Scientific Computing
MATH 5340

Lecture 1

Syllabus

Scientific Computation. MATH 5340 (Fall 2023)

Instructor: Nestor Guillen (nestor@txstate.edu)

Course overview

Welcome to Scientific Computation!. In this course you will be given a modern introduction to the field of scientific computing. In short, scientific computing is the area of mathematics concerned with the analysis and creation of algorithms for the problems of continuum mathematics -- such as the solution of a differential equation, the determination of an equilibrium probability distribution for a stochastic process, the design of surfaces which optimize their shape according to some geometrical or mechanical criterium, and more. By its very nature this is a field that combines tools and ideas from advanced mathematics (real analysis, multivariable calculus, calculus of variations, differential equations) with computer programming (in our course's specific case, the Python programming language). This means that in this course you will be dealing with pure mathematical questions as well as with practical coding assignments. Both will involve creative problem solving and applying the theories and methods learned in lectures and in the material.

From a more official perspective, *The Texas State Course Catalog* entry for MATH 5340 is as follows

> MATH 5340 : This course will involve the analysis of algorithms from science and mathematics, and the implementation of these algorithms using a computer algebra system. Symbolic numerical and graphical techniques will be studied. Applications will be drawn from science, engineering, and mathematics. A knowledge of differential equations is expected.

Contact information. My office is in MCS 468 and my email is nestor@txstate.edu.

Lectures time and location. Class meets Tuesdays and Thursdays from 3:30 to 4:50 pm in Derrick 331.

Office Hours. Tuesdays 2:00-3:00 pm, or by appointment (request via email at least one day in advance).

Emails. I will get back to any email within 72 hours. Preferably, questions about the content of the class should be posted on the course forum, with email used for individual concerns or questions -- but this is not a strict rule. **Please make sure to start the subject line of your emails with 'MATH5340', so that I can reply promptly.**

Prerequisites. An practical and theoretical understanding of multivariable calculus, differential equations, and linear algebra is a must for the course. Basic programming skills, as well as familiarity with Python and mathematical topics like real analysis or topology are helpful additions but not strictly necessary for the course.

Course materials. You will need access to a computer that can run Python as well as Jupyter Notebooks -- more information on this will be given in class.

We will not be following one book -- a lot of the course material will be provided in the form of Jupyter Notebooks and slides in PDF format. That being said, the course will refer to or follow with some frequency each of the following textbooks:

J. Solomon's Numerical Algorithms by Solomon.

The 2023 edition of the SciPy Lectures.

L.C. Evans' Partial Differential Equations

T. Hastie, R. Tibshirani, and J. Friedman's Elements of Statistical Learning

A. Ern and J. Guermond's Theory and Practice of Finite Elements

Course evaluation

Overview For this course's evaluation I will attempt to follow Patrick Winston's grading philosophy, and the course evaluation will consist of the following

- 4 quizzes (each quiz takes place in class and lasts 1 hour)
- 1 Final (consisting of 5 parts)
- 5 Problem Sets

Each quiz, problem set, and each part of the final will receive a numerical grade in the form of an integer between 0 and 5. These numbers are combined to form a *score* according to the following rule

$$s = \frac{3*(\text{Problem Set Average}) + \max\{\text{Quiz 1}, \text{Final Part 1}\} + \ldots + \max\{\text{Quiz 4}, \text{Final Pa}}{8}$$

The final exam will have five topics, the first four corresponding to the quizes done throughout the semester -- you only need to excel in either the quiz or the corresponding part of the final to get the full grade (this is the function of the $\max$ in the formula for s). The last and fifth part of the final will be a topic not covered in the quiz so it stands on its own. Note the factor of 3 in front of the Problem Set Average -- this means the Problem Set Average amounts for 3/8 of the final score.

Quizzes: The four quizzes will be on each of the first four topics of the class, and they will take place on the following weeks at the time of Lecture:

1. Function spaces (end of week 4)
2. ODE (end of week 7)
3. Graphs, Laplacians, and Markov Chains (end of week 11)
4. Gradient descent and stochastic optimization (end of week 13)

Problem Sets: The problem sets will have flexible deadlines (to be announced throughout the semester). If a student does not submit a problem set they will get an automatic grade of 0 for that problem set.

Submission guidelines: Your submitted problem sets must be submitted in PDF format plus a .ipynb file. Files should be named as: YourLastName_ProbSetN.pdf and YourLastName_ProblemSetN.ipynb, here N is the corresponding number for the problem set.

Important dates: As mentioned earlier, problem sets will be due on Fridays at 11:59 pm.

- First two problem sets due by Friday, September 29th at 11:59 pm
- Quizzes: Thursday lecture on weeks 4, 7, 11, and 13.
- Final: Week of December 4th.

Final Course Grade: Your final course grade is computed fromm the score s according to the following table

Letter grade	Score range
A	$4.4 \leq s \leq 5$
B	$3.4 \leq s < 4.4$
C	$2.4 \leq s < 3.4$
D or F	$s < 2.4$

Course plan

The course is roughly divided in 5 parts.

- Introduction / motivation
- Part 1: Function spaces (their representation, discretization, and approximation)
- Part 2: Ordinary differential equations
- Part 3: Graphs, Laplacians, and Markov Chains
- Part 4: Gradient descent and stochastic gradient descent
- Part 5: Basics of finite elements

In [ ]:

Scientific Computing
This course has five parts

1. Representation of function spaces (approximation, interpolation, Fourier series, artificial neural networks)
2. Schemes for Ordinary Differential Equations
3. Graphs, Laplacians, and Markov Chains
4. Nonlinear optimization (gradient descent and stochastic gradient descent)
5. Basics of the Finite Element Method

Scientific computing and its problems

Scientific computing is concerned with algorithms for mathematical problems involving the continuum:

dynamical systems
continuum mechanics
calculus of variations
linear programming
optimal control
statistical inference
shape optimization
and many more...

What are some of these mathematical problems?

Some you might have seen before – in a differential equation course or statistis course, for example.

Often – almost always, but not always – the problem involves determining an unknown function, which we know to have some important physical, statistical, or computational property.

Let's discuss some basic examples

Ordinary Differential Equations

Consider the generic **Initial value problem** for an ODE:

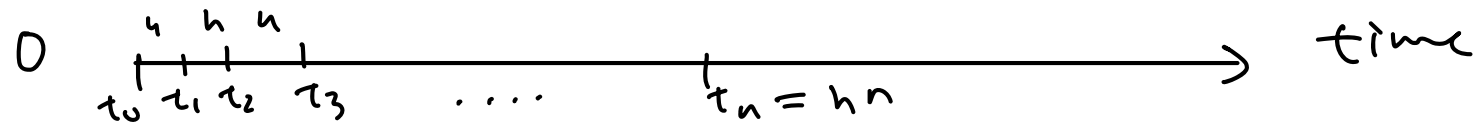
Find a function $x : [0, T] \to \mathbb{R}^d$ solving

$$\dot{x}(t) = f(x(t), t)$$
$$x(0) = x_0$$

In general there is no useful analytical formula for the solution to this problem – save for some rare but important special cases.

$\Delta t = 0.001$

Ordinary Differential Equations



As far back as Euler, we have tried to deal with this by solving approximate discrete problems

x_{n+1} x_n x_{n+1}

$$x(t_{n+1}) - x(t_n) = (\Delta t_n) f(x(t_{n+1}), t_{n+1}), \; n = 0, 1, \ldots$$

x_0 $x(t_0) = x_0$

$$\dot{x}(t) = \lim_{\varepsilon \to 0} \frac{x(t+\varepsilon) - x(t)}{\varepsilon} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

$$\approx f(x(t+\Delta t), t+\Delta t)$$

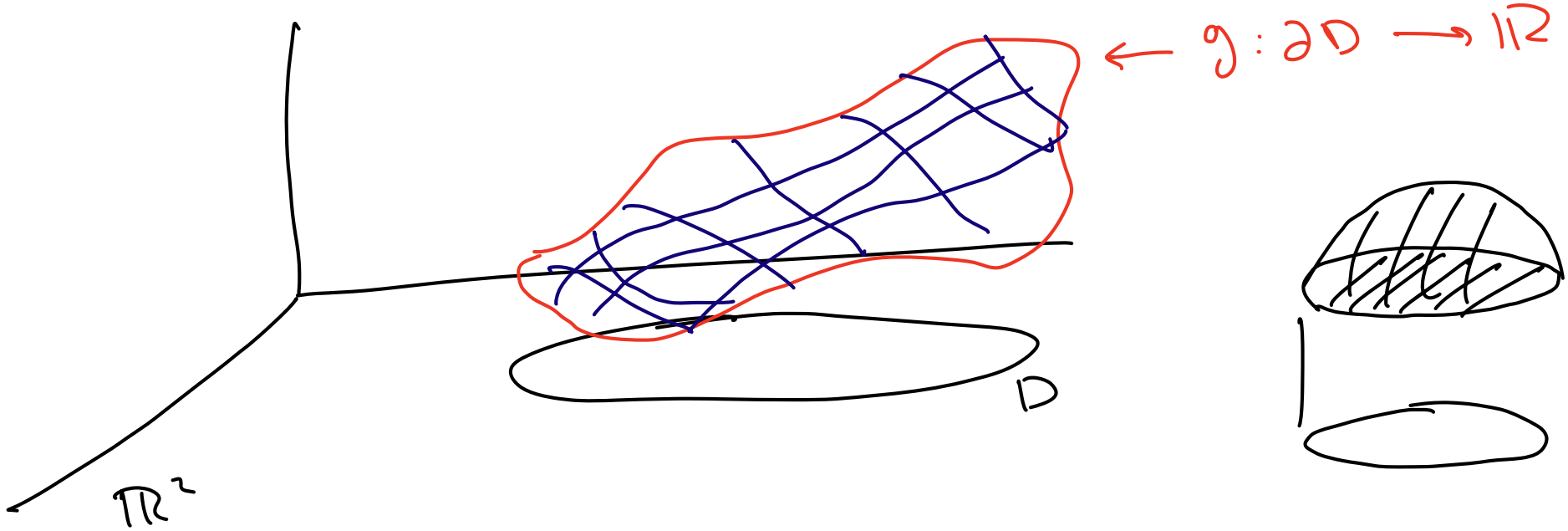
$x_{n+1} \approx x(t_{n+1})$

Partial Differential Equations

Problem

Given $D \subset \mathbb{R}^2$ and a function $g : \partial D \to \mathbb{R}$, find a function $u : D \to \mathbb{R}$ solving

$$\operatorname{div}\left(\frac{1}{\sqrt{1+|\nabla u|^2}}\nabla u\right) = 0 \text{ in } D, \quad u = g \text{ on } \partial D$$



Partial Differential Equations

The Plateu Problem

Given $D \subset \mathbb{R}^2$ and a function $g : \partial D \to \mathbb{R}$, consider the problem

$$\text{Minimize} \quad \int_D \sqrt{1 + |\nabla u|^2}\, dx$$

$$\text{subject to:} \quad u = g \text{ on } \partial D$$

These two problems are basically the same, the Euler-Lagrange equation for this variational problem is $\operatorname{div}\left(\frac{1}{\sqrt{1+|\nabla u|^2}} \nabla u \right) = 0$

Partial Differential Equations

The Dirichlet problem

Given $D \subset \mathbb{R}^d$ and a function $g : \partial D \to \mathbb{R}$, find a function $u : D \to \mathbb{R}$ solving

$$\Delta u = 0 \text{ in } D, \quad u = g \text{ on } \partial D$$

$$\left(\partial_{x_1}^2 u + \partial_{x_2}^2 u + \cdots + \partial_{x_d}^2 u = \operatorname{div}(\nabla u) \right)$$

Partial Differential Equations

The Dirichlet problem (variational formulation)
Given $D \subset \mathbb{R}^d$ and a function $g : \partial D \to \mathbb{R}$, consider the problem

$$\text{Minimize} \quad \int_D |\nabla u|^2 \, dx$$
$$\text{subject to:} \quad u = g \text{ on } \partial D$$

(The minimizer, if it exists, will solve

$$\Delta u = 0 \quad \text{in} \quad D \;)$$

Partial Differential Equations

More general variational problems

Given $D \subset \mathbb{R}^d$ and a function $g : \partial D \to \mathbb{R}$, consider the problem

$$\text{Minimize} \quad \int_D L(\nabla u, u, x)\, dx$$

$$\text{subject to:} \quad u = g \text{ on } \partial D$$

This contains the two previous examples:

$$L(\nabla u, u, x) = \sqrt{1 + |\nabla u|^2}$$

$$L(\nabla u, u, x) = |\nabla u|^2$$

Partial Differential Equations

Euler's equation for incompressible homogeneous fluids

Given $u_0 : \mathbb{R}^3 \to \mathbb{R}^3$ find a function $u : \mathbb{R}^3 \times [0, T] \to \mathbb{R}^3$ solving

$$\partial_t u + u \cdot \nabla u = 0$$
$$\operatorname{div}(u) = 0$$
$$u(t = 0) = u_0$$

Supervised Learning

(First, some background on images)

In principle, a (grayscale) image is a function

$$f : [0,1]^2 \to [0,1]$$

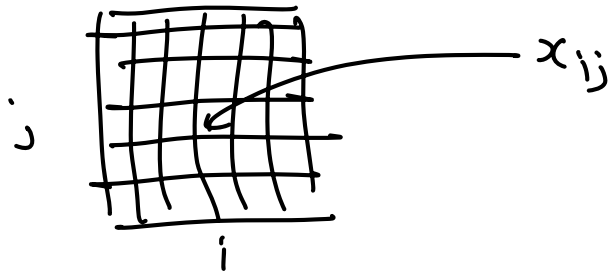
Using RGB, color images can be understood as

$$f : [0,1]^2 \to [0,1]^3$$

where f_1, f_2, f_3 correspond to red, green, and blue.

Supervised Learning

(First, some background on images)



This is an idealization, as screens have finite resolution.

More realistically, an image can be seen as a function

$$f : \{x_{ij} \mid i = 1, \ldots, N,\ j = \ldots, M\} \to [0,1]^3$$

where x_{ij} are the picture elements (**pixels**) forming the picture. The numbers N and M give the **resolution** of the image.

Supervised Learning

Thus, a color image of 1080×1080 pixels can be represented as a vector in $\mathbb{R}^p$, where $p = 3499200$.

A typical supervised learning problem
We would like a piece of code that takes as input a 1080×1080 photo and returns "dog" or "not a dog" according to whether the photo is a portrait of a dog or not. The only thing we are given for this task is a larga data set of examples.

Supervised Learning

Concretely, we are given

Training data: (x_k, y_k) $k = 1, \ldots, N$

Here, $y_k = 1$ if x_k is a photo of a dog and $y_k = 0$ otherwise

Next, we decide on (or are given as part of the learning problem) a class of functions.

The class of functions is one that we beieve is likely to have "the" function that determines if a photo has a dog or not.

Supervised Learning

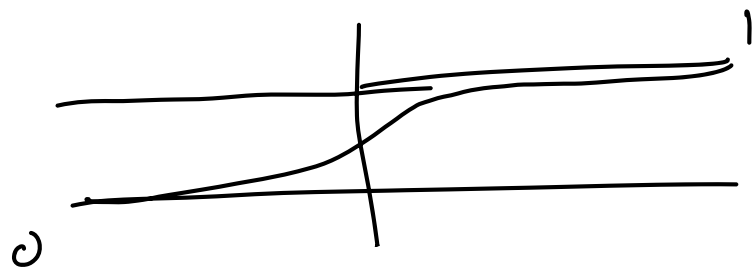
For example, functions given by a feed forward neural network

$$f_\theta(x) = \sum_{k=1}^{N} c_k \phi(a_k \cdot x + b_k)$$

$$\theta = (a_1, b_1, c_1, \ldots, a_N, b_N, c_N) \in \mathbb{R}^{(2d+1)N}$$

The vector θ is called the weights of the network.
Here, ϕ is the logit function

$$\phi(t) = \frac{1}{1 + e^{-t}}$$



Supervised Learning

Then, to find our function, we look for a θ such that the resulting function f_θ makes as small as possible an error given our training data. For example,

$$\text{Minimize}_\theta \;\; E(\theta) := \frac{1}{M} \sum_{k=1}^{M} |f_\theta(x_k) - y_k|^2$$

After finding the minimizer $\theta_{\min}$, if our modeling aassumptions are correct, we can use $f_{\theta_{\min}}$ to estimate if a new given photo x is a photo of a dog or not.

Key points

- Many problems in science and technology amount to
  'Find a function f that does X'
- This f is typically a function of a continuous variable.
- Functions – at least real valued ones– can be reasonably described by high dimensional vectors.
- Often, what f "does" is minimizing a certain functional.
- Once we approximate functions with vectors, the approximation we seek is found by solving a corresponding approximate problem – through and algorithm.
- It is **extremely rare** for any of these problems to have a reasonably useful analytical expression.