Computational Physics at the Physics and Astronomy Department, Michigan State University

Danny Caballero, Sean Couch, Wade Fisher, Connor Glosser, Morten Hjorth-Jensen, Claire Kopenhafer, Brian O'Shea, and Carlo Piermarocchi

Mar 26, 2017

Introduction: Scientific and educational motivation

Numerical simulations of various systems in science are central to our basic understanding of nature and technology. The increase in computational power, improved algorithms for solving problems in science as well as access to high-performance facilities, allow researchers to study complicated systems across many length and energy scales. Applications span from studying quantum physical systems in nanotechnology and the characteristics of new materials or subatomic physics at its smallest length scale, to simulating galaxies and the evolution of the universe. In between, simulations are key to understanding cancer treatment and how the brain works, predicting climate changes and this week's weather, simulating natural disasters, semi-conductor devices, quantum computers, as well as assessing risk in the insurance and financial industry.

Computing competence. Computing means solving scientific problems using computers. It covers numerical as well as symbolic computing. Computing is also about developing an understanding of the scientific process by enhancing algorithmic thinking when solving problems. Computing competence has always been a central part of education in the sciences and engineering disciplines.

On the part of students, this competence involves being able to:

- understand how algorithms are used to solve mathematical problems,
- derive, verify, and implement algorithms,
- understand what can go wrong with algorithms,

- use these algorithms to construct reproducible scientific outcomes and to engage in science in ethical ways, and
- think algorithmically for the purposes of gaining deeper insights about scientific problems.

All these elements are central for maturing and gaining a better understanding of the modern scientific process *per se*.

The power of the scientific method lies in identifying a given problem as a special case of an abstract class of problems, identifying general solution methods for this class of problems, and applying a general method to the specific problem (applying means, in the case of computing, calculations by pen and paper, symbolic computing, or numerical computing by ready-made and/or self-written software). This generic view on problems and methods is particularly important for understanding how to apply available, generic software to solve a particular problem.

Computing competence represents a central element in scientific problem solving, from basic education and research to essentially almost all advanced problems in modern societies. Computing competence is simply central to further progress. It enlarges the body of tools available to students and scientists beyond classical tools and allows for a more generic handling of problems. Focusing on algorithmic aspects results in deeper insights about scientific problems.

Why should basic university education undergo a shift towards modern computing?

- Algorithms involving pen and paper are traditionally aimed at what we often refer to as continuous models.
- Application of computers calls for approximate discrete models.
- Much of the development of methods for continuous models are now being replaced by methods for discrete models in science and industry, simply because much larger classes of problems can be addressed with discrete models, often also by simpler and more generic methodologies.

However, verification of algorithms and understanding their limitations requires much of the classical knowledge about continuous models.

So, why should basic university education undergo a shift towards modern computing?

The impact of the computer on mathematics and science is tremendous: science and industry now rely on solving mathematical problems through computing.

- Computing can increase the relevance in education by solving more realistic problems earlier.
- Computing through programming can be excellent training of creativity.

- Computing can enhance the understanding of abstractions and generalization.
- Computing can decrease the need for special tricks and tedious algebra, and shifts the focus to problem definition, visualization, and "what if" discussions.

The result is a deeper understanding of mathematical modeling and the scientific method (we hope, and here our physics education research group can play a central role in promoting this). Not only is computing via programming a very powerful tool, it can also be a great pedagogical aid.

For the mathematical training, there is one major new component among the arguments above: understanding abstractions and generalization. While many of the classical methods developed for continuous models are specialized for a particular problem or a narrow class of problems, computing-based algorithms are often developed for problems in a generic form and hence applicable to a large problem class.

Computing competence represents a central element in scientific problem solving, from basic education and research to essentially almost all advanced problems in modern societies. Computing competence is simply central to further progress. It enlarges the body of tools available to students and scientists beyond classical tools and allows for a more generic handling of problems. Focusing on algorithmic aspects results in deeper insights about scientific problems.

Moreover, today's projects in science and industry tend to involve larger teams. Tools for reliable collaboration must therefore be mastered (e.g., version control systems, automated computer experiments for reproducibility, software and method documentation).

General learning outcomes for computing competence

Below, we articulate high-level learning outcomes that we expect students to develop through comprehensive and coordinated instruction in numerical methods over the course of their bachelor's program at Michigan State. These learning outcomes are different from specific learning goals in that the former reference the end state that we aim for students to achieve. The latter references the specific knowledge, tools, and practices with which students should engage and discusses how we expect them to participate in that work. We reserve the discussion of specific learning goals to individual course experiences (e.g., Electrostatics).

Learning outcomes for numerical algorithms. Numerical algorithms form the basis for solving science and engineering problems with computers. An understanding of algorithms does not itself serve as an understanding on computing, but it is a necessary step along the path. Through comprehensive and coordinated instruction, we aim for students to have developed:

- A deep understanding of the most fundamental algorithms for linear algebra, ordinary and partial differential equations, optimization, and statistical uncertainty quantification
- A working knowledge of advanced algorithms and how they can be accessed in available software
- A working knowledge of high-performance computing elements including memory usage, vectorized and parallel algorithms
- A deep understanding of approximation errors and how they can present themselves in different problems
- The ability to apply fundamental and advanced algorithms to classical model problems as well as real-world problems as well to assess the uncertainty of their results

Learning outcomes for symbolic computing. Symbolic computing is a helpful tool for addressing certain classes of problems where a functional representation of the solution (or part of the solution) is needed. Through engaging with symbolic computing platforms, we aim for students to have developed:

- A working knowledge of at least one computer algebra system (CAS)
- The ability to apply a CAS to perform classical mathematics including calculus, linear algebra, differential equations
- The ability to verify the results produced by the CAS using some other means

Learning outcomes for programming. Programming is a necessary aspect of learning computing for science and engineering. The specific languages and/or environments that students learn are less important than the nature of that learning (i.e., learning programming for the purposes of solving science problems). By numerically solving science problems, we expect students to have developed:

- An understanding of programming in a high-level language (e.g., MATLAB, Python, R).
- An understanding of programming in a compiled language (e.g., Fortran, C, C++).
- The ability to to implement and apply numerical algorithms in reusable software that acknowledges the generic nature of the mathematical algorithms.

- A working knowledge of basic software engineering elements including functions, classes, modules/libraries, testing procedures and frameworks, scripting for automated and reproducible experiments, documentation tools, and version control systems (e.g., GitHub).
- An understanding of debugging software, e.g., as part of implementing comprehensive tests.

Learning outcomes for mathematical modeling. Preparing a problem to be solved numerically (i.e., modeling) is a critical step in making progress towards an eventual solution. By providing opportunities for students engage in modeling, we aim for them to develop:

- The ability to solve real problems from applied sciences by:
  - Deriving computational models from basic principles in physics and articulating the underlying assumptions in those models,
  - Constructing models with dimensionless forms to reduce and simplify input data, and
  - Interpreting the model's dimensionless parameters to increase their understanding of the model and its predictions

Learning outcomes for verification. Verifying the model and the resulting outcomes it produces are essential elements to generating confidence in the model itself. Moreover, such verifications provide evidence that the work is reproducible. By engaging in verification practices, we aim for for students to develop:

- An understanding of how to program testing procedures
- A deep knowledge of testing/verification methods including the use of:
  - Exact solutions of numerical models
  - Method of manufactured solutions (i.e., choose solution and fit a problem)
  - Classical analytical solutions including asymptotic solutions
  - Computed asymptotic approximation errors (i.e., convergence rates)
  - Step-wise construction of tests to aid debugging.

Learning outcomes for presentation of results. The results of a computation need to be communicated in some format (i.e., through figures, posters, talks, and other forms of written and oral communication). Computation affords the experience of presenting original results quite readily. Through their engagement with presentations for their findings, we aim for students to develop:

- The ability to make use of different visualization techniques for different types of computed data
- The ability to present computed results in scientific reports and oral presentations effectively
- A working knowledge of the norms and practices for scientific presentations in various formats (i.e., figures, posters, talks, and written reports)

Specific algorithms and computational skills

The above learning goals and outcomes are of a more generic character. What follows here are specific algorithms that occur frequently in scientific problems. The implementation of these algorithms in various physics courses, together with problem and project solving, is a way to implement large fractions of the above learning goals. we reserve the coupling of the the broad learning goals above to the algorithms articulated below to our discussion of specific course (or topical) learning goals.

We list also tools that are important in developing numerical projects. These tools allow students to develop a better understanding of the scientific process. In addition, use of the algorithms can facilitate instruction in an ethical approach to science.

The algorithms and tools listed here can be intergated in different ways depending on the specific learning goals for the course or topic. Furthermore, several of these algorithms can be used and, then, revisited in the various courses.

Central algorithms. The following mathematical formulations of problems from the physical sciences play a prominent role and should be reflected in how we teach physics:

- Ordinary differential equations
    1. Euler, modified Euler, Verlet and Runge-Kutta methods with applications to problems in electromagnetism, methods for theoretical physics, quantum mechanics and mechanics.
- Partial differential equations
    1. Diffusion in one and two dimensions (statistical physics), wave equation in one and two dimensions (mechanics, electromagnetism, quantum mechanics, methods for theoretical physics) and Laplace's and Poisson's equations (electromagnetism).

- Numerical integration

    1. Trapezoidal and Simpson's rule and Monte Carlo integration. Applications in statistical physics, methods of theoretical physics, electromagnetism and quantum mechanics.

- Statistical analysis, random numbers, random walks, probability distributions, Monte Carlo integration and Metropolis algorithm. Applications to statistical physics and laboratory courses.

- Linear Algebra and eigenvalue problems.

    1. Gaussian elimination, LU-decomposition, eigenvalue solvers, and iterative methods like Jacobi or Gauss-Seidel for systems of linear equations. Important for several courses, classical mechanics, methods of theoretical physics, electromagnetism and quantum mechanics.

- Signal processing

    1. Discrete (fast) Fourier transforms, Lagrange/spline/Fourier interpolation, numeric convolutions & circulant matrices, filtering. Applications in electromagnetics, quantum mechanics, and experimental physics (data acquisition)

- Root finding techniques, used in methods for theoretical physics, quantum mechanics, electromagnetism and mechanics.

In order to achieve a proper pedagogical introduction of these algorithms, it is important that students and teachers alike see how these algorithms are used to solve a variety of physics problems. The same algorithm, for example the solution of a second-order differential equation, can be used to solve the equations for the classical pendulum in a mechanics course or the (with a suitable change of variables) equations for a coupled RLC circuit in the electromagnetism course. Similarly, if students develop a program for studies of celestial bodies in the mechanics course, many of the elements of such a program can be reused in a molecular dynamics calculation in a course on statistical physics and thermal physics. The two-point boundary value problem for a buckling beam (discretized as an eigenvalue problem) can be reused in quantum mechanical studies of interacting electrons in oscillator traps, or just to study a particle in a box potential with varying depth and extension.

In order to aid the introduction of computational exercises and projects, we will need to develop educational resources for this. The PICUP project, Partnership for Integration of Computation into Undergraduate Physics, develops resources for teachers and students on the integration of computational material. We strongly recommend these resources.

As part of this proposal, we have developed several examples of problems and projects that can be included in our undergraduate courses in physics. You can click on the following links (ipython notebooks or PDF formats)

- Mechanics
  1. Ipython notebook
  2. Standard PDF file
- Quantum mechanics
  1. Ipython notebook
  2. Standard PDF file
- Electromagnetism
  1. Ipython notebook
  2. Standard PDF file
- Statistical and thermal physics
  1. Ipython notebook
  2. Standard PDF file
- Methods in Theoretical Physics (not yet ready)
  1. Ipython notebook
  2. Standard PDF file
- Physics laboratory (not yet ready)
  1. Ipython notebook
  2. Standard PDF file

Central tools and programming languages. We will strongly recommend that Python is used as the high-level programming language for all the courses proposed below. Other high-level environments like Mathematica and Matlab can also be presented and offered as special courses like PHY102 - Physics Computations I. This means that students can apply their knowledge from CMSE 201, which makes use of Python, and extend their computational knowledge in various physics classes. We recommend strongly that the following tools are used

1. Jupyter and ipython notebook.
2. Version control software like git and repositories like GitHub
3. Other typsetting tools like LaTeX.
4. Unit tests and using existing tools for unit tests. Python has extensive tools for this

The notebooks can be used to hand in exercises and projects. They can provide the students with experience in presenting their work in the form of scientific/technical reports.

Version control software allows teachers to bring in reproducibility of science as well as enhancing collaborative efforts among students. Using version control can also be used to help students present benchmark results, allowing others to verify their results.

Unit testing is a central element in the development of numerical projects, from microtests of code fragments, to intermediate merging of functions to final test of the correctness of a code.

Suggested learning goals for specific physics courses

For a major in physics degree at Michigan State University, the course CMSE 201 Introduction to Computational Modeling is compulsory and it lays the foundation for the use of computational exercises and projects in various physics courses.

We propose that the following courses aim to integrate the above learning outcomes and goals:

== Classical Mechanics 1 PHY321 === After completing Classical Mechanics 1 PHY321, students should be able to:

1. Here one could focus on ordinary differential equations like modified Euler, Verlet and perhaps RK methods. There are several interesting problems that can deepen the understanding of newtonian problems, from modeling planetary motion to models for friction and earthquake simulations. With not too advanced programs one can easily bring in actual research problems (friction is a good example). The classical pendulum is also a good case to illustrate. The code can easily be reused for studies of for example an RLC circuit in electromagnetism. The equations for a buckling beam lead to an eigenvalue problem which can reused in the quantum mechanics course.

Thermal and Statistical Physics PHY410. After completing Thermal and Statistical Physics PHY410, students should be able to:

1. Random numbers, random walks, probability distributions, Monte Carlo integration and Metropolis algorithm. Applications to statistical physics. Simulation of simple phase transitions with spin models like the Ising model. Studies of Brownian motion and diffusion can easily be done.

Methods in Theoretical Physics PHY415. After completing Methods in Theoretical Physics PHY415, students should be able:

1. Here one could discuss algorithms which are relevant for the other physics courses discussed here, such as particular eigenvalue solvers. Furthermore, when solving famous differential equations in physics, one can write functions to generate polynomials like Legendre, Laguerre, Hermite functions etc. Here it would be natural to discuss Gaussian quadrature and how to use this to integrate numerically.

Advanced Laboratory course in Physics PHY451. After completing Advanced Laboratory course in Physics PHY451, students should be able to:

1. Data analysis plays a central role, natural to have algorithms on data fitting, computations of mean values, variance and standard deviations, covariance, famous distributions. Many of these topics find also their natural place in a course on statistical physics like PHY410.

Quantum mechanics 1 PHY 471. After completing Quantum mechanics 1 PHY 471, students should be able to:

1. Ordinary and partial differential equations, eigenvalue solvers and root finding methods. Can explore interacting two-electron problem with simple rewriting of for example harmonic oscillator problem. The Metropolis algorithm can be used to simulate one and two-body problems like the Helium atom using Variational Monte Carlo. Gives students a feeling for variational calculus which can be included and taught in PHY415 and possible extensions to Quantum mechanics 2 PHY 472

Electromagnetism 1 PHY481. After completing Electromagnetism 1 PHY481, students should be able to:

- use symbolic computing tools to determine the gradient of various scalar fields
- use symbolic computing to determine the divergence and curl of various vector fields
- represent the vector (e.g., electric) field visually using vector plots and stream plots
- represent a 2D scalar (i.e., potential) field visually using 2D contour plots and 3D surface plots
- apply motion prediction algorithms Euler, Verlet, and Runge-Kutta to model the motion of charged particles in electric and magnetic fields

- compare the quality of simulations (i.e., number of iterations, step size, and error control) of charged particle motion that use different motion prediction algorithms
- apply Coulomb's law iteratively to determine the electric field produced by a given charge distribution
- apply Biot-Savart's law iteratively to determine the magnetic field produced by a given current distribution
- explain how the application of superposition iteratively gives rise to approximate field solutions
- explain how the simple relaxation algorithm works (i.e., iteratively averaging neighboring points) and how it is derived from the properties of the solutions to Laplace's equation
- apply this simple relaxation method to find the potential for 1D and 2D electrostatic situations where Laplace's equation is satisfied
- explain how to use the finite-differencing to recast Poisson's equation into a discrete formulation and how the resulting discretized form compares with the simple relaxation method (i.e., iteratively averaging neighboring points)
- apply the Jacobi and Gauss-Seidel methods to solve 2D Laplace and Poisson problems including graphing the results in three dimensions
- explain the differences between the Jacobi and Gauss-Seidel methods and how these methods are connected to the derivation using finite differencing
- Compare the quality of the simulations (i.e., number of iterations, step size, and error control) that employ the Jacobi method and the Gauss-Seidel method
- See the detailed learning outcomes for PHY 481 for more information.

Physics Education research and computing in science education

The introduction of computational elements in the various courses should be stronly integrated with ongoing research on physics education.

The Physics and Astronomy department at MSU is in a unique position due to its strong research group in physics education, the PERL group. This means that it is possible to develop research motivated curricular changes. There are many interesting challenges here, like which are the main obstacles when transferring from a classical pen and paper approach to actually have a working program which solves the same (and more general problems). What is a good

progression in presenting numerical topics in physics courses? Are there specific mathematical skills we would like our students to have? How do we integrate student-active teaching, how do we develop and test various assessment methods?

Links with CSME and Mathematics courses

We need to figure out, beyond CSE 201, if computational elements, or some of the abovementioned algorithms are included in the mathematics courses our physics students take.

Advanced computational physics courses

Towards the end of undergraduate studies it is useful to offer a course which focuses on more advanced algorithms and presents compiled languages like C++ and Fortran, languages our students will meet in actual research. Furthemore, such a course, like the present PHY480 Computational Physics offers as well more advanced projects which train the students in actual research, developing more complicated programs and working on larger projects. Such a course could cover

- C++ and/or Fortran programming
- Numerical derivation and integration
- Random numbers and Monte Carlo integration
- Monte Carlo methods in statistical physics
- Quantum Monte Carlo methods
- Linear algebra and eigenvalue problems
- Non-linear equations and roots of polynomials
- Ordinary differential equations
- Partial differential equations
- Parallelization of codes
- High-performance computing aspects and optimization of codes

Then we should consider more advanced (at the graduate level) courses which could cover specialized topics. These could be (presently some of this is covered by PHY905, sections 002 and 003)

1. Computational quantum mechanics, spanning from Monte Carlo methods to other methods used in atomic, molecular, nuclear, condensed matter and high-energy physics

2. Computational Statistical mechanics, covering topics spanning from Molecular dynamics to studies of phase transitions, highly relevant for several subdisciplines
3. Computational Astrophysics and Astronomy
4. Data analysis and machine learning tailored to physics problems (PA+CSME?)
5. High-performance computing (CSME)
6. More...

A possible Physics major in computational physics and a possible MSc in Computational Physics

To be discussed.