EXPLORING UNDERGRADUATE STUDENTS' COMPUTATIONAL LITERACY IN THE CONTEXT OF PROBLEM SOLVING

Camilo Vieira, Alejandra J. Magana
Department of Computer and Information Technology
Purdue University

Anindya Roy, Michael L. Falk, and Michael J. Reese Jr.
Department of Materials Science and Engineering
Johns Hopkins University

Abstract

This paper evaluates undergraduate students' performance during a problem-based computational science course in a materials science and engineering program. The course guides students to apply computational tools and methods to solve problems in materials science and engineering. The study assesses the relationship between phases of the problem-solving process and computational literacy skills in the context of MATLAB® computational challenges. Students complete five projects that require combined problem-solving skills and computational skills. Results suggest that aligning computational challenges with problem solving phases can support student learning and computational literacy skills development. The findings also suggest that different computational challenges require different forms of support for the learners to successfully complete the problem solving process.

Introduction

Computational Science and Engineering (CSE) has emerged as an important tool to solve complex engineering problems [1]. Engineers need an ability to use computational tools, integrated with strong problem-solving skills, to tackle complex problems [2-4]. For example, in Materials Science and Engineering, a sub discipline called Computational Materials Science [5] has been established. This trend is reflected in educational settings too --- there has been a call to integrate computational tools and methods into different disciplinary engineering curricula sooner and often [6]. Aligned with this idea, the department of Materials Science and Engineering at Johns Hopkins University started a novel computational course for its undergraduate students, titled Computer Programming for Materials Science and Engineers (CPMSE).

In this study, we investigate how students in the CPMSE course applied computational tools and methods to solve materials science and engineering problems. We seek to answer the following questions in connection to CPMSE:

What are students' performances on disciplinary computational challenges when configured as problem solving phases?

How does students' performance on specific phases of the problem-solving process on different projects relate to the other phases, and to student overall performance?

How do students' problem-solving approaches relate to their computational literacy skills?

Literature Review

Many current engineering curricula are designed to introduce computation in an isolated way from the disciplinary core courses. Students enrolled in such programs acquire the disciplinary knowledge and the computational knowledge separately. It is not clear whether they would know how to apply these together [6]. Educators proposed several approaches to better integrate the two [1]. Some of these include creating individual courses as part of computational concentrations, creating interdisciplinary collaborative project courses, using small add-on courses to supplement existing math or science courses, and introducing computational concepts through tools such as computer graphics.

These curricular practices rely on different pedagogical strategies. Some of the most effective techniques in the context of programming and CSE courses include the inverted classroom approach [7, 8, 9], pair programming [10], worked-out examples for introducing complex learning contexts to novice learners [11], project-based and problem-based learning in collaborative settings [7], and so on. Research studies investigating these pedagogical strategies sought to measure student performance or student perceptions. However, it is not clear how these pedagogies work, and the relevant aspects that support the development of student computational skills in the context of disciplinary problems.

This study will explore the relationship of the use of computational tools and methods as part of specific steps in the problem-solving process. We will also investigate how this practice relates to student achievement of the learning outcomes.

Theoretical Framework

In this study we explore the importance of computational literacy that enables the students to solve materials science problems. The theoretical frameworks underpinning this study are computational literacy [12] and problem solving [2]. Computational literacy refers to an understanding that goes beyond just using a computer and its components. The problem-solving process requires the conceptual and the procedural knowledge along with the problem states [2] to be intimately connected. These connections are made by creating different representations of the problem --- verbal, mathematical, computational and visual representations of a phenomenon in the present context. A mastery of computational literacy would enable students to create and manipulate computational representations to learn scientific phenomena [12]. Students' strategic knowledge [13] dictates their choice and understanding of representations, and the quality of these representations will, in turn, determine their ability to solve a problem. Students build or use these representations while they are solving a disciplinary problem using computational tools.

In this study we understand the problem-solving process using the Integrated Problem-Solving (IPS) [2] model, and adapt this model to include the concept of computational literacy. The first of three phases is the problem recognition. Here, the student will understand the problem and create a plan to solve it. The student will use verbal and mathematical representations for that purpose. On a second phase called problem framing, the students execute the plan creating computational representations of the phenomenon (i.e., the implementation of the model). Finally, on the problem synthesis phase, the students will complete the plan by evaluating the solution. They will use computational, visual, mathematical, and verbal representations to ensure the solution is correct. The Methods section contains a detailed description of how we implement IPS. We argue in this work that computational literacy can be acquired more effectively when we introduce computational tools and methods in the context of solving disciplinary engineering challenges.

Methods

The Course

The CPMSE course was designed using the How People Learn framework [14]. It is knowledge centered, learner centered, and community centered. It uses MATLAB as the programming environment and the learning objectives are [7]:

(1) Write MATLAB programs to execute well-defined algorithms.
(2) Design algorithms to solve engineering problems by breaking these into small tractable parts.
(3) Model physical and biological systems by applying linear systems and ordinary and partial differential equations.

In this course we employed an inverted classroom approach where students are required to watch recorded lectures before coming to class. During the class time they focused on practice exercises. The students were required to complete five computational projects related to their core courses, which counted for about 52.5% of the final grade in CPMSE. The disciplinary and computational learning outcomes for each of the projects are described in Table 1. A brief description of the assignments is included as Appendix A.

Project evaluations were organized based on IPS [2] --- which divides the problem-solving process into three distinct phases, as described in the previous section.

Other activities that contributed to the final student course score included: (1) 17 quizzes, (2) two exams, and (3) a final project.

Table 1: Computational and disciplinary learning outcomes per project.

	Computational Learning Outcomes	**Disciplinary Learning Outcomes**
Project 1	The student demonstrates the ability to apply the techniques of modeling and simulation to a range of problem areas. The student uses MATLAB to create visual displays of data, including graphs, charts, tables, and histograms.	The student graphically represents and calculates the phases present in a binary phase diagram.
Project 2	The student implements algorithms for solving differential equations. The student models biological systems by applying linear systems and ordinary and partial differential equations	The student models the progress of HIV infection in a patient that is being treated with a drug of a given effectiveness.
Project 3	The student demonstrates the ability to apply the techniques of modeling and simulation to a range of problem areas. The student uses MATLAB to create visual displays of data, including graphs, charts, tables, and histograms.	The student models crystal structures cleaved along a plane and generates a three-dimensional representation of them.
Project 4	The student models biological systems by applying linear systems and ordinary and partial differential equations. The student demonstrates the ability to apply the techniques of modeling and simulation to a range of problem areas.	The student simulates the cardiac tissue and the ventricular fibrillation process.
Project 5	The student demonstrates the ability to apply the techniques of modeling and simulation to a range of problem areas.	The student creates and interprets a molecular dynamics simulation in terms of kinetic energies.

Participants

Twenty-three freshmen and sophomore material science and engineering students completed the CPMSE course in spring 2014, and participated in the study. The five computational projects the students solved were analyzed on a rubric (Table 2) designed to explore the relationships of computational literacy with the three phases of IPS [12].

Data Collection and Data Analysis

According to IPS, the three phases of problem solving are: (1) problem recognition; (2) problem framing; and (3) problem synthesis. Note that the IPS framework [2] is originally presented for solving physics problems using free-body diagrams and mathematical equations. In that context, solving the mathematical equations would determine the completion of the process. Projects in CPMSE are more involved in comparison, and the use of mathematical representations is part of the *problem recognition*; in fact, most of the projects provided the mathematical representation as part of the problem statement.

The *problem framing* phase comprises the implementation of the code. It is important to highlight that these projects are not simple programming projects. Hence, the problem is not solved once the program is built. In addition to writing codes, the students need to be able to run their codes and check the output against test cases --- to evaluate how appropriate their programs are to answer questions based on disciplinary problems. This last requirement --- their ability to demonstrate disciplinary understanding --- forms the *problem synthesis* step.

A priori, we expect computational literacy to overlap with all of the three phases of problem solving as defined here. We hypothesize that a correlation of computational literacy to different phases will be found. In order to explore the interdependence, we designed a rubric with two distinct components: (1) problem solving phases; and (2) computational literacy (see Table 2). The five student projects were evaluated using this rubric. The student projects were graded on the rubric criteria on a scale of 0-10, and then each criterion was converted to a score of 100. We present the resulting descriptive statistics in Table 3. We then calculated Pearson correlation between each two criteria of the rubric, along with the project scores and the final course grades, for all five projects separately. We interpret a weak Pearson correlation coefficient to be 0.1 or lower; a moderate correlation to be between 0.25 and 0.45; and, a strong correlation coefficient is taken to be 0.5 or higher [15].

Results

Table 3 depicts average scores for all projects grouped by the four different criteria. A positive performance for each criterion was set as 70% or above. Students obtained the highest scores for projects two and four, and the lowest score was recorded for Project 1. The students obtained high scores in the implementation phase for all the projects except project one. The problem synthesis phase of project 1 was found to be low-scoring as well. This result suggests that students may have taken some time to acquire these skills, but later on, they were able to use their skillsets in different disciplinary contexts. An alternative explanation is that, after the first project, students better understood what was expected of them. Results also suggest that students had difficulties in the recognition of the last two problems.

Note that some of the scores do not have a standard deviation, and therefore it was not possible to identify a correlation for those items in the individual projects. Hence, these cells are marked with ‘N/A’ in the correlation tables.

In the correlational analysis of individual projects (Appendix B), there are similar patterns of correlations for projects 1, 3, and 5, while project 2 and 4 behave more like each other. The main difference between these two groups of projects is the strong correlation found between problem framing and problem synthesis for the first group of projects ($r > 0.6$).

Table 2: Rubric for Project and Application CPMSE –
All scoring descriptions are not shared due to space limitations.

Criterion	Description	Poor (0-2)	…	Excellent (9-10)
Problem Recognition (10%)	Evaluates the student's plan for completing the project. Student instructions: **Summarize** the nature of the algorithm briefly, identifying the most relevant information from the project description. **Articulate** a well thought-out strategy for designing, coding, testing and debugging your work.	- No strategy is articulated for the design, coding, testing or debugging.	…	- All four areas (designing, coding, testing, debugging) are addressed clearly in the context of the project. - The summary references the project description and identifies relevant aspects of the project. - The strategy is articulated clearly and is logical and well thought-out.
Problem Framing / Implementation (40%)	*Coding style (10%)* Measures the extent to which the code is presented in a manner that is clearly readable by others. Is the code indented, commented and are variable and function names chosen to enhance readability? Does the code appropriately deploy language capabilities to avoid redundant structures, global variables and unnecessarily lengthy blocks of code?	- Code is entirely uncommented. - Global variables are used without justification due to exceptional circumstances. - Code is not differentiated into functions or m-files; i.e. spaghetti code.	…	- Code is well commented. - Code is properly indented and variable and function names are well chosen. - Code is well structured.
	Program execution (30%) Evaluates the extent to which the program functions in a way that conforms to specifications. Does the program execute? Is the input and output of the expected form?	- Program does not run at all.	…	- Program is free of syntax errors that impede execution. - Program takes the expected input parameters and returns the expected output as required in the specification in all respects.
Problem Synthesis (30%)	Evaluates the degree to which the solution satisfies the specification. Is the solution accurate and robust? Does it conform to the problem specifications regarding format, order and presentation?	- The solution produces wholly incorrect output under all of the tests run.	…	- The solution produces correct output in all cases with only minor exceptions. - All output meets specifications regarding format, order and presentation.
Computational Literacy (20%)	Evaluates whether the student can use the solution to approach a disciplinary problem. Can the student use their code to address the disciplinary issue or to solve a related problem?	- No solution provided.	…	- A solution is provided that is correct, clear and well documented.

Table 3: Descriptive Statistics of student overall scores for each computational project.

	Problem Solving Phases					
	Problem Recognition	**Problem Framing / Implementation**		**Problem Synthesis**	**Computational Literacy**	**Total**
		Coding Style	**Execution**			
Project 1 (N=21)						
Mean	87.81	84.05	66.81	59.20	100	74.99
Std. Dev.	22.49	20.36	31.21	30.47	0	19.11
Project 2 (N=21)						
Mean	100	100	96.10	98.43	96.62	97.68
Std. Dev.	0	0	9.05	3.84	4.79	3.14
Project 3 (N=21)						
Mean	72.24	86.29	91.57	76.05	100	86.14
Std. Dev.	30.36	21.18	22.76	25.41	0	14.14
Project 4 (N=21)						
Mean	68.95	98.24	98.29	97.57	89.90	93.46
Std. Dev.	44.38	2.82	5.71	5.76	22.10	8.21
Project 5 (N=20)						
Mean	69.5	94	91.25	77.40	75.95	81.72
Std. Dev.	45.55	6.05	17.63	16.44	33.75	17.84

For the latter group of projects (i.e., projects 2 and 4), this coefficient depicts a weak correlation ($r < 0.2$). Note that the problem framing phase includes programming skills --- suggesting that, at least for certain type of projects, good programming skills support students' ability to evaluate their work. This leads us to ask why the same programming skills did not enable them to evaluate the solution for projects 2 and 4.

We hope to shed light on this in the next section as we discuss project characteristics in detail.

The projects with available correlation analysis of computational literacy with the problem solving phases show a range of correlation values: from weak ($r=0.153$) to strong ($r=0.529$). The correlations for the problem recognition phase with the other two phases also showed a broad spectrum: strong for project 5 ($r>0.61$), a moderate one for projects 1 and 4 ($0.2<r<0.5$), and weak for project 3 ($r<0.2$). Finally, the course final grade had a positive strong correlation with the average project score and each project score individually ($r>0.5$). Only for project 4, students score was not related with their overall performance of the course ($r=0.187$).

Discussion and Conclusions

This study explored the implementation of a problem-based computational science course for materials engineering. The course employed an inverted classroom approach [5] and organized the different projects using the Integrated Problem-Solving (IPS) [2] model. Specifically, it assessed how it is possible to define student performance through the relationship among phases of the problem-solving process, and computational literacy skills.

What are students' performances on disciplinary computational challenges when configured as problem solving phases?

Student performance was considered to be positive for all the project scores and for most of the individual rubric criteria. This result suggests that organizing the projects to follow the problem solving phases can be effectively used as a form of process scaffolding for challenges in computational science and engineering.

Students struggled to understand project 4 and project 5, as evident from the low score in the problem recognition phase. This could be attributed to the design of the course, where projects became progressively more challenging as the semester advanced. The last two projects consisted of modeling complex systems (heart tissue and atoms in a material, respectively). In both these projects the students programmed a set of rules to reflect complex system behavior. While they struggled to understand projects 4 and 5, they could still implement a solution. They were also able to infer conclusions using their code, a result that suggests that students were able to obtain an understanding of the system-level behavior.

The problem framing and problem recognition scores in project 1 were not high, and student performance for these two phases increased for the rest of the projects. We considered two hypotheses here: (1) students were not very comfortable solving computational challenges but their confidence increased as they completed the first project; and (2) after implementing the first projects they developed a better sense of what was expected from the projects.

How does students' performance on specific phases of the problem-solving process on different projects relate to the other phases, and to student overall performance?

For the individual projects, different correlation patterns were found. For projects 1, 3, and 5, problem framing and problem synthesis were strongly related. These two problem solving phases were also strongly correlated to the overall project score and final course grade. This result suggests that students who adequately followed the problem solving process, performed better than those who did not. This is a direct implication for instructional design, reinforcing the idea that setting up projects so that students are required to go through the problem solving process can help them to solve computational challenges.

However, these relationships were not found for projects 2 and 4. Moreover, projects 2 and 4 showed the highest scores among all the projects with the lowest standard deviation. The main difference between these two groups of projects is that, in projects 2 and 4 students were provided with more information about how to structure the underlying algorithm. This additional structure was necessary since the subject of the modules, ordinary and partial differential equations were not subjects with which students were familiar prior to the course. The additional support provided in crafting the algorithm appears to have been sufficient for students to properly implement a solution. On the other hand, because of the higher level of scaffolding, students may have been less engaged in higher order levels of thinking, limiting their ability to interpret their solution. Another possible explanation relates to the nature of the scaffolding that was provided. The scaffolding was focused on the algorithm structure, but the disciplinary and mathematical content may have become the challenge to interpret the results of the simulation.

Finally, the fact that student score for project 4 was weakly related to the course score also suggests that the additional scaffolding provided for project 4 may not necessarily have contributed to the overall learning outcomes. Nevertheless, project 4 needs to be further explored in order to understand its particularities.

How do students' problem solving approaches relate to their computational literacy skills?

The relationship between computational literacy and the stages of the problem solving process and the scores is not clear. For the individual project scores no correlation was found between computational literacy and student performance. Scores with a zero or very small standard deviation make it difficult to identify these relationships. However, it is important to highlight that most of the scores for this particular criterion were very high (~92% in average), and only for the last project was less than 89%. This suggest that the effect on disciplinary learning from building and interpreting the solutions is helping students approach the solution to the posed problems.

Overall, results from this study suggest that the use of computational challenges aligned with the problem solving phases can scaffold students' learning and computational literacy. For the projects where the disciplinary content is overwhelming for students, the scaffolding should be focused on the specific subjects instead of providing too much computational support. Thus, further research is necessary to identify what are the differences between different types of challenges and the level of scaffolding in student understanding and student performance in transfer tasks.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation under the awards #EEC1329262 and #EEC1449238.

References

1. P. Turner, L. Petzold, A. Shiflet, I. Vakalis, K. Jordan, and S. St. John, "Undergraduate computational science and engineering education," Society for Industrial and Applied Mathematics Review (SIAM Rev.), vol. 53, pp. 561-574, 2011.

2. T. A. Litzinger, P. V. Meter, C. M. Firetto, L. J. Passmore, C. B. Masters, S. R. Turns, . . . S. E. Zappe, "A cognitive study of problem solving in statics," Journal of Engineering Education, vol. 99, no. 4, 2010.

3. SCANS Commission. What work requires of schools: A SCANS Report for America 2000. Washington, DC: The Secretary's Commission on Achieving Necessary Skills, U. S. Department of Labor, 1991.

4. C. M. Vest, Educating Engineers for 2020 and Beyond, The Bridge, Washington, DC: National Academy of Engineering, 2006

5. J. Hafner, "Atomic-scale computational materials science," Acta Materialia, vol. 48, 2000.

6. NSF, National Science Foundation Advisory Committee for Cyber Infrastructure Task Force on Grand Challenges Final Report, 2011.

7. A. J. Magana, M. L. Falk, J. M. Reese, JR, "Introducing Discipline-Based Computing in Undergraduate Engineering Education," ACM Transactions on Computing Education, vol. 13, no. 4, 2013.

8. M. C. Carlisle, "Using YouTube to enhance student class preparation in an introductory Java course". In Proceedings of the 41st Annual Technical Symposium of Computer Science Education (SIGCSE'10), Milwaukee, WI, March 2010.

9. G. C. Gannod, J. E. Burge, and M. T. Helmick, "Using the inverted classroom to teach software engineering". In Proceedings of the 30th International Conference on Software Engineering (ISCE'08), Leipzig, Germany, May 2008.

10. N. Nagappan, L. Williams, M. Ferzli, E. Wiebe, K. Yang, C. Miller, and S. Balik, Improving the CS1 experience with pair programming. In Proceedings of the 34th Annual Technical Symposium of Computer Science Education, Reno, NV, February 2003.

11. J. G. Trafton, and R. J. Reiser, "The contribution of studying examples and solving problems to skill acquisition". In M. Polson (Ed.), Proceedings of the 15th

Annual Conference of the Cognitive Science Society (pp. 1017–1022). Hillsdale: Lawrence Erlbaum, 1993.

12. A. A. diSessa, Changing Minds: Computers, Learning, and Literacy. Cambridge: The MIT Press 2001.

13. W.J. Leonard, W.J. Gerace, R. J. Dufresne, and J.P. Mestre, "Concept-based problem solving: Combining educational research results and practical experience to create a framework for learning from physics and to derive effective classroom practices," In William J. Gerace, William J. Leonard, Robert J. Dufresne, Jose P. Mestre, (Eds.), Teacher's Guide to accompany Minds•On Physics: Motion Kendall/Hunt, Dubuque, IA, 1997.

14. J. Bransford, How People Learn: Brain, Mind, Experience, and School, National Academies Press, Washington, DC., 2000.

15. A. Rubin, Statistics for Evidence-Based Practice and Evaluation; Cengage Learning: Belmont, CA, 2009.

Biographical Information

Camilo Vieira is a Ph.D. candidate in Computer and Information Technology at Purdue University. He completed his undergraduate and his master's studies at Universidad Eafit, in Medellin Colombia. He holds a bachelor's degree in Systems Engineering and a master's degree in engineering. He is currently research assistant for the ROCkETEd group, where he works with computing education, and learning analytics for engineering education.

Alejandra J. Magana is an Associate Professor in the Department of Computer and Information Technology and an affiliated faculty at the School of Engineering Education at Purdue University. She holds a B.E. in Information Systems, a M.S. in Technology, both from Tec de Monterrey; and a M.S. in Educational Technology and a Ph.D. in Engineering Education from Purdue University. Her research is focused on identifying how model-based cognition in STEM can be better supported by means of expert technological and computing tools such as cyber-physical systems, visualizations and modeling and simulations.

Anindya Roy is a Postdoctoral Fellow in the Department of Materials Science and Engineering at Johns Hopkins University. He received his Ph.D. in 2011 from Rutgers University. As a computational physicist, Anindya's primary research focus is on understanding materials important for energy harvesting, storage and management, using quantum-chemistry-based calculations. Besides materials research, he is also interested in teaching at the undergraduate level, and understanding the pedagogical aspects of physics and engineering education.

Michael L. Falk is a Professor in the Department of Materials Science and Engineering at Johns Hopkins University's Whiting School of Engineering with secondary appointments in Mechanical Engineering and in Physics and Astronomy. He holds a B.A. in Physics and a M.S.E. in Computer Science from Johns Hopkins University and a Ph.D. in Physics from the University of California, Santa Barbara. His education research focuses on integrating computation into the undergraduate core curriculum. Falk also serves as the lead investigator for STEM Achievement in Baltimore Elementary Schools (SABES) an NSF funded Community Enterprise for STEM Learning partnership between JHU and Baltimore City Schools.

Michael J. Reese, Jr. is the Associate Director of the Johns Hopkins Center for Educational Resources. He holds a B.S. in electrical engineering from Virginia Tech, a M.Ed. in instructional technology from the University of Virginia, and a Ph.D. in sociology from Johns Hopkins University. His research investigates how educational innovations diffuse through higher education along with their impact on student learning.

Appendix A: Description of the projects

Before the problem statement, all the projects included a description of the disciplinary problem and the mathematical model to represent it. All the projects described in this section included an additional activity in which students were required to interpret the programmed solution.

Project 1: Calculating binary phase diagrams

Your assignment is to write 4 functions that graphically represent and calculate the phases present in a binary phase diagram. You will build your program in 4 parts. Some functions will need to use the prior functions, so make sure each works well before moving on to the next. All functions must be well documented with comments in order to receive full credit. You must use the function headers given below.

1. A function to calculate the free energy of a pure phase at a given composition (15 points)

   function G = FreeEnergy(x, HA, SA, HB, SB, w, T)

2. A function that computes the convex hull. (40 points)

   function [Gmin,Phases] = ConvexHull(x,G1,w1,G2,w2)

3. A function that shows the convex hull. (15 points)

   function ShowHull(HA1, SA1, HB1, SB1, w1, HA2, SA2, HB2, SB2, w2, T)

4. A function that plots a phase diagram (35 points)

   function PhaseDiagram(HA1, SA1, HB1, SB1, w1, HA2, SA2, HB2, SB2, w2, Tmin, Tmax)

Project 2: Modeling HIV Response to Immune Therapy

Your assignment is to write a computer program that will model the progress of the HIV infection in a patient that is being treated with a drug of a given effectiveness, Q. The HIV infected patient is assumed to start with a T-cell count of T(0)=1, this being a healthy level, and having no infected T-cells, I(0)=0. We assume that infection occurs at day 0 a viral load of V(0)=0.01. We will assume that if a drug is administered, therapy starts on the day of infection. In the model, if the HIV infected patient's T-cell count, including both infected an uninfected cells, falls below T+I=0.01 the patient is considered to have developed AIDS. Your assignment should follow the scheme laid out above and should be composed of the following primary function and sub-functions:

1. A function to initialize the variables and runs the simulation

   function Project2a(V0, Q, maxtime, minT)

2. A function to calculate the next day's T-cell count based on the current viral load and T-cell count

   function Tnext = NewT(T, V)

3. A function to calculate the next day's infected T-cell count based on the current viral load and T-cell count

   function Inext = NewI(T, I, V)

4. A function to calculate the next day's viral load count based on the current viral load, T-cell count, and infected T-cell count

   function Vnext= NewV(T, I, V, Q)

5. A function to increment the current viral load, T-cell count, and infected T-cell count by one day

   function [Tnext, Inext, Vnext]= Increment(T, I, V, Q)

Project 3: Crystal Structures and Cleavage Planes

Your assignment is to write a MATLAB function that will write a file that contains the positions of the atoms in a representative crystal that is either simple cubic, body centered cubic or face centered cubic. These crystals will have been cleaved along a plane specified by the user such that the files only contain the atoms on one side of the cleavage plane as well as the atoms on the cleavage surface. The atoms on the cleavage surface will be specially labeled. The file output by your code will have 4 columns. The columns will contain the following data about the atoms:

x-position y-position z-position label

1. A function to return the atom positions and a label for a specified crystal structure

   function cleave(xtal, unitcells, plane, filename)

2. A function to read the positions and label from the file and to plot spheres at the 3 dimensional coordinates specified by the positions

   function atomplot(filename)

Project 4: Modeling Heart Tissue and Diffusion of the Electrical Potential

Write a program that will simulate the diffusion of the electrical potential in the heart tissue. It should contain the following procedures:

1. A function to take an NxN array and stimulate a circular region with radiohs N/8 centered at row r and column c by setting the values of U in that region to 0.8

   function U = StimTissue(U, r, c)

2. A function to create an initial condition where U and V are NxN arrays that are zero everywhere except for a circle of radius/8 centered at row (N+1)/2 and col (N+1)/2 which should have U=0.8.

   function [U, V] = InitTissue(N)

3. A function to advance the clock on U and V by one time step

   function [newU, newV] = StepTissue(U, V, D, dt)

4. A function to simulate a tissue that is represented by two NxN arrays. The simulation lasts T time steps. Every *stime* steps a randomly located region of radius N/8 is electrically simulated

   function SimTissue(N, T, stime, ptime, D, dt)

5. A function to simulate a tissue that is represented by two NxN arrays. The simulation lasts T time steps. Once, after *stime* steps a region centered on row r and column c of radius N/9 is stimulated.

   function TestTissue(N, T, stime, r, c, ptime, D, dt)

Project 5: Molecular Dynamics Simulation

Molecular dynamics simulates atoms as point particles. Each atom has a position and a velocity, and these quantities are updated according to Newton's equations of motion. The model we will consider will be as simple as we can make it and still see interesting behavior. We will consider a two-dimensional system composed of 64 like atoms interacting via a very simple pair-wise interaction. The atoms will only interact with other atoms within some interaction range. The atoms will be kept in a defined region of space by imposing "periodic boundary conditions". You will provide the option of holding the average kinetic energy of the atoms fixed during the simulation. The code structure was at student discretion.

Appendix B: Pearson correlation of individual project scores by rubric criteria.

				Project 1			
	Problem Recognition	**Problem Framing**	**Problem Synthesis**	**Computational Literacy**	**Average Project Score**	**Course Score**	**Total Project 1**
Problem Recognition	1.000						
Problem Framing	0.484	1.000					
Problem Synthesis	0.215	**0.657**	1.000				
Computational Literacy	NA	NA	NA	NA			
Average Project Score	0.453	**0.888**	**0.864**	NA	1.000		
Course Score	0.171	**0.682**	**0.733**	NA	**0.753**	1.000	
Total Project 1	0.450	**0.839**	**0.886**	NA	**0.933**	**0.689**	1.000

Project 2							
	Problem Recognition	**Problem Framing**	**Problem Synthesis**	**Computational Literacy**	**Average Project Score**	**Course Score**	**Total Project 2**
Problem Recognition	NA						
Problem Framing	NA	1.000					
Problem Synthesis	NA	0.174	1.000				
Computational Literacy	NA	0.261	0.280	1.000			
Average Project Score	NA	**0.735**	0.459	0.362	1.000		
Course Score	NA	**0.644**	0.483	0.158	**0.741**	1.000	
Total Project 2	NA	**0.736**	0.444	0.353	**0.709**	0.586	1.000

Project 3							
	Problem Recognition	**Problem Framing**	**Problem Synthesis**	**Computational Literacy**	**Average Project Score**	**Course Score**	**Total Project 3**
Problem Recognition	1						
Problem Framing	0.220	1.000					
Problem Synthesis	-0.147	**0.607**	1.000				
Computational Literacy	NA	NA	NA	NA			
Average Project Score	0.270	**0.735**	0.644	NA	1.000		
Course Score	-0.013	**0.644**	0.605	NA	**0.741**	1.000	
Total Project 3	0.166	**0.703**	**0.885**	NA	**0.658**	0.507	1.000

Project 4							
	Problem Recognition	**Problem Framing**	**Problem Synthesis**	**Computational Literacy**	**Average Project Score**	**Course Score**	**Total Project 4**
Problem Recognition	1						
Problem Framing	0.306	1.000					
Problem Synthesis	0.175	0.101	1.000				
Computational Literacy	0.444	0.228	0.230	1.000			
Average Project Score	0.353	**0.735**	0.205	0.390	1.000		
Course Score	0.258	**0.644**	0.022	0.096	**0.741**	1.000	
Total Project 4	**0.808**	0.353	0.427	**0.812**	0.437	0.187	1.000

Project 5							
	Problem Recognition	**Problem Framing**	**Problem Synthesis**	**Computational Literacy**	**Average Project Score**	**Course Score**	**Total Project 5**
Problem Recognition	1.000						
Problem Framing	**0.669**	1.000					
Problem Synthesis	**0.612**	**0.622**	1.000				
Computational Literacy	0.476	0.153	0.529	1.000			
Average Project Score	**0.851**	**0.729**	**0.649**	**0.607**	1.000		
Course Score	0.598	**0.643**	**0.728**	0.477	**0.751**	1.000	
Total Project 5	0.839	**0.616**	**0.859**	**0.750**	**0.819**	**0.723**	1.000