December 2011

Volume 2 Issue 1

JoCsE

Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

Volume 2, Issue 1 **December 2011**

JOCSE

Journal Of Computational Science Education

The Journal Of Computational Science Education (JOCSE), ISSN 2153-4136, is published quarterly in online form, with more frequent releases if submission quantity warrants, and is a supported publication of the Shodor Education Foundation Incorporated. Materials accepted by JOCSE will be hosted on the JOCSE website, and will be catalogued by the Computational Science Education Reference Desk (CSERD) for inclusion in the National Science Digital Library (NSDL).

Subscription: JOCSE is a freely available online peer-reviewed publication which can be accessed at `http://jocse.org`.

Contents

Using Spreadsheets to Visualize Virus Concentration

Jyoti Champanerkar*
Department of Mathematics
William Paterson University
Wayne, NJ 07470.
champanerkarj@wpunj.edu

Christina Dizzia
Department of Mathematics
William Paterson University
Wayne, NJ 07470.

ABSTRACT
In this paper, we model the growth of a virus in an infected person, taking into account the effect of antivirals and immunity of the person. We use discrete dynamical systems or difference equations to model the situation. Excel is then used to obtain and visualize numerical solutions.

Keywords
Difference Equation, Recurrence Relation, Virus Concentration, Discrete Dynamical Systems

1. INTRODUCTION
Hantavirus can cause life-threatening infection which is spread to humans by rodents. Early symptoms of hantavirus disease are similar to flu, including chills, fever and muscle aches [3]. A virus found at a construction site in Rwanda is similar in behavior to the hantavirus. The virus grows rapidly from just a single cell, doubling every hour, and remains undetected by the human body until it reaches one million copies. Once the immune system detects the virus, it raises the body temperature, lowers the virus replication rate to 150% per hour and can kill a maximum of $200,000$ copies of the virus per hour. An hourly dose of antivirals and the immune system together can kill $500,000,000$ copies of the virus per hour while keeping the replication rate at 150% per hour. If the number of virus cells reach one billion before the antivirals are prescribed, the virus cannot be stopped and the infected person will die when the number of copies reach one trillion[1].

We will model the phases of the disease using discrete dynamical systems, analyze them using phase line analysis and obtain numerical solutions using Excel.

*Faculty Advisor
[1]Problem taken from `Mathmodels.org` [2]

2. DISCRETE DYNAMICAL SYSTEM

2.1 A Brief Background
A discrete dynamical system consists of a difference equation or a recurrence relation, along with an initial condition. A difference equation describes any $n+1^{th}$ term of a sequence using some of the previous n terms. For example, the difference equation $x_{n+1} = 2x_n - 1$ along with the initial condition $x_0 = 2$ describes the sequence $2, 3, 5, 9, 17, 33. \cdots$. The zeroth term of the sequence $x_0 = 2$ is given and each subsequent term is obtained by doubling the previous term and subtracting one from it. Thus we have $x_1 = 2(2) - 1 = 3$ and so on.

Observe that for the difference equation above, $x_{n+1} = 2x_n - 1$, if we started with an initial condition $x_0 = 1$, we would generate the constant sequence $1, 1, 1, 1, \cdots$. Such a value, if it exists, is called an equilibrium or a fixed point. An equilibrium p of a difference equation is a value such that if the initial condition is p, all subsequent terms in the generated sequence are also p. If the initial condition is chosen close to an equilibrium value and subsequent terms generated eventually get closer and closer to the equilibrium, then we say that the equilibrium is stable otherwise it is unstable. In the example above, $x_{n+1} = 2x_n - 1$, if we choose $x_0 = 1.1$, the subsequent terms are $1.2, 1.4, 1.8 \cdots$ which are farther from $p = 1$ than even the initial condition. So $p = 1$ is an unstable equilibrium.

In general, for a difference equation $x_{n+1} = ax_n + b$, the equilibrium is given by $p = \frac{b}{1-a}$, which exists if $a \neq 1$; and any initial point is an equilibrium if $a = 1$ and $b = 0$. The equilibrium $p = \frac{b}{1-a}$, if it exists, is stable when $|a| < 1$ and unstable when $|a| > 1$.

2.2 Modeling virus concentration
Let x_n denote the number of copies of the virus in the body at the end of n hours.

2.2.1 Initial Phase/ Phase 1
We assume that a single virus cell has infected a soldier. Hence, the initial condition is $x_0 = 1$. The virus doubles every hour. Hence at the end of $n + 1$ hours, there are double the number of virus copies of what were at the end of n hours. This gives the system:

$$x_{n+1} = 2x_n \quad (1)$$
$$x_0 = 1 \quad (2)$$

This system has the equilibrium point $p = 0$ and it is unstable (since $|2| > 1$). Which means that in the initial phase, even if a single copy of virus enters the body, the number of copies of the virus will increase (away from 0). See Figure 1.

We numerically solve this system using Excel and find that it would take, approximately, **20 hours** before the number of copies of the virus reaches one million, at which time the immune system begins to respond. See Figure 2.

2.2.2 *Response of the Immune System / Phase 2*

When the immune system begins to respond, the temperature of the body rises, the replication rate of the virus reduces to 150% per hour and the immune system kills up to $200,000$ copies of the virus per hour. Thus we have,

$$x_{n+1} = 1.5x_n - 200,000 \quad (3)$$
$$x_0 = 1,048,576 \quad (4)$$

This system has the equilibrium point $p = 400,000$ and it is unstable (since $|1.5| > 1$). See Figure 1. This means that if the immune system responded before the number of copies reached $400,000$ ($x_0 < p$), the virus would be cleaned out of the body, but if the immune system responds after the number of copies reached $400,000$ ($x_0 > p$), the number of copies of virus would continue to increase. However, since the immune system responds after the virus has reached one million copies, this virus cannot be cleaned out of the body by the immune system alone. We use the initial condition $x_0 = 1,048,576$ instead of one million, since by the end of 20 hours the virus had reached $1,048,576$ copies and between 19 and 20 hours, the virus had not crossed one million copies and to avoid fractional hours.

We numerically solve the system using Excel and find that it would take, approximately, 19 hours for the number of copies of virus to cross one billion. Thus, it would take 39 hours since the first copy of virus enters the body, for the virus to reach one billion copies. See Figure 3.

2.2.3 *Administration of Antivirals / Phase 3*

When hourly antivirals are administered, the immune system along with the antivirals kills $500,000,000$ copies of the virus and replication stays at 150%. This gives the difference equation:

$$x_{n+1} = 1.5x_n - 500,000,000 \quad (5)$$

This system has an equilibrium $p = 1,000,000,000$, which is unstable. See Figure 1. We consider two different initial conditions.

If the antivirals are started after the virus had reached one billion copies, the initial condition is $x_0 = 1,438,187,806$, the number of copies reached at the end of Phase 2, that is 39 hours after being infected. In this case, the replication process cannot be stopped and the virus reaches one trillion copies in next 20 hours. This means the person will die in spite of hourly administration of antivirals. We illustrate this using Excel. See Figure 4.

If the antivirals are started before the virus has reached one billion copies, the initial condition is $x_0 = 958,925,204$, the number of copies reached an hour before the end of Phase 2. In this case, the virus can be eradicated from the body within 8 hours of administering antivirals. We illustrate this using Excel. See Figure 5.

Thus a person can be saved if antivirals are administered within 39 hours since the virus has entered the body. If it is 40 hours since the infection, it is too late. We once again emphasize that we consider only hourly increments in order to keep the model simple.

3. AN APPLICATION

Consider a soldier in the field getting infected by the virus studied in the previous section. *At* 0200 *hours on Tuesday a soldier's body temperature soared to* 104 *degrees. Thinking quickly back across his day, he realizes that he started feeling hot and achy at about* 2000 *hours the previous day. The soldier cannot receive medicine until* 1400 *hours the next day. Is it in time?*

Using the model developed and numerical solution obtained using Excel in the previous sections, we come up with the following time line. Also see Figures 6 & 7.

- 0600 hours on Monday - Working backwards, since the immune system responds after about 20 hours, this is the approximate time that the soldier got infected by the virus.
- 2000 hours on Monday - soldier felt hot and achy at this time: The soldier has, $16,384$ copies of the virus in his body at this time.
- 0200 hours on Tuesday - body temperature soars to 104 degrees : This is the time when immune system starts responding. At this time there are approximately $1,048,576$ copies of virus in the soldier's body.
- 2100 hours on Tuesday - This is 19 hours after the immune response has begun; the number of copies of the virus reaches one billion. Any time after that, the antivirals will be ineffective.
- 1400 hours on Wednesday - At this time, the number of copies of the virus reach one trillion and the soldier dies. Unfortunately, this is also the time that the medicine reaches the camp.

4. CLASSROOM EXPERIENCE

In Spring 2009, this work was first assigned as a project to an undergraduate math major. The student had prior knowledge of discrete dynamical systems, and spent several weeks

on solving the entire problem. This student then presented the results to other students and faculty in the mathematics department.

In Fall 2011, based on the previous success, this project was assigned to undergraduate biology majors enrolled in a mathematical biology class, as an in-class project for 90 minutes. These students had some experience with modeling discrete dynamical systems, determining fixed points and their stability, and determining numerical solutions using Excel. At the end of this project, these 12 participants were surveyed to assess their understanding of fixed points and stability. The survey also gauged their impression of whether this project showed them the role of mathematics in understanding biological phenomenon. The complete survey is provided in the Appendix.

4.1 Survey Results

Number of respondents = 12

Question number	% of correct responses
1	91.7
2	100
3	66.7
4	50
5	33.3

The first five questions assessed their understanding of an equilibrium point and its stability. Most of the students (91.7%) could determine a fixed point of a simple difference equation and all (100%) of them could determine its stability. More than half could visualize the dynamics of an abstract system with a stable fixed point (50-66.7%). However, only a third (33.3%) understood that if the initial condition (M_0) is a fixed point (even if it is unstable), then there is no change, and all iterates are the same as the initial condition ($M_t = M_0$ for all t).

Majority of the students (86.7%) recognized the use of piecewise functions in the project (question 6), and all of them (100%) agreed (or strongly agreed) that discrete dynamical systems (question 7) and using Excel to solve discrete dynamical systems (question 8) were useful in modeling biological phenomena. More than half (58.8%) could relate the stability of an equilibrium point to the stability of the underlying system (question 9).

Only three comments (question 10) were received: "very useful but takes a lot of abstract thinking to understand", "if unfamiliar with math terminology it can be difficult to understand what solution is being asked for", and "Good lab and interesting [application]".

The problem of predicting the concentration of a virus is a straight forward application of discrete dynamical systems. It can easily be done in one or two lab-periods with guidance from an instructor or over several weeks by students working independently.

5. REFERENCES

[1] *Introduction to Mathematical Modeling Using Discrete Dynamical Systems*, F. Marotto

[2] `http://www.mathmodels.org/problems/` Problem Number 20027: *Predicting the concentration of a Virus*

[3] PubMed `http://www.ncbi.nlm.nih.gov/pubmedhealth/PMH0002358/`

APPENDIX

Survey Questions

The following questions were asked on the survey given to students enrolled in a mathematical biology class after they were done working on the virus concentration project (Fall 2011).

Circle the best option for questions 1 - 9.

1. The equilibrium value of the discrete dynamical system $M_{t+1} = 0.75M_t + 1$ is
   - $M^* = 0$
   - $M^* = 2.0$
   - $M^* = 4.0$
   - does not exist
2. The equilibrium value of the discrete dynamical system $M_{t+1} = 0.75M_t + 1$ is **stable /unstable/ does not exist**?
3. The equilibrium value of a discrete dynamical system is 3.5 and is known to be stable. If the initial condition is $M_0 = 5.0$ which of these can be M_1?
   - $M_1 = 3.5$
   - $M_1 = 4.0$
   - $M_1 = -4.25$
   - $M_1 = 7$
4. The equilibrium value of a discrete dynamical system is 9.0 and is known to be stable. If the initial condition is $M_0 = 5.0$ which of these can be M_1?
   - $M_1 = 6.0$
   - $M_1 = 4.0$
   - $M_1 = 5.0$
   - $M_1 = 0$
5. The equilibrium value of a discrete dynamical system is 4.0 and is known to be unstable. If the initial condition is $M_0 = 4.0$ which of these can be M_1?
   - $M_1 = 3.5$
   - $M_1 = 4.0$
   - $M_1 = -7$
   - $M_1 = 7$
6. The virus concentration problem was solved by using piecewise defined functions since an updating function was defined for each phase.
   - Strongly agree
   - Agree
   - Neutral
   - Disagree
   - Strongly Disagree
7. Discrete dynamical systems are useful in modeling biological phenomena.

- Strongly agree
- Agree
- Neutral
- Disagree
- Strongly Disagree

8. Excel is a helpful tool for solving and visualizing solutions to discrete dynamical systems.
   - Strongly agree
   - Agree
   - Neutral
   - Disagree
   - Strongly Disagree
9. Stability of an equilibrium point for a discrete dynamical system corresponds to stabilizing of the system towards the equilibrium in the long run, for reasonable initial conditions.
   - Strongly agree
   - Agree
   - Neutral
   - Disagree
   - Strongly Disagree
10. **Any other comments:**

FIGURES

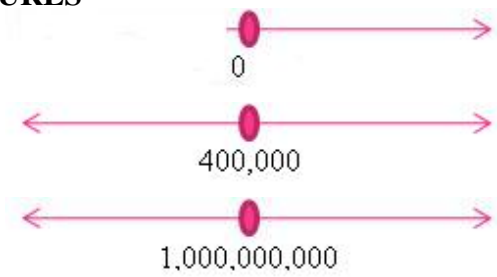
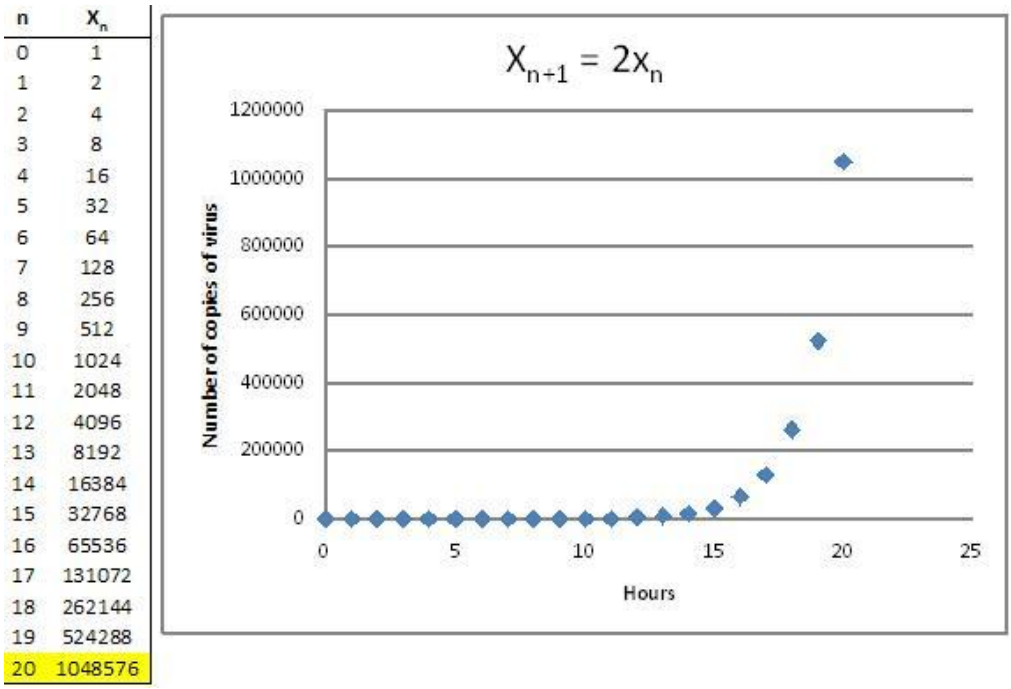
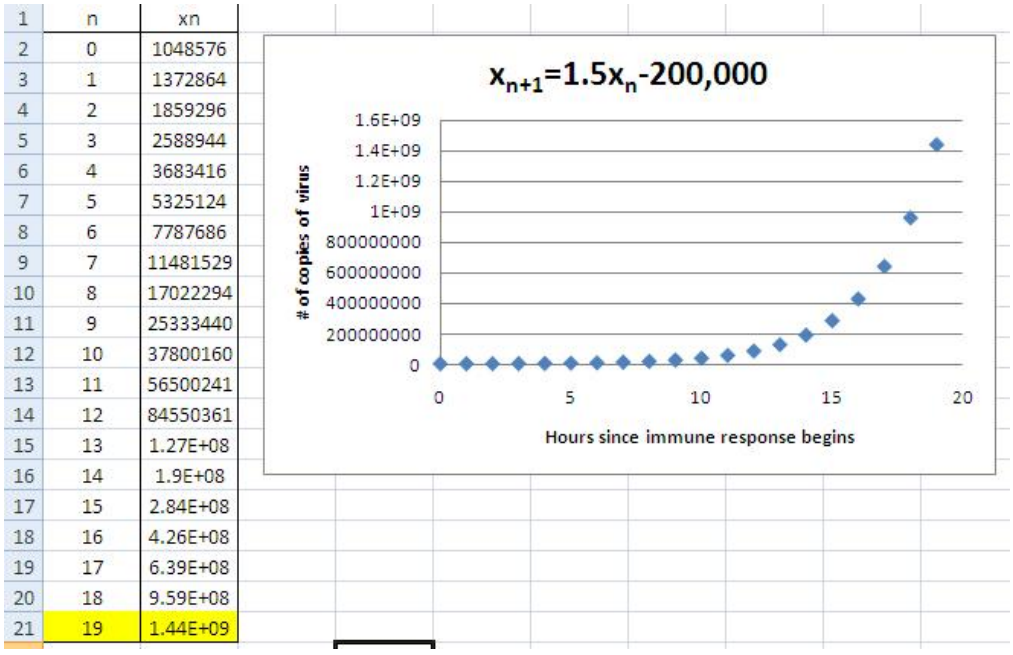


Figure 1: TOP: Phase line diagram for Phase 1, showing 0 as an unstable equilibrium. MIDDLE: Phase line diagram for Phase 2, showing $400,000$ as an unstable equilibrium. BOTTOM: Phase line diagram for Phase 3, showing $1,000,000,000$ as an unstable equilibrium.



n	x_n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576

Figure 2: Initial phase of the infection: It takes about 20 hours for the virus to reach one million copies, at which time the immune system begins to respond.



	n	xn
2	0	1048576
3	1	1372864
4	2	1859296
5	3	2588944
6	4	3683416
7	5	5325124
8	6	7787686
9	7	11481529
10	8	17022294
11	9	25333440
12	10	37800160
13	11	56500241
14	12	84550361
15	13	1.27E+08
16	14	1.9E+08
17	15	2.84E+08
18	16	4.26E+08
19	17	6.39E+08
20	18	9.59E+08
21	19	1.44E+09

Figure 3: Response of the immune system: It takes about 19 hours after the immune response begins for the virus to reach one billion copies.

n	xn
0	1.44E+09
1	1.657E+09
2	1.986E+09
3	2.479E+09
4	3.218E+09
5	4.327E+09
6	5.991E+09
7	8.487E+09
8	1.223E+10
9	1.785E+10
10	2.627E+10
11	3.89E+10
12	5.785E+10
13	8.628E+10
14	1.289E+11
15	1.929E+11
16	2.888E+11
17	4.327E+11
18	6.486E+11
19	9.724E+11
20	1.458E+12

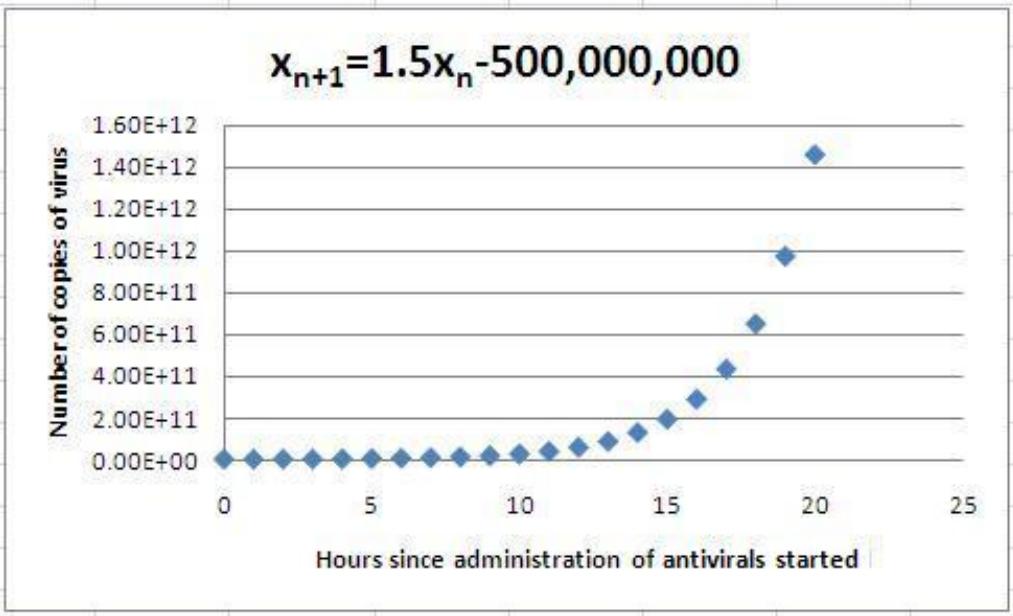


Figure 4: Administration of antivirals not helpful: If antivirals are administered after the virus has reached one billion copies, it takes about 20 more hours, for the virus to reach one trillion copies, at which time the infected person will die.

n	xn
0	9.59E+08
1	9.38E+08
2	9.08E+08
3	8.61E+08
4	7.92E+08
5	6.88E+08
6	5.32E+08
7	2.98E+08
8	-5.3E+07

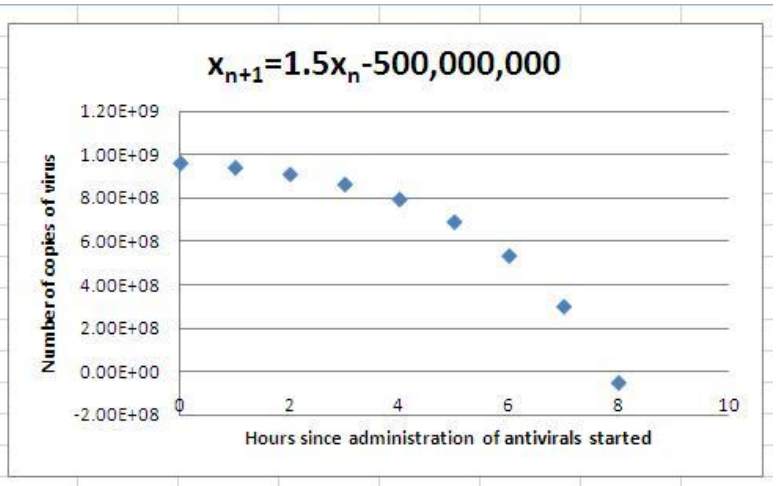


Figure 5: Administration of antivirals is helpful: If antivirals are administered an hour before the virus reaches one billion copies, it takes about 8 hours, for the virus to be eradicated from the body.

Day	Hours	X_n
Monday	0000	
Monday	0600	1
		2
		4
		8
		16
		32
	1200	64
		128
		256
		512
		1024
		2048
		4096
		8192
	2000	16384
		32768
		65536
	2300	131072

Day	Hours	X_n
Tuesday	0000	262144
	0100	524288
	0200	1048576
		1372864
		1859296
		2588944
		3683416
		5325124
		7787686
		11481529
		17022294
		25333440
	1200	37800160
	1300	56500241
		84550361
		1.27E+08
		1.9E+08
		2.84E+08
		4.26E+08
		6.39E+08
		9.59E+08
	2100	1.44E+09
		2.16E+09
	2300	3.24E+09

Day	Hours	X_n
Wednesday	0000	4.853E+09
	0100	7.279E+09
		1.092E+10
		1.638E+10
		2.457E+10
		3.685E+10
		5.527E+10
		8.291E+10
		1.244E+11
		1.865E+11
		2.798E+11
		4.197E+11
	1200	6.296E+11
	1300	9.444E+11
	1400	1.417E+12

Figure 6: This table shows the number of copies of the virus in the soldiers body, obtained using MS Excel. The discrete dynamical systems used are described in phases I and II in the paper. Given that the immune system responds on Tuesday 0200 hours (in the case described in the paper), we work backwards and estimate that the infection began on Monday at 0600 hours.

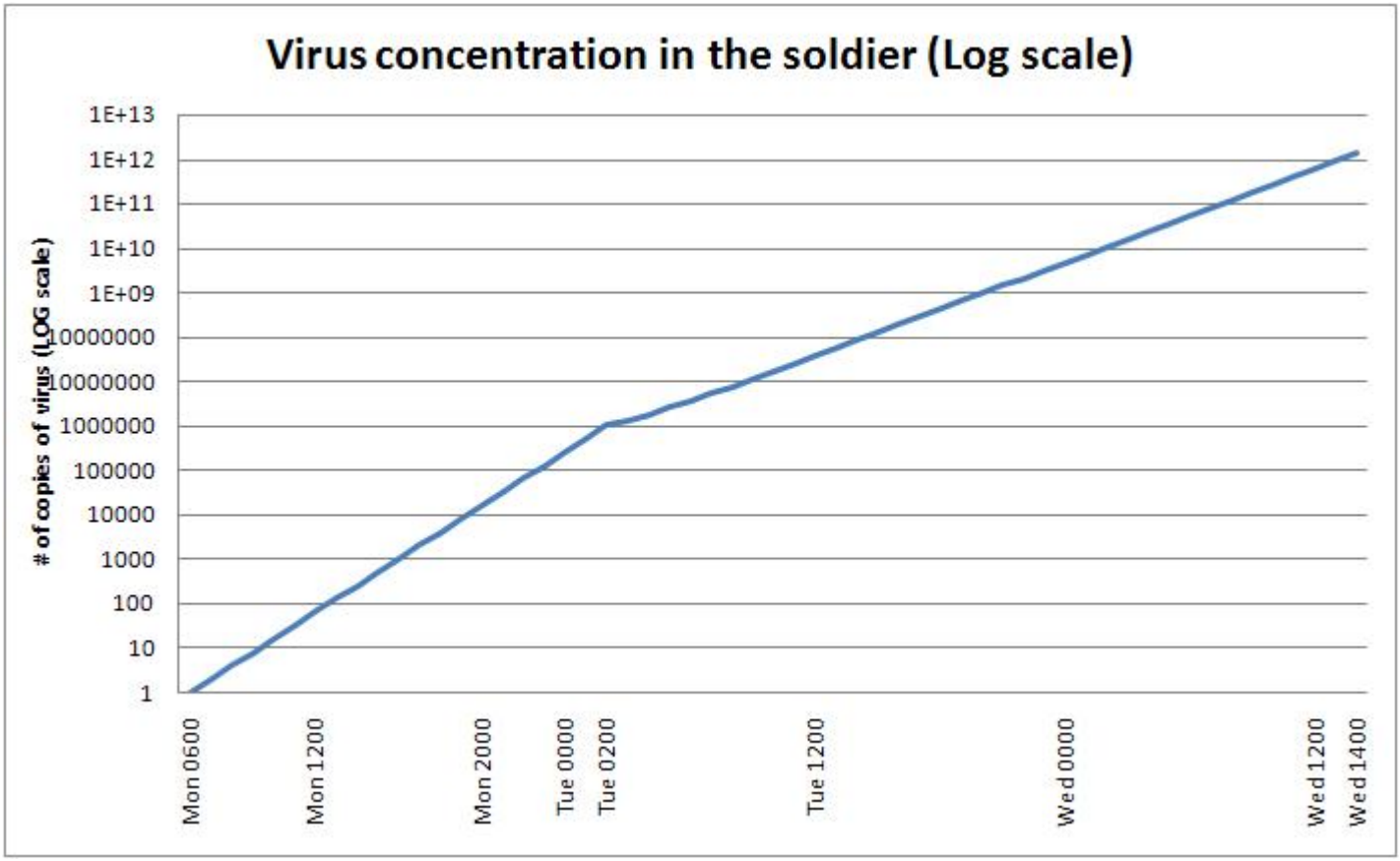


Figure 7: This graph shows the virus concentration on a log scale. The bend in the graph, is where the immune system begins to respond, which occurs on Tuesday at 0200 hours for the case described in the paper. The soldier dies, right as the antivirals arrive at the camp on Wednesday, at 1400 hours. The antivirals would have been ineffective anytime after Tuesday, 1900 hours.

Preparing Teachers to Infuse Computational Science into their Classroom Instruction

Susan J. Ragan
Maryland Virtual High School
14217 Bradshaw Drive
Silver Spring, MD 20905
301-989-0151
mvhsragan@gmail.com

Cheryl Begandy
Pittsburgh Supercomputing Center
300 S. Craig Street
Pittsburgh, PA 15213
412-268-5129
begandy@psc.edu

Nancy R. Bunt
Math & Science Collaborative
Allegheny Intermediate Unit
475 E. Waterfront Dr.
Homestead, PA 15120
412-394-4598
nancy.bunt@aiu3.net

Charlotte M. Trout
Washington County Public Schools
820 Commonwealth Ave.
Hagerstown, MD 21740
301-766-8790
troutcha@wcboe.k12.md.us

Scott A. Sinex
Prince George's Community College
301 Largo Road
Largo, MD 20774
301-341-3023
ssinex@pgcc.edu

ABSTRACT
Establishing consistent use of computer models and simulations in K-12 classrooms has been a challenge for the computational science education community. Scaling successful local efforts has been particularly difficult. In this article we describe how a training model from one place and time can be translated into a training model for another very different place and time if critical factors such as school system culture, professional development organization, local learning standards and goals, and collaboration between STEM disciplines are taken into account.

Categories and Subject Descriptors
K.3 **Computers in Education**

General Terms
Management, Design, Human Factors

Keywords
Secondary Science Education, Computer Models, Simulations, Professional Development Program

1. INTRODUCTION
A recent publication by the National Academy Press, Learning Science through Computer Games and Simulations, [1] devotes a chapter to the question of how to promote the use of computer simulations in science classrooms. While acknowledging that research on their effectiveness still needs additional input, they note some of the obstacles to the consistent adoption of computer simulations and games in K12 classrooms. One of the constraints cited has to do with the fact that teachers do not always possess the content knowledge and teaching strategies needed to achieve the full potential of the simulation.

The Maryland Virtual High School of Science and Mathematics (MVHS) has over seventeen years of experience working with Maryland teachers to help them use computer models and simulations as tools to assist their students' comprehension of complex concepts. In 2006, MVHS was invited by the Pittsburgh Supercomputing Center (PSC) to bring its training model to the Pittsburgh area. Four years later the Math & Science Collaborative (MSC) joined the effort to bring computational thinking and reasoning into science and math classrooms in southwestern Pennsylvania. Through our experiences in two neighboring states, we have found that school system culture must be taken into account when developing workshops for teacher training in computational science. In this article, we describe the approaches we have used and the lessons we have learned. The setting for our work in Maryland was 24 school districts which are county or city-wide and where statewide learning goals in science began to influence local instruction in the mid-90's. In contrast, in Pennsylvania, 500 local school districts are served by 29 intermediate units, which are regional educational service agencies covering the 67 counties. Statewide standards in science were also established in the 90s in Pennsylvania. Both states are signatories to the new Common Core State Standards [2] in mathematics and language arts—which will increase the potential for transfer of learning from one state to another.

2. BACKGROUND
2.1 Maryland Virtual High School
MVHS [3] has been working with teachers since 1994 when its first grant from the National Science Foundation (NSF) was awarded. Over its lifetime, MVHS has demonstrated methods by which science teachers can integrate computational science projects into their classrooms and, in the process, provide students with a modern and compelling introduction to the concepts of

science through inquiry-based activities while supporting both state and national standards for content. In the beginning, it was challenging to get schools involved in computational science since Internet connectivity was slow or nonexistent and there were barely any secondary school-level appropriate applications available. On the other hand, standardized testing had not yet taken a stranglehold on the curriculum so innovative teachers were free to try new things and their principals were supportive of their efforts.

During MVHS's first ten years, the focus was on preparing teachers to not only use and build computer models and simulations to help teach science concepts, but also to become trainers themselves so they could disseminate these new methods to their colleagues. A community of teachers who were comfortable assisting one another as coaches in the classroom was developed [4]. According to the final evaluation of MVHS's second NSF grant, CoreModels [5], the successes of MVHS were due to these key factors [6]:

- The close-knit, peer-to-peer structure of the CoreModels community of teachers
- The presence of a core group of relatively senior and especially dedicated teachers who were able to act as a vanguard in exploring the relative value of student-driven model construction and open-ended inquiry into systems
- Its sustained, long-term commitment to curriculum development, testing and revision

After CoreModels ended, MVHS used its funding through various partners to support the CoreModels teachers as they conducted workshops in their own school districts. Although the workshop evaluations from the teachers were overwhelmingly positive, resources were inadequate to determine how much computational science was actually being infused into the classroom outside of our CoreModels group. Anecdotal information implied that the impact of the computational science innovation varied significantly from one district to another.

2.2 Computation and Science for Teachers

In 2006, members of the MVHS team were invited to work with the Pittsburgh Supercomputing Center to develop a "Core Models" type of program within the Pittsburgh area schools. Using MVHS's "train the trainer" model, a series of one-week summer workshops with follow-up sessions during the school year were developed and delivered. Known as Computation and Science for Teachers (CAST) [7], three cohorts of teachers benefitted from the program and responded enthusiastically to the CAST workshops. The following comments are representative:

- "…this workshop provided me with more ideas on how to infuse technology in my lessons…"
- "…I saw connections between data collection and modeling, and better connection between math and science…"
- "…I am now thinking more about mathematical modeling, not just physical model…I plan to model more astronomy and chemistry concepts..."

Here are the words of one teacher on how CAST affected his teaching and his students:

> *"The CAST experience has been an invaluable asset in my teaching evolution. Initially it began as a completely utilitarian device that allowed me to provide students with an alternative to missed lab work, and then I began to explore the possibilities of doing labs that were either too costly or too dangerous to perform in a high school environment, but it has grown into something more...*
>
> *Systems modeling has fundamentally altered the way I teach, and not with just software. At the heart of systems modeling is the notion of connections. The examination of the connections can be done independent of the application software without losing the scientific and educational relevance. Demonstrating to a young mind the relationships that exist in nature without the cloud of mathematical complexity is crucial in the development of a scientifically literate generation."*

While the enthusiasm of these teachers is encouraging and many teachers did develop lessons to use with their students, the overall impact of this approach on student learning was difficult to measure and continued use by teachers has been limited. The fact that few teachers elected to return to the project for advanced training in the second and third years made the MVHS leaders question their approach. Is it possible for a traveling team of trainers (MVHS) to provide enough support to get a true learning community off the ground in another state? How can MVHS facilitate the integration of the knowledge and skills of the various players? The PSC has a vested interest in helping the local math and science education community embrace computational science, but has limited funding to do so. MVHS leaders are willing to share their expertise with others outside Maryland; but, clearly, that is not enough. In this article, the lessons learned from the adaptation of the Maryland approach to one that better fit southwestern Pennsylvania will be examined.

3. LESSONS LEARNED

3.1 Understand the State's Public Education Structure

Maryland is based on a county/city-wide system of public schools which means that most school districts have multiple secondary schools whose teachers are accustomed to meeting and collaborating with their peers. Pennsylvania, on the other hand, has over 500 school districts that have a long history of operating independently. Some districts are so small that a single teacher has responsibility for all biology or all chemistry or even all science classes at the high school level. These structural differences lead to school culture differences which affect the implementation of innovation as the examples below will illustrate.

In 1994, when the MVHS project was started, the Maryland State Department of Education (MSDE) was already discussing the establishment of core learning goals for science and other subjects [8]. When MVHS leaders went to MSDE with a proposal to bring Internet connectivity to several high schools for the purpose of improving science instruction, MSDE gave its whole-hearted support. Since Maryland is based on a county/city-wide model for school districts, it was relatively easy to contact 24 superintendents to explain the project to them. Sixteen middle and high schools in thirteen school districts [9] agreed to participate in a three year project in which Internet connectivity would be established at the school and two or three teachers from the school

would receive in-depth training in the use of the Internet as a collaborative tool for science.

At the end of the three years, the participating teachers were convinced that Internet connectivity was important, but they weren't sure how to use computational science in their classrooms. They had used their connectivity to collaborate with each other on projects such as the virtual earthquake, the boiling point project, and the Eratosthenes project [10]; but they admitted that sustaining such collaboration beyond the end of the project was unlikely. Those outcomes led to the formation of the MVHS CoreModels Project, another three-year NSF-funded project, whose mission was to prepare teachers to create and use computer models in their classroom teaching.

The CoreModels project included 27 high schools from 14 of the 24 school districts in Maryland. In 1997, there weren't many ready-made models for high school science, so project goals included creating a repository of models and lesson plans that would address the Maryland State Science Core Learning Goals [8]. To enable a system of local support for the teachers, the state of Maryland was divided into 3 regions, each with a lead teacher who was released half-time from classroom responsibilities to provide on-site assistance to the other teachers. In the first year of the project, summer workshops and quarterly Saturday workshops were held in a central location so that everyone could get to know one another and establish a basic level of competency. During the second and third years, the lead teachers conducted after-school and summer workshops in their own regions to encourage the development of learning communities where teachers could collaborate and learn from one another.

The collaboration between MVHS and the PSC began in 2006 with meetings to establish goals and analyze the availability of resources and funding. The PSC was very knowledgeable about the politics of their local schools so they carefully selected the school districts to be invited to the CAST project. Each district superintendent or designated representative attended a kick-off meeting where MVHS presented the objectives of the training and urged them to recruit two math/science teachers to attend the training with the goal of becoming a resource for the district. The ten teachers who attended the first year of training [11] were an excellent group of educators, but only two of them returned for further training. The CAST team hypothesized that the lack of local representatives who could travel to the schools to observe the teachers in action was a major reason for a lack of cohesiveness among the group members.

The second and third cohorts of teachers were recruited in various ways, including through the MSC [12], a regional organization charged with providing professional development for math and science instruction throughout western Pennsylvania. Again, these were excellent educators who enthusiastically embraced the computer models we showed them, but the CAST team had difficulty maintaining contact with them. Why was that? What were the differences between Pennsylvania and Maryland?

After much discussion, we concluded that the small, independent school district structure in Pennsylvania leads to a heightened sense of individualism, autonomy, and isolation on the part of the teachers. Having only one or two teachers trained in a number of geographically dispersed districts made the formation of any type of CAST "learning community" almost impossible. Without the support of peers, the teachers were challenged in just feeling comfortable using the new techniques, let alone teaching others. And when schools are so small that there is only one physics teacher (or chemistry or biology) and teachers have not been encouraged to collaborate within their own buildings, it is quite a culture shift to suggest that those same teachers become peer trainers for others. In Maryland, the county-wide school systems meant that subject-area teachers across the high schools in the district were expected to follow the same curriculum. In Pennsylvania, subject area teachers in the high school were likely to be free to teach their subject as they saw fit. Once we admitted that replicating the Maryland CoreModels experience in Pennsylvania was impossible, we were free to think more creatively. We realized that the key to working with teachers in Pennsylvania was to take advantage of the existing professional development infrastructure, the Math & Science Collaborative, which leads us to the next lesson.

3.2 Leverage the Professional Development Program at the Local or State Level

In Maryland, the school districts are large enough to have their own central office-based professional development staff who are responsible for providing training for teachers. Often these teacher trainers are master teachers who have been encouraged to leave classroom teaching to share their expertise with others. Since Pennsylvania has so many small school districts that cannot afford their own dedicated professional development staff, twenty-nine regional training sites have been established to provide professional development to their schools. The examples below illustrate how important it is to include the leaders of the professional development program in any teacher training initiative.

During the MVHS CoreModels project, the participating teachers were encouraged to keep their department heads and science supervisors informed of changes they were making in their classrooms. Some of the teachers enjoyed a great deal of support from their central office, others worked in a laissez-faire environment, and some found their supervisors to be almost hostile. In retrospect, it is clear that the CoreModels leadership needed to pay more attention to those instructional leaders to be sure that they felt valued and informed of the benefits coming into their schools.

However, those teachers who did enjoy the support of their districts were able to make a significant impact in their counties. In St. Mary's County, a CoreModels biology teacher rose to become the science coordinator where she was influential in infusing technology and computer modeling into the science curriculum across her district. A CoreModels physics teacher in the same county led many district-supported workshops for the teachers in his district and was so inspirational to his students that he was selected the 2011 Teacher of the Year for his school district [13]. In Anne Arundel County, a CoreModels physics teacher trained other teachers in his county even after the CoreModels project ended. When he retired from full-time teaching, he became involved in a NASA-funded project to bring systems modeling into a freshman engineering course in the county. In Montgomery County, a CoreModels biology teacher was hired by the Maryland State Department of Education where he has promoted the use of computer models across the science curriculum.

Washington County (WCPS), a semi-rural school district serving over 21,000 students (pre-K – grade 12), has experienced the most widespread use of computer models across secondary science classrooms due to the intersection of two important people – a CoreModels teacher with superior content and leadership skills

and a science supervisor with vision and patience. Like most school districts, WCPS educators struggled with finding the most effective ways to use the computer technology that fills their buildings. Fortunately, strong and consistent leadership in science instruction at the district level has resulted in a climate in which secondary science teachers are provided high-quality training and support. In particular, since the science supervisor understands the value of simulations as one tool among many to help students master science concepts, she provides both workshops and in-class support for the teachers as they learn to seamlessly integrate computer models in their teaching. While the CoreModels teacher was still full-time in the classroom, she led most of the summer workshops. Now that she is an instructional specialist in the science supervisor's office, she is able to work with teachers directly in their classrooms, even designing custom simulations to meet a teacher's needs [14]. The benefit of this partnership is that while the classroom teacher is building confidence in the effective use of the simulation, s/he is also gaining in content knowledge.

The first three years of CAST demonstrated that a regional infrastructure was needed to provide training and support and to establish a learning community. The MSC seemed to be the ideal partner. Leveraging the existing professional development system in southwestern Pennsylvania to bring computational science into the curriculum became our goal. We are now in the first year of implementing this strategy.

When the PSC contacted the MSC about including computational science modules in their annual training materials, they were very interested. At an introductory workshop [15], the MVHS team demonstrated three modeling approaches: agent-based, aggregate-based, and interactive spreadsheets. An interactive spreadsheet [16] illustrating coin flipping served as an introduction to the role that random numbers play in simulating real world events. Shodor's Forest Fire applet [17] was used to illustrate the role of probability in an agent-based model. Then a Vensim model of a forest fire spreading [18] was demonstrated to show the similarities and differences between a probabilistic and a deterministic approach to a problem. Since the role of computational thinking in math and science education was already being discussed in the MSC through the Math Science Partnership Circles, the CAST workshop training model proposed was received with open arms. The MSC was as eager to learn how to teach computational thinking and reasoning skills to area science and math teachers as the CAST team was to help them do so.

The next step was to define the organizational structure of the partnership. The PSC was in charge of fund-raising to make the CAST Professional Development Program (CAST-PDP) possible, the MSC was to provide the instructional specialists who would learn how to train and support teachers in using computational science, and MVHS would provide the training materials and the initial training to prepare the MSC for their role. Once the PSC had obtained funding from two Pittsburgh area foundations (DSF Charitable Foundation and the Frick Fund of the Buhl Foundation) to support the conversion of CAST workshop materials into well-defined modules, the partners began meeting to learn how to package the materials to fit the MSC's standard training methodology. After conducting walkthroughs of seven of the twelve modules, the MVHS team modified their materials to include details that the MSC trainers requested. During the summer of 2011, we piloted these modules with groups of teachers and trainers to receive more feedback and to further refine the modules.

The twelve CAST modules are designed to be used in various configurations to support those who wish to focus on using pre-built models as well as those who wish to become experts in one or more tools. The first group of modules already piloted (1, 2, 3, 5, 7, 10, and 12) are designed for model users [15]. The remaining modules, which focus on model building, will be piloted in 2012.

The titles of the modules are:

1. Introduction to Computational Reasoning
2. Deriving a Mathematical Model: An Experimental and Virtual Approach via Spreadsheets
3. Turning Multivariable Models into Interactive Animated Simulations
4. Building Interactive Excel Simulations
5. Introduction to Agent Modeling
6. Building an Agent Model with NetLogo
7. Introduction to Systems Modeling
8. Building a Systems Model with Vensim
9. Time-Based Models in Excel
10. Comparing Model Environments
11. Choosing a Model Environment to Build Your Model
12. What's Out There: Readily Available Models

3.3 Provide Materials Relevant to Local Learning Goals

When MVHS started in 1994, learning goals and expectations varied from one science classroom to another. We had to talk to the teachers in our project to find out what topics were common across all schools and build models addressing the associated concepts. Once MSDE established core learning goals for the sciences, it became easier to identify the computer models and simulations that would be relevant to all science teachers. Thanks to the recent release of the Common Core State Standards [2], computational science educators can now expect that models and simulations developed for one state will be applicable to other states. The fact that there are numerous resources available [19] means that those wishing to work with K12 school districts have numerous models and simulations to draw from. However, going to the trouble of finding a selection of models that will have your particular audience experiencing "ah-ha" moments will ensure that your message is heard and embraced.

For the CAST-PDP project, the MVHS trainers began by selecting pre-built models that had been shown to be classroom relevant through their use in other teacher training workshops. These models were then linked to specific goals from the Pennsylvania Math and Science Standards [20], and they were demonstrated to the MSC science and math coordinators to assess their impact. If the instructional leaders had "ah-ha" moments, the models were deemed worthy of inclusion in the modules. On the other hand, if a model seemed too confusing or uninteresting, it was either significantly modified or abandoned in favor of a more compelling topic. The models currently in use are available at http://www.psc.edu/eot/k12/2011yr.php.

Another challenge in the development of materials to help teachers infuse computational science in their classrooms is the pedagogy that needs to accompany the models. It isn't enough to know where to find the tools. One also needs to know how to use the tool effectively. In the CAST-PDP project, we demonstrate an inquiry-based pedagogy with every model we include in our modules. Our goal is to prepare teachers to be critical users of computational science tools so that they know how to elicit both computational and conceptual learning from their students.

3.4 Provide Opportunities for STEM Teachers to Work Together

Workshop leaders addressing a mixed audience of science and math teachers have a challenging task ahead of them. The silo-based system of college majors has resulted in a lack of communication among biology, chemistry, physics, earth/space science, and math teachers except in very small schools where one person teaches multiple subjects. But, teachers are like students in that they construct their knowledge through interaction with peers, applying their ideas in the classroom, discussing results to refine their understanding, and extending their learning to new situations [21]. Research has shown that richer instruction and improved student learning occur when science, math, technology and engineering teachers and professionals collaborate [22].

The MVHS CoreModels project found that explicit teaching of math terminology helped the science teachers to use those terms in their classroom, thereby helping their students make a connection between math and science. Linear growth, exponential growth or a J-curve, bounded growth or an S-curve – all of these terms have their place in math and science. It is important to make the teachers aware of differences and similarities in terminology so they can help their students see the connections. And, science teachers were often surprised to learn that similar graph shapes were found across their specialty areas. Moreover, the realization that similar graph shapes resulted from similar model structures was a revelation to them. These crossovers between mathematics and the sciences gave the teachers a common ground for discussion about issues of pedagogy and student learning

The current southwest Pennsylvania Math Science Partnership (MSP), facilitated by MSC, creates professional learning communities (PLCs) both within districts, and regionally. Via MSC, the CAST modules will be introduced within the regional learning communities, in order for those leaders to take their understandings back to their building based PLCs. That will enable more than 100 teachers, who are already involved in strengthening their teaching of mathematics and science to consider computer modeling as a new strategy. These regional professional learning communities meet 8 days each school year. The schedule for the 2011-2012 school year begins with the introductory CAST module, facilitated by MSC coordinators who have been trained by the CAST team and who have contributed to the module development. Two additional PLC days will be devoted to additional CAST modeling modules, this time presented centrally in the context of a regional conference bringing all 120 PLC participants to the Carnegie Science Center. Those sessions will be presented by the MVHS team, as the MSC team continues to build capacity for on-going support. The other five PLC sessions will allow the participants to integrate their learning into the on-going strengthening of their practice, supported locally by the MSC Coordinators. In addition, ten day summer institutes, supported by the MSP funding, will feature CAST modules, as an integrated means of exploring life sciences with school year follow-up allowing continuing support.

Our hope is that the science teachers from the school districts in the Pittsburgh area will evolve into self-sustaining PLCs in which teachers share their methods for using computational science in their classrooms. We have already seen a successful example of this in the WCPS district in Maryland where science teachers at both the middle and high school levels incorporate modeling into their instruction. The 2010-11 school year began with a new series of middle school science textbooks which directed students to use a variety of modeling tools, some free like NetLogo [23], others requiring costly GIS software. To enable the full use of the textbook without additional expense, the district-level science instructional specialist collaborated with the teachers to develop interactive web pages incorporating the pertinent NetLogo applets along with the GIS capabilities of NetLogo. At various points during the year, prior to the units in which the models would be used, teachers participated in professional development workshops designed to familiarize them with the models and the applet format. When asked, the science specialist co-taught lessons with teachers as they were using the model applets with students. Based on the statistics option built into the wiki, it is clear that teachers and students are using the models on a regular basis to support student learning.

At the high school level, WCPS is seeing the active use of simulations across the sciences. Physics teachers use the PhET [24] site on a regular basis and some are still using Interactive Physics [25] in their classrooms. Biology teachers are using a sequence of NetLogo based web pages developed by the science specialist to help students with concepts of genetics [14]. This kind of systemic change happens only through consistent opportunities for collaboration.

4. CONCLUSION

Preparing teachers to infuse computational science into their classroom instruction is hard work that requires patience, persistence, flexibility, creativity and time. To reach beyond the subset of teachers who actively seek new learning experiences to influence those who are comfortable with the status quo requires the confluence of several factors, including motivation to learn new skills, access to well-designed training materials, local professional development support, and a collaborative environment. External motivation in the form of national standards is now available through the Common Core State Standards which include this goal for grades 11-12 in science: “Synthesize information from a range of sources (e.g. texts, experiments, simulations) into a coherent understanding of a process, phenomenon, or concept, resolving conflicting information when possible” [26]. At this time, all but six states have adopted these standards [27]. The National Research Council’s new K-12 science framework includes systems and system models as crosscutting concepts throughout science and engineering [28]. The years of experience that MVHS and others in the computational science education community have accumulated has resulted in an abundance of models, simulations, and training methods available for use. The challenge is forging the alliances with local professional development leaders so those materials can be disseminated more broadly. Finally, as more schools embrace the philosophy behind professional learning communities [29], teachers will feel empowered to try new approaches, knowing that they will have the freedom and support to experiment until they have achieved mastery. We believe that the time is right for advancing the integration of computational science into classroom instruction.

5. ACKNOWLEDGEMENTS

The authors wish to thank Dr. Robert M. Panoff, Executive Director and President of the Shodor Education Foundation, for the resources and inspiration he has provided the computational science education community. We also thank Sandy Graff, Supervisor of Secondary Science for the Washington County Public Schools, for her years of support of MVHS initiatives in

WCPS science classrooms. In addition, we wish to thank all of the Maryland and Pittsburgh-area educators whose participation in and feedback for workshops held over the years have made our current program possible. The following funding agencies provided the support that made this work possible: The National Science Foundation, the DSF Charitable Foundation, the Frick Fund of the Buhl Foundation, the Grable Foundation, and the Heinz Endowments.

6. REFERENCES

[1] Honey, M and Hilton, M, eds., **Learning Science Through Computer Games and Simulations,** The National Academies Press, Washington DC (2011) http://www.nap.edu/catalog.php?record_id=13078

[2] Common Core State Standards http://www.corestandards.org/ *Accessed August 12, 2011*

[3] MVHS website http://mvhs.shodor.org/origins/orindex.html *Accessed August 12, 2011*

[4] Killion, J. and Harrison, C. **Taking the Lead,** National Staff Development Council, Oxford, Ohio (2006)

[5] MVHS Core Models http://mvhs.shodor.org/coremodels/cmindex.html *Accessed August 12, 2011*

[6] Friedman, W. and Culp, K., **Evaluation of the CoreModels project: Final report,** Education Development Center/Center for Children & Technology, New York, New York, (2001)

[7] CAST website http://www.psc.edu/eot/k12/cast.php *Accessed August 12, 2011*

[8] MSDE website http://mdk12.org/assessments/high_school/index.html *Accessed August 12, 2011*

[9] MVHS school list http://mvhs.shodor.org/origins/MVHSSchools.html *Accessed August 12, 2011*

[10] MVHS Collaborative Projects list http://mvhs.shodor.org/mvhsproj/projects/collab.html *Accessed August 12, 2011*

[11] CAST 2006 link http://www.psc.edu/eot/k12/first_year.php *Accessed August 12, 2011*

[12] MSC website http://www.aiu3.net/Level2.aspx?id=480 *Accessed August 12, 2011*

[13] St. Mary's County website http://www.somdtoday.com/2011/04/15/skinner-selected-as-smcps-teacher-of-the-year/ *Accessed August 12, 2011*

[14] Butterfly Genetics website http://web.me.com/scitrout/NetLogo%20Models/butterflyintro.html *Accessed August 12, 2011*

[15] CAST Modules http://www.psc.edu/eot/k12/2011yr.php *Accessed August 12, 2011*

[16] Flipping Pennies http://academic.pgcc.edu/~ssinex/excelets/flipping_pennies.xls *Accessed August 12, 2011*

[17] Shodor's link to Fire!!! http://www.shodor.org/interactivate/activities/Fire/ *Accessed August 12, 2011*

[18] MVHS link to Forest Fire page http://mvhs.shodor.org/mvhsproj/forestfire/forestfire.html *Accessed August 12, 2011*

[19] Computational Science Education Reference Desk http://www.shodor.org/refdesk/ *Accessed August 12, 2011*

[20] Pennsylvania State Standards for Math and Science http://www.pdesas.org/Standard/StandardsDownloads *Accessed August 12, 2011*

[21] Krajcik, J.S., Blumenfeld, P.C., Marx, R.W., and Soloway, E., A collaborative model for helping middle grade science teachers learn project-based instruction. *Elementary School Journal,* 94, 483-497 (1994).

[22] STEM Teachers in Professional Learning Communities http://www.wested.org/online_pubs/1098-executive-summary.pdf *Accessed August 12, 2011*

[23] NetLogo http://ccl.northwestern.edu/netlogo/ *Accessed August 12, 2011*

[24] PhET http://phet.colorado.edu/ *Accessed August 12, 2011*

[25] Interactive Physics http://www.design-simulation.com/ip/index.php *Accessed August 12, 2011*

[26] Common Core State Standards for Literacy in History/Social Studies, Science, and Technical Subjects, page 62, http://www.corestandards.org/assets/CCSSI_ELA%20Standards.pdf *Accessed August 12, 2011*

[27] Common Core State Standards Adoption Status http://www.corestandards.org/in-the-states *Accessed August 12, 2011*

[28] Committee on Conceptual Framework for the New K-12 Science Education Standards; National Research Council, **A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas**, The National Academies Press, Washington DC (2011) http://www.nap.edu/openbook.php?record_id=13165&page=R1

[29] DuFour, R., Schools as Learning Communities. *Educational Leadership, 61, 6-11, (2004).* http://pdonline.ascd.org/pd_online/secondary_reading/el200405_dufour.html

Introducing Matrix Operations through Biological Applications

Angela B. Shiflet
Department of Computer Science
Wofford College
Spartanburg, S. C. 29303 USA
001-864-597-4528
shifletab@wofford.edu

George W. Shiflet
Department of Biology
Wofford College
Spartanburg, S. C. 29303 USA
001-864-597-4625
shifletgw@wofford.edu

ABSTRACT
For the Blue Waters Undergraduate Petascale Education Program (NSF), we developed a computational science module, "Living Links: Applications of Matrix Operations to Population Studies," which introduces matrix operations using applications to population studies and provides accompanying programs in a variety of systems (C/MPI, MATLAB, Mathematica). The module provides a foundation for the use of matrix operations that are essential to modeling numerous computational science applications from population studies to social networks. This paper describes the module; details experiences using the material in two undergraduate courses (High Performance Computing and Linear Algebra) in 2010 and 2011 at Wofford College and two workshops for Ph.D. students at Monash University in Melbourne, Australia, in 2011; and describes refinements to the module based on suggestions in student and instructor evaluations.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer and Information Science Education - Computer Science Education, Curriculum

General Terms
Design, Experimentation, Measurement.

Keywords
Computational Science, Matrices, Linear Algebra, Educational Modules, High-Performance Computing, Petascale, Blue Waters, Undergraduate.

1. INTRODUCTION
The Blue Waters Undergraduate Petascale Education Program [1] with NSF funding is helping to prepare students and teachers to utilize high performance computing (HPC), particularly petascale computing, in computational science and engineering (CSE). UPEP supports three initiatives:

- *Professional Development Workshops* for undergraduate faculty
- *Research Experiences* for undergraduates
- *Materials Development* by undergraduate faculty for undergraduates

The Materials Development initiative has as its goal "to support undergraduate faculty in preparing a diverse community of students for petascale computing."

For this program, the authors developed and class tested the computational science modules "Living Links: Applications of Matrix Operations to Population Studies," which is available at [2] on the UPEP Curriculum Modules site. This paper describes and discusses the module and our experiences using it in the courses High Performance Computing and Linear Algebra at Wofford College [3] in 2010 and 2011, respectively, and using some of the examples, applications, and projects from the module in "Quantitative Modelling Using MATLAB: Introduction," a component of two 2011 workshops ("Introduction to Computational Thinking" and "Computational Workshop for the Life Sciences: Bringing Computation to Life") for Ph.D. graduate students, sponsored by the Monash eScience and Grid Engineering Laboratory (MeSsAGE Lab) at Monash University in Melbourne, Australia [4].

Several of the students in the classes at Wofford are obtaining the Emphasis in Computational Science (ECS). Bachelor of Science students may obtain an ECS by taking Calculus I, Introduction to Programming and Problem Solving (in Python), Data Structures (in Python and C++), Modeling and Simulation, and Data and Visualization and doing a summer internship involving computation in the sciences [5]. Matrices are an important data structure in numerous computational models, and introducing operations on matrices with population applications provides motivation to students in mathematics, computer science, and the other the sciences as well as in the Emphasis in Computational Science.

2. MODULE

2.1 Pedagogy

Prerequisites for the module "Living Links: Applications of Matrix Operations to Population Studies" are minimal, requiring the maturity to read the material but no programming or calculus background. Students who used the module at Wofford College ranged from first- to fourth-year with majors from biology, chemistry, physics, mathematics, computer science, and undecided. Those attending the workshops at Monash University were mainly Ph.D. science students from such areas as biology, chemistry, engineering, mathematics, psychology, pharmacy, and medicine. The module provides the biological background necessary to understand the applications, the mathematical background needed to complete the exercises and projects, and references for further study. Ten (10) multi-part quick review questions with answers at the end of the module provide immediate feedback. The module also provides seventy-five (75) exercises for reinforcement and practice and seven (7) project assignments for further exploration using a computational tool.

To help with implementation of models using matrix operations, example solutions involving equivalence testing, vector addition, and scalar and matrix multiplication are available for download from the UPEP Curriculum Modules site [2] in the following systems: *MATLAB*, *Mathematica*, and C with MPI for high performance computing. (Blue Waters Student Intern Jesse A. Hanley implemented the latter.) Several datasets for use in projects also accompany the module.

2.2 High Performance Computing in Module

In line with the aims of UPEP, the module has an introductory section on "Population Matrices and High Performance Computing" that discusses the need for high performance computing (HPC) within the context of an ecological study of blue crabs. One such study has collected over a terabyte of data (10^{12} characters), and the researchers estimate that simulations would take about 5.7 years on a sequential computer. As the module points out, "With such massive amounts of data and such intensive computations, researchers must used high performance computing with multiple computer processors to store the data and large matrices and to perform the needed simulations in a reasonable amount of time" [6].

The section on "Population Matrices and High Performance Computing" can be covered for information only, as a starting point for class discussion, or as motivation for the students' own HPC project development. Moreover, students can develop sequential or high performance computing versions of the projects. For example, three projects use synthetic datasets for the activities of the population of Portland, Oregon, generated from real data by the Network Dynamics and Science Simulation Laboratory (NDSSL) at Virginia Technical University [7]. One of the module's projects requires high performance computing to process NDSSL's synthetic data involving 1,615,860 people having 8,922,359 activities.

2.3 Module Content and Applications

Matrices, vectors, and operations involving these data structures are essential to many network/graph-based computational science models. Thus, the module introduces the following important and foundational mathematical concepts: vectors, vector addition, multiplication of vectors by a scalar, dot product, matrices, scalar multiplication, matrix sums, matrix multiplication, square matrices, and the association of matrices and systems of equations.

These concepts are introduced in an environment of biological applications. As indicated above, the first section motivates the study of vectors and matrices and the need for high performance computing (HPC) with a discussion of a scientific study of blue crabs that includes high performance computing simulations.

The foundational material includes a discussion of vectors with such terms as "element," "size," and "index" and such concepts as equality and addition of vectors and multiplication by a scalar. Examples involve vectors of simulated changes in populations of competing white tip reef sharks and black tip sharks in an area.

Then, a section on "Dot Product" uses another biological example in estimating the number of eggs laid by Hawaiian green sea turtles in one year. As indicated in the module, scientists around the world have studied the magnificent green sea turtle and used mathematics and computer science to make predictions about their populations in efforts to keep these them from extinction. A figure features the Caribbean Conservation Corp John H. Phipps Biological Field Station Costa Rica for the study of the green sea turtle. The section's example begins: "We deal with a different type of multiplication in estimating the number of eggs laid by Hawaiian green sea turtles in one year. We can consider their life cycle to be in five stages with egg layers in two stages, novice breeders of age 25 years and mature breeders from age 26 through 50 years. On the average, a novice breeder lays 280 eggs in a year, and a mature breeder lays 70 eggs per year. We can combine these data in a vector $\mathbf{e} = (280, 70)$. Suppose also that there are 291 novice and 9483 mature breeders, which we store in the vector $\mathbf{b} = (291, 9483)$. To approximate the total green sea turtle egg production in a year, we multiply together corresponding terms and add the results, as follows:

$$\begin{aligned} \mathbf{e} \cdot \mathbf{b} &= (280, 70) \cdot (291, 9483) \\ &= 280 \cdot 291 + 70 \cdot 9483 \\ &= 81{,}480 + 663{,}810 \\ &= 745{,}290 \text{ eggs} \end{aligned}$$

This type of multiplication, the **dot product**, involves two vectors of the same size and results in a number, *not* another vector."

After a discussion of the option of writing a dot product with the first term as row vector and the second as a column vector, the section concludes with the following quick review question, whose answers are in a section at the end of the module:

"**Quick Review Question 4** The first stage in the life of the Hawaiian green sea turtle, consisting of eggs and hatchlings, occurs during the first year. Stage 2, juveniles, extends from year 1 to 16. Suppose 23% of the hatchlings survive and move to stage 2, while 67.9% of those in Stage 2 remain in that stage each year. In one year, suppose Stage 1 has 808,988 individuals, and Stage 2 has 715,774 (Green Sea Turtle).

- **a.** Give a vector, **p**, with real number elements representing the percentages.
- **b.** Give a vector, **s**, storing the individuals in Stages 1 and 2.
- **c.** Using variables **p** and **s**, not the data, give the vector operation to determine the number of individuals that will be in Stage 2 the following year.
- **d.** Calculate this value."

With the foundation of vectors and vector operations, the next three sections lead the student carefully through explanations of definitions involving matrices, scalar multiplication, matrix sums, and matrix multiplication using examples with populations of white tip reef sharks and black tip sharks and additional quick review questions.

A subsequent section on "Square Matrices" defines "square matrix" and "diagonal element." These terms are illustrated with hypothetical data of the distribution of ABO blood types (A, B, AB, O) for mothers and newborns (multiple births omitted) in a county over a year. In another example involving a square matrix, a table (Table 1 here) presents similarity measures (specifically, Euclidean distances) of the 18S rRNA sequences of pairs of animals, where smaller numbers indicate closer relationships. Using this example as motivation, the section defines "symmetric matrix".

Table 1. Similarity measures (specifically, Euclidean distances) of the 18S rRNA sequences of pairs of animals (Table 3 in Lockhart, 1994)

	Frog	**Bird**	**Human**	**Rabbit**
Frog	0	0.316	0.350	0.336
Bird	0.316	0	0.130	0.102
Human	0.350	0.130	0	0.028
Rabbit	0.336	0.102	0.028	0

For the concluding section on "Matrices and Systems of Equations," we return to the earlier example involving the dot product, where a Hawaiian green sea turtle novice breeder lays an average of 280 eggs per year, while a mature breeder only lays 70. The material continues, "Instead of specifying the number of turtles in each category, let n be the number of novice breeders and m the number of mature breeders with $\mathbf{b} = (n, m)$. In general, the average annual egg production, a, is computed as follows:

$$\mathbf{e} \cdot \mathbf{b} = (280, 70) \cdot (n, m)$$
$$= 280n + 70m = a$$

Thus, the dot product translates into one side of a linear equation." Moreover, the section indicates that a matrix-vector product involving the black-tip/white-tip shark application is equivalent to a system of linear equations.

2.4 Module Exercises and Answers

After the body of educational material, the module contains a section with 75 exercises. Instructions encourage students to check their work with a computational tool, while a subsequent section has answers to 15 exercise parts. Many exercises are routine practice problems with vectors and matrices. However, several "word problems" involve applications, such as real spectrophotometer readings to indicate the number of bacteria in a broth; development of an age-structured model for an animal; a threshold matrix; and a dither matrix for enhancement of a digital image, such as a medical image from a CT (computerized tomography) scan.

2.5 Module Projects

After the reinforcement of concepts in the exercises, seven large projects are available for students to complete as individuals or with a team. Instructions indicate to use a computational tool, optionally with high performance computing except as indicated.

Three projects consider matrices and vectors as part of network-based epidemiology simulations. The current module can serve as a basis for another Blue Waters module by the authors, "Getting the 'Edge' on the Next Flu Pandemic: We Should'a 'Node' Better," which develops this graph theory based model in detail [8]. In preparation for development of simulations in this subsequent module or as stand-alone applications, the three projects lead students through various aspects involving vectors and matrices, such as formation of vectors of IDs for people and locations and of a corresponding people-to-people connection matrix.

Four projects employ colon crypt data that were output from simulations by Ornella Cominetti and the authors. The simulations were performed with Chaste (Cancer, Heart and Soft Tissue Environment), "a general purpose simulation package aimed at multi-scale, computationally demanding problems arising in biology and physiology" that a team centered in the Computational Biology Group at Oxford University Computing Laboratory is developing [9]. Scientists believe that colorectal cancer originates in tiny crypts that descend from the colon's epithelium into the underlying connective tissue. Projects, which use data downloadable from the website [2], are variations of those employed in research at Oxford and include plotting the trajectory of a cell in the crypt; generating a stacked bar chart of the average number of cells in various categories by time; plotting the mean migration velocities of cells at different heights in the crypt; and plotting the mean spatial correlation of velocity, a metric of the amount of coordinated movement of the cells, along with standard error bars. The projects develop the background necessary for their completion, and instructions ask the students to discuss their results.

2.6 Blue Waters UPEP Internship Involvement

During the summer of 2010 and following academic year, student Jesse Hanley held a Blue Waters UPEP Internship to develop a parallel version of a program using C and MPI to support this and other modules. His program accompanying the module is available on the NCSI UPEP Curriculum Modules site [2]. Jesse's experiences as an intern and program developer enhanced his understanding of programming in general and HPC in particular.

3. TESTING AND EVALUATION

3.1 Class Testing in High Performance Computing

The first author taught Wofford College's High Performance Computing (HPC) course (COSC 365) in the spring of 2010. A mixture of Emphasis in Computational Science (ECS) students and computer science majors (five students: one biology/ECS, one chemistry/ECS, one computer science/chemistry/ECS, two computer science; sophomore to senior level) populated the class. All students had taken Data Structures with programming in Python and C++ and at least one other computer science or computational science course.

In preparation for discussion of HPC implementation of matrices and of applications involving matrices, the class was assigned reading of a preliminary version of the "Living Links" module, various graded and non-graded exercises from the module, and a quiz with questions taken from its Quick Review Questions. This background helped to prepare the students to develop several projects, including population dynamics model of skates, which

are similar to sharks; performance analyses of sequential and parallel programs that raise square matrices of various sizes to a range of powers, run on a local cluster, NCSA's Teragrid computer Abe (a Dell Intel 64 Linux Cluster), and NICS' Teragrid computer Kracken (a 99072-processor Cray XT5 computer) [10]; and a social network/individual-based epidemiology simulation using another Blue Waters curriculum module by the authors, "Getting the 'Edge' on the Next Flu Pandemic: We Should'a 'Node' Better" [8].

3.2 Evaluation in High Performance Computing

The students demonstrated their understanding of the module's material with their performance on homework and a major exam. Feedback, both formal and informal, from class members about the preliminary version of the model helped revisions and formulation of the completed version.

3.3 Class Testing in Linear Algebra

Linear Algebra at Wofford College is a sophomore-level class required of mathematics majors and minors and computer science majors. In spring 2011, after covering the basics of vector operations, the instructor, Dr. Joseph Spivey, required that the students read the module before discussing matrix multiplication and its applications in class. The professor used one 50-minute class for the module but mentioned that had he covered everything in the module, the material would have taken one-and-a-half to two full periods. After the class, he assigned a number of the exercises to be done by hand.

3.4 Evaluation in Linear Algebra

Immediately after using the material, students in Linear Algebra along with their professor completed a questionnaire about the module. The questionnaires for the Linear Algebra class had the students rate the following statements from 1 (strongly disagree) to 5 (strongly agree):

- I understood the science applications in the module.
- I understood the mathematics in the module.
- The module was readable.
- The Quick Review Questions helped me understand the material.
- The exercises helped me understand the material.

Means of the responses of the 20 students in the class, which were between 4.30 and 4.50, reflect favorably on the module (see Table 2). For similar questions, the professor gave ratings of all fives (5s).

Table 2. Means and standard deviations of $N = 20$ responses to rated questions, 1 (strongly disagree) to 5 (strongly agree)

Question	Mean	Standard Deviation
I understood the science applications in the module.	4.30	0.47
I understood the mathematics in the module.	4.50	0.69
The module was readable.	4.33	0.69
The Quick Review Questions helped me understand the material.	4.38	0.84
The exercises helped me understand the material.	4.34	0.68

Students and the faculty member were also asked to elaborate about the above scores. Dr. Spivey commented, "I was impressed with how much they understood even before I went over it in class. Some students said to me that they learn better through the use of applications and that the math was explained very well." Student comments reflected the same impressions: "I felt the module was very well put together in a way that was easy to follow with just enough breaks in text to keep track of equations but not enough to lose track of the subject." "I like the idea of using math as a tool to help predict how our actions may affect populations or show how past actions did." "The applications allowed for better understanding of the mathematics material itself."

Participants were also asked to indicate what they liked best about the module. The instructor said, "The Quick Review Questions were nice, and several students wrote about that in their journal entries." Some class members indicated on their questionnaires that they liked these best, and one student wrote, "The Quick Review Questions served both as a learning tool and a reference for the exercises as it gave me practice, and its solutions pages in the back showed enough for me to really grasp the processes of some problems…." Other students liked best the "readability" of the module, "the explanations," the arrangement of the material, and "the real-world applications." As one student wrote, "What I liked most about the module were the real life and science applications that involved the topics, many of which I am also studying in my science courses. These references kept the topics intriguing."

The questionnaire also asked what they found most difficult about the module and to give corrections and suggestions for improvement. For revision, Dr. Spivey suggested, "You may want to make it clear at the beginning that the answers to the Quick Review Questions are at the end." In response to this comment, we added such a paragraph statement before the first question about the location of the answers and how to use the Quick Review Questions and answers as a learning tool. As the instructor suggested, we also revised the phrasing of one of the exercises and added four matrix multiplication exercises. Two of the students requested a section for answers to selected exercises. In response to this suggestion, we added such a section with answers to fifteen (15) exercise parts.

In reply to the questionnaire's request for further comments, Dr Spivey summarized, "The students responded well to the material. They enjoyed the reading and could follow it easily. They also really enjoyed learning about the applications." Student comments were in a similar vein: "I think the module was wonderfully written and everything was explained in an easy-to-read way." "Their [The authors'] ability to relate vectors and matrices to sea creatures is astounding, and it gives the math an entirely new dimension. It shows the reader how widespread the

influence of mathematics is, while helping them learn new techniques and concepts." "I wish all math textbooks were easy to read and understand like this module!"

3.5 Workshop Testing

In February, 2011, Monash eScience and Grid Engineering Laboratory (MeSsAGE Lab), directed by Professor David Abramson, sponsored two computational science workshops at Monash University in Melbourne, Australia, for Ph. D. students. Dr. Bob Panoff, Executive Director of the Shodor Foundation [11], was the main leader of the first week-long workshop, "Introduction to Computational Thinking," which had an afternoon session conducted by the authors on "Quantitative Modelling Using MATLAB: Introduction." Later in the month, the authors lead the second week-long workshop, "Computational Workshop for the Life Sciences: Bringing Computation to Life," in which half the time involved modeling using MATLAB, including a similar introduction. Other topics in this half were user-defined functions, looping, decisions, model fitting, and a day on parallel programming. The format of this part of the workshop was presentations interspersed with frequent exercises for participant pairs to complete. The presentations and exercises included a number of examples, applications, and projects from the "Living Links" module. For example, the class developed versions of three projects from the module: visualizing the trajectory of a simulated cell in a colon crypt; plotting the mean migration velocities of simulated crypt cells; and modeling a network of individuals in a community with matrices and vectors and computing in parallel the distribution of the number of contacts such individuals have with other people. Wikis [12] and [13] give presentation files, exercises sets, and more details about the workshops.

3.6 Workshop Evaluation

Questionnaires for the workshops were more general than those for the linear algebra class. With a rating scale of 1 to 4 for Poor to Excellent, respectively, two questions for participants in the second workshop, "Computational Workshop for the Life Sciences: Bringing Computation to Life," seem most relevant to module evaluation: "The clarity of the information provided was:" and "The program materials were:" High mean scores and participants' comments about the workshop, particularly the applications, were gratifying (see Table 3).

Table 3. Means and standard deviations of $N = 18$ responses to rated questions, 1 (Poor), 2 (Adequate), 3 (Good), 4 (Excellent)

Question	Mean	Standard Deviation
The clarity of the information provided was:	3.56	0.62
The program materials were:	3.56	0.62

4. CONCLUSION

"Living Links: Applications of Matrix Operations to Population Studies" and its associated programs in MATLAB, Mathematica, and C/MPI are currently available on the UPEP Curriculum Modules website [2]. Class testing in High Performance Computing (HPC) of a preliminary version of the module helped in its development and showed the utility of the module in introducing matrices and some of their applications as a component of a HPC course. Class testing in Linear Algebra lead to refinement of the module and demonstrated its value in introducing matrix and vector operations with numerous applications to a mathematics class. Class testing the module as a base for workshop lectures, exercises, and projects illustrated its value as a resource for a faculty member. High questionnaire scores and enthusiastic comments from undergraduate level computer science, mathematics, and computational science students and graduate level science workshop participants verify the conclusion that "Living Links: Applications of Matrix Operations to Population Studies" can be an effective educational module in a variety of classes, levels, and settings.

5. ACKNOWLEDGEMENTS

We would like to acknowledge the generous help of many people and organizations. The National Computational Science Institute Undergraduate Petascale Education Program (UPEP), which is an NCSA Blue Waters project in collaboration with the National Computational Science Institute (NCSI) and national HPC programs, funded development of the module and supported UPEP intern Jesse Hanley, who implemented the module's HPC programs. NCSI under the direction of the Shodor Foundation with Executive Director Dr. Bob Panoff is hosting the module and associated materials on its website. The Monash e-Research Centre with Science Director Dr. David Abramson provided travel support for the authors during their stay in Australia and also organized and sponsored the workshops. Dr. Joe Spivey class tested the module in his Linear Algebra class. NCSA and NICS provided access to their Teragrid computers for the UPEP intern, the High Performance Computing class, and the first author. The authors worked with Ornella Cominetti at Oxford developing Chaste simulations that provide the foundations and data for four projects.

6. REFERENCES

[1] National Computational Science Institute Undergraduate Petascale Education Program (UPEP). http://computationalscience.org/upep Accessed 3/5/11.

[2] Shiflet, A. and Shiflet, G. 2011. "Living Links: Applications of Matrix Operations to Population Studies." National Computational Science Institute Undergraduate Petascale Education Program (UPEP) Curriculum Modules, UPEP Curriculum Modules site. http://shodor.org/petascale/materials/UPModules/populationMatrices/ Accessed 5/21/11.

[3] Wofford College. http://www.wofford.edu/ Accessed 3/5/11.

[4] Monash eScience and Grid Engineering Laboratory (MeSsAGE Lab) at Monash University in Melbourne, Australia. https://messagelab.monash.edu.au/ Accessed 3/5/11.

[5] Computational Science - Wofford College. http://www.wofford.edu/computationalscience/ Accessed 3/5/11.

[6] Taylor, Caz and Erin Grey. "Population Dynamics of Gulf Blue Crabs" 2010. Tulane University. http://leag.tulane.edu/PDFs/Grey-LEAG-4.28.10.pdf Accessed 10/13/10.

[7] NDSSL (Network Dynamics and Simulation Science Laboratory, Virginia Polytechnic Institute and State

University). 2009. "NDSSL Proto-Entities" http://ndssl.vbi.vt.edu/opendata/ Accessed 8/27/9.

[8] Shiflet, A. and Shiflet, G. 2010. "Getting the 'Edge' on the Next Flu Pandemic: We Should'a 'Node' Better." National Computational Science Institute Undergraduate Petascale Education Program (UPEP) Curriculum Modules, UPEP Curriculum Modules site. http://shodor.org/petascale/materials/UPModules/socialNetworks/ Accessed 5/21/11.

[9] Chaste, Cancer, Heart and Soft Tissue Environment. 2010. http://web.comlab.ox.ac.uk/chaste/ Accessed 10/14/10.

[10] Teragrid. 2010. https://www.teragrid.org/ Accessed 3/5/11.

[11] Shodor, a national resource for computational science education. 2011. http://www.shodor.org/ Accessed 3/5/11.

[12] Panoff, R., Shiflet, A. and Shiflet, G. "Introduction to Computational Thinking." 2011. https://messagelab.monash.edu.au/IntroductionToComputationalThinking/ Accessed 3/5/11.

[13] Shiflet, A. and Shiflet, G. "Computational Workshop for the Life Sciences: Bringing Computation to Life." 2011. https://messagelab.monash.edu.au/ComputationalThinkingForLifeSciences Accessed 3/5/11

Accelerating Geophysics Simulation using CUDA

Brandon Holt
University of Wisconsin–Eau Claire
105 Garfield Ave, Eau Claire, WI
holtbg@uwec.edu

Daniel Ernst
University of Wisconsin–Eau Claire
105 Garfield Ave, Eau Claire, WI
ernstdj@uwec.edu

ABSTRACT

CitcomS, a finite element code that models convection in the Earth's mantle, is used by many computational geophysicists to study the Earth's interior. In order to allow faster experiments and greater simulation capability, there is a push to increase the performance of the code to allow more computations to complete in the same amount of time. To accomplish this we leverage the massively parallel capabilities of graphics processors (GPUs), specifically those using NVIDIA's CUDA framework. We translated existing functions to run in parallel on the GPU, starting with the functions where the most computing time is spent. Running on NVIDIA Tesla GPUs, initial results show an average speedup of 1.8 that stays fairly constant with increasing problem sizes. Though many applications can see even orders of magnitude improvement from GPGPU acceleration, the potential improvement for CitcomS is limited by several factors, including being bound by MPI collective communication. With newer GPGPU frameworks such as Fermi, further performance improvements can be expected, though a more significant overhaul of the CitcomS code would be necessary for any significantly better speedup.

General Terms

Parallel programming, GPGPU, finite element

Keywords

CitcomS, Blue Waters Undergraduate Petascale Internship, CUDA

1. INTRODUCTION

Graphics processors are commodity hardware, found in nearly every modern personal computer, which are highly specialized to do all of the computations required to draw pixels on the screen. For intensive graphics applications such as high-resolution three-dimensional games, this means transforming, calculating lighting, and mapping and applying textures on millions of vertices and pixels. To do this, GPUs have banks of hundreds of small compute cores that process a stream of data together in parallel. By sacrificing some of their independence and flexibility, these cores save hugely on power consumption compared to more general-purpose CPUs. In addition, in order to keep all of those cores busy, GPUs have particularly high memory bandwidth.[8]

The high performance computing community has long been interested in using specialized *accelerators* to speed up certain parts of their computations, but these processors were often prohibitively expensive. In contrast, because of their commodity nature, GPUs are much cheaper. Instead of simply drawing graphics on the screen, we are interested in using these cards for more general purpose work. By using the massively parallel capabilities of GPUs for computation across large datasets, applications can see huge performance improvements. NVIDIA's CUDA parallel computing architecture is currently the most popular technology for general purpose programming of GPUs. Many high performance applications have seen order of magnitude speedups using CUDA, including NAMD (nanoscale molecular dynamics) with a 20x speedup, and Havok FX with 10x speedup for realtime physics simulation.[10]

In the field of geophysics, CitcomS is used by many to study convection problems in Earth's mantle. By accelerating the application we can enable simulations with finer detail or more time steps to complete in the same amount of time. At this time when single cores are not getting any faster, speedups come from parallelizing computation. CitcomS already uses Message Passing Interface (MPI) libraries to split up work across multiple compute nodes. However, the price of communicating boundary values limits the amount that the simulation can be split up in this way. To further parallelize the work done on each node, we use the massively parallel capabilities of graphics processors (GPUs). Using NVIDIA's CUDA programming model, we are able to parallelize each node's data-parallel computations using CUDA-capable GPUs if they are available. This report discusses the techniques used to accelerate CitcomS, describes

performance challenges and optimizations, and lists the results of our experiments showing speedup of the application as a whole.

2. BACKGROUND

2.1 CitcomS

CitcomS is a finite element code developed and supported by the Computational Infrastructure for Geodynamics (CIG). Written in C, it has support for MPI to allow it to be run in parallel on shared and distributed memory platforms. It is designed to solve compressible thermochemical convection problems, but it also support variable viscosity and so can be used to study the movement of plates. The model consists of a grid of points arranged in a spherical shell (either a full sphere or a restricted region). Starting with a set of initial values including temperature, pressure, and velocity at each point, it repeatedly solves the momentum and advection-diffusion equations, giving the new state of the system at each successive time step. An iterative relaxation scheme is used to solve the partial differential equations for velocity and pressure across the grid domain, using either a conjugate gradient or multi-grid solver to find solutions for the equations which are represented as discrete matrices.[12]

As a typical finite element code, with its regular grid and high spatial locality, CitcomS is characterized as a *structured grid* problem as defined by Asanovic and colleagues in their report. Because CitcomS does not do adaptive mesh refinement, this means that it should be relatively simple to parallelize by simply breaking up the grid onto different nodes and sharing updated boundary values between iterations.[3] In fact, this is precisely what is done using MPI already, and because each point only relies on its immediate neighbors in the grid, calculations on chunks distributed to each node should be able to be parallelized even further.

2.2 CUDA Programming Model

While CUDA aims to make GPUs programmable like CPUs, the mindset is quite different. Instead of one primary thread (or a minimal number of software threads) on CPUs, the idea behind the CUDA programming model is to have hundreds or thousands of hardware threads running the same *kernel* function on different pieces of a large data set. In general, each GPU thread individually runs slower than on the CPU, but together, hundreds of threads have a much higher throughput. Because of the less flexible nature of GPU cores, these threads also have extremely limited branching options: no function calls are allowed (except inline) within a kernel.

In the CUDA model, there is a reference hierarchy to threads with different localities: a *block* of threads all run on the same *streaming multiprocessor (SM)* which has a number of compute cores, a bank of registers local to each thread, and a fast *shared memory* cache. Thread blocks are organized in a *grid*, all running the same kernel and sharing the *global memory* bank on the GPU. Only threads within a block can be synchronized using primitive barrier calls. Because global synchronization is impossible within a kernel, the algorithm must have work that can be completely de-coupled.

Porting existing CPU code to CUDA primarily involves finding which parts do a lot of computation on a large set of data in a serial loop and splitting up that work onto many threads. However, to get optimum performance, the programmer must know the intricate details of the hardware and how they affect performance. Optimization practices include: managing the severely limited shared memory cache, coalescing global memory accesses, minimizing in-warp divergence, avoiding bank conflicts, and maximizing thread occupancy.[9]

Table 1: Profiling Results
(Conjugate Gradient Solver)

% Time	Function
78.82	`n_assemble_del2_u`
4.70	`conj_grad`
4.06	`global_vdot`
3.49	`assemble_div_u`
2.82	`get_elt_k`
2.12	`assemble_grad_p`
1.17	`regional_exchange_id_d`
...	...

3. IMPLEMENTATION

We used the current release of CitcomS, version 3.1.1, as the basis of our CUDA accelerated version.

3.1 Profiling

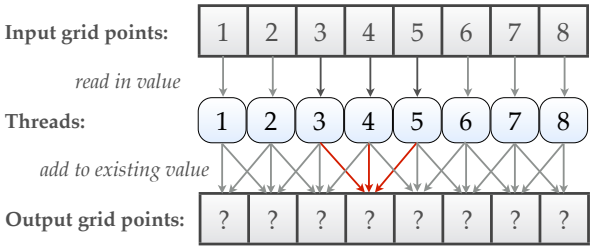
Translating the entire application to CUDA would be neither practical nor desirable. CitcomS consists of a large codebase, with many different functions for calculating each parameter of the simulation. Many of these operations, particularly input and output tasks, are simply not conducive to massively parallel execution. Therefore, in order to maximize our impact on the overall speed of the code, we carefully profiled it to see where in the code most of the time was being spent.

Using GNU gprof[4], we ran a series of simulations with different grid sizes and input parameters. Shown in Table 1 are results for a typical run using the conjugate gradient solver. Actual numbers for each function fluctuated slightly between different simulations, but the trend was consistent. It was clear that the function `n_assemble_del2_u`, at 78% of the overall time, was where we should focus our efforts to improve the conjugate gradient solver. According to Amdahl's Law, the theoretical speedup of an algorithm from parallelization is a function of the speedup of the parallelized part (S) and the proportion of the computation that this part represents (P):

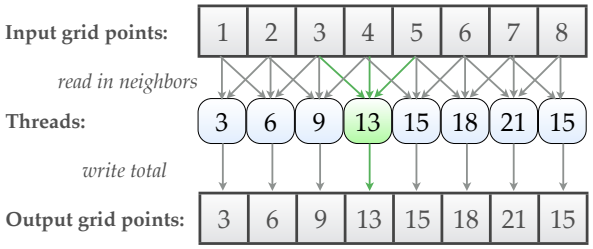
$$\frac{1}{(1-P)+\frac{P}{S}}$$

Speedup here simply refers to the ratio of serial execution time to the parallelized execution time ($S = T_{serial}/T_{parallel}$). Speeding up only `n_assemble_del2_u` with $P = 79\%$, assuming perfect speedup ($S = \infty$), the maximum theoretical speedup overall would be 4.72.

Further inspection of the code showed that this function is used to calculate the Laplacian on the discretized temperature, pressure, and velocity matrices, which correspond to the size of the spherical grid. As we expected for this structured grid code, this matrix computation corresponds



(a) In the original algorithm, each point added its own contribution to each of its neighbors' new values. Concurrently writing to the same spot in memory from multiple threads like this leads to nondeterministic behavior because the reads and writes could happen in any order.



(b) To eliminate conflicting read/writes, the algorithm was restructured so that each thread calculates its own new value by computing what each of its neighbors would have contributed based on their current values and storing just its own value.

Figure 1: Reordering Reads and Writes

to an operation being executed across all points in the spherical grid, marking this as a good candidate for GPU execution and making this our primary target for translation to CUDA.

3.2 Translation

The first part of translating `n_assemble_del2_u` was deciding how to break up the work onto GPU threads. For CUDA, using as many threads as possible is often the best option. Each streaming multiprocessor can execute several blocks of threads. If a large number of threads in the same block make memory requests, they can queue up waiting for their data while other threads are executed. Also, if these threads access memory local relative to each other, their accesses can be coalesced into a single global load. Therefore, the more threads the CUDA scheduler has available in each thread block, the more opportunities it has to hide memory latencies. On the other hand, once each thread gets its memory, it ought to have a significant amount of work to do with it, otherwise all the time used getting the memory there was wasted.

Taking those factors into consideration, we decided to make each thread handle a single grid point. This is justified because the calculation of each point requires aggregating values from several matrices for each of its neighbors. In the original CPU code for `n_assemble_del2_u`, there was a primary loop that iterated over each point in the grid, so we simply refactored it into a CUDA kernel, using the thread and block IDs in place of the loop variable.

However, running iterations of a loop concurrently introduces problems that did not exist when run serially. Threads running on the GPU have extremely limited synchronization options. CUDA cards with compute capability of at least 2.0 introduced some atomic store operations but at a significant performance hit from serialization. Without synchronizing, it is unsafe for concurrent threads to write to the same spot in memory, so kernel functions should be designed such that each thread will have "ownership" of a subset of memory that only it stores results into. In the case of our kernel, each thread was straightforwardly responsible for calculating its own grid point's future state. However, each iteration of the original loop had one loop over its neighbors where it incremented each neighbor's values based on its own value. This would have led to multiple threads concurrently incrementing the same spots in memory, resulting in nondeterministic behavior. Fortunately it proved possible to reverse the second loop so that each thread pulled together all the values that were previously being incremented, resulting in a working, deterministic CUDA translation, calculation for calculation. See Figure 1 for an illustration of this.

3.3 Verification

Because our goal is a functionally equivalent refactoring, it was essential that we verified our function's output with the original at every step of development. This was accomplished by simply executing both versions each time `n_assemble_del2_u` was called. A single array constitutes the output for the function, so both original and CUDA output were stored, with the following condition used to flag an error:

$$\frac{|x_i - y_i|}{y_i} > 5.0 \times 10^{-15}$$

where y_i is an element in the original function result and x_i is the corresponding CUDA result. Even though it is expected that all of the same calculations will be done, some small floating point precision error is expected due to some the reordering of calculations and minor differences in how GPU cores implement floating point operations, but those errors should be relatively insignificant.

3.4 Optimization

As mentioned before, the CUDA architecture can provide massive amounts of parallel computation with particularly high memory bandwidth. However, achieving that maximum performance is not always possible, and approaching that limit is what the majority of the novel work for this project was devoted to. Here we will step through several major iterations of the CUDA kernel, highlighting the most relevant performance aspects of each.

The CUDA profiler is a very simple tool which comes bundled with the CUDA framework, but it was particularly useful for diagnosing each of the bottlenecks described below. The profiler is very simple and supports the output of only four variables from a list of over 40 options. On systems with CUDA installed, profiling can be enabled by setting an environment flag (`CUDA_PROFILE=1`) before executing the CUDA program. Values of each variable for every kernel invocation is sent to a file (`./cuda_profile0.log` by default).[7] Below we will highlight profiling options we used to obtain information we needed about our kernel's performance.

Table 2: Kernel Optimizations
Speedup shown is for the ratio of the original CPU function's execution time to the time spent executing the `n_assemble_del2_u` *kernel itself.*

	Time (ms)	Speedup
Original:	2.00	1.00
Kernel 1:	880	0.0023
Kernel 2:	21.78	0.09
Kernel 3:	0.44	4.55
Kernel 4:	0.29	6.90

Kernel 1: The first working implementation of the kernel was 400 times slower than the original CPU version. On the CPU there are several layers of caches that make sure that successive reads and writes to a chunk of memory are as fast as possible, which are automatically handled in hardware on most systems. Because access patterns are potentially much more complicated to predict for thousands of threads, automatic caching is only available for the newest CUDA architectures. Therefore, in this first kernel, when each thread accesses its nearest neighbors' values, it must go all the way to global GPU memory, an operation that takes on the order of milliseconds. The GPU memory manager can *coalesce* memory accesses, that is load multiple threads' values all together, if the accesses are sequential and regular. However, because each thread is accessing each of its neighbors' values, which are not contiguous, these accesses are not able to be coalesced, even on newer architectures where caching is handled automatically. The CUDA Profiler options `gld_incoherent`, `gld_coherent`, `gst_incoherent` and `gst_coherent` show the number of uncoalesced (incoherent) loads and stores compared to coalesced ones. Ideally, there should be no incoherent loads or stores, which was achieved in subsequent kernels.

Kernel 2: Luckily there are closer and faster memory banks available on each streaming multiprocessor that can be shared among threads in a block. Moving the most heavily accessed data in our kernel into this *shared memory* can easily be done in a coalesced fashion, and once in this cache, accesses to the data are comparable to working with registers. In our kernel, the most heavily used array, which we determined by counting the number of reads from it in our code, is `Node_map`, which holds a map of each grid point to indices for all of its neighboring points. By explicitly loading all of the maps for each thread in a block into shared memory at the beginning of the kernel, the global accesses are regular and sequential, allowing the loads to be coalesced, and all subsequent accesses are essentially free. We were able to observe this on the CUDA cards with compute capability of 1.3 by observing a significant drop in the number of global loads (`gld_incoherent` and `gld_coherent`) from millions of loads to 40,000-80,000 per invocation (varying depending on the amount of branching, actually). Note: newer cards with compute capability greater than 2.0 have the option of outputting shared memory loads and stores directly.[7] Just doing this sped up the kernel by 40 times. Clearly even minor changes to CUDA kernels can either severely damage performance or greatly improve it.

Kernel 3: Using an array of indices to map into another array is a common method of saving computation time on the CPU because it allows for all the neighbors to be calculated once and then simply looked up every subsequent time they are needed. This made sense on the CPU where there was a deep cache and only one compute unit. On the GPU, however, it often makes sense to recompute some values if it would take less time than waiting on extremely slow un-cached memory accesses. Once an (x, y, z) location in the grid is calculated from the thread index, three levels of nested loops can iterate over all the neighbors by simply oscillating on each side of each dimension. This was observed again as a drastic decrease in the number of global loads in our profiling results. Eliminating `Node_map` in this way improved the performance of the kernel nearly another 50 times.

Kernel 4: With `Node_map` no longer taking up space in shared memory, other data was able to take its place. However, the Tesla generation of CUDA cards have only 16 kilobytes of shared memory per multiprocessor.[8] Even just the primary input array will not fit completely in shared memory for typical grid sizes. Because we have such limited synchronization ability, we must ensure that all values of the array that might need to be accessed by a thread in a block are pre-loaded into shared memory. By loading in a 3-dimensional tile of points surrounding the points referenced by the current block of threads, we are able to maximize the amount of caching we can do.

Working off of the final kernel design, there were a number of minor changes that we tried to maximize our usage of all of the GPU's resources. Each multiprocessor can have up to eight resident blocks, which allows it to hide memory latencies by scheduling warps from other blocks to run while some are waiting. This *occupancy* is determined by the amount of registers allocated per thread, the number of threads, and the amount of shared memory that each block uses. Using the verbose `ptxas` compiler option, the programmer is able to see these parameters, and the CUDA profiler displays the precise occupancy of each kernel invocation as well. Using the CUDA Occupancy Calculator, a spreadsheet available from NVIDIA's website, developers can put in the various parameters for their run, such as shared memory and register usage and it will show what the limiting factor for occupancy is.[6] Using this and the output of the CUDA profiler, it was clear to us that the major limiting factor was the number of registers being used by each thread, which we were unable to minimize. Instead, we adjusted the number of threads per block, observing the performance of each kernel and settling on the best configuration. We found that 192 threads per block best balanced the tradeoff between using too many registers and losing shared memory benefits by having thread blocks that are too small. However, these results would not be optimal for other CUDA architectures. It would have to be retested and optimized for each architecture to achieve maximum performance.

At this point, even with that primary array taking up all available space in shared memory, there are still several more global arrays that are accessed regularly during the computation. Several large floating point arrays that represent the stiffness matrix are much too large to for us to cache. As a result, our threads are constantly waiting on slow global memory loads instead of making full use of the GPU com-

pute units. There are a number of other performance issues that affect kernels at the instruction level, such as branch divergence when threads executing in lock-step on an SM take different paths and must be serialized, and bank conflicts when shared memory accesses are strided incorrectly. However, examining and eliminating these kinds of issues is only useful when the majority of the time is already being spent executing instructions. Because we are bound by memory accesses, not by compute resources, there is little more that can be done to further optimize the kernel.[5]

3.5 Automatic Integration

Not all users will have NVIDIA GPUs available, so we provide users of CitcomS with the option to use the CUDA-accelerated version or not. `Autoconf` and `automake` are two tools already being used in the code to automatically find libraries and configure the build. A simple addition to the configure script adds an option to build with CUDA support. However, we also did not want to require that everyone using CitcomS on a particular cluster would be required to use the CUDA version, nor did we want to force two versions to be built for such a small difference in code. When built with CUDA support, our `use_cuda` option is added to the configuration file that is used to specify each simulation, which can be used to select the CUDA-accelerated version at runtime with little to no overhead.

4. RESULTS

In the section on Optimization and in Table 2, our speedups only referred to the computation of the `n_assemble_del2_u` function. To see how these results translated into overall system acceleration, we ran a series of simulations and measured their runtimes. Each simulation was run twice per trial, once using the original CPU-only version and once with the CUDA version. Shown in Figure 2 are average times for three runs of a typical simulation with varying grid sizes and number of MPI processes, run using Tesla-generation cards that were available on Earlham College's Al-Salam cluster[1] and NCSA's Lincoln cluster[2]. Our timings for each setup had standard deviations of less than 5% of the average times after 3 trials, with most times varying less than 2.0 seconds between trials. Because of this consistency, the three trials should be accurate enough to make judgements about. Similar results were observed for several other simulations using different physical parameters (varied initial conditions and viscosity parameters), which was expected because this function is an integral part of both solvers, so its usage pattern is highly regular.

Seen in Figure 2a, as the problem size was increased, better speedups were observed. This is because as we increase the number of grid points, we were able to better take advantage of the massive parallelism provided by the GPU. However, the speedup tapered off around 1.8x, which we hypothesize is because the amount of memory that needs to be transferred to and from the GPU increases as well. From Figure 2b, it can be seen that the speedups we observe in the one-node case do not scale perfectly with increasing number of MPI processes. However, larger problem sizes still showed greater speedups. This is to be expected because for the same problem size, increasing the number of MPI processes gives each CUDA kernel smaller pieces of the grid to work with, which was just shown in Figure 2a to make speedup worse. Therefore overall, the best performance is still to be had with larger grids because it will give the best CUDA performance and allow more divisions into MPI processes.

Based on the actual speedup of `n_assemble_del2_u`, we can revisit Amdahl's Law, this time with $S = 6.9$ and $P = 78.82\%$ still. This results in a predicted overall speedup of 3.07. This is obviously less than the theoretical best we computed of 4.72 which is to be expected because perfect speedup is obviously impossible. However, it is significantly better than the 1.8x we actually achieved. This can be partly explained by the time needed to allocate and copy all of the memory onto the GPU, which was not included in the kernel timings because the memory copies are done in several different places in the code. Despite the high bandwidth to the GPU, the entire set of data used within the function must be transferred both ways every time it is called.

5. FUTURE WORK

As the CUDA-accelerated version of CitcomS stands right now, one function, `n_assemble_del2_u`, has been successfully translated and optimized for NVIDIA Tesla GPUs.

5.1 Multi-grid Solver

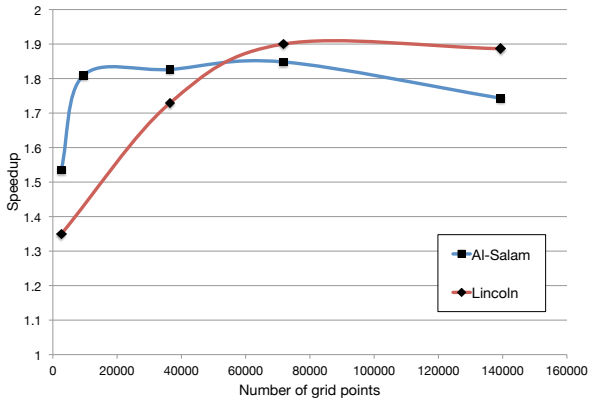
As previously stated, `n_assemble_del2_u` played the greatest role in the conjugate gradient solver. While it also was used in the multi-grid solver, another function, `gauss_seidel`, was the most time-intensive function there. We spent some time attempting to translate this function as well so that both solvers could be significantly accelerated. However, it proved significantly more difficult than the previous translation. For `n_assemble_del2_u`, all of the calculations simply read from the original array and stored their new values in the output array. The Gauss-Seidel method is an iterative relaxation method for solving a linear set of equations. It is based on the simpler Jacobi method which averages its nearest neighbors's current values to compute its own new value. In order to converge in fewer iterations, Gauss-Seidel makes use of any new values that are available in calculating each new value.[11] This is perfectly acceptable in a serially executed loop, but when loop iterations are run concurrently, this causes problems. This meant that we could not do a functionally equivalent translation of `gauss_seidel`. We made several attempts to write a Jacobi function that would operate in the same way as `gauss_seidel` but simply required more iterations to converge. However, everything we tried caused issues when run as part of the multi-grid solver, causing the solutions to never converge or to end up at infinity. At this time, the code has been left out of the production version of our CUDA-accelerated CitcomS. Perhaps with more experience with numerical methods we would be able to find the places where our Jacobi method failed to duplicate what the Gauss-Seidel function was doing. It might be that future work could be done to re-implement the multi-grid solver from the ground up in CUDA.

5.2 MPI Communication Barriers

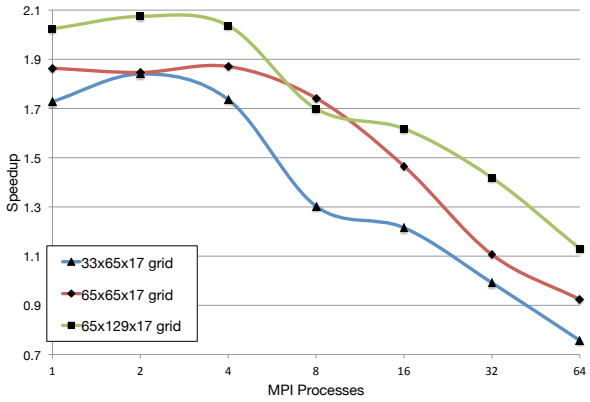
Beyond the Gauss-Seidel/Jacobi issues, the most obvious course for future work would be to continue translating more of CitcomS's code to CUDA. However, there is a limit to the amount that patchwork translation such as this can be used to improve overall performance. The existing MPI commu-

Figure 2: Speedup Results
Timing results used for speedup calculations are each averages of at least 3 separate runs.

(a) Scaling Problem Size



(b) Scaling MPI Processes



Grid Size	Orig. (s)	CUDA (s)	Speedup
2601	26.79	19.856	1.3492
36465	745.68	431.23	1.7292
71825	1609.46	847.04	1.9001
139425	5116.58	2712.16	1.8865

MPI	33x65x17	65x65x17	65x129x17
1	1.728	1.864	2.024
2	1.840	1.846	2.075
4	1.736	1.871	2.037
8	1.302	1.742	1.699
16	1.215	1.466	1.618
32	0.992	1.106	1.418
64	0.757	0.924	1.130

nication pattern in CitcomS frequently does collective operations sharing updated boundary values among all of the processes to keep the individual pieces of the grid synchronized. These MPI calls are an absolute barrier to what can be computed uninterrupted on the GPU. Prior to communication, the kernel must complete and the values must be copied back to main memory from the GPU's memory. Once communication is complete, the data can be copied back to the GPU and the kernel resumed. With these hard barriers in place, even if all data-parallel computations were done by the GPU, the limiting factor would be all the memory movement. A more complete rewrite of the CitcomS codebase might be able to minimize the amount of communication needed and perhaps coalesce it into a single update per step. However, with current cluster architectures where all inter-node communication goes through CPU nodes, the communication will likely still be a major limiting factor.

5.3 Other Directions

Future work on accelerating CitcomS could go several directions. If a version that will work across different vendors is desired, an OpenCL version of the current translation should be straightforward because the basic kernel model remains largely unchanged between the two. Along the same vein, the current version has certain parameters, such as the number of threads per block, hand-optimized and hard-coded into the source. Running optimally on even the newer Fermi generation of GPUs, which have larger shared memory caches among many other improvements, would require adjusting these parameters based on experimental results. For other GPU architectures similar adjustments would need to be made. Perhaps some future work could be done to automate this optimization task, for CitcomS or for CUDA/GPGPU applications in a more general sense. Because GPU acceleration is a highly popular area of research right now, it is likely that many new tools will soon be available to potentially be applied to CitcomS.

6. REFLECTIONS

This project was not only a scientific venture attempting to enhance geophysicists' tools. As part of the Undergraduate Petascale Internship Program, the goal is also to enhance undergraduate education, so here I will reflect on my own experience and how it can be duplicated for other undergraduates.

Working on this project has given me a great deal of experience working on a number of different high performance clusters. The two-week workshop at the beginning of the summer kickstarted my work, but through the process of studying, translating, and optimizing, I have gotten much more comfortable with it all. I have become adept at debugging all kinds of issues with building, installing, and running all manners of programs. In working with CUDA for the last year, I have come to understand the architecture in great depth, and I have a feeling for a wide variety of programming problems and performance issues. Because of my experience with CUDA and connections through the UPEP instructors, I have gotten the opportunity to help as an assistant instructor at an intermediate parallel programming workshop, as well as at this year's Blue Waters UPEP Institute. These teaching experiences have further reinforced my understanding of all of the parallel programming techniques that are taught there. I think getting undergraduates to

spend a summer digging down into real scientific code and get their hands dirty writing real high performance code will give them experience that will be invaluable to them in pursuing a research career later.

While working at the extremely low-level of CUDA optimization has helped me understand the architecture and programming model quite well, I have recognized that the amount of time it took me to get better at this is simply not efficient for the majority of programmers to do. In the coming age of computing, it is likely that, in order to continue to scale in performance without excessive power expenditure, computing will become increasingly heterogeneous, with specialized processors such as GPUs playing an important role. It is already infeasible to expect every programmer to become an expert in all of the different kinds of accelerator hardware that are available now. My work hand-tooling CUDA code and optimizing it by experimentation and often guesswork has made it obvious to me that new programming models need to be explored.

I am interested in finding out how to separate the mechanical processes, such as finding the optimal number of threads per block or managing limited shared memory space, from the creative ones, such as thinking of the best way to parallelize the algorithm. The mechanical tasks could be accomplished in many different ways, either by the compiler at build time, by a runtime system, or one of many other techniques. Another interesting area of research would be to better facilitate the creative part so that programmers can effectively represent their ideas in a way that compilers and the rest can take advantage of the available compute resources.

In the interest of pursuing these research directions, I will be attending graduate school at the University of Washington. My planned research goal stems from these issues with programming in CUDA: I am interested in applying aspects of programming languages, compilers, runtime systems, and software engineering tools to assist programmers in developing applications for the heterogeneous parallel computers, smart phones, or other ubiquitous computing devices that will need software in the future.

7. ACKNOWLEDGEMENTS

This research was conducted through the Blue Waters Undergraduate Petascale Education Program (BW-UPEP). In an effort to broaden and diversify the high performance community, this program funds a number of undergraduate students to do computational science research with a faculty advisor for a summer, as well as a stipend to continue their research during the school year. To prepare these students for their projects, they attend a two week workshop on parallel programming, scientific computing, and high performance architectures. Thanks to Shodor and the National Computational Science Institute (NCSI) for their financial support of this internship.

This research was supported in part by the National Science Foundation through TeraGrid resources provided by the National Center for Supercomputing Applications (NCSA) under grant number TG-CCR100014.

Thanks also to Charlie Peck from the Computer Science Department at Earlham College for the use of their Al-Salam cluster. Thanks also to the University of Wisconsin–Eau Claire Chemistry Department and Professor Christine Morales for use of their cluster, EB-Wilson.

8. REFERENCES

[1] Earlham College Cluster Computing Group. `http://cluster.earlham.edu/`.

[2] NCSA Scientific Computing: Intel 64 Tesla Linux Cluster Lincoln. `http://www.ncsa.illinois.edu/UserInfo/Resources/Hardware/Intel64TeslaCluster/`.

[3] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. The landscape of parallel computing research: A view from Berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Dec 2006.

[4] S. L. Graham, P. B. Kessler, and M. K. Mckusick. Gprof: A call graph execution profiler. *SIGPLAN Not.*, 17:120–126, June 1982.

[5] D. B. Kirk and W. W. Hwu. *Programming Massively Parallel Processors*. Morgan Kaufman Publishers, 2010.

[6] NVIDIA. Cuda occupancy calculator. `http://developer.download.nvidia.com/compute/cuda/CUDA_Occupancy_calculator.xls`.

[7] NVIDIA. *CUDA Profiler README, Version 3.0*. NVIDIA, 2009.

[8] NVIDIA. Nvidia's next generation cuda compute architecture: Fermi, white paper. Technical report, NVIDIA Corporation, 2009.

[9] NVIDIA. *CUDA C Programming Guide, Version 4.0*. NVIDIA, March 2011.

[10] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[11] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988.

[12] E. Tan, M. Gurnis, L. Armendariz, L. Strand, and S. Kientz. *CitcomS: User Manual*. Computational Infrastructure for Geodynamics (CIG), 3.1.1 edition, July 2009. `http://geodynamics.org/cig/software/citcoms`.

Understanding the Structural and Functional Effects of Mutations in HIV-1 Protease Mutants Using 100ns Molecular Dynamics Simulations

Blue Waters Undergraduate Petascale Education Program

Christopher D. Savoie
University of New Orleans
2000 Lakeshore Drive
New Orleans, LA 70148
csavoie@uno.edu

David L. Mobley
University of New Orleans
2000 Lakeshore Drive
New Orleans, LA 70148
dlmobley@uno.edu

ABSTRACT

The Human Immunodeficiency Virus type 1 protease (HIV-1 PR) performs a vital role in the lifecycle of the virus, specifically in the maturation of new viral particles. Therefore, delaying the onset of AIDS, the primary goal of HIV treatment, can be achieved by inhibiting this protease[2]. However, the rapidly mutating virus quickly develops drug resistance to current inhibitors, thus novel protease inhibitors are needed.

Here, 100ns molecular dynamics (MD) simulations were conducted for the wild type and two mutant proteases to gain insight into the mechanisms by which the mutations confer drug resistance. Several different metrics were used to search for differences between the wild type and mutant proteases including: flap tip distance and root-mean-square deviation (RMSD), mutual information, and Kullback-Leibler divergence. It was found that at the 100ns timescale there were no large differences in the structure, flexibility and motions of the wild type protease relative to the mutants, and longer simulations may be needed to identify how the structural changes imparted by the mutations affect the protease's functionality.

General Terms

Measurement, Experimentation, Design

Keywords

Blue Water Undergraduate Petascale Internship, HIV-1 Protease, Parallel Computing, Molecular Dynamics Simulation

1. INTRODUCTION

Currently, there is a considerable effort being put into the development of novel drugs to combat HIV by many researchers around the world. Part of that work focuses on designing drugs that inhibit the homodimeric, aspartyl HIV-1 PR. This enzyme, one of only a handful of targets for HIV treatment, serves a key role in the lifecycle of HIV by processing gag-pol polyproteins into discrete, functional proteins. Inhibiting the protease prevents it from performing proteolysis, resulting in immature and noninfectious virus particles[2]. Therefore, inhibition of the protease can delay the onset of AIDS; however, mutations rapidly develop in it from one viral generation to the next in response to antiviral treatment[4]. These mutations often lead to the formation of antiviral drug resistance.

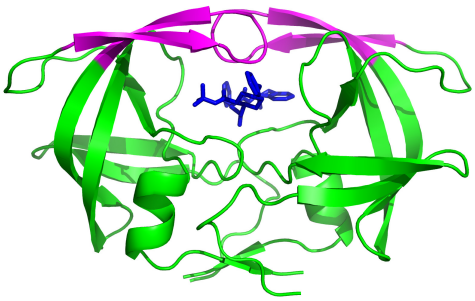


Figure 1: A cartoon representation of the 198 amino acid wild type HIV-1 PR bound to indinavir. The flap tips are colored magenta.

By using MD simulations to gain insight into the mechanism by which drug resistance occurs, we hope to aid in the development of adaptive inhibitors which show high activity towards both wild type and mutant proteases. Because a protease inhibitor could be designed to take advantage of conserved regions vital for the protease's function, there is hope adaptive inhibitors can be developed.

After being selected to take part in the Blue Waters Undergraduate Petascale Education Program (BW-UPEP), I,

Christopher Savoie, was given the opportunity to participate in this research project. The BW-UPEP began with a two week intensive training workshop held at the National Center for Supercomputing Applications on the University of Illinois at Urbana Champaign campus. The workshop was both a hands on learning experience and a survey of high performance computing and parallel programing. It provided an introduction to using MPI for running parallel programs along with an overview of how to use programs such as VMD and biomolecular simulation software, among other things. Following the workshop I returned the University of New Orleans where this research project was performed under the guidance of my mentor, Dr. David Mobley.

2. RESEARCH PLAN

2.1 Goals and Milestones

Our chief goal is to understand how mutations in the protease's amino acid sequence adversely affect ligand binding. By looking at inhibitors' binding affinities relative to the wild type and the mutants, we commonly see at least an order of magnitude decrease in the mutant's binding affinity, resulting from a single amino acid substitution[15, 11]. To understand why this occurs, we planned to setup, run, and analyze MD simulations of known HIV-1 PR variants with several ligands. Experimentally, each protease and ligand pair we chose shows varying degrees of affinity loss when mutations are introduced in the wild type[11].

To set up the simulations, we began with protein structures taken from the Protein Data Bank (PDB)[3] with mutations introduced as needed using PyMOL's mutagenesis tools [14]. After the models were built, the system was benchmarked to determine an efficient number of nodes to run it on. That is, we wanted to know how many nodes would give us the best value in terms of the wall clock time and the computational expense associated with that wall clock time. Because eventually the additional reduction in wall clock time from using an additional node becomes relatively small, adding more computing resources to the simulation becomes counterproductive at some point.

To accomplish our main goal, we planned to analyze the simulations' trajectories, using several methods to gauge how the mutations alter the function of the wild type protease. First, because we hypothesized that the relatively flexible flap tip region of the protease might show differences across HIV-1 PR variants, we chose to monitor the flap tip distance and RMSD. This flexible flap tip region of the protease is vital for a functional protease as well as ligand binding. As a result, changes in the interactions between the flap tips along with changes in their mobility and flexibility are important in the develop of novel inhibitors.

While hydrophobic interactions are primarily responsible for stabilizing a protein in solution, hydrogen bonds play an important role in fine tuning the structure. Because of this, we also wanted to explore the hydrogen bonding between the flap tips to see what effect mutations have there. Finally, we wanted to study correlated motions in the protease using mutual information, and we planned to see how mutations alter the torsional degrees of freedom at side chain dihedrals by comparing the distributions of angles from the wild type and mutant using Kullback-Leibler (KL) divergence.

Because of the amount of work involved and to provide a measure of progress, we formulated several milestones for the project with the majority of the work taking place during the summer. The four main milestones were as follows: (1) set up the various protease/ligand combinations and scripts for running the simulations, then submit the jobs for the first set of simulations to the queue; (2) write Python scripts utilizing GROMACS tools to perform a structural analysis by making measurements of flap tip hydrogen bonding, distance, and RMSD; (3) while running the second set of simulations, begin structural studies of the first by reviewing the initial analysis results and by producing plots of the data; and (4) perform additional structural studies using KL divergence and mutual information to look for patterns of change introduced by the drug resistance mutations.

2.2 Simulation Setup

Simulations were performed on the wild type HIV-1 PR and two variants. The first variant of HIV-1 PR contains the mutation I50V which is located in the protease's flap tips. The second variant has mutations V82F and I84V which are located towards the catalytic dyad in the center of the protein. Moreover, lopinavir and indinavir were simulated in the presence of the wild type and I50V proteases while ritonavir was simulated with the wild type and I82F/I84V protease. Experimentally, these mutants dramatically reduce inhibitor binding affinities[11]. For example, the V82F/I84V HIV-1 PR mutant nearly shows a 400 fold decrease in affinity for ritonavir and the I50V HIV-1 PR mutant shows a roughly 600 fold decrease in affinity for lopinavir[11].

The wild type protease's crystallographic structure was obtained from the PDB file 2BPX. From this structure, all attached water molecules were removed. Next, using PyMOL's Mutagenesis Wizard, the above mentioned mutations were inserted and the two resulting structures stored along with the wild type's. Following this, the three protein structures were protonated using Muti-Conformation Continuum Electrostatics (MCCE)[13, 1]. Following protonation, it was confirmed that both catalytic, aspartyl residues (25D and 25D') retained a neutral protonation state. It has been suggested that correct protonation state prediction at these residues is necessary for accurately calculating binding free energies[18].

As a starting point for simulating the bound protease, the binding mode for indinavir, lopinavir, and ritonavir was derived respectively from the PDB files: 2BPX, 1MUI, and 1HXW. Next, AM1-BCC partial charges were assigned for each ligand. Then a Generalized AMBER Force Field (GAFF) was used for ligand paramaters[16, 17] and ffamber99sb force field parameters for the protein[6]. After the force fields for the ligand and protein were selected and applied to the system, a simulation box was set up. A dodecahedral shaped box with edges 1.2nm from the solute was used and applied to the models using the GROMACS tool editconf. Finally, bulk TIP3P water was inserted to fill the simulation box using the GROMACS program genbox.

Finally, using GROMACS 4.0.7 in conjunction with MPI, we performed energy minimization, equilibration and dynamics on our system[5]. A steepest decent minimization of 500 steps was performed. Next, constant pressure and constant

volume equilibration was carried out on the system. While this was occurring, the system's temperature was raised to 300K and the pressure was adjusted to 1atm. Equilibration steps were carried out for a total of 1100ps. Finally, during the production stage, we used a 2fs timestep for our simulations which were ran to 100ns, and we recorded data every picosecond. Ultimately, two duplicate simulations were ran for each protein and ligand pair.

3. CHALLENGES AND OPPORTUNITIES

As a Chemistry major, I do not have an extensive background in computer programming, having had only one formal programming course in C++. In addition, I had little experience with scientific computing environments such as Linux. To be successful in reaching the set milestones, I had to first become familiar with Linux, learning how to navigate the new environment. In addition, I had to learn Python scripting skills to run most of the analysis tools on the simulation data. At times, some Perl and R were necessary, and a level of understanding had to be achieved to move forward.

Human error guarantees that mistakes will be made when scripting, therefore skills must be developed to find and correct bugs to again move forward. Troubleshooting is not only limited to scripting errors; and occasionally, this project required looking for errors in environmental variables to ensure a script had access to the required modules.

An additional challenge was learning how to deal with such a large problem. One way this was done was using a reductionist approach. Large problems, such as the central question of this project, can't be solved as a single problem. Instead, it was broken down into many smaller problems. Learning how to break large assignments like this down to smaller pieces that could be handled day-to-day was key to success.

Overall, reaching this project's main goal was a challenge, requiring the development of new skills ranging from scripting to troubleshooting. However, it clearly provided many opportunities to advance my skills as a researcher. That is, it fostered my ability reduce large problems down to ones of workable size and taught me how to work with others in a research setting.

4. ANALYSIS AND RESULTS

4.1 Flap Tip Distance, RMSD and Hydrogen Bonding

Searching for clues to give us insight into the mechanism by which drug resistance arises, we first investigated several structural and chemical features of the flap tips. A list of the tools used in this analysis and the properties they measure are given in Table 1. The flap tips are relatively dynamic compared to the rest of the protease, moving between closed, semi-open, and open states when binding ligands. Thus, their distance of separation may be important for gauging their activity towards a given inhibitor. The flap tip distance was defined as the distance between the centers of mass of residues 50 and 50'. There were not any large differences in the flap tip distance between mutant and wild type pairs in many cases as seen in Figures 2-3. And, there was not a consistent pattern of the mutant or wild type proteases displaying a larger distance of separation between the flap tips on average. Moreover, while the flap tip distance in the mutant in Figure 3 does appear to have more mobility, this is not consistently seen in other simulations.

Tool	Property
g_dist	Flap tip distance
g_rms	Flap tip RMSD
g_hbond	Number of hydrogen bonds
MutInf	Mutual information and KL divergence

Table 1: Analysis tools used and properties measured

For each protein and ligand pair that was simulated, the RMSD of the flap tips was determined to give us more insight into the behavior of the flap tips. Included in the RMSD measurement were residues numbered 46-55 and 46'-55'. As seen in Figures 4-5 for one particular wild type and mutant set, there is not a large large difference in the ending value of the RMSD. That is, the flap tips in both the mutant and wild type end up displaying nearly the same about of flexibility. While the flap tips do show greater flexibility in some of the mutant cases, this is offset by an equal number of wild type cases showing a higher RMSD.

Another possible effect of mutations on the protease was hypothesized to be a decrease in the interflap hydrogen bonding. Therefore, the number of hydrogen bonds between the flap tips over time was also used as a metric for identifying differences between the wild type proteases and the mutants. Residues 48-53 and 48'-53' were included in the calculation; other residues in the flap are too far away from the opposing flap to form hydrogen bonds. Thus, they were not included in the calculation. Overall, the results do not show either the mutants or wild type proteases having more hydrogen bonds between flap tips on average at this timescale.

While we did not see a difference in the overall amount of hydrogen bonding between the flap tips in the protease and wild type proteases, we wanted to know if the mutations possibly interfered or changed the network of hydrogen bonds between the flap tips. Moreover, it has been suggested in the literature that the stability of particular hydrogen bonds there plays a key role in governing the transitions of the flap tips from a closed to semi-open conformation[9]. Therefore, the number of hydrogen bonds between specific pairs of interflap residues was calculated as a function of time. From our simulations, however, there is no clear change in the hydrogen bonding pattern in the flap tips. Specifically, across cases containing the wild type protease we do not see consistent hydrogen between particular interflap residues which may suggest the structure and properties of the ligand alter the dynamics there. However, comparing duplicate simulations of the same wild type/ligand pair, we again did not see any agreement in the pattern of interflap hydrogen bonding from one copy to the other. After we completed our analysis using the above three mentioned metrics and having not found significant structural differences between the wild type and mutant proteases, we decided to explore the correlations and patterns of motion within the protease.

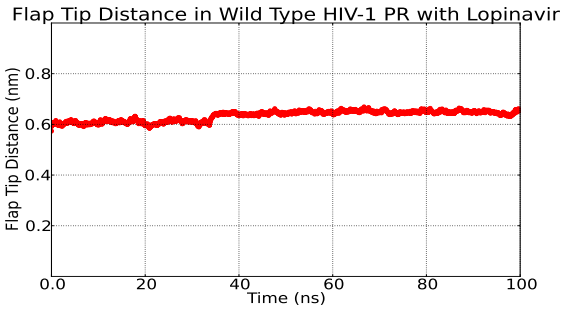


Figure 2: A running average of the flap tip distance verses time in the wild type HIV-1 PR with lopinavir bound.

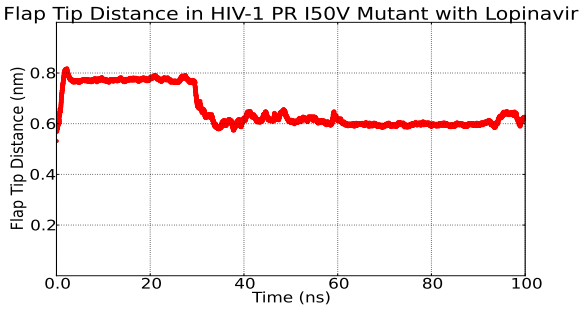


Figure 3: A running average of the flap tip distance verses time in the I50V HIV-1 PR mutant with lopinavir bound.

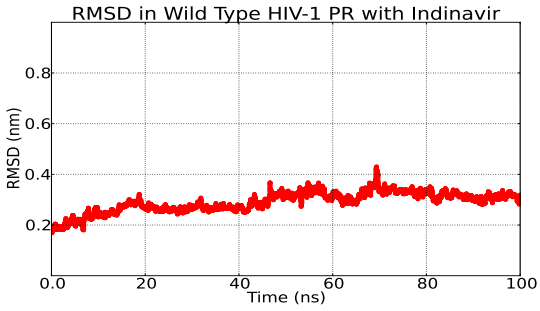


Figure 4: A running average of the flap tip RMSD verses time in the wild type HIV-1 PR with indinavir bound.

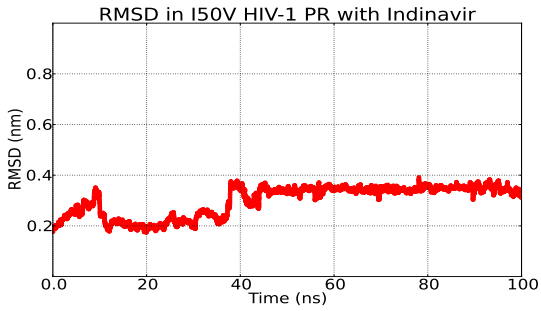


Figure 5: A running average of the flap tip RMSD verses time in the I50V HIV-1 PR mutant with indinavir bound.

4.1.1 Mutual Information

One way we sought to quantify the degree of correlation between a pair of residues was to measure their mutual information. This metric communicates to us the degree to which two random variables are linked; for example, high mutual information indicates a low uncertainty in one random variable given information about the other, and zero mutual information between two random variables implies the two variables are independent[8].

4.2 Correlation patterns

Using MutInf, a tool for measuring the mutual information between pairs of residues, we were able to put together mutual information matrices showing the degree to which one residue's dihedral movements are correlated to another's [10]. This was done by comparing the distribution of angles for each torsional degree of freedom for each residue to those of each other residue. Once we identified which residues were correlated in each simulation with MutInf and put together matrices showing that information using the R statistical package, we sought to see how the correlations were affected by the presence of mutations[12].

Identifying differences between wild type and mutant mutual information matrices cannot be achieved by simply taking the difference between two matrices because of a high level background noise. This was found by taking the difference between matrices from duplicate simulations of the same protease and ligand pair. As a result, we found it more useful to look at residues exhibiting a correlation above a cut off of 1.2kT in each case. This cutoff was the maximum correlation in the I82F/I84V HIV-1 PR variant bound to ritonavir in the first set of simulations, and it was the smallest correlation we found in all the cases we examined. This helped by eliminating correlated, adjacent residues from consideration which are not very interesting. One would expect two adjacent residues with bulky side chains to show some degree of correlation since steric hindrance by one residue's side chain may influence when the other's can and cannot move.

Once we identified the pairs of residues showing a high degree of correlation, the ones sharing a common residue were grouped together. We then compared the groups of correlated residues in the wild type protease to those in the mutant protease. We saw a high degree of variability in the location of the groups as well as the number of groups in

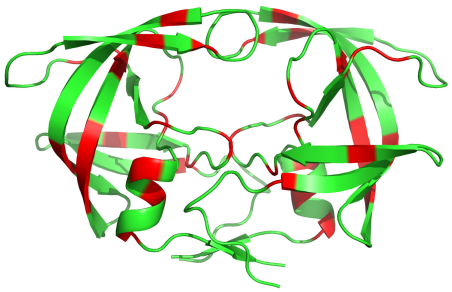


Figure 6: HIV-1 PR with residues found to have a 5% or greater mutation probability during drug treatment colored red[11].

most cases. Moreover, comparing the results from copies of the same case to one another, we again do not see much overlap between the locations of the groups of correlated residues. Further, in some cases, we had very few groups to use as a basis for comparison because of the chosen cut off value.

4.2.1 Mutation Sites and Correlated Residues

We also thought it interesting to explore whether drug induced mutations in the protease arise in correlated residues more often than not. It has been determined experimentally that 44 residues in the protease show a mutation probability greater than 5% in patients undergoing treatment for AIDS with protease inhibitors[11]. These 44 residues are shown in Figure 6. We calculated a 39% probability that a residue in a random picked pair would be at a mutation site. Next, when we determined the percentage of residues in correlated groups that are at mutation sites for each case, and we did not find a large deviation from that percentage. This suggests that these mutations do not occur at residues with correlated motions more often than other sites.

4.3 Kullback-Leibler divergence

A final analysis was done using Kullback-Leibler (KL) divergence. This analysis compared the overlap between a distribution of dihedral angles for each residue in the wild type protease to the corresponding residue in the mutant protease, according to equation 1. Ideally, this would allow one to identify areas where mutations alter the flexibility and motions of the protease. Before we performed this analysis, the simulation trajectory data from duplicate simulations of the same protein/ligand case was combined.

$$D_{KL}(P||L) = \int P(i) \log \frac{P(i)}{Q(i)} \qquad (1)$$

Our results show large differences between the wild type and mutants' distributions of dihedral angles for residues at the mutation sites as expected. However, the KL divergence at other residues, which were not mutated, is more interesting. In the second HIV-1 PR variant with mutations V82F and I84V, we still see a significant divergence in several of the flap tip residues as shown in Figure 7. Perhaps, the motions of these residues are transferred though the ligand to the residues at the flap tips.

5. CONCLUSIONS

In summary, we do not see significant structural differences between the wild type and mutant proteases at the 100ns timescale. Specifically, both mutants and the wild type do not largely differ with respect to their flap tip distance and RMSD on average. Also, we did not observe any patterns in the interflap hydrogen bonding in the same case

duplicate simulations, making it hard to identify alterations in the hydrogen bonding network induced by mutations. Next, using the results of a mutual information analysis to group correlated residues, we did not observe consistent patterns in the location of the correlated groups. And, there was not a correlation between the location of correlated residues and mutation sites. Finally, we preformed a KL divergence analysis which showed the mutated residues do have significantly different distributions of torsional angles. However, only in one case involving the V82F/I84V HIV-1 PR variant do we see the mutations possibility altering the motions of residues elsewhere in the protease. Moreover, it is seen that the residues in the flap tips also show a high KL divergence from the wild type in this case, suggesting the motions of residues 82, 84, 82', and 84' may be transferred though the ligand to the flap tips.

Despite the lack of large structural differences between wild type and mutant strains of the HIV-1 PR observed in our simulations, there is hope that future studies may reveal key changes conferred by the mutations onto the protease thereby lending insight into the mechanism by which resistance arises. One strategy may be to run simulations of the unliganded HIV-1 PR variants at the same timescale, since large scale flap tips conformational change has been observed to occur spontaneously on the nanosecond timescale by others[7]. Alternately, for the ligated enzyme, longer simulations on the microsecond timescale may be needed.

6. IMPACTS

This undergraduate research internship has helped motivate my decision to go to graduate school to study computational chemistry. While I considered going to graduate school a possibility prior to participating in this internship, the internship helped reaffirm my interest in it by providing the opportunity to participate in real research, proving it can be quite a rewarding experience.

In graduate school, I hope to continue my study of computational chemistry and learn more about different simulation methodologies and how they can be applied to solving real world problems. To further my abilities in this area, I plan to take courses in statistical mechanics, protein chemistry, and chemical kinetics and dynamics.

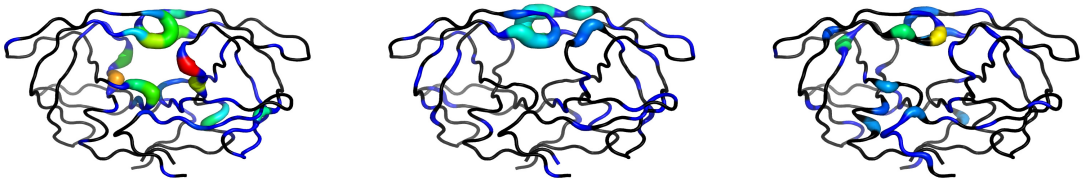


Figure 7: Putty representation of HIV-PR showing KL divergence for three variants. From left to right: a.) V82F/I84V, b.) I50V, and c.) wild type. Thicker putty indicates a greater divergence. Color also is used to indicate each residue's KL divergence with black indicating zero divergence followed by blue (low divergence) progressing to red (high divergence).

7. RECOMMENDATIONS

The Blue Waters Undergraduate Petascale Education Program provided a great introduction to the world of scientific computing. At the workshop, each intern was provided an abundance of resources, ranging from notes on different HPC architectures to information on topics specific for that interns project. Moreover, this workshop served a vital role in providing skills necessary to successful utilize high performance computing resources. Furthermore, the BW-UPEP provided excellent training for someone with little prior experience in HPC, making parallel computing concepts readily understood.

Unfortunately, the limited time of the two week workshop only allows so many topics to be covered. One way to maximize the benefit to each intern may be to schedule times were the interns are separated into groups based on their project's area. For example, groups could be centered around the life sciences, physical sciences and engineering, and computer science. When the groups meet, topics taught would relate only to the relevant subject area.

8. ACKNOWLEDGMENTS

I would like to thank Dr. David Mobley of the University of New Orleans for his guidance throughout this project and for the time he spent teaching me about computational chemistry. Also, thanks to Chris McClendon at UCSF for his technical help and insight with his mutual information and KL divergence code. Finally, thanks to Shodor and NCSA for supporting this project.

9. REFERENCES

[1] E. Alexov and M. Gunner. Incorporating Protein Conformational Flexibility into pH- Titration Calculations: Results on T4 Lysozyme. *Biophys. J*, 74:2075–2093, 1997.

[2] P. Ashorn, T. J. McQuade, S. Thaisrivongs, A. G. Tomasselli, W. G. Tarpley, and B. Moss. An Inhibitor of the Protease Blocks Maturation of Human and Simian Immunodeficiency Viruses and Spread of Infection. *PNAS*, 87(19):7472–7476, 1990.

[3] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The Protein Data Bank. *Nucl. Acids Res.*, 28:235–242, 2000.

[4] L. Galiano, F. Ding, A. M. Veloro, M. E. Blackburn, C. Simmerling, and G. E. Fanucci. Drug Pressure Selected Mutations in HIV-1 Protease Alter Flap Conformations. *J. Am. Chem. Soc.*, 131:430–431, 2009.

[5] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. *Journal of Chemical Theory and Computation*, 4(3):435–447, 2008.

[6] V. Hornak, R. Abel, A. Okur, B. Strockbine, A. Roitberg, and C. Simmerling. Comparison of Multiple Amber Force Fields and Development of Improved Protein Backbone Parameters. *Proteins Struct. Funct. Bioinf.*, 65:715–725, 2006.

[7] V. Hornak, A. Okur, R. C. Rizzo, and C. Simmerling. HIV-1 Protease Flaps Spontaneously Open and Reclose in Molecular Dynamics Simulations. *PNAS*, 103(4):915–920, January 2006.

[8] P. E. Latham and Y. Roudi. Mutual Information. *Scholarpedia*, 4(1):1658, 2009.

[9] D. Li, B. Ji, K. Hwang, and Y. Huang. Cruicial Roles of the Subnanosecond Local Dynamics of the Flap Tips in the Global Conformatinoal Changes of HIV-1 Protease. *J. Phys. Chem. B*, 114:3060–3069, 2010.

[10] C. L. McClendon, G. Friedland, D. L. Mobley, H. Amirkhani, and M. P. Jacobson. Quantifying Correlations Between Allosteric Sites in Thermodynamic Ensembles. *J. Chem. Theory Comput.*, 5(9):2486–2502, 2009.

[11] H. Ohtaka, S. Muzammil, A. Schon, A. Velazquez-Campoy, S. Vega, and E. Freire. Thermodynamic Rules for the Design of High Affinity HIV-1 Protease Inhibitors with Adaptability to Mutations and Highly Selectivity Towards Unwanted Targets. *Int. J. Biochem. Cell Biol.*, 36:1787–1799, 2004.

[12] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation

for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[13] G. R.E, A. E.G, and G. M.R. Combining Conformational Flexibility and Continuum Electrostatics for Calculating pKa's in Proteins. *Biophys J*, 83:1731–1748, 2002.

[14] Schrödinger, LLC. The PyMOL Molecular Graphics System, Version 1.3r1. August 2010.

[15] A. Valazquez-Campoy, S. Vega, E. Fleming, U. Bacha, Y. Sayed, H. W. Dirr, and E. Freire. Protease Inhibition in African Subtypes of HIV-1. *AIDS Rev.*, 5:165–171, 2003.

[16] J. Wang, W. Wang, K. P. A., and D. A. Case. Automatic Atom Type and Bond Type Perception in Molecular Mechanical Calculations. *J. Mol. Graphics Modell.*, 25:247–260, 2006.

[17] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case. Development and Testing of a General AMBER Force Field. *J. Comput. Chem.*, 25:1157–1174, 2004.

[18] K. Wittayanarakul, S. Hannongbua, and M. Feig. Accurate Prediction of Protonation State as a Prerequisite for Reliable MM-PB(GB)SA Binding Free Energy Calculations of HIV-1 Protease Inhibitors. *A. Comput. Chem.*, 29:673–685, 2007.

December 2011

Volume 2 Issue 1

TABLE OF CONTENTS

ISSN 2153-4136 (online)