PHY 415 Fall 2023

Danny Caballero, Alia Valentine

Aug 21, 2023

CONTENTS

I 0 - Intro and Syllabus 3

1 Syllabus and Overview of PHY 415 5
- 1.1 Contact Information 5
- 1.2 Grading 6
- 1.3 Course Objectives 6
- 1.4 Course Design 8
- 1.5 Course Activities 10
- 1.6 Assessments 12
- 1.7 Project Rubrics 13
- 1.8 Calendar 14
- 1.9 Classroom Environment 14
- 1.10 Resources 15

II 1 - Mechanics and ODEs 17

2 Intro to Classical Mechanics 19
- 2.1 Interpreting the statement of ODEs 19
- 2.2 Initial Conditions 20
- 2.3 A note on differentiability 20

3 31 Aug 23 - Frames and Coordinates 21
- 3.1 What is a Frame? 21
- 3.2 Forces in polar coordinates 22

4 5 Sep 23 - Calculus of Varations and Lagrangian Dynamics 25
- 4.1 Activity 26
- 4.2 Adding Constraint Forces 26

5 7 Sep 23 - Numerical Integration and More Lagrangians 31
- 5.1 Harmonic Oscillator 31
- 5.2 Setting up to numerically integrate 32
- 5.3 Back to Paraboloid Paradise 34
- 5.4 How Does Numerical Integration actually work? 37
- 5.5 Numerically analyzing this system 38
- 5.6 Angular Momentum Analysis 40
- 5.7 Constraints Revisited - Lagrange Multipliers 41

6 12 Sep 23 - The Dynamical Systems Approach and Phase Portraits 43
- 6.1 1D System 43
- 6.2 Fixed Points 44

6.3 The Phase Portrait of the SHO . . . 46
6.4 Phase Portrait of the Simple Harmonic Oscillator . . . 50
6.5 The Large Angle Pendulum . . . 53
6.6 (Time Permitting) Fixed Points in 2 Dimensions . . . 55

7 14 Sep 23 - Dynamical Systems Continued **57**
7.1 Eigenvalues . . . 57
7.2 Activity: Lotka - Volterra Equations . . . 59
7.3 Investigating the Van der Pol Oscillator . . . 60

8 19 Sep 23 - CHAOS **65**
8.1 Definition of Chaos . . . 65
8.2 Halvorsen Attractor . . . 65
8.3 Lyapunov Exponents . . . 67
8.4 Driven Damped Pendulum . . . 68
8.5 Poincaré Section . . . 70

9 21 Sept 23 - Activity: ODE Games **73**
9.1 Activity . . . 74

III 2 - E&M and PDEs **79**

10 Intro to Electricity and Magnetism **81**

11 26 Sep 23 - Graphing Electric Fields **83**

12 28 Sep 23 - Laplace's Equation **87**

13 3 Oct 23 - More PDEs **89**

14 5 sep 23 - Method of Relaxation **91**

15 10 Oct 23 - Magnetic Fields **93**

16 Electromagnetic Waves & the Wave Equation **95**

IV 3 - Waves and Complex Analysis **97**

17 Intro to Waves **99**

18 17 Oct 23 - Normal Modes **101**
18.1 Three Coupled Oscillators . . . 101
18.2 Extending your work . . . 102
18.3 Even further . . . 103

19 19 Oct 23 - Mechanical Waves **105**

20 26 Oct 23 - Examples of Waves **107**

21 31 Oct 23 - Complex Analysis & the Fourier Transform **109**

22 2 Nov 23 - Discrete and Fast Fourier Transforms **111**

V 4 - Random Processes and Distributions 113

23 Intro to Distributions 115

PHY 415, called, "Mathematical Methods for Physicists" is a course the brings together many of the mathematical approaches that we commonly use in physics and apply them to variety of problems. In this course, we will take a modeling-based approach where we focus on the mathematical descriptions of physical phenomenon and determine what mathematical and analytical approaches are useful in exploring those models.

To get a sense of the course, please read all the pages associated with our syllabus.

Learning Objectives

In this course, you will learn to:

- investigate physical systems using a variety of tools and approaches,
- construct and document a reproducible process for those investigations,
- use analytical, computational, and graphical approaches to answer specific questions in those investigations,
- provide evidence of the quality of work using a variety of sources, and
- collaborate effectively and contribute to a inclusive learning environment

Table of contents:

- 0 - Intro and Syllabus
  - *Syllabus and Overview of PHY 415*
- 1 - Mechanics and ODEs
  - *Intro to Classical Mechanics*
  - *31 Aug 23 - Frames and Coordinates*
  - *5 Sep 23 - Calculus of Varations and Lagrangian Dynamics*
  - *7 Sep 23 - Numerical Integration and More Lagrangians*
  - *12 Sep 23 - The Dynamical Systems Approach and Phase Portraits*
  - *14 Sep 23 - Dynamical Systems Continued*
  - *19 Sep 23 - CHAOS*
  - *21 Sept 23 - Activity: ODE Games*
- 2 - E&M and PDEs
  - *Intro to Electricity and Magnetism*
  - *26 Sep 23 - Graphing Electric Fields*
  - *28 Sep 23 - Laplace's Equation*
  - *3 Oct 23 - More PDEs*
  - *5 sep 23 - Method of Relaxation*
  - *10 Oct 23 - Magnetic Fields*
  - *Electromagnetic Waves & the Wave Equation*
- 3 - Waves and Complex Analysis
  - *Intro to Waves*
  - *17 Oct 23 - Normal Modes*

  - *19 Oct 23 - Mechanical Waves*
  - *26 Oct 23 - Examples of Waves*
  - *31 Oct 23 - Complex Analysis & the Fourier Transform*
  - *2 Nov 23 - Discrete and Fast Fourier Transforms*
- 4 - Random Processes and Distributions
  - *Intro to Distributions*

Part I

0 - Intro and Syllabus

CHAPTER

ONE

SYLLABUS AND OVERVIEW OF PHY 415

In designing this course, I plan to emphasize more independent learning on your part and greater agency for you in determining what you learn and how you demonstrate you have learned. So you should expect:

- to read a variety of pieces of information to coordinate information
- to present your ideas publicly and to discuss them
- to learn new approaches and novel techniques on your own
- to become more expert than me in the areas of your interest
- to learn more about scientists that you have not learned about

This is not to say that you are on your own. Here's what you can expect from me:

- resources, information, and tools to help you learn
- support and scaffolding to move you towards more independence in your learning
- timely and detailed feedback to help you along
- a commitment to an inclusive classroom

1.1 Contact Information

1.1.1 Web page

- Web page for this class: https://valentine-alia.github.io/phy415fall23/content/intro

1.1.2 Instructor

- Prof. Danny Caballero (he/him/his)
- Class Meetings: Tuesdays and Thursdays 10:20am-12:10pm (Location: 1300 BPS)
- Email: caball14@msu.edu, office: 1310-A BPS
- Office hrs: To be scheduled, but I also have an open door policy. I enjoy visiting and talking with you about physics.

1.1.3 Learning Assistant

- Alia Valentine (she/her/hers)
- Email: valen176@msu.edu
- Office hrs: TBD, feel free to email me if you want to set up a time. I too enjoy visiting and talking with you about math and physics.

1.2 Grading

Details about *course activities are here* and *information regarding assessment is here*. Your grade will be comprised of weekly discussion questions and several projects that you will complete in the form of a Jupyter notebook (a "computational essay", which we will discuss later). Your grade on each project is split between completion (50%) and quality (50%). We will collectively define "quality" in class, but we have provided *a preliminary rubric* for us to work from for the first project. Your final grade will be scaled based on your best performances; there will be slightly more projects than what comprises your grade. *The intent here is to to allow you space to explore a model or project that you really enjoy, and to reward you for doing that.* How your grade is calculated appears below.

Activity	Percent of Grade
Best Project Grade	30%
2nd Best Project Grade	25%
3rd Best Project Grade	20%
4th Best Project Grade	10%
Weekly Discussion Questions (completion)	15%

While attendance is not required, you are unlikely to succeed with your projects without regular attendance and engagement.

1.3 Course Objectives

This course emphasizes making models of physical phenomenon and how we use various tools at our disposal to investigate those models. Hence, we have learning objectives for making models of these systems and for learning specific tools.

1.3.1 Investigate physical systems

Clearly, one of our central goals is learning how to make models of physical systems. This means learning about and developing fluency with a wide variety of mathematical and computational tools. In this courses, we will make extensive use of Jupyter notebooks for homework and projects. In fact, what you are reading is a set of Jupyter notebooks! Below, you will see the list of objectives for this principal objective.

Investigating Physical Systems Learning Objectives

Students will demonstrate they can:

- use mathematical techniques to predict or explain some physical phenomenon
- employ computational models and algorithms to investigate physical systems
- compare analytical and computational approaches to these investigations

- provide coherent explanations for their investigations buttressed by physical, mathematical, and/or computational knowledge and principles

Principle Learning Objectives

Students will demonstrate they can:

- investigate physical systems of their choosing using a variety of tools and approaches
- construct and document a reproducible process for those investigations
- use analytical, computational, and graphical approaches to answer specific questions in those investigations
- provide evidence of the quality of their work using a variety of sources
- collaborate effectively and contribute to a inclusive learning environment

Each of these learning objectives contributes to your development as a physicist. I recognize that these are **big** ideas to think about. What I mean is that the objectives above are quite broad and you might be able to see a little about what or why they are included. But, below, I added more detail about each one along with a smaller scale list of objectives that you will engage with. Throughout our course, you will have opportunities to demonstrate these objectives in your work. *My aim is to make what you are assessed on in this course something you are interested in, so these objectives reflect that.*

1.3.2 Construct and document a reproducible process

A critical element of physics work is making sure that with the same setup and approach, others can reproduce the work you have done. This provides validity to your work and evidences how we develop collective understanding of physics. Physics is a social enterprise and the ensuring the reproducibility of work supports that enterprise. Below are the learning objectives for this principal objective.

Reproducibility Learning Objectives

Students will demonstrate they can:

- document their work and analysis such that others can reproduce their work
- consistently reproduce their work and results in a variety of contexts
- provide an explanation for why certain work or results are not (or should not be) reproducible

1.3.3 Use analytical, computational, and graphical approaches

The main approaches that we use to make models are mathematical, computational, and graphical. In this class, we will aim to leverage the benefits of each to learn more about the physical systems that we are investigating. Indeed, much of the "knowledge" that you are going to develop will be about specific analytical, computational, or graphical approaches to investigate physical systems. Below are the learning objectives for this principal objective.

Modeling Approaches Learning Objectives

Students will demonstrate they can:

- Use a wide variety of modeling techniques to investigate different physical systems
- Choose and employ appropriate approaches to modeling physical systems of their choosing

- Explain how those approaches lead to different results or conclusions

1.3.4 Provide evidence of the quality of their work

The definition of the quality of a piece of science is a collective decision by the scientific community. In established communities, like physics, there are commonly-accepted ways of defining the quality of work (norms, customs, and rules all play a role). But that is not to mean those ways can't change; papers describing quantum physics and relativity brushed up hard against this issue of quality and were both dismissed and celebrated. Newer disciplines are still establishing those norms and rules. And in some cases, disciplines are pushing back against Western norms of quality. In our class, we will collectively decide what we mean by ”high quality” work. Below are the learning objectives for this principal objective.

Quality Control Learning Objectives

Students will demonstrate they can:

- describe what it means to have high quality work in our class
- look for and evaluate when work meets those standards
- provide suggestions (or act on suggestions) to improve the quality of their work

1.3.5 Collaborate effectively

Physics is a social enterprise that relies on effective and productive collaborations. Very little (if any) science is done alone; the scale of science is too grand for individuals to effectively work – everyone needs a team. In this spirit, in this classroom, we deeply encourage collaboration. We will try to develop effective collaboration through your work on projects and our in-class activities. Below are the learning objectives for this principal objective.

Collaboration Learning Objectives

Students will demonstrate they can:

- Collaborate on a variety of activities in and out of class
- Document the contributions in these collaborations and make changes if contributions are unbalanced
- Develop personally effective strategies for collaboration

1.4 Course Design

For most of you, 4415 is an elective course that you are taking to learn more about how we use mathematical techniques in physics. As such, this course is designed under several different principles than a standard course. Below, I provide those principles and their rationale.

- 415 should help you learn the central tenets of modeling physical systems
  - The sheer volume of mathematical and computational physics knowledge out there is immense and impossible for any one person to learn. However, the central elements of making models, how to learn about specific techniques, and how to debug your approaches are things we can learn and employ broadly as well as to specific problems.
- 415 should be a celebration of your knowledge

  - For most of you, this course is part of your senior level coursework. What you have achieved in the last three to four years should be celebrated and enjoyed. This course will provide ample opportunities for you to share what things you know and what things you are learning with me and with each other.
- 415 should give you opportunities to engage in professional practice
  - As you start towards your professional career, it's important to learn what professional scientists do. You have probably already begun this work in advanced lab and research projects that you have worked on. We will continue developing your professional skills in this course through the use of course projects.
- 415 will illustrate that we can learn from each other
  - Even though I've been learning physics for almost 20 years, I don't know everything. I am excited to learn from you and I hope that you are excited to learn from me and each other.

1.4.1 Optional purchases:

The core readings and work for this course will be this jupyterbook. I will find resources online, make my own, and provide as much organized free material as possible. If you want to have a textbook that helps you organize your readings, please obtain copies of:

1. Mary Boas, *Mathematical Methods in the Physical Sciences* (Wiley; 2005). This book is the definitive text on mathematical approaches, written by Dr. Boas originally in 1966! Any 3rd edition will be useful and I will put the section numbers from Boas in the online readings.
2. Mark Newman, *Computational Physics* (CreateSpace Independent Publishing Platform; 2012). This book is a great introduction to a variety of computational physics techniques, written by UMich professor Mark Newman for a computational physics course. I will put section numbers from Newman in the online readings.

Additional sources

In addition, I will draw from the following books. I have copies of them if you want or need scans of sections. But they can found online in Google Books and other places as well. no need to purchase unless you want a copy for your personal library.

Mechanics

- Edwin Taylor, Mechanics
- Jerry Marion and Stephen Thornton, Classical Dynamics of Particles and Systems
- Charles Kittel, Walter D. Knight, Malvin A. Ruderman, A. Carl Helholtz, and Burton J. Moyer, Mechanics

Electromagnetism

- Edward Purcell, Electricity and Magnetism
- David J. Grriffths, Introduction to Electromagnetism

Quantum Mechanics

- David McIntyre, Quantum Mechanics
- David J. Griffiths, Introduction to Quantum Mechanics

Waves and Thermal Physics

- Frank S. Crawford, Waves
- Charles Kittel, Thermal Physics
- Ashley Carter, Classical and Statistical Thermodynamics
- Daniel Schroeder, Thermal Physics

Additional Physics Topics

- Steven H. Strogatz, Nonlinear Dynamics and Chaos
- B Lautrup, Physics of Continuous Matter
- Frank L. Pedrotti and Leno S. Pedrotti, Introduction to Optics

Mathematics

- Susan M. Lea, Mathematics for Physicists
- William E. Boyce and Richard C. DiPrima, Elementary Differential Equations
- James Brown and Ruel Churchill, Complex Variables and Applications
- Jerrold Marsden and Anthony Tromba, Vector Calculus
- Sheldon Ross, A First Course in Probability

Presenting (Visual) Information

- Edward Tufte, The Visual Display of Quantitative information
- Albert Cairo, The Truthful Art
- Stephen E. Toulmin, The Uses of Argument

1.5 Course Activities

1.5.1 “Readings”

“Reading” is an essential part of 415! Reading the notes before class is very important. I use “reading” in quotes, because in our class this idea goes beyond just reading text and includes understanding figures and watching videos. These should help inform the basis of your understating that we will draw on in class to clarify your understanding and to help you make sense of the material. I will assume you have done the required readings in advance! It will make a huge difference if you spend the time and effort to carefully read and follow the resources posted. The calendar has the details on videos and readings that will be updated.

Weekly Questions: To encourage and reward you for keeping up with the "readings", there will be weekly questions about the readings posted for you to respond to. These are not meant to test your knowledge, but rather to focus your "reading" towards what you understand, and what you don't yet understand. I will ask you about those things weekly and use that information to tailor in-class activities based on what I am hearing is confusing, unclear, or challenging. These questions are only graded for completion, but I do want your honest attempt.

1.5.2 Class Meetings

Classroom Etiquette: Please silence your electronic devices when entering the classroom. I don't mind you using them (in fact, see below, we will use them). But, sometimes, they can very distracting to your neighbors, so use your judgement. I appreciate that you might have questions or comments about things in class. We are going to be having short lectures combined with longer project work in class. So you will have plenty of time to catch up with social media and the news.

If you and/or your group mates are confused, just raise your hand and ask questions. If you are confused, you are likely not the only one and it's better to chat about it, then move on. Questions are always good, and are strongly encouraged! *The only way we learn is to question what we know and how we know it.*

Computing Devices: Please bring some sort of computing device to class everyday. You might be researching information online, reviewing work you have done, or actively building models of systems together. This device can be a computer, a tablet, or a phone. You can also partner up with folks because we will use them in groups. *If you need a computing device brought to class for you or your group mates to use, let me know. I will organize for some small collection of laptops if we need it.*

In-Class: We will have some short lectures about topics or concepts; some of those will be in-the-moment as needed. The idea is that you are developing a basic understanding through readings and videos, practicing using those new ideas with me and with your classmates in class, and then applying what you are learning to new ideas. So, we will also use a variety of in-class activities that help you construct an understanding of a particular topic or concept. These will not be collected or graded, but we will discuss the solutions in class. *I will not post solutions for these activties as we have no exams or quizzes.*

1.5.3 Projects

In-class Projects: The class is designed to support your independent research into ideas that you are excited about. So in-class projects are meant to equip you with the knowledge and practice to learn new things for your projects. These in-class projects will be short demonstrations of models that you complete in groups. We will circulate around the room and check on you and your group's progress and understanding. At the end of the class period, we will share the results of the in-class project and discuss any sticking points. These in-class activites will not be graded, but they will be essential for your out-of-class projects.

Out-of-class Projects: For this class, we anticipate 6 projects to be turned in roughly every 2-3 weeks, with a weeklong turn-in window (see calendar). Except for the first project, up to 3 of these projects can be completed as partner projects. Partner projects are subject to a different grading rubric that evaluates collaborative efforts and increases the expectation for other areas compared to an individual project. A preliminary rubric appears here, but we will define these collectively after the first project.

These projects will take the form a computational essay, which provides documentation and rationale for the exploration that you are completing. We will model a computational essay project in our first project and we will reflect on the rubric after it, and make changes collectively as a class to it.

I strongly encourage collaboration, an essential skill in science and engineering (and highly valued by employers!) Social interactions are critical to scientists' success – most good ideas grow out of discussions with colleagues, and essentially all physicists work as part of a group. Find partners and work together. However, it is also important that you OWN the material. I strongly suggest you start working by yourself (and that means really making an extended effort on every activity). Then work with a group, and finally, finish up on your own – write up your own work, in your own way. There will also be time for peer discussion during classes – as you work together, try to help your partners get over confusions, listen to them, ask each other questions, critique, teach each other. You will learn a lot this way! For all assignments, the work you turn in must in the end be your own: in your own words, reflecting your own understanding. (If, at any time, for any reason, you feel disadvantaged or isolated, contact me and I can discretely try to help arrange study groups.)

Help Session

Help sessions/office hours are to facilitate your learning. We encourage attendance - plan on working in small groups, our role will be as learning coaches. The sessions are concept and project-centric, but we will not be explicitly telling anyone how to do your project (how would that help you learn?) I strongly encourage you to start all projects on your own. If you come to help sessions "cold", the value of the project to you will be greatly reduced.

1.6 Assessments

1.6.1 Formative Assessment

Formative assessment is often ungraded and reflective assessment. It is meant to help you make changes to your thinking, approaches, or practice. It is not evaluative, it's corrective; to help you make changes. We will make heavy use of ungraded formative feedback throughout the course.

1.6.2 Summative Assessment

Summative assessment is typically evaluative and will take the form of course projects completed out of class. These projects will take the form of a computational essay in which you write mathematics and code to investigate and explain a given phenomenon of interest. We will explore those essays in class and talk about what makes a useful one as we define a rubric for evaluation.

Preliminary Rubric

A preliminary rubric has been posted. We will use this rubric for the first out-of-class project evaluation. We will then reflect on it and make changes to collectively as a class.

Resources for Computational Essays

If you want to read more about computational essays, here's a few links in the order utility/readability:

- Steven Wolfram - What is a Computational Essay?
- University of Oslo Physics - Examples of Computational Essays
- Odden and Burk, The Physics Teacher - Computational Essays in the Physics Classroom
- Odden, Lockwood, and Caballero, Physical Review PER - Physics computational literacy: An exploratory case study using computational essays

1.7 Project Rubrics

1.7.1 Preliminary (For first out of class project)

We have worked together to define elements of a rubric that matter for making physical models. These elements appear as part of major learning goals below.

Goal	Fractional Importance
Investigate physical systems	0.30
Construct and document a reproducible process	0.10
Use analytical, computational, and graphical approaches	0.30
Provide evidence of the quality of their work	0.10
Collaborate effectively	0.20

Goal: Investigate physical systems (0.30)

- How well does your computational essay predict or explain the system of interest?
- How well does your computational essay allow the user to explore and investigate the system?

Goal: Construct and document a reproducible process (0.10)

- How well does your computational essay reproduce your results and claims?
- How well documented is your computational essay?

Goal: Use analytical, computational, and graphical approaches (0.30)

- How well does your computational essay document your assumptions?
- How well does your computational essay produce an understandable and parsimonious model?
- How well does your computational essay explain the limitations of your analysis?

Goal: Provide evidence of the quality of their work

- How well does your computational essay present the case for its claims?
- How well validated is your model?

Goal: Collaborate effectively

- How well did you share in the class's knowledge?
  - How well is that documented in your computational essay?
- How well did you work with your partner ? *For those choosing to do so*

1.8 Calendar

In this course, we will cover four principal topics in physics (in this order):

We will spend roughly 1/3 of the course on each, and the expectaitn is that each topic has 2 completed projects.

A Google Calendar appears below for class. If you review the notes in a given event, you will find the details for a class or the readings to do.

1.9 Classroom Environment

1.9.1 Commitment to an Inclusive Classroom

I am deeply committed to creating an inclusive classroom - one where you and your classmates feel comfortable, intellectually challenged, and able to speak up about your ideas and experiences. This means that our classroom, our virtual environments, and our interactions need to be as inclusive as possible. Mutual respect, civility, and the ability to listen and observe others are central to creating a classroom that is inclusive. I will strive to do this and I ask that you do the same. If I can do anything to make the classroom a better learning environment for you, please let me know.

If you observe or experience behaviors that violate our commitment to inclusivity, please let me know as soon as possible.

If I violate this principle, please let me know or please tell the undergraduate department chair, Stuart Tessmer (tessmer@pa.msu.edu), who I have informed to tell me about any such incidents without conveying student information to me.

1.9.2 Comments on preparation:

Physics 415 covers material you might have seen before. Many of the topics stem from a wide variety of physics courses you might have already taken. But, we might be applying them at a higher level of conceptual and mathematical sophistication.

Therefore you should expect:

- a large amount of material to review and digest.
- no recitations, and few examples covered in lecture. Most of the learning will be done through projects and questions you and your group mates raise.
- long, hard problems that usually cannot be completed by one individual alone.
- challenging projects.
- to learn more about being a physicist that you have in another class (I hope!).

Physics 415 is a challenging, upper-division physics course. Unlike more introductory courses, you are fully responsible for your own learning. In particular, you control the pace of the course by asking questions in class. I tend to speak quickly, and questions are important to slow down. This means that if you don't understand something, it is your responsibility to ask questions. Attending class and the help sessions gives you an opportunity to ask questions. I am here to help you as much as possible, but I need your questions to know what you don't understand.

Physics 415 covers some of the most important physics and mathematical methods in the field. Your reward for the hard work and effort will be learning important and elegant material that you will use over and over as a physics major. Here is what I have experienced, and heard from other faculty teaching upper division physics in the past:

- most students reported spending a minimum of 10 hours per week on the homework (!!)

- students who didn't attend the help sessions often did poorly in the class.
- students reported learning a tremendous amount in this class.

The course topics that we will cover in Physics 415 are among the greatest intellectual achievements of humans. Don't be surprised if you have to think hard and work hard to master the material.

1.10 Resources

1.10.1 Confidentiality and Mandatory Reporting

College students often experience issues that may interfere with academic success such as academic stress, sleep problems, juggling responsibiities, life events, relationship concerns, or feelings of anxiety, hopelessness, or depression. As your instructor, one of my responsibilities is to help create a safe learning environment and to support you through these situations and experiences. I also have a mandatory reporting responsibility related to my role as a University employee. It is my goal that you feel able to share information related to your life experiences in classroom discussions, in written work, and in one-on-one meetings. I will seek to keep information you share private to the greatest extent possible. However, under Title IX, I am required to share information regarding sexual misconduct, relationship violence, or information about criminal activity on MSU's campus with the University including the Office of Institutional Equity (OIE).

Students may speak to someone confidentially by contacting MSU Counseling and Psychiatric Service (CAPS) (caps.msu.edu, 517-355-8270), MSU's 24-hour Sexual Assault Crisis Line (endrape.msu.edu, 517-372-6666), or Olin Health Center (olin.msu.edu, 517-884-6546).

1.10.2 Spartan Code of Honor Academic Pledge

As a Spartan, I will strive to uphold values of the highest ethical standard. I will practice honesty in my work, foster honesty in my peers, and take pride in knowing that honor is worth more than grades. I will carry these values beyond my time as a student at Michigan State University, continuing the endeavor to build personal integrity in all that I do.

1.10.3 Handling Emergency Situations

In the event of an emergency arising within the classroom, Prof. Caballero will notify you of what actions that may be required to ensure your safety. It is the responsibility of each student to understand the evacuation, "shelter-in-place," and "secure-in-place" guidelines posted in each facility and to act in a safe manner. You are allowed to maintain cellular devices in a silent mode during this course, in order to receive emergency SMS text, phone or email messages distributed by the university. When anyone receives such a notification or observes an emergency situation, they should immediately bring it to the attention of Prof. Caballero in a way that causes the least disruption. If an evacuation is ordered, please ensure that you do it in a safe manner and facilitate those around you that may not otherwise be able to safely leave. When these orders are given, you do have the right as a member of this community to follow that order. Also, if a shelter-in-place or secure-in-place is ordered, please seek areas of refuge that are safe depending on the emergency encountered and provide assistance if it is advisable to do so.

Part II

1 - Mechanics and ODEs

CHAPTER TWO

INTRO TO CLASSICAL MECHANICS

Welcome to Mathematical Modeling in physics! Our first unit will focus on Classical Mechanics and Ordinary Differential Equations (ODEs). Classical Mechanics is all about the motion of macrosopic objects, typically ones that are moving slow enough so that we can ignore special relativity. In particular, we are interested in systems that obey Newton's second law:

$$\mathbf{F} = \frac{d\mathbf{p}}{dt} = \dot{\mathbf{p}}$$

or equivalently (if m is constant):

$$\mathbf{F} = m\frac{d\mathbf{r}}{dt^2} = m\ddot{\mathbf{r}}$$

Here we're using dot notation to mean derivatives with repsect to time. We'll continue to see this notation through this course.

The key thing to note here is that both of these equations are statements of **ordinary differential equations**. A huge portions of problems in physics boil down to differential equations, its probably more difficult to think of physics problems that aren't differential equations than ones that are. Broadly speaking, differential equations are equations of some **unkown** function and its derivatives. Often we are concerned with the form and/or behavior of this unknown function. In terms of newtons laws, if we are able to solve the differential equation $\mathbf{F} = m\ddot{\mathbf{r}}$, then we get some function $\mathbf{r}(t)$ that tells us exactly how our system evolves in time.

2.1 Interpreting the statement of ODEs

A nice way to think of differential equations is as a set of instructions for how a function should change in time. For example, consider the first order ODE:

$$\dot{x} = x$$

This equation is saying that the function $x(t)$ should change according to what its current value is. We can also think of $\dot{x}$ as the velocity of function $x(t)$, so this equation is also saying that the velocity of $x(t)$ needs to be equal to its current position at all times, or that when we take a derivative of this function we get itself back. The function e^t has this property, so it might be our go-to guess for the solution of this system. Since this equation is seperable, we can solve it exactly by integrating:

$$\int \frac{dx}{x} = \int dt \implies \log(x(t)) = t + c$$

$$\implies x(t) = e^{(x+c)} = Ae^t \text{ if we let } A = e^c$$

Our guess at a solution was close, but the actual solution ended up with an extra A term. In fact we've found **infinitley many** solutions, or the **general solution** since we don't know the value of A that came from the integration constant. Think for second about how we might go about finding what A is.

2.2 Initial Conditions

To find A, we would need to know what $x(t=0)$ is. Let's say for the particular solution we're interested in has $x(t=0)=x_0$. Then its straightforward to solve for A:

$$x(t=0)=x_0=Ae^0=A \implies A=x_0 \implies x(t)=x_0e^t$$

So we've found a **specific** or **unique** solution to this ODE. In general, for an n th-order (the highest order is of n th degree) ODE, you need n iniial conditions to find a specific solution. We'll see why this is in the coming weeks.

2.3 A note on differentiability

When we are concerned with differential equations that show up in classical mechanics, we often secretly make the assumption that the function that we are looking for is differentiable in the first place, i.e. that the function is sufficiently smooth so that $\lim_{t\to 0}\frac{\Delta\mathbf{r}}{\Delta t}$ exists. For the systems we'll concern ourselves with in this class, we will take this for granted. Some differential equation models do run into this being an issue though, such as in fracture mechanics or models of swarming behavior, where one needs to employ what is called nonlocal analysis, which is a really interesting bit of math that we just don't have the time to cover in this class sadly.

But before we can start modeling more or less whatever we'd like with ODEs, we need to get familiar with some different frames of reference, or coordinate systems that solutions to ODEs often live in. That leads us to our first in-class activity.

CHAPTER

THREE

31 AUG 23 - FRAMES AND COORDINATES

3.1 What is a Frame?

A **Frame of Reference**, **Reference Frame**, or simply **Frame** is a set of coordinates. These coordinates describe where things are.

An **inertial frame** is a Frame where newtons first law holds, which means that in this frame a net force of zero means an acceleration of zero. In practice, this means that if you have two frames that are moving at a constant velocity relative to each other (without rotation), these are inertial frames.

3.1.1 Relative Velocities in cartesian coordinates

Suppose you're walking down shaw lane eager to get to BPS for your favorite physics class at constant velocity v_A when you notice a squirrel sitting stationary on the sidewalk d meters in front of you. You're a physics major so you naturally feel the urge to unnececarily mathematically analyze this situation using reference frames.

Let's call your reference frame A and the squirrel's B. We'll ignore height differences between you and the squirrel.

3.1.2 Questions

A note on notation: $r_{A/B}$ means "the position of **object** A in **frame** B," or simply "the position of A relative to B."

▯ Do this

Answer the following Questions:

1. What is the squirrels position in your frame, $r_{B/A}$?
2. What is your position in the squirrel's frame, $r_{A/B}$?
3. What is your velocity in your frame, $\dot{r}_{A/A} = v_{A/A}$?
4. What is the sqirrel's velocity in your frame, $v_{B/A}$?
5. What is your velocity in the squirrels frame, $v_{B/A}$?
6. Suppose that the squirrel starts walking to the right at the same velocity that you are walking at. What is $v_{B/A}$ now?

Hopefully these are intuitive from what you've learned back in PHY 183, but there are subtleties to the math you just did. Is there a relationship between $v_{B/A}$ and $v_{A/B}$?

In general, for inertial frames A, B, and object C one has the following:

$$\mathbf{r}_{A/C} = \mathbf{r}_{A/B} + \mathbf{r}_{B/C}$$

taking a derivative gives:

$$\dot{\mathbf{r}}_{A/C} = \dot{\mathbf{r}}_{A/B} + \dot{\mathbf{r}}_{B/C}$$

This derivative works out so simply only because the **unit vectors are fixed in cartesian coordinates**. That is, if we write a generic vector in one of these frames as $\mathbf{r} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}$, and then take a time derivative, we simply get $\dot{\mathbf{r}} = \frac{d}{dt}\mathbf{r} = \frac{dx}{dt}\hat{\mathbf{x}} + \frac{d\hat{\mathbf{x}}}{dt}x + \frac{dy}{dt}\hat{\mathbf{y}} + \frac{d\hat{\mathbf{y}}}{dt}y + \frac{dz}{dt}\hat{\mathbf{z}} + \frac{d\hat{\mathbf{z}}}{dt}z = \dot{x}\hat{\mathbf{x}} + \dot{y}\hat{\mathbf{y}} + \dot{z}\hat{\mathbf{z}}$ because the derivatives of the unit vectors vanish. Later, you'll figure out what happens when you have non-constant unit vectors.

Let's use these transformations in a problem.

A second squirrel appears, which we'll call squirrel C. This scares the first squirrel into running toward you, while the new squirrel runs away.

3.1.3 Questions

Do this

Answer the following question:

What is the velocity of B relative to C?

3.1.4 The Galilean Transformation

We just did an example of a **Galilean Transformation**. Traditionally, if we have a frame B that with moves with constant velocity $v_{B/A}$ with respect to frame A, we re-write the above equations as the following:

$$\mathbf{v}_B = \mathbf{v}_A - \mathbf{v}_{B/A}$$

$$\mathbf{r}_B = \mathbf{r}_A - \mathbf{v}_{B/A}t$$

Where we have taken $_A$ to mean $_{C/A}$ and $_B$ to mean $_{C/B}$. Also note that since we are ignoring relativistic effects, we also have that:

$$t_B = t_A$$

3.1.5 Checking the physics

This is all well and good, but we should make sure to check that newtonian mechanics works the same in both of these frames, that they are really inertial frames.

Do this

In your group, show that forces experienced in frame A are the same as forces experienced in frame B. (Hint: $v_{B/A}$ is constant)

3.2 Forces in polar coordinates

Many problems in physics require the use of non-cartesian coordinates, such as the Hydrogen atom or the two-body problem. One such coordinate system is **polar coordinates**. In this coordinate system, any vector $\mathbf{r} \in \mathbb{R}^2$ is described by a distance r and angle ϕ insead of cartesian coordinates x and y. The following four equations show how points transform in these coordinate systems.

$$x = r\cos\phi \qquad\qquad y = r\sin\phi$$

$$r = \sqrt{x^2 + y^2} \qquad \phi = \arctan(y/x)$$

This also let's us write kinetic energy as $T = \frac{1}{2}m(\dot{r} + r^2\dot{\phi}^2)$.

We'd like to know how to write down Newton's second law in this coordinate system, but this is not as simple as before because the unit vectors in polar coordinate are NOT constant. We'll denote the unit vectors for polar coordinates by $\hat{\mathbf{r}}$ and $\hat{\phi}$. $\hat{\mathbf{r}}$ points in the direction of increasing r with ϕ fixed, and similarly $\hat{\phi}$ points in the direction of increasing ϕ with r fixed.

However, this doesen't stop us from being able to break down a net force into its components along each unit vector:

$$\mathbf{F} = F_r\hat{\mathbf{R}} + F_\phi\hat{\phi}$$

And the second law still holds:

$$\mathbf{F} = m\ddot{\mathbf{r}}$$

It would be very useful if we had expressions for F_r and F_ϕ in terms of r and ϕ. Toward finding these, we can start by writing:

$$\mathbf{r} = r\hat{\mathbf{r}}$$

3.2.1 Task

? Do this

Find $\dot{\mathbf{r}}$ by differentiating $\mathbf{r}$ with respect to time. (see hint below)

? Do this

Find $\ddot{\mathbf{r}}$ by differentiating $\dot{\mathbf{r}}$ with respect to time. Then find F_r and F_ϕ. (see hint below)

Hint: During these problems, you'll need to find expressions for $\frac{d\hat{\mathbf{r}}}{dt}$ and $\frac{d\hat{\phi}}{dt}$, which these pictures might help with.

Angular momentum

One of the powerful things you get from using this coordinate system is a handy way to represent angular momentum:

$$|\mathbf{L}| = |\mathbf{r} \times \mathbf{p}| = |mr^2\dot{\phi}|$$

In the interest of time we won't go through the whole calculation to arrive at this, but this is handy to know.

3.2.2 (time permitting) Non-Inertial Frames

Broadly speaking, non-inertial frames are frames that undergo some sort of acceleration. In general for inertial frame A and non-inertial frame B, we can write:

$$r_B = r_A - r_{B/A}$$

$$v_B = v_A - v_{B/A}$$

as before, as well as:

$$\ddot{r}_B = \ddot{r}_A - \ddot{r}_{B/A}$$

Interestingly, this means a new (ficticious) force has sprung into being, equal to $-m\ddot{r}_{B/A}$! This lets us write newton's second law for frame B as:

$$m\ddot{r}_B = F - m\ddot{r}_{B/A}$$

3.2.3 Leaning on a bus problem

Suppose you're standing on a bus that is accelerating forward with constant acceleration $r_{B/A}$. If we approximate you as a pendulum, what angle should you lean at so you can stay at equillibrium without having to exert any force to stay at that angle?

□ Do this

Solve this problem using the the as a non-inertial frame. Solution

3.2.4 Forces in Rotating Frames

For intertial frame A and non-inertial, rotating frame B, rotating at an angular velocity of □ relative to A, we can write the second law as:

$$m\ddot{\mathbf{r}} = \mathbf{F} + 2m\dot{\mathbf{r}} \times \square + m(\square \times \mathbf{r}) \times \square$$

Where $\mathbf{F}$ is any "standard" forces from inertial frames, $2m\dot{\mathbf{r}} \times \square$ is the Coriolis force, and $m(\square \times \mathbf{r}) \times \square$ is the centrifugal force.

CHAPTER

FOUR

5 SEP 23 - CALCULUS OF VARATIONS AND LAGRANGIAN DYNAMICS

The name of the game in calculus of variations is finding minimums,maximums, or stationary points of integrals that have the form:

$$S = \int_{x_1}^{x_2} f[y(x), \dot{y}(x), x]dx$$

While you are trying to find the minimize S, what you end up finding is the **function** $y(x)$ that satisfies this minimization. It turns out that for S to have extrema, the Euler-Lagrange equation (below) must be satisfied.

$$\frac{\partial f}{\partial y} - \frac{d}{dx}\left(\frac{\partial f}{\partial \dot{y}}\right) = 0$$

In practice, when approaching a varational problem, the typical worflow if something like this:

1. Write your problem down in the form of an integral like S.
2. Use the Euler-Lagrange equation to get a differential equation for the unknown function y.
3. Solve the differential equation.

We can extend this framework for use in classical mechanics by defining the lagrangian of a system with independent, generalized coordinates $(q_1, \dot{q}_1 ... q_n, \dot{q}_n)$ as the kinetic energy minus potential energy of a system:

$$\mathcal{L}(q_1, \dot{q}_1 ... q_n, \dot{q}_n) = T(q_1, \dot{q}_1 ... q_n, \dot{q}_n) - V(q_1, \dot{q}_1 ... q_n, \dot{q}_n)$$

Then we write the action of the system as:

$$S = \int_{t_1}^{t_2} \mathcal{L}(q_1, \dot{q}_1 ... q_n, \dot{q}_n)dt$$

It turns out that the path a system takes between points 1 and 2 in the generalized coordinates is the path such that S is stationary. This is called the principle of least action. This lets us leverage the Euler-Lagrange equation for the generalized coordinates of our system q_n.

$$\frac{\partial \mathcal{L}}{\partial q_i} - \frac{d}{dx}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) = 0$$

This gives us n equations of motion (EOM) for our system. Note how we didn't have to know anything about the forces acting on our system to arrive at equations of motion.

4.1 Activity

4.1.1 Simple Harmonic Oscillator (SHO)

✎ Do this

1. Starting with the 1d energy equations (T and V) for a SHO; derive the equations of motion. Did you get the sign right?

4.1.2 Canonical Coupled Oscillators

Let's assume you have a chain of two mass connected by springs (all with the same k) as below.

✎ Do this

1. Write down the energy equations for this system (using x_1 and x_2 for coordinates)
2. Write the Lagrangian and derive the two equations of motion.
3. Do all the signs makes sense to you?
4. Could you have arrived at these equations in the newtonian framework?

4.1.3 2-Body Problem

Consider the 2 body problem of a star and an orbiting planet under the force of gravity. Assume the star is stationary.

✎ Do this

1. Write down the energy equations for this system using polar coordinates.
2. Write the Lagrangian and derive 2 equations of motion for r and ϕ

4.2 Adding Constraint Forces

The Lagrangian framework also excells at dealing with constrained motion, where it is usually not obvious what the constraint forces are. This is because you can write your generalized coordinates for your system in such a way that it contains the information

Consider a particle of mass m constrained to move on the surface of a paraboloid $z = r^2$ subject to a gravitational force downward, so that the paraboloid and gravity are aligned.

✎ Do this

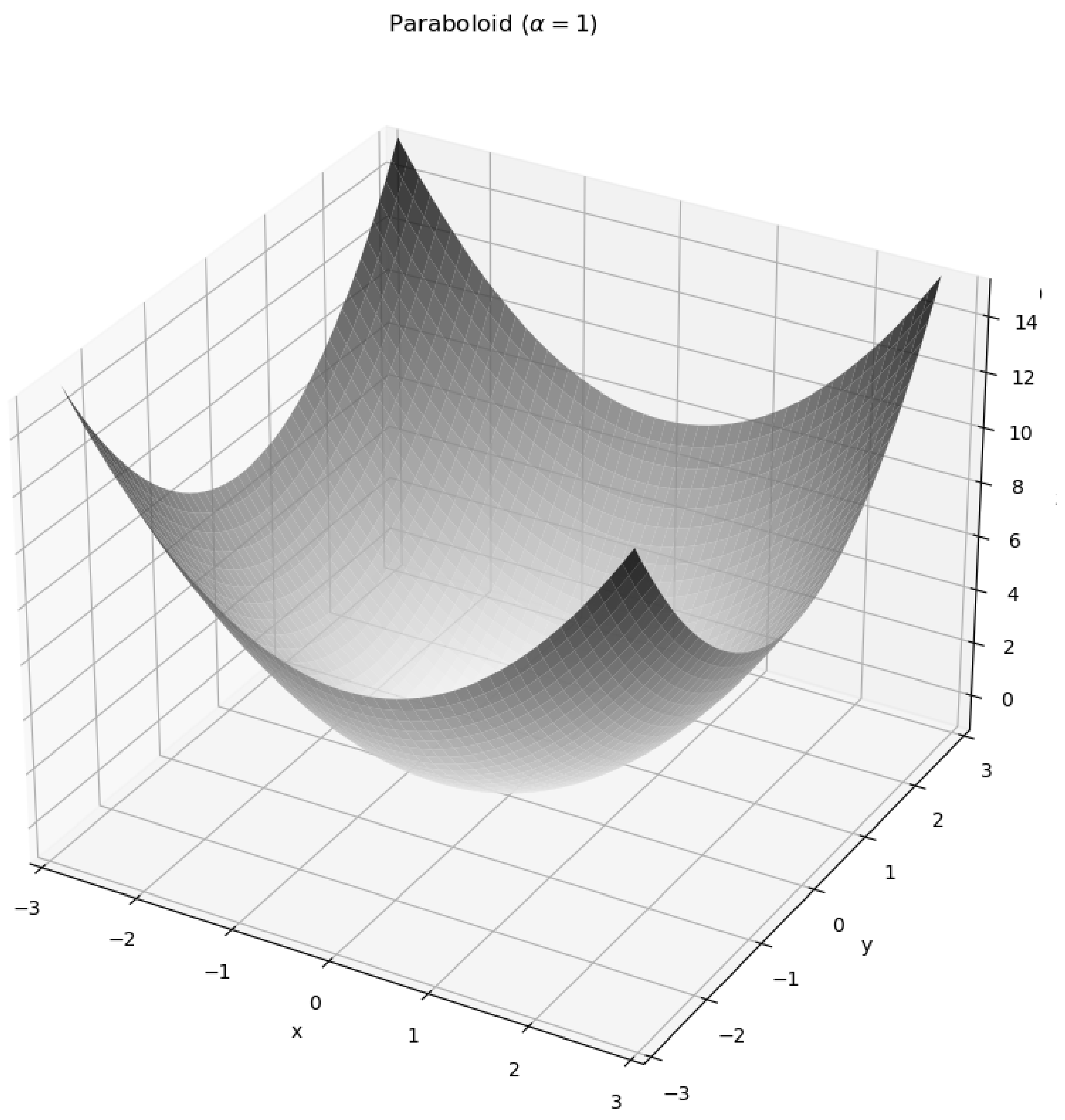
1. Using cylindrical coordinates (why?), write down the equation of constraint. Think about where the mass must be if it's stuck on a paraboloid.
2. Write the energy contributions in cylindrical coordinates. (This is where you put in the constraint!)
3. Form the Lagrangian and find the equations of motion (there are two!)

```
import numpy as np
import matplotlib.pyplot as plt

def parabaloid(x,y,alpha):
    # function of a paraboloid in Cartesian coordinates
```

(continues on next page)

(continued from previous page)

```
    return alpha * (x**2 + y**2)

# points of the surface to plot
x = np.linspace(-2.8, 2.8, 50)
y = np.linspace(-2.8, 2.8, 50)
alpha = 1
# construct meshgrid for plotting
X, Y = np.meshgrid(x, y)
Z = parabaloid(X, Y,alpha)

# do plotting
fig = plt.figure(figsize = (10,10))
ax = plt.axes(projection='3d')
plt.title(r"Paraboloid ($\alpha = $" + str(alpha)+ ")")
ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z, cmap='binary', alpha=0.8)
ax.set_xlim(-3, 3); ax.set_ylim(-3, 3); ax.set_zlim(-1 ,15)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.show()
```



4.2.1 Roller Coaster

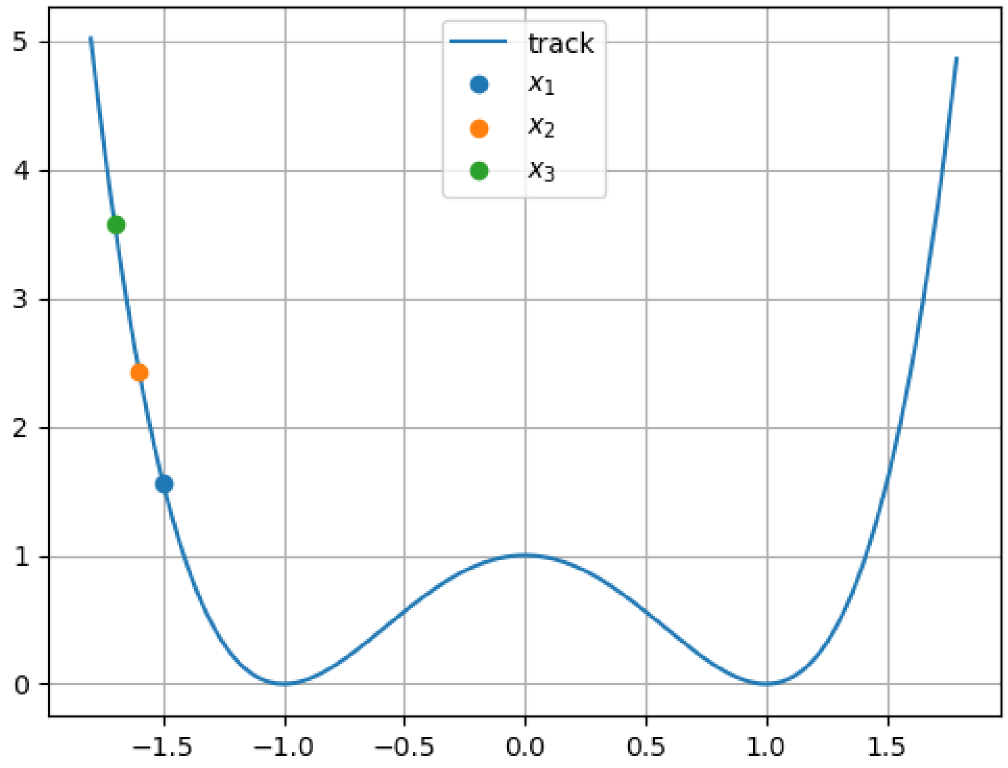
Consider 3 roller coaster cars of equal mass m and positions x_1, x_2, x_3, constrained to move on a one dimensional "track" defined by $f(x) = x^4 - 2x^2 + 1$. These cars are also constrained to stay a distance d apart, since they are linked. We'll only worry about that distance d in the direction for now (though a fun problem would be to try this problem with a true fixed distance!)

```
x = np.arange(-1.8,1.8,0.01)
track = lambda x : x**4 - 2*x**2 + 1
y = track(x)
d = 0.1
x1_0 = -1.5
x2_0 = x1_0 - d
```

(continues on next page)

(continued from previous page)

```
x3_0 = x1_0 - 2*d
plt.plot(x,y, label = "track")
plt.scatter(x1_0,track(x1_0),zorder = 2,label = r"$x_1$")
plt.scatter(x2_0,track(x2_0),zorder = 2,label = r"$x_2$")
plt.scatter(x3_0,track(x3_0),zorder = 2,label = r"$x_3$")
plt.legend()
plt.grid()
plt.show()
```



? Do this

1. Write down the equation(s) of constraint. How many coordinates do you actually need?
2. Write the energies of the system using your generalized coordinates.
3. Form the Lagrangian and find the equation(s?) of motion (how many are there?)
4. Are the dynamics of this system different that the dynamics of a system of just one roller coaster car?

CHAPTER

FIVE

7 SEP 23 - NUMERICAL INTEGRATION AND MORE LAGRANGIANS

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

Now that we have an idea how to find a wealth of interesting ordinary differential equations using lagrangians, we'll work on building up ways to understand these equations, their solutions and behavior. The issue with this is that **most ODEs do not have analytical solutions**. That means we can't write down nice closed-form solutions for them using trancendental functions. However, don't despair, because that does not mean there is no solution. In fact, the vast majority of non-pathological ODEs one might come across in physics are **garunteed** to have unique solutions (at least for finite time). We can easily calculate these solutions using **numerical integration**. Next week we'll also see how we can characterize the behavior of ODEs even without acess to numerical integration.

Numerical Integration is a vast and wide topic with lots of different approaches, important nuances, and difficult problems. Some of the most high profile numerical integration was done by NASA's human computers – a now well-known story thanks to the film Hidden Figures. Black women formed a core group of these especially talented scientists (including Mary Jackson, Katherine Johnson, and Dorothy Vaughn), without whom, John Glenn would not have orbited the Earth in 1962. This is also a very interesting story about the importance of Historically Black Colleges and Universities to American science.

5.1 Harmonic Oscillator

Let's start simple with everyone's favorite differential equation, the simple harmonic oscillator. Recall that we can write the SHO as:

$$\ddot{x} = -\omega_0^2$$

where $\omega_0^2 = \frac{k}{m}$. This equation is 2nd order, but numerical integration techniques only work on 1st order equations. Thankfully they work on any number of potentially coupled 1st order equations. This means that with a quick change of variables, we can write the SHO as a system of 2 first order equations by introducing a new variable v equal to the velocity of the oscillator.

$$v = \dot{x}$$

Then the accelleration of the oscillator can be written as:

$$\dot{v} = -\omega_0^2$$

This trick for writing higher order differential equations as first order equations is incredibly common.

5.2 Setting up to numerically integrate

We need a few things to numerically integrate using `solve_ivp` in python.

5.2.1 1. Derivatives Function

First, we need to set up a derivatives function that calculates and returns a list of the values of the first order derivatives given an imput list of current values. These current values represent a location in **phase space**. Phase Space is a space that contains all the information about the state of an ODE. The simple harmonic oscillator has a 2D phase space since its state is totally defined by its position and velocity.

Here's what our derivatives function looks like for a SHO:

```
def diffyqs(t,curr_vals, omega2):
    # 2 first-order differential equations for a SHO
    # first 2 arguments are always t and curr_vals, which are followed by any␣
↪parameters of your ODEs
    x, v = curr_vals   # unpack current values

    vdot = -omega2 * x # calculate derivative

    return v,vdot # return derivatives
```

We will pass this function to our solver, which will give us back integrated solutions of our list of derivatives. So since $v = \dot{x}$, our solution will return x first, and v.

5.2.2 2. Time Setup

We need to define the time span to solve the ODE for AND the specific times we'd like solution points for. Here it is also convienient to choose a time step dt. Here's one way we could do this in python:

```
tmax = 15
dt = 0.1
tspan = (0,tmax)         # time span
t = np.arange(0,tmax,dt) # specific times to return solutions for
```

5.2.3 3. Parameters and Initial Conditions

Since we're dealing with ODEs, we need to supply an initial condition to be able to solve. The SHO has 2D phase space so we need 2 values for our initial condition. We'll also define parameter value(s) in this step.

```
omega2 = 2
initial_condition = [1, 0] # pull back 1m, no initial velocity
```

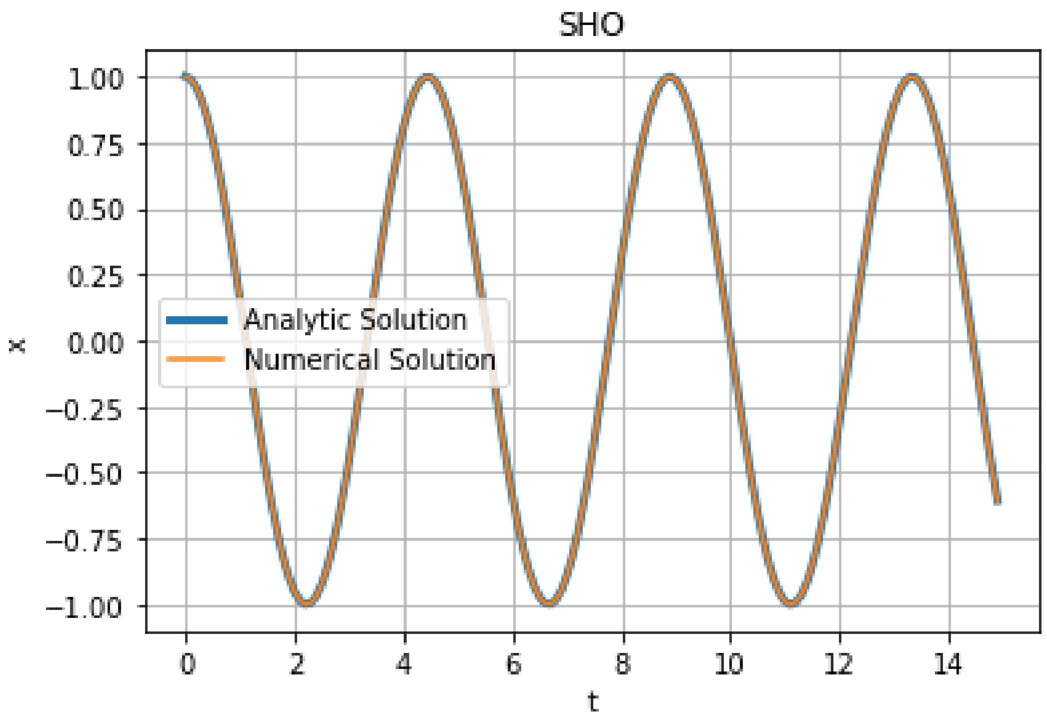
5.2.4 4. Call Integrator

Now all we have left to do is to actually use `solve_ivp` to do the integration. The syntax for how to do this is shown below. We also get the oppourtunity to tell `solve_ivp` exactly what numerical integration method we'd like it to use. For now we can think of the integrator as a magic box and choose `RK45`, or a Runge-Kutta 4th order method.

```
solved = solve_ivp(diffyqs,tspan,initial_condition,t_eval = t, args = (omega2,),
 ↪method="RK45")
```

To access the solution directly, use `solved.y`. `solved.y[0]` is the solved for position array and `solved.y[1]` is the velocity array in this case. Now let's see a full implementation of this below, including some visualization that compares our numerical solution to the analytical solution of the SHO.

```
# 1. Derivatives Function
def diffyqs(t,curr_vals, omega2):
    x, v = curr_vals
    vdot = -omega2 * x
    return v,vdot

# 2. Time Setup
tmax = 15
dt = 0.1
tspan = (0,tmax)
t = np.arange(0,tmax,dt)

# 3. Parameters and Initial Conditions
omega2 = 2
initial_condition = [1, 0]

# 4. Call Integrator
solved = solve_ivp(diffyqs,tspan,initial_condition,t_eval = t, args = (omega2,),
 ↪method="RK45")

# 5. Visualization and Comparision to analytical solution
def analytic_sol(t,omega0,initial_condition):
    x0,v0 = initial_condition
    return (v0/omega0)*np.sin(omega0*t) + x0 * np.cos(omega0*t)

plt.plot(t,analytic_sol(t,omega2**0.5,initial_condition),label = "Analytic Solution",
 ↪linewidth = 3)
plt.plot(t,solved.y[0],label = "Numerical Solution")
plt.title("SHO")
plt.xlabel("t")
plt.ylabel("x")
plt.legend()
plt.grid()
plt.show()
```



✎ Do this

1. Plot the numerically calculated trajectory in phase space. Use velocity as the y axis and position as the y axis. What shape does it make?
2. Add a drag term equal to $-\beta v$ with $\beta \in [0, 1)$ and numerically integrate again. What does this trajectory look like on x vs t plots and phase space plots?

5.3 Back to Paraboloid Paradise

Let's consider the problem we we're working on on Tuesday, Where a particle was constrained to move on the surface $z = r^2$. The EOM we arrived at are complex ($\ddot{r} = \frac{1}{1+4r^2}(-4rv^2 + r\omega^2 - 2gr)$ and $\ddot{\theta} = -2\frac{v\omega}{r}$)and it was unclear if those ODEs had solutions at all. Now That we're armed with numerical integration, we can tackle the problem.

✎ Do this

Introduce variables v and ω to use our trick for reducing > 1 order differential equations to first order equations to write the equations of motion for this problem as a system of four first order differential equations (shown below).

$$\dot{r} = ??$$

$$\dot{v} = ??$$

$$\dot{\theta} = ??$$

$$\dot{\omega} = ??$$

✎ Do this

Use these equations to correct the `diffyqs` function in the cell below.

```python
# 1. Derivatives Function
def diffyqs(t,curr_vals, g):

    r, v, theta, omega = curr_vals

    vdot = 0

    omegadot = 0

    return v, vdot, omega, omegadot # solution will return in this order, but␣
 ↪integrated (r,v,theta,ω)

# 2. Time Setup
tmax = 40
dt = 0.01 # unneccecarily small dt to make plot super smooth
t = np.arange(0,tmax,dt)

# 3. Parameters and Initial Conditions
g = 9.81
x0 = [2.6,0,0,2]

# 4. Call Integrator
solved = solve_ivp(diffyqs,(0,tmax),x0,t_eval = t, args = (g,),method="RK45")
```

� Do this

1. Make r vs t and theta vs t plots of this trajectory. Can you think of what that trajectory would look like in cartesian coordinates? 2. Run the cell below to see what the trajectory looks like in 3D. How does the true trajectory compare to your prediction?

2. Change the initial condtion to examine the following cases and plot the trajectories in 3d:

   a. Particle starts from rest and is let go

   b. Particle starts at a given height and is given a low speed (less than needed to orbit)

   c. Particle starts at a given height and is given a low speed (more than needed to orbit)

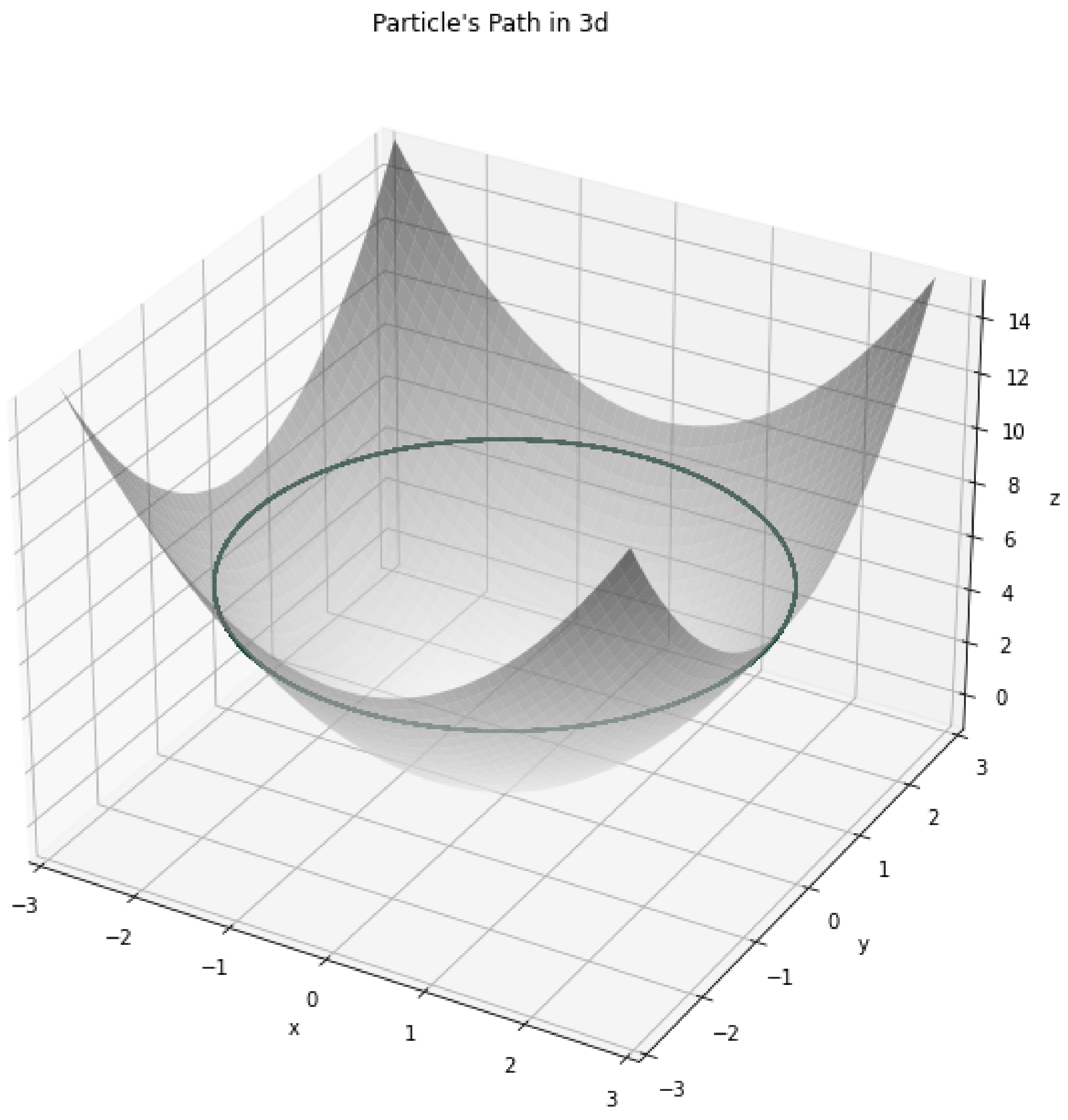
   d. Can you find a flat horizontal circular orbit?

```python
def parabaloid(x,y,alpha=1.):
    # function of a paraboloid in Cartesian coordinates
    return alpha * (x**2 + y**2)

def cylindrical_to_cartesian(r, th):
    # convert back to cartesian coordinates for ease of plotting
    r = np.array(r)
    th = np.array(th)
    x = r*np.cos(th)
    y = r*np.sin(th)
    return x,y,parabaloid(x, y)

def plot_solution(solved):
    # Function to plot the trajectory

    # points of the surface to plot
    x = np.linspace(-2.8, 2.8, 50)
    y = np.linspace(-2.8, 2.8, 50)
    alpha = 1
```

(continues on next page)

(continued from previous page)

```
    # construct meshgrid for plotting
    X, Y = np.meshgrid(x, y)
    Z = parabaloid(X, Y,alpha)

    # get trajectory in cartesian coords
    xtraj, ytraj, ztraj = cylindrical_to_cartesian(solved.y[0], solved.y[2])

    # plot plot plot
    fig = plt.figure(figsize = (10,10))
    ax = plt.axes(projection='3d')
    plt.title("Particle's Path in 3d")
    ax.plot_surface(X, Y, Z, cmap='binary', alpha=0.5)
    ax.plot3D(xtraj, ytraj, ztraj, c = "#18453B")
    ax.set_xlim(-3, 3); ax.set_ylim(-3, 3); ax.set_zlim(-1 ,15)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('z')
    plt.show()

plot_solution(solved)
```



5.4 How Does Numerical Integration actually work?

A computer understands things like updating individual variables with a change. It turns out this process of updating things in steps is the basis for numerical integration. We need a set of update equations. Making those update equations is effectively choosing our integrator.

5.4.1 Update equations

The critical part of numerical integration is approximating the change to variables you are investigating. Going back to our differential equations, we can rewrite them as approximate equation, which a computer understands because it involves discrete steps. How we choose to approximate this update indicates which integration routine we've chosen and sets the irreducible error we are stuck with (i.e., $O((\Delta t)^2)$, $O((\Delta t)^3)$, etc.)

We will illustrate three approximations to the slope of these functions:

- **Euler-Cromer (EC)** - definitely the most intuitive of the approaches, where we approximate the slope with two points separated by Δt in time. It is quick to write, slow to solve, and requires small steps for accurate results. Even so, it fails to integrate periodic motion well because it doesn't always conserve energy in periodic motion. Turns out it's the best tool to use when you have random noise added to the model though (e.g., $\eta_n(\sigma(t))$). For a first order eqn, $\dot{x} = f(x,t)$,

$$x(t+\Delta t) = x(t) + \text{change} = x(t) + \Delta t\left(f(x(t+\frac{1}{2}\Delta t), t+\frac{1}{2}\Delta t\right)$$

- **Runge-Kutta 2nd order (RK2)** - just a step above Euler-Cromer; it uses three points to approximate the slope giving two measures of the slope (hence, 2nd order). It's not much more complex than Euler-Cromer, but gives an order of magnitude lower error. It's a good starting point for simple systems. For a first order eqn, $\dot{x} = f(x,t)$,

$$k_1 = \Delta t\left(f(x,t)\right),$$
$$k_2 = \Delta t\left(x + \frac{1}{2}k_1, t+\frac{1}{2}\Delta t\right),$$
$$x(t+\Delta t) = x(t) + \text{change} = x(t) + k_2$$

- **Runge Kutta 4th order (RK4)** - this is the gold standard. Most researchers start with RK4 on most problems. It uses 5 points to build 4 slope profiles and integrates the system in 4 steps. It is highly adaptable and supported – it can be modified to take smaller or longer steps depending on the specific nature of the problem at the time. I mean that it can change step size in the middle of its work; including within the step it is taking presently. For a first order eqn, $\dot{x} = f(x,t)$,

$$k_1 = \Delta t\left(f(x,t)\right),$$
$$k_2 = \Delta t\left(x + \frac{1}{2}k_1, t+\frac{1}{2}\Delta t\right),$$
$$k_3 = \Delta t\left(x + \frac{1}{2}k_2, t+\frac{1}{2}\Delta t\right),$$
$$k_4 = \Delta t\left(x + k_3, t+\Delta t\right),$$
$$x(t+\Delta t) = x(t) + \text{change} = x(t) + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

We don't expect you memorize these approaches or to derive them, but to understand how they work and what their limitations are.

5.5 Numerically analyzing this system

5.5.1 Analysis of the Energy

We know that this is a system that should conserve energy. There's no dissipation and there's only conservatives forces acting to change the speed of the object. The contact forces only change direction, so the system is "conservative".

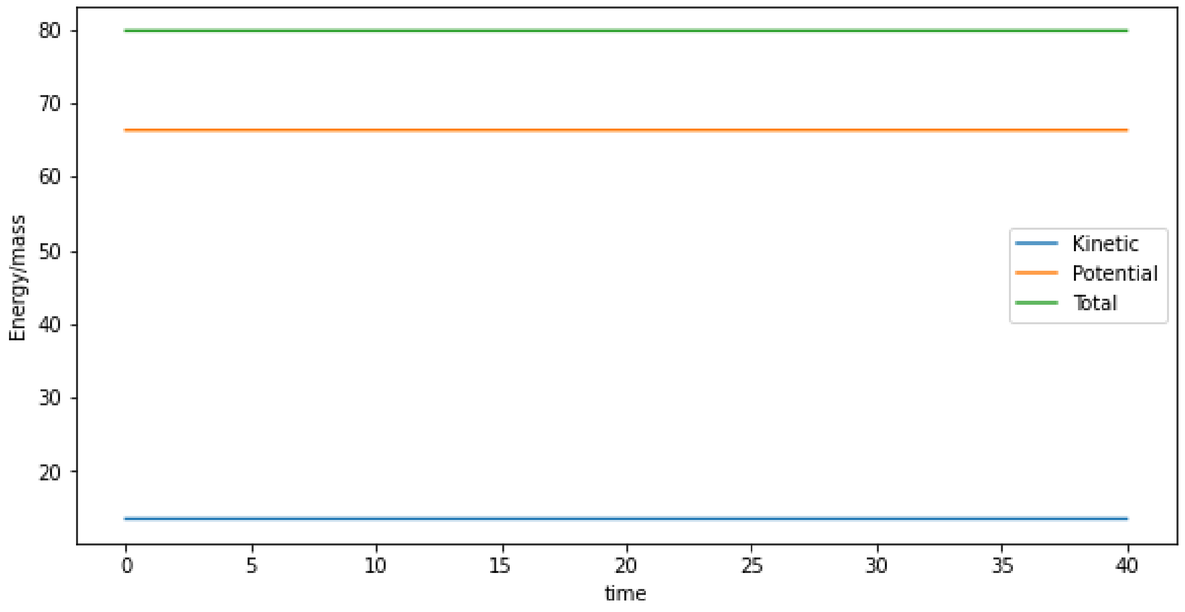
Let's compute the energy:

$$T = \frac{m}{2}\left(\dot{r}^2 + r^2\dot{\theta}^2 + \dot{z}^2\right) = \left(\dot{r}^2 + r^2\dot{\theta}^2 + 4r^2\dot{r}^2\right)$$

$$U = mgz = mgr^2$$

We have all these quantities except m. Let's divide it out for just set it to 1. The code below computes and plots the energies.

```
# Unpack solution in a convienient way
r,v,theta,omega = solved.y

## Kinetic
T = 0.5*(v**2 + r**2 * omega**2 + 4 * r**2 * v**2)

## Potetial
U = g*r**2

## Total
E=T+U

ax = plt.figure(figsize=(10,5))
plt.plot(t, T, label='Kinetic')
plt.plot(t, U, label='Potential')
plt.plot(t, E, label='Total')
plt.legend()
plt.xlabel('time')
plt.ylabel('Energy/mass')
plt.show()
```



5.6 Angular Momentum Analysis

We argued that the equation \$$\frac{d}{dt}\left(mr^2\dot{\theta}\right) = 0$\$ was a statement of conservation of the z-component of angular momentum.

Recall that angular momentum is a vector quantity and can be conserved in total, but also a given component might be conserved while others are not. Let's compute the angular momentum and see what the deal is. This will involve taken cross products in cylindrical coordinates (which also obey the right hand rule!).

Starting with the classical relationship:

$$\frac{\mathbf{L}}{m} = \mathbf{r} \times \mathbf{v}$$

We can write down position and velocity vectors in general:

$$\mathbf{r} = r\hat{r} + z\hat{z}$$

$$\mathbf{v} = v_r\hat{r} + v_\theta\hat{\theta} + v_z\hat{z}$$

Let's take the cross product:

$$\mathbf{r} \times \mathbf{v} = (r\hat{r} + z\hat{z}) \times \left(v_r\hat{r} + v_\theta\hat{\theta} + v_z\hat{z}\right)$$

Which is

$$\frac{\mathbf{L}}{m} = \mathbf{r} \times \mathbf{v} = -(zv_\theta)\hat{r} + (zv_r - rv_z)\hat{\theta} + rv_\theta\hat{z}$$

Or:

$$\frac{L_r}{m} = -(zv_\theta)$$

$$\frac{L_\theta}{m} = (zv_r - rv_z)$$

$$\frac{L_z}{m} = rv_\theta$$

Yep, L_z just pops out:

$$\frac{L_z}{m} = rv_\theta = r^2\dot{\theta}$$

$$L_z = mr^2\dot{\theta}$$

A good physics question is 'why?'

Let's plot it

? Do this

Calculate and plot L_z/m vs t.

```
## your code here
```

? Do this

Calculate L_r/m, L_θ/m, and L_{tot}/m. Plot these against time alongside L_z/m, all on the same plot. What do you observe?

```
# your code here
```

5.7 Constraints Revisited - Lagrange Multipliers

Thus far we've been treating constrained motion problems by including the constraint information in the generalized coordinates themselves, but there is a more general approach for these kinds of problems - Lagrange Multipliers. Toward finding a modified Euler-Lagrange equation that deals with these, we can start by looking at a way that one might arrive at the original Euler-Lagrange equation.

We start By defining $L = T - V$ and we define the action of our system by:

$$S = \int_{t_1}^{t_2} (\mathbf{q}, \dot{\mathbf{q}}, t) dt$$

Via the principle of least action we know that $\delta S = 0$ for our ideal path since the S integral is stationary along that path. If we nudge our generalized coordinates a bit:

$$\mathbf{q} \rightarrow \mathbf{q} + \delta \mathbf{q}$$

This lets us define a small change in the lagrangian:

$$\delta L = \frac{\partial L}{\partial \mathbf{q}} \cdot \delta \mathbf{q} + \frac{\partial L}{\partial \dot{\mathbf{q}}} \cdot \delta \dot{\mathbf{q}}$$

Which in-turn gives us a small change in the action:

$$\delta S = \int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \mathbf{q}} \cdot \delta \mathbf{q} + \frac{\partial L}{\partial \dot{\mathbf{q}}} \cdot \delta \dot{\mathbf{q}} \right) dt$$

Integrating by parts gives: $= \int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \mathbf{q}} \cdot \delta \mathbf{q} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) \cdot \delta \mathbf{q} \right) dt + \left[\frac{\partial L}{\partial \dot{\mathbf{q}}} \cdot \delta \mathbf{q} \right]_{t_1}^{t_2} = \int_{t_1}^{t_2} \left(\frac{\partial L}{\partial \mathbf{q}} - \frac{d}{dt} \left(\frac{\partial L}{\partial \mathbf{q}} \right) \right) \cdot \delta \mathbf{q} dt$

Which finally lets us argue: $\sum_{i=0}^{n} \left(\frac{\partial L}{\partial q_i} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) \right) \delta q_i = 0$

Since our generalized coordinates are independent, this gives us the original Euler-Lagrange equation:

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) = 0$$

But let's say our coordinates are constrained by some $f(\mathbf{q}, t) = 0$. Noting that $\delta f = \frac{\partial f}{\partial \mathbf{q}} \cdot \delta \mathbf{q} = 0$, we can modify the above equation to read:

$$\sum_{i=0}^{n} \left(\frac{\partial L}{\partial q_i} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) + \lambda \frac{\partial f}{\partial q_i} \right) \delta q_i = 0$$

Where we've introduced λ, a **lagrange multiplier**. This gives us a new modified version of the Euler-Lagrange Equation:

$$\frac{\partial L}{\partial q_i} - \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) + \lambda \frac{\partial f}{\partial q_i} = 0$$

The beautiful thing about this equation is that λ itself often will take the form of the constraint force itself (though this is nuanced and depends on the system at hand). So if you do want to know what your constraint forces are - for example say you're designing a roller coaster and you need to know the forces that passengers experience for safety reasons - you can still obtain them via the lagrangian approach, which still lets you bypass the Newtonian framework.

✎ Do this

Use the modified Euler-Lagrange equation with generalized coordinates $\mathbf{q} = (r, \theta, z)$ and constraint equation $f = r^2 - z = 0$ to find the equations of motion for the paraboloid problem. Also solve for λ. What is the physical meaning of this lagrange multiplier?

Note: The procedure here is a bit different now, since you impose the constraint **after** iterating the modified Euler-Lagrange equation 3 times. Then you'll have a system of equations to solve for $\ddot{r}$ and λ.

CHAPTER

SIX

12 SEP 23 - THE DYNAMICAL SYSTEMS APPROACH AND PHASE PORTRAITS

Up to now, most of your work with models in physics are those you can solve analytically in terms of known functions. Think about solving differential equations that produce polynomials or sines and cosines.

But what happens when the solution to the problem is not obviously tractable in an analytical form. Rather, how can we investigate systems that are new to us?

In today's activity you will:

- Remind yourself how to interpret a 1d phase portrait using the differential equation $\dot{x} = x^2 + 1$
- Remind yourself how to interpret a 2d phase portrait (phase space plot) using the SHO model
- Explain what you see in the phase space figure for the SHO
- Develop the ODE for the large angle pendulum
- Show how we can recover the SHO using mathematics and graphs
- Use an existing program to work with a new system
- Explain the insights developed from a phase space plot of the Large Angle Pendulum
- (if time) Plot a trajectories in the phase space
- (if time) Add damping to the model
- (if time) Analyze fixed points for 2d systems

6.1 1D System

Let's look at a simple 1d system first, given by the differential equation:

$$\dot{x} = x^2 - 1$$

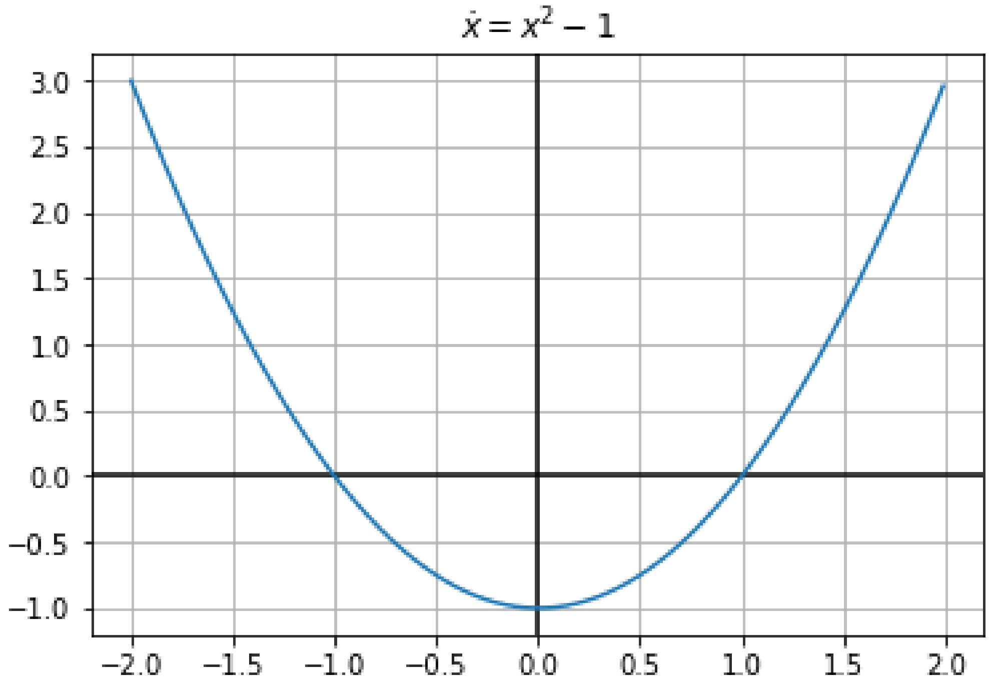
Let's start by looking at what the plot of this differential equation looks like:

```python
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-2,2,0.01)
xdot  = x**2 - 1
plt.title(r"$\dot{x} = x^2 - 1$")
plt.axvline(x=0, c="black")
plt.axhline(y=0, c="black")
plt.plot(x,xdot)
```

(continues on next page)

(continued from previous page)

```
plt.grid()
plt.show()
```



Since this is a first order equation we have a nice physical way of thinking about it: at position x, a particles velocity is $x^2 - 1$. So when the $x^2 - 1$ is positive, the particle's velocity is positive (or to the right) and vice versa. We can use this knowledge to work out qualitative behavor of our system.

✅ Do this

1. Consider a trajectory that starts at $x = -1.5$. Does this trajectory move to the right or left?
2. What direction does a trajectory that starts at $x = 0$ go?
3. What direction does a trajectory that starts at $x = 1.5$ go?
4. What trends do you notice? Do trajectories settle down at certain locations or blast off to infinity?

6.2 Fixed Points

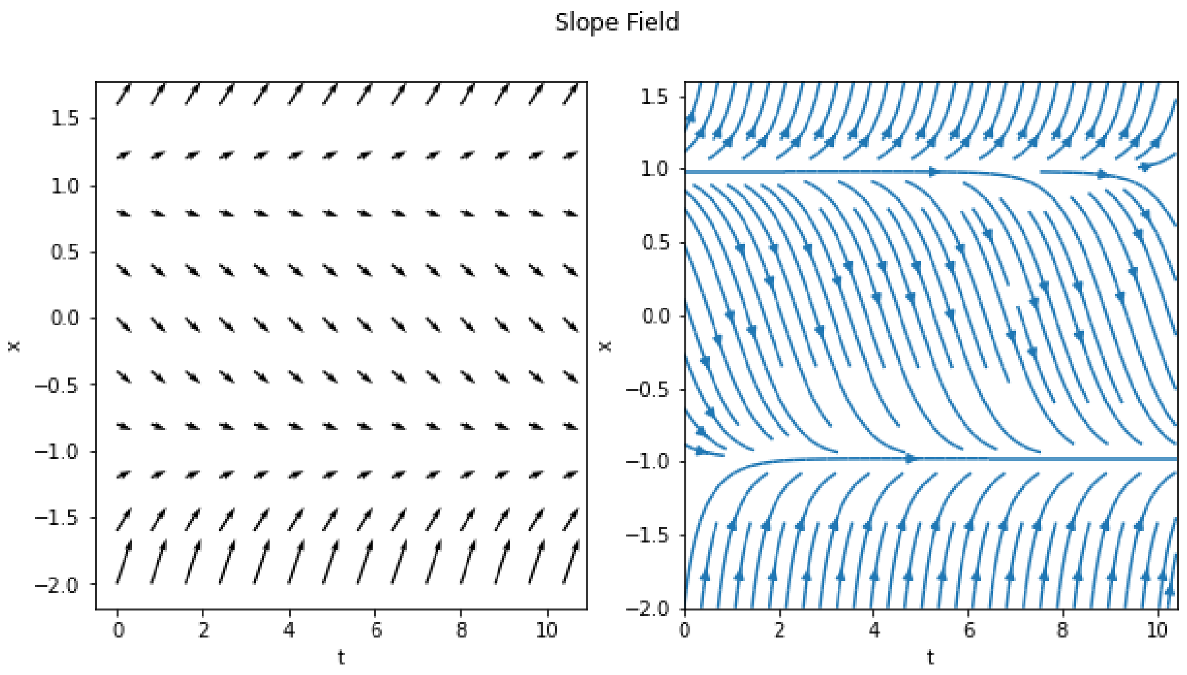
The **fixed points** of a system are where its derivative vanishes, that is $\dot{\mathbf{x}} = 0$. At any fixed point, a system is constant (why?). In the dynamical systems approach, we often care about charactarizing the behavor of systems near fixed points. In 1d, most fixed points are either stable (they attract trajectories) or unstable (they repel trajectories).

✅ Do this

Find and characterize the fixed points of $\dot{x} = x^2 - 1$ as stable or unstable.

Another way we can visualize these is with **slope fields**. Here we essentially plot the slopes $\dot{x}(x)$ over and over again for many values of t. The actual solution of the differential equation will be a curve that is always tangent to the local slope. Below we've plotted this using both `plt.quiver` and `plt.streamplot`

```python
import numpy as np
import matplotlib.pyplot as plt

t = np.arange(0, 11, 0.8)
x = np.arange(-2, 2, 0.4)

# Make grid
T, X = np.meshgrid(t, x)

# calculate derivative (dt is const so just use ones)
dx = X**2 - 1
dt = np.ones(dx.shape)

# plot
fig = plt.figure(figsize = (10,5))
plt.subplot(1,2,1)
plt.quiver(T,X,dt,dx)
plt.xlabel("t")
plt.ylabel("x")
plt.subplot(1,2,2)
plt.streamplot(T,X,dt,dx)
plt.xlabel("t")
plt.ylabel("x")
plt.suptitle("Slope Field")
plt.show()
```



6.3 The Phase Portrait of the SHO

To get this started, let's remind ourselves of the phase portrait of the SHO. Recall that we separated the second order ODE into two first order ODEs, one for x and one for v_x,

$$\dot{x} = v_x$$

$$\dot{v}_x = -\omega^2 x$$

We then map out the phase space with the following conceptual interpretation:

- Phase space is a space in which all possible states of the system can be shown
  - a state is a collection of conditions of the state (it's known position and velocity in our case)
- Each state is a unique point in phase space
  - Think about ordered Cartesian pairs, there's a pair of numbers for every point in a 2D space
- Remember that knowing x_0 and $v_{x,0}$ means we can know $x(t)$ for all time (for that one trajectory/particular solution) given a linear ODE

We map the differential equation to the following conceptual interpretation: **How the state changes depends on location in phase space.** We can understand this as the time derivative for x and v_x change throughout the space.

For our 2D SHO case we are saying that how x and v_x change is proportional to the position in space:

$$\langle \dot{x}, \dot{v}_x \rangle = \langle v_x, -\omega^2 x \rangle$$

The process is:

1. Determine the location(s) of interest (i.e., x, v_x)
2. Compute the change in those quantities at the location (i.e., calculate $\dot{x}$ and $\dot{v}_x$ using our prescribed 1st order ODEs above)
3. At a given point (x_0, $v_{x,0}$), create an arrow the indicates the direction and magnitude of the changes to x and v_x at that location.
   - That arrow represents the local flow of the system at that point
4. Repeat for all points of interest
5. Plot arrows to demonstrate flow of the solutions in phase space

6.3.1 Let's focus on axes first

We talked about how we can look at the axes ($x = 0$ and $v_x = 0$) to help get a sense of the flow in phase space. Below, we have some code that does this in two parts:

1. We created a function to produce arrows of the right length given a line of points
2. We call that function for each axis and for a line at a diagonal

6.3.2 Discussion Question

✎ Do this

1. Review the phase portraits below. Talk with your neighbors about how they are constructed.
   - Work to identify which components of the code look familiar and which you have more questions about
2. Make a fourth plot that looks at the other diagonal line that runs at a 45 degree angle to each axes

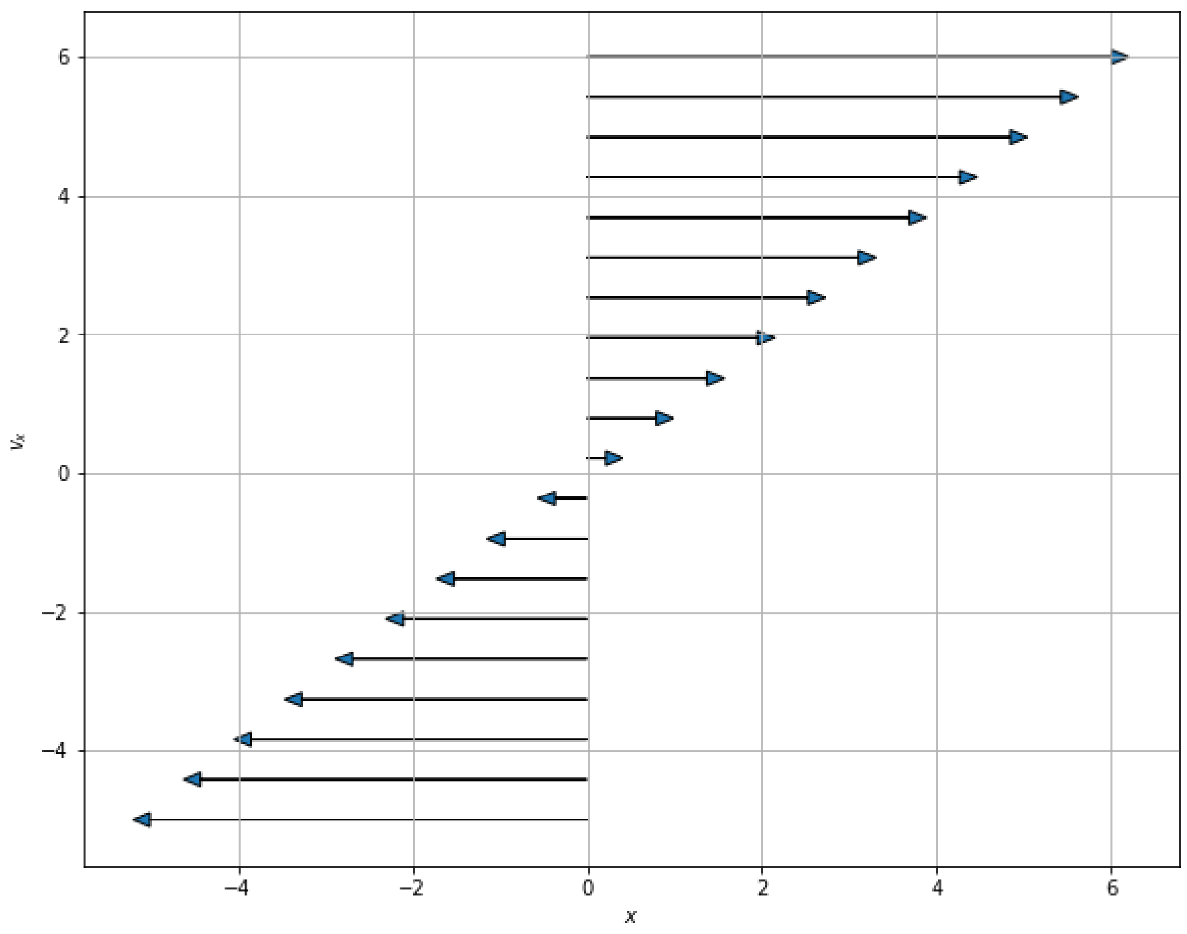
You should be able to explain what the code is doing. We avoided using meshgrid here to make this a smaller bit of code.

6.3.3 PlotPhaseSpaceAxesSHO

This function is computing the arrows for a given line of points in phase space. Send it a line of points in two arrays (one for $x$ and one for $v_x$) and it plots the resulting arrows. The code is documented below with comments and then used several times.

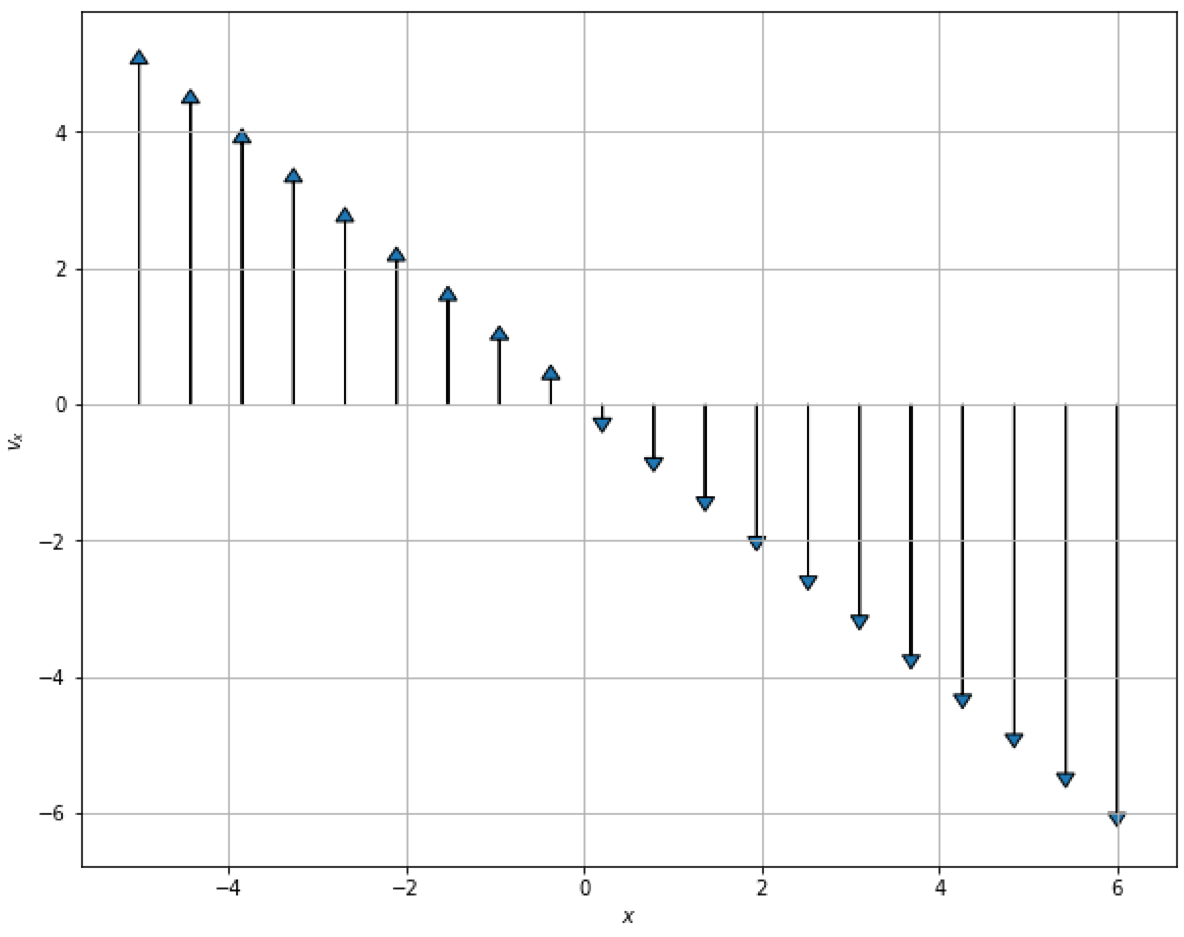
```python
def PlotPhaseSpaceAxesSHO(x, vx, N=20):
    """Takes two one-dimensional arrays
    and computes the resulting arrow to
    represent the flow of the system in
    phase space. This code is specifically
    designed for the SHO with omega=1"""

    ## Map the points to the arrows using the
    ## 1st order ODEs for the SHO
    ## Returns two arrays of the same length
    ## as the inputs
    xdot, vxdot = vx, -1*x

    ## Create a figure with a known size
    plt.figure(figsize=(10,8))

    ## Go through all the arrays we created to plot the arrows
    ## Syntax for arrow is:
    ## arrow(xpos, ypos, xchange, ychange, other_parameters)
    for i in np.arange(N):

        plt.arrow(x[i], vx[i], xdot[i], vxdot[i],
                  head_width=0.2,
                  head_length=0.2)
        plt.xlabel('$x$')
        plt.ylabel('$v_x$')

    plt.grid()
```

```python
## Plotting along the vx axis
N = 20

x = np.zeros(N)
vx = np.linspace(-5,6,N)

PlotPhaseSpaceAxesSHO(x, vx, N)
```



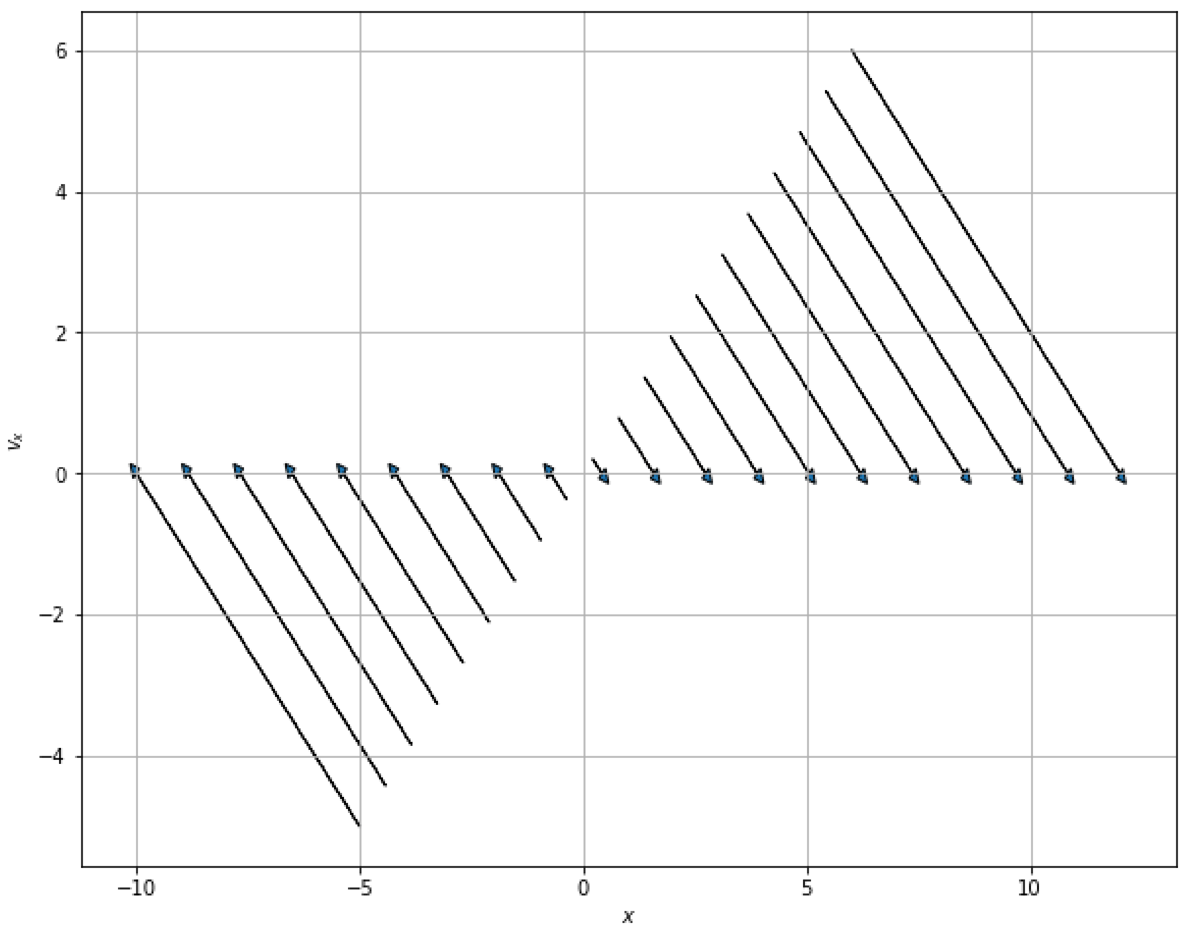
6.3.4 Plotting along the x axis

```
## Plotting along the x axis
N = 20

x = np.linspace(-5,6,N)
vx = np.zeros(N)

PlotPhaseSpaceAxesSHO(x, vx, N)
```



6.3.5 Plotting along the 45 degree line between the x and vx axes

```
## Plotting along the 45 degree line between the x and vx axes
N = 20

x = np.linspace(-5,6,N)
vx = np.linspace(-5,6,N)

PlotPhaseSpaceAxesSHO(x, vx, N)
```



6.3.6 Make a Graph

? Do this

Make a fourth plot that looks at the other diagonal line that runs at a 45 degree angle to each axes

6.4 Phase Portrait of the Simple Harmonic Oscillator

Below, we have written code that makes a phase portrait from the simple harmonic oscillator. It's written in terms of three functions that serve three purposes that you might want to modify in your own work:

- `SHOPhasePortrait` is a function that simply returns the relationship between the locations in phase space and how the phase variables change at that location.
- `ComputeSHOPhase` is a function that uses that relationship and computes the values of the changes at every location. It returns two arrays, which contain all those values.
- `SHOTrajectory` is a function that takes a pair of points in space and computes the trajectory in phase space

By separating these ideas, we are illustrating the process for computing these phase portraits:

- Translate one Nth order differential equation to N 1st order (Done earlier in this case)
- Put that into a code so you can compute the value of the changes at a location (`SHOPhasePotrait`)

- Call that computation a bunch to compute it at every location you want (`ComputeSHOPhase`)
- investigate specific trajectories in the space (`SHOTrajectory`)

We can then call these functions can plots the results.

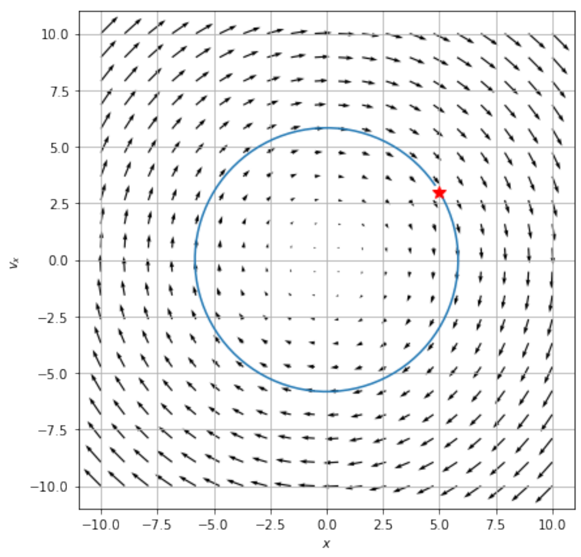
```
def SHOPhasePortrait(x, vx, omega):
    '''SHOPhasePortrait returns the value of
    the change in the phase variables at a given location
    in phase space for the SHO model'''

    xdot, vxdot = [vx, -1*omega**2*x] ## Specific to this problem

    return xdot, vxdot

def ComputeSHOPhase(X, VX, omega):
    '''ComputeSHOPhase returns the changes in
    the phase variables across a grid of locations
    that are specified'''

    ## Prep the arrays with zeros at the right size
    xdot, vxdot = np.zeros(X.shape), np.zeros(VX.shape)

    ## Set the limits of the loop based on how
    ## many points in the arrays we have
    Xlim, Ylim = X.shape

    ## Calculate the changes at each location and add them to the arrays
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = VX[i, j]
            xdot[i,j], vxdot[i,j] = SHOPhasePortrait(xloc, yloc, omega)

    return xdot, vxdot

def SHOTrajectory(x0, vx0, omega, N=100):
    '''SHOTrajectory computes the phase space
    trjectory using the analytical forms of the
    solution. Note this sloppy analytical approach
    only works because the SHO is perfectly sinusoidal.'''

    ## Only work with one period
    T = 2*np.pi/omega
    t = np.arange(0,T,T/N)

    ## I derived this in general with Acos(wt+phi)
    ## It's not in general a good approach
    ## because you are not guaranteed analytical
    ## closed form trajectories in phase space

    phi = np.arctan2(-1*vx0, omega*x0) ## arctan(-vxo/(omega*x0)) taken correctly for␣
↪the quadrant
    A = x0/np.cos(phi)
    x_traj = A*np.cos(omega*t+phi)
    v_traj = -omega*A*np.sin(omega*t+phi)

    return x_traj, v_traj
```

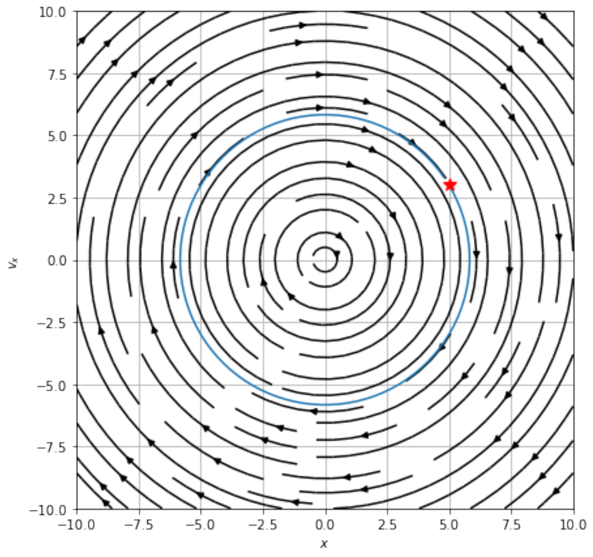
6.4.1 Putting the functions to use

With these two functions, all we are left to do is specify the size of the space and the grid points (that is where exactly we are computing the changes). We use meshgrid to make those arrays a set of Cartesian coordinates and then send that to our functions.

We then plot the results.

```
## Setting parameters and the phase space variables

omega = 1
x = np.linspace(-10.0, 10.0, 20)
vx = np.linspace(-10.0, 10.0, 20)

## Get back pairs of coordinates for every point in the space
X, VX = np.meshgrid(x, vx)

## Run our calculations
xdot, vxdot = ComputeSHOPhase(X, VX, omega)

x0 = 5
vx0 = 3
x_traj, v_traj = SHOTrajectory(x0, vx0, omega)

## Plot. plot. plot.
ax = plt.figure(figsize=(15,7))
plt.subplot(1,2,1)

## Plot with Quiver
Q = plt.quiver(X, VX, xdot, vxdot, color='k')

## Plot trajectory and the starting location
plt.plot(x_traj,v_traj)
plt.plot(x0, vx0, 'r*', markersize=10)

plt.xlabel('$x$')
plt.ylabel('$v_x$')
plt.grid()

plt.subplot(1,2,2)
## Plot with streamplot for subplot
Q = plt.streamplot(X, VX, xdot, vxdot, color='k')
plt.plot(x_traj,v_traj)
plt.plot(x0, vx0, 'r*', markersize=10)

plt.xlabel('$x$')
plt.ylabel('$v_x$')
plt.grid()
```





6.4.2 What can phase space help us do?

□ Do this

Let's remember a few things about the SHO.

1. With your neighbors, list all the things you know about the SHO. Include anything we haven't discussed (e.g., the energetics of the problem).
2. Name which of those things you can see in the phase diagram. Which things are you sure you can see? What things don't seem to be able to be seen from the phase diagram?
3. What do you remember about the energy of an SHO? Consider a harmonic oscillator in a known trajectory ($x(t) = A\cos(\omega t)$). Compute the total (conserved) energy of the oscillator as a function of time.
   - Explain how your expression for energy conservation can be seen in your phase diagram.
   - You might try to show analytically that the ellipse above is related to your energy conservation expression
4. What are the fixed points for this system? Can it be classified as stable/unstable or do we need to think of something new?

What do these plots tell you about all potential solutions?

6.5 The Large Angle Pendulum

The Large Angle Pendulum is the first of a number of nonlinear differential equations out there. This one is quite special in that the integral that solves for the period of this pendulum has a name! It's called and Elliptical Integral of the First Kind. Elliptical because of the nature of the kernel of the integral, which has an elliptic form (in our case, one over the square root of a quantity squared subtracted from one, yes, seriously, we have a name for that).

Here's the pendulum in all it's glory.

The analytical solution for the period is given by:

$$T = 4\sqrt{\frac{L}{g}}\int_0^{\pi/2}\frac{d\theta}{\sqrt{1-k^2\sin^2(\theta)}}$$

To find the period, we have to use some form of computation, even it's the "well-known" recurrence relationship that was used for centuries to compute this integral **by hand**.

But let's try to gain insight from the phase space instead. We can [derive] the differential equation that describes the motion of the pendulum through an angle θ thusly:

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta)$$

You have a second order differential equation for θ.

6.5.1 Make a new phase portrait

? Do this

With your partners,

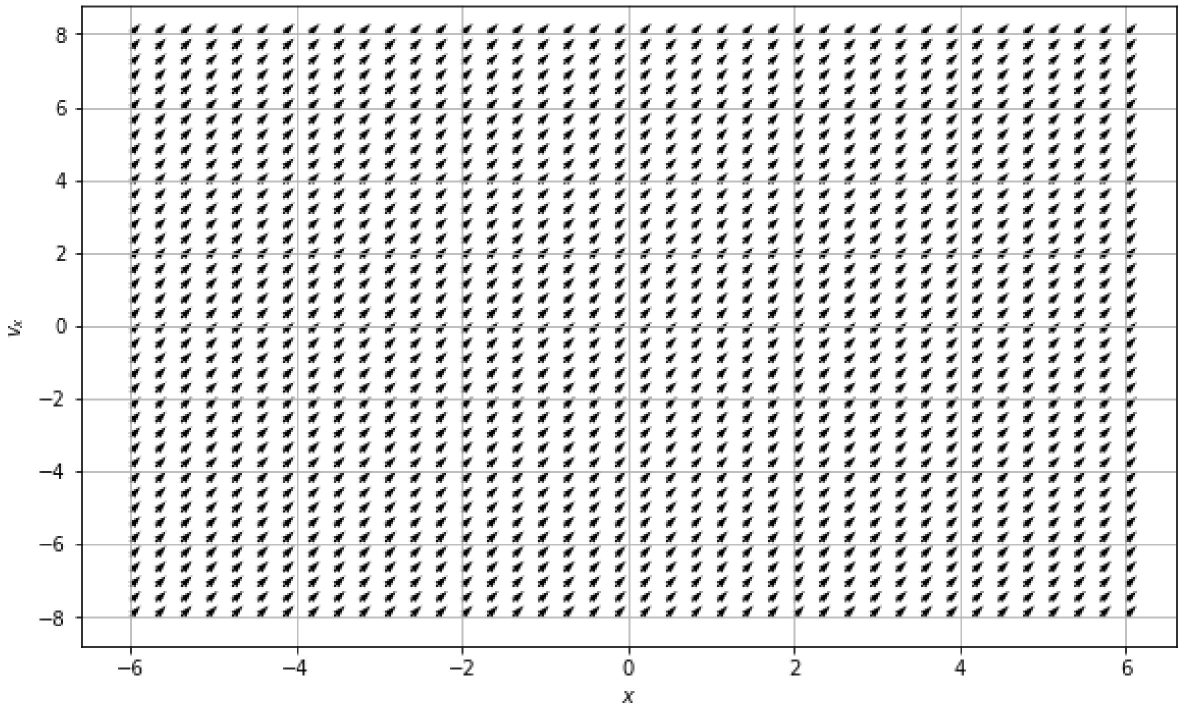
1. Take the 2nd order ODE and make it two 1st order ODEs (one for θ and one for $\omega = \dot{\theta}$). Make sure you agree on the analytics.
2. Add those expressions to the function `LAPPhasePortrait`
3. The rest of the code runs the same as before (we've engaged in reproducible and adaptable work!), so make some phase portraits.
   - What do you notice?
   - What physics is new?
   - What physics is old?
4. Play with parameters and build a story for what is going on with the motion.

```
def LAPPhasePortrait(x, vx, omega0 = 10):

    #################
    ## CHANGE THIS ##
    #################
    xdot, vxdot = [1, 1] ## Specific to the problem

    return xdot, vxdot

def ComputeLAPPhase(X, VX, omega0):

    xdot, vxdot = np.zeros(X.shape), np.zeros(VX.shape)

    Xlim, Ylim = X.shape

    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = VX[i, j]
            xdot[i,j], vxdot[i,j] = LAPPhasePortrait(xloc, yloc, omega0)

    return xdot, vxdot

omega0 = 2
N = 40

x = np.linspace(-6.0, 6.0, N)
```

(continues on next page)

(continued from previous page)

```
vx = np.linspace(-8.0, 8.0, N)

X, VX = np.meshgrid(x, vx)

xdot, vxdot = ComputeLAPPhase(X, VX, omega0)

ax = plt.figure(figsize=(10,6))
Q = plt.quiver(X, VX, xdot, vxdot, color='k')
plt.grid()

plt.xlabel('$x$')
plt.ylabel('$v_x$')
plt.show()
```



? Do this

Find the fixed points of this system. Are they the same as in the SHO case? If they're different, what about this system makes it different?

6.6 (Time Permitting) Fixed Points in 2 Dimensions

If we don't get to this in class today don't worry, we'll start out with this on thursday!

Now that we've stepped into two dimensional phase space, the amount of interesting geometry that our solutions can have is much greater. In particular, local behavior near fixed points can now exhibit much more complex behavior than just being stable or not, as we can see in this figure.

So how do we definitively say what the behavior of our system is like near a fixed point? The anwer lies in **linearization**.

A linear system in 2d is a system of the form:

$$\dot{x} = ax + by \qquad\qquad \dot{y} = cx + dy$$

Note we can convieniently write this in matrix notation:

$$\dot{\mathbf{x}} = A\mathbf{x}$$

Where

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \text{and} \qquad \mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

These systems are understood quite well and play very nice mathematically (see strogatz chapter 5). Miraculously, the mathematical tools for classifying fixed points of linear systems carry over into nonlinear systems with little tweaking. This is because most nonlinear systems behave in linear ways near fixed points. This means we can **linearize** nonlinear systems. Here's how you do it:

Suppose you have a system given by:

$$\dot{x} = f(x, y)$$

$$\dot{y} = g(x, y)$$

With fixed point (x^*, y^*). Near this fixed point, distrubances to the system will evolve approximatley according to:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} \begin{bmatrix} x - x^* \\ y - y^* \end{bmatrix}$$

This is called the **linearized system**. The matrix

$$A = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix}_{(x^*, y^*)}$$

is the **Jacobian matrix** at this fixed point, which is the multivariable version of the derivative. To categorize the fixed points of a given system at a fixed point, calculate A for said fixed point, then find its Eigenvalues. Recall Eigenvalues are the λ in $A\mathbf{v} = \lambda\mathbf{v}$. 2×2 matricies have (up to) 2 eigenvalues. These eigenvalues tell you about the stability of the system:

- Re$(\lambda) > 0$ for both eigenvalues: Repeller/Source (unstable)
- Re$(\lambda) < 0$ for both eigenvalues: Attractor/Sink (stable)
- One eigenvalue positive, one negative: Saddle
- Both eigenvalues pure imaginary: Center

In fact one can learn quite a bit more from a these eigenvalues (see strogatz chapter 6 or section 5.4 here), but these charactarizations are a great starting point.

? Do this

Calculate the Jacobian matrix A for a fixed point of the large angle pendulum.

CHAPTER

SEVEN

14 SEP 23 - DYNAMICAL SYSTEMS CONTINUED

Last time we investigated the phase portrait of the large angle pendulum, we we could arrive at by re-writing the differential equation

$$\ddot{\theta} = -\frac{g}{L}\sin(\theta)$$

as 2 first-order differential equations:

$$\dot{\theta} = \omega \qquad \text{and} \qquad \dot{\omega} = -\frac{g}{L}\sin(\theta)$$

By setting both of these equations equal to zero simultaneously, we also argued that this system has (countably) infinite fixed points at $(n\pi, 0)$ for $n \in \mathbb{Z}$ in (θ, ω) phase space.

Now we turn to the challenge of characterizing these fixed points with the linearization of the system (see the end of tuesday's activiy for some more notes on this). Recall that we can do this by finding the eigenvalues of the Jacobian Matrix of the system at its fixed point. For the system $\dot{x} = f(x, y)$, $\dot{y} = g(x, y)$ the jacobian matrix looks like this:

$$A = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix}_{(x^*, y^*)}$$

Do this (this is the same problem as the last problem from tuesday)

Calculate the general Jacobian matrix A for this system, then calculate what it is at the fixed point $(0, 0)$.

We have the Jacobian at $(0, 0)$ now but we still need to find its eigenvalues. Let's take a quick detour to remember how to do that.

7.1 Eigenvalues

Eigenvalues and the closely related Eigenvectors are indispensible in physics, math, and computational science. These ideas for the basis (pun somewhat intened) for countless problems, from the energy eigenvalue equation that is the founcation of quantum mechanics, to the stability of complex nonlinear systems, to Normal Modes of oscillators, which we'll study later in this course, eigenproblems show up all over in physics. I can't resist a brief tangent: Once some scientists were using an eigenvalue driven algorithm called principal component analysis to study the genes of people that live in Europe. They found that these egenvalues/vectors reproduced a map of Europe with surprising accuracy (link). So these tools are extremely, and often unreasonably powerful.

Eigenvalues are the λ in the equation:

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

Where A is a linear operator of the vector space that $\mathbf{v}$ lives in. In finite-dimensional vector spaces, like what we're considering today, these linear operators are always matricies. There is a bit of physical intuition behind this equation: An eigenvector of A is a vector that only gets stretched or squished by λ when A acts on $\mathbf{v}$. Here's a gif from Grant Sanderson's fantastic video on eigenvalues and eigenvectors that shows this:

7.1.1 Finding Eigenvalues

To actually find the eigenvalues of a matrix, you solve the **characteristic polynomial** of the matrix, which you obtain by solving the equation:

$$|A - \lambda I| = 0$$

Where the vertical bars means determinant.

To find Eigenvectors, simply plug in the values you found for λ into the original eigenvalue equation $A\mathbf{v} = \lambda\mathbf{v}$, using $\mathbf{v} = \begin{bmatrix} x \\ y \end{bmatrix}$. You'll find some simple relationship between x and y. Any scalar multiple of an eigenvector is also an eigenvector so we usually just choose the simplest one. Say if you found that $x = -y$. Then for a nice clean looking eigenvector you could choose $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$.

? Do this

Analytically, find the eigenvalues of the Jacobian matrix you calculated earlier. Use the below bullets to identify these eigenvalues with the type of the fixed point.

- $\text{Re}(\lambda) > 0$ for both eigenvalues: Repeller/Source (unstable)
- $\text{Re}(\lambda) < 0$ for both eigenvalues: Attractor/Sink (stable)
- One eigenvalue positive, one negative: Saddle
- Both eigenvalues pure imaginary: Center

Note: You can actually learn quite a bit more from this analysis, see Strogatz chaper 6.

7.1.2 Eigenvalues, Computationally

We can use `np.linalg.eig()` to find the eigenvalues (and normalized eigenvectors) of a matrix which we represent as numpy array. Below is some doe that does this (note the imaginary unit is represented as j in python):

```python
import numpy as np
A = np.array([[0,1],[-1,0]])
eigvals = np.linalg.eig(A)[0]
eigvecs = np.linalg.eig(A)[1]

print("eigenvalues:", eigvals)
```

```
eigenvalues: [0.+1.j 0.-1.j]
```

This can be super handy when you just need to do some quick caracterization from the eigenvalues of a matrix. However, be warned - since you only get numerical answers you can lose quite a bit of the nuance that comes from if you had calculated these. We'll see how that can be an issue later in the semester when we tackle normal modes.

7.2 Activity: Lotka - Volterra Equations

The Lotka -Volterra Equations are a pair of coupled ODEs

$$\dot{x} = x(A - Bx - Cy)$$

$$\dot{y} = y(D - Fx - Gy)$$

with $A, B, C, D, F, G > 0$

That model the time evolution of the competition between two species, say rabbits and sheep. We'll say x is the number of rabbits while y is the number of sheep. This model reduces to the logistic growth model if we were to ignore the competition, say if $\dot{x} = x(A - Bx)$

✅ Do this

1. What do each of the parameters A, B, C, D, F, G represent? Why do you say so?
2. Identify the fixed points of this system (there might be more than 2!)
3. Find the Jacobian for these equations
4. Modify the starter code below to so it gives you the eigenvalues of the jacobian for a given A, B, C, D, x^*, y^*.
5. For the set of values of A, B, C, D, F, G given in the code below, sketch what you expect the phase portrait of this system to look like. Then run the the code 2 cells below to see how well you did.
6. Experiment with choosing different values for A,B,C,D,F,G. Does the behavior of the system change for with different choices? (the initial values given below should be a good starting point).

```python
def jacobian(A,B,C,D,x,y):
    return np.array([[0,0],[0,0]]) # CHANGE

A,B,C,D,F,G = 3,1,2,2,1,1
x1,y1 = 0,0 # 1st fixed point
x2,y2 = 0,0 # 2nd fixed point CHANGE
# more fixed points here...

print("eigenvalues, 1st fixed point:",np.linalg.eig(jacobian(A,B,C,D,x1,y1))[0])
print("eigenvalues, 2nd fixed point:",np.linalg.eig(jacobian(A,B,C,D,x2,y2))[0])
```

```
eigenvalues, 1st fixed point: [0. 0.]
eigenvalues, 2nd fixed point: [0. 0.]
```

```python
import matplotlib.pyplot as plt
def LV_eqns(x, y):
    xdot, ydot = [x*(A - B*x - C*y), y*(D - F*x - G*y)]
    return xdot, ydot

def LV_phase(X, VX):
    xdot, ydot = np.zeros(X.shape), np.zeros(VX.shape)
    Xlim, Ylim = X.shape
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = VX[i, j]
            xdot[i,j], ydot[i,j] = LV_eqns(xloc, yloc)
    return xdot, ydot
```

(continues on next page)

(continued from previous page)

```
N = 40
x = np.linspace(0., 3.5, N)
y = np.linspace(0., 3.5, N)
X, Y = np.meshgrid(x, y)
xdot, ydot = LV_phase(X, Y)
ax = plt.figure(figsize=(10,10))
Q = plt.streamplot(X, Y, xdot, ydot, color='k',broken_streamlines = False)
plt.scatter(x1,y1, label = 'fixed point 1')
plt.scatter(x2,y2, label = "fixed point 2")
plt.legend()
plt.grid()
plt.xlabel('$rabbits$')
plt.ylabel('$sheep$')
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [3], in <cell line: 22>()
     20 xdot, ydot = LV_phase(X, Y)
     21 ax = plt.figure(figsize=(10,10))
---> 22 Q = plt.streamplot(X, Y, xdot, ydot, color='k',broken_streamlines = False)
     23 plt.scatter(x1,y1, label = 'fixed point 1')
     24 plt.scatter(x2,y2, label = "fixed point 2")

TypeError: streamplot() got an unexpected keyword argument 'broken_streamlines'
```

```
<Figure size 720x720 with 0 Axes>
```

7.3 Investigating the Van der Pol Oscillator

It turns out there is some more interesting behavior other than just the behavior around fixed points. Toward seeing that, let's look at the Van der Pol Oscillator. This equation originates from lonlinear circuits in early radios, but has now also been used in neuroscience and geology. It is given by the differential equation:

$$\ddot{x} = -\mu(x^2 - 1)\dot{x} - x$$

or, written as two first order equations:

$$\dot{x} = v \qquad\qquad \dot{v} = -\mu(x^2 - 1)v - x$$

With $\mu > 0$. Note that this equation is simply the harmonic oscillator when $\mu = 0$. The strange $-\mu(x^2 - 1)\dot{x}$ represents damping, but this damping behaves strangely, because when $|x| < 1$ it is negative damping, that is it boosts oscillations smaller than 1, while still slowing down oscillations larger than 1.

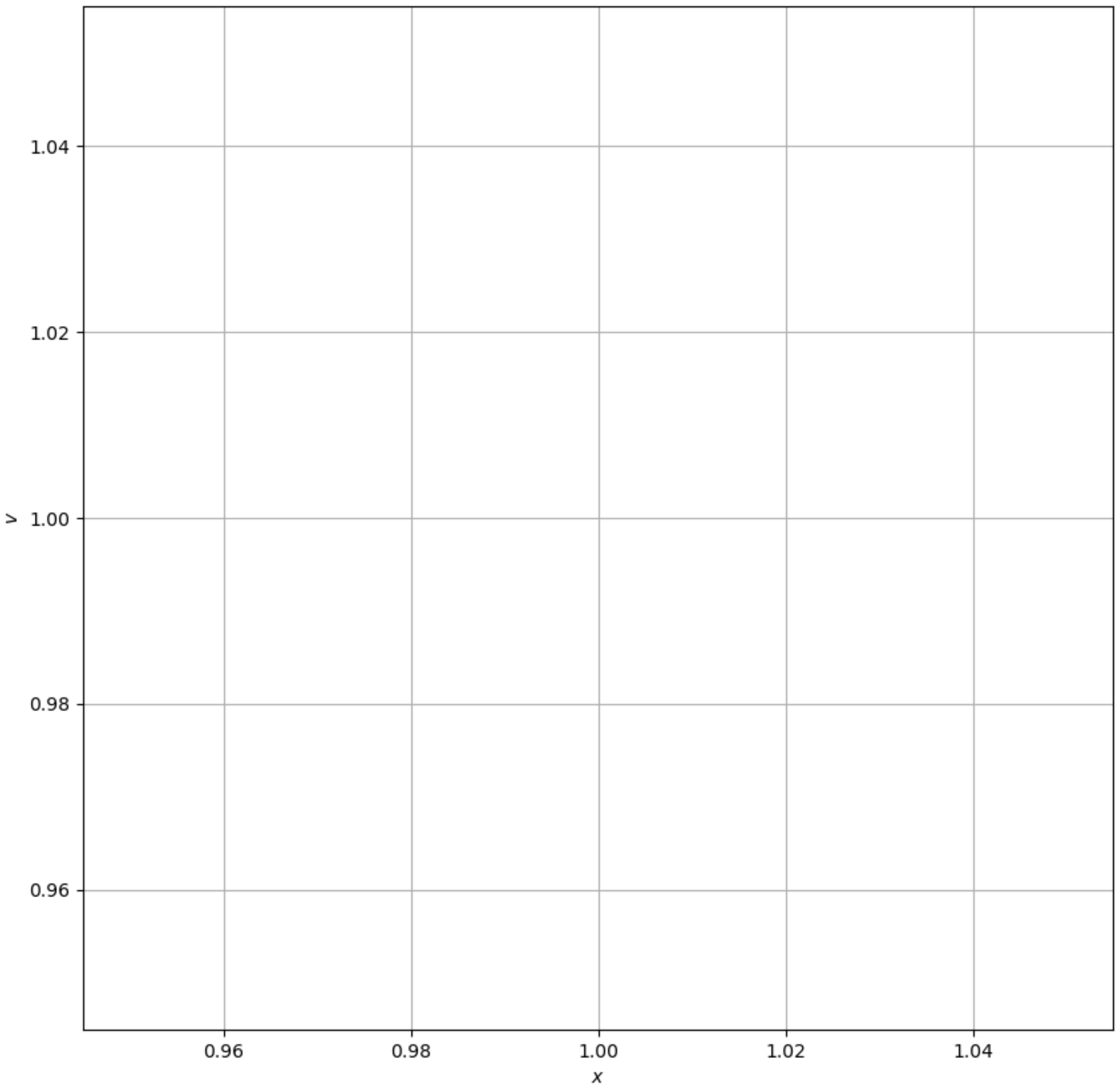
Now we play the usual game of trying to figure out how this system behaves:

✎ Do this

1. Identify the fixed point of this system. Follow the linearization procedure to characterize it.
2. Edit the code below to produce a phase plot for the Van der Pol oscillator. This code also numerically integrates a trajectory and plots it. Add a second trajectory and plot that as well.

3. What happens to phase space when you change the value of μ? What if you make it negative?
4. What behavior do you notice here that's different than you've seen before? What is attracting the trajectories?

```python
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

def VP_eqn(x, v, mu = 1.):
    xdot, vdot = [0,0] ## CHANGE
    return xdot, vdot

def VP_phase(X, VX, mu):
    xdot, vdot = np.zeros(X.shape), np.zeros(VX.shape)
    Xlim, Ylim = X.shape
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = VX[i, j]
            xdot[i,j], vdot[i,j] = VP_eqn(xloc, yloc,mu)
    return xdot, vdot

def VP_eqn_for_solve_ivp(t,curr_vals, mu=1): # need to rephrase this to work with␣
↪what solve_ivp expects
    x, v = curr_vals
    xdot, vdot = VP_eqn(x,v,mu)
    return xdot,vdot

# Numerical Integration
tmax = 20
dt = 0.05
tspan = (0,tmax)
t = np.arange(0,tmax,dt)
mu = 1.
initial_condition = [1, 1]
solved = solve_ivp(VP_eqn_for_solve_ivp,tspan,initial_condition,t_eval = t, args =␣
↪(mu,),method="RK45")


# Plotting stuff
N = 40
x = np.linspace(-3., 3., N)
v = np.linspace(-3., 3., N)
X, V = np.meshgrid(x, v)
xdot, vdot = VP_phase(X, V,mu)
ax = plt.figure(figsize=(10,10))
Q = plt.streamplot(X, V, xdot, vdot, color='k',broken_streamlines = False)
plt.plot(solved.y[0],solved.y[1],lw = 3,c = 'red') # plot trajectory from solve_ivp
plt.grid()
plt.xlabel('$x$')
plt.ylabel('$v$')
plt.show()
```



✎ Do this

Based on the phase space diagram, what do you expect actual trajectories to look like in x vs t space? Use the numerically integrated trajectories to plot that.

```
## your code here
```

7.3.1 Limit Cycles

The new behavior we've seen from this equation is what's called a **limit cycle**, where the system is attracted/reppeled from a closed curve instead of a fixed point(s). There's a lot of really great math here that's a bit beyond what we can cover in class, but it would be a great thing to look into for a project!

✅ Do this

Spend the rest of class investigating the Van der Pol oscillator. Here are a few investigations you could do:

- When μ changes from negative to positive, this system undergoes what is known as a **Hopf Bifurcation** Look up what bifurcations are to understand what this means and show that it is true using numerical integration.
- Add an $A\sin(t)$ driving force term to the differential equation and numerically integrate. What do these trajectories look like in x vs t and in phase space?
- Examine the energetics of this system. Is energy conserved or does it have some interesting behavior? Why?

```
# code here
```

CHAPTER

EIGHT

19 SEP 23 - CHAOS

Over the last couple weeks we've been looking into a lot of 2 dimensional autonomous (the differential equations don't depend on time) systems of differential equations. Another thing we've noticed is that initial conditions in these systems tend to either blast off to infinity, form a closed loop, or are drawn to attracting fixed point(s) or limit cycle(s). In the case of attracting fixed points or limit cycles, many different initial conditions can be drawn to the same fixed point or limit cycle. In fact, we saw that it is possible for **every** initial condition to be drawn to the same limit cycle for the Van der Pol oscillator. This means that trajectories that are initially far apart eventually become close together.

This seems all well and good, but when we step up to 3 dimensional autonomous systems (or as we'll see later, non-autonomous 2D ones), some more interesting behavior starts to emerge. Trajectories that start out very close to each other can diverge from each other, while still not diverging to infinity. And it turns out that these sorts of **chaotic** systems arise surprisingly often in systems we're interested in in physics. The classic example of this is the double pendulum. Its also been shown that the solar system is chaotic for large time scales, and countless other systems exhibit this property.

8.1 Definition of Chaos

Here we'll follow Strogats's definition of chaos, which is:

***Chaos** is aperiodic long-term behavior in a deterministic system that exhibits sensitive dependence on initial conditions.*

1. **Aperiodic long-term behavior** means that some trajectories don't settle into fixed points, periodic orbits, or quasi-periodic orbits, while still not diverging to infinity.
2. **Deterministic** means that the equations evolve in totally predictable ways, without any randomness. There are no noisy parameters or imputs.
3. **Sensitive dependence on initial conditions** means that nearby trajectories separate exponentially fast.

Let's investigate this phenomenon for a particular system.

8.2 Halvorsen Attractor

One system we can start looking at is given by the **Halvorsen Equations:**

$$\dot{x} = -ax - 4y - 4z - y^2$$

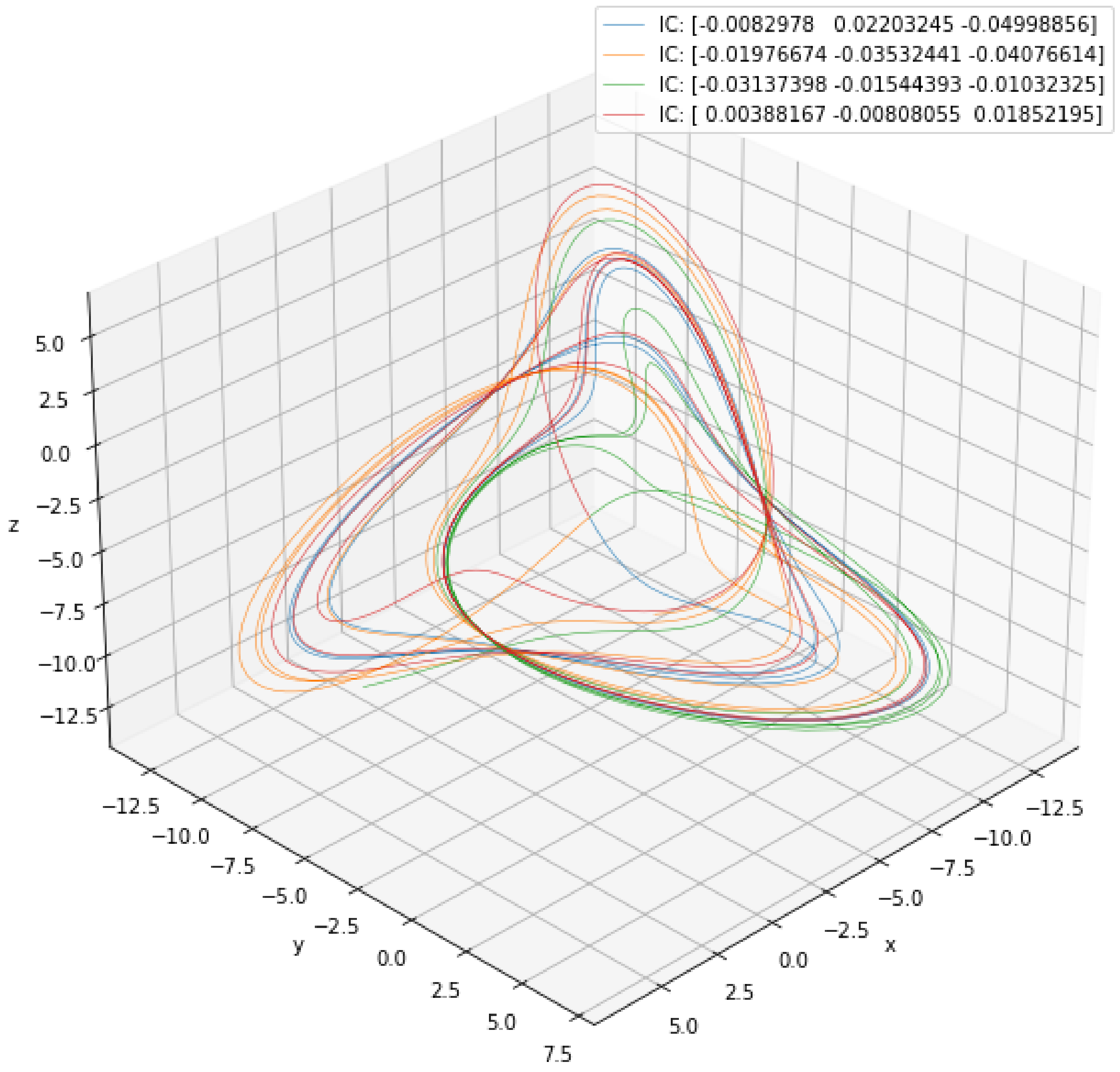
$$\dot{y} = -ay - 4z - 4x - z^2$$

$$\dot{z} = -az - 4x - 4y - x^2$$

✎ Do this

1. Review the code below. Talk to your neighbors about what it is doing. Feel free to experiment with more or less initial conditions. **You should be able to explain what this code is doing**
2. Based on our definition above, is this system chaotic? Why or why not? Discuss with your neighbors.
   - If it is chaotic and there is no limit cycle, what is drawing the trajectories?

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

def Halvorsen(t,curr_vals, a):
    # Derivatives function for solve_ivp
    x,y,z = curr_vals
    xdot = -a*x - 4*y - 4*z - y**2
    ydot = -a*y - 4*z - 4*x - z**2
    zdot = -a*z - 4*x - 4*y - x**2
    return xdot,ydot,zdot

# Time Setup
tmax = 10
dt = 0.01
tspan = (0,tmax)
t = np.arange(0,tmax,dt)

# Parameters and initial conditions
a = 1.4
n_ics = 4 # number of initial conditions
n_dim = 3 # 3 dimensional problem
np.random.seed(1) # control randomness
initial_conditions = np.random.uniform(-0.05,0.05,(n_ics,n_dim)) # get n_ics initial␣
↪conditions randomly from small box by the origin

# Call integrator for each initial condition
solutions = []
for initial_condition in initial_conditions:
    solved = solve_ivp(Halvorsen,tspan,initial_condition,t_eval = t, args = (a,))
    solutions.append(solved.y)

# Plotting
#%matplotlib widget ## UNCOMMENT TO BE ABLE TO PAN AROUND
fig = plt.figure(figsize = (10,10))
ax = plt.axes(projection='3d')
ax.view_init(30, 45) # Pick a nice initial viewing angle
for i,initial_condition in enumerate(initial_conditions):
    x,y,z = solutions[i]
    ax.plot3D(x,y,z, label = "IC: " +str(initial_condition),lw = 0.5)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.legend()
plt.show()
```



8.3 Lyapunov Exponents

One of our criteria for a system to be chaotic was that nearby trajectories must separate exponentially fast. Let's see if this holds true for this system:

✎ Do this

1. Numerically calculate and plot the Euclidean distance ($||\delta(t)||$) between the first two trajectories as a function of time. What do you notice?
2. Also calculate $\log ||\delta(t)||$ and plot it. What do you notice now?

```
## your code here
```

The way that we characterize this divergence mathematically is with what are known as **Lyapunov Exponents**. Suppose we have some trajectory of a dynamical system $\mathbf{x}_1(t)$. Then we could write a nearby trajectory, lets say $\mathbf{x}_2(t)$ as $\mathbf{x}_2(t) =$

$\mathbf{x}_1(t) + \delta(t)$ where $\delta(t) = \mathbf{x}_2(t) - \mathbf{x}_1(t)$. If trajectories initially separated by δ_0 separate exponentially fast, then we would expect to see:

$$||\delta(t)|| \sim ||\delta_0|| e^{\lambda t}$$

Where we call λ the **Lyapunov Exponent** (technically it is the largest one of multiple). This also lets us write a very useful equation for how long a prediction of a chaotic system is within tolerance a:

$$t_{\text{horizon}} \sim O\left(\frac{1}{\lambda} \log \frac{a}{||\delta_0||}\right)$$

? Do this

1. Use $\log ||\delta(t)||$ that you calculated above and `np.polyfit` to estimate the value of λ for the Halvorsen system.
   - You'll need to eyeball where $\log ||\delta(t)||$ stops being linear.
2. Repeat this calculation for another set of 2 trajectories. Do you get a similar value for λ?
3. Many systems have negative values for λ. What does a negative value for λ mean?

```
## your code here
```

8.4 Driven Damped Pendulum

Now let's turn our attention to a more physics-y chaotic system, the Driven Damped Pendulum (DDP). This system is similar to the large angle pendulum that we've studied before, but it has a damping term $-2\beta\dot{\theta}$, as well as a driving time-dependent torque $\gamma\omega_0^2 \cos(\omega t)$. The full equation after sufficient non-dimensionalization looks like this (note: $\omega_0 \neq \omega$):

$$\ddot{\theta} = -2\beta\dot{\theta} - \omega_0^2 \sin(\theta) + \gamma\omega_0^2 \cos(\omega t)$$

It may be tempting to try to construct a phase portrait of this thing since it looks so similar to the large-angle pendulum, but since this equation is now time dependent, the phase portrait itself is time dependent, which makes it tricky to visualize without a third dimension or animations (making that could be part of a project though!).

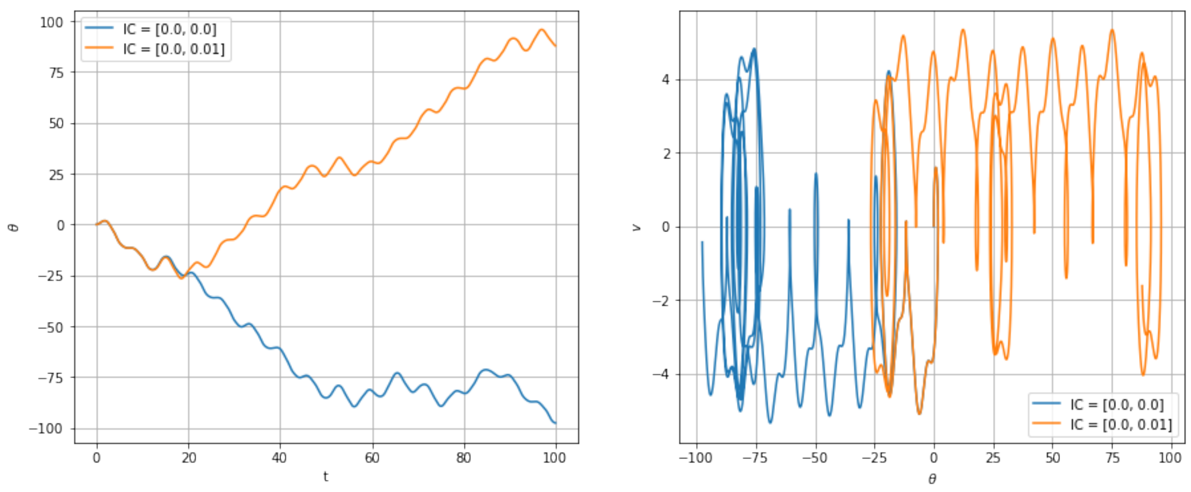
The code below numerically integrates and plots the two trajectories of similar initial condition of the DPP for a set of parameters that are known to be chaotic.

```
def DDP(t,curr_vals, beta,omega_natural,gamma,omega_drive):
    # Derivatives function for solve_ivp
    theta,v = curr_vals
    thetadot = v
    vdot = -2*beta*v - omega_natural * np.sin(theta) + gamma*omega_natural*np.
 ↪cos(omega_drive*t)
    return thetadot,vdot

# Parameters and initial conditions (chosen so that they give chaos)
beta = 0.375/2
omega_natural = 1.5**2
omega_drive = 1
gamma = 1.5
initial_condition = [0.,0.]
initial_condition2 = [0.,0.01]

# Time Setup
tmax = 100.
```

(continues on next page)

(continued from previous page)

```
dt = 0.01
t = np.arange(0,tmax, dt)
tspan = (t[0],t[-1])

# Call integrator for each initial condition
solved = solve_ivp(DDP,tspan,initial_condition,t_eval = t, args = (beta,omega_natural,
↪gamma,omega_drive))
solved2 = solve_ivp(DDP,tspan,initial_condition2,t_eval = t, args = (beta,omega_
↪natural,gamma,omega_drive))

# Plotting
fig = plt.figure(figsize = (15,6))
plt.subplot(1,2,1)
plt.plot(t,solved.y[0],label =  "IC = " + str(initial_condition))
plt.plot(t,solved2.y[0], label = "IC = " + str(initial_condition2) )
plt.xlabel("t")
plt.ylabel(r"$\theta$")
plt.legend()
plt.grid()
plt.subplot(1,2,2)
plt.plot(solved.y[0],solved.y[1], label =  "IC = " + str(initial_condition))
plt.plot(solved2.y[0],solved2.y[1], label = "IC = " + str(initial_condition2))
plt.xlabel(r"$\theta$")
plt.ylabel(r"$v$")
plt.legend()
plt.grid()
plt.show()
```



? Do this

1. Discuss the above plots with your neighbors.
   - Do you notice any patterns or is the motion totally unpredictable?
   - Try increasing the integration time. Does that reveal any structure?
   - Can you think of looking at this system another way that would reveal more?

8.5 Poincaré Section

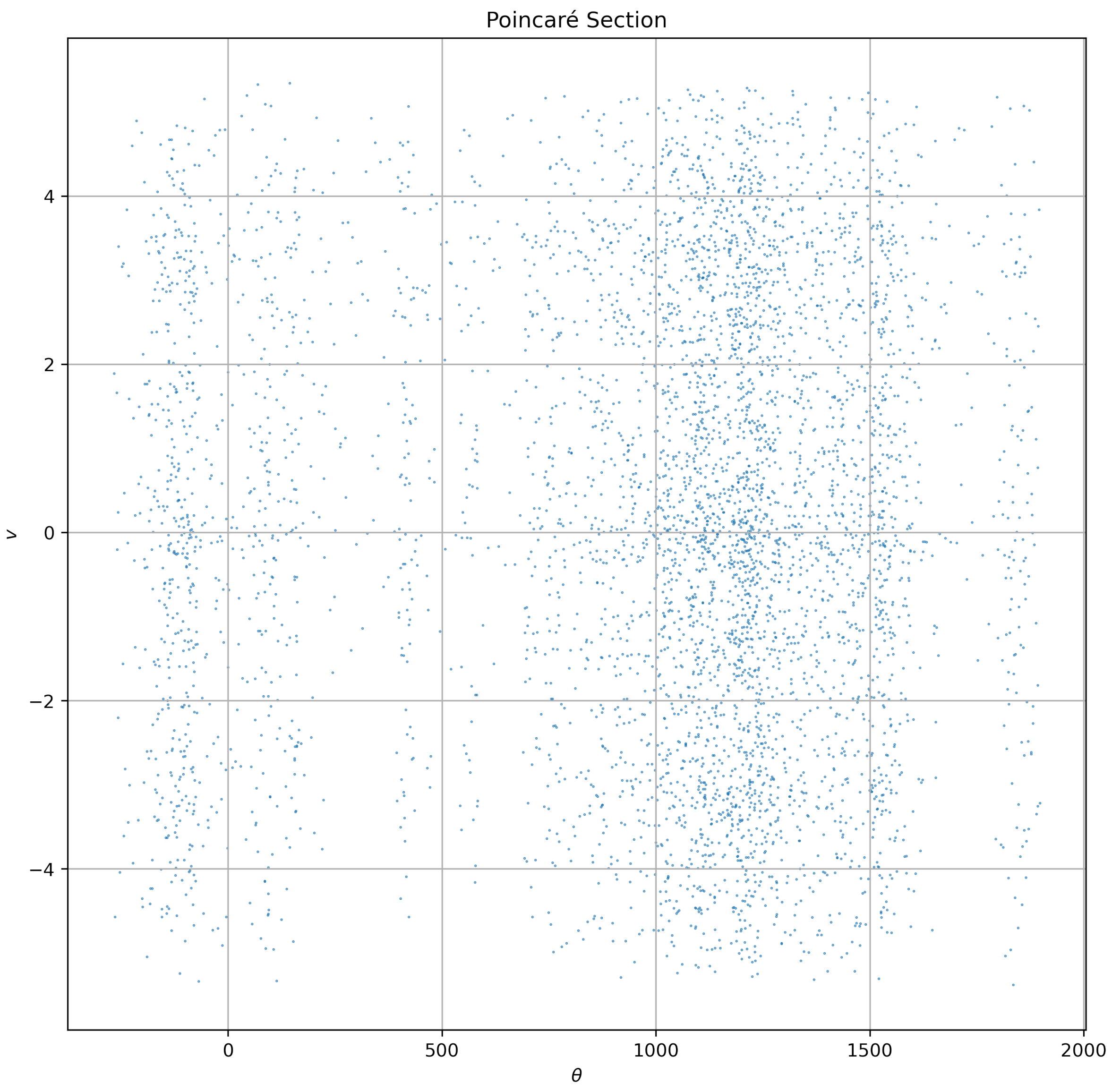
To get a better sense of what's actually going on here, and to maybe have a hope of actually seeing if this thing has an attractor, we can use what is called a **Poincaré section**. The idea behind a Poincaré section is as follows: since the force term of this system is periodic, we should only look at points in phase space where the force term is at the same point in its cycle. Doing this reveals the breautiful fractal cross section of an attractor.

✅ Do this

Modify the code below to create a Poincaré section of the DPP.

- Hint for line 5: How long is a drive period?
- Hint for line 17: The range of values that $\theta$ takes on is too large. What can you restrict them to?

```
# Time Setup
N = 5000 # Number of Drive Periods to integrate for

##########
t_period = 2. ## CHANGE
##########

t = np.linspace(0,N*t_period, N+1) # Note: this is not the same as the points used␣
↪for integration, so we need not worry about this ruining our numerical accuracy
tspan = (t[0],t[-1])

# Call integrator for each initial condition
solved = solve_ivp(DDP,tspan,initial_condition,t_eval = t, args = (beta,omega_natural,
↪gamma,omega_drive))

poincare_theta, v = solved.y[0],solved.y[1]

#######
poincare_theta = poincare_theta ## CHANGE
#######

# Plotting
fig = plt.figure(figsize = (10,10),dpi = 300)
plt.scatter(poincare_theta,v,s = 0.1)
plt.title("Poincaré Section")
plt.xlabel(r"$\theta$")
plt.ylabel(r"$v$")
plt.grid()
plt.show()
```



🔲 Do this

Spend any time you have left in class investigating the DDP. Some things to try:

- Try decreasing the strength of the driving force by decreasing γ.
  - Is the system always chaotic for any $\gamma > 0$?
  - How does the system's behavior change as you vary γ?
  - Here you might want to look up **transient chaos** and **fractal basin boundaries**. Strogatz chapter 12 explains this well (and has some pretty pictures!).
- What does the Poincare section look like when you have more predictable motion?
  - How many points does it have?
  - Can you vary ICs or parameters to change how many points it has without giving in to total chaos?
  - Here you might want to look up **period doubling** and **fiegenbaum number** Taylor classical mechanics chapter 12 has some great stuff on this.

CHAPTER

NINE

21 SEPT 23 - ACTIVITY: ODE GAMES

Y'all have worked with several different ODEs. We've learned that we can use phase space to investigate different potential families of solutions. We've learned how to read information from phase space for systems we are familiar with like the SHO and the large angle Pendulum. We've also learned how to use numerical integration to find trajectories of the system in time and in phase space, and how chaotic systems behave. Today, you will investigate a new model. There's quite a few well-known models.

Some of these models are listed below with links to articles describing them or the ideas related to them. *This is for reference, you don't need to read or understand anything deeply from these articles.*

A few 2nd order models:

1. Double Well Potential:

$$\dot{x} = y; \dot{y} = -x + y(1 - x^2)$$

1. Dipole Fixed Points:

$$\dot{x} = 2xy; \dot{y} = y^2 - x^2$$

1. Anharmonic Oscillator (Symmetric)

$$m\ddot{x} + b\dot{x} + k_1 x + k_2 x^3 = 0$$

1. Duffing Oscillator

$$\ddot{x} + x + \epsilon x^3 = 0$$

1. Glycolysis model

$$\dot{x} = -x + ay + x^2 y; \dot{y} = b - ay - x^2 y$$

1. Coupled synchronizing oscillators:

$$\dot{\theta}_1 = \omega_1 + K_1 \sin(\theta_2 - \theta_1); \dot{\theta}_2 = \omega_2 + K_2 \sin(\theta_1 - \theta_2)$$

1. Drop in stokes flow

$$\dot{x} = \frac{\sqrt{2}}{4} x(x-1)\sin(\phi); \dot{\phi} = \frac{1}{2}\left[\beta - \frac{1}{\sqrt{2}}\cos(\phi) - \frac{1}{8\sqrt{2}} x \cos(\phi)\right]$$

We want to let your group decide what you want to explore. But a little guidance when making those choices:

1. The 2nd order oscillators are relatively straightforward to implement in the code below if you break them into 1st order ODEs. Remember that you have to do that!
2. Several of these models have parameters, which you can choose, but maybe look into reasonable values. These models are likely the more challenging for this assignment because you have to keep track of and pass parameters.
3. Some of these models have a versions with time depdendent components , i.e., some $F(t)$. Adding these components once you've explored the non-time-dependent version gives a ton more to explore!

9.1 Activity

Pick an ODE as a group and explore it as we have done in class. The critical element here is not only working on making the models but finding where the key components of our model evaluation framework appear in your work. It is ok if it doesn't all show up for you. We will dsicuss as a class how we might see these components in our modeling work. As a reminder, here's the framework:

9.1.1 Goal: Investigate physical systems (0.30)

- How well does your computational essay predict or explain the system of interest?
- How well does your computational essay allow the user to explore and investigate the system?

9.1.2 Goal: Construct and document a reproducible process (0.10)

- How well does your computational essay reproduce your results and claims?
- How well documented is your computational essay?

9.1.3 Goal: Use analytical, computational, and graphical approaches (0.30)

- How well does your computational essay document your assumptions?
- How well does your computational essay produce an understandable and parsimonious model?
- How well does your computational essay explain the limitations of your analysis?

9.1.4 Goal: Provide evidence of the quality of their work

- How well does your computational essay present the case for its claims?
- How well validated is your model?

9.1.5 Goal: Collaborate effectively

- How well did you share in the class's knowledge?
  - How well is that documented in your computational essay?
- How well did you work with your partner ? *For those choosing to do so*

Below is some starter code to help you get started.

9.1.6 Starter Code

Here's some working code from last week when we looked at the Van der Pol oscillator to help you get started on your invesigation.

For reference, the Van der Pol oscillator is given by:

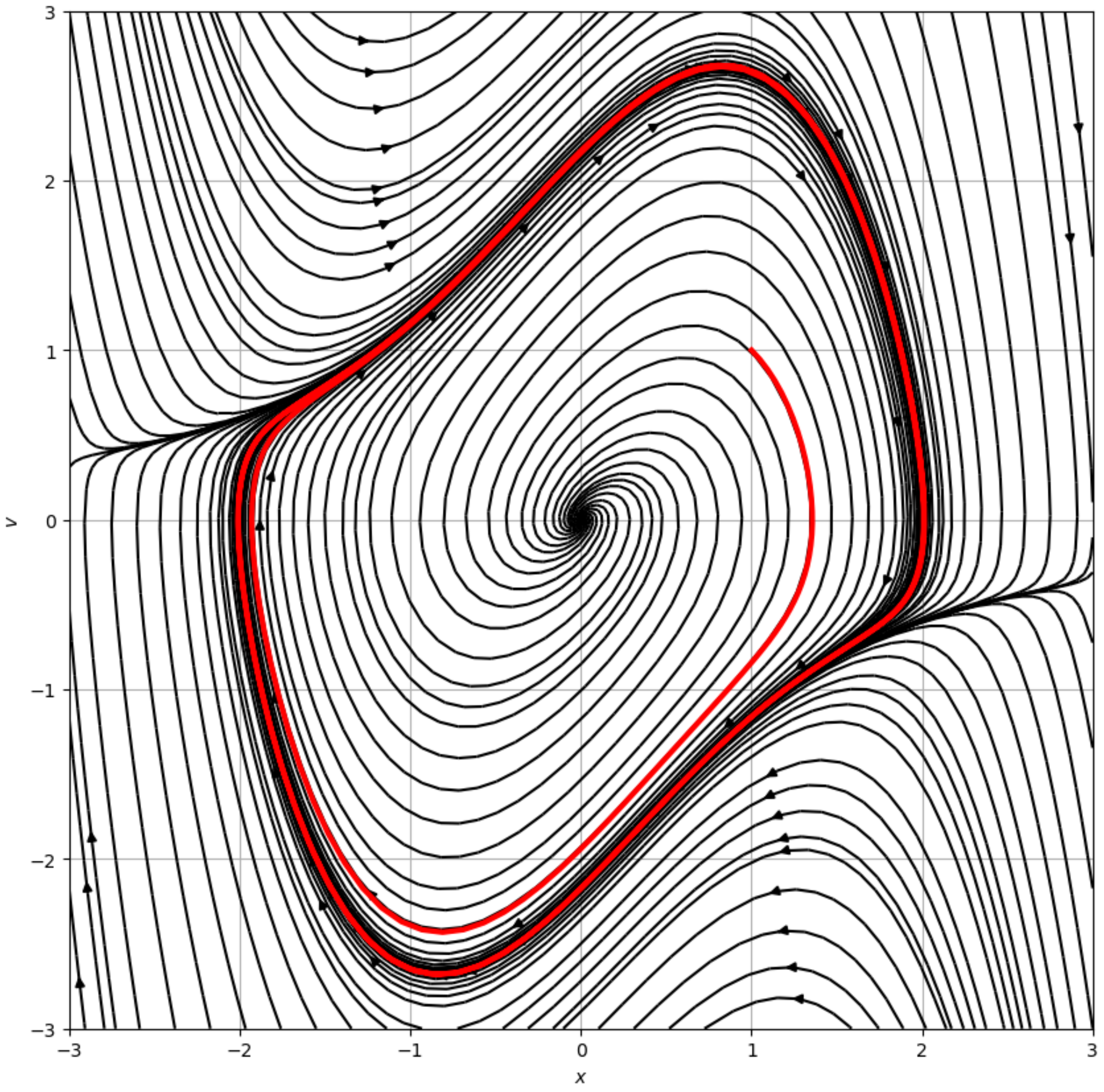
$$\dot{x} = v \qquad \dot{v} = -\mu(x^2 - 1)v - x$$

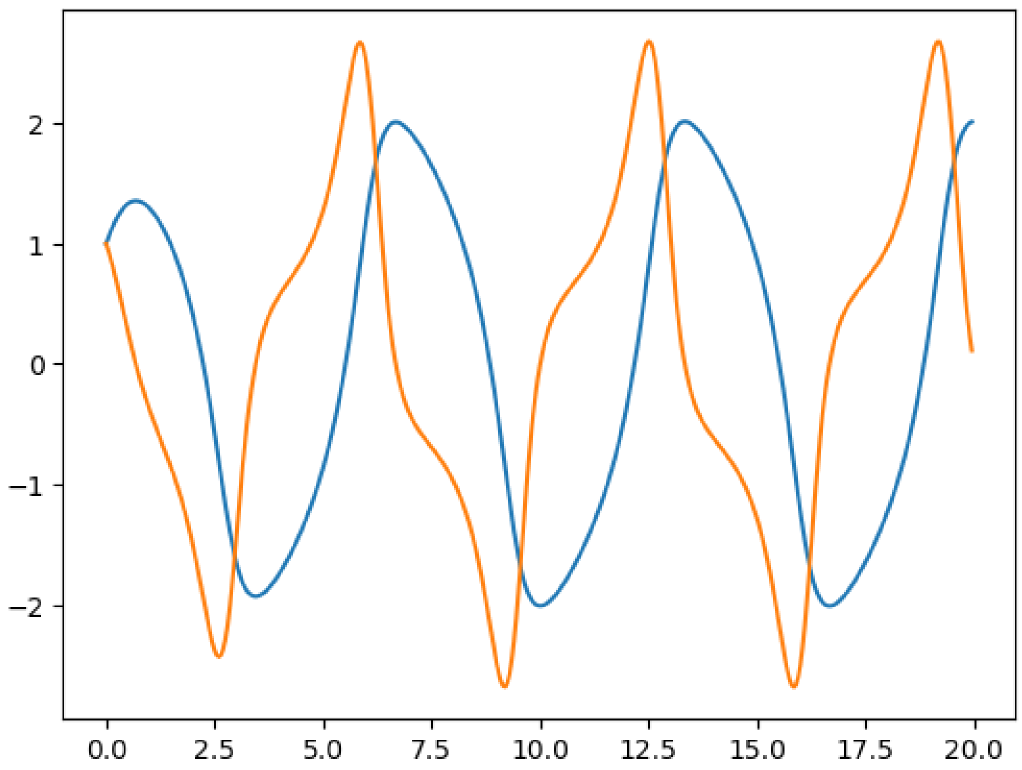
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

def VP_eqn(x, v, mu = 1.):
    xdot, vdot = [v,-mu * (x**2 - 1)*v - x]
    return xdot, vdot

def VP_phase(X, VX, mu):
    xdot, vdot = np.zeros(X.shape), np.zeros(VX.shape)
    Xlim, Ylim = X.shape
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = VX[i, j]
            xdot[i,j], vdot[i,j] = VP_eqn(xloc, yloc,mu)
    return xdot, vdot

def VP_eqn_for_solve_ivp(t,curr_vals, mu=1): # need to rephrase this to work with␣
 ↪what solve_ivp expects
    x, v = curr_vals
    xdot, vdot = VP_eqn(x,v,mu)
    return xdot,vdot

# Numerical Integration
tmax = 20
dt = 0.05
tspan = (0,tmax)
t = np.arange(0,tmax,dt)
mu = 1.
initial_condition = [1, 1]
solved = solve_ivp(VP_eqn_for_solve_ivp,tspan,initial_condition,t_eval = t, args =␣
 ↪(mu,),method="RK45")

# Plotting stuff
N = 40
x = np.linspace(-3., 3., N)
v = np.linspace(-3., 3., N)
X, V = np.meshgrid(x, v)
xdot, vdot = VP_phase(X, V,mu)
ax = plt.figure(figsize=(10,10))
Q = plt.streamplot(X, V, xdot, vdot, color='k',broken_streamlines = False)
plt.plot(solved.y[0],solved.y[1],lw = 3,c = 'red') # plot trajectory from solve_ivp
plt.grid()
plt.xlabel('$x$')
plt.ylabel('$v$')
plt.show()
```

(continues on next page)

(continued from previous page)

```
plt.figure()
plt.plot(t,solved.y[0])
plt.plot(t,solved.y[1])
plt.show()
```




2
1
0
−1
−2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Part III

2 - E&M and PDEs

CHAPTER

TEN

INTRO TO ELECTRICITY AND MAGNETISM

CHAPTER

ELEVEN

26 SEP 23 - GRAPHING ELECTRIC FIELDS

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5,5,5)
y = np.linspace(-5,5,5)
X,Y = np.meshgrid(x,y)
u = X**2
v = Y**2

q = 1e-4
r_source = np.array([0,0])
k = 9e9

def VP_eqn(x, v, mu = 1.):
    xdot, vdot = [v,-mu * (x**2 - 1)*v - x]
    return xdot, vdot

def point_charge_E(X, Y, q,r_source):
    xdot, vdot = np.zeros(X.shape), np.zeros(Y.shape)
    Xlim, Ylim = X.shape
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = Y[i, j]
            xdot[i,j], vdot[i,j] = VP_eqn(xloc, yloc,mu)
    return xdot, vdot

N = 40
x = np.linspace(-3., 3., N)
v = np.linspace(-3., 3., N)
X, V = np.meshgrid(x, v)
xdot, vdot = point_charge_E(X, V,mu)
ax = plt.figure(figsize=(5,5))
Q = plt.quiver(X, V, xdot, vdot, color='k')
plt.grid()
plt.xlabel('$x$')
plt.ylabel('$v$')
plt.show()
```

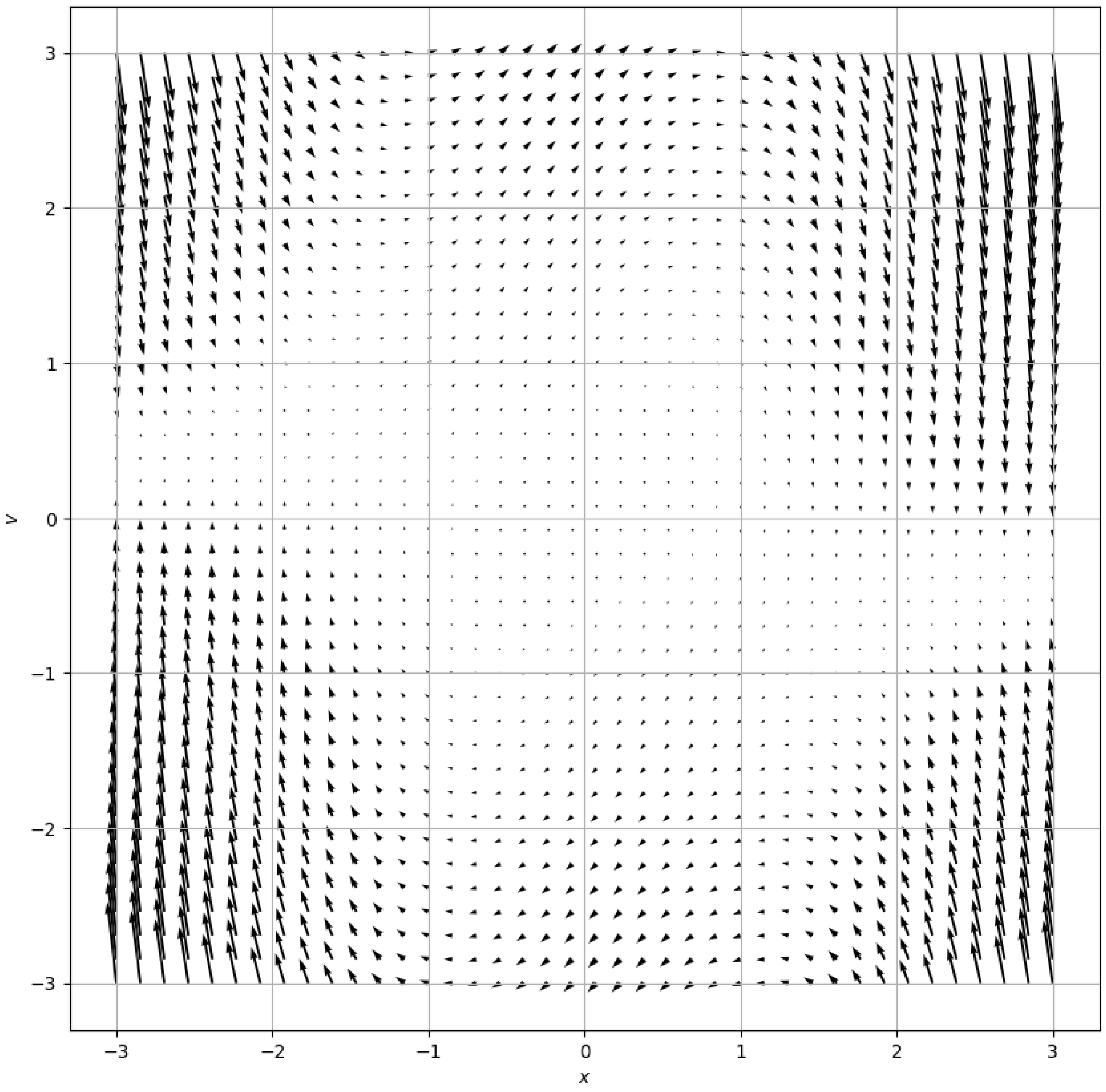
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [1], in <cell line: 32>()
```

(continues on next page)

(continued from previous page)

```
     30 v = np.linspace(-3., 3., N)
     31 X, V = np.meshgrid(x, v)
---> 32 xdot, vdot = point_charge_E(X, V,mu)
     33 ax = plt.figure(figsize=(5,5))
     34 Q = plt.quiver(X, V, xdot, vdot, color='k')

NameError: name 'mu' is not defined
```

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

def VP_eqn(x, v, mu = 1.):
    xdot, vdot = [v,-mu * (x**2 - 1)*v - x]
    return xdot, vdot

def VP_phase(X, Y, mu):
    xdot, vdot = np.zeros(X.shape), np.zeros(Y.shape)
    Xlim, Ylim = X.shape
    for i in range(Xlim):
        for j in range(Ylim):
            xloc = X[i, j]
            yloc = Y[i, j]
            xdot[i,j], vdot[i,j] = VP_eqn(xloc, yloc,mu)
    return xdot, vdot

def VP_eqn_for_solve_ivp(t,curr_vals, mu=1): # need to rephrase this to work with␣
↪what solve_ivp expects
    x, v = curr_vals
    xdot, vdot = VP_eqn(x,v,mu)
    return xdot,vdot

# Numerical Integration
tmax = 20
dt = 0.05
tspan = (0,tmax)
t = np.arange(0,tmax,dt)
mu = 1.
initial_condition = [1, 1]
solved = solve_ivp(VP_eqn_for_solve_ivp,tspan,initial_condition,t_eval = t, args =␣
↪(mu,),method="RK45")

# Plotting stuff
N = 40
x = np.linspace(-3., 3., N)
v = np.linspace(-3., 3., N)
X, V = np.meshgrid(x, v)
xdot, vdot = VP_phase(X, V,mu)
ax = plt.figure(figsize=(10,10))
Q = plt.quiver(X, V, xdot, vdot, color='k')
plt.grid()
plt.xlabel('$x$')
plt.ylabel('$v$')
plt.show()
```


3
2
1
0
−1
−2
−3
v
−3
−2
−1
0
1
2
3
x

CHAPTER

TWELVE

28 SEP 23 - LAPLACE'S EQUATION

CHAPTER

THIRTEEN

3 OCT 23 - MORE PDES

CHAPTER

FOURTEEN

5 SEP 23 - METHOD OF RELAXATION

CHAPTER

FIFTEEN

10 OCT 23 - MAGNETIC FIELDS

CHAPTER

SIXTEEN

ELECTROMAGNETIC WAVES & THE WAVE EQUATION

Part IV

3 - Waves and Complex Analysis

CHAPTER

SEVENTEEN

INTRO TO WAVES

CHAPTER

EIGHTEEN

17 OCT 23 - NORMAL MODES

18.1 Three Coupled Oscillators

Consider the setup below consisting of three masses connected by springs to each other. We intend to find the normal modes of the system by denoting each mass's displacement (x_1, x_2, and x_3).

18.1.1 Finding the Normal Mode Frequencies

? Do this

This is not magic as we will see, it follows from our choices of solution. Here's the steps and what you might notice about them:

1. Guess what the normal modes might look like? Write your guesses down; how should the masses move? (It's ok if you are not sure about all of them, try to determine one of them)
2. Write down the energy for the whole system, T and U (We have done this before, but not for this many particles)
3. Use the Euler-Lagrange Equation to find the equations of motion for x_1, x_2, and x_3. (We have done this lots, so make sure it feels solid)
4. Reformulate the equations of motion as a matrix equation $\ddot{\mathbf{x}} = \mathbf{A}\mathbf{x}$. What is $\mathbf{A}$? (We have done this, but only quickly, so take your time)
5. Consider solutions of the form $Ce^{i\omega t}$, plug that into x_1, x_2, and x_3 to show you get $\mathbf{A}\mathbf{x} = -\omega^2\mathbf{x}$. (We have not done this, we just assumed it works! It's ok if this is annoying, we only have to show it once.)
6. Find the normal mode frequencies by taking the determinant of $\mathbf{A} - \mathbf{I}\lambda$. Note that this produces the following definition: $\lambda = -\omega^2$

18.1.2 Finding the Normal Modes Amplitudes

Ok, now we need to find the normal mode amplitudes. That is we assumed sinusoidal oscillations, but at what amplitudes? We will show how to do this with one frequency (ω_1), and then break up the work of the the other two. These frequencies are:

$$\omega_A = 2\frac{k}{m}; \qquad \omega_B = \left(2-\sqrt{2}\right)\frac{k}{m}; \qquad \omega_C = \left(2+\sqrt{2}\right)\frac{k}{m}$$

? Do this

After we do the first one, pick another frequencies and repeat. Answer the follow questions:

1. What does this motion physically look like? What are the masses doing?

2. How does the frequency of oscillation make sense? Why is it higher or lower than ω_A?

The two cells below have some code that shows how you could've used python to help you when solving this problem:

```python
import numpy as np
import matplotlib.pyplot as plt
from sympy import *
from sympy import Matrix # get symbolic matrix methods
init_printing(use_unicode=True) # make math display good

A = np.array([[-2, 1, 0], [1, -2, 1], [0, 1, -2]]) ## numpy matrix
A_sympy = Matrix(M) ## Take numpy matrix and make it a sympy one

eigenvals, eigenvects = np.linalg.eig(A) # numpy numerical methods
print("numpy eigenvals:",eigenvals)
print("numpy eigenvects:",eigenvects)
print("sympy eigenvals:")
A_sympy.eigenvals() # sympy symbolic methods WARNING: slow for big matrices
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [1], in <cell line: 8>()
      5 init_printing(use_unicode=True) # make math display good
      7 A = np.array([[-2, 1, 0], [1, -2, 1], [0, 1, -2]]) ## numpy matrix
----> 8 A_sympy = Matrix(M) ## Take numpy matrix and make it a sympy one
     10 eigenvals, eigenvects = np.linalg.eig(A) # numpy numerical methods
     11 print("numpy eigenvals:",eigenvals)

NameError: name 'M' is not defined
```

```python
print("sympy eigenvects:")
A_sympy.eigenvects()
```

```
sympy eigenvects:
```

$$\left[\left(-2,\ 1,\ \left[\left[\begin{matrix}-1\\0\\1\end{matrix}\right]\right]\right),\ \left(-2-\sqrt{2},\ 1,\ \left[\left[\begin{matrix}1\\-\sqrt{2}\\1\end{matrix}\right]\right]\right),\ \left(-2+\sqrt{2},\ 1,\ \left[\left[\begin{matrix}1\\\sqrt{2}\\1\end{matrix}\right]\right]\right)\right]$$

18.2 Extending your work

Given what we have done thus far, you can see that we could easily construct the matrix for a N dimensional chain of 1D oscillators. So let's do that.

✎ Do this

Repeat this analysis for a set of N oscillators. Your code should be able to:

1. Take a value of N and construct the right matrix representation
2. Find the eigenvalues and eigenvectors for this matrix.
3. (BONUS) plots the modes automatically
4. (CHALLENGE) time the execution of the analysis

Be careful not to pick too large of an N value to work with because you could melt your CPU easily. Make sure your code can do something like $N = 10$. If you get the timing working, plot time vs number of objects to see how the problem scales with more oscillators.

```
## Your code here
```

18.3 Even further

These models can be used with lattices (solid objects). Draw a sketch of 4 oscillators in a plane connected together in a square shape. Write down the energy equations for this system (assume the springs do not move laterally much). What do the EOMs look like?

18.3.1 Notes

- Partial Solution to Activity

CHAPTER

NINETEEN

19 OCT 23 - MECHANICAL WAVES

CHAPTER

TWENTY

26 OCT 23 - EXAMPLES OF WAVES

CHAPTER

TWENTYONE

31 OCT 23 - COMPLEX ANALYSIS & THE FOURIER TRANSFORM

CHAPTER

TWENTYTWO

2 NOV 23 - DISCRETE AND FAST FOURIER TRANSFORMS

Part V

4 - Random Processes and Distributions

CHAPTER

TWENTYTHREE

INTRO TO DISTRIBUTIONS