AI-based Materials and Process Simulation I & II

Course Syllabus — Finite Element Methods with FEniCSx (DOLFINx 0.11) on Google Colab

Kee-Youn Yoo

2026-11-07

Table of contents

1 Course Overview 1
- 1.1 Course Format . . . 2
- 1.2 Working with LLM Coding Assistants . . . 2
- 1.3 Learning Objectives . . . 2
- 1.4 Prerequisites . . . 3

2 Computing Environment — Google Colab + fenicsx-colab 3

3 Course I — Weekly Schedule 3

4 Course II — Weekly Schedule 5

5 Grading (per course) 6

6 Academic Integrity 6

7 Textbooks and References 7

1 Course Overview

This two-semester graduate sequence covers the finite element method (FEM) from its mathematical foundations, through implementation, to applications in solid mechanics, fluid mechanics, and coupled transport problems relevant to materials and process engineering. All computational work is done in **Google Colab with FEniCSx (DOLFINx 0.11)**. No local installation is required.

AI-based refers to how simulation code is written in this course: students are **actively encouraged to use large language model (LLM) coding assistants** (e.g., Gemini in Colab, ChatGPT, Claude) to write, debug, and extend FEniCSx code. The emphasis therefore shifts from typing code to **formulating the problem precisely, directing the assistant, and rigorously verifying what it produces**. The mathematical theory taught in the course is what makes that verification possible: a student who knows the expected convergence rate, the inf-sup requirement, or the analytical benchmark can tell when generated code is wrong.

Course	Focus
AI-based Materials and Process Simulation I	Mathematical foundations (Lax–Milgram, Céa, inf-sup), the internal structure of FEniCSx, nonlinear and time-dependent problems, and incompressible flow (Stokes → Navier–Stokes)
AI-based Materials and Process Simulation II	Computational solid and structural mechanics (20 chapters), coupled heat and mass transfer in flows (natural convection, reactive flow, porous media), and parallel solvers

1.1 Course Format

- Each week covers about two chapters: 3 hours of lecture (1.5 h theory and 1.5 h implementation) plus independent work in Colab.
- Slides and notebooks are released before each lecture. Students are expected to run the notebook once before class.
- A weekly assignment is due before the next lecture. Each lecture opens with a 10-minute quiz on the previous week's material.
- The term project starts in Week 8 with a proposal and runs through the second half of the semester.
- LLM assistants may be used for all coding work. Exams and quizzes are taken without AI tools, so that each student's own understanding of the theory and the code is assessed.

1.2 Working with LLM Coding Assistants

LLMs are fast at producing FEniCSx code, but they are frequently wrong in ways that run without error. Common failure modes include:

- **Outdated API**: code written for legacy FEniCS (`from fenics import *`) or older DOLFINx releases (0.5–0.10) instead of DOLFINx 0.11.
- **Silent mathematical errors**: wrong sign in the weak form, missing boundary terms, unstable element pairs (e.g., P1–P1 for Stokes), incorrect Jacobians, missing stabilization.
- **Plausible but unverified results**: smooth-looking plots that do not satisfy the PDE, the boundary conditions, or conservation laws.

Every LLM-assisted notebook in this course therefore follows the same workflow:

1. **Formulate** — derive the strong and weak forms, function spaces, and boundary conditions by hand *before* prompting.
2. **Prompt with context** — give the assistant the weak form, the target version (DOLFINx 0.11), and the course's API reference sheet.
3. **Read and annotate** — understand every line; mark which parts were generated and which you changed.
4. **Verify** — MMS convergence rates, analytical or benchmark comparison, conservation checks. Unverified code is not a result.
5. **Log** — keep a short AI-usage log (key prompts, what failed, how you fixed it) in the notebook.

1.3 Learning Objectives

By the end of **Course I**, students will be able to:

1. Explain the Ciarlet definition of a finite element, unisolvence, and local-to-global DOF construction, and verify them with Basix.
2. Prove well-posedness and convergence rates using Lax–Milgram, Céa's lemma, and the Aubin–Nitsche argument, and confirm the rates numerically with the method of manufactured solutions (MMS).
3. Solve nonlinear problems with Newton/SNES and time-dependent problems with the θ-method.
4. Explain the inf-sup condition. Solve the Stokes equations with Taylor–Hood elements and the Navier–Stokes equations with a splitting scheme (IPCS) and with Newton's method, and validate against the DFG benchmark.
5. Apply SUPG stabilization to advection-dominated problems and build block (fieldsplit) preconditioners.
6. Use LLM coding assistants effectively to produce FEniCSx code from a mathematical formulation, and detect and correct errors in generated code through theory-based verification.

By the end of **Course II**, students will be able to:

7. Solve plane-stress, axisymmetric, generalized plane-strain, and thermoelastic problems and verify them against analytical solutions.
8. Implement plasticity, viscoelasticity, and phase-field fracture using quadrature-space internal variables and return mapping.
9. Implement beams, plates, and shells; diagnose and cure shear locking; solve dynamics, buckling, homogenization, and cohesive-zone problems.

10. Implement and validate multiphysics flow problems relevant to chemical engineering: Boussinesq natural convection, reactive species transport, and Brinkman flow in porous media.
11. Run problems in parallel with MPI and analyze solver performance with AMG and block preconditioners.
12. Lead an LLM-assisted workflow for a research-scale multiphysics simulation: decompose the problem, direct code generation module by module, and document the verification of each component.

1.4 Prerequisites

- **Course I**: Undergraduate partial differential equations and linear algebra; basic Python.
- **Course II**: Course I (or equivalent FEM experience); a first course in solid mechanics or elasticity; introductory transport phenomena.

---

2 Computing Environment — Google Colab + fenicsx-colab

Every notebook starts with the Quick Start cell from SeoulTechPSE/fenicsx-colab.

```
# First cell of every notebook (summary; see the repository README for the full cell)
REPO_URL = 'https://github.com/seoultechpse/fenicsx-colab.git'
DOLFINX_VERSION = '0.11'   # course default
USE_COMPLEX = False        # set True (with USE_CLEAN=True) only when complex PETSc is required
```

```
%%fenicsx -np 4 --time
from mpi4py import MPI
import dolfinx
print(MPI.COMM_WORLD.rank, dolfinx.__version__)
```

Topic	Guideline
Week 1 (HW0)	Run the Quick Start in Colab and submit a screenshot showing `dolfinx 0.11.x` and 4 MPI ranks
LLM assistants	Any assistant may be used (Gemini in Colab, ChatGPT, Claude, etc.). Always provide the course **DOLFINx 0.11 API reference sheet** as context, and run generated code only inside the Colab environment above
Saving results	Save outputs (`.xdmf`, `.bp`, figures) to Google Drive. Restarting the runtime removes the environment; re-run the Quick Start cell
Parallel runs	A Colab VM has 2 CPUs, so `-np 4` is oversubscribed. Interpret scaling measurements from `-np 1` and `-np 2` only
Heavy problems	3D and long transient runs (DFG unsteady cylinder, phase-field fracture, shells, homogenization) are assigned in reduced 2D/coarse form; full-size runs are optional for Colab Pro users
Complex PETSc	Used only for electromagnetic and frequency-domain examples; switch with `USE_CLEAN=True`
Visualization	matplotlib / off-screen PyVista inside notebooks; ParaView locally for result files

---

3 Course I — Weekly Schedule

Notation: *I-n* = theory chapter *n*; *II-n* = implementation chapter *n* with its notebook; **[D]** = official DOLFINx 0.11 demo; **[T]** = Dokken, *The FEniCSx Tutorial*.

Week	Theory	Implementation / Colab	Assignment (due next week)
1	Course introduction; I-1 Poisson, triangulations, weak form, P1 space	Colab setup; **LLM-assisted coding workflow** (generate a Poisson solver with an LLM, then find and fix its errors); II-1 numerical quadrature (Gauss, Duffy); [D] Poisson	HW0 environment + LLM Poisson exercise; HW1 quadrature accuracy
2	I-2 Ciarlet's element, Vandermonde matrix, unisolvence, Hermite/Argyris	II-2 Basix elements and tabulation; [D] Variants of Lagrange elements	HW2 unisolvence proofs (P1–P3) + basis plots
3	I-2 geometric decomposition, C^0/C^1 continuity, DG spaces	II-3 mesh connectivity; II-4 function spaces and DOF numbering (vector and mixed)	HW3 implement a cell-node map
4	I-3 interpolation operators, H^k norms, averaged Taylor polynomials, scaling	II-5 `fem.Function`, interpolation, L^2/H^1 errors; [D] Interpolation and IO	HW4 interpolation convergence rates
5	I-4 Riesz representation, **Lax–Milgram**, well-posedness of Helmholtz/Poisson	II-6 assembly + PETSc KSP; MMS	HW5 Lax–Milgram proof + MMS
6	I-4 Neumann problems, trace inequality, partial Dirichlet	II-7 Dirichlet BCs; [T] multiple Dirichlet/Neumann/Robin, subdomains; [D] Gmsh	HW6 multi-material and Robin BCs
7	I-5 Sobolev spaces, **Céa's lemma**, elliptic regularity, **Aubin–Nitsche**	[T] computing convergence rates, error control	HW7 theoretical vs. observed rates (report)
8	**Midterm exam** (I-1 to I-5, first 90 min)	Second half: II-8 nonlinear problems — Gâteaux derivative, Newton	**Project proposal**
9	Newton convergence, continuation	II-8 hand-written Newton → `NonlinearProblem`/SNES; [T] nonlinear Poisson, custom Newton solvers	HW8 nonlinear MMS
10	Time discretization (θ-method, stability)	[T] heat equation (Gaussian diffusion, analytical solution); [D] Cahn–Hilliard	HW9 temporal convergence rates
11	I-6 Stokes as a mixed problem, **inf-sup**, Brezzi's theorem, MINI/Fortin	II-9 Taylor–Hood, Poiseuille flow, Stokes MMS	HW10 P1–P1 vs. P2–P1 vs. MINI
12	Block preconditioning for saddle-point problems, Schur complement	[D] Stokes (Taylor–Hood; nested, block, and monolithic solvers); [D] mixed Poisson with block preconditioner (Darcy flow)	HW11 solver iteration/time comparison
13	Advection–diffusion, Péclet number, Galerkin oscillations, **SUPG**	Boundary-layer advection–diffusion with and without SUPG	HW12 Péclet-number sweep
14	Navier–Stokes, splitting schemes (Chorin/**IPCS**), CFL condition	[T] channel flow (Poiseuille verification); [T] **DFG 2D-3 flow past a cylinder** (drag and lift)	HW13 reproduce the DFG benchmark
15	Newton's method for steady Navier–Stokes, Reynolds-number continuation, H(div)-conforming DG	Lid-driven cavity (compare with Ghia et al.); [D] H(div)-DG Navier–Stokes	HW14 cavity at Re = 100/400/1000
16	**Final project presentations** (no final exam)		Final project report (Quarto) + code

Course I project: Choose one of the topics A–F in the project catalog (`AMPS_I/projects/index.html`: PEM fuel-cell cathode, packed-bed runaway, buoyant vertical bed, Nernst–Planck electrochemistry, ignition–extinction by arclength continuation, catalyst-activity inverse problem with adjoints) or propose your own nonlinear or flow problem; derive the weak form, argue well-posedness/stability, implement it in Colab, and verify it by MMS convergence and a benchmark. Examples: power-law (non-Newtonian) channel flow, Darcy flow in porous media, reaction–diffusion Turing patterns, backward-facing step flow.

4 Course II — Weekly Schedule

Chapter names refer to the COMet-FEniCSx modules (Introduction, Linear, Nonlinear, Beams, Plates, Shells, Dynamics, Eigenvalue, Homogenization, Interfaces). **[D]** and **[T]** as in Course I.

Week	Topic	Chapters / Colab	New techniques	Verification
1	Elasticity fundamentals (review) + LLM workflow for multi-file projects	Intro-1 linear elasticity; Intro-2 hyperelasticity; Tips & Tricks; [D] elasticity with algebraic multigrid	SNES, AMG, reaction forces; structuring prompts for larger codes	Beam deflection
2	Linear specializations I	Linear-1 plane stress (orthotropic); Linear-2 axisymmetric	Gmsh API, r-weighted weak form	Lamé solution
3	Linear specializations II + thermoelasticity	Linear-3 generalized plane strain; Linear-4/5 weak and full thermoelastic coupling; [D] static condensation	Mixed (T, u) space, e_0 DOF	Weak vs. full coupling
4	Plasticity	Nonlinear-1 J2 plasticity	Quadrature spaces, radial return, consistent tangent	Limit load
5	Viscoelasticity + fracture	Nonlinear-2 viscoelasticity; Nonlinear-3 phase-field fracture (2D reduced)	Generalized Maxwell, staggered solver	Relaxation solution, load–displacement
6	Dynamics + buckling	Dynamics-1 Newmark; Eigen-1 linear buckling (SLEPc)	Mass lumping, generalized eigenproblem	Energy conservation, Euler load
7	Beams	Beams-1 truss; Beams-2 3D Timoshenko; Beams-3 finite rotations	Manifold meshes, local frames, $R(\theta)$	Helical beam benchmark
8	**Midterm exam** (first half) + plates	Intro-3 Reissner–Mindlin; Plates-1 locking with quadrilaterals	Selective reduced integration, serendipity elements	Thin-plate limit · **Project proposal**
9	Shells + homogenization	Shells-1 linear shell; Homogenization-1 periodic homogenization (`dolfinx_mpc`)	P2/CR mixed space, periodic constraints	Mori–Tanaka
10	Interfaces	Interfaces-1 intrinsic CZM; Interfaces-2 submesh-based CZM	DG + Nitsche, `create_submesh`, block systems	Analytical debonding
11	**Thermal flow: natural convection**	Boussinesq differentially heated cavity (Newton, mixed (u, p, T))	Multiphysics mixed spaces, Rayleigh-number continuation	de Vahl Davis Nusselt numbers
12	**Mass transfer: reactive flow**	Tubular reactor — Navier–Stokes velocity + species advection–diffusion–reaction (SUPG), 1st/2nd-order kinetics	One-way coupling, nonlinear reaction terms	Plug-flow / Graetz solutions
13	**Porous media + conjugate heat transfer**	Brinkman / Darcy–Stokes (packed bed); solid–fluid conjugate heat transfer (submesh, reusing Week 10)	Domain coupling, interface continuity	Analytical solutions, energy balance
14	**Phase separation + parallel performance**	[D] Cahn–Hilliard extension (binary phase separation); [D] matrix-free solvers; parallel communication analysis	Solver configuration, MPI scaling	Iteration/time analysis report

Week	Topic	Chapters / Colab	New techniques	Verification
15	Project workshop	Individual code reviews; 5-minute progress presentations		
16	**Final project presentations** + course synthesis			Final report (Quarto)

Course II project: Choose one of the topics A–J in the project catalog (`AMPS_II/projects/index.html`), each taken from a current research field of chemical and biomolecular engineering: printed hydrogel scaffold, supercritical CO_2 extraction, catalyst pellet and monolith, compressed flow-battery electrode, diffusion-induced stress in battery particles, film stress and wrinkling, pneumatic soft actuator, phase separation and coarsening, microfluidic biosensor, laminar polymerization reactor — or propose a solid, fluid, or coupled problem from your own research; build the model, verify it by an exact or manufactured solution and a topic-specific physical check, and present it in Week 15 (progress) and Week 16 (final).

---

5 Grading (per course)

Component	Weight	Notes
Weekly assignments (Course I: 15, HW0–HW14; Course II: 13, HW1–HW13)	30%	Executed Colab notebook + short summary of results + AI-usage log; lowest score dropped. Graded on formulation and verification, not on who typed the code
Weekly quizzes	10%	Previous week's material, 10 minutes
Midterm exam	15%	Course I: proofs / Course II: concepts + code reading
Term project	40%	Course I: proposal 5 + final report and code 25 + final presentation 10 (no progress presentation, no final exam). Course II: proposal 5 + progress presentation 5 (Week 15) + final report and code 20 + final presentation 10. The report includes a section on the LLM workflow: what was generated, which errors were found, and how each component was verified. The presentation includes live questions on the code
Participation	5%	Code reviews, discussion

- **Late policy**: Assignments are due before the next lecture; late submissions are not accepted.
- **Verification**: Every reported number must come from a Colab notebook the student actually ran, and must be verified against an analytical solution, a benchmark, or a mesh-convergence study.
- **Reproducibility**: Submitted notebooks must run end to end after "Restart runtime → Run all". Notebooks that do not run receive zero credit.

6 Academic Integrity

- **LLM assistants are encouraged** for all coding work in assignments and projects. Their use must be recorded in the AI-usage log, but it is never penalized.
- **The student is the author of record.** You are responsible for every line you submit and every number you report, whoever or whatever generated it. Being unable to explain your own submitted code during a quiz, code review, or presentation is treated the same as not having done the work.
- **Exams and quizzes are closed-AI.**
- Discussing theory, mechanics, prompting strategies, and debugging with classmates is encouraged; submitting another student's code or results as your own is not.
- Fabricated or unverified results (e.g., numbers copied from an LLM response without running the code) are treated as academic misconduct.

7 Textbooks and References

- *Finite Elements: Analysis and Implementation* (course lecture notes, PDF)
- J. Bleyer, *Numerical Tours of Computational Mechanics with FEniCSx* — https://bleyerj.github.io/comet-fenicsx
- J. S. Dokken, *The FEniCSx Tutorial* — https://jsdokken.com/dolfinx-tutorial/stable/index.html
- DOLFINx 0.11 demos — https://docs.fenicsproject.org/dolfinx/v0.11.0.post0/python/demos.html
- Colab environment — https://github.com/SeoulTechPSE/fenicsx-colab
- Supplementary: Brenner & Scott, *The Mathematical Theory of Finite Element Methods*; Boffi, Brezzi & Fortin, *Mixed Finite Element Methods and Applications*; Elman, Silvester & Wathen, *Finite Elements and Fast Iterative Solvers*; Donea & Huerta, *Finite Element Methods for Flow Problems*
- Flow benchmarks: Schäfer & Turek (1996), DFG cylinder; Ghia, Ghia & Shin (1982), lid-driven cavity; de Vahl Davis (1983), natural convection