UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

BACHELOR'S DEGREE IN INFORMATICS ENGINEERING

SPECIALIZATION IN COMPUTING

Numerically solving Hilbert's 16th problem

Author
Víctor GONZÁLEZ PRIETO

Supervisor
Grigori ASTRAKHARCHIK

January 2021

Abstract

The second part of Hilbert's 16th problem remains unresolved more than a century after he first formulated it. This Degree Final Project (TFG) is an attempt to bring the power of modern computers into the equation.

In the course of the project, we have designed and implemented a number of methods for exploring the phase space of two-dimensional quadratic systems in search of limit cycles.

These methods include computing heatmaps of the trajectories in the space, locating the regions in the space where the system's partial derivatives change sign, or classifying the trajectories in one region depending on whether they approach or escape from a given stationary point.

Although we have not reached a final solution for locating the limit cycles of a dynamical system, we have studied some possibilities and we propose an approach that may be more fruitful than the ones we had time to pursue during the project.

The end result is an application that is easy to use and configure, modular and extensible. We hope that it may serve as the springboard for further numerical studies of this problem.

Resum

La segona part del 16è problema de Hilbert roman sense resoldre més d'un segle després que el postulés per primer cop. Aquest Treball de Fi de Grau (TFG) és un intent de portar la potència dels computadors moderns al problema.

En el decurs del projecte, hem dissenyat i implementat una sèrie de mètodes per explorar l'espai de fase de sistemes bidimensionals quadràtics en cerca de cicles límit.

Aquests mètodes inclouen computar mapes de calor de les trajectòries de l'espai, localitzar les regions de l'espai on les derivades parcials del sistema canvien de signe, o classificar les trajectòries en una regió depenent de si s'apropen o s'allunyen d'un punt estacionari determinat.

Si bé no hem arribat a una solució definitiva per localitzar els cicles límit d'un sistema dinàmic, hem estudiat algunes possibilitats i proposem un apropament que podria ser més fructífer que els que hem tingut temps d'explorar durant aquest projecte.

El resultat final és una aplicació fàcil d'utilitzar i de configurar, modular i extensible. Tenim l'esperança que pugui servir com a punt de partida per a posteriors estudis numèrics del problema.

Resumen

La segunda parte del 16o problema de Hilbert permanece sin resolver más de un siglo más tarde de que lo postulara por primera vez. Este Trabajo de Fin de Grado (TFG) es un intento de llevar la potencia de los computadores modernos al problema.

En el transcurso del proyecto, hemos diseñado e implementado una serie de métodos para explorar el espacio de fase de sistemas bidimensionales cuadráticos en búsqueda de ciclos límite.

Estos métodos incluyen computar mapas de calor de las trayectorias del espacio, localizar las regiones del espacio donde las derivadas parciales del sistema cambian de signo, o clasificar las trayectorias en una región dependiendo de si se acercan o se alejan de un punto estacionario determinado.

Si bien no hemos alcanzado una solución definitiva para localizar los ciclos límite de un sistema dinámico, hemos estudiado varias posibilidades y proponemos un acercamiento que podría ser más fructífero que los que hemos tenido tiempo de desarrollar durante este proyecto.

El resultado final es una aplicación fácil de emplear y de configurar, modular y extensible. Tenemos la esperanza de que pueda servir como punto de partida para posteriores estudios numericos del problema.

Contents

List of Figures

List of Tables

1 Context

1.1 Introduction

This project is a graduation thesis for the Bachelor's Degree in Informatics Engineering from the Facultat d'Informàtica de Barcelona (FIB), part of Universitat Politècnica de Catalunya (UPC). The thesis' supervisor is Grigori Astrakharchik, from the Department of Physics. My specialization, and so the thesis', is Computing.

The project was proposed by the supervisor, seeing an opportunity to harness the capacity of modern hardware in order to explore the second part of Hilbert's 16th problem, which is among the few Hilbert's problems that remain entirely unresolved.

We will leverage the computing power of a GPU cluster to perform a numerical study of limit cycles in two-dimensional quadratic systems.

1.2 Terms and concepts

1.2.1 Dynamical systems

A dynamical system is one in which a function describes the time dependence of a point in a geometrical space. At any given time, from a given starting point, a dynamical system has a state represented by a point in state space (also known as phase space in some fields). For two-dimensional dynamical systems, that state space equates to the plane.

The function that describes what future states follow from the current state is called the evolution rule. This evolution is described in terms of the direction and the magnitude (e.g.: speed) from a given point in state space.

A dynamical system can be visualized as a vector field, with the vectors being the value of the evolution rule at their point of origin.

1.2.2 Limit cycles

A trajectory is the set of points in state space that are the future states resulting from a given initial state. In the domain of real numbers, a trajectory is a curve in state space.

A limit cycle is a closed trajectory in state space having the property that at least one other trajectory spirals into it either as time approaches infinity or as time approaches negative infinity. Limit cycles can only occur in nonlinear systems.

1.2.3 The second part of Hilbert's 16th problem

The Hilbert problems are a list of open problems proposed by David Hilbert at the Second International Congress of Mathematicians celebrated in Paris in 1900. [1]

In Hilbert's original formulation, the 16th problem splits into two parts. The first part is about the topology of real algebraic varieties and is researched in the field of real algebraic geometry. It is of no concern to this project.

The second part concerns the topology of limit cycles of dynamical systems. It is defined in [2] as follows.

Consider planar polynomial systems of the form

$$\dot{x} = P(x, y), \quad \dot{y} = Q(x, y), \tag{1}$$

where P and Q are polynomials in x and y. The question is to estimate the maximal number and relative positions of the limit cycles of system (1).

Let H_n denote the maximum possible number of limit cycles that system (1) can have when P and Q are of degree n. More formally, the Hilbert numbers H_n are given by

$$H_n = \sup\{\pi(P, Q) : \partial P, \partial Q \leq n\},$$

where ∂ denotes "the degree of" and $\pi(P, Q)$ is the number of limit cycles of system (1).

For $n = 2$, the system is of the form

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = a_1x^2 + b_1xy + c_1y^2 + \alpha_1x + \beta_1y, \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = a_2x^2 + b_2xy + c_2y^2 + \alpha_2x + \beta_2y, \end{cases} \tag{2}$$

[3] proves how it can be reduced to

$$\begin{cases} \dfrac{\mathrm{d}x}{\mathrm{d}t} = x^2 + xy + y, \\ \dfrac{\mathrm{d}y}{\mathrm{d}t} = a_2x^2 + b_2xy + c_2y^2 + \alpha_2x + \beta_2y. \end{cases} \tag{3}$$

For quadratic systems, Shi proposed a domain in the parameter space in which systems have 4 limit cycles [4].

1.3 Problem to be resolved

Even for the lowest, non-trivial Hilbert number with $n = 2$, the lower bound hasn't changed for 40 years. There's a lack of numerical approaches to Hilbert's 16th problem, as the vast majority of the research in the field (dynamical systems) is of analytical nature.

We aim to make the largest numerical study to date of the second part of Hilbert's 16th problem.

We are not setting a goal of raising that lower bound, because H_n may equal 4 for all we know. Finding configurations of parameters with 4 limit cycles that don't belong in already known domains would already be a very good result.

Furthermore, known examples with 4 limit cycles have 3 nested limit cycles and fourth limit cycle around a different center of equilibrium. Being able to find a system with 4 concentric limit cycles would be an even better result.

In a worse scenario, even one were we are unable to find a system with even 1 limit cycle, we will have set the foundations for a numerical study of the problem. A software project that may be improved upon.

1.4 Stakeholders

The main stakeholders are Grigori Astrakharchik, as **the supervisor for this project**, and me, as **its researcher and developer.** Grigori will provide guidance on the technical aspects and make sure that the project advances in the right direction.

The **scientific community** may benefit from the project, be it in the form of new results or from the program itself. In particular, researchers working in the field of dynamical systems could use a new result on the problem to induce new theories and methods, because a conclusive solution to Hilbert's 16th problem can only be analytical.

Furthermore, anyone wanting to launch a numerical study on the subject could **use our project as a reference** of where to start, which pitfalls to avoid, etc.

2 Justification

2.1 Previous studies

I have mainly studied the work of Leonov, Kuznetsov and others in [3, 5–7]. The focus of their research are methods for approaching the second part of Hilbert's 16th problem, with an emphasis on computational methods.

The main method they use is the computation of Lyapunov quantities. As far as I understood, they are a means for analyzing the behavior of a dynamical system in the neighborhood of a stability point, and can be applied to locate limit cycles in that neighborhood.

Of particular interest to me is their work in [7], which provides a Matlab program for the computation of the third Lyapunov quantity.

A few years before their research, Lynch explains in [8] the basics behind Lyapunov quantities and what is their use in the context of Hilbert's 16th problem. He also provides a Matlab program for the calculation of the first two quantities.

2.2 This work

These precedents are not quite useful to our project because they concern analytical methods, not numerical. For instance, their computation of Lyapunov quantities is in symbolic form. Also, being honest, the math involved is well beyond my reach.

Instead, we have access to a GPU cluster, so we can carry out a brute-force study of the problem that we hope will be helpful to those same mathematicians.

We will explore parameter space, generating random configurations of the parameters in system (3), and then scanning the resulting phase space in search of limit cycles.

As for the previous studies outlined here, we may consider including some of their methods in the later part of the project, provided there is time. But it is definitely outside of the project's MVP.

Lastly, a word on my choice of programming language for the project. I will use Python because I am familiar with it, it is versatile and good for fast-prototyping and it has a wealth of handy libraries, including mathematical methods and a CUDA front-end.

3 Scope

3.1 Objectives

We have defined the following objectives for the realization of the project:

- **Literature review**
  - Understanding the second part of Hilbert's 16th problem: studying limit cycles, basic concepts of dynamical systems, etc.
  - Searching for algorithms to better approach the problem.
- **Baseline version:** writing a program that can be validated against well-known results. With the following components:
  - Integrating trajectories: coding a program that given a configuration of the parameters of the system and a starting point, determines if that point belongs to a limit cycle.
  - Searching limit cycles: encapsulating the previous program into one that searches the limit cycles of a given configuration of the system parameters.
  - Visualizing limit cycles: representing limit cycles in phase space, to better convey our results.
- **Prototype version:** adapting the sequential version for parallel computing with CUDA. Obtaining results.
  - Parallelizing the sequential program with CUDA.
  - Random generation of parameters: a program that creates configurations of parameters that can be feed to the previous program.
  - Schema for experiment results: defining a format for the output of the previous programs. It should be easy to share, process and visualize.
  - Running the program to obtain results, as often as is allowed to and is reasonable.

- **Possible optimizations**
  - Architecture-aware programming: profiling the program to find its bottlenecks, removing them so the program can explore more configurations in less time. Also, testing algorithms that are specifically designed for GPUs.
  - Better algorithms: through achieving a better understanding of the literature. Possible improvements: heuristic generation of parameters, detection of limit cycle-less systems early, smarter localization of limit cycles.

3.2 Requirements

We have considered a few other requirements for the project:

- Correctness: the programs must be mathematically correct.
- Validatable against known results: as a weaker condition for correctness, we will provide tests that ensure that our programs work well with known results.
- Shareable results: their format must be lightweight, easy to visualize, and easy to process from another program. We may consider to share results in more than one format. This is a numerical study, so our results should definitely be accessible.
- Clean code: one of the possible outlets we consider for this project is as a component or a reference for further numerical studies. For that purpose, the code should be readable. This is also important in order for external stakeholders to be able to verify the program's correctness.

3.3 Potential obstacles and risks

We have identified the following obstacles and potential risks for the project:

- My lack of mathematical acumen: this obstacle has already hampered our progress with my slow review of the literature. Furthermore, it represents a possible risk if my lack of clarity on the matter results in incorrect programs. I trust that the valuable technical review of my supervisor will prevent any mishaps in this direction.
- My lack of experience with CUDA: I have never programmed for CUDA or GPUs. On the other hand, I'm pretty skilled at dealing with CPU parallel programs.
- Numerical precision: this is a two-pronged worry. On the one hand, a numerical approach is inherently inexact, it deals in approximations. On the other hand,

binary representation of decimal numbers is also lossy. For the first worry, we have already discussed the matter with my supervisor and he has proposed some methods (e.g.: predictor-corrector) that will keep the error in check. We also have the validation with known results to rely on. For the second worry, GPUs are engineered for floating point operations, so I trust that our use case will not be the one that breaks usability.

- Mishandling of results: we will have to be careful with keeping the output data of the programs safe and coherent. I'm particularly worried about changes on the output format (e.g.: reporting new heuristics for every considered system) rendering previous results obsolete.

4 Methodology and rigor

4.1 Methodology

My supervisor and I have been following a pretty loose Agile methodology so far. [9] We have a meeting once a week, where we review this period's progress and set immediate goals for the next week or so.

The time planning carried out for the GEP module, included in Chapter 5, has helped us define which tasks are required to reach a working prototype of our solution. With this, two thirds of the project are well-defined in terms of functional requirements, but any potential iterations over the prototype are yet to be decided.

But, even though we have clear requirements for this part, we are not going to follow a Waterfall approach. We will keep meeting to evaluate the project's progress and verbalize new goals. If nothing else because I am not certain that tasks will be completed in the time frames I have estimated, and having weekly meetings will let us adapt the planning if and when that happens.

I also think that we don't need to worry too much about adopting one particular agile methodology or another. If I am not mistaken, methods like Scrum [10] or Kanban [11] were mainly designed for teams. They make sense when the distribution of work is complex because several team members are concurrently doing work on a number of tasks. Our team does not present such complexity, as I am the one who is supposed to carry out all of the tasks, with my supervisor providing guidance on technical aspects and ensuring the project is on track.

As such, I will only explain a couple of guidelines that, in my experience, work well for me.

In the first place, I will try not to work on too many tasks in parallel. In particular, I will generally avoid juggling two or more development tasks for two reasons. Firstly, in order to minimize task switching costs. Secondly, in order to focus on completing tasks early, reach new milestones often.

In the second place, I will still do documentation tasks on the side. The Thesis Report

is one of the required outcomes for this project, and I plan to compile it throughout the project. Documentation tasks will include technical definitions of the functionalities have to implement, implementation notes, comments on any results we obtain, etc. And also management control aspects as explained in Chapter 6.

In any case, I will hardly have more than two or three tasks going on at any given time.

4.2 Monitoring tools

As previously mentioned, our most important monitoring tool is a proved and tested practice: weekly meetings. It consists of a Skype meeting on Fridays, where I report my recent progress to the supervisor and we discuss on how to proceed next.

We complement that with communication via e-mail for those issues that should not wait until the weekly meeting.

I am also sharing drafts of the documentation I write through the Overleaf web application, so the supervisor can add his comments. We have also discussed that I will share the source files of the project via e-mail. If need be, we will switch to GitHub.

In terms of keeping track of pending tasks, I had been writing and updating my work plan on a .txt file, which is a bit chaotic. The tasks I annotate there are much more granular than the ones defined in Chapter 5, in the order of minutes rather than days of work. I mainly use it for planing my work in the next few hours.

I am currently migrating pending tasks to Asana [12], a web and mobile application with a pretty simple workflow but much better structured than the lines of text in my .txt work plans.

As a matter of fact, I use Jira for my personal to do list, because it includes tasks from a variety of domains and I need to write proper descriptions for them so I do not forget. But I prefer to use Asana here because the focus will be less on curating the tasks to do and more on actually getting things done.

A less bloated solution like Asana will encourage us to keep task descriptions to a bare minimum (we are two people in the project, we already know what needs to be done) and focus on development.

Once I have finished setting Asana up, I will share the project with my supervisor, so he may know what I am currently working on and which tasks are finished.

5 Time planning

The expected duration for the project is of approximately 140 days, in the period comprised between September 14 (beginning of the fall quarter) and January 25 to 29 (oral defence dates).

Furthermore, the graduation project for my bachelor's degree is assigned 18 ECTS credits, which equate to 450 hours of work. During this period, I'm committing an average of 4 daily hours to the project, including weekends and holidays.

5.1 Resources

In this section I am going to summarize the resources that I will use in the project's tasks.

In terms of human resources, this project requires the following job roles: a Junior Research Engineer, a Senior Research Engineer and a Project Manager.

I have taken the two research engineer roles from Barcelona Supercomputing Center (BSC) post descriptions[1]. The Junior Research Engineer (JRE) is supposed to handle most of the programming workload. The Senior Research Engineer (SRE henceforth) is responsible for the literature review, the technical definitions and ensuring the correctness of the programs.

I have included a Project Manager role (PM) because the management part of this project is structured much like in a company project, and I don't think Research Engineers would carry out that type of tasks.

I am using the BSC posts as a reference because the tasks they carry out on the daily should be very similar to the ones in this project, with how we are tackling a scientific problem by computational methods. They are also based in Barcelona, which makes the human resource costs estimated in Chapter 6 more accurate.

The PM and SRE responsibilities are shared between my supervisor and me, but I am not going to factor the supervisor as a human resource for simplicity's sake. This choice

[1]BSC calls "Research Engineer" what I call "Senior Research Engineer". I added the "Senior" prefix for easier reference.

is in accordance with available projects from previous years.

In terms of material resources, I will differentiate between hardware, software and workspace resources. The later are generic and will not be considered in this section, but they will be properly identified in Chapter 6.

In terms of hardware, we will only use two items: the GPU unit, composed of two Nvidia Titan V graphics cards, and my work laptop. The laptop is a required resource for every task in the project, so I am not going to include it when discussing a task's specific resources. Nor am I going to list the operating system it runs (Ubuntu 18.04 LTS).

For the rest of software resources, I am grouping them in four suites:

- Development suite
  - Python: the programming language that we will use for the project.
  - CUDA libraries: to interface with the GPU.
  - Atom: my usual IDE.
  - git: the most extended version control system.
- Documentation suite
  - Overleaf: web application for editing LaTeX documents.
  - Google Sheets: with the LatexKit extension, this integration makes inputing table data for LaTeX much easier. Included in Google Drive.
  - GanttProject: desktop application for designing Gantt diagrams.
  - Google Slides: for designing the oral defence's slide deck. Also in Google Drive.
- References suite
  - Mendeley: desktop application for reading papers and keeping track of references.
  - eBIB: a bookmarklet provided by UPC to access the electronic resources of its digital library.
- Management suite
  - Skype: video conferencing application where we host our meetings.
  - Gmail: web app that I use for e-mail communication with my supervisor.
  - Asana: web and mobile application that I use as a task list.

5.2 Description of tasks

In this section I am going to describe the tasks in which we have separated the work required to complete the project. Towards the end of the section, Table 1 shows a summary of the specific resources required for every task, while Table 2 shows a summary of their time constraints.

5.2.1 Literature Review tasks

LR1 Understanding the problem

Studying the mathematical concepts behind the second part of Hilbert's 16th problem. This task was completed before the formal beginning of the project and it took me about 20 hours.

Human resources: SRE (20h)

Material resources: References suite

LR2 Researching methods

The next logical step after LR1, studying existing literature on the problem in search of methods for approaching it. From a few papers that came up in our initial search and diving into their references and citations by other papers. This part took me 50 hours.

Human resources: SRE (50h)

Material resources: References suite

5.2.2 Project Management tasks

The tasks in this block include the project management documents and the meetings.

PM1 Weekly meetings

With the supervisor, about one hour every week for monitoring the progress of the project and discussing our next steps. It takes me another hour to compile and process the meeting minutes.

Human resources: PM (1h/week), SRE (1h/week)

Material resources: Management suite

PM2 Context and scope document

Documenting the context and scope of the project. It took me about 20h. This task came after the Literature Review tasks.

Human resources: PM (15h), SRE (5h)

Material resources: Documentation and References suites

PM3 Time planning document

Documenting the time planning for the project. I estimate this task at 15h. This task depends on [PM2].

Human resources: PM (15h)

Material resources: Documentation suite

PM4 Budget and sustainability document

Documenting the budget for the project and analyzing its sustainability. I estimate this task at 10h. This task depends on [PM3].

Human resources: PM (10h)

Material resources: Documentation suite

PM5 Initial milestone document

Bringing together the three previous documents into a final document for the Initial Milestone of the project. I am supposed to apply the partial document's feedback provided by the GEP tutor, and I anticipate that I will need to do an overhaul of some of the chapters.

I estimate this task at 20h. This task depends on [PM2-4].

Human resources: PM (15h)

Material resources: Documentation and References suites

PM6 Continuous documentation

This task represents the ongoing documentation of the project as it progresses. Most of it will be technical documentation before and after implementation, but it also includes

management control tasks.

I'll start performing this task after reaching the Initial Milestone because I don't have the bandwidth to do development while I am writing the Project Management documents.

Human resources: SRE (2h/week), PM (1h/week)

Material resources: Documentation and Development suites

PM7 Follow up meeting

A special meeting with the supervisor that marks the Follow up Milestone of the project. I am to prepare a progress report and submit it to the supervisor for qualification. I estimate this task at 20h.

This task depends on [PM5] and [PT3-6], as it would be ideal to hold the meeting with the project's prototype finished or nearing completion.

Human resources: PM (10 h), SRE (10h)

Material resources: Management and Documentation suites

5.2.3 Baseline tasks

The following tasks consist of the development of sequential programs that will be used as a reference throughout the duration of the project: validating and displaying results, serving as the reference for the parallel programs, etc.

It is worth saying that all development tasks include testing and/or validation by default.

BL1 Integrating trajectories

Coding a program that, given a configuration of the system's parameters and an starting point, integrates the trajectory of the function. Then detect if it is a limit cycle. We will validate this program by using known results. This task depends on the knowledge acquired in [LR2]. I estimate this task at 10h.

Human resources: JRE (10h)

Material resources: Development suite

BL2 Plotting limit cycles

Coding a program that, given a configuration of the system's parameters and a set of starting points, plots their trajectories in the plane. Particularly, for points belonging to a limit cycle. This task depends on the trajectories calculated in [BL1]. I estimate this task at 5h.

Human resources: JRE (5h)

Material resources: Development suite

5.2.4 Prototype tasks

The following tasks drive the project to its MVP of a numerical study of the problem. Approaches and algorithms will probably be naive, but they will bring the computational power of CUDA into the equation.

PT1 CUDA development environment

Setting up the development environment for writing CUDA programs. Including the utilities for transferring those programs to the GPU-connected host, running them and then recovering their output. I estimate this task at 10h.

Human resources: JRE (10h)

Material resources: GPU, Development suite

PT2 Learning CUDA programming

Learning CUDA programming by means of tutorials, and making some toy programs. This task depends on [PT1]. I estimate this task at 20h.

Human resources: JRE (20h)

Material resources: GPU, Development suite

PT3 Parallelizing the integration of trajectories

Coding a CUDA program that launches an array of trajectory integrations in a GPU, from different starting points. Then retrieve the results for each node and process them in search of limit cycles. This task depends on [BL1], as it will use that program as a

reference. It also depends on the knowledge acquired in [PT2]. I estimate this task at 30h.

Human resources: SRE (10h) JRE (20h)

Material resources: GPU, Development suite

PT4 Parameter configurations generator

Coding a sequential program that generates configurations of parameters to feed the programs in [BL1] [PT3], though it doesn't necessarily depend on their completion. I estimate this task at 5h.

Human resources: JRE (5h)

Material resources: Development suite

PT5 Formalizing output data

Defining a JSON schema for the results. Defining a summary csv table with the best results, etc. I estimate this task at 15h. It depends on [PT3].

Human resources: SRE (5h), JRE (10h)

Material resources: GPU, Development suite

PT6 Gluing everything together

Integrating the programs from [PT3-5] with a script that automates execution on the GPU, from creating parameters for the problem, to exploring them in search of limit cycles, to processing those results and saving them in files that I can export. I estimate this task at 10h.

Human resources: JRE (10h)

Material resources: GPU, Development suite

5.2.5 Optimization tasks

The tasks in this block will optimize the individual components of the broader program: a more intelligent system generation, broader plane exploration, etc.

These tasks are purposely left open because we don't know the course the project will take after completing the prototype. It will depend on the mathematical methods we are

able to glean from previous studies, on our understanding of GPU-aware algorithms, etc.

Consequently, we are defining the following tasks in terms of generic iterations that will take 1-2 weeks of work.

IT1 Iteration 1

This iteration will take 40h, 2 weeks worth of development time. It depends on [PT6].

Human resources: SRE (10h), JRE (30h)

Material resources: GPU, Development and References suites

IT2 Iteration 2

From this iteration on, as we are closing on the finishing date for the project, we will reduce the time of every iteration to 20h, or a week of work. It depends on [IT1].

Human resources: SRE (5h), JRE (15h)

Material resources: GPU, Development and References suites

IT3 Iteration 3

Same reasoning. It depends on [IT2].

Human resources: SRE (5h), JRE (15h)

Material resources: GPU, Development and References suites

Table 1: Summary of the tasks' resources.

Code	Name	Human Resources (h)	Material Resources
LR1	Understanding the problem	SRE (20)	Refs
LR2	Researching methods	SRE (50)	Refs
PM1	Weekly meetings	PM (1/w), SRE (1/w)	Mgmt
PM2	Context and scope document	PM (15), SRE (5)	Doc, Refs
PM3	Time planning document	PM (15)	Doc
PM4	Budget and sustainability document	PM (10)	Doc
PM5	Initial milestone document	PM (15)	Doc, Refs
PM6	Continuous documentation	SRE (2/w), PM (1/w)	Doc, Dev
PM7	Follow up meeting	PM (10), SRE (10)	Doc, Mgmt
BL1	Integrating trajectories	JRE (10)	Dev
BL2	Plotting limit cycles	JRE (5)	Dev
PT1	CUDA development environment	JRE (10)	GPU, Dev
PT2	Learning CUDA programming	JRE (20)	GPU, Dev
PT3	Parallelizing the integration	SRE (10), JRE (20)	GPU, Dev
PT4	Parameter configurations generator	JRE (5)	Dev
PT5	Formalizing output data	SRE (5), JRE (10)	GPU, Dev
PT6	Gluing everything together	JRE (10)	GPU, Dev
IT1	Iteration 1	SRE (10), JRE (30)	GPU, Dev
IT2	Iteration 2	SRE (5), JRE (15)	GPU, Dev
IT3	Iteration 3	SRE (5), JRE (15)	GPU, Dev
PC1	Thesis report	PM (30), SRE (20)	Doc
PC2	Oral defence	PM (15)	Doc

Source: compiled by the author.

5.2.6 Project Closure tasks

PC1 Thesis report

Bringing together the project management documents, the continuous technical documentation and making a final thesis report. I estimate this task at 50h. It depends on all prior tasks but I am going to specifically list [PM7] and [PT6].

Human resources: PM (30h), SRE (20h)

Material resources: Documentation suite

PC2 Oral Defence

Making the slide deck for the oral defence and presenting in front of the thesis jury. The presentation is supposed to have a duration of less than 30 minutes. I estimate this task at 15h. It depends on [PC1].

Human resources: PM (15h)

Material resources: Documentation suite

Table 2: Summary of the tasks' time constraints.

Code	Name	Dependencies	Estimate (h)
LR1	Understanding the problem		20
LR2	Researching methods	LR1	50
PM1	Weekly meetings		2/w
PM2	Context and scope document	LR2	20
PM3	Time planning document	PM2	15
PM4	Budget and sustainability document	PM3	10
PM5	Initial milestone document	PM2-4	15
PM6	Continuous documentation	PM5	3/w
PM7	Follow up meeting	PM5, PT3-6	20
BL1	Integrating trajectories	LR2	10
BL2	Plotting limit cycles	BL1	5
PT1	CUDA development environment		10
PT2	Learning CUDA programming	PT1	20
PT3	Parallelizing the integration	BL1, PT2	30
PT4	Parameter configurations generator		5
PT5	Formalizing output data	PT3	15
PT6	Gluing everything together	PT3-5	10
IT1	Iteration 1	PT6	40
IT2	Iteration 2	IT1	20
IT3	Iteration 3	IT2	20
PC1	Thesis report	PM7, PT6	50
PC2	Oral defence	PC1	15
	TOTAL		471

Source: compiled by the author.

5.3 Gantt diagram

Figure 1 shows a Gantt diagram for the project's planning.

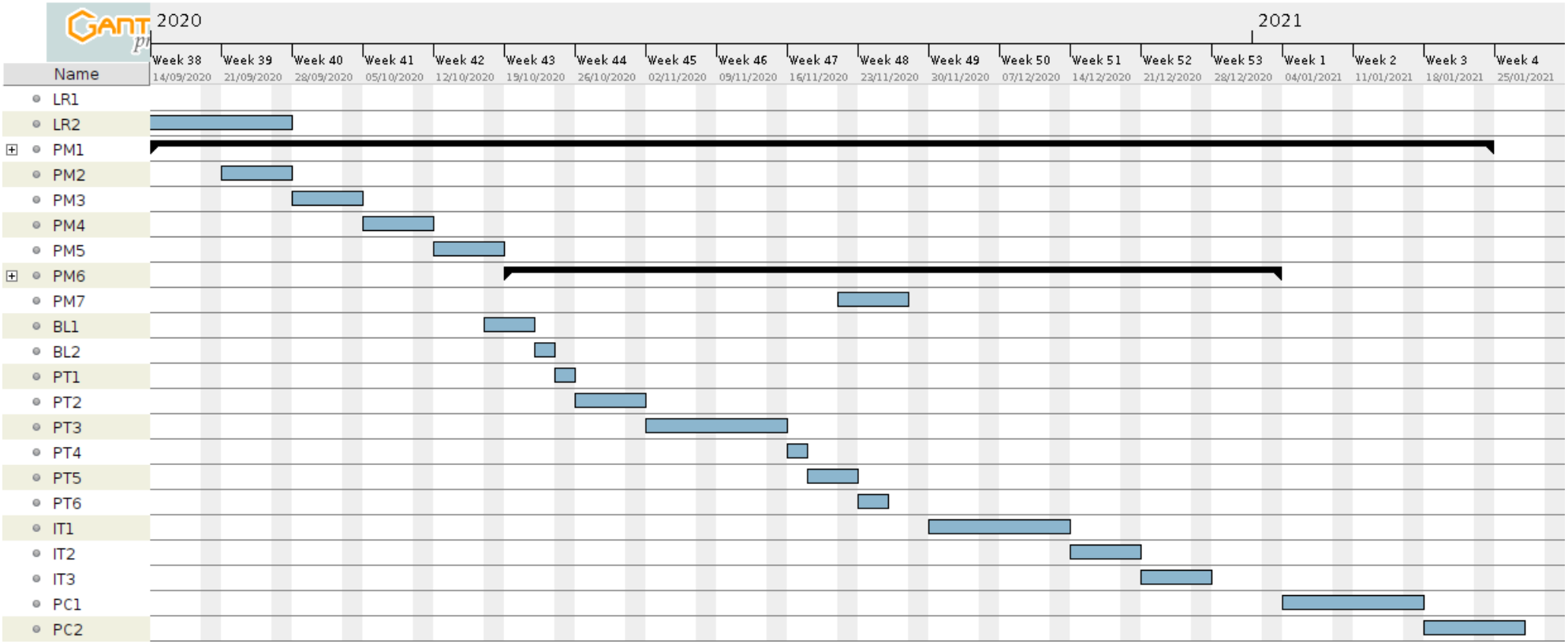

Figure 1: Gantt diagram of the project.
Source: compiled by the author.

5.4 Risk management

5.4.1 Analysis of risks

I am going to discuss in this section about the potential obstacles and risks defined in Subsection 3.3. How probable they are, how much of an impact would they have on the project, what measures am I taking to prevent or mitigate them.

First, a couple of the identified obstacles and risks will be prevented with dedicated tasks, with ample time to spend. It's the case with the obstacle presented by my lack of experience with CUDA programming, and with the risk of mishandling the numerical study's results, which will be tackled in [PT2] and [PT5] respectively.

These tasks have been allocated 20 and 15h respectively, so a week's worth of technical work will be spent on preventing those risks. The tasks are also strategically scheduled just before the phase of the project where their associated risk has a chance to rear its head. As such, regardless of their potential impact, their probability is negligible.

Another worry was my lack of mathematical acumen and how it may have a bad impact on the correctness of programs, thus invalidating our results. This risk can be broken into two different scenarios: that any potential bugs are noticed by us (Grigori or me) during the course of the project, or that they are detected by external stakeholders after we have finished the project.

For the first scenario, the impact would mainly be one of time. We would need to fix the bug and rerun simulations in order to obtain new results. Any results from the inception of the bug would be invalidated, but I recon that rerunning is not much of an added cost if the GPU cluster is still available.

For the second scenario, it would invalidate the results of the numerical study, but it would not necessarily render the project useless. Firstly, it is not rare in research for previously accepted results to be proven wrong, yet that failure still being useful information. Secondly, our project would still provide a working, if incorrect, program that can perform its intended function once it is patched. Future parties interested in this problem may still be able to use our program.

In any case, we fully intend to validate often, against known results, and more so if we obtain promising results. Furthermore, I recon that, with the supervisor's help, we will be able to stop any mathematical bugs in its tracks.

Table 3 shows a summary of the risks discussed in this subsection.

Table 3: Summary of risk elements.

Risk element	Probability (%)	Impact (1-10)	Weight (1-10)
Inexperience with CUDA	10	2	0,2
Mishandling of data	5	7	0,35
Bugs detected by us	20	3	0,6
Bugs detected by others	10	6	0,6

Source: compiled by the author.

5.4.2 Alternative plans

If the proposed prevention mechanisms don't work, or if other obstacles pop up that we didn't expect, we will consider the following two alternative plans.

Firstly, doing a reshuffle of the plan that sacrifices tasks from the Optimization block. Those iterations are defined very loosely in order to be flexible. I may use that time for optimization, but also for debugging, for finishing the prototype if it is overdue, etc.

It is also with this flexibility in mind, that I have left one week without tasks before work on the thesis report starts (see the Gantt). It gives me a buffer zone of sorts, in terms of time availability, where I can allocate unplanned tasks if need be. This reshuffle may also include overtime, perhaps going from 4 hours of work per day to 6-8 hours.

In terms of resources, it would imply a larger expense of human resources (and perhaps in different roles than previously planned too) and material resources (the GPU specially). It is also worth saying that the quality of the thesis would probably take a hit from having to complete the project against the clock.

Secondly, revising the scope of the project, making it less ambitious in order to be able to finish the project in its alloted time.

5.5 Deviation from the original plan

I severely underestimated the difficulty of making an algorithm that finds the limit cycles of a system. It turned out to be a complex problem in its own right and most of the project's work has been dedicated to this task. In the end I was unable to find a satisfactory solution.

The project still constitutes a valid exploration of possible approaches to that problem, and I extracted a few valid conclusions from it.

In regards to the time dedication of each role, I would say that it has been roughly the same as it was estimated in regards to the original planned tasks, so there is no need to update the estimations.

6 Budget

6.1 Identification of costs

This budget is structured into the following blocks: costs per activity, general costs, contingency and incidentals. The following subsections detail the costs that go into each of these categories.

6.1.1 Costs per activity

This category covers the staff cost for each of the planned tasks described in Section 5.2. It considers the job profiles involved in those tasks, with their time investment for the task, as it is summarized in Table 1. Subsection 6.2.1 estimates these costs pertaining to human resources.

6.1.2 General costs

This category comprises a number of costs that we're estimating globally, without calculating their cost per activity. In particular: the amortization costs of hardware and software resources and the expenses derived from my workspace.

Table 4 lists the hardware resources I plan to use, with their acquisition cost (VAT included). Subsection 6.2.2 estimates the amortization costs for these items.

Table 4: Acquisition cost of the hardware resources.

Hardware resource	Quantity	Price (€)	Cost (€)
Lenovo Ideapad 720S-13IKB	1	699,99	699,99
NVIDIA Titan V	2	2.547,65	5.095,30

Source: compiled by the author.

Table 5 shows the lack of acquisition costs for the software resources.

Table 5: Acquisition cost of the software resources.

Software resource	Price (€)	Comments
Ubuntu 18.04 LTS	0,00	Free software
Python	0,00	Free software
CUDA libraries	0,00	Free software
git	0,00	Free software
Overleaf	0,00	Freemium web application
Google Drive	0,00	Freemium web application
Gantt Project	0,00	Free software
Mendeley	0,00	Freemium software
eBIB	0,00	Free for UPC students
Skype	0,00	Freemium software
Gmail	0,00	Freemium web application
Asana	0,00	Freemium web application

Source: compiled by the author.

As for workspace costs, I will provide estimates for the part of my household expenses tied to the project in Subsection 6.2.3.

I am not going to consider any traveling expenses because I am working from home due to the ongoing COVID-19 pandemic.

6.1.3 Contingency

This budget item consists of a generic cost overrun provided in order to deal with unforeseen obstacles and expenses. We calculate it as a percentage of the total costs (CPA + GC), and common rates in the IT sector range from 10 to 20%.

I will allocate a contingency of 20% because software development projects never quite pan out as they were planned and I am not very good at estimating task durations.

6.1.4 Incidentals

This category covers the potential costs that may originate from the risks and obstacles discussed in Section 5.4, taking into account their probability and their economical impact.

Subsection 6.2.4 makes an estimation of the incidental costs for those risks and obstacles that have a significant chance of happening.

6.2 Cost estimates

In this section I am going to provide estimates for the different budget items.

6.2.1 Human resource costs

Firstly, in order to estimate the cost of the project's human resources, we need to know the hourly salary for each of the three roles required for the project. I will use data from glassdoor.com, a widely used website where current and former employees anonymously review companies, including on matters of pay.

For the two Research Engineer roles, I am going to use salary data on the equivalent BSC posts. [13, 14] For the Project Manager role, I am going to use salary data from IT Project Managers working in Barcelona. [15]

Table 6 presents that data and calculates the data we are interested in, hourly cost. Salaries reported on Glassdoor are gross, so we need to calculate their net cost by adding Social Security expenses.

Table 6: Wages for every job profile.

Job profile	Yearly Salary (€)	Yearly Cost (€)	Hourly Cost (€)
Project Manager	46.492,00	62.764,20	31,38
Senior Research Engineer	26.427,00	35.676,45	17,84
Junior Research Engineer	22.693,00	30.635,55	15,32

Source: compiled by the author.

Table 7 breaks down the cost per activity taking into account the hourly cost of human resources and combining it with their dedication to every task as represented in Table 1.

Table 7: Human resource costs per activity.

Task Code	PM (h)	SRE (h)	JRE (h)	Cost (€)
LR1		20		356,80
LR2		50		892,00
PM1	19	19		935,18
PM2	15	5		559,90
PM3	15			470,70
PM4	10			313,80
PM5	15			470,70
PM6	22	11		886,60
PM7	10	10		492,20
BL1			10	153,20
BL2			5	76,60
PT1			10	153,20
PT2			20	306,40
PT3		10	20	484,80
PT4			5	76,60
PT5		5	10	242,40
PT6			10	153,20
IT1		10	30	638,00
IT2		5	15	319,00
IT3		5	15	319,00
PC1	30	20		1.298,20
PC2	15			470,70

Source: compiled by the author.

6.2.2 Hardware amortization

As we saw in Subsection 6.1.2, we do not plan to spend money in any particular software resource. As such, I am going to list the cost of each software resource at 0 € and focus on estimating the amortization of the hardware resources.

The Spanish Treasury allows the amortization of hardware in 3 to 4 years, at which point it becomes obsolete and you have to substitute it.

I am going to calculate the amortization period of my personal laptop at 4 years, because I have been using it for 3 years and counting.

I am unable to find any references on the expected lifespan (in hours) of the NVIDIA Titan V, or any other scientific computing GPUs. There are some forum discussions estimating the lifespan (in years) for consumer market graphics cards, but I highly doubt those make for a good comparison.

In conclusion, I am going to use the following simplified formula for all hardware items:

$$\begin{aligned}\text{amortization (€)} &= \text{price} \times \frac{1}{4 \text{ years}} \times \frac{1 \text{ year}}{250 \text{ working days}} \times \frac{1 \text{ day}}{8 \text{ working hours}} \times \text{time used} \\ &= \text{price (€)} \times \frac{1}{8000 \text{ working hours}} \times \text{time used (h)}\end{aligned}$$

I require my laptop for every single task so its time usage is of 476 hours, which corresponds to the total estimated time for the tasks summarized in Table 2. On the other hand, I don't expect to run programs on the GPU for more than 20h, and that is already a very generous estimate.

Table 8 summarizes these calculations.

Table 8: Hardware resources amortization.

Hardware resource	Cost (€)	Time used (h)	Amortization (€)
Lenovo Ideapad 720S-13IKB	699,99	476	41,65
NVIDIA Titan V (x2)	5.095,30	20	12,74

Source: compiled by the author.

6.2.3 Workspace costs

It is good to remember here that the project's duration is supposed to be approximately 140 days, and that I have a dedication to the project of 4h per day, on average, regardless of weekends or holidays.

My monthly rent amounts to 357 €, so the total expense during project hours is:

$$357 \text{ €} \text{ per month} \times \frac{140 \text{ project days}}{30 \text{ days per month}} \times \frac{4 \text{ project hours}}{24 \text{ hours per day}} = 277{,}67 \text{ €}.$$

The electricity and gas bill comes to 175 € every two months, on average, and we're three people living under the same roof. So my contribution during project hours is:

$$(175 \text{ €} \text{ / 2 months / 3 tenants}) \times \frac{140 \text{ project days}}{30 \text{ days per month}} \times \frac{4 \text{ project hours}}{24 \text{ hours per day}} = 22{,}69 \text{ €}.$$

As for the internet bills, we spend 29 € monthly, also divided between the three of us:

$$(29\ € \text{ per month} / 3 \text{ tenants}) \times \frac{140 \text{ project days}}{30 \text{ days per month}} \times \frac{4 \text{ project hours}}{24 \text{ hours per day}} = 7{,}52\ €\,.$$

6.2.4 Estimation of incidentals

I am going to consider the alternative plans defined in Subsection 5.4.2 for this category.

First, the reshuffling of tasks in December in order to finish previous tasks that are overdue, with a probability of 30%. The costs are derived from the human resources' overtime and more usage of the material resources.

I am going to estimate this as if two more weeks were added to the project:

$$\text{Reshuffling of tasks } (€) = \text{Total Costs } (€) \times \frac{14 \text{ extra days}}{140 \text{ project days}} \times 0.3 \text{ probability}$$

Second, the revision of the project's scope. While it would make for a less valuable end result, I do not think that the costs would increase. As such I am estimating its potential cost at 0 €.

6.3 Budget summary

Table 9 presents a summary of the project's budget.

Table 9: Project budget.

Budget item	Amount (€)
[LR1] Understanding the problem	356.80
[LR2] Researching methods	892.00
[PM1] Weekly meetings	935.18
[PM2] Context and scope document	559.90
[PM3] Time planning document	470.70
[PM4] Budget and sustainability document	313.80
[PM5] Initial milestone document	470.70
[PM6] Continuous documentation	886.60
[PM7] Follow up meeting	492.20
[BL1] Integrating trajectories	153.20
[BL2] Plotting limit cycles	76.60
[PT1] CUDA development environment	153.20
[PT2] Learning CUDA programming	306.40
[PT3] Parallelizing the integration	484.80
[PT4] Parameter configurations generator	76.60
[PT5] Formalizing output data	242.40
[PT6] Gluing everything together	153.20
[IT1] Iteration 1	638.00
[IT2] Iteration 2	319.00
[IT3] Iteration 3	319.00
[PC1] Thesis report	1298.20
[PC2] Oral Defence	470.70
Total CPA (Costs Per Activity)	10069.18
Lenovo Ideapad 720S-13IKB	41.65
NVIDIA Titan V	12.74
Ubuntu 18.04 LTS	0.00
Development software suite	0.00
Documentation software suite	0.00
References software suite	0.00
Management software suite	0.00
House rent	277.67
Electricity and gas bills	22.69
Internet bills	7.52
Total GC (General Costs)	362.27
Total Costs (Total CPA + Total GC)	10431.45
Contingency (20%)	2086.29
Total CPA + GC +Contingency	12517.74
Reshuffling of tasks	312.94
Scope revision	0.00
Total Incidentals	312.94
TOTAL:	**12830.68**

Source: compiled by the author.

6.4 Management control

It is common for task estimations to not be perfectly fulfilled, and so the project running into more expenses than estimated. This section provides a few indicators that I will use to control any potential budget deviations during the execution of the project.

For human resource costs, or costs per activity, I will keep track of the time spent executing each task, and I will calculate its cost deviation as follows:

$$\text{CPA deviation} = \text{estimated cost} \times \left(\frac{\text{actual time}}{\text{estimated time}} - 1 \right)$$

It is not a perfect formula because I am not taking into account that, on tasks that require several job profiles, each profile's overtime need not be in proportion to their estimated dedication to the task. In that case, the cost of overtime hours cannot be calculated as a function of the cost I estimated with the original distribution of hours. I am still choosing this formula because it is a good enough approximation, considering that I am wearing all the hats for the project.

The total CPA deviation will be calculated like this:

$$\text{accumulated CPA deviation} = \sum_{i \in ct} \text{CPA deviation}_i$$

where ct refers to completed tasks. We can use this formula to calculate the total CPA deviation at the end of the project, but also for the partial value of the indicator at any point during the project.

For hardware amortization costs, the general formula is as follows:

$$\text{HW amortization deviation} = \text{estimated amortization} \times \left(\frac{\text{actual usage}}{\text{estimated usage}} - 1 \right)$$

I will derive the actual usage of the laptop from the aggregated time of completed tasks, just like its estimated usage comes from the sum of time estimations for those tasks.

The GPU is a bit more tricky because I have not made an estimation of usage per task. I will keep track of usage more carefully because it is an external resource, and calculate the deviation at the end of the project.

For workspace costs, I will keep track of any significant deviations between the actual bills and the averages per bill that I have used in my estimations.

As for incidental events, if they do happen, I will compute their costs as if they were any other task (i.e.: human resource costs, hardware amortizations, etc.), and we will use the incidentals fund to cover them. If and when it runs out, we will use the contingency fund. The costs for any unforeseen event will also be covered by the contingency fund.

At the end of the project, I will compare the total deviation of the budget with the contingency margin I provisioned at the beginning. But we can take a look at the indicators at any given time, so I will also make a partial analysis of deviations in the report for the Follow up Milestone.

6.5 Final budget

Because I did not go past the stage of implementing a program that is able to locate limit cycles, I never started the parallelism tasks. As such, I have not used the GPU cluster.

Nevertheless, the amortization cost I had estimated for it was very small (12€) and so the impact on the budget is minimal.

In terms of the cost of human resources, while there are a lot of tasks that were never executed, in their place I have spent a lot of time in the task of designing, implementing and testing algorithms for locating the limit cycles of a dynamical system.

As mentioned in section 5.5, the time dedication for each role was roughly the same as it was estimated when accounting for the initial plan, so there is no need to update the cost of human resources.

7 Sustainability

This chapter consists of two parts. First, a summary of the self-evaluation performed while answering the EDINSOST survey[1]. Then, my answers for the questions in the matrix of sustainability for the bachelor's thesis. [16]

7.1 Self-assessment

I am pretty familiar with most of the concepts mentioned in the survey, from broad concepts like social justice or equity to more particular concerns such as accessibility or circular economies. I have to admit, though, that I am more attuned to social justice than to environmental sustainability.

It is not that I ignore that our planet is on a countdown. Or rather, our survival on it, and also most other species'. But it just does not hit as close to home as people struggling because of inequality.

That does not mean I fail to do my part. I am a well-behaved citizen, recycling anything that I know can be recycled and asking Google (and Ecoembes) when I am not sure.

I would argue that this attention imbalance between social and environmental concerns works for the best, because as an software engineer I am much more likely to make a disruptive impact on the social fabric than I am to cause environmental harm.

On the one hand, if a server or a data center somewhere causes any environmental damage, it is the responsibility of the people who designed and maintained that infrastructure. At most, I may exercise my responsibility when choosing infrastructure providers, looking at their environmental footprint and track record. And that is assuming that I am high enough in the IT department's hierarchy to hold any sway over that that decision.

On the other hand, the solutions we develop often have the side effect of rendering jobs redundant, the people who did those jobs unemployed. Our algorithms can also turn professions into precarity (e.g.: the gig economy brought about by the success of delivery apps).

[1]Link: goo.gl/kWLMLE

That said, I am aware that any low to mid-ranking employee will be powerless to stop socially unfair decisions from management. Even in the face of a violation of human rights, the only "fair" course of action given to you, that is viable, is refraining from participating, walking out of the door. If nothing else because you probably signed an NDA when you joined the company.

As for economic sustainability concerns, I am only interested in them insomuch as they overlap with social sustainability concerns (e.g.: departments of my company earning significantly less than others, regardless of talent and effort).

For other matters, such as economic viability, I'm comfortable with leaving them in the hands of the business departments of the company.

7.2 Economic dimension

PPP: Have you estimated the cost of undertaking the project?

Yes, I have. The estimation can be found on Chapter 6 of this document. I have taken into account human resource costs, hardware and software amortization, workspace costs, etc. I have also included provisions for contingencies and incidentals.

In my humble opinion, it is a reasonably accurate analysis of the costs of the project, if it were to be carried out as a business endeavour instead of a graduation thesis.

There is little doubt that any Business Administration student would spot mistakes and oversights at first reading. But that is just fine, budgeting is not supposed to be part of my skillset, not at a professional level at least.

PPP: Is the expected cost similar to the final cost? Have you justified any differences (lessons learnt)?

In terms of material resources it is. In terms of human resources, a lot of the tasks that were planned for, with their costs estimated, were not executed.

On the other hand, a lot of time was spent into a task initially estimated as something trivial, and the employment of the different professional roles evens out with the costs we initially estimated for them.

Exploitation: How is the problem that you wish to address currently resolved? In what ways will your solution economically improve existing solutions?

This question doesn't quite apply to our project. That the problem we wish to address is currently unresolved is part of the issue. In fact, that is the core motivation for our project.

If we look in a broader sense, at how the second part of Hilbert's 16th problem is currently addressed (not resolved), then we can say that it's the mathematical community trying to come to a solution of the problem by analytical means only.

This project's mission is to give them more data, more known results, for them to base their analysis on. So the problem we wish to address is more of a complementary, synergistic problem.

Neither do apply questions about the costs during the project's useful life, or about adaptations, updates or repairs. Nor about the risks. This project will not have an exploitation stage.

7.3 Environmental dimension

PPP: Have you estimated the environmental impact of the project?

I have not. I honestly think that I lack the necessary conceptual frameworks to do so, not accurately at least.

PPP: Did you plan to minimize its impact, for example, by reusing resources?

Every non-free, personal resource that I am using in this project is something that I owned before the project, that I use for purposes other than the project, and that I expect to keep using after I finish the project.

This is a statement that stands for my computer and for the different elements of my workspace. It also stands for most of the software, which is all free anyway.

For the software tools that I have adopted for the sake of the project, I think that they are worth learning. Namely, LaTeX is a document preparation system that I am finding very useful and plan to use after the project for most of my "presentable" documents.

PPP: If you carried out the project again, could you use fewer resources?

Nothing in particular comes to mind.

Exploitation: How is the problem that you wish to address currently resolved? In what ways will your solution environmentally improve existing solutions?

As stated in the equivalent question for the economic dimension row of the matrix, there is no existing solution. So there is nothing to improve in this aspect.

Likewise, the questions about the Exploitation and Risks columns do not apply to this project. Ideally, the lessons I learned during the project will be used by someone else to further advance this research subject. But that is not an exploitation phase.

7.4 Social dimension

PPP: Has undertaking this project led to meaningful reflections at the personal, professional or ethical level among the people involved?

It allowed me to delve into the field of scientific programming. I have learned that it is a very challenging line of work. The anxiety of not knowing whether there may be an algorithm to solve the problem has been stressing.

We do study algorithms during the degree, more so in this specialization, but in the end they are usually easy to reason about, at least if you have been keeping up with the course's theory.

But this here was an open problem. And not one of optimization, where you know that at least there is an inefficient solution that you can build from, but a completely open question of not knowing how to approach the problem.

I think that I have also been hampered by my poor understanding of the subject matter. At the beginning of the project I even had to brush up on my knowledge of dynamical systems because I was unable to distinguish partial derivative notation for what it was.

In any case, after working on it for all this time, I do think that this is a very interesting subject, worthy of exploration. I hope someone else with a better mathematical foundation will bring it to fruition.

Last but not least, this project is for me to graduate from university. It is a very important milestone for me.

Exploitation: How is the problem that you wish to address currently resolved? In what ways will your solution improve quality of life with respect to existing solutions?

As previously stated, there is nothing to improve in this aspect because there is no existing solution.

Exploitation: Is there a real need for the project?

We believe there is. Mathematicians working on the second part of Hilbert's 16th problem, and in dynamical systems in general, will benefit from the new data we generate. They will also be able to use our program to calculate even more data, or adapt it to fit the requirements of their particular subjects.

Exploitation: Could any group be adversely affected by the project?

No. It is a very inoffensive endeavor.

For this reason, the questions about Risks do not apply.

Exploitation: To what extent does the project solve the problem that was established initially?

Not very, to be honest. But I have explored the prerequisite problem of finding limit cycles in a dynamical system and have come out of that exploration with some lessons, some pitfalls to avoid and the proposal of a couple of approaches that should work better than the methods I had time to implement during the project.

8 Design and implementation

8.1 Exploration of the problem

8.1.1 Analysis of trajectories

The first approach I tried was integrating the trajectory. For that, I used the function odeint, from the package scipy.integrate.

The code for this approach was shared between `app/deprecated/trajectory_analysis.py` and an earlier version of the function integrate_with_events in `app/functions/integrate_trajectory.py`. The final version of the program provides a controller for running this function in `app/controllers/trajectory_controller.py`.

We knew that there are four kinds of trajectories, or rather, of starting points, in relation to where they converge at infinite time:

- Trajectories that tend to infinite, in any direction of the plane.
- Trajectories that tend to a stationary point, to an attractor.
- Trajectories that start on a limit cycle, and so always remain in the limit cycle.
- Trajectories that converge to a limit cycle, are "pulled" into it.

The work made on this stage allowed us to understand that, with numerical methods, we would not be able to detect that a point lies in a limit cycle just by integrating its trajectory.

Because floating point representation is inexact, and even if we start the trajectory in a limit cycle, the inexactitude of numerical methods will inevitably bring the trajectory out of the limit cycle.

I considered defining an acceptable range of error, but I was uncertain of how to define it, in base to which parameters. Instead, there were other avenues that we could pursue.

In particular, it became apparent that analyzing at the phase space level, taking into account many trajectories and related metrics, was a much better approach.

8.1.2 Scanning incidence, heatmaps

The second approach was based on the concept of heatmaps.

The program scans a region of phase space and computes the trajectory for every point of the scan, up to an arbitrary (but high) time limit. Every point traversed by a trajectory is projected into the heatmap.

The heatmap must obviously be finite, as it is represented by an array of subregions of phase space.

This method is implemented in `app/functions/scan_phase_space/heatmap.py`. Figure 2 shows an example of the results for this approach.

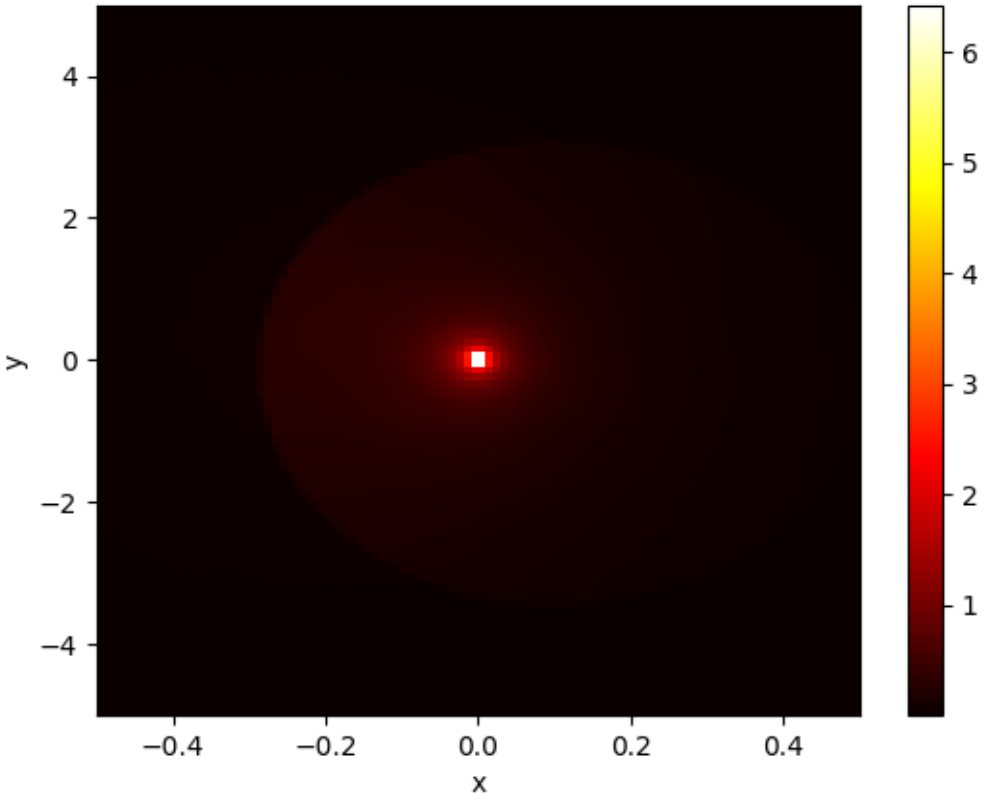

Figure 2: Heatmap scan.
Source: plotted from the computation results stored in `results/heatmap_demo2.json`, which are computed from the job configuration defined in `jobs/heatmap_demo2.yaml`.

Heatmap seemed a very good idea on paper, as it is easy to parallelize and it is a good application for running on GPUs. But trajectories spiral into a limit cycle very slowly. For that reason, when we calculate a heatmap on phase space, the limit cycles don't resurface because their surrounding areas also have a very high incidence.

8.1.3 Scanning sign change

Another approach I tried was scanning the sign change with the rationale that, trajectories that belong to a limit cycle experiment many sign changes of the partial derivatives.

That is, for every "lap" around the cycle, the value of the partial derivative with respect to x equals zero at two points, at the leftmost and rightmost points of the trajectory in the span of that lap.

With numerical methods, we cannot find those two points with certainty nor check whether a given point presents that property (unless we define an acceptable margin of error). But we can check whether the partial derivative evaluates to a positive value at one point of the trajectory and to a negative value at the next step.

As the same holds true for $\partial f \partial y$, a lap around the cycle experiments 4 changes of sign.

The code for this approach is in `app/functions/scan_phase_space/sign_change.py`

Figure 3 presents the results of a sign change scan. It shows how it is hard to extract any relevant information in regards to limit cycles.

The problem with this method is that, again, this principle also holds for trajectories that are spiral, and the regions of phase space next to a limit cycle are composed of spiral trajectories to or away from the limit cycle.

That said, scanning phase space for changes of sign has a relatively low time complexity ($O(n)$ where n is the number of points to be evaluated by the scan) and may perhaps be used to classify systems into those that present spiraling trajectories, and so have a potential for limit cycles, and those that do not.

8.1.4 Analyzing the tendency across loops of spiral trajectories

The last approach I tried, late into the allotted time for the project, was scanning for the tendency between loops.

I went back to the basics of limit cycles, at how one of the properties of limit cycles is that neighbouring trajectories spiral into it as time approaches infinite, or alternatively, as time approaches negative infinite. These are known respectively as stable limit cycles and unstable limit cycles.

There are also other classes of limit cycles that break this assumption. In particular, semi-stable limit cycles spiral into the limit cycle at infinite time on one side, but spiral into into the limit cycle as time approaches negative infinite on the other side.

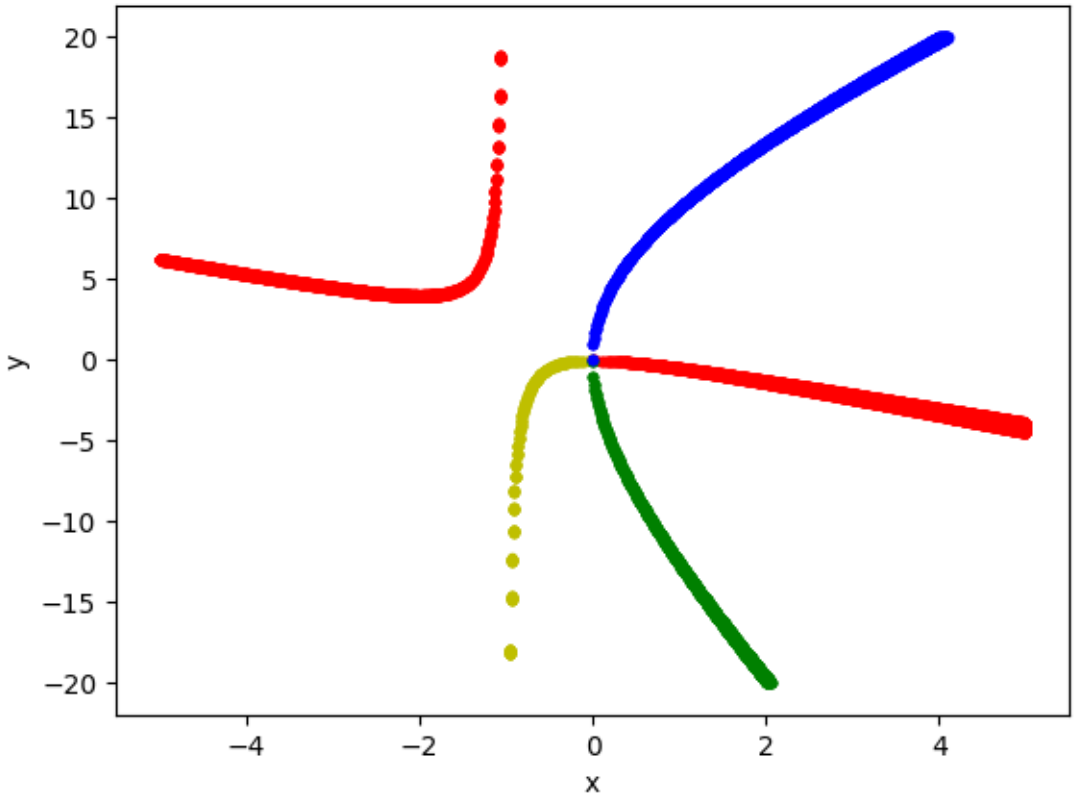

Figure 3: Sign change scan.
Source: plotted from the computation results stored in `results/sign_change_demo.json`, which are computed from the job configuration defined in `jobs/sign_change_demo.yaml`.

I came up with the basis of a method that would find limit cycles of the first two classes, but that would not be able to find semi-stable limit cycles. In any case, exhaustive, perfect solutions were never the goal with a numerical study.

The algorithm assumes that trajectories in a region are either a limit cycle or present a spiral form. It takes a stationary point as reference and evaluates whether the trajectory from a given point grows closer or distant to that point. It applies this evaluation over every point of the region, at a given resolution between points.

In the dynamical system we test our methods against, we know that there is an stationary point at (0, 0) because the coefficient of degree 0 in both partial derivatives is equal to zero.

So this algorithm seeks to "paint" every computed point of phase space in one of two colors, depending on whether the trajectory from the point spirals in or against the direction of the stationary point. The limit cycle would naturally surface in the border between two adjacent regions of different color, provided that one contains the other.

The method is sound, in theory, but in practice I was unable to make a successful implementation out of it. I implemented three different versions.

There is no code for the first version because I lost it when I overwrote the second one on the same file with no backups.

But I can explain that it consisted of integrating the whole trajectory from a point, until the trajectory made one lap around the stationary point, according to the following heuristic for deciding when a complete lap has been traversed:

- the trajectory crosses x_0, with (x_0, y_0) being the initial point
- the trajectory crosses x_0 a second time.

Then the distances to the stationary point from (x_n, y_n) and from (x_0, y_0) are compared. If the end point is closer, then the "interloop" tendency at point (x_0, y_0) is inbound, outbound otherwise.

This program doesn't terminate. The problem lies in how there are inbound trajectories that start at a point where $\partial f \partial x \simeq 0$. Those trajectories never cross x_0 again and so the heuristic for deciding when a lap has been rounded never stops the program.

I considered several solutions to this problem:

- To fall back to using y_0 for the heuristic if $\partial f \partial x(x_0, y_0) \simeq 0$.
- To define a cutoff in terms of time, an integration limit. If the integration function goes past that point, the interloop value for (x_0, y_0) is undefined.
- To consider x_0 and y_0 at the same time. The function stops whenever one of the two axes is crossed for the second time.

The first and third solutions can potentially be another non-terminating program (corner case: starting on an stationary point or its neighborhood). But they can always be complemented with the second solution.

Nevertheless, in the end I abandoned this line of development for a different method that seemed much more promising at that moment.

This next interloop implementation consisted of a "lookahead" of sorts. The code for it is in `app/functions/scan_phase_space/interloop_v2.py`.

Instead of integrating the trajectory until reaching a full lap (approximated by heuristics), we can integrate every trajectory for an arbitrary (and short) amount of time. Then we derive the tendency by computing the distance to the stationary point at the starting and the ending points.

The issue with this that I failed to take into account is that a limit cycle (and the spiraling

trajectories around it) does not form a perfect circle around the stationary point. They can have an elliptic form. In fact, that should be the case for almost every dynamical system presenting limit cycles or spiraling trajectories. Including the known system that I have tested the programs against.

In practice, this means that even if the trajectory is outbound, if its position at the integration limit is in a segment of the ellipse that is significantly closer to the stationary point than the starting point is, then the algorithm will classify the trajectory as inbound.

The third and final try, which seems very promising to this day, is an iteration over this second version. It is implemented in `app/functions/scan_phase_space/interloop_v3.py`.

This method takes yet another point of reference, calculated as a displacement of the starting point in the opposite direction of the stationary point, with arbitrary magnitude. The intuition here is that if there is a spiraling trajectory that connects the two points, then computing the distance between (x_0, y_0) and (x_0^{ref}, y_0^{ref}) and then between (x_n, y_n) and (x_n^{ref}, y_n^{ref}) and comparing them, we can deduce whether the trajectory is inbound or it is outbound.

Again, this implementation did not obtain any satisfactory results. I believe the algorithm is sound, and I suspect there may be a problem with either the precision of the integration method, or the arbitrary integration limit.

Another possible issue that came to mind is that integration precision and limit shouldn't be constant. Trajectories move at more speed the more distant they are from the origin, it is a simple matter of the dynamical system's variables taking larger values.

Another approach I have considered is adding a second reference point, in the direction of the stationary point (so the reference points "sandwitch" the point to be evaluated). Then I would compare the distance from the point of interest to the inner point of reference and the distance from the point of interest to the outer point of reference. Applying this comparison to lookahead points of their trajectories would perhaps be a better evaluation of interloop tendency.

In any case, the idea and implementation of interloop_v2 and v3 was well into the discount time for the project and I did not have the bandwidth to iterate on them any further.

I do believe that both interloop_v1 (more correct) and interloop_v3 (more efficient) are lines of development worth pursuing if anyone resumes this research where I left off.

8.2 Non-functional requirements

This section describes some of the design choices and implementation I made throughout the project in regards to aspects like usability, maintainability, configuration, reproducibility, etc.

Figure 4 shows the files and directories at the root level of the project.

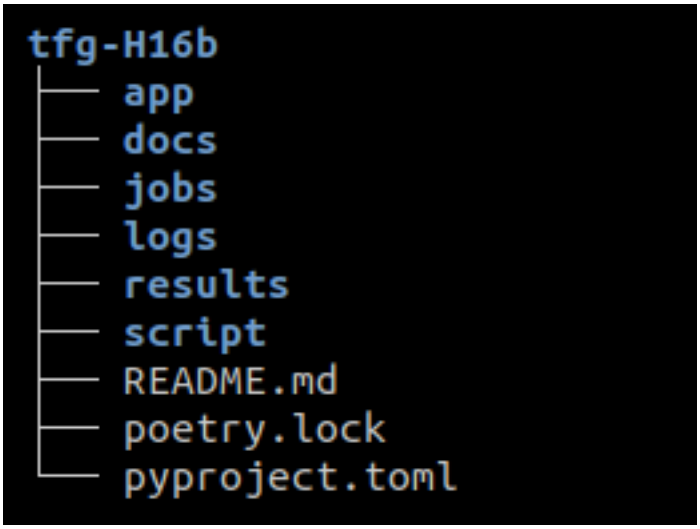

Figure 4: Structure of the project.
Source: compiled by the author.

The Python code lies in `app/`. The parameters for a given computation are given through the job configuration files stored inside `jobs/`.

Programs log progress and debugging events to files in `logs/`. They output their results to `results/`.

`script/` contains a number of shell utilities I developed to streamline development and other tasks such as consumption of results.

The `docs/` directory contains usage documentation for the main program, and also about the configuration options for each kind of job.

The following subsections explore these items in more detail.

8.2.1 Application architecture

As the kinds of scans I was programming grew, common behavior became increasingly obvious because it was repeated in several files. Each of the different methods was a standalone program.

Maintainability was hard because writing a program implied starting from scratch, copying bits and pieces from here and there, but without a collection of components that I could draw from.

Another issue was that these programs were mostly Python scripts with everything happening in a single file, bar some helper classes that performed the analysis of a scan (see `app/deprecated/trajectory_analyzer.py` and `app/deprecated/heatmap_analyzer.py` for examples of this early approach).

This made development hard, because I had to navigate through all the clutter (from parsing the input, to unpacking parameters, or iterating over the scan space) in order to find where new code was supposed to go, or where bugs could possibly be at.

It was also a bit counter-intuitive to have the iteration over scan space, and the trajectory integration, dissociated from the functions that analyzed those points and the trajectories originating from them.

At one point, I decided to refactor all of these programs into a single application, with a single entry point and a number of controllers that deploy one function or another depending on the job configuration file the user provides.

Now the logic for computing a scan, and the one for evaluating (or analyzing) the points and trajectories of that scan, are both in a single self-contained module, and I avoid the need to jump from file to file in order to change the algorithms related to a scan.

The architecture for the final version of the application can be observed in Figure 5.

As explained in the README for the project, the application has a single entry point: `app/cli.py`. It is a command-line interface that exposes two subcommands: compute and view.

The compute command receives a job configuration file (further explained in subsection 8.2.2) and it decides which controller and action will handle the job with regard to the type of job specified in that configuration file.

The controller glues all of the components together (from logging and output configuration to parsing the job configuration file or gathering scan results) and allows me to encapsulate these nitty-gritty interfacing details away from the scan and integration functions, which is where the algorithms lie and where having so much clutter lying around severely hampers comprehension.

Another component I decoupled are the scan maps where a scan function projects its results. I made those into very simple models that only keep track of points and their evaluation and can be easily encoded to and decoded from dictionaries, as an intermediate

step to and from json serialization.

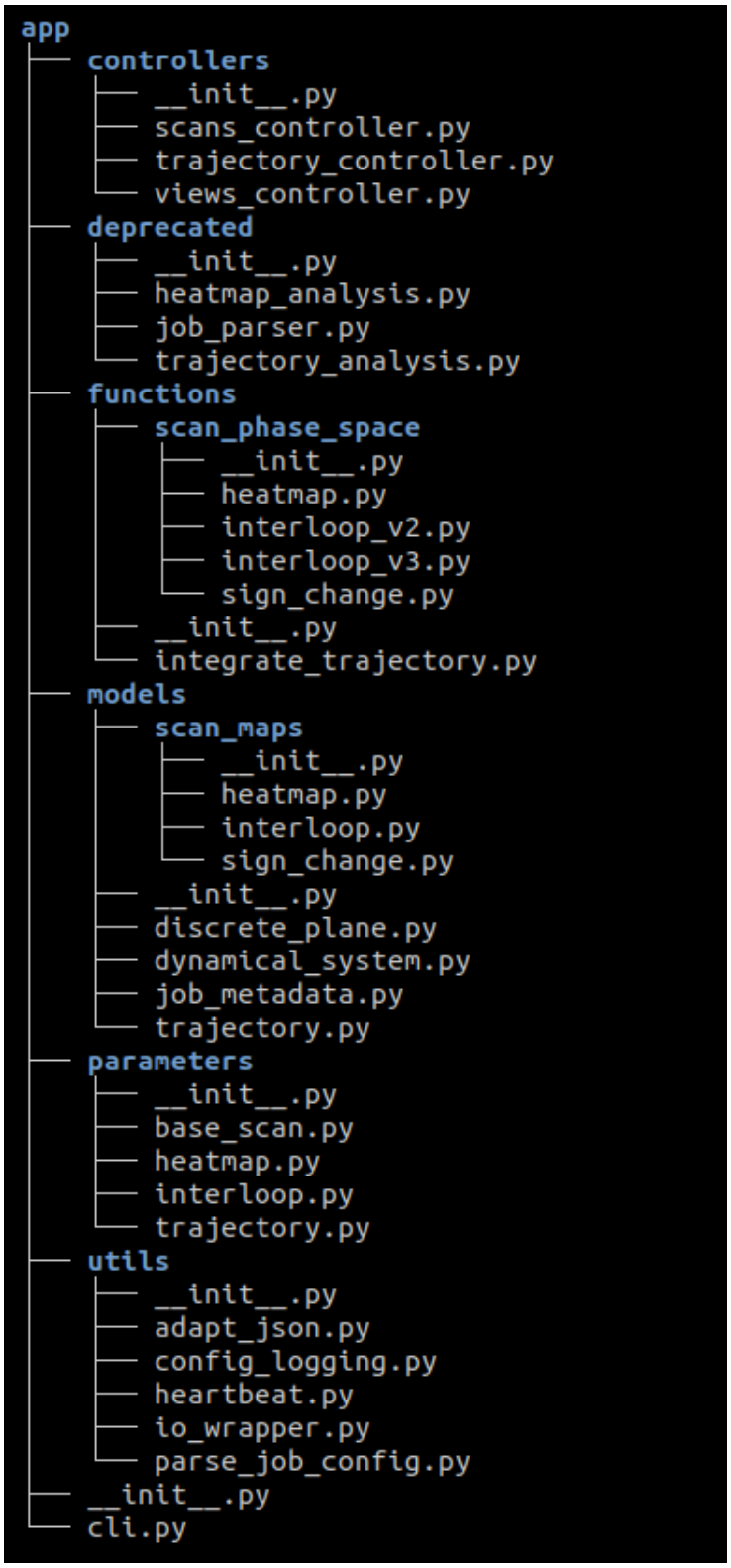

Figure 5: Application architecture.
Source: compiled by the author.

Lastly, I gathered all of the logic for calculating the scan map coordinates of a point to be projected into the model DiscreteMap. I also included here the logic for generating the points to be traversed for a given scan. This proved a bad design decision because this double responsibility and the fact that both these roles depended on a common instance attribute caused a bug in a late stage of the project, when I was making some readability improvements.

Luckily, I test very often so I noticed early, and after one, two hours of pulling at my hair, I noticed the error-prone coupling of those two responsibilities and decoupled them. I decided to leave the model as it was because, again, this was very late into the development stage and I wanted to avoid breaking more things.

It is also worth mentioning that `cli.py` dispatches the job to the required controller and action at run time by leveraging Python's first-class functions (see the compute_routes dictionary). In a similar fashion, `scans_controller.py` funnels the logic for every action into a shared run() method that takes as arguments the scan function to execute and a reference to the class that models the parameters for that scan function.

Using dynamic dispatch like this has allowed me not just to DRY the code, but also to make its maintenance and extension much easier, because I only have to change things in one place and that change is propagated to every scan job type.

Adding a new action, or a new controller, is as simple as importing it into `cli.py` and adding it to the compute_routes dictionary.

As for the view command, it takes a path to the json output by a compute job, unpacks its scan map (" job_results " in the root level of the json hierarchy) and calls its plot function to display it on the screen.

Alternatively, view can also echo the results file into human-readable json (with indentation and newlines), as opposed to the minified json that compute jobs generate.

All in all, this is an application that is very easy to extend and maintain, because things are mostly handled in one way, and programmed in one place.

8.2.2 Configuration of job parameters

I had the intention to handle job configuration with YAML files from the beginning of the project. As you can see in any of the job configs in `jobs/`, there are a lot of parameters to configure for a every job, and YAML provides a very expressive format for doing so.

Passing every parameter by hand, as program arguments and function parameters, would have made the program very hard to use.

Ironically, the first library I used for parsing these configurations was ConfigArgParse, a library that provides a single fron-tend for parsing command-line arguments, configuration files (like YAML or INI) and even environment variables.

I fancied the idea of parsing all the program input with a single dependency, one API. In the end though, it was a bad choice for several reasons. The most important of all, ConfigArgParse only has support for a subset of YAML. It does not allow for nested keys, and that takes away a great deal of the expressiveness of YAML.

Later into the project, I decoupled this into the very limited number of arguments that are accepted by the commandline-interface, parsed in `cli.py` with argparse; and the rather large amount of job options that can be read and updated from the ease of use of a YAML file.

8.2.3 Script utilities

Alongside the application itself, I have programmed a number of small utilities for consuming results, logs, and even for testing.

I haven't written tests per se, but I came up with `script/test.sh` that, depending on the job-type argument that it is passed, runs one of the test jobs defined in `jobs/` or another.

These test jobs take 1-2 seconds to execute and then the script opens their results with view. Every time I made a little change for optimization or readability, I ran these tests.

8.2.4 Presentation of results and reproducibility

I had defined this as an important aspect at the onset of the project, when I was defining objectives and decomposing them in tasks.

I went with JSON files for the results because it is a standard in modern interfaces, because it is easy to code and decode, it can be easily read by humans after a little formatting, etc.

For reproducibility, the output of every job includes a copy of the job config options with which it was called. Along with some useful metadata like date and time of the execution, program duration, etc.

Another aspect where I strived for reproducibility is in the management of dependencies and the development environment.

I had the luck to work for a while in the Ruby and Rails ecosystem, and Ruby has a tool

called Bundler that manages dependencies in a very straightforward manner, with declarative configuration files that "lock" packages to a certain version, or range of versions. These package lists can be committed to source control, and then shared by the entire team in order to keep everyone's local environments in the same page with production and with each other.

From then on, in any language or framework I start a project with, I always look for a Bundler equivalent. I used Pipenv during the first half of this project, but I switched to Poetry when I noticed that Pipenv lacks some very basic features that Poetry has. For instance, and in particular, bumping the version of a package without also updating every other package.

9 Conclusions

In this project, I have explored the second part of Hilbert's 16th problem.

I have studied the fundamentals of dynamical systems in order to get an understanding of the problem. I have also reviewed the state of the art for the problem in search of methods that I could apply to the project.

I have programmed a number of explorations of phase space to try solving the problem of detecting limit cycles on a given dynamical system. I have tested the programs against a well-known configuration of the coefficients which gives a dynamical system with 4 limit cycles.

While I cannot say that the project has been a success, it does accomplish quite a few of the objectives, especially the non-functional requirements. The objectives that are not reached were blocked by the unexpected bottleneck that finding an algorithm for locating limit cycles turned out to be.

Nevertheless, towards the end of the project's allocated time, I came up with an approach that could potentially solve the issue. My implementations of the algorithm failed for diverse reasons, but I am positive that it points in the right direction.

This project also provides a working, modular application that is easy to use and easy to extend. It provides a lot of tooling and conveniences that anybody wishing to pick up from where I left off should be able to leverage. I am confident that it will considerably reduce the "upfront costs" for taking on this research.

In hindsight, I see this project as laying the foundation stone, and a good deal of groundwork, for a numerical study of the second part of Hilbert's 16th problem. I hope that someone will take the baton and bring this study to fruition.

Bibliography

[1] *Hilbert problems - Encyclopedia of Mathematics*. [Online]. Available: https://encyclopediaofmath.org/wiki/Hilbert%7B%5C_%7Dproblems (visited on 10/19/2020).

[2] S. Lynch, "The Second Part of Hilbert's Sixteenth Problem," in *Dynamical Systems with Applications using Mathematica®*, Birkhäuser Boston, Sep. 2007, pp. 217–239, ISBN: 978-0-8176-4586-1. DOI: 10.1007/978-0-8176-4586-1_11. [Online]. Available: https://link.springer.com/chapter/10.1007/978-0-8176-4586-1%7B%5C_%7D11.

[3] G. A. Leonov, "Efficient methods in the search for periodic oscillations in dynamical systems," *Journal of Applied Mathematics and Mechanics*, vol. 74, no. 1, pp. 24–50, 2010, ISSN: 00218928. DOI: 10.1016/j.jappmathmech.2010.03.004.

[4] S. Shi, "A concrete example of the existence of four limit cycles for plane quadratic systems," *Scientia Sinica*, vol. 23, no. 2, p. 153, 1980. DOI: 10.1360/ya1980-23-2-153. [Online]. Available: http://engine.scichina.http//engine.scichina.com/publisher/scp/journal/MathA0/23/2%20https://www.semanticscholar.org/paper/A-CONCRETE-EXAMPLE-OF-THE-EXISTENCE-OF-FOUR-LIMIT-Graduate/76ef93012c1057008df403c5bd3d6da66a9f6242.

[5] N. V. Kuznetsov, O. A. Kuznetsova, and G. A. Leonov, "Visualization of Four Normal Size Limit Cycles in Two-Dimensional Polynomial Quadratic System," *Differential Equations and Dynamical Systems*, vol. 21, no. 1-2, pp. 29–34, 2013, ISSN: 09713514. DOI: 10.1007/s12591-012-0118-6.

[6] G. A. Leonov, I. G. Burova, and K. D. Aleksandrov, "Visualization of four limit cycles of two-dimensional quadratic systems in the parameter space," *Differential Equations*, vol. 49, no. 13, pp. 1675–1703, Dec. 2013, ISSN: 00122661. DOI: 10.1134/S0012266113130028.

[7] G. A. Leonov, N. V. Kuznetsov, and E. V. Kudryashova, "Cycles of two-dimensional systems: Computer calculations, proofs, and experiments," *Vestnik St. Petersburg University: Mathematics*, vol. 41, no. 3, pp. 216–250, Sep. 2008, ISSN: 10634541. DOI: 10.3103/S1063454108030047. [Online]. Available: https://link-springer-com.recursos.biblioteca.upc.edu/article/10.3103/S1063454108030047.

[8] S. Lynch, "Symbolic Computation of Lyapunov Quantities and the Second Part of Hilbert's Sixteenth Problem," in *Differential Equations with Symbolic Computation*, Birkhäuser-Verlag, Mar. 2006, pp. 1–22. DOI: 2. [Online]. Available: https://link-springer-com.recursos.biblioteca.upc.edu/chapter/10.1007/3-7643-7429-2%7B%5C_%7D1.

[9] *Principles behind the Agile Manifesto*. [Online]. Available: https://agilemanifesto.org/principles.html (visited on 10/19/2020).

[10] *What is Scrum?* [Online]. Available: https://www.scrum.org/resources/what-is-scrum (visited on 10/19/2020).

[11] *Kanban Explained in 10 Minutes — Kanbanize*. [Online]. Available: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban (visited on 10/19/2020).

[12] *Manage your team's work, projects, & tasks online · Asana*. [Online]. Available: https://asana.com/ (visited on 10/19/2020).

[13] *Barcelona Supercomputing Center Junior Research Engineer Salaries in Barcelona — Glassdoor*. [Online]. Available: https://www.glassdoor.com/Salary/Barcelona-Supercomputing-Center-Junior-Research-Engineer-Barcelona-Salaries-EJI%7B%5C_%7DIE382342.0,31%7B%5C_%7DKO32,56%7B%5C_%7DIL.57,66%7B%5C_%7DIM1015.htm (visited on 10/14/2020).

[14] *Barcelona Supercomputing Center Research Engineer Salaries in Barcelona — Glassdoor*. [Online]. Available: https://www.glassdoor.com/Salary/Barcelona-Supercomputing-Center-Research-Engineer-Barcelona-Salaries-EJI%7B%5C_%7DIE382342.0,31%7B%5C_%7DKO32,49%7B%5C_%7DIL.50,59%7B%5C_%7DIM1015.htm (visited on 10/14/2020).

[15] *Salary: IT Project Manager in Barcelona, Spain — Glassdoor*. [Online]. Available: https://www.glassdoor.com/Salaries/barcelona-it-project-manager-salary-SRCH%7B%5C_%7DIL.0,9%7B%5C_%7DIM1015%7B%5C_%7DKO10,28.htm (visited on 10/14/2020).

[16] "Sustainability Analysis for the Bachelor's Thesis," Facultat d'Informàtica de Barcelona, Tech. Rep., 2018. [Online]. Available: https://www.fib.upc.edu/sites/fib/files/documents/estudis/sustainability-report-english-2018.pdf.