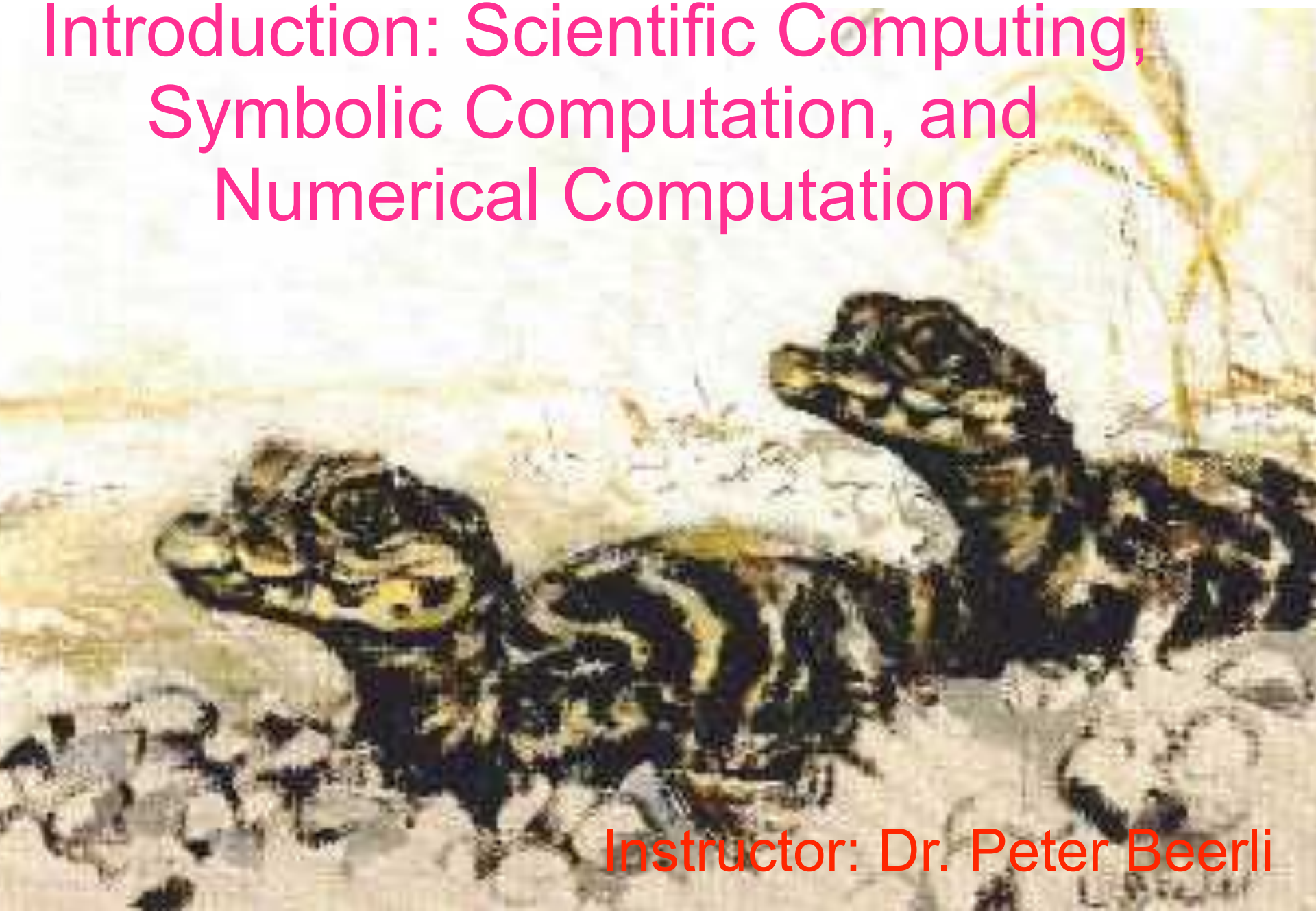
Introduction: Scientific Computing, Symbolic Computation, and Numerical Computation



Instructor: Dr. Peter Beerli

Instructor: Peter Beerli

Education

- Masters in Zoology, 1986 University of Zurich, Switzerland
- Ph.D. Zoology, 1994, University of Zurich, Switzerland

Experience

- 1994-2003, Postdoc, Research Associate, Research Assistant Professor, University of Washington, Seattle WA
- 2003-present, Assistant/Associate/Full Professor, FSU, Tallahassee, FL

Scientific Computing: From Wiki

- **Computational science** (or **scientific computing**) is the field of study concerned with constructing mathematical models and quantitative analysis techniques and using computers to analyze and solve scientific problems.
- In practical use, it is typically the application of computer simulation and other forms of computation to problems in various scientific disciplines.
- The field is distinct from computer science (the study of computation, computers and information processing).
- It is also different from theory and experiment which are the traditional forms of science and engineering. The scientific computing approach is to gain understanding, mainly through the analysis of mathematical models implemented on computers.
- Scientists and engineers develop computer programs, application software, that model systems being studied and run these programs with various sets of input parameters. Typically, these models require massive amounts of calculations (usually floating-point) and are often executed on supercomputers or distributed computing platforms.

CSE: Computational Science and Engineering

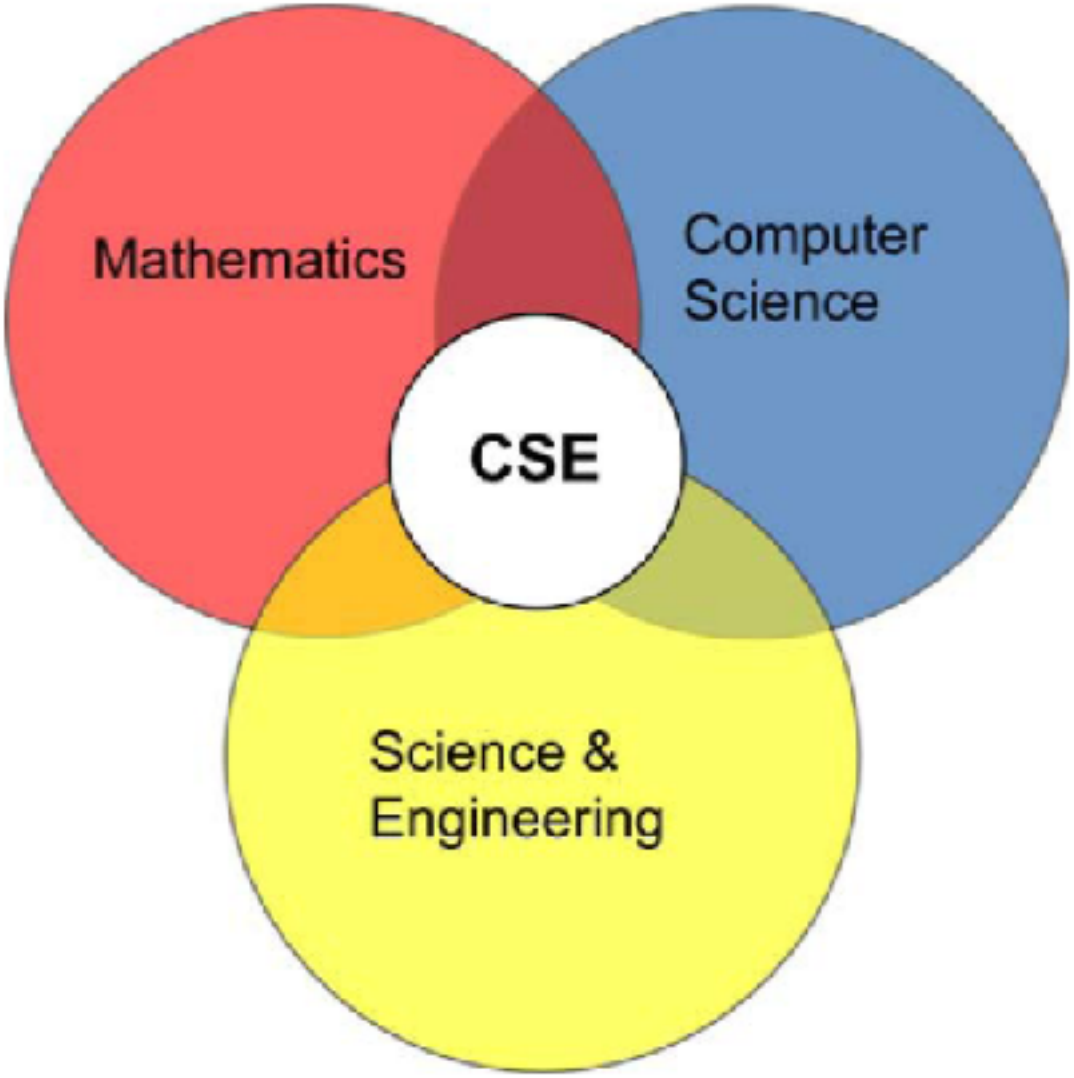


Figure 1: CSE includes, *but is greater than*, the intersection of mathematics, computer science and science & engineering.

- Computational science and engineering (CSE) is a rapidly growing multidisciplinary area with connections to the sciences, engineering, mathematics and computer science.
- CSE focuses on the development of problem solving methodologies and robust tools for the solution of scientific and engineering problems.
- We believe that CSE will play an important if not dominating role for the future of the scientific discovery process and engineering design.
- Computational science should be treated as a “discipline” in its own right – it has its own identity that should be recognized and nurtured as such.

Scientific Computing: An Introductory Survey

- Scientific computing
- System of linear equations
- Linear least squares
- Eigenvalue problems
- Nonlinear equations
- Optimization
- Interpolation
- Numerical integration and differentiation
- Initial value problems for ODEs
- Boundary value problems for ODEs
- Partial differential equations
- Fourier transforms
- Random numbers of simulation

Numerical Calculation vs. Symbolic Calculation

- Numerical calculation: involves number directly
  - manipulate *numbers* to produce a numerical result
- Symbolic calculation: symbols represent numbers
  - manipulate *symbols* according to mathematical rules to produce a symbolic result

Example (numerical)

$$\frac{(17.36)^2 - 1}{17.36 + 1} = 16.36$$

Example (symbolic)

$$\frac{x^2 - 1}{x + 1} = x \quad 1$$

Analytical Solution vs. Numerical Solution

- Analytical solution (a.k.a symbolic): the exact numerical or symbolic representation of the solution
  - may use special characters
- Numerical solution: the computational representation of the solution
  - Entirely numerical
  - An important aspect of numerical computation is the use of approximation instead of exact numerical values.

Example (analytic)

$\frac{1}{4}$

$\frac{1}{3}$

π

$\tan(83)$

Example (numerical)

0.25

0.33333...(?)

3.14159...(?)

0.88472...(?)

Symbolic and Numerical Calculations with Rational Numbers

- One of the essential features of a numerical calculation is that exact symbolic quantities are represented by approximate numerical values.
- Numerical approximation is needed to carry out the steps in the numerical calculation. The overall process is a numerical computation.

Example (symbolic computation, numerical solution)

$$\frac{1}{2}+\frac{1}{3}+\frac{1}{4}-1=\frac{1}{12}=0.083333333\ldots$$

Example (numerical computation, numerical approximation)

0.500 | 0.333 | 0.250 1.000 – 0.083

Numerical and Symbolic Computations

- Numerical computations are performed with programming language such as FORTRAN, Basic, C, C++, and Java. MATLAB is used in this course. MATLAB also supports symbolic computation through the Symbolic Mathematics toolbox. (Octave is the free clone of MATLAB)
- Symbolic computations are usually performed by computer programs such as Derive, Macsyma, Maple, Mathematica, and SAGE. We will use SAGE in this course.

Numerical Method vs. Algorithm

- Numerical Method: a mathematical description of the calculation to be performed; a general description of a process
- Algorithm: a precise sequence of actions taken to obtain a desired result; a detailed description of the method
- A method may be implemented with different algorithms, and one algorithm may be better than another.
- Implementation: a particular instantiation of the algorithm

- Goal: Make a pot of coffee
- Method: Use drip coffeemaker (versus percolator or espresso machine)
- Two algorithms

Algorithm 1

1. Grind fresh beans
2. Place new filter in filter holder
3. Measure 4 tablespoons into the filter
4. Wash and rinse the coffeepot
5. Fill coffeemaker with 4 cups of water
6. Start the machine

Algorithm 2

1. Take coffee from the can
2. Place new filter in filter holder
3. Pour coffee into filter
4. Empty coffeepot
5. Fill coffeemaker with water
6. Start the machine

Which algorithm is better?

Your example?

How Will I Teach/You Learn?

- Repeat
  - ➢Lectures (mathematics and physics behind mathematics).
  - ➢Science and engineering examples to demonstrate how the methods are used in practice.
  - ➢Homework and lab assignments to solve simple examples
- Test
  - ➢Class questions
  - ➢Quizzes
  - ➢Midterm exam
  - ➢Final project
- Ask questions and learn
- Frustration in programming

RECOMMENDATIONS TO SUCCESFULLY FAIL IN CFD (COMPUTATIONAL FLUID DYNAMICS)

Sometimes you cannot avoid being successful. But in CFD, if you are careful enough, you can make sure to fail totally. Not only for you but also for people working with you. This small text provides rules to make sure that anyone given a large CFD project will be reasonably sure to fail...

Rule 1:

Always change two things at the same time in your code. If you need to change something in the numerical method and some sub model, it is absolutely necessary to do the two changes at the same time. This way, when the code will fail, you will be unable to say why.

Rule 3:

Debugging is what CFD is about. 5 minutes to modify your code, 5 months to find why it does not work anymore. The very important thing is here: 'NEVER come back to a previous version!'. The reason why the code does not work now is NOT the lines you changed.... it is the lines you haven't changed yet