# Syllabus

# Numerical Linear Algebra. MATH 5374 (Fall 2022)

## Instructor: Nestor Guillen (nestor@txtstate.edu)

## Course overview

Welcome to Numerical Linear Algebra!. This course deals with numerical algorithms used to solve linear problems. Topics include condition numbers and the numerical stability of linear algorithms; matrix factorizations like the Singular Value Decomposition; the least squares method; iterative algorithms like the conjugate gradient method; and more. We will illustrate such topics by bringing examples of linear problems that arise in applications. The theory will be balanced with exercises involving the implementation of algorithms in Python.

**Lectures time and location**. Class meets Tuesdays and Thursdays from 9:30 to 10:50 am in Derrick 233.

**Office Hours**. Tuesdays and Thursdays from 11 am to noon, or by appointment (request via email at least one day in advance).

**Forums**. Apart from lectures and office hours, an important point of interaction for the class will be **the Canvas Discussions page**. If you have a question related to a particular problem set, or a question about a method discussed in a lecture, this should be the first place to share that question. Sometimes I will provide answers to forum questions, or use a question to explain a broader point or a method. Experience with public forums like Stack Exchange and MathOverflow suggests this can be a great learning tool in their own right.

**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 'MATH5374', so that I can reply promptly.**

**Prerequisites**. Linear algebra at the level of Math 3377. Basic programming skills, as well as familiarity with real analysis are helpful additions but not strictly necessary for the course.

**Course materials**. We will follow the linear algebra chapters of Numerical Algorithms by Solomon and parts of Numerical Linear Algebra by Trefethen and Bau.

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. A helpful resource on Python is the SciPy Lectures.

## Course evaluation

**Overview**: There will be problem sets due approximately every two weeks (about 6 total), a midterm, and final exam. These are weighted as follows

Problem sets: 50%

Midterm: 25%

Final: 25%

Bonus points: 0.5% extra point per bonus point earned.

**Your weekly work flow:** Apart from watching/attending the week's lectures, you are expected to check the course homepage and the Canvas Discussion page regularly. Announcements will be made via email and through the Announcement function on Canvas. I may from time to time highlight any particularly helpful forum discussions. Problem sets will be due **on Fridays at 11:59 pm** and submitted on Canvas.

**Problem sets**: These amount to 50% of the final numerical grade and will be due approximately every two weeks. Your lowest problem set grade will be dropped when computing your problem sets average. **Late problem sets will not be accepted save for exceptional circumstances.** If you don't submit a problem set on time you will be given a zero on that problem set. On the other hand you could miss one problem set and still be able to get a perfect final grade.

**Grading of Problem Sets**: You will receive graded problem sets/exams a week after they were submitted. I will not be posting solutions to **all** problem or exams but will aim to post solutions for about half of the problems. You are encouraged to stop by office hours to discuss the solution of any given problem or to discuss in the Forum.

**Bonus points**: Bonus points can be obtained by solving more difficult (optional) problems, such problems will be present in several weekly problems sets and will be denoted with an asterisk: *. In a few instances **additional Bonus Problems will be posted in the Course Forum**. To illustrate the role of Bonus Points, if your final numerical grade (before Bonus Points) is 83% and you earned credit for 14 bonus problems throughout the semester, you would get 14 Bonus Points = 7% points so your final numerical grade will be 90% (corresponding to A). Bonus problems will be more challenging and you will only earn credit for them if the solution is entirely correct (that is, you cannot earn partial Bonus Points).

**Submission guidelines**: Your submitted problem sets and exams must be submitted in PDF format. Files should be named as: YourLastName_ProbSetN.pdf, 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 problem set due: Friday, September 2nd
- Midterm: Thursday, October 27th
- Final: Tuesday, December 6th at 8 am

**Final Course Grade**: Your final numerical grade is placed on a scale of 100. Your final letter grade will be computed from your final numerical grade as follows

A 90-100 • B 80-89.9 • C 70-79.9 • D 60-69.9 • F 0-59.9

# Course topics (tentative)

- Basics about scientific computing and numerical analysis
- Norms, of all kinds, and the notion of a condition number
- Floating point representation - a quick glance and some warnings.
- Examples of linear problems: regression, high dimensional statistics, basis functions, least squares, finite elements, Tikhonov regularization, image matching, deconvolution, network analysis
- Linear algebra basics review — vector spaces, bases, linear transformations, kernels, inner products
- Numerically solving a linear system, the basics!: Never invert A. Backsubstitution and Gauss-Jordan elimination. LU decomposition

- Algorithmic complexity
- Dense matrices, sparse matrices, positive definite matrices
- Cholesky and eigenvector factorizations for positive definite matrices
- Orthogonality: the QR decomposition and the Householder algorithm
- Eigenvectors, Krylov subspace methods
- Iterative methods: splitting, gradient descent, Conjugate gradient descent
- The SVD and low rank approximation

# Numerical Linear Algebra (Fall 2022)

## Lecture 1

At the simplest level, this class will be about solving

$$Ax = b$$

where $A, b$ are given ($n \times m$ matrix and $m$-dimensional vector)

Now this simple is a bit misleading, it undersells the power lying behind this equation. Let's look at some examples

## Examples of linear problem

### 1. Regression and/or interpolation

Suppose you want to determine / estimate / learn a function $f(x)$ and all you have available is a finite but potentially large data set of examples, i.e. pairs $(x_1, y_1), \ldots, (x_m, y_m)$ such that

$$y_k = f(x_k) \qquad k = 1, \ldots, m$$

Problem: From this data, determine $f(x)$

(of course, this problem has infinitely many solutions)

One version of this problem that is "less" ill-posed involves adding a modeling assumption: that $f$ is given by a linear combination of some predetermined family of functions: $f_1, f_2, \ldots, f_n$

So in this case we are looking for $n$ coefficients $c_1, c_2, \ldots, c_n$ such that the function

$$f(x) = c_1 f_1(x) + \cdots + c_n f_n(x)$$

solves $f(x_k) = y_k$ for $k = 1, \ldots, m$.

Now we have reduced our problem to finding $n$ numbers $c_1, c_2, \ldots, c_n$ such that the

following $m$ equation are satisfied:

$$\begin{cases} c_1 f_1(x_1) + c_2 f_2(x_1) + \cdots + c_n f_n(x_1) = y_1 \\ c_1 f_1(x_2) + c_2 f_2(x_2) + \cdots + c_n f_n(x_2) = y_2 \\ \vdots \\ c_1 f_1(x_m) + c_2 f_2(x_m) + \cdots + c_n f_n(x_m) = y_m \end{cases}$$

This is, in term of the $c$'s, a linear system of equations with $n$ unknowns and $m$ equations.

If we set $\overline{C} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$, $\overline{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}$

$$A = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_n(x_2) \\ \vdots & & & \\ f_1(x_m) & f_2(x_m) & \cdots & f_n(x_m) \end{pmatrix}$$

Then what we want to solve is

$$A\bar{c} = \bar{y}$$

## 2. Discretization of Partial Differential Equations

A fundamental equation in science is the Poisson equation and its associated "Dirichlet problem" which asks for the following:

We are given the following data:

* A domain $D \subset \mathbb{R}^d$ $(d = 2, 3)$
* Two functions, $f : D \longrightarrow \mathbb{R}$,
  $g : \partial D \longrightarrow \mathbb{R}$

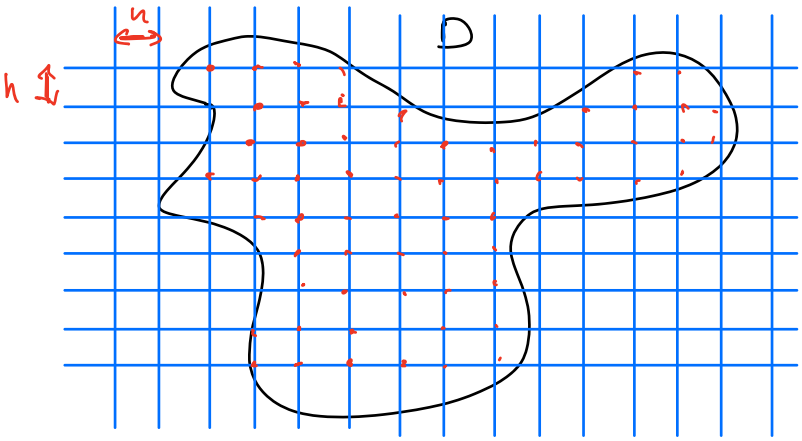
$D$ is assumed bounded with a smooth boundary, and $f, g$ are assumed to be smooth.

Problem: Find a function $u : D \longrightarrow \mathbb{R}$ such that

$$\begin{cases} \Delta u = f & \text{in } D \\ u = g & \text{on } \partial D \end{cases} \quad \left( \begin{array}{c} \text{Dirichlet} \\ \text{Problem} \end{array} \right)$$

(Here, $\Delta u = \partial_{x_1}^2 u + \cdots + \partial_{x_d}^2 u$ )

One way to approach this problem is through a discretization scheme; one such scheme being as follows:




Fix $h>0$ (small) and consider

$$G := D \cap h\mathbb{Z}^d$$

$$\partial G = \{ x \in G \mid x \text{ has a neighbor in } h\mathbb{Z}^d \text{ that's outside of } D \}$$

Henceforth, $d=2$ in this example

Now I am going to write down a discrete counterpart to the Dirichlet problem above

Problem: Find a function

$$u_h : G \longrightarrow \mathbb{R}$$

such that

(Discrete Dirichlet Problem)

$$\begin{cases} \Delta_h u_h(x) = f(x) & \text{if } x \in G \setminus \partial G \\ u_h(x) = g(x^*) & \text{if } x \in \partial G \end{cases}$$

where

$$\Delta_h V(x) = \frac{1}{4h^2} \sum_{\substack{y\text{'s neighbors} \\ \text{of } x}} (V(y) - V(x))$$

and $x^* =$ closest point to $x$ on $\partial D$.

This last problem approximates the original Dirichlet problem as $h \to 0$, and becomes a linear algebra problem.

How? Label the points in $G$ as

$$x_1, x_2, \ldots, x_n$$

where the last $n-k$ points lie in $\partial G$

We are looking for a vector $u_h \in \mathbb{R}^n$ representing the value of the solution function $u_h : G \to \mathbb{R}$, in the sense that

$$u_h = \begin{pmatrix} u_h(x_1) \\ u_h(x_2) \\ \vdots \\ u_h(x_n) \end{pmatrix}$$

The Discrete Dirichlet Problem is equivalent to solving a system of the form

$$A u_h = b$$

where $b = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_k) \\ g(x_{k+1}) \\ \vdots \\ g(x_n) \end{pmatrix} \in \mathbb{R}^n$

and $A$ is a $n \times n$ matrix where first $k$ rows encode the data of neighbors in $G$, i.e.

$$A_{ij} = \begin{cases} \frac{1}{4h^2} & \text{if } x_i \text{ and } x_j \text{ are neighbors, and } i \le k \\ -\frac{1}{h^2} & \text{if } i = j, \ i \le k \end{cases}$$

$$A_{ij} = \begin{cases} 0 & \text{if } i \neq j, \ i > k \\ 1 & \text{if } i = j, \ i > k \end{cases}$$

If your domain $D$ in $\mathbb{R}^2$ has area of about 1, and $h \approx 10^{-3}$, then $G$ will have about $10^6$ gridpoints, and $A$ will be roughly a $10^6 \times 10^6$ matrix.

Other examples we will discuss throughout the semester

2. Image processing algorithm.
3. Signal compression algorithm
4. Analysis of networks (like social networks)

5. Usage in nonlinear algorithms.

---

## The Scientific Computing Philosophy

"Numerical analysis is the study of algorithms for the problem of continuum mathematics"

— Nick Trefethen

---

## Preview of next class:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

$$A^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

Cramer's rule

$$A^{-1} = \frac{1}{\det A} \overbrace{\begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & & \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}}^{\text{Cofactor matrix}}$$

Computing one $C_{ij} \approx (n-1)!(n-1) \approx n!$
and there are $n^2$ of them

If $n=20$, this is about $10^{20}$ arithmetic operations

$n=30$, more than $10^{30}$ operations.