MATH089
FYS: Literate Scientific Computing
Syllabus

Summary. A project-based introduction to scientific computing that in addition to covering foundational mathematical concepts, presents the Julia language as an introductory programming environment and uses the TEXMACS scientific editing platform to refine the concept of literate programming to the requirements of reproducible computational research.

Times	MWF 9:05-9:55AM, Phillips 224 with Zoom synchronous meeting.
Office hours	Tu 4:00-4:30PM, We 3:00PM-4:00PM Chapman 451, and by email appointment, Zoom
Instructor	Sorin Mitran
Jump to	Policies: 3.2 Lessons: 4.2 Software: 5.2 Live documents: 5.4

(*The instructor reserves the right to make changes to the syllabus. Any changes will be announced as early as possible.*)

1 Course overview

Computational modeling of natural phenomena has become a cornerstone of scientific inquiry, completing the traditional methods of theory construction and experimentation. The distinctive feature of scientific computation is exhaustive testing of our understanding of well-defined theoretical models, to an extent that is not possible without machines to rapidly carry out arithmetic operations. This seminar will introduce students to the art of successful scientific simulation. Simple models from the physical, biological, and social sciences will be introduced, given correct mathematical formulations, implemented in computer code, and analyzed. Concepts from the sciences, mathematics, and programming will be introduced as needed with no formal prerequisites beyond typical high school material. The objective will be to produce "live" computational documents that serve as virtual experiments for some field of scientific inquiry.

Relevance of computational approaches to science requires adoption of the scientific method of verification of the predictions resulting from conjectures (or hypotheses or theories). For scientific computing, the conjectures are the mathematical approach and implementation into a program executed by a computer. Predictions are obtained from program execution and verified by comparison to known results or experiments. Such computational predictions should be reproducible. This course adopts the Julia programming language, a general-purpose language with many features useful for scientific computing, as the environment to introduce the practices of reproducible computational research.

A key part of the scientific method is documentation of an investigation, clearly citing sources, approaches, hypotheses, and results. Several specialized programming languages have been developed for this purpose (TEX, LATEX, Markdown), some with an explicit focus on documenting theoretical approach and computer implementation simultaneously (web), a practice known as literate programming. The project-based approach of this course seeks to instill this practice of schol-

arship into all aspects of scientific computing, defining a literate programming approach based upon the TEXMACS platform in conjunction with Zotero reference management.

2 Course outcomes

Upon successful course completion students will be:

- able to recognize a well-formulated computational model of a natural phenomenon;
- exposed to fundamental mathematical concepts arising in computational modeling including:
  - *difference calculus*, a computational system based on considering small increments of quantities of interest, and that leads to differential calculus when the increments become infinitesimally small
  - *algebraic structures*, a consistent set of rules on manipulation of mathematical objects
  - *linear algebra*, a particular algebraic structure that constructs complex objects by scaling and composition of simpler objects
  - *random numbers* arising from the probability of specific events arising in some phenomenon
  - *stochastic calculus*, a computational system for random variables
  - *graphs and networks* to model phenomena in which component parts have a finite number of interconnections
  - *agent-based modeling* as an introduction to the dichtomy between microscopic and macroscopic modeling;
- introduced to basic programming techniques (e.g., comparable to COMP110 or COMP116) including:
  - data types, variables, constants, definition scope
  - statements, expressions, and operators
  - control flow, conditional, iterative and recursive code
  - data structures, lists, trees, dictionaries
  - functions, arguments, parameters, return types
  - object-oriented programming;
- able to document computational research using rigorous scholarly practices.

3 Course information

3.1 Honor code

Unless explicitly stated otherwise, all work is individual. You may discuss various approaches to homework problems with students, instructors, but must draft your answers by yourself. All external sources consulted must be acknowledged and cited. Students implicitly accept this honor code by submission of any work for grading.

3.2 Course policies

- Class attendance is expected and highly beneficial to understanding of course topics. There is no need to inform instructor of planned absences.
- Course grade is based upon accumulation of credit points (0-100). There is no "grading on a curve", but 112 total points are possible, enabling some margin of error in the coursework.
- Homework is to be submitted in typeset form as a $\TeX_{\text{MACS}}$ document through Sakai. Late homework is accepted only in the case of University approved class absences. E-mail messages requesting acceptance of late homework due to any other circumstance are deleted without review or response. Students are advised to prepare and submit homework well in advance of the Sakai deadline to allow for unforseen difficulties. Suspension of classes due to campus-wide events (weather, pandemic, etc.) will lead to modification of due dates or elimination of specific assignments for the entire class.

Accessibility resources and services. The University of North Carolina at Chapel Hill facilitates the implementation of reasonable accommodations, including resources and services, for students with disabilities, chronic medical conditions, a temporary disability or pregnancy complications resulting in barriers to fully accessing University courses, programs and activities.

Accommodations are determined through the Office of Accessibility Resources and Service (ARS) for individuals with documented qualifying disabilities in accordance with applicable state and federal laws. See the ARS Website for contact information: https://ars.unc.edu or email ars@unc.edu.

Counseling and psychological services (CAPS). CAPS is strongly committed to addressing the mental health needs of a diverse student body through timely access to consultation and connection to clinically appropriate services, whether for short or long-term needs. Go to their website: https://caps.unc.edu/ or visit their facilities on the third floor of the Campus Health Services building for a walk-in evaluation to learn more.

Title IX resources. Any student who is impacted by discrimination, harassment, interpersonal (relationship) violence, sexual violence, sexual exploitation, or stalking is encouraged to seek resources on campus or in the community. Reports can be made online to the EOC at https://eoc.unc.edu/report-an-incident/. Please contact the University's Title IX Coordinator (Elizabeth Hall, interim – titleixcoordinator@unc.edu), Report and Response Coordinators in the Equal Opportunity and Compliance Office (reportandresponse@unc.edu), Counseling and Psychological Services (confidential), or the Gender Violence Services Coordinators (gvsc@unc.edu; confidential) to discuss your specific needs. Additional resources are available at safe.unc.edu.

3.3 Grading

Coursework is centered around the development of seven computational projects, with 16 points possible in each project for a total of 112 offered course credit points.

Mapping of point scores to letter grades

Grade	Points	Grade	Points	Grade	Points	Grade	Points
A+	101-112	B+	86-90	C+	71-75	D+	56-60
A	96-100	B	81-85	C	66-70	D	50-55
A-	91-95	B--	76-80	C-	61-65	F	0-49

4 Lesson plan

4.1 Course projects

The following seven projects will be developed during the course at a pace of approximately two weeks per project:

1. *Population models*. P01
2. *Crystals, quasicrystals, and tilings*.
3. *Image processing*.
4. *Random walks and diffusion in physics and finance*.
5. *Monte Carlo, legislative redistricting*.
6. *Graph Laplacian and spectral clustering in machine learning*.
7. *Flocking, schooling, and folding*.

Each project introduces:

- theoretical background drawn from the appropriate field of science;
- mathematical concepts suitable for phenomenon description;
- programming approaches to implement mathematical concepts;
- representative scientific literature for the problem.

4.2 Lesson schedule

The course workcycle consists of five lessons to present theoretical material and computational techniques, followed by an in-class session to address final questions on project preparation. The project is due at 11:55PM on day of following class meeting. Due dates are marked in **bold**. A new project is started on the project due dates.

Week	Notes	Date	Topic
01	L01	08/18	Introduction to Julia and TeXmacs
	L02	08/20	Exponential, logistic population growth
	L03	08/23	Difference equations and solutions
02	L04	08/25	Interacting populations: predator-prey
	L05	08/27	Population states: Susceptible, Infectious, Recovered
	L06	08/30	Differential equation models, correspondence principle
03		09/01	In-class completion of Project 1
	L07	**09/03**	Periodic structures, crystals, mathematical groups
	L08	09/08	Tilings, mathematical groupoids
04	L09	09/10	Crystals and quasicrystals in nature
	L10	09/13	Tilings in art
	L11	09/15	Diffraction, projection, visualization
05		09/17	In-class completion of Project 2
	L12	**09/20**	Representation of visual information
	L13	09/22	Vectors and matrices
06	L14	09/24	Matrix factorization, singular value decomposition
	L15	09/27	Image transformations
	L16	09/29	Image compression and reconstruction
07		10/01	In-class completion of Project 3
	L17	**10/04**	Probability and set theory
	L18	10/06	Random walks
08	L19	10/08	Probability distributions
	L20	10/11	Scaling, diffusion
09	L21	10/13	Financial market modeling
		10/15	In-class completion of Project 4
10	L22	**10/18**	Deterministic versus random modeling
	L23	10/20	Summation of many random events, Monte Carlo integration
	L24	10/22	Stochastic modeling
11	L25	10/25	Packing within physical systems
	L26	10/27	Legislative district allocation
		10/29	In-class completion of Project 5
12	L27	**11/01**	Discrete versus continuum models
	L28	11/03	Graphs and networks
	L29	11/05	Discrete and continuum Laplace operator
13	L30	11/08	Machine learning and clustering
	L31	11/10	Spectral clustering
		11/12	In-class completion of Project 6
14	L32	**11/15**	Interacting system components: nucleons, particles, atoms, molecules
	L32	11/17	Interacting system components: agents
	L34	11/19	Clustering behavior: flocks, schools
15	L35	11/22	Correlated motion: protein folding
	L36	11/29	Stochastic interactions: Langevin models
		12/01	In-class completion of Project 7

5 Computational resources

5.1 Hardware

Students are required to have a laptop, that conforms to CCI minimal standards, and is brought to every class session.

5.2 Software

Scientific computation benefits from freely available software of high quality. One goal of the course is to familiarize students with these capabilities and acquire the practical skills needed for scientific computing. Students are asked to carry out the following steps to install course software.

5.2.1 Linux virtual machine

Students with laptops that use x86-64 architecture, and interested in gaining familiarity with the Linux operating system, widely used in scientific computing, especially in high-performance computing, can install the SciComp@UNC environment. All course tools have been preconfigured for immediate use. Follow instructions at SciComp@UNC to install on a laptop with at least 24GB free disk space and 8GB RAM.

5.2.2 Windows

1. Create a directory named `C:\courses`

2. Install TortoiseSVN

3. Open File Explorer and right-click to open options ("See more options" in Windows 11) for folder `C:\courses`. Select SVN checkout option and enter:

   URL repository: `svn://mitran-lab.amath.unc.edu/courses/MATH089`

   Checkout directory: `C:\courses\MATH089`

   Click OK, and a copy of the course material repository is downloaded to your laptop.

4. Julia programming language. Choose installation directory `C:\courses\julia`

   Modify the System variable PATH to include `C:\courses\julia\bin`

5. TeXmacs editing platform. Choose installation directory `C:\courses\texmacs`

6. Open File Explorer and right-click to open options ("See more options in Windows 11") for folder `C:\courses\texmacs\plugins`. Select SVN checkout option and enter:

URL repository:

`svn://mitran-lab.amath.unc.edu/courses/texmacs/plugins/julia`

Checkout directory: `C:\courses\texmacs\plugins\julia`

5.2.3 macOS

1. Open the Terminal app and create a directory named `~/courses`

   ```
   cd ~; mkdir courses
   ```

2. Install SmartSVN

3. Open SmartSVN and select option Check out project from repository, click OK. Enter:

   Repository: `svn://mitran-lab.amath.unc.edu/courses/MATH089`, select MATH089 directory

   Local directory: `~/courses/MATH089`

   Click Continue, and a copy of the course material repository is downloaded to your laptop.

4. Julia programming language. Choose installation directory `C:\courses\julia`

   Modify the System variable PATH to include `/Applications/Julia-1.6.app/Contents/Resources/julia/bin`

5. TEXMACS editing platform.

6. Open SmartSVN and select option Check out project from repository, click OK. Enter:

   Repository:

   `svn://mitran-lab.amath.unc.edu/courses/texmacs/plugins/julia`

   Checkout directory: `/Applications/TeXmacs.app/Contents/Resources/share/TeXmacs/plugins`

5.3 Tutorials

Software usage is introduced gradually in each class, so the first resource students should use is careful, active reading of the material posted in class. In particular, carry out small tasks until it becomes clear what the software commands accomplish. Some additional resources:

- TeXmacs:
  - http://www.texmacs.org/tmweb/help/tutorial.en.html

  - https://www.youtube.com/watch?v=mlcqGRv7xhc

- Julia:
  - https://julialang.org/learning/
  - https://www.youtube.com/user/JuliaLanguage/playlists

5.4 Interactive documents

All course material is presented as TEXMACS documents with embedded interactive Julia sessions. Such documents have a `.tm` extension and are available through svn download from the course repository. Notes posted on the lesson plan contain translations of the live documents to `.pdf` or `.html` formats.

Live documents allow immediate application of course topics, as shown here to construct the Fibonacci numbers, defined as $F_0 = 0$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$ for $n > 1$ a natural number, and an early model (1202) for rabbit population growth. This is an example of recursion, a function that calls itself, a useful programming paradigm, though not very efficient in this case.

Algorithm - Fibonacci numbers $F(n)$

Input: n a natural number
if $n = 0$ then return 0
if $n = 1$ then return 1
return $F(n-1) + F(n-2)$

```
Julia (1.6.1) session in GNU TeXmacs
∴ function F(n)
    if (n==0) return 0; end
    if (n==1) return 1; end
    return F(n-1)+F(n-2)
  end;
∴ [F(0) F(1) F(2) F(3) F(4) F(5)]
                [0 1 1 2 3 5]                    (1)
∴ F.(0:5)'
                [0 1 1 2 3 5]                    (2)
∴ F.(0:2:10)'
                [0 1 3 8 21 55]                  (3)
∴ F(25)
75025
∴
```