UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

DIPARTIMENTO DI
INFORMATICA

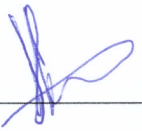
PhD Program in Computer Science and Mathematics
XXXIII cycle

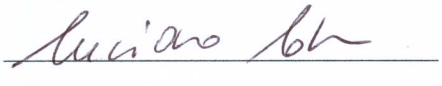
Research Project

PhD Student: Harold Taeter

Supervisor: Prof. Luciano Lopez

Coordinator: Prof. Maria F. Costabile

PhD student signature 

Supervisor signature 

1. **Research title**
   Retrieving the Index of Differential-Algebraic Equations

2. **Research area**
   Numerical analysis

3. **Research motivation and objectives**
   In order to study the behaviour of a physical process, we create a mathematical model of the system. If we are especially interested in its dynamical evolution, the tool we usually use is a system of differential equations. However, if the states of the physical system are constrained then algebraic equations representing these constraints will appear in the mathematical model. An example would be the Kirchhoff's laws in electrical networks. Such systems, containing at the same time differential equations and algebraic equations, are referred as *differential-algebraic systems of equations* or *implicit systems of differential equations*.

   Among this category of mathematical problems, a subset will be of particular interest. It has the following form

$$\lambda B x(t) = A x(t), \quad x(0) = x_0, \quad A, B \in \mathbb{R}^{n \times n}, \tag{1}$$

   where λ stands for the differential operator. It is worth noting that this problem is linear and its coefficients (the entries of the matrices A and B) are constant.

   It is evident that, if B is non-singular, i.e. $\det(B) \neq 0$, the problem (1) can be reduced to the explicit system of equations $\lambda x(t) = B^{-1}Ax(t)$. In this case, the solution to (1) can be recovered by studying the Jordan structure of the eigenvalues of $B^{-1}A$. However, we will rather be interested in a weaker assumption than forcing B to be non-singular, we will impose that the system is *regular*, i.e. $\det(\lambda B - A) \neq 0$ for at least one value of λ. We will then define the roots of the non-zero polynomial $\det(\lambda B - A)$ to be the so-called *generalized eigenvalues*. It is possible to construct a general solution to this system by using its generalized eigenvalues. The Jordan form is then replaced by the so-called Weierstrass form, which is a generalization of the Jordan form for regular pencils. Studying the Weierstrass structure of the matrix pencil $\lambda B - A$ will allow us to construct a general solution to (1).

   An interesting point is that, since B can be singular, we can actually have infinite generalized eigenvalues. Furthermore, the Weierstrass form has a well defined structure for $\lambda_0 = \infty$. That structure corresponds to what we call the *impulsive* solutions of (1) and the size of the corresponding largest Jordan block defines the so-called *index* of the infinite eigenvalue. Retrieving this index will be of paramount importance when running a numerical simulation of a differential-algebraic system of equations.

   Dr. Nicola Mastronardi and Prof. Paul Van Dooren recently proposed a new method for solving explicit systems of differential equations, which is a subcase of the problem (1) where B is the identity matrix. The algorithm, based on a preliminary reduction to Hessenberg form, is of overall cubic complexity and has the additional advantages that known eigenvalues can be deflated exactly, by using so-called QR steps with "perfect" shift strategy. The goal of my research project will be to extend this algorithm to the general case (1) and to study its behaviour.

4. **State of the art**
   The basic theory for the type of problem (1) had already been established in the nineteenth

century by the pioneering work of Weierstrass [14, 15] and Kronecker [9] on matrix pencils. However, we had to wait until the work of Gear [5] for the scientific communities in computer science, engineering and mathematics to realize that the theory of differential-algebraic equations could be used to model dynamical systems. Subsequent works finally lead to the use of differential-algebraic equations in direct numerical simulation. Since then, the number of research in this area has been exponentially increasing and it is now widely accepted that dynamical systems can be studied using differential-algebraic systems of equations.

As mentioned above, explicit systems of differential equations, i.e.

$$\lambda x(t) = Ax(t), \quad x(0) = x_0, \quad A \in \mathbb{R}^{n\times n}, \tag{2}$$

are a special case of general differential-algebraic equations and it is only natural that scientists first started by focusing on this very important type of problem.

In order to solve (2), one needs to construct the Jordan structure of A at each of its eigenvalue. Let's suppose A has an eigenvalue λ_0 and we want to construct the Jordan structure of A at this eigenvalue. To do so, we need to compute the dimension of the null spaces $\mathcal{N}_i := \mathrm{Ker}(A - \lambda_0 I)^i$. We define n_i to be the dimension of each of these spaces, i.e. $n_i := \dim(\mathcal{N}_i)$. Clearly, these null spaces are nested and we define the *index* k to be the smallest integer such that $n_k = n_{k+1}$. We have:

$$\mathcal{N}_1 \subset \mathcal{N}_2 \subset \cdots \subset \mathcal{N}_k = \mathcal{N}_{k+1} \tag{3}$$

$$n_1 < n_2 < \cdots < n_k = n_{k+1}. \tag{4}$$

Since the works of Kublanovskaya [10], Ruhe [12] and Golub&Wilkinson [6], we know there is a one-to-one relation between the space dimensions n_i and the sizes of the Jordan blocks corresponding to λ_0. This means that the values n_i are basically what we will be trying to compute to be able to solve (2).

At first sight, in order to compute n_i, it seems to be necessary to compute $(A-\lambda_0 I)^i$ and then to compute its null space. However, in terms of computational cost, computing $(A - \lambda_0 I)^i$ can be extremely expensive. Kublanovskaya and Ruhe noted that the construction of the powers $(A - \lambda_0 I)^i$ was actually avoidable and working directly on $(A - \lambda_0 I)$ was sufficient to compute the integers n_i. Their idea was to construct a unitary matrix V such that

$$V = \left[\begin{array}{c|c|c|c|c} V_1 & V_2 & \cdots & V_k & V_{k+1} \end{array} \right] \tag{5}$$

where

$$\mathcal{N}_i = \mathrm{Im}(\left[\begin{array}{c|c|c|c} V_1 & V_2 & \cdots & V_i \end{array} \right]), \quad i = 1, \ldots, k$$

i.e. V_i completes the orthogonal basis of $\mathcal{N}_{i-1}$ to an orthogonal basis for the larger space $\mathcal{N}_i$. We now define the integers r_i to be the increments in dimension of the null spaces $\mathcal{N}_i$, i.e.

$$r_1 = n_1, \quad r_i = n_i - n_{i-1},\ i = 2, \ldots, k. \tag{6}$$

By applying the unitary matrix V to $(A - \lambda_0 I)$, we end up with the following form:

$$\tilde{A} = V^T(A - \lambda_0 I)V = \left[\begin{array}{ccccc|c} 0_{r_1} & \tilde{A}_{1,2} & \cdots & \tilde{A}_{1,k-1} & \tilde{A}_{1,k} & \tilde{A}_{1,k+1} \\ 0 & 0_{r_2} & \tilde{A}_{2,3} & \cdots & \tilde{A}_{2,k} & \tilde{A}_{2,k+1} \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & 0_{r_{k-1}} & \tilde{A}_{k-1,k} & \tilde{A}_{k-1,k+1} \\ 0 & 0 & \dots & \cdots & 0_{r_k} & \tilde{A}_{k,k+1} \\ \hline 0 & 0 & 0 & \cdots & 0 & \tilde{A}_{k+1,k+1} \end{array}\right] \tag{7}$$

where

- the diagonal blocks 0_{r_i} of $\tilde{A}$ are square and of dimension r_i, for $i = 1, \dots, k$,
- the blocks $\tilde{A}_{i-1,i}$ are of full column rank r_i, for $i = 2, \dots, k$,
- the block $\tilde{A}_{k+1,k+1}$ is nonsingular (provided it is not empty).

This comes from the fact that

$$(A - \lambda_0 I)^i V_i = 0 \iff (A - \lambda_0 I)V_i = \left[\begin{array}{c|c|c|c} V_1 & \cdots & V_{i-1} \end{array}\right] C_{i-1}, \quad i \geq 2,$$

which implies

$$V_i^*(A - \lambda_0 I)V_j = 0 \iff i \geq j.$$

Since each n_i can be computed from the integers r_i, it means that these indices also define uniquely the Jordan structure of A at λ_0. There are exactly $r_i - r_{i+1}$ Jordan blocks of size i corresponding to λ_0 (where r_{k+1} has been assumed to be zero). It is now possible to develop a recursive algorithm to compute the successive r_i.

This is the idea that was developed by Kublanovskaya and Ruhe in [10, 12] and the corresponding algorithm had a worst-case complexity of $\mathcal{O}(rn^3)$ flops (one flop is the work needed for one addition and one multiplication) for $r = \sum_{i=1}^{k} r_i$ recovered eigenvalues ($r = n_k$ is the algebraic multiplicity of the given eigenvalue λ_0). Only unitary transformations were used which is necessary to ensure backward stability in the method. A software implementing this algorithm was given in [8]. At this point, cubic complexity could not be achieved and this was because these methods used full matrix decompositions at each rank determination and there could potentially be $\mathcal{O}(r)$ rank tests during the construction of the staircase form (7). Golub&Wilkinson [6] proposed the first algorithm that had at most cubic complexity but, because non-orthogonal bases were used during some intermediate calculations, backward stability could not be accomplished. The same problem appeared with the method proposed by Anstreicher&Rothblum [1], which relies on Gauss-Jordan elimination, a potentially unstable technique. Finally, the first backward stable algorithms with cubic complexity were proposed by Beelen&Van Dooren in [2, 3]. In 2015, Guglielmi et al. [7] have revisited the problem and developed a new method of cubic complexity. Then, as mentioned previously, Mastronardi et al. [11] proposed very recently a new method, based on a preliminary reduction to Hessenberg form. This last algorithm is also of overall cubic complexity, but has the additional advantages that known eigenvalues can be deflated exactly, by using so-called QR steps with "perfect" shift strategy.

Concerning the differential-algebraic systems of equations of the form (1), extensions of the procedures mentioned above have been developed. At first, the work of Kublanovskaya and Ruhe dealing with problems of the form (2) was generalized in [13] and later [4] to cover the Jordan structure of a regular pencil $\lambda B - A$. Unsurprisingly, the overall complexity was of the same order $\mathcal{O}(rn^3)$ for r recovered generalized eigenvalues. In [2, 3], Beelen&Van Dooren actually tackle the case of an arbitrary pencil and provide backward stable methods of cubic complexity, which can be used to compute the form (7) for a regular pencil.

5. **Problem approach**
   Before considering working on general differential-algebraic systems of equations as they were defined previously, numerous steps will have to be taken. Each step will deal with a particular kind of problem, always related to the ultimate goal of numerically solving systems of the form (1).

   These intermediate problems belong to the field of numerical linear algebra and are typically explored through theoretical considerations first and then a strategy is implemented on computer. Usually, an algorithm is first designed theoretically and the mathematics show that the algorithm is supposed to do what is expected. However, the fact that computers work on finite precision arithmetic usually make the whole situation a lot more complex. It happens that calculations involving at the same time large quantities and smaller ones have the effect of disregarding those small values. Most of the time, this is not an issue as those small quantities have an equally small impact on the final values returned by the algorithm. However, this can also have a dramatic effect on the solution and one has to be very careful during implementation and result analysis. Usually, it is necessary to go back and forth between theory and implementation as the failure of the algorithm almost always has the potential to give new insights into the algorithm.

   When implementing an algorithm on a computer, numerous options can be considered. However, as a researcher in numerical linear algebra, the language MATLAB is a good choice. Far from being the most efficient when it comes to computational speed, it is nevertheless a powerful tool when you want to write a complex algorithm in a simple way. All the built-in functions and the intuitiveness of the language itself can save a lot of precious time to the programmer who can then focus on the behaviour of the algorithm. These are the reasons why the language I've been using so far and that I will use during most of the project is MATLAB.

6. **Expected results**
   As mentioned above, the natural way to study the dynamical behaviour of constrained physical systems is to use systems of differential-algebraic equations. Such physical systems include (but are not limited to) electrical circuits, mechanical systems and chemical reactions kinetics. Since constrained physical systems appear in so many situations, applications of the theory of differential-algebraic equations are virtually infinite. No application in particular has been considered yet but many options stay open.

7. **Phases of the project**
   The first phase of the project consists primarily in studying the QR algorithm. It is one of the most important algorithm of numerical analysis and a lot as been said and written about it for the last fifty years or so. A deep understanding of the algorithm, its properties and

its variants will be of paramount importance for the following research. Activities of this phase are mainly restricted to surveying the literature and synthesizing information in order to gain new insights into the algorithm. This phase is rather short and corresponds only to the beginning of the first year. At the time of writing this report, it could be said that this phase is already done, even if complementary information about the QR algorithm might have to be acquired later in the process.

Phase two focuses a problem that could be interesting to the research community and might have its importance in later phases of the research project. It's a problem that was first enunciated by Wilkinson in the late 1950s and, though he could not solve it at the time, several solutions have been proposed over the years. Given a symmetric tridiagonal matrix and an accurate approximation of one of its eigenvalues, the problem consists in finding the index of the largest entry of the corresponding eigenvector. When this index has been found, it is then quite easy to compute the whole eigenvector of interest. Several algorithms doing just that already exist and we are proposing a new way to do it. Hopes are that this new algorithm will show some advantages over the existing ones, like fastness or reliability of the solution. The first activity of this phase consists in understanding every step of the method. When this objective has been reached, the second activity can be started: implementation. If the algorithm is clear enough, implementation is quite direct. When a functioning implementation of the algorithm is at hand, the third activity can start: algorithm analysis. As stated previously, the finite precision arithmetic on which computers rely can have a strong impact on the behaviour of the algorithm. It is then essential to test the algorithm on different sets of data in order to get a glimpse of the way finite precision arithmetic influences the process. Unexpected results and careful study of the way data is handled throughout the algorithm can provide new insights into the functioning of the algorithm and the way it should be modified. This too corresponds to the first year of the PhD and progress is already well under way.

The third phase consists in modifying the procedure to handle non-symmetric tridiagonal matrices. Even if it might seem like a somewhat straightforward process, assuming the matrix of interest to be symmetric really facilitates many aspects of the algorithm and the extension to non-symmetric matrices will represent a significant amount of work. Many of the nice properties linked to symmetric matrices will have to be put on the side and the theory will most likely have to be created from scratch. This will be the direct follow-up to the previous case and will almost certainly be taken care of during the first year of the research project.

Phase four will consist in studying recent research by Dr. Mastronardi and Prof. Van Dooren on the subject of the QR algorithm (mainly [11]) and whether the activities carried out in the previous phases can be of any use there. This will hopefully be done during the first year.

Phase five will be concerned with studying the literature on the subject of matrix pencils and differential-algebraic systems of equations. Particular attention will be paid to the way the QR algorithm can be used in this problem. Once again, it is hoped that this phase will be completed during the first year of the PhD, most likely at the end.

Phase six is the largest by far and should occupy the two last years of PhD. It consists in extending the ideas of [11] to the subject of generalized eigenvalue problems. This extension is expected to be complicated and to require a lot of work. Generalized eigenvalue problems are directly linked to differential-algebraic equations and being able to compute the Weierstrass form of the corresponding matrix pencil will yield general solutions to the corresponding

differential-algebraic system of equations. Expectations are that the advantages displayed by Mastronardi and Van Dooren method for the problem of explicit systems of differential equations will have an equivalent when extended to the case of differential-algebraic systems of equations.

8. **Result evaluation**
   When assessing the quality of a numerical algorithm, the main aspects the need to be evaluated are the speed of the algorithm and the quality of the returned solutions.

   Speed is usually quite easy to evaluate as it suffices to compute how long the algorithm takes to complete its procedure. Size of the input data obviously have an influence and a statistical study should be conducted in order to link the size of the problem to the time it takes for the algorithm to give a solution. Such an analysis will provide the programmer with an empirical complexity that could be linked to the theoretical complexity (ideally, they should be similar). In terms of speed, comparison with other algorithms is difficult unless they are manually implemented as well (and this might represent a significant amount of work). Algorithms already implemented in MATLAB usually benefits from heuristics and small tricks that make the algorithm much faster then a raw implementation would have been. If the new algorithm appears to display some real assets over older ones, an efficient implementation (in C, for example) can be considered.

   When considering systems of differential-algebraic equations, assessing the solution can be tricky. Indeed, the solution would be a vector of functions and many ways could be used to evaluate the quality of the solution returned compared to the real solution. Integrals often have to be used in this regard and the evaluation procedure can be numerically expensive while not illuminating at all. Instead of working with the solution itself, we can work on the corresponding Weierstrass form. This form possesses all the information needed (since it gives rise to the numerical solution) and is easier to manipulate. Two aspects that have to be evaluated would be the quality of the approximated generalized eigenvalues and whether or not the algebraic and geometric multiplicities are correct (and, if they are not, how does that affect the solution).

9. **Possible reference persons external to the department**

   - Nicola Mastronardi, Research Director, Istituto per le Applicazioni del Calcolo, CNR, Bari, Italy
   - Paul Van Dooren, Professor of Mathematical Engineering, Université catholique de Louvain, Louvain-la-Neuve, Belgium

10. **References**
    See next page.

Bibliography

[1] K. M. Anstreicher and U. G. Rothblum, *Using Gauss-Jordan elimination to compute the index, generalized nullspaces, and Drazin inverse*, Linear Algebra and its Applications, 85 (1987), pp. 221—-239.

[2] T. 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 (1988), pp. 9—-65.

[3] ——, *Computational aspects of the Jordan canonical form*, in Reliable Numerical Computation, M. Cox and S. Hammarling, eds., Oxford Univ. Press, New York, 1990, pp. 57—-72.

[4] J. Demmel and B. Kågström, *Computing stable eigendecompositions of matrix pencils*, Linear Algebra & Applications, 88/89 (1987), pp. 139—-186.

[5] C. W. Gear, *The simultaneous numerical solution of differential-algebraic equations*, IEEE Trans. Circ. Theor., (1971).

[6] G. H. Golub and J. H. Wilkinson, *Ill–conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Rev., 36 (1976), pp. 578—-619.

[7] N. Guglielmi, M. Overton, and G. Stewart, *An efficient algorithm for computing the generalized null space decomposition*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 38–54.

[8] B. Kågström and A. Ruhe, *An algorithm for the numerical computation of the Jordan normal form of a complex matrix*, ACM Trans. Math. Software, 6 (1980), pp. 398—-419.

[9] L. Kronecker, *Algebraische reduction der schaaren bilinearer formen*, Stizungsber. Akad. der Wiss., Berlin, (1890), pp. 763–776.

[10] V. N. Kublanovskaya, *On solving the complete eigenvalue problem for a degenerate matrix*, USSR Computational Math. and Math. Phys., 6 (1968), pp. 1—-14.

[11] N. Mastronardi and P. Van Dooren, *Computing the Jordan structure of an eigenvalue*, SIAM J. Matrix Anal. Appl., to appear, (2016).

[12] A. Ruhe, *An algorithm for numerical determination of the structure of a general matrix*, BIT, 10 (1970), pp. 196—-216.

[13] P. Van Dooren, *The computation of Kronecker's canonical form of a singular pencil*, Linear Algebra and its Applications, 27 (1979), pp. 103—-140.

[14] K. Weierstrass, *Uber ein die homogenen funktionen zweiten grades betreffendes theorem, nebst anwendung desselben auf die theorie der kleinen schwingungen*, Monatsh. Akad. der Wissensch., Berlin, (1858), pp. 207–220.

[15] ——, *Zur theorie der bilinearen quadratischen formen*, Monatsh. Akad. der Wissensch., Berlin, (1867), pp. 310–338.

Courses and international summer schools

Courses

List of courses I'm planning on taking during the year 2017/2018:

Basic courses

- Project management (3 credits), Dr. Danilo Calvano.
- Empirical Methods and Data Analysis (3 credits), Prof. Fabio Q. B. Da Silva and Dr. Teresa Baldassarre.

Mathematics curriculum

- Elements of Calculus of Variations with Applications to the Study of Geodesics (4 credits); Teacher Prof. Anna Maria Candela.
- Dissipative hyperbolic equations (2 credits); Teacher Prof. Marcello D'Abbicco.
- Differential equations of fractional order (3 credits), Prof. Roberto Garrappa.
- Eigenvalues, computation of canonical forms and applications (4 credits), Dr. Nicola Mastronardi.

Computer science curriculum

- Mining Real-time Data Streams (4 credits), Prof. Herna Lydia Viktor and Prof. Donato Malerba.
- Empirical Methods and Data Analysis (3 credits), Prof. Fabio Q. B. Da Silva and Dr. Teresa Baldassarre.
- Machine Learning and Semantic Web (3 credits), Dr. Claudia d'Amato and Prof. Nicola Fanizzi.

International summer schools

- Gene Golub SIAM Summer School, June 17-30 2018, Breckenridge, Colorado, USA