92, 1996.

[52] J-L Lagrange. Mémoire sur la théorie du mouvement des fluides. *Nouv. Mém. Acad. Berlin*, page 196, 1781.

[53] J-L Lagrange. Sur la maniére de rectifier deux entroits des principes de newton relatifs á la propagation du son et au mouvement des ondes. *Nouv. Mém. Acad. Berlin*, 1786.

[54] J-L Lagrange. *Méchanique Analitique*. la Veuve Desaint, Paris, 1788.

[55] P-S Marquis de Laplace. Suite des récherches sur plusieurs points du système du monde (xxvxxvii). *Mém. Présentés Divers Savans Acad. R. Sci. Inst. France*, pages 525–552, 1776.

[56] B. Mohammadi and O. Pironneau. *Applied Shape Optimization for Fluids*. Oxford University Press, New York, 2001.

[57] N. I. Muskhelishvili. *Singular Integral Equations, 2nd Edition*. Dover, New York, 1992.

[58] I. Newton. *Philosophiae Naturalis Principia Mathematica*. Jussu Societatis Regiae ac Typis J. Streater. Engl. transl. N Motte, London, 1687.

[59] D. P. Nicholls. Traveling water waves: Spectral continuation methods with parallel implementation. *J. Comput. Phys.*, 143:224–240, 1998.

[60] D. P. Nicholls. Boundary perturbation methods for water waves. *GAMMMitt.*, 30(1):44–74, 2007.

[61] D. P. Nicholls and F. Reitch. Stable, high-order computation of traveling water waves in three dimensions. *Eur. Jour. Mech. B/ Fluids*, 25:406–424, 2006.

[62] D.P. Nicholls. Spectral data for travelling water waves: singularities and stability. *J. Fluid Mech.*, 624:339–360, 2009.

[63] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 1999.

[64] P. Plotnikov and J. Toland. Nash-moser theory for standing water waves. *Arch. Rat. Mech. Anal.*, 159:1–83, 2001.

[65] S. D. Poisson. Mémoire sur la théorie des ondes. *Mém. Acad. R. Sci. Inst. France*, 1:70–186, 1818.

[66] B. Rayleigh. On waves. *Philos. Mag.*, 1:257–279, 1876.

[67] J. S. Russell. Report on waves. *Rep. Br. Assoc. Adv. Sci.*, pages 311–390, 1844.

[68] P.G. Saffman. *Vortex Dynamics*. Cambridge University Press, Cambridge, UK, 1995.

[69] L. F. Shampine. Some practical runge-kutta formulas. *Mathematics of Computation*, 46:135–150, 1986.

[70] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis, 3rd Edition*. Springer, New York, 2002.

[71] G. G. Stokes. On the theory of oscillatory waves. *Trans. Camb. Philos. Soc.*, 8:441–455, 1847.

[72] A. Storjohann. Computation of Hermite and Smith normal forms of matrices. Master's thesis, Dept. of Computer Science, Univ. of Waterloo, Canada, 1994.

[73] A. Storjohann and G. Labahn. A fast las vegas algorithm for computing the Smith normal form of a polynomial matrix. *Linear Algebra and its Applications*, 253:155–173, 1997.

[74] G. Villard. Computation of the Smith normal form of polynomial matrices. In *International Symposium on Symbolic and Algebraic Computation, Kiev, Ukraine*, pages 209–217. ACM Press, 1993.

[75] G. Villard. Fast parallel computation of the Smith normal form of polynomial matrices. In *International Symposium on Symbolic and Algebraic Computation, Oxford, UK*, pages 312–317. ACM Press, 1994.

[76] G. Villard. Generalized subresultants for computing the Smith normal form of polynomial patrices. *J. Symbolic Computation*, 20:269–286, 1995.

[77] G. Villard. Fast parallel algorithms for matrix reduction to normal forms. *Appli. Alg. Eng. Comm. Comp.*, 8(6):511–538, 1997.

[78] G. B. Whitham. *Linear and Nonlinear Waves*. Wiley, New York, 1974.

[79] J. Wilkening. An algorithm for computing Jordan chains and inverting analytic matrix functions. *Linear Algebra Appl.*, 427/1:6–25, 2007.

[80] J. Wilkening. Math 228 lecture notes: Numerical solution of differential equations. available from author's website, 2007.

[81] V. Zakharov. Stability of periodic waves of finite amplitude on the surface of a deep fluid. *J. Appl. Mech. Tech. Phys.*, 9:190–194, 1968.