UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea in Informatica

Computation of Kronecker's Canonical Form in a Computer Algebra System

Relatore
Prof. Federico Poloni

Candidato
Giacomo Trapani

Anno Accademico 2021/2022

Contents

List of Algorithms

Notation

I	Identity matrix
H	Upper shift matrix
O	Null matrix
A^T	Transpose of matrix A
A^*	Transpose conjugate of matrix A
A^{-1}	Inverse of matrix A
$a_{i,j}$	Element on row i, column j of matrix A
$A^{(m,n)}$	A is an $m \times n$ matrix
$A^{(m)}$	A is an $m \times m$ matrix
$det(A)$	Determinant of matrix A
$ker(A)$	Right kernel of matrix A
$\|A\|$	Euclidean norm of matrix A
$\|\mathbf{v}\|$	Euclidean norm of vector $\mathbf{v}$
$\gg$	Much greater than
$\subseteq$	Is a subset of
$sign(x)$	Function defined as 1 if $x > 0$, -1 if $x < 0$
$\dot{f}(x)$	First derivative of f with respect to x
$\ddot{f}(x)$	Second derivative of f with respect to x
$\mathbb{R}$	Set of real numbers
$\mathbb{C}$	Set of complex numbers
e	Euler's number

Introduction

This work deals with the problem of *computing precisely* **Kronecker's canonical form** for an arbitrary pencil of matrices and defines an algorithm to compute it (exactly) using the open-source computer algebra system **SageMath**[19]. We may think of Kronecker's canonical form as a generalization of Jordan's canonical form. Next, we shall try and make sense of this abstract. Let us start with an example.

Let A be a matrix defined in $\mathbb{C}^{m\times n}$ and consider the linear differential equation

$$\dot{x} + Ax = f.$$

The set of solutions of this equation is characterized by matrix A and, in particular, by Jordan's canonical form (of matrix A).

Next, with $B \in \mathbb{C}^{m\times n}$, take the equation

$$B\dot{x} + Ax = f. \tag{1}$$

At this point, we can define what a pencil of matrices is. We call the pair $(A,\, B)$ a **linear pencil of matrices** or, for the sake of brevity, a pencil of matrices [11].

To solve (1), we shall distinguish two cases.

Case 1. B is nonsingular.

The set of solutions is characterized by Jordan's normal form of

$$-B^{-1}A.$$

Case 2. B is singular.

The linear system given by (1) may contain algebraic equations or, in other words, equations which do not contain any derivatives. As an example, take the equation

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}.$$

The linear system contains an algebraic equation

$$\begin{cases} \dot{x_1} + \dot{x_3} + x_1 = f_1 \\ x_1 + x_2 + x_3 = f_2 \\ \dot{x_2} + \dot{x_3} + x_3 = f_3 \end{cases}$$

The set of solutions is characterized by **Kronecker's canonical form** of (A, B) [7, 13].

The problem of computing precisely Kronecker's canonical form is *ill-conditioned*, meaning that small changes in its input values may cause a large change in the answer. The current state of the art is to compute a **Kronecker-like form**, which can be found with numerically stable algorithms [2]. An implementation of those algorithms has been written in Julia [3] by Andreas Varga, it can be found in https://github.com/andreasvarga/MatrixPencils.jl.

Here, we take *a different approach*; but first, let us investigate whether there exists a way for a calculator to reason over an ill-conditioned problem.

As an example, take another ill-conditioned problem, i.e. the problem of computing Jordan's canonical form of an arbitrary matrix: how does SageMath calculate Jordan's canonical form? Computer algebra makes it possible for a computer to manipulate mathematical expressions involving symbols with no given value, and it is the tool for this very job: SageMath allows a user to work with symbolic expressions via the symbolic expression ring **SR**.

```
sage: A = matrix(SR, [[-11, -14, 0], [0, -4, 0], [1, 4, -1]])
sage: J = A.jordan_form()
sage: J
[-11|  0|  0]
[---+---+---]
[  0| -4|  0]
[---+---+---]
[  0|  0| -1]
sage: J.base_ring()
Symbolic Ring
```

Indeed, this work takes this kind of approach to compute Kronecker's canonical form. We define step-by-step a procedure to compute exactly

Kronecker's canonical form of an arbitrary pencil of matrices and prove its correctness. Next, we provide an implementation of such a procedure in [8] making use of SageMath's support for symbolic expressions, and a suitable test suite to prove it correctly mirrors its specification. Now, it is a pertinent fact that, while this technique is used to calculate Jordan's canonical form, *no other works in literature follow this approach for Kronecker's canonical form* (at least, there are none to our knowledge).

Last, even though the focus of this work is not on the applications of pencils of matrices, it may be relevant for a reader to know they lie in the fields of control systems [5, 14] and signal processing [16]; further examples can also be found in [13, pp. 8-11], such as electrical networks, multibody systems, chemical engineering, semidiscretized Stokes equations.

Now, we shall summarize the structure of this work.

In the first chapter, we recapitulate the basics of numerical analysis, computer algebra and linear algebra required to understand the following chapters. We also introduce SageMath and use it to show the reader consequences of the given theorems and the definitions.

In the second chapter, we give a self-contained introduction to the theory of linear pencils of matrices: we define regular and singular pencils and the conditions for equivalence (between pencils of matrices). Last, we define Kronecker's canonical form and briefly introduce linear differential-algebraic equations with constant coefficients.

In the third chapter, we prove Kronecker's theorem by defining a procedure to compute Kronecker's canonical form for an arbitrary pencil of matrices; we also provide the pseudocode for such an algorithm.

In the fourth chapter, we discuss technical details of the SageMath algorithm and give an approximation of its time complexity; we give a minimal working example and discuss the reasoning behind the way the test suite has been written.

The fifth chapter concludes this paper and outlines possible improvements for the algorithm.

Background

This chapter will serve as prerequisite knowledge throughout the rest of this thesis.

We shall briefly present SageMath, the software system used to implement the algorithm discussed in the following chapters, by introducing computer algebra systems and comparing numerical computations against computer algebra showcasing an example; then, the reader shall familiarize with the concept of condition number as an emphasis on it will be put in every part.

Subsequently, definitions and properties of eigenvalues and eigenvectors shall be concisely introduced.

Lastly, we shall describe Jordan's canonical form of a matrix.

Computer algebra.

Computers have fundamentally two ways to reason about a mathematical expression: **numerical computations**, which are performed using *only numbers* to represent values and **computer algebra** (or **symbolic computations**), which - by contrast - use *both numbers and symbols.*

First, we shall introduce the concept of **floating point number system**, which is the system used to handle numerical computations.

For further explanations, refer to Quarteroni, Saleri, Sacco [18].

Definition 2.1 (Normalized-floating point number system)**.** [18, 2.5.2] A normalized-floating point number system F is characterized by the 4-tuple of integers β, p, L, U:

- β is called base or radix,
- p precision,
- $[L, U]$ exponent range (with L, U denoting lower and upper bound respectively).

Given a number $x \in \mathbb{R}$, $x \neq 0$ its representation in a floating point number system shall be written out as $fl(x)$ and has the form

$$x = sign(x)\beta^E \sum_{i=0}^{p-1} d_i \beta^{-i}$$

with $L \leq E \leq U$ and the sequence $\{d_i\}_{i \geq 0}$ (which is called mantissa) made up of natural numbers such that $d_0 \neq 0$, $0 \leq d_i \leq \beta - 1$ and d_i eventually different from $\beta - 1$.

The notation δx shall be used to denote the difference between a symbol x and its floating point approximation $fl(x)$

$$\delta x = x - fl(x).$$

It is important to notice that a floating point number system F is discrete and finite: it approximates real numbers with finite numbers; in other words, a floating point number system may introduce errors when representing a real number.

A de facto standard for computers to work with floating point approximations is IEEE 754 [10], the details of which shall not be discussed.

Definition 2.2 (Machine epsilon)**.** [18, p. 49] Machine epsilon is the maximum possible absolute relative error in representing a nonzero real number x in a floating point number system

$$\epsilon_{mach} = \max_{x} \frac{|x - fl(x)|}{|x|}.$$

Example 2.1. Let us define the matrix (made up of both symbols and numbers) M

$$\begin{bmatrix} \sqrt{2} & 1 \\ 2 & \sqrt{2} \end{bmatrix}.$$

Consider the matrix $\tilde{M}$, having as entries the floating point approximation of those of M

$$\begin{bmatrix} fl(\sqrt{2}) & 1 \\ 2 & fl(\sqrt{2}) \end{bmatrix}.$$

Computing its determinant gives out $2 + 2\epsilon\sqrt{2} + \epsilon^2 - 2 \doteq 2 + 2\epsilon\sqrt{2} - 2 \neq 0$.

Introducing a small change (i.e. an "error") in the input argument may either cause a large or a small change in the result. Now, we shall define what condition numbers are.

Definition 2.3 (Condition number)**.** [18, p. 33] A condition number of a problem measures the sensitivity of the solution to small perturbations in the input data. Given a function f, we define

$$cond(f, x) = \lim_{\epsilon \to 0} \sup_{\|\Delta x\| \leq \epsilon \|x\|} \frac{\big\| f(x + \Delta x) - f(x) \big\|}{\epsilon \big\| f(x) \big\|}.$$

Given a problem, if its condition number is low it is said to be **well-conditioned** (typically $cond(f, x) \sim 1$), while a problem with a high condition number is (said to be) **ill-conditioned** ($cond(f, x) \gg 1$).

Let us now consider the problem of solving a linear equation subjected to a perturbation.

Let A be a non-singular matrix and assume we introduce a perturbation in the constant term $\tilde{\mathbf{b}} = \mathbf{b} + \delta\mathbf{b}$. The equation can be written as

$$A\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$$

with $\tilde{\mathbf{x}} = \mathbf{x} + \delta\mathbf{x}$. We can obtain

$$\tilde{\mathbf{x}} - \mathbf{x} = A^{-1}\tilde{\mathbf{b}} - A^{-1}\mathbf{b} = A^{-1}\delta\mathbf{b}$$

and, by using matrix norms, we can write

$$\|\tilde{\mathbf{x}} - \mathbf{x}\| = \left\|A^{-1}\delta\mathbf{b}\right\| \leq \left\|A^{-1}\right\|\|\delta\mathbf{b}\|.$$

It is also known that

$$\|\mathbf{b}\| = \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|$$

which implies

$$\|\mathbf{x}\| \geq \frac{\|\mathbf{b}\|}{\|A\|}.$$

Tying all this together we can conclude

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A\|\left\|A^{-1}\right\|\frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}.$$

Definition 2.4 (Condition number of a matrix)**.** [18, p. 36] The condition number of a non-singular matrix A is defined as

$$\kappa(A) = \left\|A^{-1}\right\|\|A\|.$$

Now, let us refocus on the topic of math expressions. Let us investigate what would happen if symbols are allowed in computations by introducing a framework that allows us to work with computer algebra.

Definition 2.5 (Computer algebra system)**.** A computer algebra system (CAS) is a mathematics software package that can perform *both symbolic and numerical mathematical computations.*

A CAS is usually a **REPL** expected to support a few functionalities [12]:

- **Arithmetic**: arithmetic over different fields with arbitrary precision.
- **Linear algebra**: matrix algebra and knowledge of different operations and properties of matrices (i.e. determinants, eigenvalues and eigenvectors).
- **Polynomial manipulation**: factorization over different fields, simplification and partial fraction decomposition of rational functions.
- **Transcendental functions**: support for transcendental functions and their properties.
- **Calculus**: limits, derivatives, integration and expansions of functions.
- **Solving equations**: solving systems of linear equations, computing with radicals solutions of polynomials of degree less than five.
- **Programming language**: users may implement their own algorithms using a programming language.

The CAS chosen for this work is **SageMath** [19], the features and functionalities of which shall not be discussed here.

SageMath is an open source CAS distributed under the terms of the GNU GPLv3 [9].

Hereafter, an example in which symbolic computations are put against numerical (computations) shall be made.

Example 2.2. Take matrix M from Example 2.1:

$$\begin{bmatrix} \sqrt{2} & 1 \\ 2 & \sqrt{2} \end{bmatrix}.$$

Compare the different results given out when computing its determinant by defining M over the *symbolic ring SR* and the *finite-precision ring CDF*:

```
sage: matrix(SR, [[sqrt(2), 1], [2, sqrt(2)]]).det()
0
sage: matrix(CDF, [[sqrt(2), 1], [2, sqrt(2)]]).det()
-3.14018491736755e-16
```

We can observe that in SR $(\sqrt{2})^2 = 2$ since no approximations are made.

Now, take the polynomial $p(x)$:

$$p(x) = x^6 + 5\,x^5 - 3\,x^4 - 42\,x^3 + 12\,x^2 - x + 1.$$

If an attempt to calculate its roots over SR is made an exception will be thrown (here, a reader may refer to Abel-Ruffini theorem for further explanations); however, doing this over a finite-precision ring (such as CDF) will work:

```
sage: p = x^6 + 5*x^5 - 3*x^4 -42*x^3 + 12*x^2 - x + 1
sage: p.roots(ring=SR)
    RuntimeError: no explicit roots found
sage: p.roots(ring=CDF)
[(-3.865705050148171 - 1.5654017866113432*I, 1),
(-3.8657050501481702 + 1.5654017866113419*I, 1),
(-0.04843174828928114 - 0.2430512799158686*I, 1),
(-0.048431748289281144 + 0.24305127991586856*I, 1),
(0.38275295887213723 + 7.286537374692244e-17*I, 1),
(2.4455206380027437 - 1.995314986816126e-16*I, 1)]
```

What we may conclude from such an example is that numerical analysis is certainly a powerful tool as it allows for computations which could not happen with computer algebra, **but** computer algebra being able to compute an exact answer without any approximation will prove to be useful in our use case.

For deeper reasoning about the limits of computer algebra systems, one may refer to Mitic [15].

Eigenvalues, eigenvectors

In the following section, we shall define **eigenvalues** and **eigenvectors** and discuss the numerical stability of their computation; a reader may also consult Axler [1] or Strang [21] for further explanations.

Definition 2.6 (Eigenvalue, eigenvector)**.** Given a linear transformation T in a finite-dimensional vector space V over a field F into itself and a nonzero vector $\mathbf{v}$, $\mathbf{v}$ is an eigenvector of T if and only if

$$A\mathbf{u} = \lambda\mathbf{u}$$

with A the matrix representation of T, $\mathbf{u}$ the coordinate vector of $\mathbf{u}$ and λ a scalar in F known as eigenvalue associated with $\mathbf{v}$.

Similarly, we can define a row vector $\mathbf{x}_L$, and a scalar λ_L such that

$$\mathbf{x}_L A = \lambda_L \mathbf{x}_L,$$

which are called **left eigenvector** and **left eigenvalue** respectively.

Remark. Note that writing $A\mathbf{u} = \lambda\mathbf{u}$ is equivalent to $(A - \lambda I)\mathbf{u} = 0$.

It follows that the eigenvalues of A are the roots of

$$det(A - \lambda I)$$

which is a polynomial in λ known as the **characteristic polynomial** $ch_A(\lambda)$.

Definition 2.7 (Eigenspace)**.** Given a square matrix A and its eigenvalue λ, we define the eigenspace of A associated with λ the subspace E_A of all vectors satisfying the equation

$$E_A = \{\mathbf{u} : (A - \lambda I)\mathbf{u} = 0\} = ker(A - \lambda I).$$

Definition 2.8 (Algebraic, geometric multiplicities of eigenvalues)**.** Given a square matrix A and a scalar $\lambda \in \mathbb{C}$: we define the algebraic multiplicity of λ as

$$m_A(\lambda) = max\{k : (\exists s(x) : s(x)(x - \lambda)^k = ch_A(x))\}.$$

The geometric multiplicity of λ is defined as

$$\nu_A(\lambda) = dim(ker(A - \lambda I)).$$

Remark. Suppose A is a real square matrix, then the following statements are true:

- the eigenvalues of the left and right eigenvectors of A are the same,
- the left eigenvectors simplify into the transpose of the right eigenvectors of A^T.

Now, let us investigate how introducing perturbations in the representation of a matrix may influence the numerical stability of its eigenvalues.

Let A be a square matrix, $\lambda \in \mathbb{C}$ its eigenvalue, $\mathbf{x}$, $\mathbf{y}$ the right and left eigenvectors associated with λ. Consider the perturbed problem

$$\tilde{A}\tilde{\mathbf{x}} = \tilde{\lambda}\tilde{\mathbf{x}}$$

with ϵ the machine epsilon, $\tilde{A} = A + \epsilon\delta A$, $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon\delta\mathbf{x}$, $\tilde{\lambda} = \lambda + \epsilon\delta\lambda$.

Differentiating w.r.t. ϵ and multiplying by $\mathbf{y}^T$ on the left side gives

$$\mathbf{y}^T \delta A\mathbf{x} + \mathbf{y}^T A fl(\mathbf{x}) = fl(\lambda)\mathbf{y}^T\mathbf{x} + \mathbf{y}^T \lambda fl(\mathbf{x})$$

and, since $\mathbf{y}$ is the left eigenvector we can rewrite it as

$$\frac{\delta\lambda}{\delta\epsilon} = \frac{\mathbf{y}^T \delta A\mathbf{x}}{\mathbf{y}^T\mathbf{x}}.$$

Assuming $\|\delta A\| = 1$ and using the definition of dot product for $\mathbf{y}^T\mathbf{x}$ we get

$$|\delta\lambda| \leq \frac{1}{|\cos(\theta_\lambda)|}|\delta\epsilon|.$$

Definition 2.9 (Condition number of an eigenvalue)**.** [20] Given a square matrix A, the eigenvalue $\lambda \in \mathbb{C}$ and θ_λ the angle between the left and right eigenvectors associated with λ, the quantity

$$k_A(\lambda) = \frac{1}{\cos(\theta_\lambda)}$$

is called the condition number of the eigenvalue λ.

Jordan's canonical form

In the following section, we shall define **Jordan matrices** and discuss the stability of a transformation of a matrix into Jordan's canonical form.

Definition 2.10 (Jordan matrix)**.** A diagonal block matrix M is called a Jordan matrix if and only if each block along the diagonal is of the form

$$\begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & 0 & \lambda \end{bmatrix},$$

and we can write such a matrix M as $M = diag(J_{\lambda_1,n_1}, ..., J_{\lambda_k,n_k})$ with k the number of diagonal blocks it is made up of.

Each $n \times n$ block can be completely characterized by the tuple (λ, n) as it can fully describe both the structure and the dimension of a block.

Remark. Let V be a vector space defined over a field F and A a matrix defined in V. If the characteristic polynomial of A $ch_A(t)$ can be factorized into its linear factors over K, then A is similar to a Jordan matrix J. We define J **Jordan's canonical form** (**JCF**) of A.

Definition 2.11 (Defective matrix, defective eigenvalue)**.** Given a square $n \times n$ matrix A, if it has less than n distinct eigenvalues then it is called a defective matrix.

Furthermore, we define an eigenvalue λ of such a matrix as a defective eigenvalue if and only if

$$m_A(\lambda) > \nu_A(\lambda).$$

Now, we shall give a result on the stability of such a transformation the proof of which can be found in other works, such as Datta [6].

Theorem 2.1 (Stability of the JCF transformation)**.** Given a matrix A and its JCF $A = P^{-1}JP$, the transforming matrix P is highly ill-conditioned whenever A has at least a defective or nearly defective eigenvalue.

Lastly, we shall give an example to show the implications of this theorem by showing the differences in the JCF of a matrix and its perturbed version.

Example 2.3. Consider the $n \times n$ matrices

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ \alpha & 0 & 0 & 0 & 0 \end{bmatrix}$$

with $\alpha > 0$.

It is evident how A has a defective eigenvalue in $\lambda_A = 0$ and $m_A(0) = n$, $\nu_A(0) = 1$; furthermore, A is already in JCF.

Now, let us switch our focus to B. To compute its eigenvalues, take the characteristic polynomial $ch_B(t) = t^n - \alpha$: it has n distinct roots in

$$t_j = z_n^j \sqrt[n]{\alpha}$$

with $j = 1, ..., n$, $z_n = \cos\left(\frac{2\pi}{n}\right) + \mathrm{i} \sin\left(\frac{2\pi}{n}\right)$ and i imaginary unit such that $\mathrm{i}^2 = -1$.

To conclude, we shall show the JCF of A and B defined in SR computed by SageMath when $n = 4$.

```
sage: A = matrix(SR, [
            [1 if i == j-1
                else 0 for j in range(4)]
            for i in range(4)
        ])
sage: B = matrix(SR, [
            [1 if i == j-1
                else x if j == 0 and i == 3
                    else 0 for j in range(4)]
            for i in range(4)
        ])
sage: A
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
[0 0 0 0]
sage: A.jordan_form()
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
[0 0 0 0]
sage: B
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
[x 0 0 0]
sage: B.jordan_form()
[ I*x^(1/4)|         0|         0|         0]
[----------+----------+----------+----------]
[         0|  -x^(1/4)|         0|         0]
[----------+----------+----------+----------]
[         0|         0|-I*x^(1/4)|         0]
[----------+----------+----------+----------]
[         0|         0|         0|   x^(1/4)]
```

For the sake of clarity, we shall also show what the implications of B having such eigenvalues are.

Suppose $x = 10^{-10}$.

```
sage: B = matrix(SR, [
            [1 if i == j-1
                else 10**-10 if j == 0 and i == 3
                    else 0 for j in range(4)]
            for i in range(4)
        ])
sage: B
[           0            1            0            0]
[           0            0            1            0]
[           0            0            0            1]
[1/10000000000            0            0            0]
sage: P =  = B.jordan_form(transformation=True)[1]
sage: cond = norm(P.inverse()) * norm(P)
sage: cond
31622776.60168379
```

We can see that $\kappa(P) \gg 1$, as stated in Stability of the JCF transformation (theorem 2.1).

Theory and applications of pencils of matrices

This chapter will introduce the reader to the concept of a linear pencil of matrices and its properties. Throughout this chapter and the following one, a reader may refer to Gäntmacher [7], Kunkel, Mehrmann [13] and Beelen, Van Dooren [2].

Definition 3.1 (Linear pencil of matrices)**.** A linear pencil of matrices is defined as a 1-degree polynomial with matrix coefficients

$$\Gamma(\lambda) = A + \lambda B$$

with $\lambda \in \mathbb{C}, A$ and B $m \times n$ matrices. A linear pencil of matrices may also be called a **pair of matrices** and, in this thesis, we shall use synonymously both terms.

Regular pencils.

Now, we consider the case where $(A + \lambda B)$ is a regular pencil of matrices.

Definition 3.2 (Regular pencil)**.** [7, p. 25, Definition 2] A matrix pair (A, B) is said to be regular if and only if A and B are square matrices of the same size and the determinant $det(A + \lambda B)$ is not identically zero.

Consider the regular pencil of matrices

$$\Gamma(\lambda) = A + \lambda B,$$

let F be the field the entries of A and B belong to and r the rank of the pencil.

Denote with $D_j(\lambda)$ the greatest common divisor of all minors of order j of $\Gamma(\lambda)$ (with $j = 1, ..., r$) and assume without any loss of generality $D_j(\lambda)$ is monic and $D_0(\lambda) = 1$. Given the sequence,

$$D_r(\lambda),\ D_{r-1}(\lambda),\ ...,\ D_1(\lambda), D_0(\lambda)$$

we define the **invariant polynomials** [7, p. 26] of the pencil of matrices $\Gamma(\lambda)$ as the fractions

$$i_1(\lambda) = \frac{D_r(\lambda)}{D_{r-1}(\lambda)},\ i_2(\lambda) = \frac{D_{r-1}(\lambda)}{D_{r-2}(\lambda)},\ ...,\ i_r(\lambda) = D_1(\lambda).$$

We can now write the expansion of the invariant polynomials into irreducible factors in F as

$$i_1(\lambda) = \prod_{i=1}^{k} p_i(\lambda)^{\alpha_{1,i}},\ i_2(\lambda) = \prod_{i=1}^{k} p_i(\lambda)^{\alpha_{2,i}},\ \dots$$
$$i_r(\lambda) = \prod_{i=1}^{k} p_i(\lambda)^{\alpha_{r,i}},$$

with p_i an irreducible polynomial appearing in the expansion.

We define the **elementary (finite) divisors** [7, p. 27] e_i of the pencil of matrices $\Gamma(\lambda)$ all the polynomials $p_i(\lambda)^{\alpha_{j,i}}$ (with $j = 1, ..., r$) that are not equal to one.

A similar procedure may be defined for the pencil of matrices

$$\Theta(\mu, \lambda) = \mu A + \lambda B$$

leading to polynomials in two variables (μ, λ). Clearly, having $\mu = 1$ would lead to obtaining the elementary finite divisors of $\Gamma(\lambda)$; however, for each elementary divisor of degree q we have

$$e_i(\mu, \lambda) = \mu^q e_i(\frac{\lambda}{\mu}),$$

and, with this technique, it is possible to generate all the elementary divisors of $\Theta(\mu, \lambda)$ except for those of the form μ^q, which are called **elementary infinite divisors** [7, p. 27] of the pencil of matrices $\Theta(\mu, \lambda)$.

Remark. [7, p. 27] A regular pencil of matrices $\Gamma(\lambda) = A + \lambda B$ has elementary infinite divisors if and only if $det(B) = 0$.

We can now give a result on the equivalence of regular pencils.

Theorem 3.1 (Equivalence of regular pencils of matrices)**.** [7, p. 27, Theorem 2] Two regular matrix pairs

$$\Gamma_1(\lambda) = (A_1, B_1) \qquad\qquad \Gamma_2(\lambda) = (A_2, B_2)$$

are equivalent if and only if they have the same finite and infinite elementary divisors.

In other words, Γ_1 and Γ_2 are equivalent if and only if the invariant polynomials of the two pencils $\Theta_1(\mu, \lambda)$ and $\Theta_2(\mu, \lambda)$ are equivalent, with

$$\Theta_1(\mu, \lambda) = \mu A + \lambda B \qquad\qquad \Theta_2(\mu, \lambda) = \mu A + \lambda B$$

Lemma 3.2. [11, Equation 2.14] For any two regular equivalent pencils of $n \times n$ matrices the entries of which lie in a field F

$$\Gamma_1(\lambda) = A_1 + \lambda B_1 \qquad\qquad \Gamma_2(\lambda) = A_2 + \lambda B_2$$

there exist two nonsingular matrices $P \in F^{n\times n}$, $Q \in F^{n\times n}$ such that

$$P\Gamma_1(\lambda)Q = \Gamma_2(\lambda),$$

and we call such a transformation an **equivalence transformation**.

Singular pencils.

Next, we shall investigate the most general case of $m \times n$ pencils of matrices in order to introduce the reader to the concept of minimal indices of a pencil of matrices.

Definition 3.3 (Singular pencil)**.** [7, p. 25, Definition 2] A matrix pair A, B is said to be singular if and only if it is not regular.

Consider the singular pencil of (rectangular) matrices $\Gamma(\lambda) = A + \lambda B$ and assume its rank r is smaller than its number of columns n.

This implies the equation

$$(A + \lambda B)\mathbf{x} = 0$$

has nontrivial solutions $\mathbf{x}_1(\lambda), ..., \mathbf{x}_k(\lambda)$ and let X be the polynomial matrix made up of such polynomials

$$X = \begin{bmatrix} x_{1,1} & x_{2,1} & \dots & x_{k,1} \\ x_{1,2} & x_{2,2} & \dots & x_{k,1} \\ \vdots & & & \vdots \\ x_{1,n} & x_{2,n} & \dots & x_{k,n} \end{bmatrix}$$

The columns $\mathbf{x}_i(\lambda)$ (with $i = 1, ..., k$) of X can be chosen to be linearly independent; as a matter of fact, the columns are linearly dependent if the rank of X is less than k and, (only) in this case, we can choose k nontrivial polynomials $p_i(\lambda)$ such that

$$\sum_{i=1}^{k} p_i(\lambda)\mathbf{x}_i(\lambda) \equiv 0.$$

We choose the nontrivial polynomial $\mathbf{x}_1(\lambda)$ of least degree ϵ_1 in λ; next, amongst the very same solutions, we choose $\mathbf{x}_2(\lambda)$ of least degree ϵ_2 etc.

Since the maximum number of linearly independent solutions of the aforementioned equation is no more than $n - r$, this process has a finite number of steps.

To summarize, from an equation $(A+\lambda B)\mathbf{x} = 0$ we can obtain a sequence of solutions of non-increasing degree

$$\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_p,$$

and we define ϵ_i a **minimal index for the columns** [7, p. 38] of the pencil of matrices $\Gamma(\lambda)$.

We can also introduce **minimal indices for the rows** [7, p. 38] $\eta_1, \eta_2, \ldots, \eta_q$ of the pencil $(A + \lambda B)$ which we can yield working with the transpose of the pencil $A^T + \lambda B^T$ or, in other words, with the equation

$$(A^T + \lambda B^T)\mathbf{y} = 0.$$

We can now give another result on the equivalence of pencils of matrices.

Theorem 3.3 (Necessary condition for the equivalence of pencils)**.** [7, p. 39] Two arbitrary equivalent pencils have the same minimal indices for rows and columns.

Kronecker canonical form.

At this point, we can put together the theorems on the equivalence of regular and singular pencil of matrices, namely Equivalence of regular pencils of matrices (theorem 3.1) and Necessary condition for the equivalence of pencils (theorem 3.3) and give a result on the **equivalence of arbitrary matrix pairs**.

Theorem 3.4 (Kronecker)**.** [7, p. 40, Theorem 5] Two pencils $(A + \lambda B)$, $(A_1 + \lambda B_1)$ of rectangular $m \times n$ matrices are equivalent if and only if they have the same minimal indices for rows and columns and the same elementary finite and infinite divisors.

Remark. Restating Kronecker (theorem 3.4) using different words, we can say that a matrix pair (A, B) is completely characterized by its minimal indices for rows and columns and (its) elementary finite and infinite divisors and does not depend on their order.

Coherently, it should be possible to define a canonical form for pencils of matrices completely determined by both the minimal indices for rows and columns and the elementary finite and infinite divisors, and it is. We can now introduce the Kronecker canonical form of a matrix pair.

Theorem 3.5 (Kronecker canonical form)**.** Let $\Gamma(\lambda) = A + \lambda B$ be an arbitrary pencil of matrices and h, g the maximal number of constant independent solutions of the two equations

$$(A + \lambda B)\mathbf{x} = 0 \qquad\qquad (A^T + \lambda B^T)\mathbf{y} = 0.$$

The pencil $\Gamma(\lambda)$ is strictly equivalent to a quasi-diagonal matrix

$$\begin{bmatrix} O^{(h,g)} & & & & \\ & L & & & \\ & & L^T & & \\ & & & N & \\ & & & & G + \lambda I \end{bmatrix}$$

with G a Jordan matrix, and the (other) blocks defined as

$$L = \begin{bmatrix} L_{\epsilon_{h+1}} & & & \\ & L_{\epsilon_{h+2}} & & \\ & & \ddots & \\ & & & L_{\epsilon_p} \end{bmatrix}, \qquad L^T = \begin{bmatrix} L^T_{\eta_{h+1}} & & & \\ & L^T_{\eta_{h+2}} & & \\ & & \ddots & \\ & & & L_{\eta_q^T} \end{bmatrix},$$

$$N = \begin{bmatrix} N^{(u_1)} & & & \\ & N^{(u_2)} & & \\ & & \ddots & \\ & & & N^{(u_s)} \end{bmatrix}, \qquad L_i = \begin{bmatrix} \lambda & 1 & 0 & \dots & 0 & 0 \\ 0 & \lambda & 1 & & \vdots & \vdots \\ \vdots & \vdots & & \ddots & & \\ 0 & 0 & & & \lambda & 1 \end{bmatrix};$$

L_i is a rectangular $i \times (i + 1)$ matrix and

$$N^{(u)} = I^{(u)} + \lambda H^{(u)}.$$

We shall call such a matrix the Kronecker canonical form of $\Gamma(\lambda)$ and write it out as $K(\Gamma(\lambda))$.

For a proof of Kronecker (theorem 3.4) and Kronecker canonical form (theorem 3.5) the reader shall refer to the following chapter.

Corollary 3.5.1. Given a matrix pair (A, B) there exists a pair of nonsingular matrices P, Q such that

$$PAQ = K(A) \qquad\qquad PBQ = K(B),$$

with P an $m \times m$ and Q an $n \times n$ matrices respectively.

Example 3.1. Kronecker's canonical form of the pencil of matrices

$$\begin{bmatrix} 0 & & & & & \\ & \lambda & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & \lambda & \\ & & & 0 & 1 & \\ & & & & & \lambda+1 \end{bmatrix}.$$

is made up of three blocks $O^{(1,1)}$, L_0, L_1^T, two nilpotent blocks $N^{(1)}$, $N^{(2)}$ and one Jordan block $G + \lambda I^{(1)}$.

Fundamental applications.

The following section shall introduce the reader to differential-algebraic equations (which we may shorten as DAE) in order to explain what is the use case of the KCF of a pencil of matrices while keeping a strong focus on examples.

Definition 3.4 (Differential-algebraic equation)**.** [13, p. 3, Equation 1.1] An equation

$$F(t, x, \dot{x}) = 0$$

with $F : I \times D_x \times D_x \to \mathbb{C}^m$, $I \subseteq \mathbb{R}$ is a compact interval, D_x, $D_x \subseteq \mathbb{C}^n$ are open, $x : I \to \mathbb{C}^n$ a differentiable function and m, $n \in \mathbb{N}$ is called differential-algebraic equation.

What we're attempting to determine is a function x such that

$$\forall x \in I. F(t, x(t), \dot{x(t)}) = 0.$$

Differential-algebraic equations are used to model physical systems with a dynamic behaviour the states of which are subjected to certain constraints. For our purposes, we'll narrow the definition given above and rewrite it as follows.

Definition 3.5 (Linear DAE with constant coefficients)**.** [13, p. 13, Equation 2.1] An equation

$$B\dot{x} + Ax = f(t)$$

with $(A,\ B \in \mathbb{C}^{m\times n})$, $f : I \to \mathbb{C}^m$ a continuous and twice-differentiable function, $I \subseteq \mathbb{R}$ a compact interval and $m \in \mathbb{N}$ is called a linear differential-algebraic equation with constant coefficients.

It may be paired with an initial condition

$$x(t_0) = x_0.$$

Remark. [13, p. 13] The equations $B\dot{x} + Ax = f(t)$ and $\tilde{B}\dot{\tilde{x}} + \tilde{A}\tilde{x} = \tilde{f}(t)$, with

$$\tilde{A} = PAQ, \qquad \tilde{B} = PBQ, \qquad \tilde{f} = Pf, \qquad \tilde{x} = Q^{-1}x,$$
$$P \in \mathbb{C}^{m\times m}, \qquad Q \in \mathbb{C}^{n\times n}$$

and P, Q nonsingular matrices, are both linear differential-algebraic equations with constant coefficients and the relation $x = Q\tilde{x}$ provides a one-to-one mapping between their solution sets.

Next, we shall take as examples pencils of matrices made up of the blocks described before in order to show how the linear DAE with constant coefficients associated may be solved.

Example 3.2. Consider the pencil

$$\Gamma(\lambda) = \begin{bmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \end{bmatrix}.$$

We write the linear DAE with constant coefficients associated with $\Gamma(\lambda)$ as

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} f_1(t) \\ f_2(t) \end{bmatrix}.$$

To compute its solutions we must solve the linear system associated with the DAE specified above

$$\begin{cases} \dot{x_1} + x_2 - f_1(t) = 0 \\ \dot{x_2} + x_3 - f_2(t) = 0 \end{cases} \tag{2}$$

Denoting with g a generic C^2 function, the solution is

$$\begin{cases} x_1 = g \\ x_2 = -\dot{x_1} + f_1(t) \\ x_3 = \ddot{x_1} - \dot{f_1}(t) + f_2(t) \end{cases}$$

Example 3.3. For this example, we'll take the transpose of the pencil of matrices given in example 3.2

$$\Gamma(\lambda) = \begin{bmatrix} \lambda & 0 \\ 1 & \lambda \\ 0 & 1 \end{bmatrix}.$$

The DAE associated with $\Gamma(\lambda)$ is

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \end{bmatrix}.$$

We write this as a linear system

$$\begin{cases} \dot{x_1} - f_1(t) = 0 \\ \dot{x_2} + x_1 - f_2(t) = 0 \\ x_2 - f_3(t) = 0 \end{cases}$$

The system can be solved if and only if

$$f_1(t) = \dot{f}_2(t) - \ddot{f}_3(t),$$

and its solution is

$$\begin{cases} x_1 = f_2(t) - \dot{f}_3(t) \\ x_2 = f_3(t) \end{cases}$$

Example 3.4. Let $\Gamma(\lambda)$ be the pencil of matrices

$$\Gamma(\lambda) = \begin{bmatrix} 1 & \lambda & 0 & 0 \\ 0 & 1 & \lambda & 0 \\ 0 & 0 & 1 & \lambda \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Rewriting this as a linear DAE with constant coefficients yields

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{bmatrix}$$

We follow the same steps as in other examples

$$\begin{cases} \dot{x_2} + x_1 - f_1(t) = 0 \\ \dot{x_3} + x_2 - f_2(t) = 0 \\ \dot{x_4} + x_3 - f_3(t) = 0 \\ x_4 - f_4(t) = 0 \end{cases}$$

The solution is

$$\begin{cases} x_1 = f_1(t) - \dot{f_2}(t) + \ddot{f_3}(t) - \dddot{f_4}(t) \\ x_2 = f_2(t) - \dot{f_3}(t) + \ddot{f_4}(t) \\ x_3 = f_3(t) - \dot{f_4}(t) \\ x_4 = f_4(t) \end{cases}$$

Example 3.5. Let us take the pencil

$$\Gamma(\lambda) = \begin{bmatrix} \lambda & 1 & 0 & 0 \\ 0 & \lambda & 0 & 0 \\ 0 & 0 & \lambda+2 & 1 \\ 0 & 0 & 0 & \lambda+2 \end{bmatrix}$$

and write the DAE associated with it

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \\ f_4(t) \end{bmatrix}.$$

The linear system to solve is

$$\begin{cases} \dot{x_1} + x_2 - f_1(t) = 0 \\ \dot{x_2} - f_2(t) = 0 \\ \dot{x_3} + 2x_3 + x_4 - f_3(t) = 0 \\ \dot{x_4} + 2x_4 - f_4(t) = 0 \end{cases}$$

Denoting with g a C^2 function such that

$$\ddot{g} = \dot{f_1}(t) - f_2(t),$$

and (with) D, K arbitrary constants, the solution is

$$\begin{cases} x_1 = g \\ x_2 = f_1(t) - \dot{g} \\ x_3 = -\dfrac{1}{4}\big(4Kt - 2f_3(t)e^{2t} + f_4(t)e^{2t} - 4K\big)e^{-2t} \\ x_4 = \dfrac{1}{2}\big(f_4(t)e^{2t} + 2D\big)e^{-2t} \end{cases}$$

Computation of Kronecker's Canonical Form

The following chapter deals with the problem of computing Kronecker's canonical form for a matrix pair (A, B). The approach described will be shown to be correct, and an implementation of it using the CAS SageMath has been made publicly available under MIT License on GitHub [8].

We shall divide it into two subproblems: the first one deals with regular pencils of matrices, the other with singular pencils.

At the end of each of the following sections, the steps described shall be summarised in pseudocode.

Regular pencils.

Let $\Gamma(\lambda) = A + \lambda B$ be a regular pencil of matrices defined in a vector space over a field F.

First, let us clarify what we are to prove.

Theorem 4.1 (KCF of a regular pencil of matrices)**.** [7, p. 28, Theorem 3] Every regular pencil of matrices $\Gamma(\lambda) = A + \lambda B$ can be reduced to a matrix of the form

$$\begin{bmatrix} N^{(u_1)} & & & & \\ & N^{(u_2)} & & & \\ & & \ddots & & \\ & & & N^{(u_s)} & \\ & & & & G + \lambda I \end{bmatrix},$$

where the first s diagonal blocks correspond to infinite elementary divisors $\mu^{u_1}, ..., \mu^{u_s}$ of $\Gamma(\lambda)$ and the last block is uniquely determined by the finite elementary divisors of the given pencil.

Proof. The following proof will describe a procedure to reduce $\Gamma(\lambda)$ to Kronecker's canonical form.

First, we shall find a value $c \in F$ such that $det(\Gamma(c)) \neq 0$ and define the matrix

$$A_1 = A + cB.$$

Now, we rewrite $\Gamma(\lambda)$ in terms of A_1,

$$\Gamma(\lambda) = A_1 + (\lambda - c)B,$$

and premultiply it by A_1^{-1}

$$A_1^{-1}\Gamma(\lambda) = I + (\lambda - c)A_1^{-1}B. \tag{3}$$

Let us denote with J the JCF of $A_1^{-1}B$ and P_1 the similarity matrix used to compute it

$$J = P_1^{-1}(A_1^{-1}B)P_1,$$

and let us assume it is of the form

$$J = \begin{bmatrix} J_1 & \\ & J_0 \end{bmatrix}$$

with J_0 a nilpotent Jordan matrix and J_1 such that $det(J_1) \neq 0$ (which is the form returned by SageMath when computing Jordan matrices); call j, k the number of rows in J_0 and J_1 respectively.

Define the permutation matrix

$$P_\pi = \begin{bmatrix} & I^{(k)} \\ I^{(j)} & \end{bmatrix}$$

so that we can write

$$P_\pi^T J P_\pi = J' = \begin{bmatrix} J_0 & \\ & J_1 \end{bmatrix}.$$

Now, we rewrite (3) substituting $A_1^{-1}B$ with the permuted Jordan's canonical form

$$A_1^{-1}\Gamma(\lambda) = I + (\lambda - c)P_1 P_\pi^{-T} J' P_\pi^{-1} P_1^{-1}. \tag{4}$$

We can write the identity matrix in terms of P_1 as $I = P_1 P_1^{-1}$ in (4)

$$\begin{aligned} &A_1^{-1}\Gamma(\lambda) = \\ &= P_1 P_1^{-1} + (\lambda - c)P_1 P_\pi^{-T} J' P_\pi^{-1} P_1^{-1} = \\ &= P_1(P_1^{-1} + (\lambda - c)P_\pi^{-T} J' P_\pi^{-1} P_1^{-1}). \end{aligned}$$

We premultiply it by P_1^{-1}

$$P_1^{-1} A_1^{-1}\Gamma(\lambda) = P_1^{-1} + (\lambda - c)P_\pi^{-T} J' P_\pi^{-1} P_1^{-1}$$

and then postmultiply by P_1

$$P_1^{-1}A_1^{-1}\Gamma(\lambda)P_1 = I + (\lambda - c)P_\pi^{-T}J'P_\pi^{-1}.$$

The very same steps can be followed for P_π^{-T} and P_π^{-1}.
The final result is

$$P_\pi^T P_1^{-1}A_1^{-1}\Gamma(\lambda)P_1 P_\pi = I + (\lambda - c)J'. \tag{5}$$

Now, we may work on the blocks J_0, J_1 of J'.

For the following steps, it is useful to rewrite the expression on the right side of (5) so that the form of its diagonal blocks is explicitly readable, meaning

$$I + (\lambda - c)J' = \begin{bmatrix} I^{(j)} - cJ_0 + \lambda J_0 & \\ & I^{(k)} - cJ_1 + \lambda J_1 \end{bmatrix}.$$

Let us start from the first diagonal block

$$I^{(j)} - cJ_0 + \lambda J_0.$$

First, we need to postmultiply it in (5) by $K = (I^{(j)} - cJ_0)^{-1}$

$$P_\pi^T P_1^{-1}A_1^{-1}\Gamma(\lambda)P_1 P_\pi \begin{bmatrix} K & \\ & I^{(k)} \end{bmatrix} = \begin{bmatrix} I^{(j)} + \lambda K J_0 & \\ & I^{(k)} - cJ_1 + \lambda J_1 \end{bmatrix}.$$

Let us denote with $H^{(j)}$ the JCF of KJ_0, P_2 the similarity matrix used to compute it, and $N^{(j)} = I^{(j)} + \lambda H^{(j)}$

$$N^{(j)} = I^{(j)} + \lambda H^{(j)} = I^{(j)} + \lambda P_2^{-1}(KJ_0)P_2.$$

Remark. H is an upper shift matrix as it is the JCF of a nilpotent matrix.

Following the analogous steps to handle P_2 and its inverse yields us

$$\begin{bmatrix} P_2^{-1} & \\ & I \end{bmatrix} P_\pi^T P_1^{-1}A_1^{-1}\Gamma(\lambda)P_1 P_\pi \begin{bmatrix} KP_2 & \\ & I^{(k)} \end{bmatrix} = \begin{bmatrix} N^{(j)} & \\ & I^{(k)} - cJ_1 + \lambda J_1 \end{bmatrix}. \tag{6}$$

At this point, we can focus on the second diagonal block

$$I^{(k)} - cJ_1 + \lambda J_1.$$

We postmultiply it in (6) by J_1^{-1}

$$\begin{bmatrix} P_2^{-1} & \\ & I \end{bmatrix} P_\pi^T P_1^{-1} A_1^{-1} \Gamma(\lambda) P_1 P_\pi \begin{bmatrix} K P_2 & \\ & J_1^{-1} \end{bmatrix} = \begin{bmatrix} N^{(j)} & \\ & J_1^{-1} + (\lambda - c) I^{(k)} \end{bmatrix}. \quad (7)$$

Let us denote with G the JCF of the constant term $J_1^{-1} - cI^{(k)}$ and P_3 the similarity matrix used to compute it

$$J_1^{-1} - cI^{(k)} = P_3^{-1} G P_3.$$

Again, we follow the same procedure to handle P_3 and its inverse, thus obtaining our end result

$$\begin{bmatrix} P_2^{-1} & \\ & P_3^{-1} \end{bmatrix} P_\pi^T P_1^{-1} A_1^{-1} \Gamma(\lambda) P_1 P_\pi \begin{bmatrix} K P_2 & \\ & J_1^{-1} P_3 \end{bmatrix} = \begin{bmatrix} N^{(j)} & \\ & G + \lambda I^{(k)} \end{bmatrix}. \quad (8)$$

□

To conclude, we shall present the aforementioned steps used in order to compute Kronecker's canonical form $K(\Gamma(\lambda))$ in pseudocode.

Algorithm 1: Procedure to compute Kronecker's canonical form of a regular pencil

```
Data: $\Gamma(\lambda) = A + \lambda B$: regular pencil
Result: $(P, Q), K$: KCF of the pencil of matrices $\Gamma(\lambda)$
while True do
    $c \leftarrow$ random value;
    if $det(\Gamma(c)) \neq 0$ then
        continue
    end
    $A_1 \leftarrow A + c * B$;
    $P_1, J \leftarrow$ jordan($A_1^{-1} * B$);
    $\{J_0, J_1\} \leftarrow$ submatrices($J$);
    if $det(J_1) = 0$ then
        break
    end
end
$j, k \leftarrow$nrows($J_0$), nrows($J_1$);
$M \leftarrow (I^{(j)} - c * J_0)^{-1}$;
$P_2, N \leftarrow$ jordan($M * J_0$);
$P_3, G \leftarrow$ jordan($J_1^{-1} - c * I^{(k)}$);
$K \leftarrow$ diag($N, G + \lambda I$);
$P \leftarrow A_1 * P_1 *$ block diagonal matrix([$P_2, P_3$]);
$Q \leftarrow P_1 *$ block diagonal matrix([$M * P_2, J_1^{-1} * P_3$]);
return $(P, Q), K$;
```

Singular pencils.

Let $\Gamma(\lambda)$ be a singular pencil of $m \times n$ matrices the entries of which lie in a field F

$$\Gamma(\lambda) = A + \lambda B.$$

The pencil being singular implies either $r < m$ or $r < n$ with r rank of the pencil in the field $F(\lambda)$ of rational functions; let us assume without loss of generality the relation $r < n$ holds: otherwise, we shall work with the transpose of the pencil of matrices $\Gamma^T(\lambda) = A^T + \lambda B^T$ in order to reduce the problem to the case we shall describe hereafter.

First, we shall find a polynomial $x(\lambda)$ of minimal degree ϵ in the (right) kernel of $\Gamma(\lambda)$

$$x(\lambda) = x_0 - \lambda x_1 + \lambda^2 x_2 + ... + (-1)^\epsilon \lambda^\epsilon x_\epsilon.$$

In order to define a procedure to compute it, we consider the family of $(k+2) \times (k+1)$ matrices $M_k^{(A,B)}$ defined as

$$M_0^{(A,B)} = \begin{bmatrix} A \\ B \end{bmatrix}, \quad M_1^{(A,B)} = \begin{bmatrix} A & 0 \\ B & A \\ 0 & B \end{bmatrix}, \quad M_k^{(A,B)} = \begin{bmatrix} A & 0 & \dots & 0 \\ B & A & & \vdots \\ 0 & B & \ddots & \\ \vdots & \vdots & \ddots & A \\ 0 & 0 & \dots & B \end{bmatrix}.$$

The basis matrix of the (right) kernel of a matrix M_k - split into blocks of size $k+1$ - identifies a polynomial $g(\lambda)$ of degree k which satisfies the equations

$$Ag_0 = 0, \quad Bg_0 - Ag_1 = 0, \quad ..., \quad Bg_{k-1} - Ag_k = 0, \quad Bg_k = 0.$$

Note that if there is no polynomial of degree k in the kernel of the pencil $\Gamma(\lambda)$, then $ker(M_k)$ is empty.

Iteratively building the succession of matrices $\{M_i\}_{i \geq 0}$ and computing their kernels yields us the polynomial $x(\lambda)$, which is what we were looking for.

Depending on the value of ϵ, we shall distinguish two cases.

Case 1. $\epsilon > 0$.

We want to prove the following theorem.

Theorem 4.2 (Reduction theorem)**.** [7, p. 30, Theorem 4] Given a pencil of matrices $\Gamma(\lambda)$, if the polynomial of minimal degree ϵ in the (right) kernel of $\Gamma(\lambda)$ has degree $\epsilon > 0$, then $\Gamma(\lambda)$ is equivalent to a pencil of the form

$$\begin{bmatrix} L_\epsilon & 0 \\ 0 & \widehat{A} + \lambda \widehat{B} \end{bmatrix}.$$

Proof. We shall prove the theorem by defining a procedure to compute such a transformation.

We know both the vectors in B_1, B_2 are linearly independent [7, pp. 30-31], with

$$B_1 = \{x_0, x_1, ..., x_\epsilon\}, \qquad\qquad B_2 = \{Ax_0, Ax_1, ..., Ax_\epsilon\}.$$

Denote with P, Q the matrices obtained by completing the vectors in B_2, B_1 to a basis in F^m, F^n respectively; applying a change of basis to the pencil of matrices $\Gamma(\lambda)$ gives us

$$\tilde{\Gamma}(\lambda) = \tilde{A} + \lambda \tilde{B} = P^{-1}AQ + \lambda P^{-1}BQ,$$

with $\tilde{A}$, $\tilde{B}$ of the kind

$$\tilde{A} = \begin{bmatrix} \boxed{O^{(\epsilon,1)} \quad I^{(\epsilon)}} & * \\ & \widehat{A} \end{bmatrix}, \qquad\qquad \tilde{B} = \begin{bmatrix} \boxed{I^{(\epsilon)} \quad O^{(\epsilon,1)}} & * \\ & \widehat{B} \end{bmatrix};$$

the notation $*$ is used for a block with no relevant structure.

For the sake of clarity, we shall partition the pencil of matrices $\tilde{\Gamma}(\lambda)$ as

$$\begin{bmatrix} L_\epsilon & D + \lambda F \\ 0 & \widehat{A} + \lambda \widehat{B} \end{bmatrix}$$

with L_ϵ the first rectangular $\epsilon \times (\epsilon + 1)$ block.

Now we need to transform the pencil of matrices $\tilde{\Gamma}(\lambda)$ so that it is upper triangular.

We shall do this via the transformation

$$\begin{bmatrix} I^{(\epsilon)} & Y \\ 0 & I^{(m-\epsilon)} \end{bmatrix} \begin{bmatrix} L_\epsilon & D+\lambda F \\ 0 & \widehat{A}+\lambda\widehat{B} \end{bmatrix} \begin{bmatrix} I^{(\epsilon+1)} & -X \\ 0 & I^{(n-\epsilon-1)} \end{bmatrix}. \tag{9}$$

Applying the transformation in (9) yields

$$\begin{bmatrix} L_\epsilon & D+\lambda F+Y(\widehat{A}+\lambda\widehat{B})-L_\epsilon X \\ 0 & \widehat{A}+\lambda\widehat{B} \end{bmatrix}.$$

In other words, to make the pencil $\tilde{\Gamma}(\lambda)$ upper triangular, we shall determine two matrices X, Y such that

$$D+\lambda F+Y(\widehat{A}+\lambda\widehat{B})-L_\epsilon X=0.$$

Denoting with $\mathbf{y}_i$, $\mathbf{a}_i$, $\mathbf{b}_i$ the i-th row of Y, column of A and B respectively, we shall rewrite this condition as a system of linear equations

$$\begin{cases} x_{2,k}+\lambda x_{1,k}=d_{1,k}+\lambda f_{1,k}+\mathbf{y}_1\mathbf{a}_k+\lambda\mathbf{y}_1\mathbf{b}_k \\ x_{3,k}+\lambda x_{2,k}=d_{2,k}+\lambda f_{2,k}+\mathbf{y}_2\mathbf{a}_k+\lambda\mathbf{y}_2\mathbf{b}_k \\ \dots \\ x_{\epsilon+1,k}+\lambda x_{\epsilon,k}=d_{\epsilon,k}+\lambda f_{\epsilon,k}+\mathbf{y}_\epsilon\mathbf{a}_k+\lambda\mathbf{y}_\epsilon\mathbf{b}_k \end{cases} \tag{10}$$

$$(k=1,...,n-\epsilon-1)$$

To approach this problem, we shall define the $1\times(\epsilon-1)(n-\epsilon-1)$ matrix W as the differences between subsequent rows in D and F flipping the sign every $m-\epsilon-1$ elements, but first let us differentiate between two other subcases.

To make this definition easier to understand, we shall present the pseudocode of an algorithm to compute W. Now, we shall distinguish two other subcases.

Subcase (i): $\epsilon=1$.

In this case, there are no conditions on the values of the elements of Y; we shall choose arbitrary values for its entries and compute those in X according to the last equation in the linear system in (10).

Subcase (ii): $\epsilon>1$.

In this case, we shall define the matrix W according to the definition given above; in an attempt to make the explanation easier to understand, we shall give its definition (again, but) in pseudocode.

Algorithm 2: Procedure to compute W.

```
$sign \leftarrow -1$;
$W \leftarrow []$;
$i \leftarrow 0$;
for $i < n - \epsilon - 1$ do
    $j \leftarrow 0$;
    for $j < m - \epsilon - 1$ do
        if $j \bmod (m - \epsilon - 1) = 0$ then
            $sign \leftarrow sign * (-1)$;
        end
        append($W$, $sign * (f_{i+1,j} - d_{i,j})$);
    end
end
return $matrix(W)$
```

Solving for Z in $ZM_{\epsilon-2}^{(\widehat{A},\widehat{B})} = W$ yields a matrix of the form (a reader shall refer to [7, p.34] for the proof of the solvability of the equation and an explanation about the form of Z)

$$Z^T = \begin{bmatrix} \mathbf{y}_1 - \mathbf{y}_2 \\ -(\mathbf{y}_2 - \mathbf{y}_3) \\ \cdots \\ \mathbf{y}_{\epsilon-1} - \mathbf{y}_\epsilon \end{bmatrix}.$$

We can now recover Y and, subsequently, X from the linear system in (10). $\square$

To help make the procedure more intelligible, we shall present the pseudocode to apply the transformation in Reduction theorem (theorem 4.2).

Algorithm 3: Procedure to reduce a singular pencil

```
Data: $\Gamma(\lambda) = A + \lambda B$: singular pencil
Result: $K$: Reduced form of $\Gamma(\lambda)$
$V \leftarrow$ polynomial of minimal degree $\in ker(\Gamma)$;
$Q \leftarrow$ complete to a basis($V$);
$P \leftarrow$ complete to a basis($AV$);
$\tilde{A} = P^{-1}AQ$;
$\tilde{B} = P^{-1}BQ$;
$L_\epsilon, D, F, \widehat{A}, \widehat{B} \leftarrow$ partition pencil($\tilde{A} + \lambda\tilde{B}$);
if $degree(V) \neq 0$ then
    $M \leftarrow$ build $M_\epsilon(\widehat{A}, \widehat{B}, \epsilon - 1)$;
    $W \leftarrow$ build $W(F, D)$;
    $Z \leftarrow WM^{-1}$;
    $Y \leftarrow$ partition(Z);
    $X \leftarrow$ solve system in (10);
    $L \leftarrow$ block matrix([[$I, Y$], [$0, I$]]);
    $R \leftarrow$ block matrix([[$I, -X$], [$0, I$]]);
    return $LP^{-1}\Gamma(\lambda)QR$;
else
    return $P^{-1}\Gamma(\lambda)Q$;
end
```

Case 2. $\epsilon = 0$.

In this case, the columns of the pencil are connected by linear relations with constant coefficients.

This implies the transformation given by matrices P, Q returns a pencil of the form

$$\begin{bmatrix} 0 & A_0 + B_0 \end{bmatrix}.$$

Assuming there are g constant independent solutions, then the pencil can be reduced to

$$\begin{bmatrix} O^{(1,g)} & A_1 + \lambda B_1 \end{bmatrix}.$$

Supposing there are h constant independent solutions in the equation for $\Gamma(\lambda)^T = 0$, then the first h rows can be reduced to zeros so that we obtain

$$\widehat{\Gamma}(\lambda) = \begin{bmatrix} O^{(h,g)} & \\ & \widehat{A} + \lambda\widehat{B} \end{bmatrix}.$$

Note there are no constant independent solutions in the pencil $\widehat{A} + \lambda\widehat{B}$.

We shall end our considerations on singular pencils by declaring the following lemma the proof of which can be found in [7, pp. 32-33].

Lemma 4.3. If a pencil of matrices $\Gamma(\lambda) = A + \lambda B$ has a polynomial of minimal degree $\epsilon > 0$ in its (right) kernel, then the pencil $\widehat{\Gamma}(\lambda)$ obtained via the transformation defined in Reduction theorem has no polynomial in its (right) kernel of degree less than ϵ, with

$$\widehat{\Gamma}(\lambda) = \widehat{A} + \lambda\widehat{B}.$$

Next, we shall show on the following page the pseudocode for an algorithm to compute Kronecker's canonical form of an arbitrary pencil of matrices.

Algorithm 4: Procedure to compute Kronecker's canonical form of an arbitrary pencil.

```
Data: $\Gamma(\lambda) = A + \lambda B$: pencil of $m \times n$ matrices
Result: $K$: Kronecker's canonical form of $\Gamma(\lambda)$
Begin
    /* Edge case handling:  A, B may be null matrices   */
    if $A$ is zero and $B$ is zero then
        return $\Gamma(\lambda)$;
    end
    /* L, R are the left and right transformation
       matrices respectively                            */
    $\tilde{\Gamma}(\lambda) \leftarrow A + \lambda B$;
    $L, R \leftarrow I^{(m)}, I^{(n)}$;
    /* Toggled on when if the rightmost block on the
       bottom row has a left kernel                     */
    $transpose \leftarrow False$;
    /* Repeat while the rightmost block on the bottom row
       is singular                                      */
    while True
        /* If the block is now regular                  */
        if dimensions($\tilde{A}$) = dimensions($\tilde{B}$) and det($\tilde{\Gamma}(\lambda) \neq 0$) then
            break;
        end
        /* If the block has a left kernel               */
        if $ncols(\tilde{A}) < nrows(\tilde{A})$ then
            $\tilde{\Gamma}(\lambda) \leftarrow \tilde{\Gamma}^T(\lambda)$;
            $L, R, transpose \leftarrow R^*, L^*, True$;
        end
        $(P, Q), \tilde{\Gamma}(\lambda) \leftarrow$ call Reduction theorem($\tilde{\Gamma}(\lambda)$);
        $u, w \leftarrow$ nrows($P$), nrows($Q$);
        /* Compose matrix transformations               */
        $L \leftarrow$ block diagonal matrix($[I^{(m-u)}, P]$) $*L$;
        $R \leftarrow R*$ block diagonal matrix($[I^{(n-w)}, Q]$);
        /* $\tilde{\Gamma}(\lambda)$ is now the rightmost block on the bottom
           row                                          */
        $\_, \tilde{\Gamma}(\lambda) \leftarrow$ partition($\tilde{\Gamma}(\lambda)$);
        /* Next iteration, we shall check if there exists a
           right kernel first                                   */
        if transpose then
            $\tilde{\Gamma}(\lambda) \leftarrow \tilde{\Gamma}^T(\lambda)$;
            $L, R \leftarrow R^*, L^*$;
            $transpose \leftarrow False$;
        end
    end
    /* Check if $\tilde{\Gamma}(\lambda)$ is non-empty          */
    if dimensions($\tilde{A}$) $\neq (0,0)$ then
        $(P, Q), \_ \leftarrow$ call KCF Regular Pencil($\tilde{\Gamma}(\lambda)$);
    else
        $P, Q \leftarrow$ empty matrix, empty matrix;
    end
    $u \leftarrow$ nrows($P$);
    /* Compose matrix transformations                           */
    $L \leftarrow$ block diagonal matrix($[I^{(m-u)}, P^{-1}]$) $*L$;
    $R \leftarrow R*$ block diagonal matrix($[I^{(n-u)}, Q]$);
    /* Return KCF of $\Gamma(\lambda)$                          */
    return $L * \Gamma(\lambda) * R$
end
```

Implementation

The following chapter shall go into a few technical details on the implementation of the algorithm provided in [8].

For the code to run on your local machine, it is required to setup **Python** [22], **SageMath** and **Pytest**.

How to use.

Invoking the command `sage` on a script named `script.sage` produces a (pre-)parsed Python module named `script.sage.py` and subsequently runs it. The newly produced file can also be run with `python script.sage.py` as it is a valid Python module, but its name containing `.sage.py` makes it unusable from either SageMath or Python as an external module. This means the following code produces errors.

```
from script.sage import *
```

This problem can be solved by renaming the resulting file, as in the **Makefile** provided.

At this point, it is possible to include the functions defined either in a Python or a SageMath environment.

The function implementing the algorithm defined in Procedure to compute Kronecker's canonical form of an arbitrary pencil. (algorithm 4) has been designed so that a SageMath developer may find its usage similar to that of `jordan_form`: its signature contains the optional boolean parameter `transformation`; if explicitly toggled on, the function shall return the transformation matrices too, as is the case with the method `jordan_form`.

Testing the code.

To ensure there are no errors in the implementation, a suitable test suite has been provided. We've chosen to use a functional testing technique known as **unit testing**, and the test suite accurately tests the function `kcf`.

The test suite consists of three files populated with functions which build pencils of matrices either with only regular blocks or only singular blocks or combinations of both, (already) in Kronecker's canonical form, apply a random change of basis transformation, compute Kronecker's canonical form and assert the transformation in output is correct. Since we are testing pencils of matrices made up of only regular blocks, only singular blocks and a mix of both, and we are applying a non-deterministic transformation, by repeating the tests we may assume we have covered as many cases as possible.

The tool used for testing - *pytest* - can also provide information on the test coverage, which is 91%; the reasons it is not 100% are two: first, there is a routine which, given a pencil of matrices returns its string representation, and it is not called unless tests fail; second, the code we're testing includes - in its internal routines (i.e. those with a leading underscore in their names) - some lines to handle invalid input values, and the "public" routines do not use invalid input values.

On the following page, we provide the reader with a minimal working example of the usage of the code written. We use two distinct variables for the (two) matrices making up the pencil of matrices.

The snippet works the very same way the functions in the test suite have been defined: starting with a pencil of matrices in Kronecker's canonical form, we apply a random change of basis transformation and compute Kronecker's canonical form; next, we assert the transformation given in output is correct. Last, we print the string representation of Kronecker's canonical form of the pencil of matrices in input.

The pencil of matrices $\Gamma(\lambda) = A + \lambda B$ the example is working with is

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & & & \\ 0 & 0 & 1 & 0 & & & \\ 0 & 0 & 0 & 1 & & & \\ & & & & 1 & 0 & \\ & & & & 0 & 1 & \\ & & & & & & 42 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 0 & 0 & 0 & & & \\ 0 & 1 & 0 & 0 & & & \\ 0 & 0 & 1 & 0 & & & \\ & & & & 0 & 1 & \\ & & & & 0 & 0 & \\ & & & & & & 1 \end{bmatrix}.$$

```
import sage.all as sa
from kcf import kcf_sage as kcf


def random_invertible_matrix(n: int):
    while True:
        M = sa.random_matrix(sa.ZZ, n, n)
        if not (M.is_singular()):
            return M.change_ring(sa.SR)


L3_A = sa.matrix(sa.SR, [[0, 1, 0, 0],
                         [0, 0, 1, 0],
                         [0, 0, 0, 1]])
L3_B = sa.matrix(sa.SR, [[1, 0, 0, 0],
                         [0, 1, 0, 0],
                         [0, 0, 1, 0]])
A = sa.block_diagonal_matrix([L3_A,
                              sa.identity_matrix(2),
                              sa.matrix(sa.SR, [[42]])])
B = sa.block_diagonal_matrix([L3_B,
                              sa.matrix(sa.SR, [[0, 1],
                                                [0, 0]]),
                              sa.identity_matrix(1)])

D = random_invertible_matrix(A.nrows())
C = random_invertible_matrix(A.ncols())

A = D.inverse() * A * C
B = D.inverse() * B * C

(L, R), (KCF_A, KCF_B) = kcf.kronecker_canonical_form(
    A, B, transformation=True)
assert ((L*A*R - KCF_A).is_zero()
        and (L*B*R - KCF_B).is_zero()
        and not L.is_singular() and not R.is_singular())
print(f'KCF:\n{kcf.stringify_pencil(KCF_A, KCF_B)}')
```

Time complexity.

As the algorithm works with computer algebra, an actual estimation of the time complexity of elementary operations (such as the sum of two variables) is implementation, ring and value-dependent; what we can do is calculate the number of operations needed at each step and give a measure of the complexity of the whole procedure based on it.

Assume the starting point is an arbitrary pencil of $m \times n$ matrices $\Gamma(\lambda) = A + \lambda B$.

The method `_reduction_theorem`, which operates on singular pencils of matrices, is called $p + q$ times, with p the number of L_ϵ blocks and q that of L_η^T blocks; internally, it calls `_compute_lowest_degree_polynomial`, which iteratively computes the right kernel of an M_i matrix with $i = 0, ..., k$ and k is the minimal degree of the polynomial at this point.

Remark. The time complexity for Gaussian elimination on an $m \times n$ matrix is $\Theta(mn \min(m, n))$ [4].

To summarize, the number of operations needed to reduce an arbitrary singular pencil of matrices is

$$\mathcal{O}((p + q)\alpha mn \min(m, n)) = \mathcal{O}(\alpha mn \min(m, n)^2),$$

with $\alpha = \max(\{\epsilon_0, ..., \epsilon_p, \eta_0, ..., \eta_q\})$.

Next, let $\Gamma(\lambda) = A + \lambda B$ be a regular pencil of $n \times n$ matrices. To make calculations easier, we shall assume the factorization of the characteristic polynomial of $\Gamma(\lambda)$ is known.

Remark. Given the factorization of the characteristic polynomial of an $n \times n$ matrix, the time complexity for the algorithm to compute Jordan's canonical form is $\mathcal{O}(n^4)$ [17].

Since the number of Jordan matrices to be computed in the case of a regular pencil is - at most - 3, we can conclude the number of operations for a regular matrix pair is

$$\mathcal{O}(3n^4) = \mathcal{O}(n^4).$$

Tying all together, the number of operations needed to compute Kronecker's canonical form is

$$\mathcal{O}(\alpha mn \min(m, n)^2 + (\max(m, n) - p - q)^4).$$

Conclusions and future work.

We have surveyed the problem of computing Kronecker's canonical form of an arbitrary pencil of matrices, which has applications in calculating the solutions for differential-algebraic equations, and made use of SageMath's [19] support for symbolic computations to implement an algorithm to compute such canonical form. It is also important to emphasize how such a procedure was not publicly available before.

Defining a procedure to calculate the matrices needed to transform a matrix pair to its Kronecker's canonical form, we have proved Kronecker canonical form (theorem 3.5) and shown an algorithm for it in Procedure to compute Kronecker's canonical form of an arbitrary pencil. (algorithm 4).

The result is a software implementation of this very same procedure [8], which has been tested to a sufficient degree to ensure it is properly working.

Despite what has been accomplished, more work can be done on optimizations. A direction for future work would include redefining the method used to compute the polynomial of minimal degree ϵ in the right kernel of the pencil of matrices in input; every iteration, it builds an M_ϵ matrix (refer to chapter **Computation of Kronecker's Canonical Form**) and computes from scratch its right kernel.

We can see how the work required to put an $M_\epsilon^{(A,B)}$ matrix in echelon form partially overlaps with that needed for an $M_{\epsilon-1}^{(A,B)}$ matrix as the equations solved by the two polynomials $g(\lambda)$, $p(\lambda)$ calculated from these two matrices are

$$\begin{aligned}
&Ag_0 = 0, \quad Bg_0 - Ag_1 = 0, \quad ..., \quad Bg_{\epsilon-1} - Ag_\epsilon = 0, \quad Bg_\epsilon = 0 \text{ and} \\
&Ap_0 = 0, \quad Bp_0 - Ap_1 = 0, \quad ..., \quad Bp_{\epsilon-2} - Ap_{\epsilon-1} = 0, \quad Bp_{\epsilon-1} = 0.
\end{aligned}$$

Bibliography

[1] Sheldon Jay Axler. *Linear Algebra Done Right*. Springer, 1997.

[2] Th. Beelen and P. Van Dooren. An improved algorithm for the computation of kronecker's canonical form of a singular pencil. *Linear Algebra and its Applications*, 105:9–65, 1988.

[3] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[4] Stephen Boyd and Lieven Vandenberghe. *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*. Cambridge University Press, 2018.

[5] Chunqi Chang, Zhi Ding, Sze Fong Yau, and F.H.Y. Chan. A matrix-pencil approach to blind separation of colored nonstationary signals. *IEEE Transactions on Signal Processing*, 48(3):900–907, 2000.

[6] Biswa Nath Datta. Numerical methods for linear control systems, 2004.

[7] Felix R. Gantmacher. *The Theory of Matrices, Vol. 2*. American Mathematical Society, 2000.

[8] Trapani Giacomo. Computation of Kronecker's Canonical Form in a Computer Algebra System, 2022. `https://github.com/liviusi/kronecker-canonical-form`.

[9] GNU General Public License, v3. `http://www.gnu.org/licenses/gpl.html`, June 2007. Last retrieved 2020-01-01.

[10] IEEE. IEEE-754, Standard for Floating-Point Arithmetic. *IEEE Std 754-2008*, pages 1–58, 01 2008.

[11] Kh. D. Ikramov. Matrix pencils: Theory, applications, and numerical methods.

[12] K. Kalorkoti. Introduction to Computer Algebra. https://www.inf.ed.ac.uk/teaching/courses/ca/notes01.pdf, January 2019.

[13] Mehrmann V. Kunkel P. *Differential-algebraic equations: Analysis and numerical solution*. EMS Textbooks in Mathematics. EMS, 2006.

[14] F. Lewis. Fundamental, reachability, and observability matrices for discrete descriptor systems. *IEEE Transactions on Automatic Control*, 30(5):502–505, 1985.

[15] Peter Mitic and Peter G. Thomas. Pitfalls and limitations of computer algebra, 1994.

[16] G. Nico and J. Fortuny. Using the matrix pencil method to solve phase unwrapping. *IEEE Transactions on Signal Processing*, 51(3):886–888, 2003.

[17] Bernard Parisse and Morgane Vaughan. Jordan normal and rational normal form algorithms. *CoRR*, abs/cs/0412005, 2004.

[18] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*, volume 37. 01 2007.

[19] W. A. Stein et al. *Sage Mathematics Software (Version x.y.z)*. The Sage Development Team, 2022. `http://www.sagemath.org`.

[20] G.W. Stewart and J. Sun. *Matrix Perturbation Theory*. Computer Science and Scientific Computing. Elsevier Science, 1990.

[21] Gilbert Strang. *Introduction to Linear Algebra*. Wellesley-Cambridge Press, 2009.

[22] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.