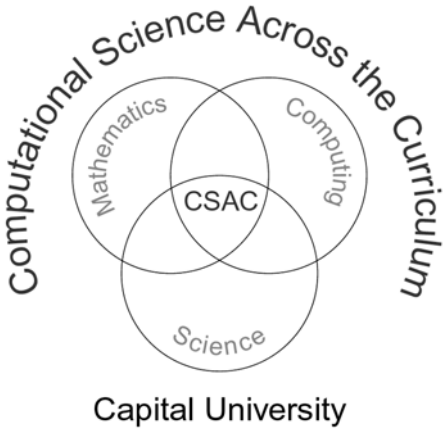
A Guidebook for the Creation of Computational Science Modules



Battelle

W. M. KECK FOUNDATION

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, Battelle, or the W. M. Keck Foundation.

This guidebook was modeled after and borrows substantially from the ChemConnections developer resource pages http://mc2.cchem.berkeley.edu/modules/index.html.

Thanks, also, to Dr. Brownstein who provided important references and comments, a couple of which are used verbatim.

Quick Start: Checklist for Submitting a Module.

To ensure that the module includes all necessary information, please use this checklist before submitting a complete module.

_____ Title

_____ Author name(s) and email address(es) and affiliation(s)

_____ Year

_____ Funding source

_____ Math level and science level of the module are identified

_____ Module description (an abstract of the module)

_____ Keywords

_____ Introduction

_____ Problem Statement

_____ Background Information

_____ Model

_____ Solution Methodology/ Implementation

_____ Assessment

_____ Empirical data (if applicable)

_____ Conceptual Questions with Answer Key

_____ Problems and Projects

_____ Solutions (if applicable)

_____ Suggestions to Instructors (if applicable)

_____ Glossary

_____ References

Introduction.

Computational science is a field at the intersection of mathematics, computer science, and science (hereafter, broadly defined to include biology, chemistry, engineering, environmental science, finance, geology, medical science, neuroscience, physics, and psychology). Computational science offers an interdisciplinary approach to scientific research and provides an important tool, alongside theory and experimentation, in the development of scientific knowledge.

Goals and Objectives.

The *problem* at the undergraduate level is a lack of educational materials for computational science. Much of the development of these computational science methods have been confined to specific disciplines within the sciences. The commonalities in modeling and visualization approaches between many disciplines provides a unique opportunity teach undergraduate students about this interdisciplinary field of study. The *objective* of this project is to develop course materials (in a modular format) that culminate in a comprehensive, interdisciplinary curriculum for computational science at the undergraduate level.

The project targets national needs to enhance students' knowledge base in computational science, and to improve student attitudes and appreciation of mathematics and science as creative, collaborative, and interdisciplinary fields of inquiry. The goals and objectives for this project are:

Primary Goal:

- To develop materials that constitute an interdisciplinary computational science curriculum

Secondary Goals:

- To emphasize an interdisciplinary, team-based approach to science problem solving
- To cultivate undergraduates' understanding of the creative nature of computational science
- To improve written and oral communication related to scientific and technical projects
- To facilitate student use of current and emerging computing technologies
- To increase the number of students who pursue graduate degrees in science and mathematics

Pedagogical Approaches.

This integrated curriculum is important because it emphasizes critical thinking skills, problem-solving techniques, and a team approach to undergraduate student research. Modules use inquiry-based pedagogy focused on a problem-oriented approach. Through the inquiry-based pedagogy, instructors use problems as the context for developing theoretical concepts. Instructors facilitate student learning by: a) presenting students with a problem to solve; b) having students formulate possible solutions; c) stimulating students' thinking by asking questions; d) having students discuss their solutions; and e) having students assess their work by comparing and defending their solutions. This pedagogical strategy is endorsed in the Boyer Commission Report (Boyer, 1998).

In addition to the inquiry-based pedagogy, modules are structured around collaborative learning

(i.e., peer instruction). Mazur (1997) developed, tested, and demonstrated the efficacy of peer instruction for an introductory physics course; this methodology serves as a model for the proposed modules. The strength of this approach is that students are not passive repositories for information; they must manipulate and verbalize their understanding as they defend their position to their peers. For each course, a set of conceptual questions serves as a resource to aid instructors in assessing students' conceptual understanding and to facilitate peer learning.

The computational science modules should challenge higher thinking skills in students and demonstrate the integration of the disciplines. This type of learning may be frustrating for the students. This frustration can be addressed through thoughtful consideration of required previous knowledge and by creating an environment where students are encouraged to take risks and attempt creative solutions. Thus, as you consider the students' experience of the module, keep in mind the following pedagogical techniques and decide which subset of these techniques best achieves the learning objectives for your module. A brief list of sources related to these techniques is available in the reference list.

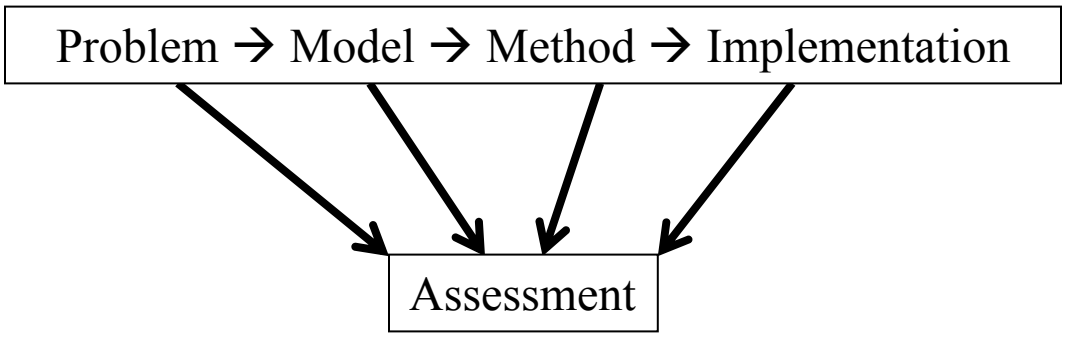
- Guided inquiry (or structured inquiry) can be used in the classroom or laboratory. In the classroom, students may be supplied with data or observe a demonstration. Through discussion in class and/or through investigating outside resources, the students learn about the modeling and computational techniques. It can be particularly intriguing to supply students with an anomalous or counter-intuitive example. In some modules, students begin by observing something interesting or generating some data. They must then discover the scientific principles behind their observations – it is this process of discovery that necessitates the use of computational tools.
- Open-ended inquiry emphasizes the process of doing science. The instructor does not have a specific outcome in mind, but rather sets up a situation where students can be creative while learning science. An open-ended question would encourage students to use both their prior knowledge and outside resources to investigate an area of interest. In the computer laboratory, students may begin by proposing a question they would like to investigate, designing experiments, collecting the necessary data, analyzing the data, and defending their results. Students are evaluated on how well they have completed the steps of the process, not on whether they got a specific result in the experiment.
- Cooperative learning (or collaborative learning) involves carefully structured group activities. The activity is structured so that group members are interdependent (they must all participate to succeed) and individually accountable (all members are responsible for learning). Part of the structure includes an evaluation that allows the students to reflect on what worked well in the group, what didn't, and how the group process could be improved. Careful structure is the key to the success of a cooperative activity.
- The interactive classroom encourages active participation of students, interaction between students, and interaction between faculty and students. Some examples are:
  - In-class problem solving in small groups.
  - Turn-to-your-neighbor activities (explain what you observed in the demo, summarize the key points that have been covered, etc.).
  - Getting students up front (to solve a problem on the board, to participate in a demo).
  - Two-minute paper (can be used at the end of class to assess what questions the students still have, good for providing instructor with feedback).

- Writing to learn. Many of these activities come from the writing-across-the-curriculum movement. Some examples are:
  - Write what you know about... (used to get students thinking about a topic, to assess student's prior knowledge, and to document the student learning process).
  - Journal-keeping.
  - Two-minute paper (used to get quick feedback from students about their concerns or questions).
- Lecture can be used when students have questions they need an "expert" to answer. That expert does not have to be the instructor, but could be one of the students in the class or an outside consultant. Lectures can also be used to motivate and develop enthusiasm.
- Concept-mapping is useful for helping students make explicit connections between the things they're learning. According to Ruiz-Primo and Shavelson (1996): "A concept map is a graph consisting of nodes representing concepts and labeled lines denoting the relation between a pair of nodes. A student's concept map is interpreted as representing important aspects of the organization of concepts in his or her memory (cognitive structure)." Students link together concepts with "logical connectors" that explain the relationships between the concepts. This may be particularly useful in helping students make connections between their own experiences and the computational science they are learning in the classroom. There is some empirical evidence that concept-mapping "effectively promotes meaningful learning and metacognition" (Materna, 2001 see also, Ruiz-Primo, Shavelson, Li, & Schultz, 2001).
- In-class debates allow students to practice using scientific arguments to support and defend a stance they may take.

What is a module? An Overview.

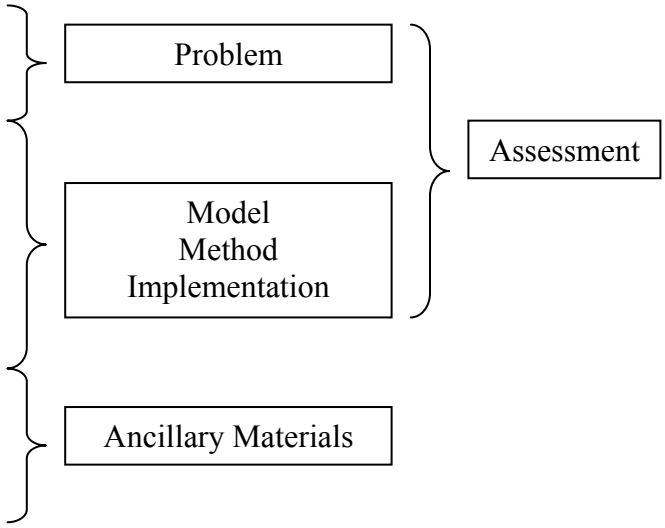
For most courses, modules are grounded in a story that asks an important question and entices students into wanting to know the answer. The questions should be answerable through computational science techniques.

We conceptualize a module in the following way:



The module includes the following sections. When you author the materials, each of these sections should be under its own heading and each section is described in more detail below.

- Overview or description of the module
- Introduction to the problem or question
- Statement of the problem or question
- Background information
- Model details
- Solution methodology/ implementation
- Assessment of the model(s)
- Empirical data, if available
- Conceptual questions
- Problems and Projects
- Solutions
- Suggestions to Instructors
- Glossary of terms
- References



Class sessions explore various aspects of the overall question by breaking it down into essential sub-questions. Students work with relevant information through a variety of activities (e.g., in class, in the laboratory, with media, and as homework) to develop an answer to the immediate question. The module should culminate in a product such as a paper, poster presentation, debate, or experiment that provides an opportunity for students to communicate their solution to their peers.

Modules should be flexible so that they can be imported into a wide variety of courses and can accommodate a variety of teaching and learning environments. While some of the models build upon material learned in earlier modules, there should be some modules that are independent.

Everything You Ever Wanted to Know about the Guts of a Module.

Math and Science Level

To facilitate adoption of the materials, it is helpful to indicate the math and science level for each module. Indicate them separately and use the following categories for an undergraduate course of study: Introductory, Intermediate, and Advanced.

Overview or description of the module including the prerequisite knowledge

Consider this an executive summary of the module. Clearly state the goals and objectives of this module for students – write the module goals and objectives in a way that facilitates evaluation of their attainment. Be specific without creating a laundry list of concepts. Identify what the students should expect to learn and, where appropriate, how this module is connected to other computational science modules. This connection should be made explicit so that both students and instructors can begin to generalize to new situations the tools and techniques they learn.

This section should also indicate the assumed prerequisite knowledge needed by the students to complete the module.

As you write this section, consider the following questions:

- Is the course in which I am going to use the module for a general science audience or for specific science majors?
- How many computational science, science, and mathematics courses should students have had prior to this course? Provide a short list of the basic concepts students need before taking the course. This should not be a list of concepts covered in your module.
- Do I envision using the module as a stand-alone component within the course, as an integrated component of the course, or as an add-on to the course?
- Do I intend to use the module in the classroom, in the laboratory, or in both?
- Where in the course do I intend to use the module? (e.g., beginning, end, an intermediate point, or throughout as part of an integrating theme or framework for the course)
- What resource/background material will students need to make sense of the material in the module?
- What knowledge should students have by the end of the module?

Keywords

Include up to five keywords that described the educational module. These may be basic concepts used in the module along with software modeling tools used to implement the module. These keywords are used to index the modules for internet search engines.

Introduction to the problem or question

This is the story. The module question, and its accompanying story line, provides a contextual framework and springboard for guided inquiry and exploration. The module story line is held together by a series of sub-questions. This template provides a simple structure for inquiry that conveys the module story line, its organization, and the direction of the associated inquiry. Variation is expected in how the inquiry is done between and within modules.

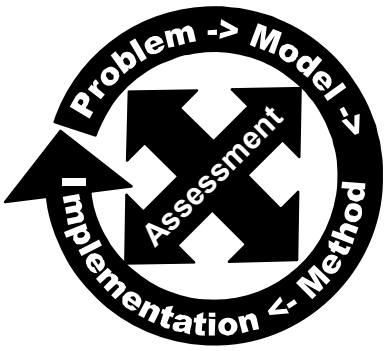


Figure 1: Problem-Solving Paradigm for Course Modules

Statement of the problem or question

The problem or question flows from the story. The problem or question provides a context for understanding and applying specific computational science concepts.

Background information – scientific, mathematical, and computer, where appropriate

This section should provide adequate background for students to follow construction of the model. Some examples:

- When exploring a groundwater model, students must learn about the local geology, become familiar with appropriate terminology, and review the mathematical methods to be used. This module would reference appropriate modules that use similar mathematical models.
- For the module on the spread of disease, students acquire background in epidemiology and the appropriate terminology.
- For the module on brain mapping in a *Computational Neuroscience and Psychology*, students review brain structure and function, needed mathematical concepts (i.e., matrix algebra), and techniques from modules in *Scientific Visualization*.

Explanation of the model(s) used to solve the problem or answer the question

A step-by-step creation and rationale of the mathematical and computational model(s). Include definitions of the variables, interrelationships among variables, and how those relationships are expressed mathematically.

Solution methodology/ implementation

A mathematical analysis, or the solution process, and the selection/ rationale for the appropriate computational technique(s). Use of appropriate software packages for the implementation and visualization of the solution. Authoring of code, where appropriate.

Assessment of how well the model solves the problem or answers the question

Students should understand that a model is only as good as the assumptions that underlie it and the data used to construct the model. To determine the value of the model, students should compare the predicted values of the model with actual data (if available) or with theoretical predictions. Consider employing a variety of techniques for assessing the model. Projects can flow out of the students' assessment as they determine when the predicted values don't match with actual data. Students should consider how the simplifying assumptions affected the model predictions and how the assumptions should be refined to acquire a better fit between predicted and actual values.

Empirical data

Whenever possible, provide sample data, plots and figures, or outline a method for having students collect such data. These data are used to assess the validity of the model(s) that they produce.

Conceptual questions to examine student's understanding of the material

This section includes a selection of questions appropriate for end-of-module and/or end-of-course assessment. Include some in a format that can be easily graded. For example, common student responses to an open-ended question can be converted into the choices for a multiple-choice question, perhaps with a follow-up question asking students to justify the answer they selected. Where possible, include questions with links to other modules.

Problems and projects

Provide a number of practice problems/questions using computational science skills and thinking skills developed in the module. These are likely to be used for homework.

The module ends with a culminating activity, often project-based, for assessment of student learning of computational science concepts and/or scientific thinking skills. Full written reports should be expected for more involved homework and projects. In these reports, students should explain all of the steps of the solution methodology and assessment – this provides an opportunity to sharpen their technical writing skills.

As you develop problems and projects for the module, keep in mind four of the goals of the project:

- To emphasize an interdisciplinary, team-based approach to science problem solving
- To cultivate undergraduates' understanding of the creative nature of computational science
- To improve written and oral communication related to scientific and technical projects
- To facilitate student use of current and emerging computing technologies

Solutions

These are supplied to the instructors who choose to adopt the materials. Keep in mind that some of the conceptual questions, problems, and projects may have more than one answer. These solutions should clearly indicate the steps toward the assigned problem solutions. Include software or model output along with the answer documentation.

Suggestions to instructors for using the module

Include a brief paragraph that emphasizes the importance of planning ahead and of choosing a pathway through the module that is appropriate for students. Demonstrating via a course calendar how to schedule class time and out-of-class assignments facilitates adoption by other institutions. Include a brief description of the course format and how the module fits into the course as a whole. Note that instructors need to generate a syllabus or schedule for their own students; here you provide a few models to convey the need for such a student guide and the range of time periods and approaches possible for a single module. You may provide an annotated list of exceptionally useful materials related to the module: books, articles, web sites, special collections of data, etc. One to two pages – be realistic and choose the few items most useful to instructors.

Glossary of terms

Provide a list of new or important terms with appropriate definitions for students.

References – both cited and for additional reading

Include a list of original references to journal articles, books, reports, and websites required as background reading for the module. Suggest background textbook reading about science, mathematics, or computer concepts covered in the module.

Rubric to Assess Student Work in Computational Science.

	Unacceptable 0	Below Average 1	Average 2	Superior 3	Score
Writing Mechanics	Many spelling and grammar errors	Several spelling and grammar errors	Few spelling and grammar errors	No spelling and grammar errors	
Writing Quality	Incompetent, not college level	Marginal, college level, but below course level	Competent, appropriate for course level	Sophisticated, graduate-level	
Completeness	Missing much information	Missing some information	All basic information is included	Goes beyond assignment's minimum requirements	
Mathematics Correctness	Many errors	Few errors	No errors	No errors and shows depth of understanding	
Science Correctness	Many errors	Few errors	No errors	No errors and shows depth of understanding	
Computing Correctness	Many errors	Few errors	No errors	No errors and shows depth of understanding	
Logic and Flow	No logic to the flow of information	Awkward flow of ideas	Logical flow of ideas	Attempts to tell a story	
Use of Sources (if appropriate)	Has less than 75% of sources	Has at least 75% of sources or not all are cited	Has minimum number of sources	Has more than minimum number of sources	
Interdisciplinary (Math, Science, Computing)	Missing two components	Missing one component	Attempts to incorporate all components	Effectively communicates all components	
Team-effort	One student completed entire assignment	At least one student did not contribute to assignment	Each student contributed, though not equally	Each student contributed equal effort to assignment	
Computing Technology	Technology not used	Technology level below assignment requirements	Technology level appropriate for assignment	Technology level higher than assignment requirements	
Creativity	Does not demonstrate even "textbook" or lecture thinking about project	Does not go beyond "textbook" or lecture thinking about project	Demonstrates novel thinking about project	Effectively and convincingly demonstrates novel thinking about project	

Tips, Tricks, and Traps.

- Developing a module is a dynamic process that may lead to minor or major changes in the initial design. As the team members, outside evaluators, and students review the materials you create, they may suggest refining small or large components of your work. Because the goal is to create the best materials that we can, you should consider the suggestions that others provide a blessing, and not an attack on your work… in other words, go with the flow.
- When including computer code, be sure to fully document the code so that students and less experienced instructors will understand the purpose of the commands.
- Avoid using highly specialized software, particularly expensive packages – dissemination of the materials is greater with software that is either widely available or relative inexpensive.
- Avoid extensive formatting of the documents that you author.
  - Using a common word processing package (i.e., Word) makes it easier for student workers to convert the files to PDF (and any other formats we select) and makes it less likely that information is lost in the conversion process.
  - Use Times New Roman for the font throughout the module.
  - Main titles should be in 18-point font and centered with one additional line both above and below.
  - Subtitles should be in 16-point font and left justified with one additional line both above and below.
  - Text should be in 12-point font, single spaced and left justified paragraphs with a blank line separating each paragraph. Do not indent the first line of the paragraph.
  - Equations should be centered and numbered, if appropriate – use Math Type
  - Include diagrams, graphs, and images in digital form and embedded within the text document
- See **Appendix B** for software used in the Capital University Computational Science program and for additional information concerning web resources on computational science.

Assessment, Assessment, Assessment. (Oh, did I mention assessment?)

Although many faculty cringe at the thought of having to do assessment, for the purposes of this project, you should consider assessment to be an integral component for ensuring high quality materials.

For all modules, two types of evaluation (formative and summative) occur in three overlapping phase: *Phase one*: Developed materials are reviewed by co-PIs within the same discipline or who are creating materials for the same course. *Phase two*: an Evaluation Team of national experts review developed materials. *Phase three*: Developed and reviewed materials are class tested. The purpose of the formative evaluation is to assess the development of the modules (*phases one* and *two*). The purpose of the summative evaluation is to determine the effectiveness of the developed modules (*phase three*). The goal for the assessment is to engage in a reflective conversation with each other, with the outside evaluators and with our students.