A model Scientific Computing course for freshman students at liberal arts Colleges

Arun K. Sharma
Wagner College
Department of Chemistry & Physics
1 Campus Road
Staten Island, NY 10301
aksharma@wagner.edu

ABSTRACT

This paper describes a course called, "Introduction to Scientific Computing" that has been developed for freshman students at Wagner college, a private national liberal arts institution. The course trains students in computational thinking and involves hands-on learning of typical work-flows in scientific data analysis and data visualization. Students develop proficiency in the symbolic computing platform, Wolfram Mathematica®, to apply functional programming to develop data analysis and problem solving skills. The course presents computational thinking examples in the framework of various scientific disciplines. This exposure helps students to understand the advantages of technical computing and its direct relevance to their educational goals. The students are also trained to perform molecular visualization using open source software packages, such as Avogadro and Visual Molecular Dynamics, to understand secondary and tertiary protein structures, construct molecular animations, and to analyze computer simulation data. These experiences stimulate students to apply these skills across multiple courses and their research endeavors. Student self-assessment data suggests that the course satisfies a unique niche in undergraduate education. We have provided a sample syllabus, homework assignments, and examples of student work to aid in the design and implementation of similar courses at other institutions.

CCS Concepts

•**Applied computing** → **Education;** *Chemistry; Physics; Computer-assisted instruction; Computer-managed instruction;* Mathematics and statistics; Collaborative learning;

Keywords

Undergraduate; Freshman Year; Scientific Computing; Education; Visualization

1. INTRODUCTION

DOI: https://doi.org/10.22369/issn.2153-4136/8/2/1

Computational Science is a relatively new phenomenon in the long march of scientific disciplines and pursuits. From their invention in 1940s[1] to large scale supercomputers that are advancing fundamental research in all disciplines, computing has a fascinating history[2]. However, a typical liberal arts student gets very little to no exposure to such inspirational and powerful uses of computers and computational thinking. Most students consider the usage of computers for routine purposes (browsing, social media, document processing, etc.) as being "technologically literate". However, this is a very limited form of literacy and educators need to raise the level of technological competency to enhance the career prospects of our students[3, 4].

The STEM education community has focused its attention on introducing students to computational thinking and computation as distinct from routine computer usage [5, 6, 3]. The physics education community has been instrumental in adopting computer aided instruction and full fledged curricula and majors in Computational Physics exist at many institutions[7, 8, 9, 10, 11, 12]. Similar attempts to introduce computing in the chemistry curriculum have also been reported in the literature [13, 14, 15, 16, 17, 18]. These curricular level transformations are important milestones in modernizing and improving educational outcomes and skills of future graduates. However, an institution without the resources to engage in curricular overhaul may not be able to take advantage of such approaches. Particularly, smaller liberal arts colleges with limited faculty size and infrastructure are in an especially unfavorable position to embed computing holistically in an entire discipline or program.

A feasible alternative is to modernize the introductory computer course to highlight computational thinking and move beyond routine usage of computers. However, this approach lacks cohesiveness and students may not see the computing as directly relevant to their studies. We advocate the alternative approach to embed computing in a freshman level course in the natural sciences and provide students with an exposure to computational thinking in the framework of their course. A striking example of such an approach in an introductory mechanics course was recently reported [19]. That course required students to learn elementary programming skills to solve physics problems using VPYTHON programming environment. These students did not possess formal computer science coursework exposure and a majority of the class successfully completed the evaluation. These approaches are warranted to ensure that future STEM grad-

uates can converse with computers with reasonable fluency in the future. We highlight an approach to inject computational thinking and molecular visualization at the freshman level. The importance of visualization in disciplines like chemistry and biology cannot be overstated[20, 21] and the power of computational thinking to assist students in physics and other disciplines has been well documented[19, 7, 8, 16, 17]. Our course provides students with a toolbox of computational thinking and molecular visualization and editing tools that can be advantageously applied to a variety of courses offered in the STEM disciplines. The easy availability and widespread acceptance of these tools by students has the potential to increase the richness and depth of coursework in many disciplines.

2. METHODOLOGY

We designed and implemented a Scientific Computing course to teach elementary computational thinking in the framework of STEM disciplines and to highlight applications of computing to the natural sciences. The course aims to develop specific competencies of computational thinking, molecular visualization and editing, and data analysis. The course also aims to provide students with an introduction to functional programming, large scale data visualization and representation. A Scientific Computing course is commonplace in engineering disciplines. Such courses are typically focused on applications of numerical methods and their applications in solving scientific and engineering problems. There are many texts available on scientific computing with this focus [22, 23, 24, 25]. However, these books and corresponding courses are designed for an audience familiar with some computer programming language and/or advanced mathematics courses like calculus or linear algebra. Such courses are generally offered by Computer Science faculty and are typical coursework for engineering concentrations or Computational Physics majors.

However, there is no such parallel at liberal arts colleges. A standard scientific computing course as previously described would not be applicable to the student population at liberal arts colleges. Wagner College, like many liberal arts colleges, requires a semester of a Computer Science affiliated course to provide students with an exposure to "technological skills". Sadly, such courses typically do not delve into high-level computational skills or computational thinking and generally provide instruction in using Microsoft Office® products and rudimentary worldwide web concepts. These courses and approaches were probably valuable a decade ago when computing devices were not quite as prevalent. However, in today's world these courses appear outdated and do not provide relevant skills to the modern undergraduate student. We devised a course called, "Introduction to Scientific Computing", with the express intention to include all potential STEM students in the course. The course is designed to be inclusive toward students with all levels of preparation and previous exposure to computers and mathematics preparation. Most importantly, the course has no computer science pre-requisites. The course pre-requisites are any introductory class in the natural science disciplines taken during the first semester freshman year. Thus, the course is designed to be appropriate and most advantageous for second-semester freshman students.

The broad spectrum of the course allows it to serve a variety of students pursuing multiple STEM career paths. The course intention is to highlight the ubiquitous thread of computing that permeates modern scientific endeavors. The course goal is not to design and produce efficient programs or code. The course goal is to reach out to students who may have never considered computers as an ally to perform the science and learning that interests them. An important course goal is to highlight the role and power of computational thinking in solving problems. The course uses the freely available text, An Elementary introduction to Wolfram language[26], by Stephen Wolfram, the creator of the Wolfram programming language which is at the core of Mathematica®.

We chose to use the Mathematica platform to reduce the entry barrier and also to take advantage of the immense variety of applications and in-built functions in Mathematica that span the STEM domains and beyond. Python programming language is similar in many respects and we think it could be suitable to construct a similar course. However, we wanted to steer clear of the notion that this is a computer programming course and eschewed the Python programming language. We wanted to ensure that we could reach the maximum number of students and even those students who had previous negative experiences with computer programming. A similar course could be constructed using Maple®, Matlab®, or other software packages based on instructor familiarity and availability. The most important consideration is that there should be an element of higher level programming and scripting that allows students to engage with the code and to construct analysis tools using functional programming ideas. Thus, we would argue that spreadsheet based packages like Microsoft Excel®, etc. are not appropriate as the primary course platform.

The course can be divided into two broad sections: Mathematica technical platform and molecular visualization and editing. The Mathematica portion familiarizes students with the basic data structures and operations. We also highlight the applications of curated databases and their integrations into solving scientific problems. Students are introduced to various forms of data presentation including static and dynamic plots and charts. Finally, we introduce students to molecular editing and visualization using open source tools like Avogadro[27] and Visual Molecular Dynamics(VMD)[28]. We combine the elements of data analysis and visualization by presenting students with molecular dynamics simulation trajectory and carrying out its analysis as a classroom exercise. The next few sections will describe each of these modules in more detail.

3. MATHEMATICA MODULE

The course starts by introducing students to the various domains that can be accessed through Mathematica. This is achieved by highlighting in-built functions in Mathematica. We start by plotting trigonometric functions and three-dimensional plots. Mathematica recently added curated databases to the program. These allow a look-up of data from domains like chemistry, physics, biology, financial markets, economic data for countries, engineering data, mortality rates, weather data, etc. For many students this is their first ever encounter with such data. The goal of the first lecture is to ensure that students feel empowered and experience that they can write the code that can access such high level tasks. We impress upon students that the course is not about nuts and bolts of programming, but learning

technical platforms that can be used to address high level questions.

The course follows a workshop model in which the instructor projects the notebook (Mathematica interface) on a large screen and students write down the code on their individual computers. This is important to get the first practice of writing down the code and to increase student familiarity and confidence with the interface. The basic data structure in Mathematica is a list. We spend at least three lectures on familiarizing students with the list (array) structure and performing operations. Standard operations like reversing the list, calculating mean, standard deviations, etc. are carried out to develop baseline competency. We perform simple algebra on lists and solve problems that require setting up lists for solutions. Such problems abound in scientific domains and provide a review of scientific concepts as well as the skill of interpreting scientific problems into a computational framework. We necessarily choose problems from different domains, physics, chemistry, etc. to highlight broad usage of the platform and to inspire students to use their newly acquired skill across other courses.

Students also learn to visually represent different types of data. The standard mathematical functions, dynamic plots, parametric plots, etc. are also explored in the course. Bar charts and histograms are also covered to provide students with an overview of different aspects of data presentation. The students appreciate different data representations and their appropriateness to highlight certain aspects of the data. Lively discussions on choosing data representation are commonplace in the classroom. In all cases the data originates from solution of a problem from one of the scientific disciplines. This builds up on previous knowledge of the core principles of Mathematica. A glimpse of the diversity of examples that we execute in class and homework assignments for this module is provided below. These examples take advantage of Mathematica's curated research databases and the coding principles taught in the course. This approach enables students to use computational skills in their other courses and research assignments.

1. Plotting density and temperature of water to determine the temperature of maximum density of liquid water
2. Construction and analysis of dice games
3. Plotting density of elements and analyzing trends
4. Identifying the 10 most populated cities in the USA and data representations
5. Determination of the number of airports in the G8 countries
6. Plotting temperature data for various cities and analyzing weather trends
7. Atomic radii of elements in the second and third period of the Periodic Table
8. Interactive histograms of distributions of randomly selected integers
9. Interactive plots of mathematical functions

We will highlight the example of a die throwing game to demonstrate the readable and expansible nature of the code. The algorithm is simply described as:

1. Create a list of 6 integers for a regular six-sided die
2. Pick an integer randomly to represent a particular face of the die
3. Accumulate the counts of each face over a number of die throws
4. Plot the distribution of each face for the specified number of trials

A sequential construction of this example is provided in the following code snippet

```
rolls = Table[RandomInteger[{1,6}],{i
    ,1,100000}];
Histogram[rolls]
```

However, these can be merged into a one-line code eliminating the need to create and store a variable. This approach where functions are input to other functions is a powerful feature of functional programming and we stress this repeatedly during the course. This feature enables students to chain complicated tasks into simple and easy to read code fragments.

```
Histogram[Table[RandomInteger[{1,6}],{i
    ,1,100000}]]
```

This can be easily extended to two dice games and plotting the histograms of sums of two dice throws. The following command simulates million throws of two dice and adds the output. It then plots a histogram to highlight the distribution of the recorded sum.

```
Histogram[Table[RandomInteger[{1, 6}] +
    RandomInteger[{1, 6}], {i, 1,
    1000000}]]
```

As students develop their skills in the language and computational thinking, we revisit this example and create a custom function to carry out these operations. The dice example is used to highlight the power of repeated trials and the need for large numbers of trials to understand stochastic phenomena. Students construct the scenario of a comparison between an unloaded regular dice and a dice loaded to favor the face bearing the numbers four or six three times over other faces. This example impresses upon students the need for repeated trials and also the role of random numbers in stochastic processes. The output of this comparison of a loaded die with a regular die is shown in Fig. 1. The result clearly shows that a loaded die behavior can be detected with few trials, however, the regular die output needs many trials to verify the equal probability of each outcome. The code snippet highlights the advantage of using Mathematica and the ease with which students progress in their knowledge of coding.

```
regularDice[throws_] :=
Module[{out = Table[RandomChoice[{1, 2,
    3, 4, 5, 6}], {throws}]},
 Histogram[out, PlotLabel -> throws "
     Unloaded Die Throws"]]

loadedDice[throws_] :=
Module[{out =
   Table[RandomChoice[{0.1, 0.1, 0.1,
       0.3, 0.1, 0.3} -> {1, 2, 3, 4,
```

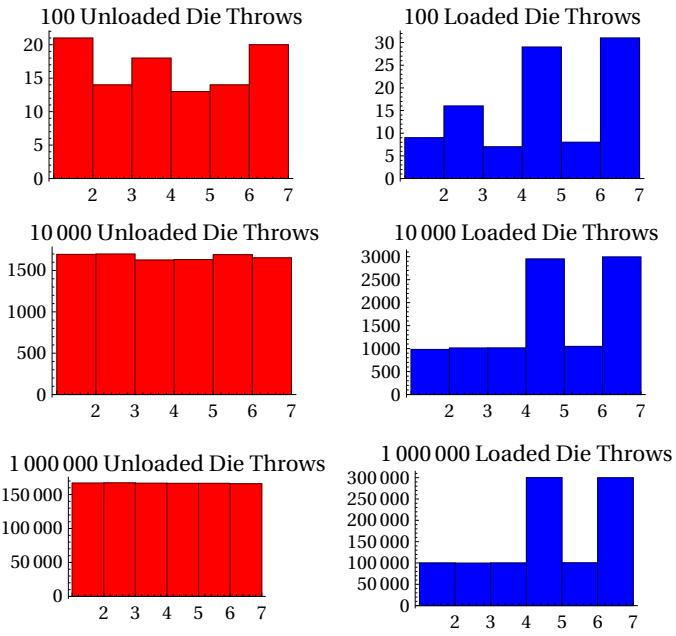

Figure 1: Histograms of outputs of an unloaded and loaded die. The behavior of a loaded die is clear even at low number of trials.

```
        5, 6}], {throws}]},
  Histogram[out, PlotLabel -> throws "
     Loaded Die Throws"]]

GraphicsGrid[{{regularDice[100],
   loadedDice[100]}, {regularDice[10000],
   loadedDice[10000]}, {regularDice
      [1000000], loadedDice[1000000]}}]
```

The focus of this module is to empower students with the skills of computational thinking and its application in various domains. The students are reminded constantly that Mathematica would be a great help for them in other courses, laboratory reports, and homework assignments. Students are also trained to write short programs (called Modules) that chain multiple built-in functions. This experience introduces students to the power of programming and its applicability to their daily activities. However, the course is not devoted to programming, rather its intention is to highlight the power and ease of functional programming and the advantages it can provide to students. The examples have been deliberately chosen to improve student understanding of pre-calculus and calculus, data analysis, data fitting, data visualization, and elementary physical science concepts. In the next section we describe the molecular visualization module.

4. MOLECULAR VISUALIZATION

Molecular editing and visualization is also embedded in the course to allow students to experience the applications of computing beyond numerical calculations. Students use an open-source application Avogadro[27] to create and edit molecular structures. Students also use Avogadro's features to create carbon nanotubes, aromatic compounds, and polypeptide sequences. This activity of building different molecules

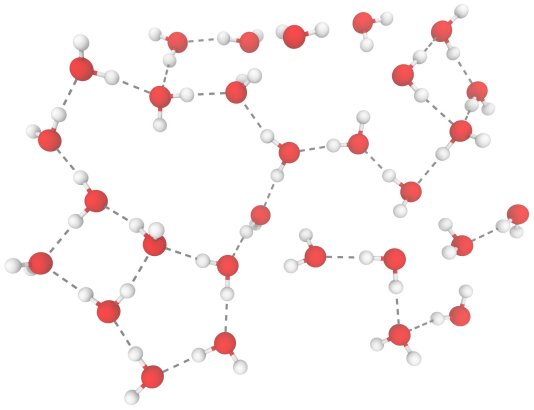

Figure 2: A snapshot of water arrangement in Avogadro. The dashed lines represent hydrogen bonding interactions.

is also helpful for students to develop skills of interaction with Graphical User Interface (GUI) of typical scientific applications. We apply Avogadro's molecular mechanics[29] capabilities to highlight intermolecular interactions and hydrogen bonding between water molecules. We also highlight Avogadro's application to analyzer stereochemistry and 3-D molecular structures. Students have informally reported after progressing to organic chemistry course that Avogadro was a very helpful tool for that course.

A signature activity that we perform with Avogadro is the study of arrangement and orientation of water molecules in a small cluster. The students follow along in this activity on their personal computers. We utilize the Auto Optimization feature of Avogadro for this exercise. This feature allows a continuous optimization of molecular geometry and arrangement using molecular mechanics. We start with 20-25 randomly placed water molecules on the screen. The classical mechanics force field MMFF94[30]is chosen to represent intermolecular interactions. This force field recognizes hydrogen bonding interactions and the resulting arrangement of water molecules in the cluster is a result of the hydrogen bonding propensity of these molecules. Students are asked to pick and drag molecules around the screen. The system responds instantaneously and rearranges other molecules in the vicinity. The program also displays the potential energy at each instant and students observe that potential energy fluctuations correspond to favorable or unfavorable local and global arrangements of water molecules. Specifically, the system of water molecules attempts to optimize the hydrogen bonding interactions. A snapshot of animation with hydrogen bonds is shown in Fig.2. An animation of this process is supplied in the Supporting Information.

The course utilizes Visual Molecular Dynamics (VMD)[28] to study three-dimensional structures of proteins and to introduce students to a tool widely used in computational chemistry and biology research laboratories. VMD is a powerful tool to visualize structural properties, perform molecular dynamics simulation setup and data analysis for the NAMD[31] simulation engine. VMD allows fine-grained control of structural representation and can also be used to visualize a simulation trajectory run in any of the popular molecular dynamics simulation engines like GROMACS[32],

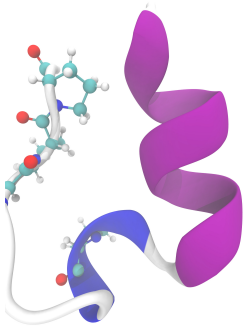

Figure 3: 1l2y mini-protein with proline residues highlighted in CPK representation. The secondary structure is emphasized by using the NewCartoon representation in VMD.

NAMD[31], LAMMPS[33], etc. This enables us to introduce the Protein Data Bank (PDB)[34] and to provide an overview of protein structure, stability, and function.

Students learn to browse the PDB resource, load molecules into VMD and extract information from the structures. They are trained to represent different molecular representations like CPK[35], Van der Waals, and secondary structure representations that are commonly seen in the literature and biology textbooks. For example, we study the structure of a small synthetic mini-protein trp-cage (PDB code: 1L2Y)[36]. The structure is shown in Fig.3 with the proline residues in CPK representation and the rest of the molecule in NewCartoon representation to highlight the α-helix motif. This also allows students to observe the general rule that proline residues do not participate in the formation of α-helix secondary structure motifs[37].

Similarly, we highlight structure of a β-barrel membrane protein (PDB code: 1G90)[38] located in the outer membrane of Gram-negative bacteria. This protein is chosen to highlight the β-sheet secondary structure element and the β-barrel morphology. Students learn to visualize the hydrogen bond connections between the sheets and to represent the structure in multiple ways. A representation of this protein is depicted in Fig.4. The red dashed lines represent hydrogen bonding interactions.

An important element of the course is to develop critical thinking skills of students and to encourage them to question their own beliefs about concepts learned in other courses. Hydrogen bonding is one of the most important concepts that students of Chemistry and Biology (the major population of our course at Wagner College) need to understand. A hydrogen bond is simply an electrostatic interaction between a hydrogen atom connected to a highly electronegative atom like fluorine, oxygen or nitrogen and another electronegative atom. Hydrogen bonds are commonly understood to be a key factor in protein structural stability, stability of DNA, RNA, and the unique properties of water. However, most students do not possess an intuitive understanding of how an interaction would be labeled as a hydrogen bond in the pictures seen in their biology or chemistry textbooks. Students use VMD options to display hydrogen bonds in protein structures. They also modify the default distance and angle cutoff to study the impact of such restrictions on the number of hydrogen bonds reported in the structure. The

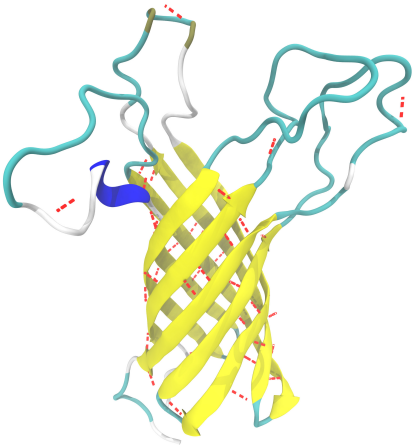

Figure 4: A β-barrel membrane protein. The red dashed lines represent hydrogen bonding interactions.

default distance criterion is 3.3 Angstrom and default angle criterion is that the Donor-Hydrogen-Acceptor angle is less than 20 degrees. Students discover that modifying these definitions leads to a change in the number of interactions flagged as hydrogen bonds and also study hydrogen bonds in different regions of proteins. This model of instruction enables students to appreciate the role of visualization and also the type of decisions that are implicit in visualization engines that study biomolecules. This exercise helps students to understand that pictures seen in books, textbooks, and research articles are the result of many decisions that must be understood from a computational and physical perspective.

The interplay of visualization with data analysis is highlighted in the next signature activity that we perform with the students. We apply VMD to visualize the trajectory of a short molecular dynamics simulation of liquid water and Mathematica to perform data analysis on observables recorded during the simulation. Such an activity would not be possible to perform with freshman level students outside of this course.

5. ANALYSIS OF MOLECULAR DYNAMICS SIMULATION

We perform the visualization and analysis of a short molecular dynamics simulation of liquid water at room temperature. We execute the simulation beforehand and project the simulation trajectory for visualization to the class. We then display hydrogen bonds during the simulation frames. Usually, a lively discussion follows about the behavior of liquid water during the simulation and how they could use VMD[28] to display the hydrogen bonds and use different representations of the simulation trajectory. At this point in the curriculum students realize that a molecule in bulk water attempts optimization of its hydrogen bonding network. The trajectory is processed using GROMACS[39] tools beforehand and students are presented with text files containing measurement of temperature, potential energy, kinetic energy, etc. during the simulation.

Students are provided a brief overview of molecular dy-

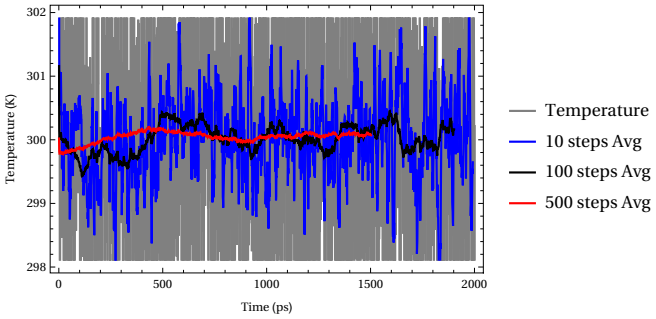

Figure 5: Moving averages to compute the average temperature during a molecular dynamics simulation. The fluctuations in the measurement are highly suppressed as the width of averaging window increases.

namics simulation and a brief explanation of the NPT ensemble[40] used to carry out the simulations. They calculate the number of water molecules and volume of simulation box to achieve appropriate density. This helps to build an appreciation of the scale of molecular simulations. A particularly illuminating aspect of this exercise is the concept of average value of a measurement over a time period. The simulation provides an output of temperature measurement and other physical observables at every picosecond during the simulation. Thus, for a 2 nanosecond simulation there are 2000 recorded temperature values. Students import the raw data into Mathematica and visualize the measurement. They notice that the temperature fluctuates and is not truly fixed at 300 K for the entire simulation. The students then perform moving averages to understand the temperature measurement. As they perform moving averages over measurement values, they realize that instantaneous measurements may differ significantly from the average. They perform a time average using a duration of 10 ps, 100 ps, and 500 ps each. They observe that with larger averaging duration the temperature measurement is very close to the preset 300 K for the simulation. Fig.5 depicts a plot that students construct using the raw data. This exercise also provides an opportunity to impart the skill of importing data from text files into Mathematica and data processing for desired analysis. Students report that this is usually their first experience analyzing large amounts of data and visualizing fluctuations in the measurement of a commonly experienced physical observable such as temperature. The exercise also involves analyzing potential and kinetic energies and the number of hydrogen bonds during the simulation.

6. SMALL PROJECT EXPERIENCE

Students perform a small research project during the last six weeks of the course. These are group projects that incorporate themes from the course. We reserve a portion of class time for students to pursue these projects and to get assistance from the instructor and other classmates. A few student projects are described below to provide an overview:

1. Mathemusica: Students explained the mechanism of sound, action of sound waves, and programmed popular music and classical music compositions using Wolfram programming language in Mathematica. This was a highly unusual and creative application of programming and computing.

2. Global warming trends: Students downloaded datasets of global surface and sea temperatures from NOAA to analyze trends and to highlight the measures of global warming.

3. Gallery of 3-D structures and functions: Students created 3-D structures of proteins, their genetic information, and their properties using VMD and Mathematica's biological data functionalities.

4. Gallery of atomic orbitals: Students plotted mathematical expressions for atomic orbitals to depict shapes of atomic orbitals and their properties.

These projects highlight the diversity of students and their interests in the course. They also demonstrate the diverse range of functions and databases that are built into Mathematica and of student efforts to apply their skills in a variety of domains.

7. STUDENT ASSESSMENT

This course has been taught twice at Wagner College to a total audience of 30 students with a distribution of students from all levels. Student response to the course has been overwhelmingly positive. The freshman population in Spring 2015 was 50% and in Spring 2016 was 43%. The strong interest from students at all levels is highly encouraging. Although, the course would be most beneficial if students take it as early as second semester of freshman year. Additionally, it also points to a pressing need for this course. Students in their junior and senior years of College have expressed amazement and disbelief about the accomplishments of computational science and its potential impact on their career paths. The course has a strong enrollment for the Spring 2017 semester with 19 students (11 freshmen) registered for the course.

The assessment was carried out using anonymous on-line survey. The average of student responses is reported along with standard deviation in parentheses. The responses have been merged for the first two iterations of the course with 30 respondents. The first set of questions had responses on scale from Strongly Agree (5) to Strongly Disagree (1). These questions have a broad focus and students seemed intent on applying the skills from this course to other courses.

1. I have acquired a better understanding of applications of computing to Science. **4.50** (0.51)

2. I plan to apply software and skills acquired from this course to other courses. **4.17** (0.81)

3. I have acquired skills that will help me in my major. **4.23** (0.77)

The second set of questions had responses on a scale used for Wagner College Chemistry graduate exit survey: A great deal (5), A lot (4), Some (3), A little (2), Not at all (1). These questions address distinct skills and tools practiced in the course. The responses are overwhelmingly positive and the course seems to be addressing a unique niche in the undergraduate liberal arts curriculum.

1. My skill in manipulation of large datasets has increased. **3.97** (0.81)

2. My skill in graphical analysis of data has increased. **4.07** (0.87)

3. My skill in setting up and solving numerical problems has increased. **3.9** (0.80)

4. My skill in visualization and analysis of mathematical functions has increased. **4.07** (0.91)

5. My skill in visualization of structure of biomolecules has increased. **4.13** (0.86)

6. My skill in molecular drawing and editing has increased. **4.23** (0.82)

The positive course evaluations and student experiences have inspired us to share this work with a larger audience.

8. CONCLUSION

We have created a course called, "Introduction to Scientific Computing" to introduce students to technical computing and functional programming at liberal arts Colleges. The course focus is to present computing and interaction with data in the framework of STEM disciplines. This approach fosters the integration of computing into the educational goals of students from all STEM branches and provides a set of tools that can be used throughout their undergraduate education and well into the professional work-place or graduate school. We think that the course utility is the highest for freshman students because of the high impact of these skills and tools on their undergraduate education. However, student response from all levels of student population has been highly encouraging. Students learn the skills of data analysis, data visualization, functional programming, molecular visualization, and molecular editing. Students also apply these skills collaboratively in small group based research projects. The diversity of projects and assignments carried out in the course highlight the broad applicability of the course to STEM education. Our hope is to transform the student mindset into accepting scientific computing as a skill that is as integral to the practice of STEM disciplines as their laboratory skills. We hope that colleagues at other institutions will consider creation of a similar course to better prepare our future generations.

9. SUPPORTING INFORMATION

The following files are provided as supporting information:

1. A notebook with code samples described in section 3 in Mathematica notebook format and PDF

2. Animation of the molecular rearrangement activity described in Section 4

3. A popular music song, "All of me" recreated by students in Mathematica (in CDF, PDF and Mathematica format). The document can be opened with the freely available Wolfram CDF player https://www.wolfram.com/cdf-player/.

4. A sample course syllabus and assignments are provided to aid in creation of a similar course.

10. ACKNOWLEDGMENTS

The author wishes to express his gratitude to Dr. Shawn Sendlinger for his encouragement during the Computational Chemistry for Chemistry educators workshop [41] to disseminate this work.

11. REFERENCES

[1] N Metropolis. The beginning of the Monte Carlo method. *Los Alamos Science*, 15:125–130, 1987.

[2] H. L. Anderson. Scientific uses of the MANIAC. *Journal of Statistical Physics*, 43(5-6):731–748, 1986.

[3] Theresa Julia Zielinski Mary L. Swift. What Chemists (Or Chemistry Students) Need to Know about Computing. *Journal of Science Education and Technology*, 4(2):171–179, 1995.

[4] J S Friedman and A A DiSessa. What students should know about technology: The case of scientific visualization. *Journal of Science Education and Technology*, 8(3):175–195, 1999.

[5] Terry Hinton. Computer assisted learning in physics. *Computers & Education*, 2(1-2):71–88, 1978.

[6] Richard Hooper. Computers in science teaching-An introduction. *Computers and Education*, 2(1-2):1–7, 1978.

[7] Vinesh Chandra and James J. Watters. Re-thinking physics teaching with web-based learning. *Computers and Education*, 58(1):631–640, 2012.

[8] Norman Chonacky and David Winch. Integrating computation into the undergraduate curriculum: A vision and guidelines for future developments. *American Journal of Physics*, 76(4):327–333, 2008.

[9] David M. Cook. Introducing Computational Tools in the Upper-Division Undergraduate Physics Curriculum. *Computers in Physics*, 4(2):197, 1990.

[10] K.R. Roos. An Incremental Approach to Computational Physics Education. *Computing in Science & Engineering*, 8(5):44–50, 2006.

[11] Ruxandra M. Serbanescu, Paul J. Kushner, and Sabine Stanley. Putting computation on a par with experiments and theory in the undergraduate physics curriculum. *American Journal of Physics*, 79(9):919, 2011.

[12] Jaime R. Taylor and B. Alex King. Using computational methods to reinvigorate an undergraduate physics curriculum. *Computing in Science and Engineering*, 8(5):38–43, 2006.

[13] Sharona T. Levy and Uri Wilensky. Crossing levels and representations: The connected chemistry (CC1) curriculum. *Journal of Science Education and Technology*, 18(3):224–242, 2009.

[14] Sharona T. Levy and Uri Wilensky. Students' learning with the connected chemistry (CC1) curriculum: Navigating the complexities of the particulate world. *Journal of Science Education and Technology*, 18(3):243–254, 2009.

[15] Ned H. Martin. Integration of Computational Chemistry into the Chemistry Curriculum. *Journal of Chemical Education*, 75(2):241, 1998.

[16] Richard A. Paselk and Robert W. Zoellner. Molecular Modeling and Computational Chemistry at Humboldt State University. *Journal of Chemical Education*,

79(10):1192, 2002.

[17] Mike Stieff and Uri Wilensky. Connected Chemistry: Incorporating Interactive Simulations into the Chemistry Classroom. *Journal of Science Education and Technology*, 12(3):285, 2003.

[18] David L. Cedeño, Marjorie A. Jones, Jon A. Friesen, Mark W. Wirtz, Luz Amalia Ríos, and Gonzalo Taborda Ocampo. Integrating Free Computer Software in Chemistry and Biochemistry Instruction: An International Collaboration. *Journal of Science Education and Technology*, 19(5):434–437, 2010.

[19] Marcos D. Caballero. Computation across the curriculum: What skills are needed? In *2015 Physics Education Research Conference Proceedings*, pages 79–82. American Association of Physics Teachers, 2015.

[20] National Academies. *Integrating Discovery-Based Research into the Undergraduate Curriculum*. National Academies Press, Washington, D.C., 2015.

[21] Loretta L. Jones, Kenneth D. Jordan, and Neil a. Stillings. Molecular visualization in chemistry education: the role of multidisciplinary collaboration. *Chemistry Education Research and Practice*, 6(3):136, 2005.

[22] Ionut. Danaila. *An introduction to scientific computing: twelve computational projects solved with MATLAB*. Springer, 2007.

[23] D Gruntz. *Symbolic Computation of Explicit Runge-Kutta Formulas*, pages 281–297. Springer, Berlin, Heidelberg, 2004.

[24] Alfio Quarteroni, Fausto Saleri, and Paola Gervasio. *Scientific Computing with MATLAB and Octave*, volume 2 of *Texts in Computational Science and Engineering*. Springer, Berlin, Heidelberg, 2014.

[25] Joseph L. Zachary. *Introduction to Scientific Programming: Computational Problem Solving Using Maple and C*. Springer New York, 1996.

[26] Stephen Wolfram. *An Elementary Introduction to the Wolfram Language*. Wolfram Media, 2015.

[27] Curtis D E Hanwell M.D. Lonie D.C., Vandermeersch, T., Zurek E., Hutchison, G. and Curtis D E Hanwell M.D. Lonie D.C., Vandermeersch, T., Zurek E., Hutchison, G. Avogadro: An Advanced Semantic Chemical Editor, Visualisation, and Analysis Platform. *Journal of Chemical Informatics*, 4.17:1–17, 2012.

[28] William Humphrey, Andrew Dalke, and Klaus. Schulten. VMD: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33–38, 1996.

[29] David C. Young. *Computational chemistry : A practical guide for applying techniques to real world problems*. Wiley, 2001.

[30] Thomas A. Halgren. Merck Molecular Force Field. *J. Comput. Chem.*, 17(5-6):490–519, 1996.

[31] James C. Phillips, Rosemary Braun, Wei Wang, James Gumbart, Emad Tajkhorshid, Elizabeth Villa, Christophe Chipot, Robert D. Skeel, Laxmikant Kalé, and Klaus Schulten. Scalable molecular dynamics with NAMD. *Journal of Computational Chemistry*, 26(16):1781–1802, 2005.

[32] H. J C Berendsen, D. van der Spoel, and R. van Drunen. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications*, 91(1-3):43–56, 1995.

[33] Steve Plimpton. Fast Parallel Algorithms for Short-Range Molecular Dynamics. *Journal of Computational Physics*, 117(1):1–19, 1995.

[34] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The Protein Data Bank. *Nucleic acids research*, 28(1):235–242, 2000.

[35] Robert B. Corey and Linus Pauling. Molecular models of amino acids, peptides, and proteins. *Review of Scientific Instruments*, 24(8):621–627, 1953.

[36] Jonathan W. Neidigh, R. Matthew Fesinmeyer, and Niels H. Andersen. Designing a 20-residue protein. *Nature Structural Biology*, 9(6):425–430, 2002.

[37] C N Pace and J M Scholtz. A helix propensity scale based on experimental studies of peptides and proteins. *Biophysical journal*, 75(1):422–427, 1998.

[38] Lukas K. Tamm, Heedeok Hong, and Binyong Liang. Folding and assembly of beta-barrel membrane proteins. *Biochimica et Biophysica Acta - Biomembranes*, 1666(1-2):250–263, 2004.

[39] David Van Der Spoel, Erik Lindahl, Berk Hess, Gerrit Groenhof, Alan E. Mark, and Herman J C Berendsen. GROMACS: Fast, flexible, and free. *Journal of Computational Chemistry*, 26(16):1701–1718, 2005.

[40] Daan Frenkel and Berend Smit. *Understanding molecular simulation: from algorithms to applications*. Academic Press, San Diego, second edition, 2002.

[41] Shawn C. Sendlinger and Clyde R. Metz. Computational Chemistry for Chemistry Educators. *The Journal of Computational Science Education*, 1(1):28–32, 2010.