Mathematical Inquiry II: Module 1

Lecture notes

MATH 3030

Winter 2025

Dr. Alex Bihlo

Department of Mathematics and Statistics

Memorial University of Newfoundland

A1C 5N3 St. John's (NL), Canada

abihlo@mun.ca

Version: January 2, 2025

Preface

The goal of this course is for you to deepen your understanding of the process of mathematical inquiry, by completing a short research project, and summarizing the results obtained using the LaTeX typesetting language. As the preceding course *MATH 2030 — Mathematical Inquiry I*, also this course has been designed to be completed asynchronously. These lecture notes should provide some brief background to the problems that will be considered in this course. For many students the information provided in these notes as well as some guided self-study will be enough to complete the projects outlined here. For others, some more information may be required, and for those I suggest to consult myself as well as the references provided in this text.

St. John's, January 2025

Contents

Chapter 1

Introduction

1.1 Motivation

The aim of this course is to teach you technical writing in the mathematical sciences. Knowing how to write high-quality technical reports is of practical relevance whether you want to become a researcher or work in industry.

There are two parts to writing a technical report, doing research work worthy of being reported, and then preparing the actual report itself. In this course you will do both. You will be working on three independent projects and will then write technical reports on the findings you obtained.

The three projects are practical and contain aspects you most likely will encounter in practice in your career after graduating. These projects are designed to challenge you, in that they introduce problems that you may have never seen before. They will introduce mathematical concepts you most likely will have never seen before. While this may seem intimidating, solving new problems and being able to figure out how to use mathematical and computational tools you have never used before is probably your most valuable skill as a mathematician. Employers will hire you because you possess this skill.

Working on problems that go beyond your current mathematical understanding is an important exercise you should aim to practice continuously during your undergraduate program. This will allow you to gain experience that is immensely valuable both for academia (should you plan to go to graduate school) and private-sector industry.

To guide your exploration on the projects to be tackled in this course, for most projects a concise introduction to these problems will be provided. This introduction should be enough to get your started but may not go deep enough for your personal taste. Therefore, some key references will be provided where further information could be found. Those references should mostly be regarded as a starting point for your own literature review, they may not be the references that will be most helpful for everyone. Some students may prefer classical textbooks, other may prefer blog posts on the internet explaining the topics covered in these projects. Learning to do a proper literature search is an important part of the scientific writing experience.

Regardless of the problem you are working on, changes are high that someone else has worked on the exact same problem before your. Chances are even higher that someone else has worked a similar problem and used methods that will be helpful for your specific problem. As such it is best practice in scientific research to begin with a literature review.

There are many ways how a scientific literature research can be conducted but one standard

tool that is frequently used is *Google Scholar*,

https://scholar.google.com/

where you can search for scientific papers and books on every subject imaginable.

1.2 Resources

For this course, no previous knowledge of LaTeX or `Python` is assumed. Below I list some of the resources that should help you getting started with both.

1.2.1 LaTeX

A short introduction to the fundamentals of LaTeX is provided as slides and lecture recording. In addition to these resources, there are many great tutorials on LaTeX available online. A simple tutorial covering similar material as the slides provided would be *Learn LaTeX in 30 minutes*:

https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes,

which is available on the website of the online LaTeX editor Overleaf. A more detailed introduction is Chapter 3 of the MATH 2130 manual available on our Department of Mathematics and Statistics website:

https://www.math.mun.ca/~m2130/Manual/ch3_typesetting.pdf.

To use LaTeX you will first have to either install it locally on your computer, with versions available for Linux, Windows and MacOS, or use an online service like Overleaf, which is free for single user. For installation guides see:

https://www.latex-project.org/get/.

While installing LaTeX is generally straightforward, should you run into any issues related to your specific hardware that cannot be resolved easily, I would suggest using Overleaf instead. Note that if you install LaTeX on your own computer you may also have to install a suitable LaTeX editor.

1.2.2 `Python`

`Python` is a high-level, interpretative programming language that has emerged as one of the standard tools in scientific computing, data analysis and machine learning. While for the purpose of this course we will use `Python` (version 3) mainly for simple short programs and visualization purposes, I strongly suggest that you make every effort to pick up as much of this language as you can. Most of the jobs available for applied mathematician in industry require solid experience in `Python` and the more often you use it for the various courses in your undergraduate program, the more experience you will gain using it.

A short (interactive) introduction to `Python` is provided for this course, both as a *jupyter notebook* and as a video recording. In addition to this introduction by me, as with LaTeX there

are a multitude of tutorials and online resources (including free online courses, for example on YouTube) available that will allow you to familiarize yourself with the basic functioning of `Python`. One short resource to start with would be *A Byte of Python*:

https://python.swaroopch.com/.

There are a great many packages available which considerably extend the base functionality of `Python`. In the applied mathematics, scientific computing and data science setting, the most important packages to get familiar with are *Numpy*, *Scipy*, *Pandas*, *Matplotlib*, *Plotly* and *Scikit-Learn*. In this course we will use only some elementary aspects of these packages, so it will be enough if you familiarize yourself with the main purpose of these extensions rather than trying to understand them exhaustively (which would be an overwhelming undertaking).

`Python` can be installed on Linux, Windows and MacOS. As with LaTeX there are also online `Python` interpreter available. One particularly simple and convenient way to use `Python` is via *jupyter notebooks*:

https://jupyter.org/.

`Jupyter notebooks` allow mixing text with code and thus allow for a neat and interactive way to run and present `Python` code and its associated results. I strongly advise to use `jupyter notebooks` for this course as a means to provide the code for the projects to accompany your written reports.

Personally I use `Google Colab` (which is free):

https://colab.research.google.com/,

for most of my `Python` programming, which combines the convenience of `jupyter notebooks` with the ability to interact with local files saved on Google Drive, exporting notebooks to `Github`, etc. It also comes with essentially all `Python` extensions pre-installed that are needed to do most projects in applied mathematics, scientific computing and data science.

Should you run into any issues installing `Python` or any of the associated packages on your local computer (which is fairly straightforward on Linux and MacOS, but slightly less straightforward on Windows), or if you encounter severe performance issues (maybe due to having an outdated computer), then I strongly advise to use an online `Python` interpreter such as `Google Colab` instead.

1.3 Style guidelines for your reports

A template for a report (along with the LaTeX source code) can be found on the course website on Brightspace. It follows the typical style of essentially all papers in the mathematical sciences:

1. *Title*: Choose a meaningful (but typically short) title that gives the reader a reason to look at your work.
2. *Authors:* The list of authors who contributed to the paper in an essential way, along with their work addresses and contact information (for you this is just your own name, since all projects are to be completed by yourself).

3. *Abstract*: This is a very short summary of the main findings of your work. The abstract would normally be the first part read by a prospective reader so be sure to make it advertising and easily readable for the reader to be keen on reading your paper in its entirety.
4. *Introduction*: The introduction would usually contain a broader, mostly non-technical discussion of the problem you are considering in your paper. This is typically where you would include the literature review for your work.
5. *Methods*: Here you would begin to delve into the technical details of your work, e.g. explaining the mathematical or computational methods used.
6. *Results*: Applying the methods described in the previous section, this would be the place to include your results, including tables, figures, etc.
7. *Conclusion*: Here you would summarize the results of your paper and could provide some outlook of what could be considered as next steps in the wider context of the work in which your paper is situated.
8. *Acknowledgements*: Here you would thank people that have helped you in some way with your work (e.g. any peer-reviewers, colleagues, etc.), and acknowledge funding from sponsors.
9. *Appendices*: This is optional, but could include the source codes used in your work, longer proofs to theorems, etc.

The above is a general formula, and not all papers have to follow the exact same formula. Shorter papers may combine the *Methods* and *Results* section, and this may also be appropriate for some of your own reports for this course. Many papers within the area of pure math also do not separate the *Methods* and the *Results* sections, since the main aim of such papers is to prove one or more statements so there is no natural separation between the methods used and the results obtained.

Besides the above general structure for a mathematical report, the following is a short collection of best practices that you should keep in mind when writing your reports:

1. Technical reports are typically written in neutral, concise language; exaggerations should be avoided.
2. Technical reports are more often written in Third Person ("We show that") rather than in First Person ("I show that"), even if there is only a single author.
3. Despite the language in technical reports being neutral, is has to follow all the regular rules of the English language.
4. Your task is to back up everything you are reporting. Sweeping statements without proofs have to be avoided.
5. If you have a conjecture about a statement that you cannot back up, it is acceptable to include this as a conjecture, and explain your reasoning behind it, and what would be required to exhaustively prove this statement.
6. Including plots and tables can make a report more easily digestible, providing summarizing information in an accessible way. For this to work, plots and tables should be as self-contained as possible: Axes have to be labelled; multiple curves should have a legend; if plots are in color, think if they would be still understandable if printed in black and white; if numbers are reported then explain the units being used.

7. If you use information taken from books or scientific articles it is imperative to cite them.

As with any kind of writing, also scientific writing requires a lot of practice. Do not get discouraged if you find it difficult or overwhelming to getting started! For more suggestions, consult the MATH 2130 course manual or [5].

1.4 References

A hallmark of a technical report is that is includes references to other technical documents, such as papers, technical reports or books. As you research a scientific problem, you would start with an extensive literature review. There is not much worse for a scientist than to write a paper on a topic only to discover later that the exact same problem has already been treated elsewhere. To avoid such an unfortunate situation, knowing your subject area is key.

While for the present course some background information on the problems to be considered is already given in these lecture notes, a good report will expand on the problems you will be working on. Doing further literature research, reading papers and appropriate sections of books will strengthen the report you will be writing.

It is crucial to stress the importance of proper citations of sources used. If you find a useful statement for your report in a paper, then you have to cite that paper, and make it part of the bibliography of your report. LaTeX provides a suitable reference management system in the form of `BibTeX`, which allows you to easily add references into a database which can then be used within LaTeX.

There are many different citation styles, but the most common in the mathematical sciences is to either cite a paper by its reference number of the bibliography (as is done in these lecture notes) or in the form of Authors/Year. To cite the book by S.G. Krantz you thus could either use the numerical form, that is [5], or the Author/Year form, that is (Krantz, 2017). The style of your bibliography can be set globally in your LaTeX document. The ordering of the bibliography can be customized as well, with alphabetic ordering being the most common (another option would be to order references according to their occurrence in your report, meaning the first paper you cite would be [1] in your bibliography, the second paper [2], etc.).

Independent of the citation style you are using, references are important. It is not a weakness to cite many papers in a report, as long as the references you cite are relevant to your work. In fact, unless a statement is within the domain of common knowledge (such as the Pythagorean Theorem, the rules of calculus, etc.), it has to be cited.

The source from which you cite is also important. Books and peer-reviewed scientific papers are more credible sources than a blog post off the internet, which usually has not been peer-reviewed and which may be taken down at any point. It is of course absolutely allowed to use information from the internet and cite it accordingly (just indicated what date you accessed that information), but actual research papers largely avoid material from the internet, simply because most of the scientifically useful material can be found in more credible sources. Please consider this for your own report and avoid using pages off the internet as your sole source of information.

As an example, when you are working on Project 1, and you want to describe physics-informed neural networks in some more detail, do not cite the Wikipedia article on PINNs. There are countless papers describing physics-informed neural networks, such as the seminal paper [9] that rekindled the interest in the original work of [6]. If that information is too hard to digest for you, try finding an introductory review paper (there are many such papers that were written

over the past few years) to provide some useful background information for you to include in your report. According to Google Scholar, the paper by Raissi et al. [9] on physics-informed neural networks has been cited 12,324 (!) times as of early January 2025, with the paper only being 5 years old, so there is an abundance of information on this (and similar) method(s) out there that will be accessible to every skill level.

Using information without properly citing the relevant sources (independent of whether these sources are books, articles or the internet) is plagiarism, which is a serious academic offence, both in science and for this course. Avoid plagiarism at any cost!

Any task completed that involves plagiarized code/text will receive zero points on that task.

1.5 Use of generative AI

Recent years have seen the proliferation for numerous generative AI tools, such as *image generators*, including Dall-E, Midjourney, and Stable Diffusion, and *large-language models*, including Claude, GPT, Gemini, LLaMa, and their chat-bot interfaces including Bing chat, chatGPT, and Gemini. These tools have also found their way into academia, with many major publishing houses now mandating disclosure of the use of generative AI in research publications.

Generative AI particularly impacts this course, as it allows the generation of images and large bodies of text from simple textual prompts. While the legality (and ethics) around these generative AI tools still largely remains in question, with regulatory frameworks currently being developed, it is hard to imagine that the importance of these tools will diminish within the next few years. As such, this course takes a pragmatic standpoint towards the use of generative AI:

You are allowed to use generative AI provided you declare that you used it.

That is, uses of generative AI in writing your project report have to be disclosed by including a *Declaration on the use of generative AI in the writing process* section in your report. In this declaration you have to clearly indicate which tools were used for which parts of your report, and for what reasons. For example, if you used chatGPT to help you write an introduction to your report, then your declaration on the use of generative AI should include a statement such as "I have used chatGPT for providing a draft for the introduction of this report, which I have then reviewed and edited.". This statement can be either in the beginning (after the abstract) or in the end (before the references) of your report.

Please pay attention that it is never advisable to simply take the output of large-language models at face value, as they typically require extensive fact checking. This is particularly true in any technical area, as the limited amount of training data in these areas often leads large-language models to hallucinate, i.e. to simply invent incorrect (but often convincingly sounding) text.

Note that you retain full responsibility for the use of any text (or images) created from generative AI, and as such will receive point deductions for incorrect statements in your reports, even if those mistakes were made by generative AI. If you use the outputs of generative AI for more than two reports, or if you do not disclose their use, this will be treated as an academic offence comparable to plagiarism. Undisclosed use of generative AI will thus automatically incur zero points on all tasks for which these tools have been used.

1.6 What to include in your report

In Section 1.3 I have provided a short possible skeleton for your report. What precisely to include in each section is largely up to you. Here are some general thoughts you might want to consider:

1. *Introduction:* A good report should be self-contained, i.e. it should be readable by a colleague who has a solid mathematical education but who would not necessarily be an expert in the specifics of the report. For Project 1 you could provide an overview of machine learning, why neural networks are important, etc., before you delve into the specifics of scientific machine learning. Your goal for the introduction would be to convince the reader that what you are presenting in your report is an important problem that needs to be studied further.
2. *Methods:* Here you would explain why you chose the methods you were using. You would explain the model you were considering in such a manner that it would be readable to someone who has never seen that model before. While for the purpose of this course the models and methods are mostly given (e.g. implementing physics-informed neural networks for Project 1), you would still provide arguments as to why this is the right choice. You could also discuss competing or more general approaches (as applicable), highlight potential limitations, explain why you believe your approach was appropriate, or how these limitations would restrict the generality of what you were doing.
3. *Results:* Here you present a selection of your graphs, tables, etc. An interpretation to all these results has to be given. That is, it is not enough to just include the plots you were asked to produce without any further explanation. Give a critical analysis, explain what these plots are showing, how they are solving the problems you set out to solve, where they fall short, etc.
4. *Conclusion:* Give a concise summary of what you have accomplished in your research and why it is relevant. Honestly assess the strengths and weaknesses, e.g. where further research would be needed to corroborate your hypotheses. Shortly describe what you think would be appropriate to do next within the wider area of that project. Would more complicated models have to be considered? Would you need more data to make more justified statements about that research problem?
5. *Appendix:* Here you could provide (parts of) the `Python` codes you have written. If your code is very long then it would not be appropriate to include all of it as this will make your report appear rather messy. In particular, the various `import` statements, variable initializations, etc. can usually be omitted; rather, you could select and discuss some of the key routines you have written, in particular if they clarify other parts of your report. In other words, if you describe your methods in the *Methods* section, you could reference your computational routines in the *Appendix* for clarification purposes.

There is no general rule as to how to structure a research paper, and how much weight to assign to each section. Different colleagues will have different opinions (hence why this course is being team-taught!), and ideally you will find your own style that will work for you. Aim for a paper that is interesting to read, factually correct, and that does a proper job in convincing the reader that what you have done is of scientific value and should be considered further.

The goal of research is to produce papers that will be read and cited by colleagues. Writing a paper that nobody wants to cite is frustrating for the authors. While most research is highly

specialized and will not gather as many citations as the aforementioned paper by Kingma and Ba, it is still the case that the presentation of your research results is a main contributor of how your paper will be received by the scientific community. You may have proven an important statement or obtained an important result but if you present the proof or that result in an incomprehensible or sloppy way, riddled with typos and grammatical mistakes, then chances are high that your paper will not be successful.

Please remember this for your reports as well, *the presentation of your results is just as important as the results themselves.*

Remark 1. The length of a report is **not** an indication for the quality of a report. A concise, well-thought out 2 page report will be better than an unstructured and unorganized 10 page report. As everybody has a different style, I will not provide guidelines on the lengths of your reports!

Remark 2. In computational mathematics it fortunately becomes more and more customary to provide the source codes for the research you have done. This was unfortunately not the case in the past, which made it hard for reviewers to assess the correctness of the results reported in a scientific paper. The source code can be provided in various ways, e.g. making it publicly available in repositories on online code hosting services such as `Github`, on the website of the journal, or in the appendix of your paper itself. Here we will follow best practices so I ask you to submit your codes along with your reports. Short codes can be provided in the appendix of your paper, longer codes could be uploaded as `.py` (`Python`) or `.ipynb` (`Jupyter notebook`) files to the respective project Dropbox.

1.7 Submitting your report

When you try to publish a scientific paper that you have written in a scientific journal, it will have to undergo *peer-review*. Here the editor of the journal you are submitting your work to will select a few experts in the field of the article and will ask them to carefully read your paper and provide reports on it. Based on these reports, your paper will either be *accepted*, has to *undergo a revision*, or will be *rejected*.

We will use the peer-reviewed method for assessing your reports as well. Once you are happy with your report (or, at the latest, at the deadline for each project) you will send it to me as the 'editor' (in practice you just upload it to the assignment Dropbox on Brightspace). I will then send your report to two of your colleagues and ask them to write a short critique on your report. This critique should honestly (but politely!) assess the strengths and weaknesses of your report. Note that peer-reviews for journals are anonymous, so you will not know who will be reviewing your work, and the reviewers should not include their names in their reports.

Learning to write a report as a reviewer for a scientific document is an important skill as well. In practice you come across a variety of reviewers and not all of them are friendly and polite, and unfortunately not all of their reports are really useful for you as author at all. Here we aim to learn best practices of being a supportive reviewer, with the goal of improving your peers' reports. There will be no *rejection* option, but your goal as reviewer will be to find as many weaknesses as possible in the report you are reviewing, along with concrete suggestions for improvements.

As a reviewer, you can also go through the list of best practices provided in Section 1.3. Have these best practices been followed? If not, then you could provide some helpful suggestions on

how the report could be improved. Are the results faulty? Are the arguments hand-wavy? Are the conclusions justified? Is the presentation of results understandable? Is enough background information provided?

Once your review report is done, you will send it back to me (again, there will be a Dropbox where you can upload the report), and I will then forward this report to the author of the paper you reviewed. The task of you as the author is then to incorporate the feedback you have received. You would correct any mistakes found by the reviewers, or, if you do not agree with a reviewer on some of his/her remarks, you would provide an argumentation as to why you did not incorporate these remarks.

The correction process thus consists of two steps: You correct your paper according to the suggestions of the reviewers, and you collect all of your corrections in a response document (usually entitled *Response to the reviewers*). To give an example, say your reviewer remarks that you forgot to label some axes in your plots, you would then (i) make new plots with the proper axes labels for your paper itself, and (ii) write in your response document that you have included these new plots. Practically, this could be done by copying the respective remark from the reviewer's report in your response document and providing your response thereafter, e.g.:

Remark by reviewer: "I should also like to note that in Figure 2 the x-axis has not been labelled."

Response: "We thank the reviewer for catching this issue. We have added a proper label to this Figure. The x-axis now correctly identifies this variable as *time*."

If there are multiple issues being raised by the reviewer it is not necessary to thank them for each and every single point. Still, try to maintain an overall grateful tone in your response document, even if you disagree with what the reviewer has been proposing. Staying polite despite having an unfriendly reviewer is a skill that unfortunately has to be honed in science (as well as in industry).

In practice, the editor would then forward your response document along with the corrected version of your paper to the same reviewers again, who then will make a final decision (or require some more modifications). While it is generally the case that there will be only one revision, some reviewers may require multiple back-and-forth until they will come to a final decision on whether your paper can be accepted or has to be rejected.

Here we will not do multiple rounds of revision. You will incorporate the feedback of your reviewers within **one week** and send the final version of your paper to me (in practice you will re-upload it to the Dropbox, along with your response document), and I will not send it back to the reviewers. This once-revised version of the paper is what I will then be grading.

As with writing a technical paper itself, also writing reviews and understanding the intricacies of the peer-review process is a skill that takes some time and practice to acquire; upon completing this course, you will have a better understanding of writing reports, reviewing and revising them.

1.8 Marking

The total marks for project 1 are as follows:

1. Project 1: 33% (25% for your own report + 8% for your peer-review reports)

I will grade your report only *after* it has been peer-reviewed by your fellow colleagues, and corrected by yourself. That is, you may have made a mistake in one of your sections and your reviewer catches this mistake. You then prepare a revision for your report upon which your work

is improved, and I will only grade the improved version (along with the response document), not the original version.

Note though that this is **not** a loophole for gaining extra time for submitting your project report. You cannot submit an incomplete project report for peer-review and then complete that project report in the course of the peer-review corrections. The purpose of peer-review is to assess a scientific document which you regard as publishable. You would not send a scientific paper to a peer-reviewed journal and have a section missing, in the hopes that the reviewers will tell you what to write for that section; that is not the task of the reviewers, and such a paper would be rejected by the editor before sending it out to peer-review! Therefore, you have to submit a complete project report at the initial submission deadline, and the reviewers' job is to assess the strengths and weaknesses of that report. They may require some extra work (e.g. saying that an extra figure or some more detailed description would be helpful), but this would be to improve your paper further, rather than because your paper was incomplete.

To reward the effort you put into peer-reviewing your colleagues' report, part of your grade will go towards the quality of your review report. Note that the amount of marks you obtain for peer-reviewing will be independent of the quality of the project report you are reviewing. If you review a report that is already excellent, then you can write a justified report as to why this report is excellent, i.e. why it should be published "as is" and does not have to be revised. If the report you review has weaknesses then you write a justified report suggesting how to improve those weaknesses, i.e. why it should be revised and in what way.

In either case, the marks you obtain for your reviewing work will depend on the quality of your review report alone, not on the quality of the report you are reviewing. Expect to review on two reports for Project 1.

Chapter 2

Project 1: Solving differential equations with neural networks

2.1 An introduction to scientific machine learnings

Scientific machine learning is an emerging discipline within the mathematical sciences, which is concerned with solving problems in real-world applications that have traditionally been solved using classical computational methods. What sets scientific machine learning apart from general machine learning and generative artificial intelligence is the inductive bias provided by the physical laws of the universe that should be accurately reflected in all developed models in this field. An example for such an inductive bias could be conservation of energy in a machine learning based model for the motions of the planets around the sun, or the preservation of angular momentum of two bodies rotating around one another. While modern scientific machine learning emerged less than 10 years ago, it has developed into a prolific research discipline at the intersection of mathematics, computer science, physics and engineering.

Most models from the mathematical sciences are described in the form of differential equations, and as such developing machine learning methods that can include differential equations in some appropriate form is one of the main areas of scientific machine learning. Specifically, recent years have seen a surge of interest in so-called *physics-informed machine learning*, which is a subfield of scientific machine learning, devoted to solving differential equations using machine learning rather than using classical scientific computing. In this project we will investigating physics-informed neural networks, which are neural networks that are trained to approximate solutions of differential equations.

2.2 An introduction to physics-informed neural networks

Differential equations are traditionally solved with classical numerical methods such as finite differences, finite volumes or finite element methods. Recent years have seen the emergence of an alternative solution strategy based on neural networks. The solution of differential equations with neural networks was first proposed in the late 1990s [6] and, as most research on neural networks, has seen an explosion of interest in recent years with the advent of affordable GPU computing and availability of computational frameworks such as `JAX`, `TensorFlow` and `PyTorch`. Today, the method originally proposed in [6] is referred to as *physics-informed neural networks* (PINNs), see [9]. Physics-informed neural networks, along with various extensions to operator learning, are one of the main areas of scientific machine learning today.

To introduce the idea behing PINNs, we begin with a high-level overview. In standard deep learning one trains a neural network solely based on data with the goal to either learn an input–output mapping (supervised learning), or to learn an underlying data distribution (unsupervised learning). If the data that is to be learned is known to come from a differential equation (for example recording the motion of the planets around the sun, which follows the laws of classical and/or relativistic mechanics), then one should aim to train a neural network not solely based on data but rather in a manner that also enforces the underlying differential equations. Including important extra information into a machine learning algorithm is referred to as an *inductive bias*. Knowing the appropriate inductive bias for a problem is typically critical as it often leads to better numerical results compared to generic machine learning algorithms. Thus, for physics-informed neural networks, the inductive bias is the given differential equation itself, and a neural network is then being trained to approximate the solution of this differential equation, along with any given initial and/or boundary conditions for that differential equation.

In practice, the neural network being learned accepts as input the independent variables of the differential equations and outputs the dependent variables. Training of this neural network is done by enforcing that *(i)* the differential equations, *(ii)* the initial condition, and *(iii)* the boundary conditions hold on finitely many collocation points sampled (typically at random) over the domain of the independent variables[1]. The neural network learns to enforce these constraints by minimizing a suitable loss function that consists of mean-squared errors of these three contributions. Hence, physics-informed neural networks are a competing approach for solving differential equations that bypasses any numerical discretization of the differential equations at hand.

2.3 Physics-informed neural networks for a single ODE

Suppose we are given the simple decay problem

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -u(t),$$

with the initial condition $u(t=0) = u_0$. We know that the analytical solution to this problem is $u(t) = u_0 e^{-t}$. We now aim to solve this problem using physics-informed neural networks. That is, we aim to train a neural network $\mathcal{N}^{\boldsymbol{\theta}}(t)$ such that the true solution $u(t)$ is approximated by the numerical solution $u^{\boldsymbol{\theta}}(t) = \mathcal{N}^{\boldsymbol{\theta}}(t)$, i.e. $u(t) \approx u^{\boldsymbol{\theta}}(t)$. The associated neural network $\mathcal{N}^{\boldsymbol{\theta}}$ thus has to accept a single scalar input (the time t) and produce a single scalar output (the numerical solution $u^{\boldsymbol{\theta}}$). Note that the quality of the numerical solution will depend on the weights and biases $\boldsymbol{\theta}$ of the neural network $\mathcal{N}^{\boldsymbol{\theta}}$. The class of neural networks typically used for PINNs are standard multi-layer perceptrons, using a suitable (differentiable!) activation function throughout all layers of the network. Other neural network architectures (e.g. convolutional neural networks [3], transformer-based architectures [10] or Kolmogorov–Arnold networks [7]) can be used as well.

To train the neural network $\mathcal{N}^{\boldsymbol{\theta}}$ to indeed solve the given differential equation, its loss function has to incorporate the associated initial value problem. Let us define the *physics-informed loss function*

$$\mathcal{L}(\boldsymbol{\theta}) \sim \mathcal{L}_\Delta(\boldsymbol{\theta}) + \gamma \mathcal{L}_\mathrm{i}(\boldsymbol{\theta}),$$

[1] We make all of this precise below.

where

$$\mathcal{L}_\Delta(\boldsymbol{\theta}) \sim \left(\frac{\mathrm{d}u^{\boldsymbol{\theta}}}{\mathrm{d}t} + u^{\boldsymbol{\theta}} \right)^2$$

is the *differential equation loss* and

$$\mathcal{L}_\mathrm{i}(\boldsymbol{\theta}) = \left(u^{\boldsymbol{\theta}}(0) - u_0 \right)^2$$

is the *initial value loss*, with γ being a positive constant. If $\mathcal{L}_\Delta = 0$ the differential equation will be exactly satisfied and if $\mathcal{L}_\mathrm{i}(\boldsymbol{\theta}) = 0$ the initial value will be exactly satisfied by the neural network solution $u^{\boldsymbol{\theta}}$. Note that the loss weight constant γ is important as there is no a priori guarantee that the magnitudes of the differential equation and the initial value losses will be comparable. If one of the two loss contributions dominates the total loss then choosing an appropriate value for γ can be essential to make sure that both loss components contribute equally to the total loss.

A novel aspect in this problem is that the loss function $\mathcal{L}$ explicitly depends on the derivative of $u^{\boldsymbol{\theta}}$. How can we compute this derivative? Fortunately, we can use the same automatic differentiation routine for *exactly* (up to machine precision) computing this derivative that is used to compute the gradient of the neural network with respect to the weights of the network. This works as a neural network is a differentiable function of its inputs (hence why differentiable activation functions are required), the derivatives of which can be obtained exactly with automatic differentiation. Thus, no numerical differentiation such as finite differences or Runge–Kutta time stepping is necessary, which strongly simplifies the discretization procedure in comparison to traditional numerical methods. Physics-informed neural networks are thus truly meshless methods, which allows using them on complicated domains, such as on embedded surfaces or for problems with complicated boundary conditions.

Here is a code snippet illustrating how the PINN loss can be computed in `TensorFlow`:

```
# Outer gradient for tuning network parameters
with tf.GradientTape() as tape:

  # Inner gradient for derivatives of u w.r.t. t
  with tf.GradientTape() as tape2:
    tape2.watch(t)
    u = model(t)

  # Derivative of the neural network solution
  ut = tape2.gradient(u, t)

  # Define the differential equation loss
  eqn = ut + u
  DEloss = tf.reduce_mean(eqn**2)

  # Define the initial value loss
  u_init_pred = model(t0)
  IVloss = tf.reduce_mean((u_init_pred - u_init)**2)

  # Composite loss function
  loss = DEloss + gamma*IVloss

# Return the model gradients for the gradient descent step
grads = tape.gradient(loss, model.trainable_variables)
```

One important detail we have omitted so far is *where* we want to evaluate/minimize the loss function. The initial value problem considered is defined over the temporal domain $[0, t_\mathrm{f}]$, so we want to evaluate our loss function for that domain only. Unfortunately, it would not be feasible to minimize the loss everywhere in $[0, t_\mathrm{f}]$, which contains infinitely many points! Instead, we sample our temporal domain at finitely many random locations t_i, $i = 1, \ldots, n$, i.e. we have $0 \leqslant t_i \leqslant t_\mathrm{f}$ for each $i = 1, \ldots, n$. We then define the differential equation loss at these finitely many points only,

$$\mathcal{L}_\Delta(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\mathrm{d}u^{\boldsymbol{\theta}}(t_i)}{\mathrm{d}t} + u^{\boldsymbol{\theta}}(t_i) \right)^2 .$$

If this loss function is zero, then the neural network will exactly satisfy the differential equations in the points t_i.[2] The actual composite physics-informed loss function being minimized is therefore the mean squared error loss

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\mathrm{d}u^{\boldsymbol{\theta}}(t_i)}{\mathrm{d}t} + u^{\boldsymbol{\theta}}(t_i) \right)^2 + \gamma (u^{\boldsymbol{\theta}}(0) - u_0)^2 .$$

The points $\{t_i\}|_{i=1,\ldots,n}$ are referred to as *collocation points*. They are akin to the time steps in a numerical solution of differential equations. Note that these collocation points could be chosen arbitrarily. They could lie on a regular mesh, etc., but it was found experimentally [9] that sampling them using suitable random sampling, such as Latin hypercube sampling, gives the most accurate numerical results. For stiff problems or for partial differential equations that can develop shocks or rapidly changing solutions, it was also found beneficial to adaptively refine the collocation points used throughout the (spatio-)temporal domain of the problem.

For our initial value problem defined over $[0, t_\mathrm{f}]$, for the example of the final time $t_\mathrm{f} = 1$, here is how 20 collocations points could be randomly distributed:

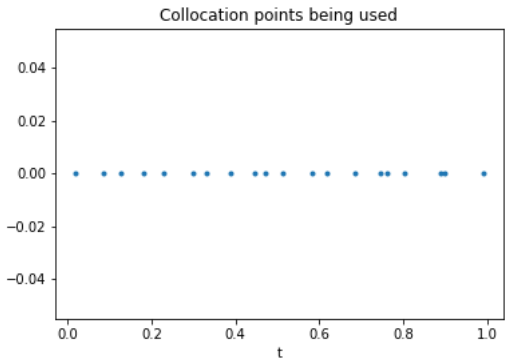

Figure 2.1: An example for a collection of 20 collocation points, randomly sampled from the interval [0,1]. In each of the collocation points, we aim that the given MLP approximation for u satisfies the differential equation $u' = u$.

With all of this in place, we can solve our differential equation using a neural network. We choose here an MLP with 4 hidden layers and 20 units each, using the hyperbolic tangent activation function, to obtain the following solution, using $n = 100$ collocation points:

[2] This is akin to Hermite interpolation, with the enforcement of derivatives at the interpolating points being replaced with enforcing that a differential equation holds for a function parameterized by a neural network. You will see/might have seen Hermite interpolation in MATH 3132.

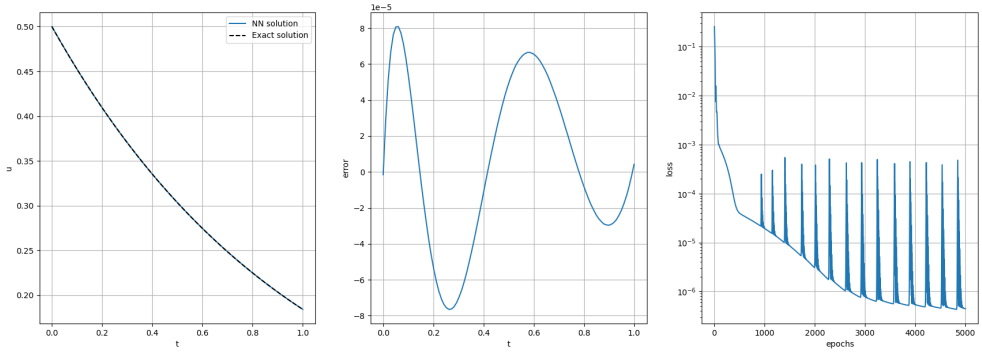

Figure 2.2: Results for a trained physics-informed neural network for the decay problem $u' = u$, with initial condition $u(0) = 1$. *Left:* Solution obtained from the physics-informed neural network. *Middle:* Numerical error compared to the exact solution $u(t) = e^{-t}$. *Right:* Loss as a function of the number of training steps.

2.4 Project

This section contains a concise description of the tasks to be complete for this module.

2.4.1 Physics-informed neural networks for systems of differential equations

The code used to train the physics-informed neural network shown in Fig. 2.2 is provided on `Brightspace`. The goal of this first part of this module is to extend this code to a system of differential equations. In particular, we consider the Lorenz–1960 model, which was derived in [8] as the minimal model that retains some relevance for the equations governing the evolution of the atmosphere. This model is given as

$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= kl\left(\frac{1}{k^2+l^2} - \frac{1}{k^2}\right) yz, \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= kl\left(\frac{1}{l^2} - \frac{1}{k^2+l^2}\right) xz, \\ \frac{\mathrm{d}z}{\mathrm{d}t} &= \frac{kl}{2}\left(\frac{1}{k^2} - \frac{1}{l^2}\right) xy, \end{aligned} \tag{2.1}$$

with initial conditions $x(0) = x_0$, $y(0) = y_0$ and $z(0) = z_0$. Here k and l are positive constants.

Complete the following tasks:

1. Extend the provided code on `Brightspace` for solving the simple decay problem to solving the Lorenz–1960 model instead. The goal is to solve the Lorenz–1960 model for a variety of final integration times t_f. In particular, experiment with $t_\mathrm{f} \in \{1, 5, 10, 20\}$, using a suitable number of collocation points n and then training the physics-informed neural network until the loss starts to level out. Choose $k = 1$ and $l = 2$, but feel also free to experiment with different values for k and l to see how your results change. As initial conditions, use $x(0) = 1$, $y(0) = 0.5$ and $z(0) = 1$.
2. In addition to the time series of the total loss (shown in Figure 2.2), also plot the time series for the individual loss components $\mathcal{L}_\Delta$ and $\mathcal{L}_\mathrm{i}$. Also note that for the Lorenz–1960 model, the differential equation and the initial condition loss each consist of three components,

one for each variable x, y and z and so multiple loss weight constants γ might be needed. Which of the loss components dominates the total loss? Does this inform how you should scale each loss component with an appropriate value for the γ?

3. To compare your results against another numerical method, use `scipy`'s `solve_ivp` to obtain a numerical solution using a classical Runge–Kutta integrator. How large is the discrepancy, i.e. the error, between the physics-informed neural network and the Runge–Kutta integration for different t_f?

Remark 3. Physics-informed neural networks requiring solving what is referred to as a *multi-task* optimization problem. There is one task associated with each differential equation and initial condition, and adding all the loss components associated with each task together in a weighted sum is just one of the many ways how multi-task optimization problems can be tackled[3]. This is still a field of active research, both for physics-informed neural networks and for general multi-task optimization problems arising in machine learning research.

2.4.2 Physics-informed neural networks for higher-order equations

In the previous task we have extended physics-informed neural networks to systems of first order differential equations. In this task, we investigate the possibility to extend them to higher-order differential equations.

You may have seen in your ODEs class that any nth order differential equation can be equivalently written as a system of n first-order equations. Incidentally, this is what you have to do to use off the shelf numerical integrators such as `scipy`'s `solve_ivp`, which only works for systems of first order differential equations. However, for physics-informed neural networks a challenge that arises is that an nth order differential equation requires just a single differential equation loss term while the associated system of n first order equations would require n loss terms. As such, the associated optimization problem to be solved can be harder as it might be more challenging to balance and minimize n loss contributions rather than a single loss contribution[4]. This is one of the many ways in which scientific machine learning differs from classical scientific computing.

Here we study the simple harmonic oscillator, a single second-order ordinary differential equation given by

$$m\frac{\mathrm{d}^2u}{\mathrm{d}t^2} + ku = mu'' + ku = 0, \tag{2.2}$$

where m and k are the mass and the spring constant of the harmonic oscillator, respectively. The initial conditions are given by $u(0) = u_0$ and $u'(0) = u_0'$. We choose $u_0 = u_0' = 1$ and $m = 1$ and $k = 2$. For the final integration time, choose 1, 2 and 5 full periods of the solution.

Complete the following tasks:

1. Solve the harmonic oscillator with a physics-informed neural network by representing (2.2) as a system of two first-order differential equations.
2. Solve the harmonic oscillator with a physics-informed neural network by solving (2.2) directly as a second-order equation.

Which of the two representations gives better numerical results?

[3] If you want to learn more about this, you can do a literature search of Pareto optimality.

[4] Of course, both representations of the nth order equation would still require n initial or boundary conditions.

2.4.3 Improving physics-informed neural networks with hard constraints

One of the main drawbacks of the above approach to physics-informed neural networks is that it requires one to learn the initial condition. This is unsatisfactory as, in contrast to the solution of the differential equation itself, the initial condition is known and therefore should not have to be learned by the neural network.

It is possible to directly include the initial condition into the solution ansatz for the neural network solution as a *hard constraint*. Hard constraining the initial condition means that the neural network solution will satisfy the initial condition for all values of the neural network weights.

There are infinitely many ways for hard-constraining an initial condition. For the single ordinary differential equation $u' = f(t, u)$ with initial condition $u(0) = u_0$, a simple hard constrained neural network is

$$u^{\boldsymbol{\theta}}(t) = u_0 + t \cdot \mathcal{N}^{\boldsymbol{\theta}}(t). \tag{2.3}$$

Note that for all values of the weights $\boldsymbol{\theta}$ of the neural network $N^{\boldsymbol{\theta}}(t)$ we necessarily have $u^{\boldsymbol{\theta}}(0) = u_0$ and as such we do not have to enforce the initial condition during training of the physics-informed neural network. This means, that the total physics-informed loss reduces to the differential equation loss, i.e. $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_\Delta(\boldsymbol{\theta})$.

Complete the following tasks:

1. Revisit the case of the Lorenz–1960 model, this time using a hard-constrained PINN. Is your result more accurate than the soft-constrained PINN?
2. Revisit the harmonic oscillator in both the first-order and second-order representation using hard constraints. For the second-order representation, note that you will have to construct a solution ansatz such that $u^{\boldsymbol{\theta}}(0) = u_0$ and $\frac{\mathrm{d}u^{\boldsymbol{\theta}}}{\mathrm{d}t}(0) = u_0'$. To construct such a solution ansatz, you can use a second-order Taylor series approximation.
3. The hard constraint (2.3) is only one possible way to enforce that the neural network solution ansatz $u^{\boldsymbol{\theta}}(t)$ satisfies the initial condition. One obvious downside of this ansatz is that for longer time intervals, the multiplying factor t grows and as such the neural network $\mathcal{N}^{\boldsymbol{\theta}}(t)$ will have to offset this growth. There are other ways how hard constraints can be constructed which can prevent this linear growth. Come up with at least two different hard constraints, and test them out for the harmonic oscillator represented as a system of two equations.

2.5 Why this project is relevant

Scientific machine learning, and the subfield of physics-informed machine learning have seen an exponential increase of interest in the last 5 years only. Today, almost all fields of science use machine learning as a main tool to advance knowledge, which is an unprecedented scientific revolution. (Scientific) machine learning is at the heart of the 2024 Nobel prices in physics and chemistry, respectively. Scientific machine learning has led to breakthroughs in areas such as protein folding [4] and weather prediction [1]. As such, getting familiar with scientific machine learning is of critical importance if you plan on pursuing research or a career in the mathematical sciences.

2.6 Further reading

Since the field of scientific machine learning is quickly evolving, so is the associated literature, and this necessitates keeping up with the current literature in the field. For one relatively recent review paper on this subject, see [2].

Bibliography

[1] Bi K., Xie L., Zhang H., Chen X., Gu X. and Tian Q., Accurate medium-range global weather forecasting with 3D neural networks, *Nature* **619** (2023), 533–538.

[2] Cuomo S., Di Cola V.S., Giampaolo F., Rozza G., Raissi M. and Piccialli F., Scientific machine learning through physics–informed neural networks: where we are and what's next, *J. Sci. Comput.* **92** (2022), 88.

[3] Goodfellow I., Bengio Y. and Courville A., *Deep learning*, MIT press, 2016.

[4] Jumper J., Evans R., Pritzel A., Green T., Figurnov M., Ronneberger O., Tunyasuvunakool K., Bates R., Žídek A., Potapenko A. *et al.*, Highly accurate protein structure prediction with AlphaFold, *nature* **596** (2021), 583–589.

[5] Krantz S.G., *A primer of mathematical writing*, vol. 243, American Mathematical Soc., 2017.

[6] Lagaris I.E., Likas A. and Fotiadis D.I., Artificial neural networks for solving ordinary and partial differential equations, *IEEE Trans. Neural Netw.* **9** (1998), 987–1000.

[7] Liu Z., Wang Y., Vaidya S., Ruehle F., Halverson J., Soljačić M., Hou T.Y. and Tegmark M., KAN: Kolmogorov-Arnold networks, *arXiv preprint arXiv:2404.19756* (2024).

[8] Lorenz E.N., Maximum simplification of the dynamic equations, *Tellus* **12** (1960), 243–254.

[9] Raissi M., Perdikaris P. and Karniadakis G.E., Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.* **378** (2019), 686–707.

[10] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser L. and Polosukhin I., Attention is all you need, in *Advances in Neural Information Processing Systems*, vol. 30, edited by I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, Curran Associates, Inc., vol. 30, 2017 .