Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 1 (2010) 857–865
Procedia Computer Science
www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

Assessing and refining an undergraduate computational science curriculum

J. Russell Manson[1], Robert J. Olsen[1]

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

Abstract

We describe our experiences with curriculum development and learning assessment in a new undergraduate 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 developing a pancurriculum rubric for scoring computational science projects.

Keywords: computational science education, computational thinking, curriculum development, assessment, placement diagnostics, Force Concept Inventory (FCI), rubrics
PACS: 01.40.Di, 01.40.G-, 01.40.gb

1. Introduction

As an emerging discipline, computational science does not yet have a customary undergraduate curriculum. Progress has been made in identifying core competencies [1, 2]. In addition, the way in which coursework is apportioned among existing disciplines and the extent to which courses overlap has been analyzed [3]. A handful of textbooks written specifically for computational science courses have appeared or are scheduled to appear soon [4, 5, 6, 7, 8]. The content of newly developed 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 major in computational science at Stockton in February 2006; the entering class of Fall '07 was the first cohort that was able to select the major. The second author has taught CPLS 2110 (Introduction to Computational Science) each fall semester since Fall '07 and also taught the

Email addresses: Russell.Manson@stockton.edu (J. Russell Manson), Robert.Olsen@stockton.edu (Robert J. Olsen)
[1]The authors would like to acknowledge the support of the U.S. Department of Education (FIPSE) for this work through grant award P116Z080098.

1877-0509 © 2010 Published by Elsevier Ltd.
doi:10.1016/j.procs.2010.04.094

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.

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 pre-requisite 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).

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

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) [9], a well-established tool that assesses mechanics knowledge, to look at 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 [1, 2]. 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.

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?

3.2. Tools for Assessment: PHYS 3220

The FCI [9] 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 [10]. 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.

3.3. Tools for Assessment: A Pancurriculum Rubric

The aforementioned rubric was first used in PHYS 3220 in the Spring '09 semester and contained six categories (accuracy, learning and understanding, narrative, introduction, analysis and conclusion). Each category was assigned a number from 0 to 3, giving a total score that ranged from 0 to 18. Despite providing a somewhat more objective way of assessing computational science projects, the rubric was often found to be too coarse-grained. Since the second author was assigning projects in CPLS 2110, the idea of developing a robust pancurriculum rubric arose.

The authors worked together to modify the original rubric, subdividing the categories to provide a more fine-grained instrument. The modified rubric had 18 categories evaluated on a 0–10 scale, giving a maximum possible score of 180. Table 3 contains a sample category. To test the modified rubric the authors created a pool of ten exemplary projects, five each from the most recent instances of CPLS 2110 (Fall '09) and PHYS 3220 (Spring '09). This pool was then graded by both authors.

Table 3: Example rubric category. The scale is labeled by the category name and followed by guidelines describing excellent (9–10, A range), good (7–8, B range), satisfactory (5–6, C range), and poor (0–4, D and F range) work.

model construction	(10)	(9)	(8)	(7)	(6)	(5)	(4)	(3)	(2)	(1)	(0)

The flow of information in the model is easily followed in an excellent project, is followed with some effort in a good project, is followed only with significant effort in a satisfactory project, and is followed only with considerable difficulty in a poor project.

4. Results and Discussion

4.1. Assessment Outcomes: CPLS 2110

14 of 17 students (82%) answered questions about slope and y-intercept of a line correctly. Given "noisy" data following exponential, linear, and sinusoidal trends and equations for each function, 14 of 17 students (82%) correctly matched the exponential data and function, 13 of 17 students (76%) correctly matched the linear data and function, and 16 of 17 students (94%) correctly matched the sinusoidal data and function. Of the seven students who answered one or more of these questions incorrectly, three withdrew from the course almost immediately, one withdrew early in the semester, and the remaining three obtained the three lowest course grades.

On the question involving rates of change and slope at a point on a curve, 13 of 17 students (76%) identified the minimum with a slope of smallest magnitude and 2 of 17 students (12%) identified the maximum with a slope of smallest magnitude. Just 4 of the 13 students selecting the minimum realized that the maximum should be likewise identified, and neither of the two students selecting the maximum realized that the minimum should be likewise identified. Although the question clearly stated that more than one point could be associated with a label, it is likely that habit built from years of test taking caused students to answer this question quickly rather than carefully. Redesigning the question to ask students to rank the rate of change at each point may provide better information. 10 of 17 students (59%) correctly identified the function as having a positive rate of change at the point with positive slope and a negative rate of change at the point with negative slope. Several incorrect responses appear to be due to differences between the slopes at the labeled points being insufficiently pronounced; a graph with more distinct features will be used in subsequent versions of the assessment.

Figure 1 demonstrates that course grade correlates well with the score on the diagnostic assessment. One student, represented by the open circle, fared considerably worse in the course than performance on the diagnostic would imply. The diagnostic provides relatively little resolution at the top end of the scale, as indicated by the clustering of points at diagnostic scores above 90. The self-assessments revealed by the survey of Table 2 were not positively correlated with course grade.

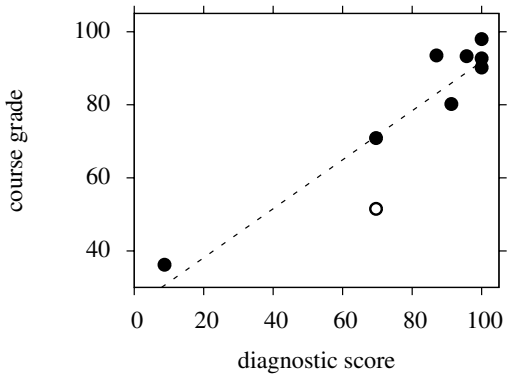

Figure 1: Course grade versus diagnostic score. There is a strong correlation ($r = 0.91$) between the diagnostic score and the course grade, indicating that the diagnostic is a useful predictor of success in CPLS 2110.

4.2. Assessment Outcomes: PHYS 3220

We present the sorted differences between after and before FCI scores in Figure 2. Nine pairs of tests were collected. Although two students appear to have diminished facility with Newtonian thinking, the majority (seven) appear to have benefited from the course and from the computational projects. A paired *t*-test indicates that the improvement is significant at the 90% confidence level. As used here, the FCI indicates that including computational projects in a mechanics course (a computational science cognate) does not detract from learning mechanics concepts and appears to help most students.

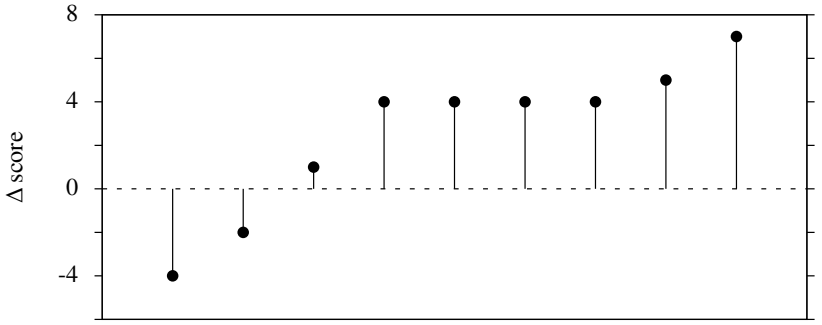

Figure 2: Change in FCI score. Nine students completed the FCI assessment at the beginning and end of PHYS 3220 in Spring '09. Seven of the nine scores increased; the difference in scores (before – after) is significant at a 90% confidence level according to a paired *t*-test.

The small sample size is certainly of statistical concern; however, this is a problem for computational science assessment not only now, owing to low enrollments in courses in this new field, but also likely in the future, owing to the relatively small class size imposed by the necessity of teaching in computer classrooms. Further work is required to tease out how much understanding is gained from the computational project work versus other course activities.

4.3. Assessment Outcomes: Test-driving the Pancurriculum Rubric

We are not aware of any other pedagogical studies in computational science wherein two individual faculty members compare their grading of the same student material in a systematic way and this makes the results particularly interesting. Considering that students in the two courses were given independent guidelines for their projects without

reference to the rubric, the correlation ($r = 0.63$) between the total scores we assigned to the ten projects is reassuring. We selected the same project as best and identified the same two projects as least accomplished. Moreover, we concurred on four of the top five projects and four of the bottom five projects. We evaluated two projects near the middle of the score distribution rather differently. The correlation between our rankings is fairly strong ($r = 0.76$), which is significant at the 98% level ($p = 0.02$; two-tailed t-test). This lends support to the notion that experienced teachers of computational science can spot good and bad projects "at a hundred paces".

When scores for all categories and all projects are compared in a two-tail t-test, the p-value is 0.19. This indicates that we cannot conclude with confidence that the scores are drawn from different populations, i.e., there is no significant difference overall in the grades we assigned when all categories are integrated. This provides some justification for the idea of different instructors using this project rubric in different computational science courses. As noted previously, the advantage of such a policy is that students find best practices and skills for their discipline being reinforced over the four years of their computational science education.

Delving more deeply into the data reveals some differences. We consider individual categories of the rubric for which our scores are not well correlated as having ambiguously defined criteria. The refinements that have resulted from this closer analysis are shown in Figure 3 and Table 4 in the following section.

5. Next Steps

The process of developing a computational science curriculum and the supporting assessment tools has been a thought-provoking and illuminating experience, and we encourage others in the wider computational science community to undertake similar projects if they have not already done so. Definite next steps for each of the three initiatives described in this manuscript are already underway.

(1) The diagnostic assessment developed for CPLS 2110 shows promise as an indicator of preparedness for beginning study in computational science. Reflecting on student work from the Fall '09 semester, facility with units seems to be a discriminator that is not currently part of the diagnostic. Before using the diagnostic in Fall '10, we will incorporate this topic and make the modifications mentioned in section 4.1.
We intend to use the CPLS 2110 diagnostic as the foundation for a series of assessments of increasing sophistication for use in all of our courses, including those at the graduate level. The development and testing of this computational thinking inventory will be a large component of our future work.

(2) The use of the FCI assessment tool in PHYS 3220 has yielded some interesting, albeit preliminary, results. Some concepts the tool examines are more closely tied to the computational projects in the course than others. We will lump the questions into groups by concept and analyze the results at this medium-grained level with a view toward extracting more information about the influence of computational thinking on cognate learning.

(3) We plan to further test and refine the pancurriculum rubric and advocate for its adoption in all CPLS courses and in any cognates for which it is appropriate. The first author will be teaching PHYS 3220 and the second author will be teaching PHYS 3352 (Nonlinear Systems), another cognate course, in Spring '10. Projects will be an integral part of both courses. We will include the rubric as part of the instructions to our students for their projects and we will grade the projects jointly (i.e., we will effectively be team-teaching these courses with regard to project evaluation). We will gladly provide the current version of the rubric to anyone in the computational science community who wishes to contribute to its further testing and refinement.

Our most immediate efforts involve sharpening the use of the FCI and testing the pancurriculum rubric. Table 4 contains the categories of the refined pancurriculum rubric and the criteria by which each category is evaluated. Most categories are rated from excellent (A) to poor/failing (D/F), with excellent corresponding to 9–10 and poor/failing to 0–4. Exceptions are "distinctive features" and "degree of difficulty". We include them to give the rubric a bit of open-endedness so that it does not become merely a checklist when used by students. We envision using these two categories to distinguish good from excellent work.

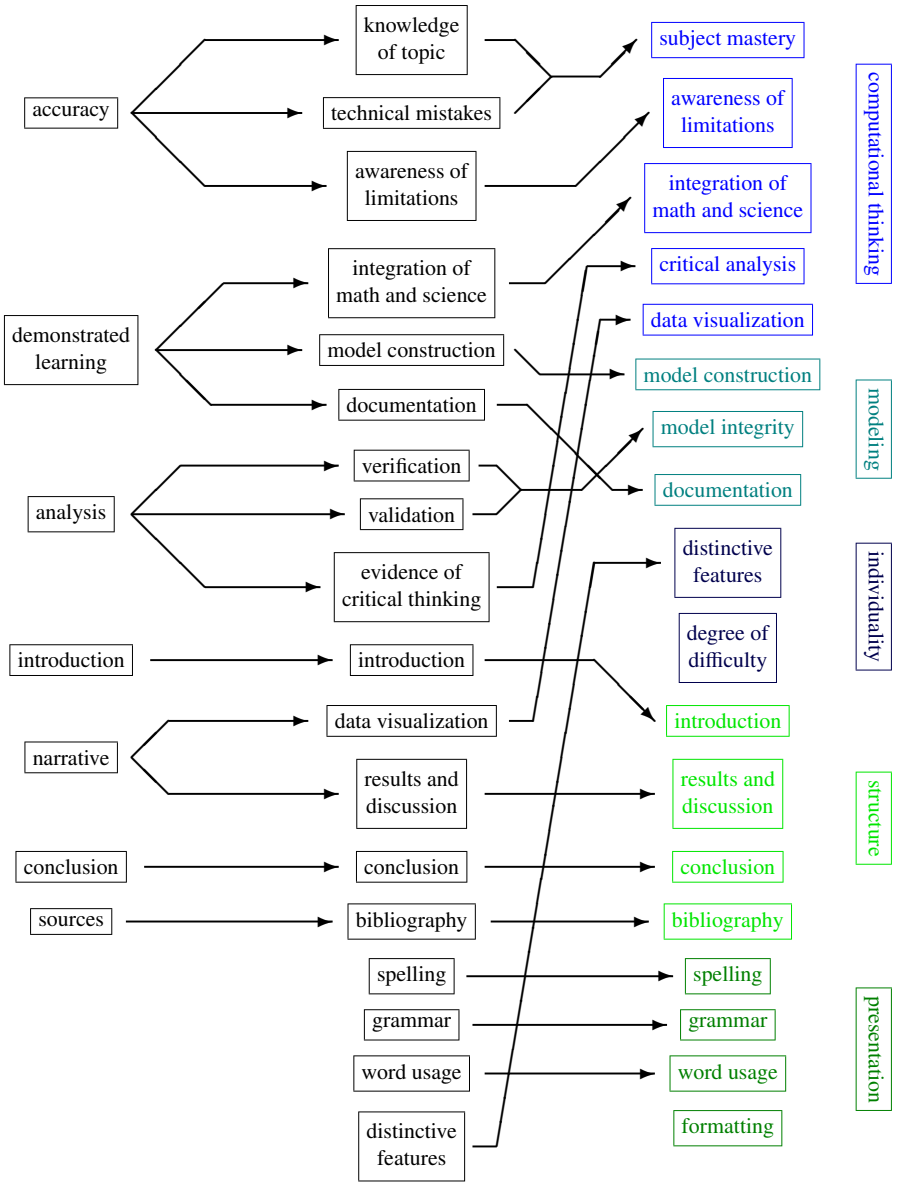

Figure 3: Evolution of the pancurriculum rubric. Categories are arranged in columns corresponding to the original rubric (first column, Sections 2.3 and 3.3), the expanded rubric (second column, Sections 3.3 and 4.3), and the rubric after further refinement (third column, Sections 4.3 and 5). Arrows between columns show the elaboration and merging of categories as the rubric evolved. Categories in the current rubric are constellated into the supercategories to the far right.

Table 4: Pancurriculum rubric categories and evaluation criteria.

subject mastery	An excellent project demonstrates complete mastery of the subject material and has no technical mistakes. A poor project has major conceptual misunderstandings or several technical errors.
awareness of limitations	Full awareness of the limitations of both the model and the methods is evident in an excellent project; a list of situations in which the model should not be used is included. At most, a poor project mentions only briefly limitations of either, but not both, of the model or the method.
integration of math and science	An excellent project links, through equations and narrative, the mathematical and physical concepts. A poor project makes at best a minimal attempt to link the mathematical and physical concepts.
critical analysis	In an excellent project all results are scrutinized in view of known limitations of the model and method; results are neither oversold nor undersold. One or more results are accepted unquestioningly in a poor project.
data visualization	Figures in an excellent project have accurately labeled axes, an appropriate choice of point markers and line styles, clearly distinguished data sets, accurate captions that highlight the most distinctive features of the data, effective use of color, minimal whitespace, and minimal clutter. Many aspects of the figures in a poor project can be improved. Figures with unlabeled axes or multiple data sets that are not distinguished ensure that this aspect of the project will be rated as poor.
model construction	In an excellent project, the rationale behind the model and how the model works is lucid and unambiguous. In a poor project, it is not clear how or why the model works, even if it does provide plausible answers.
model integrity	The accuracy of the model as well as its fidelity to applicable scientific laws, exact solutions and mathematical or experimental benchmarks is demonstrated in an excellent project. The model is shown to produce accurate results in no more than the simplest of cases in a poor project.
documentation	The documentation accompanying an excellent project allows another modeler with similar experience to use or modify the model after reading the documentation. A poor project does not describe how to use the model or has uncommented code.
distinctive features	Any of several features distinguish an excellent project from a good project. Examples of such features include especially thorough analysis of model integrity, particularly effective figures, unusually insightful discussion of model limitations, perceptive identification of further questions that the model might be used to answer, and identification of modifications required to make the model more broadly applicable.
degree of difficulty	Any of several traits differentiate a difficult project from an easy project. Projects that involve investigating a system of greater complexity than that of a typical homework problem, comparing results obtained from more than one method, or using a method not discussed in class all qualify as difficult. A project that requires effort equivalent to a homework problem is an easy project.
introduction	An excellent introduction engages the reader even if he or she is not knowledgeable about the problem at hand. A poor introduction deters the reader from reading further.
results and discussion	The results and discussion section of an excellent project guides the reader through the key results and explicitly refers to the figures and tables in support of the analysis; in a poor project, key results are omitted from the results and discussion section or no reference is made to the figures and tables.

Table 4: Pancurriculum rubric categories and evaluation criteria (continued).

conclusion	The conclusions section of an excellent project accurately and concisely summarizes the key results and includes cross-references to relevant parts of the results and discussion section. The conclusions section of a poor project gives an inaccurate account of the results.
bibliography	The bibliography of an excellent project contains accurate references to the specified number of authoritative sources. The bibliography of a poor project consists of only the course text.
spelling	An excellent project contains no errors that spellchecking the document would detect and no more than two spelling errors per page of text. The spelling errors in a poor project are so numerous that the reader is distracted from the content.
grammar	An excellent project contains no more than one grammatical error per page of text. The grammatical errors in a poor project are so numerous that the reader is distracted from the content.
word usage	In an excellent project, word use is apt and enhances the presentation. Words are used incorrectly and phrasing is immature in a poor project.
formatting	In an excellent project, equations are typeset and figures and tables are integrated at appropriate points in the manuscript. Handwritten equations or figures and tables gathered at the end of the manuscript are characteristic of a poor project.

References

[1] Ralph Regula School of Computational Science. Summary of undergraduate minor program requirements [online] (2006) [cited 7 Jan 2010].
[2] SIAM Working Group on CSE Undergraduate Education. Undergraduate computational science and engineering education [online] (2006) [cited 8 Jan 2010].
[3] O. Yasar, R. H. Landau, Elements of computational science and engineering education, SIAM Review 45 (4) (2003) 767–805.
[4] R. H. Landau, A First Course in Scientific Computing: Symbolic, Graphic, and Numeric Modeling Using Maple, Java, Mathematica, and Fortran90, Princeton University Press, 2005.
[5] A. B. Shiflet, G. W. Shiflet, Introduction to Computational Science: Modeling and Simulation for the Sciences, Princeton University Press, 2006.
[6] G. Strang, Computational Science and Engineering, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, U.S.A., 2007.
[7] R. H. Landau, J. Paez, C. C. Bordeianu, A Survey of Computational Physics: Introductory Computational Science, Princeton University Press, 2008.
[8] C. F. Van Loan, D. K.-Y. Fan, Insight Through Computing: A MATLAB Introduction to Computational Science and Engineering, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, U.S.A., 2010.
[9] D. Hestenes, M. Wells, G. Swackhamer, Force concept inventory, The Physics Teacher 30 (3) (1992) 141–158.
[10] D. Hestenes, I. Halloun, Interpreting the FCI, The Physics Teacher 33 (8) (1995) 502.