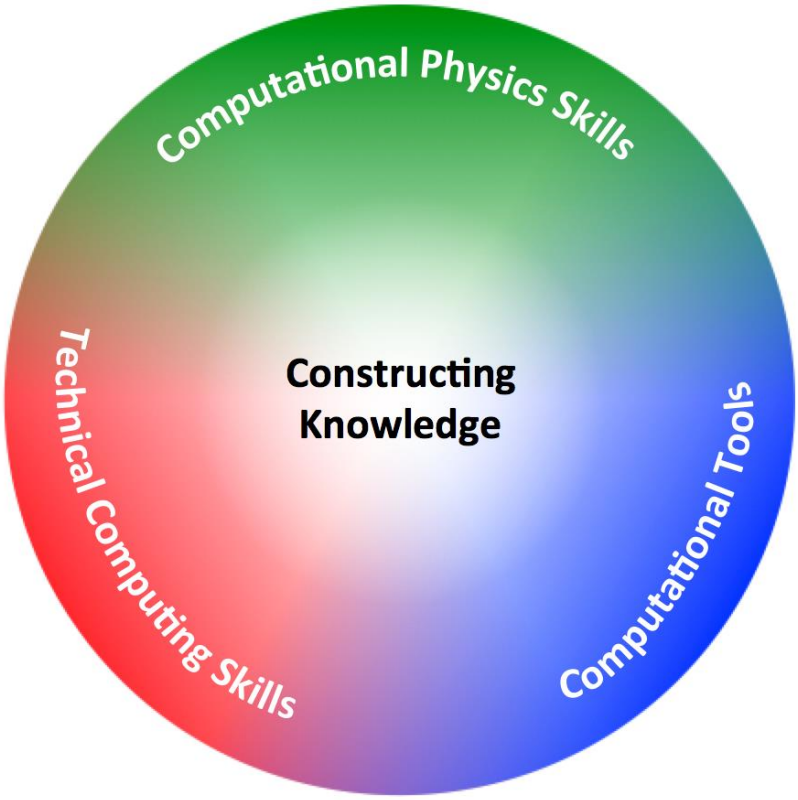
AAPT Recommendations for Computational Physics in the Undergraduate Physics Curriculum



Report prepared by the AAPT Undergraduate Curriculum Task Force
Endorsed by AAPT Board of Directors
October 2016

Table of Contents

Computational Physics in the Undergraduate Physics Curriculum

Executive Summary

The Undergraduate Curriculum Task Force (UCTF) of the American Association of Physics Teachers (AAPT) was established in 2013, in part, to develop "*…recommendations for coherent and relevant curricula…for different types of physics majors.*"

To be relevant, curricula must facilitate the development of skills that are useful to physics majors in their post-baccalaureate careers. Skills necessary for using a computer to calculate answers to physics problems, i.e., computational physics skills, are highly valued by research, industry, and many other employment sectors.

The American Association of Physics Teachers (AAPT) has therefore developed the recommendations outlined in this document. These recommendations build upon a policy that the AAPT formally adopted in 2011, urging "*that every physics and astronomy department provide its majors and potential majors with appropriate instruction in computational physics.*" Because of the variance in the student audience, the available resources, and a department's curriculum, the manner in which these recommendations are implemented is ultimately a local decision.

This document provides guidance and recommendations for incorporating computational physics into undergraduate physics programs. The reasons for doing so include the ubiquity of computation in the practice of physics, numerous immediate educational benefits of a computational approach, and the need to prepare physics graduates for the workplace and advanced degree programs. In order to develop computational physics skills, students first need to have knowledge of and experience with a variety of computational tools and fundamental computer skills. Technical computing skills and computational physics skills can then be built on this foundational knowledge. We briefly describe the types of computational tools that are widely available to enable this work. We then present a list of computation-related skills, divided into two focus areas, technical computing skills and computational physics skills. All physics majors should develop and practice these skills over the entire span of their degree programs, and we provide corresponding descriptions of learning outcome recommendations, and append specific examples. We describe different logistical ways to insert computation into the undergraduate curriculum, the many challenges that must be overcome in order to do so, and resources that can help instructors and departments implement these recommendations.

We hope this document will be useful to physics department chairs, other administrators, curriculum committees, and individual faculty members as they assess the computational elements in their programs and seek new opportunities to help their students become proficient in computational physics.

Computational Physics in the Undergraduate Physics Curriculum

I. INTRODUCTION

The practice of physics has traditionally blended two complementary approaches—theory and experiment—each with its own elaborate set of skills and methodologies. Undergraduate physics curricula have therefore evolved to emphasize instruction in pencil-and-paper calculations and laboratory skills, techniques, and practices. Since the middle of the 20th century, however, the practice of physics has increasingly incorporated an additional element: computation. Some physicists think of computation as a third fundamental approach to studying the physical world, on the same logical level as theory and experiment; others think of computation as an essential tool that both theorists and experimentalists now use in most of their work. Either way, computation has its own elaborate set of skills, and "*a curriculum in which computation is absent or plays a minor role is inauthentic to the contemporary discipline.*" [1]

Yet, despite the presence of computational work in almost all areas of physics research, undergraduate physics programs still vary enormously in the extent to which they include instruction in computational physics. Some students finish these programs with a high level of computational proficiency while others may graduate having had few useful computational experiences. Such a lack of experience and proficiency can hinder physics baccalaureates seeking employment or admission to advanced degree programs. One motivation for this report, therefore, is to help physics students develop a set of computational physics skills that helps them to achieve their career goals.

Another motivation for this report is to help students further develop their ability to construct and disseminate physics knowledge. The recent report of the AAPT Committee on Laboratories has noted that *being successful in the construction and dissemination of new knowledge requires the dynamic and interactive employment of habits of mind, skills, and knowledge that might be characterized as thinking like a physicist* [2]. Although "*thinking like a physicist*" may be difficult to specify, it would certainly include the process of modeling: conceiving, constructing, and testing models of the observable world. Because modeling is now often done computationally as well as in the laboratory, there is a close connection between the skills and practices used in computation and in the laboratory. Computational results can lead to new predictions, models, and ways of understanding physical systems. Moreover, the computational cycle is iterative, much like the experimental cycle. Here, the computational cycle is called the "*computational physics approach*" to doing physics. At the heart of the computational physics approach is computational physics thinking that builds, first, on students' fundamental computer skills using computational tools and, second, on the development of their technical computing skills.

II. RATIONALE

The rationale for integrating computational physics into the undergraduate physics curriculum rests on the following premises:

- Physics degree programs should reflect the contemporary practice of physics — and computation is now ubiquitous in that practice.
- Students gain a deeper understanding of physics and underlying mechanisms when they iteratively apply the fundamental laws of physics, whether in time to compute a system's evolution, or in space to compute the distribution of a physical quantity.
- Students using computation can develop a more exploratory approach to studying physical systems, which leads them to generate increasingly realistic models of these systems.
- Students using computation can solve a much wider range of problems, including those that are insoluble analytically or impractical to probe experimentally.
- Students using computation can develop critical skills and knowledge necessary for doing fundamental and applied research, better preparing them for a wide variety of careers. As documented in reports from the Statistical Research Center of the American Institute of Physics, there is high demand for computational skills in the workplace for physics bachelor's degree recipients [3].

III. COMPUTATIONAL TOOLS

To use the computational physics approach, students must be able to use multiple computational tools. Today's students have access to a large number of computational tools, and choosing among them may generate vigorous departmental debate. It is not the purpose of this document to endorse any particular tool, but instructors should be aware of the range of options and the trade-offs among them. We recommend that over the course of an undergraduate program, students develop proficiency in at least three different types of computational tools. The major types of computational tools are:

- **Spreadsheets.** Although they are limited in power compared to other computational tools, spreadsheets have the advantage of displaying the results of all calculations so students can see exactly what is happening. Also, many incoming physics students are already somewhat familiar with spreadsheets; and spreadsheets are used in a wide range of careers.
- **Integrated mathematical computing packages** (Mathematica, Maple, Matlab, etc.). These high-level software packages let students accomplish common computational tasks very quickly, sometimes with as little as a single

line of code, so they are often the best choice for a brief computational exercise in a course that doesn't otherwise emphasize computing.

- **General-purpose programming languages** (Fortran, C, Java, Python, etc.). These languages are the most versatile and ultimately the most powerful, and learning one or more of them can be a worthwhile long-term career investment for students.
- **Special-purpose software.** Physicists use a great variety of special-purpose software for particular tasks such as data acquisition, graphics, animations, simulations, and sharing information. Some special-purpose software packages stand on their own, while others are add-ons to one of the computing languages or integrated packages described above. Although a special-purpose software product can be the best choice for use in the classroom, instructors should consider whether it would be more empowering to teach students to use a more general-purpose tool.

The lines between these categories of computational tools are not always sharp. For example, Matlab can be used as a general-purpose programming language, while the SciPy library allows Python to be used as a high-level mathematical computing tool.

Any computational tool has an associated learning curve that requires a significant amount of time to traverse. Appropriate support must be provided so that the time students spend using the tool can be focused on learning an appropriate skill or physics concept.

Deliberately omitted from this list are tools that enable parallel computing, grid computing, and working with "big data" and large simulations. Though such tools will play a role in the future careers of many of our students, the resources needed to incorporate them into the undergraduate curriculum are not widely available. We urge the physics community to be mindful of this rapidly expanding component of computational physics. Schools with the resources and expertise available could use such tools to expand their computational curriculum beyond the recommended minimum.

IV. SKILLS

To use the computational physics approach, students must develop a large number of skills. We sort these skills into three categories: fundamental computer skills, technical computing skills, and computational physics skills.

Students should have **fundamental computer skills** that include knowledge of and experience with operating systems; file systems and file/data organization; coding, using one or more computational tools; searching effectively for technical information; and using document and presentation preparation software.

It is necessary to know the fundamental computer skills that incoming students have so that a solid foundation can be developed through appropriate advising, support, and instruction.

Building on their fundamental computer skills, students should develop **technical computing skills**, which include the ability to:

- process data;
- represent data visually;
- prepare documents and presentations that are authentic to the discipline.

Ultimately, the computational physics approach makes use of skills that are characteristic of what might be called computational physics thinking (the physics-specific instance of computational thinking [4]), which is a synthesis of physics principles and algorithmic thinking. Such **computational physics skills** include the ability to:

- translate a model into code;
- choose scales and units;
- subdivide a model into a set of manageable computational tasks;
- choose algorithms and computational tools;
- debug, test, and validate code; and
- extract physical insight.

Developing technical computing skills is co-requisite for developing computational physics skills, and both depend on the presence of appropriate computational tools and the ability of the student to use these tools. Computational tools, technical computing skills, and computational physics skills are therefore strongly coupled to one another, and it is probably more effective to learn them together, in context, rather than in isolation.

We provide more detailed descriptions of the technical computing skills and the computational physics skills in the recommendations described in the following section.

V. RECOMMENDATIONS FOR LEARNING OUTCOMES

This section describes recommendations for learning outcomes in an attempt to define a *minimum* set of technical computing skills and computational physics skills that physics majors should develop during their undergraduate programs. The development of these skills does take time and should be intentionally integrated into the undergraduate physics curriculum and reinforced throughout the curriculum, rather than taught as isolated skills.

The first focus area, Technical Computing Skills, describes skills that must be developed in order to engage productively in the production and dissemination of physics knowledge through the use of computers. The second focus area,

Computational Physics Skills, describes skills and practices that involve synthesizing physics with algorithmic ideas, coding/implementation, self-checking, and scientific argumentation. These two focus areas provide a framework for curricular and program development. In addition, mastering these two focus areas prepares students to perform computational investigations at a level appropriate for secondary teaching while providing a foundation of skills and practices that should be significantly augmented for students intending to enter industry or a graduate program. Additional educational research would help to refine detailed descriptions of the skills in these two focus areas. Such research would then enable further curriculum development and assessment of these skills.

To maintain the focus on computation-related skills, it is assumed below that students have learned to identify the physics concepts relevant to developing a theoretical model, to make reasonable simplifying assumptions, and to mathematically describe a theoretical model (e.g., write the equations of motion).

Finally, no attempt was made to distinguish between introductory-level or upper-level instruction because faculty at a particular institution [5] need to collaboratively choose the appropriate points in their curricula for students to develop different skills.

1. Technical Computing Skills

Students should be able to:

- **Process data**
  Students should be able to use computers to process data, which includes reducing, fitting, filtering, and/or averaging data, and computing uncertainties from measurements. All the computational tools described in Section III can be used to process data. It is worth noting that processing data can be done without a specific model in mind; and so this technical skill is distinguished from the computational physics skill of extracting physical insight.

- **Represent data visually**
  Students should be able to produce static visualizations (i.e., plots) of data because plots are fundamental to facilitating analysis and communication of data. Additionally, because different types of plots are appropriate for different data sets and analyses, students should be able to generate several types of plots. It is critically important for students to be able to graphically represent uncertainties on the data because the physical insights that can be extracted from data are constrained by uncertainty.

- **Prepare professional documents and presentations**
  Students should be able to prepare professional documents and presentations. This skill is required to communicate results and is necessary in any professional field. As discussed in the Laboratory Guidelines document [2], students should produce these documents and presentations in forms

authentic to the discipline such as technical memos, journal-style articles, slides for professional-style oral presentations, and scientific posters.

2. Computational Physics Skills

Students should be able to:

- **Translate a model into code**
  Students should be able to translate a theoretical or algorithmic model into code that enables computation. This is a multifaceted skill, which includes the abilities to: use a computational tool to write readable, well-documented code with correct syntax; use language documentation and/or reference materials for the chosen computational tool to resolve coding questions and expand coding vocabulary; and apply physics knowledge of the given system to make discretization choices and monitor numerical errors, convergence, etc.

- **Choose scales and units**
  Students should be able to choose physical scales and units appropriate to the system. Often the chosen scales are used to convert equations to dimensionless variables before coding. Such scales provide a measure against which one can determine the significance of different phenomena at play in a physical system.

- **Subdivide a model into a set of manageable computational tasks**
  Students should be able to logically subdivide a computational model into a set of manageable computational tasks, and organize their code accordingly. That is, they should identify the types of computational tasks to be performed, the required inputs, the sequence of tasks, and the processing of outputs to obtain a solution.

- **Choose algorithms and computational tools**
  Students should be able to choose among computational algorithms and computational tools to produce a solution. The ability to make such choices presupposes student familiarity with a variety of computational algorithms and tools as well as criteria for choosing among different algorithms and tools. For example, some mechanical systems should be simulated using a differential equation algorithm that conserves energy, or one that uses a variable time step; some many-body simulations require a dynamical algorithm, while others are suited to Monte Carlo techniques.

- **Debug, test, and validate code**
  Students should be able to debug, test, and validate computational models. This process can include resolving error messages and other incorrect behavior; checking special cases for which answers are already known; adjusting the resolution of a discretization, or the limit of a sum, to estimate the sizes of truncation errors; comparing to experimental data or to a different

computational model; and simply asking whether the results are physically plausible.

- **Extract physical insight**
  Students should be able to extract physical insight from a computation by converting the raw output of a computation into a useful form, asking interesting questions, and using the computation to answer these questions. Often this process involves repeating a computation many times using different sets of parameter values of particular interest, and communicating results effectively to others in forms authentic to the discipline. Students should use their results to determine the effectiveness and/or limitations of a model and further refine the model (e.g., by adding missing phenomena) based on a comparison to experimental or theoretical results or other validated models.. Students should navigate the cycle from model to implementation to results to concepts to revised model, in order to experience the iterative nature of constructing physics knowledge.

APPENDIX B provides several examples of these learning outcomes, organized in tabular form. We also refer the reader to examples of significantly augmented curricula [6-8] that achieve many or all of these outcomes. APPENDIX C describes several computational physics tasks that are becoming widespread in undergraduate physics curricula and can provide context for achieving several of the learning outcomes.

VI. CURRICULAR ISSUES

Once a department has decided upon a set of computational learning objectives, the faculty must decide what kinds of curricular structures are needed to help students meet those objectives. The structures should ultimately be robust enough and sufficiently supported so that all instructors (permanent or visiting) can contribute effectively.

Computation can be introduced into the undergraduate physics curriculum in any or all of the following ways:

- Adding computational exercises and projects to existing physics courses that have traditionally emphasized pencil-and-paper calculations and/or laboratory measurements;
- Designing new courses that merge traditional elements with computational work;
- Offering one or more dedicated courses in computational physics;
- Providing opportunities for computational independent study and/or research projects.

Departments should use as many of these approaches as possible and are encouraged to identify and use other approaches that lead to the development of students' computational skills. We argue that relying entirely on dedicated computational physics courses and/or independent study is not sufficient because such a program would overly segregate computational work, making too few connections to the rest of the curriculum. On the other hand, merely adding computational exercises to existing courses runs the risk of allocating too little time to computation, and/or, "hiding" it in the curriculum with too little explicit recognition. Moreover, such exercises, when attempted without the necessary preparation or scaffolding, can lead to student frustration.

Computational work should be introduced in the introductory course. The amount of computation that can feasibly be done in that course will vary among institutions, but incorporating computational physics early allows faculty to assess and build students' fundamental computer skills and experience with one or more computational tools. It also allows students to explore authentic, complex problems that they might not see until later in the curriculum.

Many instructors who teach computational physics have found that it is best taught in a "lab" setting where students can work at a more flexible pace, help each other, and obtain help when needed from an instructor or lab assistant [9]. In addition, developing these skills in a communal environment helps students to function as members of a scientific community.

VII. CHALLENGES

Integrating computation into the undergraduate physics curriculum is not easy. Let us, therefore, acknowledge some of the main challenges to doing so:

- **The hidden curriculum.** Many departments do not yet have explicit goals for most of their curricula, let alone for computational physics. An explicit statement of those goals is crucial before an effective curriculum and associated activities can be developed.
- **Curricular time.** Physics programs traditionally require a great deal of course work even without any computational components. Making time in the curriculum for computational work can therefore be a significant challenge. To address this challenge, department faculty should together carefully consider the value that computation will have for their students, and decide together how local resources are best managed to facilitate learning computational physics.
- **Time demands**. It is crucial to understand where, and how, computational work can be added to the curriculum without imposing impossible time demands on the students. It is also important to be realistic about the amount of faculty time needed to implement computational work in the curriculum.

- **Range of instructor backgrounds.** Physics instructors, although familiar with computational work in their scholarly activities, may be uncomfortable with the types of computation that can be most naturally incorporated into their courses. Workshops, conferences, and professional societies are already taking steps to help faculty learn more about teaching computational physics, as described in the Resources section below.

- **Range of student backgrounds.** Some students enter a physics program with extensive programming experience, while others have never written a single line of code and may be afraid of programming a computer. Departments will need to know and take into account the backgrounds of the local student cohort when making curricular decisions.

- **Variety of computational tools.** Students coming into a course may have already learned to use some computational tools and developed preferences for one over another. Instructors also have varied backgrounds and preferences. It is virtually impossible for a single instructor to provide technical support for all the tools that students might wish to use. Faculty need to work closely with their colleagues, both inside and outside the department, when choosing computational tools to use, and they should be willing to compromise to provide a coherent computational approach for their students.

- **Inadequate textbooks.** Few of the popular textbooks used in traditional physics courses integrate computational work in a nontrivial way; exceptions are described in the Resources section below. There is a continuing need for new (or revised) textbooks that thoughtfully integrate more computation.

- **Shortage of educational research.** Most of our knowledge of how to teach (or how not to teach) computational physics is anecdotal. There is a shortage of published research to document what works and what does not. This shortage means there are many ongoing opportunities for physics education researchers, and numerous other studies that have yet to be pursued. Support for such research and its dissemination is needed.

- **Lack of community and support.** It takes a significant amount of time for a single faculty member to effectively include computation into an existing course. To do so in isolation may make it more difficult. Consequently, a community of instructors who integrate computation in their teaching is needed to provide support and professional development.

- **Space and scheduling constraints.** Instructors who wish to teach computation in a lab-style setting may find that no suitable classroom is available, or that it is impractical to schedule a class for the longer blocks of time that lab work usually requires. Faculty may need to be flexible in finding ways to teach computational physics.

- **Hardware challenges.** There is a cost to maintaining a large number of school-owned workstations for physics students to use in their computational

work. Many students prefer to use their own computers, but the variety of hardware and operating systems then creates troubleshooting challenges for instructors.

- **Software installation.** Installing and configuring specialized software for computational physics is time-consuming and can be frustrating.

The good news is that despite all these challenges, many physics departments have successfully integrated a great deal of computation into their undergraduate curricula. The next section provides pointers to resources that can help departments address many of these challenges.

VIII. RESOURCES

The following set of resources is provided as a starting point for instructors and departments looking to implement these computational physics recommendations in their courses and curricula. It should be noted that this is neither an exhaustive nor an endorsed list of resources. There are certainly other resources available, and new resources are being produced all the time that instructors and departments may also find beneficial.

Collections of Resources

The American Journal of Physics (AJP) Resource Letters on computational physics contain annotated lists of textbooks for computational physics courses and lists of articles that discuss ways to integrate computation into the physics curriculum.

Paul L. DeVries, "Resource Letter CP-1: Computational Physics," Am. J. Phys. **64** (4), 364-8 (1996).

Rubin H. Landau, "Resource Letter CP-2: Computational Physics," Am. J. Phys. **76** (4&5), 296-306 (2008).

The second resource letter by Landau leads the April 2008 theme issue on computational physics (see below). The issue is an excellent entry point for those starting the process of integrating computation into the physics curriculum.

ComPADRE, the physics and astronomy archive, already has a significant set of holdings of computational physics articles, conference proceedings, and curricular materials. Computational physics-related materials are available in various ComPADRE collections, including the Open Source Physics collection and collections focused on particular topics, such as statistical and thermal physics, and quantum mechanics.

http://www.compadre.org/

The Partnership for Integration of Computation into Undergraduate Physics (PICUP) is now making computational activities, for both the introductory and upper level, available through ComPADRE:

http://www.compadre.org/PICUP

A summary of the 2007 Topical Conference on Computational Physics in the Upper-level Curriculum is archived on ComPADRE:

http://www.compadre.org/portal/items/detail.cfm?ID=11362

Speakers and the schedule of the *2008 Gordon Research Conference on Computation and Computer-Based Instruction* are archived on the Gordon Research Conference website. Many of the speakers contributed articles to the AJP theme issue on computational physics (see below).

https://www.grc.org/programs.aspx?id=10156

In April 2008, the American Journal of Physics published a double issue with the theme of computational physics. The issue contains a highly useful collection of articles that touch on virtually every aspect of teaching computational physics.

American Journal of Physics **76** (4&5), (2008).

The Next Generation Science Standards has a section on K-12 computational thinking skills. Mathematical and computational thinking are described in *Appendix F - Science and Engineering Practices in the NGSS*.

http://www.nextgenscience.org/next-generation-science-standards

Perspectives on Computational Physics in the Undergraduate Curriculum

The following articles also provide useful information and perspectives:

Robert G. Fuller, "Numerical Computations in US Undergraduate Physics Courses", *Comp. Sci. Eng.* **8**, 16-21 (2006).

Jeanette M. Wing, "Computational Thinking," *Comm. of the ACM* **49** (3), 33-35 (2006).

Marty Johnston, "Implementing Curricular Change," *Comp. Sci. Eng.* **4** (5), 32-37 (2006).

Norman Chonacky and David Winch, "Integrating computation in to the undergraduate curriculum: A vision and guidelines for future developments", *Am. J. Phys.* **76** (4&5), 327-333 (2008).

Knut Mørken *et al*., "Computing in Science and Engineering: A guide for universities and colleges in Norway", report to the Norwegian Ministry of Education and Research, June 15, 2011.

Ruxandra M. Serbanescu, Paul J. Kushner, and Sabine Stanley, "Putting

computation on a par with experiment and theory in the undergraduate curriculum," Am. J. Phys. **79** (9), 919-924 (2011).

Marcos D. Caballero and Steven J. Pollock, "A model for incorporating computation without changing the course: An example from middle-division classical mechanics," Am. J. Phys. **82** (3), 231-237 (2014).

Curricular Materials

Examples of mature introductory textbooks that integrate computation are

Matter and Interactions, 4th edition (Wiley, 2015), by Ruth W. Chabay and Bruce A. Sherwood.

Six Ideas that Shaped Physics, 3rd edition (McGraw-Hill, 2017), by Thomas A. Moore.

Examples of upper-level textbooks or supplements to upper-level textbooks that integrate computation are

Physlet Quantum Physics: An Interactive Introduction, 2nd edition, at http://www.compadre.org/pqp/, by Mario Belloni, Wolfgang Christian, and Anne J. Cox.

Statistical and Thermal Physics: With Computer Applications, 6.1.2010 edition (Princeton, 2010), by Harvey Gould and Jan Tobochnik.

Computation and Problem Solving in Undergraduate Physics (Lawrence University Press, Appleton, WI 2013), by David M. Cook.

APPENDIX A: The AAPT Statement on Computational Physics and its Rationale

The AAPT Statement on Computational Physics, which was approved in 2011, is as follows:

> *The American Association of Physics Teachers urges that every physics and astronomy department provide its majors and potential majors with appropriate instruction in computational physics.*

Rationale:

Contemporary research in physics and related sciences almost always involves the use of computers. They are used for data collection and analysis, numerical analysis, simulations, and symbolic manipulation. Computational physics has become a third way of doing physics and complements traditional modes of theoretical and experimental physics. In addition, almost all undergraduate students who take physics courses will use computational tools in their future careers even if they do not become practicing physicists.

One of the traits that distinguishes physics from other sciences is the ability to develop new tools as needed to do our work. These new tools include new experimental techniques, mathematical methods, theoretical formalisms, and now new computer algorithms. Thus, we should include in the physics undergraduate curriculum some level of instruction in computer algorithms appropriate for solving problems in physics.

Insight into understanding physics can be gained in many ways. Experiments emphasize that our models are connected to the real world, and frequently surprise us with new phenomena we didn't expect. Theory provides the tools for organizing our knowledge, making predictions, and describing nature in a concise and compelling manner. The computer provides a new tool that enhances both theory and experiment. Computer simulations allow us to develop models that are not solvable analytically, to test theories where traditional experiments are too difficult or expensive, to ask “what-if” questions, and to visualize the time development of dynamical systems. As a result simulations provide different insights, which may not be possible to obtain through the use of traditional theoretical and experimental methods.

APPENDIX B: Recommendations for Learning Outcomes: Examples

The recommendations for student learning outcomes given in this document are for a *minimum* set of technical computing skills and computational physics skills that physics majors should develop during their undergraduate major. These recommendations are intentionally general enough that they are universally accessible. The way in which these recommendations are implemented will vary from institution to institution, depending, for example, on the local student population and the resources available. Departments that have the expertise and/or resources to go above and beyond these recommendations are strongly encouraged to do so in order to help their majors develop additional knowledge and skills.

Each of these learning outcomes should be addressed at some point during the full span of the undergraduate physics curriculum. Similar to the experimental and theoretical physics curricula, the computational physics curriculum should be a spiral curriculum such that students develop and reinforce their skills through scaffolded, multiple experiences, beginning in the introductory courses and continuing in upper-level courses. Specific examples for the two focus areas are provided in the tables below. It is important to note that these few examples are intended only to aid departments in implementing computational physics into their curriculum. Not all of these examples need to be included in the physics curriculum; in fact, there probably is not time to include them all in a physics curriculum. Also, these examples are certainly not the only ways these learning outcomes could be demonstrated; departments are encouraged to develop, and share, their own examples.

1. Technical Computing Skills Students should be able to:	
Process Data.	**Example**: Use and/or code a least-squares fitting of data to a functional form, and plot the data with uncertainties together with the fit function, e.g., fitting the profile of a laser beam, or a far-field diffraction pattern. **Example**: Compute a data set for graphical comparison to appropriately reduced experimental data, including uncertainties. **Example**: Compute the average of several data sets to reduce the effects of run-to-run variations and compare to theoretical models. **Example**: Given a controlled series of spectra with occasional noise spikes, filter out the spikes and integrate peak areas to plot the signal as a function of the control variable.

1. Technical Computing Skills Students should be able to:	
Represent Data Visually.	**Example**: Produce two-dimensional plots of one or more sets of data with error bars in both dimensions, e.g., from video analysis of everyday sports projectiles. **Example**: Produce a contour plot of the period of a physical pendulum as a function of its center of mass position and the characteristic size of the pendulum. **Example**: Produce a histogram of detection events as a function of event characteristic, such as particle energy or time-of-flight, for different detector characteristics such as position. **Example**: Produce phase space plots, a bifurcation diagram, and a Poincaré section for a chaotic oscillator.
Prepare professional documents and presentations	**Example**: Write a technical memo that incorporates the graphical representation of data. **Example**: Prepare a professional manuscript that satisfies all journal submission guidelines, especially with regard to figures. **Example**: Prepare an oral or poster presentation that satisfies the guidelines of a professional organization. **Example**: Demonstrate awareness of audience needs during the production of visual and written representations of outputs. Actively consider different visual designs to make graphic representations more effective.

2. Computational Physics Skills Students should be able to:	
Translate a model into code.	**Example**: Write code to set up arrays of values, perform calculations in a sequence and/or under specified conditions, and generate numerical or graphical output. **Example**: Use reference materials to adapt code examples to perform a particular task, such as calculating the sum of values in a particular array (e.g., summing forces exerted by an array of charged particles).
Choose scales and units.	**Example**: For a spherical balloon (projectile) falling in air, relevant scales could include the length scale set by the balloon's diameter, and the time scale set by the amount of time for the balloon to free fall its diameter.
Subdivide a model into a set of manageable computational tasks.	**Example**: Calculating the motion of a charged particle in a region with a specified distribution of charges involves calculating the net force by applying the principle of superposition (summing), numerically integrating the equations of motion, and storing and/or plotting the trajectory. To simulate a beam of such particles, an additional task is to organize the storage of the many particles that constitute the beam.
Choose algorithms.	**Example**: When numerically integrating equations of motion, choose among algorithms such as the Euler-Cromer method, fourth-order Runge-Kutta with constant step size, or with a variable step size set by an error criterion. Consider trade-offs between complexity and execution speed when choosing an algorithm.

2. Computational Physics Skills Students should be able to:	
Debug, test, and validate code.	**Example**: Use the debugging tool in an integrated development environment (IDE) to rapidly find and resolve bugs. Isolate portions of a code to determine where a particular bug occurs. **Example:** Check that a code to solve for the motion of a falling object in air reproduces the analytical free fall result when the drag force is set to zero. **Example:** Check that a code to solve for the evolution of a harmonic oscillator reproduces the analytical solution. **Example**: Check that a code to calculate the electric field due to a symmetric distribution of charges converges to the analytical solution as the discretized distribution becomes more fine-grained. **Example**: Probe a model to compute charged particle trajectories to test whether or for what values the model breaks down for relativistic particles. **Example**: Check that a code to process a data set performs as intended by comparing outputs to expected outputs for test data. **Example**: Check that a code to solve the time-independent Schrödinger equation by the shooting method reproduces the analytical solution for the finite square well. **Example**: Check that a code to model an ideal gas reproduces the Maxwell-Boltzmann distribution for the given conditions.

2. Computational Physics Skills Students should be able to:	
Extract physical insight.	**Example**: Make a contour plot of the period of a physical pendulum versus adjustable geometric parameters (i.e., the center of mass position, the size of the pendulum) to determine the sensitivity of the period to these parameters. **Example**: Compute the motion of an object falling through air, using different models for air drag, and compare to experimental data. **Example**: Compute the electric field due to a nontrivial charge distribution and determine its sensitivity to perturbations in the distribution. **Example**: Compute the evolution of quantum-mechanical two-level system and study its dependence on the nature and strength of the coupling between the levels. **Example**: Compute the normal modes of an acoustic object and the frequency spectrum generated by a impulsive hit; compare to the corresponding experimental frequency spectrum, obtained by applying a fast Fourier transform to a recorded sound. **Example**: Use the visual representation of the output as evidence in developing clearly stated scientific arguments.

APPENDIX C: Example Computational Physics Tasks

Listed below are a number of computational physics tasks that are becoming reasonably widespread in undergraduate course work. We do not recommend that every undergraduate program try to incorporate all (or even most) of these examples, and we know that instructors are continually developing exercises that are more innovative than these. For those who may be new to the teaching of computational physics, we hope this list can serve as a starting point.

- Use a simple second-order algorithm to integrate Newton's second law, to predict the behavior of a mechanical system such as a projectile, a pendulum, or celestial bodies.
- Simulate and explore the behavior of a chaotic system such as a damped, driven pendulum or the Lorenz model.
- Simulate the dynamics of a many-body system such as a Lennard-Jones fluid, to explore phase behavior and irreversible processes.
- Calculate electric and magnetic fields of nontrivial charge and current distributions, using the principle of superposition.
- Solve Laplace's equation by the relaxation method, to obtain the electrostatic potential near a conductor with a nontrivial shape.
- Numerically integrate probability distributions (e.g., Maxwell-Boltzmann distribution, Planck spectrum, or a Gaussian wave packet) to obtain probabilities of measured values being in specific ranges.
- Solve the time-independent Schrödinger equation in one dimension by the shooting method, to obtain energy levels and stationary-state wave functions.
- Use a matrix eigen-system library routine to find the normal modes of a system of coupled oscillators, or to solve the time-independent Schrödinger equation.
- Use combinatoric functions to calculate the entropy and heat capacity of a collection of two-level systems or quantum harmonic oscillators.
- Use pseudo-random numbers to simulate radioactive decay, diffusion, or some other random process.
- Use the Metropolis algorithm to simulate a simple fluid, or the Ising model of a ferromagnet, held at constant temperature.
- Simulate the time evolution of a continuous system, according to the wave equation or the time-dependent Schrödinger equation.
- Analyze signals or wave shapes using a fast-Fourier-transform library routine.

References

[1] Wolfgang Christian and Bradley Ambrose, "An Introduction to the Theme Double-Issue", *Am. J. Phys.* **76** (4&5), 293-294 (2008).

[2] AAPT Recommendations for the Undergraduate Physics Laboratory Curriculum, available at http://www.aapt.org/Resources/. Nancy Beverly; Duane Deardorff; Richard Dietz; Melissa Eblen-Zayas; Robert Hobbs; Dean Hudek; Joseph Kozminski; Heather Lewandowski; Steve Lindaas; Ann Reagan; Randy Tagg; Jeremiah Williams; and Benjamin Zwickl.

[3] Patrick Mulvey and Starr Nicholson, "Physics Bachelor's Initial Employment", *focus on* report, American Institute of Physics, June 2015.

[4] Jeanette M. Wing, "Computational Thinking", *Comm. of the ACM* **49** (3), 33-35 (2006).

[5] In some cases, there may be a very strong link between, say, a two-year college and a four-year college or university via the transfer of students. These students would benefit from inter-institutional faculty collaboration on computational physics instruction.

[6] Marty Johnston, "Implementing Curricular Change", *Comp. Sci. Eng.* **4** (5), 32-37 (2006).

[7] David M. Cook, "Computation in undergraduate physics: The Lawrence Approach", *Am. J. Phys.* **76** (4&5), 321-326 (2008).

[8] David H. McIntyre, Janet Tate, and Corinne Manogue, "Integrating computational activities into the upper-level Paradigms in Physics curriculum at Oregon State University", *Am. J. Phys.* **76** (4&5), 340-346 (2008).

[9] Ross L. Spencer, "Teaching computational physics as a laboratory sequence", *Am. J. Phys.* **73** (2), 151-153 (2005).

ABOUT THIS DOCUMENT

This document was developed through several versions by a subset of members of the AAPT Undergraduate Curriculum Task Force (UCTF): Ernie Behringer (Eastern Michigan University), Juan Burciaga (Bowdoin College), Dick Dietz (University of Northern Colorado), Andy Gavrin (Indiana University-Purdue University at Indianapolis), Joseph Kozminski (Lewis University), and Victor Migenes (Brigham Young University).

Daniel Schroeder (Weber State University) provided an alternative version that fundamentally influenced and strongly contributed to the final version. The UCTF is very grateful for his extensive and thoughtful input.

The task force thanks the following people for the input to various early versions of this document: Mario Belloni (Davidson College), Michael Falk (Johns Hopkins University), Joe Heafner (Catawba Valley Community College), Brian O'Shea (Michigan State University), Jan Tobochnik (Kalamazoo College), and PICUP members Marcos "Danny" Caballero (Michigan State University), Norman Chonacky (Yale University), Larry Engelhardt (Francis Marion University), and Kelly Roos (Bradley University) for providing valuable feedback. The task force also thanks Elizabeth George and colleagues at Wittenberg University, and Marie Lopez del Puerto and colleagues at the University of St. Thomas for their helpful input.

ABOUT THE AAPT Undergraduate Curriculum Task Force

The Undergraduate Curriculum Task Force (UCTF) of the American Association of Physics Teachers (AAPT) was established in 2013, with the following charge:

The AAPT Undergraduate Curriculum Task Force (UCTF) is charged with developing specific, multiple recommendations for coherent and relevant undergraduate curricula (including course work, undergraduate research, mentoring, etc.) for different types of physics majors in collaboration with the APS and AIP, and with developing recommendations for the implementation and assessment of such curricula.

At the time of its establishment in 2013, the UCTF consisted of the following members, many of whom were drawn from several different AAPT Area Committees at that time:

	Name	Institution/Organization
1	Trish Allen	Appalachian State University
2	Ernie Behringer	Eastern Michigan University (Chair)
3	Juan Burciaga	Mt. Holyoke College
4	Beth Cunningham*	AAPT (Executive Officer)
5	Dwain Desbien	Estrella Mountain College
6	Dick Dietz	University of Northern Colorado
7	Jerry Feldman	George Washington University
8	Noah Finkelstein	University of Colorado, Boulder
9	Andy Gavrin	Indiana University-Purdue University, Indianapolis
10	Dennis Gilbert	Lane Community College
11	Tim Grove	Indiana-Purdue University at Fort Wayne
12	Bob Hilborn*	AAPT (Associate Executive Officer)
13	Ted Hodapp	APS (Director of Education & Diversity)
14	Seth Guinals Kupperman	NYC HS for Math, Science & Engineering
15	Joseph Kozminski	Lewis University
16	Ntungwa Maasha	College of Coastal Georgia
17	Corinne Manogue	Oregon State University
18	Victor Migenes	Brigham Young University, Provo
19	Tom Olsen**	American Institute of Physics
19	Steve Shropshire*	Idaho State U
20	Rob Steiner	American Museum of Natural History
21	Aaron Titus*	High Point U

* Ex oficio.

** AIP Representative