Journal of Computational Science 1 (2010) 55–61

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Diagnostics and rubrics for assessing learning across the computational science curriculum

J. Russell Manson, Robert J. Olsen *

School of Natural Sciences and Mathematics, The Richard Stockton College of New Jersey, Pomona, NJ 08240-0195, United States

ARTICLE INFO

Article history:
Received 24 March 2010
Accepted 26 March 2010

PACS:
01.40.Di
01.40.G–
01.40.gb

Keywords:
Computational science education
Computational thinking
Curriculum development
Assessment
Placement diagnostics
Force Concept Inventory (FCI)
Rubrics

ABSTRACT

We describe our experiences with learning assessment in a new computational science program. We report on the development and pilot testing of assessment tools in both core and cognate courses. Specifically, we detail a diagnostic assessment that predicted success in our introductory computational science course with reasonable reliability; we give an account of our use of an existing assessment tool to investigate how introducing computational thinking in a cognate course influences learning of the traditional course material; and we discuss rubric development for project evaluation.

1. Introduction

As an emerging discipline, computational science does not yet have a customary curriculum. Graduate curricula were surveyed first [1] and recommendations have been made for competencies to form the core of an undergraduate curriculum [2,3]. In addition, the way in which coursework is apportioned among existing disciplines and the extent to which courses overlap in the undergraduate curriculum has been analyzed [4]. A handful of textbooks written specifically for undergraduate and computational science courses have appeared [5–9]. The content of newly developed undergraduate courses in computational science depends on determining which core competencies are not being adequately developed in cognate courses already offered in traditional majors. The shortfalls that are identified are met both by distributing the topics in the new courses introduced with the major and by renovating existing cognate courses where possible.

The New Jersey Commission on Higher Education approved a program in computational science at Stockton in February 2006; the entering class of Fall'07 was the first cohort that was able to select the undergraduate major. The second author has taught CPLS 2110 (introduction to computational science) each fall semester since Fall'07 and also taught the course in Spring'07 in preparation for the formal initiation of the major that fall. Enrollments in CPLS 2110 prior to Fall'09 were quite small; the first two undergraduate computational science degrees will be awarded during the 2010–2011 academic year.

The computational science (CPLS) program at Stockton has close ties to the physics (PHYS) program. Therefore an early opportunity to expand the computational content of cognate courses came when CPLS faculty were asked to teach the classical mechanics course that is a standard part of the undergraduate physics major. The course was duly renamed PHYS 3220 (computational mechanics) and taught by the first author in Spring'08 and '09.

The first cohort of degree candidates was admitted into the M.S. component of the program in Spring'10. All students entering with a B.S. degree take CPLS 5100 (introduction to modeling and simulation) in the first semester. CPLS 5200 (scientific visualization) is a required course that will be taught by the second author in Fall'10.

* Corresponding author. Tel.: +1 609 626 5583; fax: +1 609 626 5515.
E-mail addresses: Russell.Manson@stockton.edu (J.R. Manson), Robert.Olsen@stockton.edu (R.J. Olsen).

1877-7503/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.jocs.2010.03.012

2. Motivation

2.1. The need for assessment: CPLS 2110

The major will take root to the extent that science, mathematics, and computer science majors see the introductory courses as electives that add value to their curricula. In other words, computational science programs must embrace the role of providing service courses to larger, longer-established majors. This is neither new nor unique. Chemistry serves biology, physics serves chemistry and biology, and mathematics serves physics, chemistry and biology.

In the Fall'08 semester, we made a concerted effort to expand the audience of CPLS 2110 beyond CPLS majors. We replaced first-semester calculus as a co-requisite with pre-calculus as a prerequisite at the request of colleagues in the environmental science program. Subsequent discussions among the CPLS faculty highlighted the importance of developing an assessment tool that would give an early warning to students needing to remediate basic mathematics skills.

2.2. The need for assessment: PHYS 3220

PHYS 3220 focuses on Newtonian mechanics at a medium to advanced level. In Spring'08 the first author introduced a sequence of computational projects involving mechanics problems of increasing difficulty. The projects ranged from modeling projectile motion to modeling motion of connected multiple rigid bodies (a complete list is given in Table 1).

Does renovating an existing course by taking a computational approach detract from the traditional content, which is after all the *raison d'être* of the course? This question motivated a study utilizing the Force Concept Inventory (FCI) [10], a well-established diagnostic tool that assesses mechanics knowledge, which we used as an anonymized pre- and post-test. The pre-test results alert the instructor to misconceptions about Newtonian mechanics held by the class as a whole. Comparison of pre- and post-test results was used to determine whether student understanding was affected by the increased emphasis on computational thinking.

2.3. Embedded assessment: a pancurriculum rubric

CPLS faculty make extensive use of projects in both CPLS and cognate courses. Project work reflects current best practice in computational science education [2,3]. Reports for computational science projects tend to follow a standard model across the curriculum regardless of who teaches the course. Broadly speaking, they include an introduction, model description, model testing, model application and a conclusion. The first author decided to develop a rubric for computational science projects for use in PHYS 3220.

Concerns about the granularity of both the categories and the scoring scale of the rubric have led the authors to collaboratively design a more robust rubric which could be used across the computational science curriculum. A decided benefit of a pancurriculum rubric is that it reinforces a student's computational skills through consistent emphasis on core competencies throughout the four years of undergraduate study.

3. Methods

3.1. Tools for assessment: CPLS 2110

The second author designed a brief assessment of basic skills that was administered on the first day of the Fall'09 semester. Questions gauged geometric understanding of the meaning of the slope and y-intercept of a straight line; recognizing common functions ($mx+b$, $\sin x$, and e^x) when plotted as data sets as they might appear if obtained as experimental results (i.e., with noise); and associating a (global) minimum, a (local) maximum, a point with positive slope, and a point with negative slope on the graph of a function with labels describing the rate of change as smallest in magnitude, largest in magnitude, positive and negative.

On the first and last days of the semester students completed a survey, shown in Table 2, aimed at ascertaining their experience with and attitudes toward computing. Questions are paired in the survey, asking first about coursework in general and second about math and science coursework in particular. Responses were on a seven point Likert-style scale, with 1, 4 and 7 corresponding to never, sometimes and regularly, respectively.

3.2. Tools for assessment: PHYS 3220

The FCI [10] is a tool developed to help assess facility with and misconceptions about Newtonian thinking as a way to explain motion and its causes. It consists of 30 conceptual questions in multiple choice format and has been extensively studied and promoted [11]. Given its widespread use, the first author chose to administer the FCI to see whether undertaking the aforementioned computational projects (Table 1) was fostering the ability to apply Newtonian thinking to problems in mechanics. We expected that implementing computational models of various mechanics problems and analyzing the results with graphing and visualization tools (e.g., movies in MATLAB) would be an aid to understanding the mechanics concepts.

The FCI was administered twice in Spring'09, once at the outset of the course and again at the end. Results of the FCI were not included in the final grade, so the assessment was of low stakes. To further alleviate test anxiety, the FCI was administered in such a way that the instructor was able to pair the results without identifying individual students.

Table 1
Topics of the projects in PHYS 3220 and the associated computational concepts.

Topic	Computational concept
Motion of a projectile	Solving ODEs using Euler's method
Motion of an object sliding down an inclined plane with friction and bouncing against a spring	Limitations of Euler's method and solving ODEs using Runge–Kutta methods
Motion of a satellite orbiting Earth	Long-time accuracy of numerical ODE solvers
Motion of a rigid body	Multiple model realizations for optimization and simulation
Motion of multiple connected rigid bodies	Solving stiff DAEs

Table 2
Survey questions about experience with and attitudes toward computing. Odd-numbered questions Q1, Q3, etc. omit the words in parentheses; even-numbered questions include them.

(Q1, 2)	How often have you used a computer when doing an assignment in a (math or science) course?
(Q3, 4)	How often have you used spreadsheet (e.g., Excel) software in a (math or science) course?
(Q5, 6)	How often do you use a computer, even if it is not required, to do an assignment in a (math or science) course?
(Q7, 8)	How often have you found that using a computer helped you understand a concept in a (math or science) course?