1-2006

The Limits of Removing Uncertainty: The Role of Computational Science in Scientific Inquiry

Sean Metz
The College at Brockport

CMST Challenge Project

Sehj Kashyap, Jennifer Pinar Yasar,

Cassandra Taylor

Teacher: Sean Metz

Brighton High School

10th Grade

The limits of removing uncertainty: The role of computational science in scientific inquiry

CMST Tools Used:

Interactive Physics (IP)

Fortran

Excel

SmartBoard (SB)

TI-84 Calculator

Background

How often do three fellow high school-attending 10th graders set out to understand the roots and foundations behind Computational Science and the uncertainty involved in gathering the accurate data that computational modeling provides? We are fellow students at Brighton High School under the direction of an Earth Science teacher, Sean Metz, who are out in the complicated world of computational science to learn not only how to use the infinite amount of software programs hypothetically at our disposal, but also what computational science can do for us. We originally started out as a five member delegation consisting of a variance of grade levels, however, due to inconsistency in dedication, two of the team members dropped out. Through the project, we not only gained the friendship and loyalty of our peers, we also acquired the understanding of computational science, its influence on modern day calculations, and the barrier that one reaches with increasing accuracy in software programs. The project however was not as smooth as our friendship came to be, but was filled with bumps and detours. We encountered many daunting problems, equally daunting results and to the delight of the group had numerous

epiphanies. These obstacles only helped us in making the project more in-depth and answering questions we had forgotten to ask ourselves.

At first we laid out the infrastructure of the project. We focused on dividing the primary part of the experiment, choosing a topic. Each of us brainstormed ideas for projects that could be simulated through the software at hand and which were related to the curriculum of our class. This was maybe the toughest part of the entire project. Science is a very broad field and the opportunities/choices are endless. It took us a while to decide on a topic which would seem interesting to us and others. Then, one day during class, Sehj's chemistry teacher briefly mentioned the Heisenberg Uncertainty Principle. Through brief research, we got the basic jest of the principle and were immediately interested in it. One thing led to another, and we all agreed to try and simulate it. We consulted our teacher, Mr. Sean Metz, and were given the green light. Hence, we set out to simulate it through CMST tools. Nevertheless, it was quite daunting and our teacher himself was not sure if the tools at hand would be capable of simulation. While it was initially frustrating to actually grasp the principle and its implications, we soon got it, and hence got to inputting that understanding into computer software. Soon, however we realized that with the tools at hand, the immensely complex principle and the supplementary equations could not be simulated. We moved on to a 'simpler to grasp' yet 'difficult to understand' topic; simple harmonics. Though the name is simple, and how to simulate it in IP is equally simple, the background calculations which are solved, hidden for the user's view in IP, are quite complex. Hence we decided to show and gain understanding for ourselves of how computational science takes the formulas and equations and puts them into a moving box on a spring.

Our new goal was not a complete turn around from the Heisenberg Uncertainty Principle. In simulation programs and in computational science there is always a need for greater accuracy. No matter what increments of a function you get, there is some uncertainty. Decreasing the increments of the function decreases this uncertainty, but there is the infinite/loop process in which even if you keep going smaller increments uncertainty keeps persisting. Uncertainty (in this case, uncertainty of position and velocity) is inversely related to the function, in this case increment of time, and hence neither will reach an absolute end, in this case we will always have a computational error limiting perfect accuracy of position or velocity.

Computational science is the use of computers to perform research in other fields. It is the application of computer simulation and other forms of computation to problems in various

scientific disciplines. It is not to be confused with computer science, which is the study of topics, related to computers and information processing.

Now that we had fully comprehended our goal, we set out on showing how computational science has allowed easy simulation of complex principles. Simple Harmonics is not a very easy concept to grasp even for the better of us. Yet advancements in computational science over the decades have allowed the principle to be simulated by an elementary school kid in a minute or less. Hence our goal was to bring into light the inner workings of simulation programs like IP. We decided that FORTRAN, Excel and the Ti-83 Graphing Calculator were the correct tools to show the numerical and formula-based aspects that allow the box to be shown as moving in IP. Recognizing our goal was a major and essential part of the project. Once we overcame this boundary, we focused on dividing the work up.

Through democratic process, we allotted each other working with different programs. Sehj was given the job to simulate the box with spring in Interactive Physics. Jennifer was given the job to write a program in FORTRAN that would output the respective velocities and positions of the spring at varied increments of time. Cassandra was given the job to simulate the problem using Excel's programming capacity. We knew modeling via Interactive Physics was much easier than Fortran and Excel, but we thought that if we could model it via Fortran and/or Excel we would have a better understanding, and maybe control, of the simulation. We used different programs to both verify our results and also gain greater control of the simulation. With careful distribution of work, in correspondence with each other's schedule, we were on the clear path of completing our project.

There were problems we encountered at various intervals of time throughout our project; however we did have the cure for the plague. In Interactive Physics, we basically had no problems what so ever. However, we had anticipated this to occur. As mentioned before, computational science has allowed simulation of complex principles to be easily done. Hence, simulating with IP was a breeze and there were minimal problems. The only problems we did have were finding out how to set the spring coefficient of the spring. After messing around with the program, and with the assistance of Mr. Metz, we figured out how to do so and were able to finish that part of the project.

FORTRAN was comparatively harder to work with as compared to Interactive Physics. Jen, who was allotted this part of the project did not even have an idea of what the program was,

let alone how to write a program in it. She had, however, previously programmed in Java and was familiar with computer languages. During one of his visits to Mr. Metz's class, we asked Dr. Yasar and he suggested that Fortran might be easy to use. He did provide a loaner software to us, and, hence, Jen set off to learn from the start how to program and how to write the language. After her first initial experience, she did confirm that Fortran was a lot easier than writing Java programs.

When using Excel, we encountered several difficulties. First and foremost was that of obtaining the proper formulas. This obstacle is detailed more when Excel is talked about in the Modeling section. The main issue here was that we did not use *x* as a changing value based on the previous value, but rather as constantly varied to other numbers. We did not notice this for a while, and assumed for a little bit that Excel could not function as seeing repetitive motion like this, that it would only see it as a straight line graph that approached zero but never curved back up. We tried using analytical formulas instead, but we could not input radians properly into these equations. We finally got the equations to work.

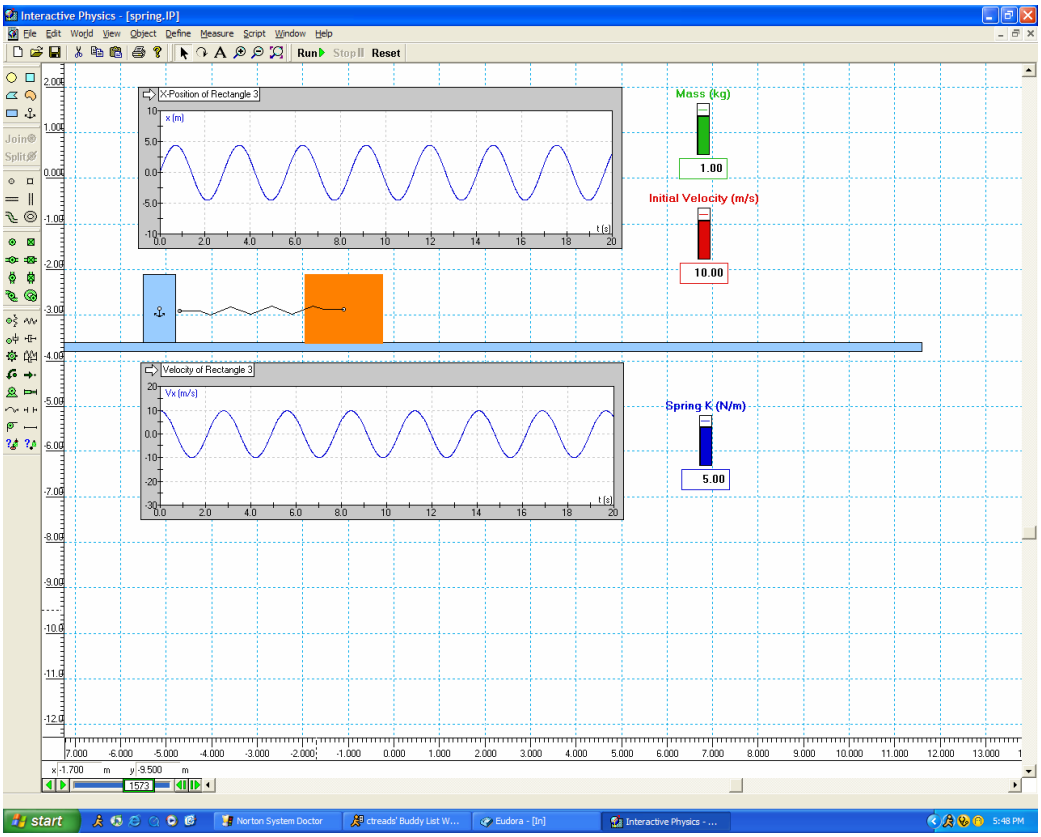
Modeling with the Programs

a) Modeling with Interactive Physics

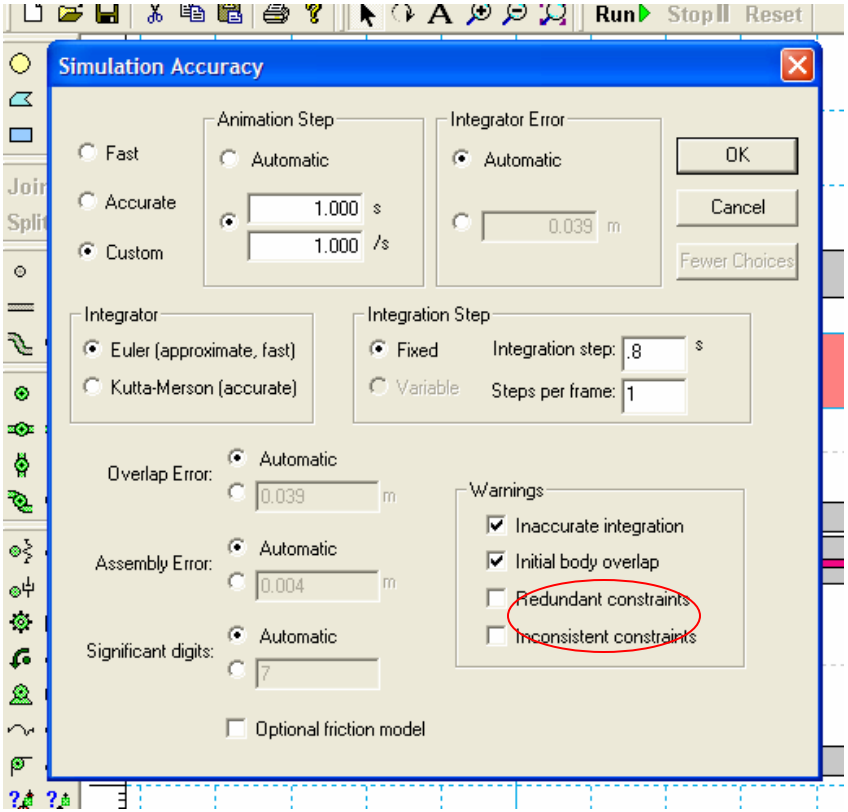
The first program that we simulated the spring system in was Interactive Physics. We chose this to start with because it was most similar to an experimental solution, which is often easiest to understand. This is the best part about IP; that it is similar to a hands on experiment. The procedure is very simple to follow.

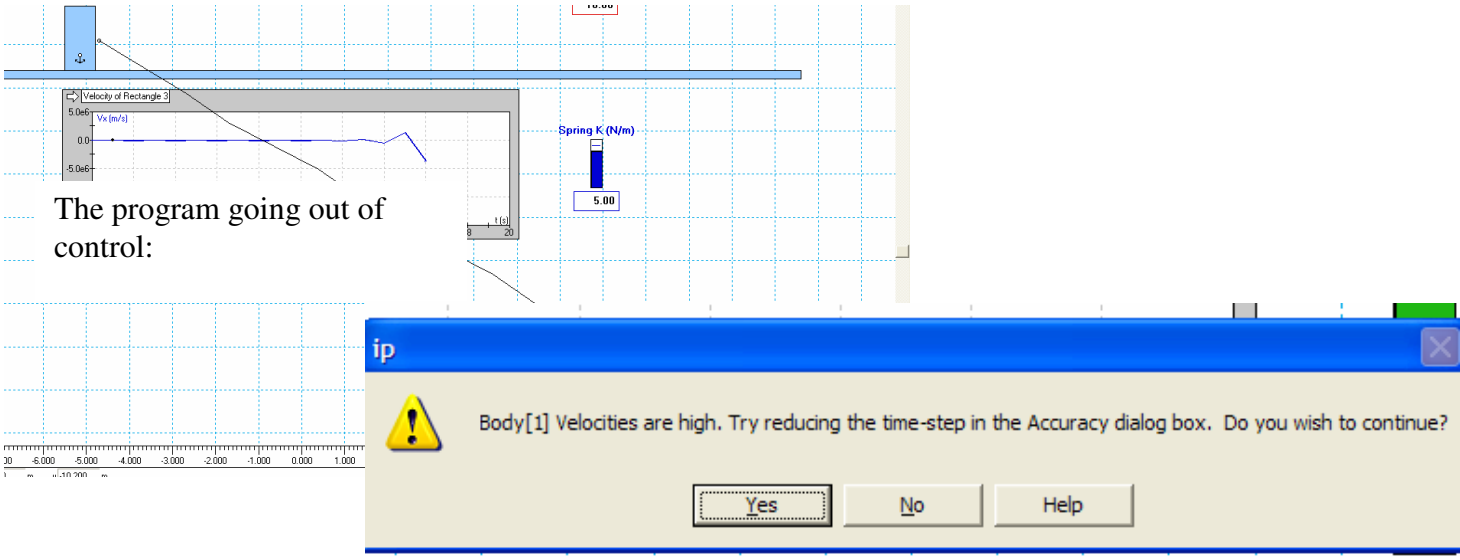
1) Use the drawing tools to create a box on a platform, with a wall on one side.
2) Anchor the platform, and attach the wall with a rigid pin joint.
3) Connect the box to the wall with a spring, attached at the center of the box.
4) Set friction coefficients (both static and kinetic) on the box and the platform to 0. To do so, double click on the object and change the values in the property box.
5) Check that air resistance is turned off. Air resistance is a category under the World menu.
6) Set the mass of the box to 1kg, the spring coefficient to 5 N/m, and give the box an initial velocity of 10 m/s to the right (away from the wall). All of these values should be set by double clicking on the object and changing the values in the property box.
7) Measure horizontal displacement and instantaneous velocity as functions of time in graphs. (The data does not need to be exported to Excel because IP is able to create

graphs itself). To measure them, click X-Position and Velocity from the Measure menu while the box (Square 1) is selected.



As with the other programs, we wanted to study how the results changed if we changed the accuracy in the simulation. To do so: World=>Accuracy. Change the Fixed Integration step to the dt desired. We started off using .05, and .1. Both of these produced reasonable graphs. However, when we changed it only as high as .8, even, the program could not work. The simulation went out of control, and it said that it could not compute the data. This was very interesting, because we found out that not only would we sacrifice our accuracy by increasing dt, we would also sacrifice the integrity of the program itself. Since it was only calculating in such large increments, it was getting positions on the graph that did not represent the simulation. On the other hand, the only limit to how accurate we can get is how many decimal places the program can compute.





Interactive Physics model (with error dialogue)

b) Modeling with Excel

To simulate the program on Excel, we developed formulas relating to motion. They were derived from basic force principles (Hooke's Law and Newton's Law), such as $F=- k \cdot x$, $F=m \cdot a$, and $a=dv/dt$. They were combined into the formula for velocity $v_{new}= v_{old} - (-k/m) \cdot dt \cdot x$, and position $x_{new}= x_{old} + v \cdot dt$.

Most of the difficulties in using Excel came from getting the formulas to work right. It was surprisingly easy to actually input them into Excel, and we learned a tool on linking cells instead of just referencing them to make the final project look nicer. When we were developing the formulas, however, the major problem was that we didn't realize that we had to solve for updated values of x, so we were just solving for x unrelated to the previous displacement value. Numbers weren't filling out right, we were getting illogical answers, and graphs that looked completely different from the other simulations.

Once we fixed that formula, however, everything was very simple. We inputted the constants, and created the two variable tables seen in the program.

We also linked everything, so that *dt* could be changed and all the data values as well as the graph would change correspondingly. This was the major point of our project: **to show if a smaller *dt* gives more accurate results.** It is obvious that .05s as dt gives the more accurate results, while .1s is still reasonably accurate in representing the trends, while 1s is hardly intelligible as the same data at all. The following graphs are dt= 0.1, 0.5, and 1, respectively.

The Limits of Removing Uncertainty:
The Role of Computational Science

CONSTANTS

k (N/m)	m (kg)	Vo (m/s)	Δt (s)
5	1	10	0.05

EQUATIONS USED

v:	Vf= Vo - (-k/m) *Δt * x
x:	Xf= Xo + v * Δt

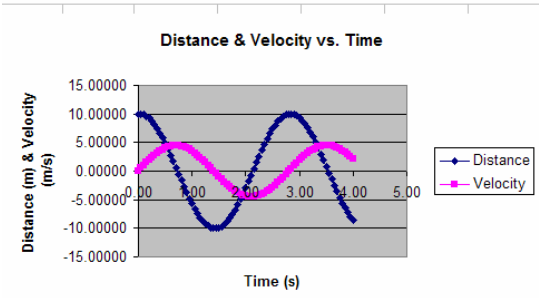
NOTE:

To change Δt, change the value in G8 to the desired increment. Suggested values are .05 (shown), .1, .5, and 2. .5 and 2 do not accurately represent the trendlines, which is the point we are trying to make, in that the less precise the measurements (the larger Δt), the less accurate the resulting data will be. The graph will change accordingly to the increment chosen.

Figures are shown to the fifth decimal, but are calculated indefinitely.

DATA

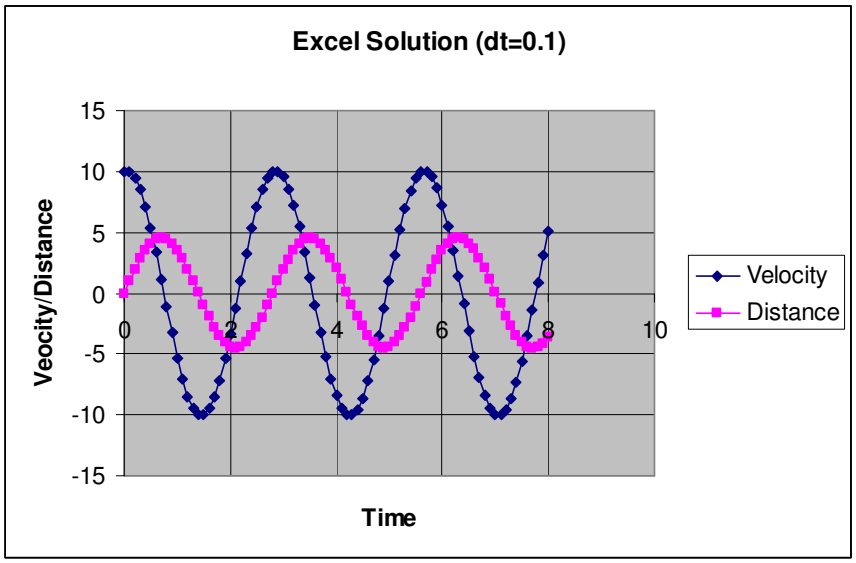
t (s)	v (m/s)	x (m)
0.00	10.00000	0.00000
0.05	10.00000	0.50000
0.10	9.87500	0.99375
0.15	9.62656	1.47508
0.20	9.25779	1.93797
0.25	8.77330	2.37663
0.30	8.17914	2.78559
0.35	7.48275	3.15973
0.40	6.69281	3.49437
0.45	5.81922	3.78533
0.50	4.87289	4.02897
0.55	3.86565	4.22226
0.60	2.81008	4.36276
0.65	1.71939	4.44873
0.70	0.60721	4.47909
0.75	-0.51256	4.45346
0.80	-1.62593	4.37217
0.85	-2.71897	4.23622
0.90	-3.77802	4.04732
0.95	-4.78985	3.80782
1.00	-5.74181	3.52073
1.05	-6.62199	3.18963
1.10	-7.41940	2.81866
1.15	-8.12407	2.41246
1.20	-8.72718	1.97610
1.25	-9.2	
1.30	-9.5	
1.35	-9.8	
1.40	-9.9	
1.45	-10.0	
1.50	-9.8	
1.55	-9.6	
1.60	-9.2	

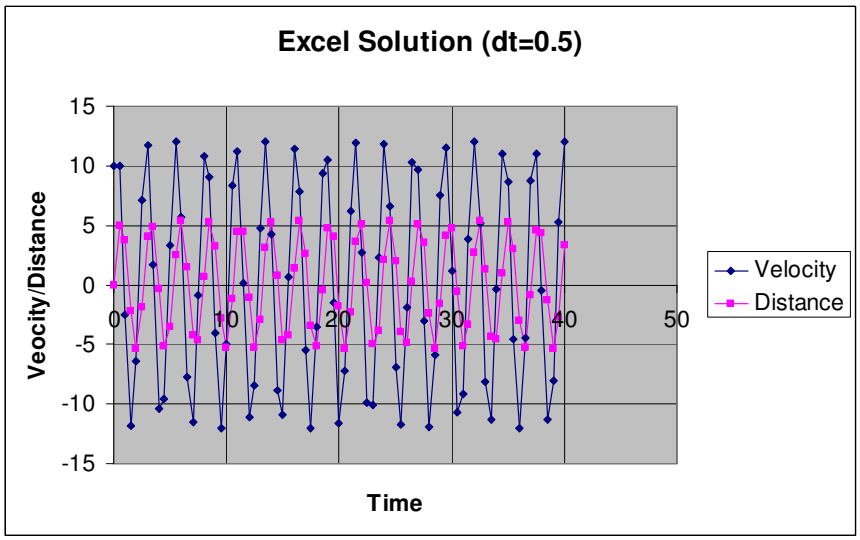


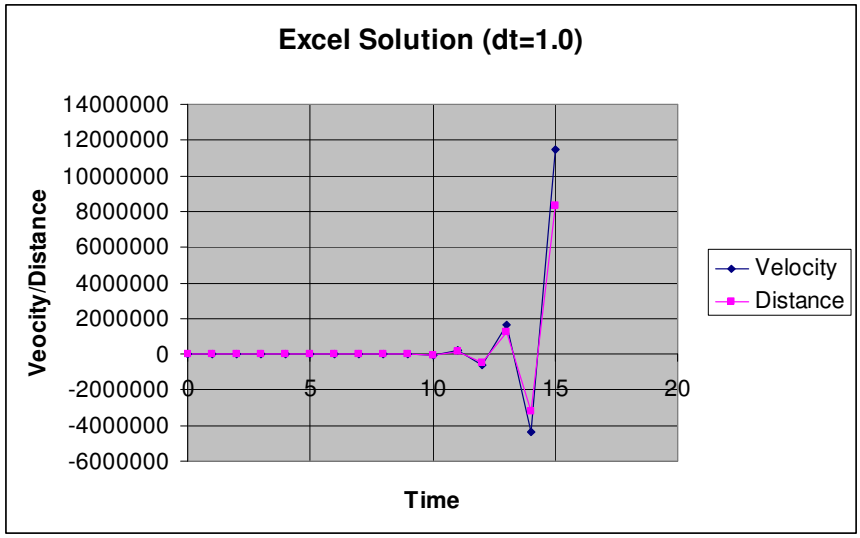
FORMULAS

F=-kx F=ma a=dv/dt
M(dv/dt)=-kx
Dv=(-k/m)*x*dt

V=dx/dt
dX=v*dt
Xf=Xo+v*dt


Excel Solution (dt=0.1)
Veocity/Distance
15
10
5
0
-5
-10
-15
0
2
4
6
8
10
Time
Velocity
Distance

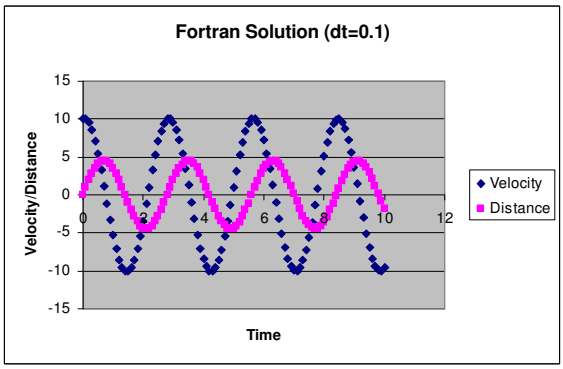

Excel Solution (dt=0.5)
Veocity/Distance
15
10
5
0
-5
-10
-15
0
10
20
30
40
50
Time
Velocity
Distance

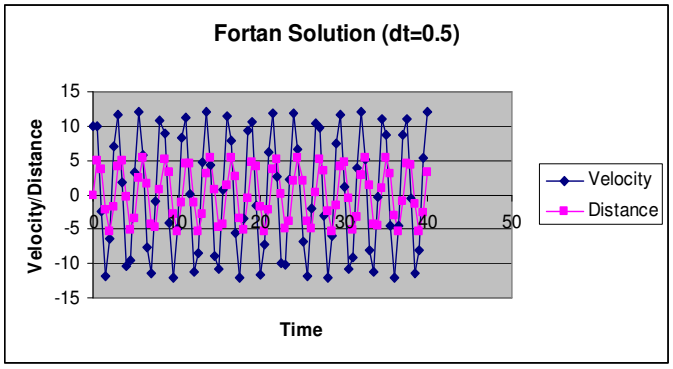

Excel Solution (dt=1.0)
Veocity/Distance
14000000
12000000
10000000
8000000
6000000
4000000
2000000
0
-2000000
-4000000
-6000000
0
5
10
15
20
Time
Velocity
Distance

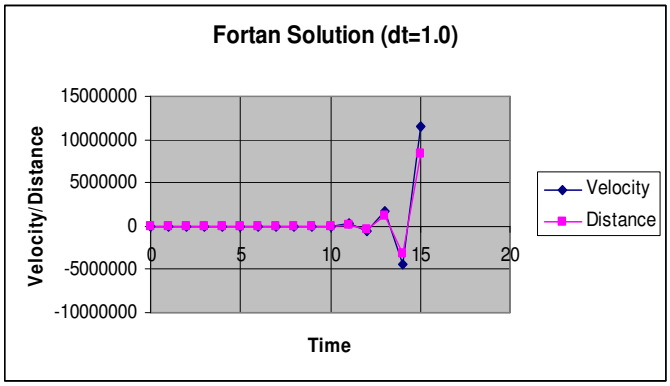
c) Modeling with Fortran

FORTRAN is said to be one of the first high-level languages. We found it to be similar to our everyday use of English language. To print something, you write print, followed by the list of variables to be printed. Since concepts were very much close to Java, it did not take much time to write a simple program. In fact, the version provided to us by the CMST office was very easy to use. To start the program, one clicks on the Compaq Visual Fortran icon, which gives an option of starting off from a sample program, which is what we did! The rules we had to learn were that each program has to have a start and end statement and that each loop has to have a start and end statement. Loops are so easy to write with Fortran. The difficult part, of course, is when the compiled program gives errors. One error may lead to a bunch of technical statements, giving an illusion that nothing is right in your program. We consulted Dr. Yasar in those cases.

One might imagine that using Fortran or Excel is not easy as the Interactive Physics. However, these not-so-easy tools give you greater control. Everything that goes into the model is out there before your eyes, whereas with Interactive Physics it is hidden behind several icons. We asked Dr. Yasar why he does not use IP to do his computational research. He said that Fortran allows him to program complex problems, which reminded us the limit of IP to handle the Heisenberg Uncertainty problem we started off with. Once we were able to get Fortran to compile and run without errors, we printed position, velocity, and time data and copied them to to make graphs. We examined impact of step size (dt) on the results as shown below. We also compared these graphs to those obtained by Excel (with its own data) and IP. It is clear again that for dt=1.0, the solution becomes meaningless.

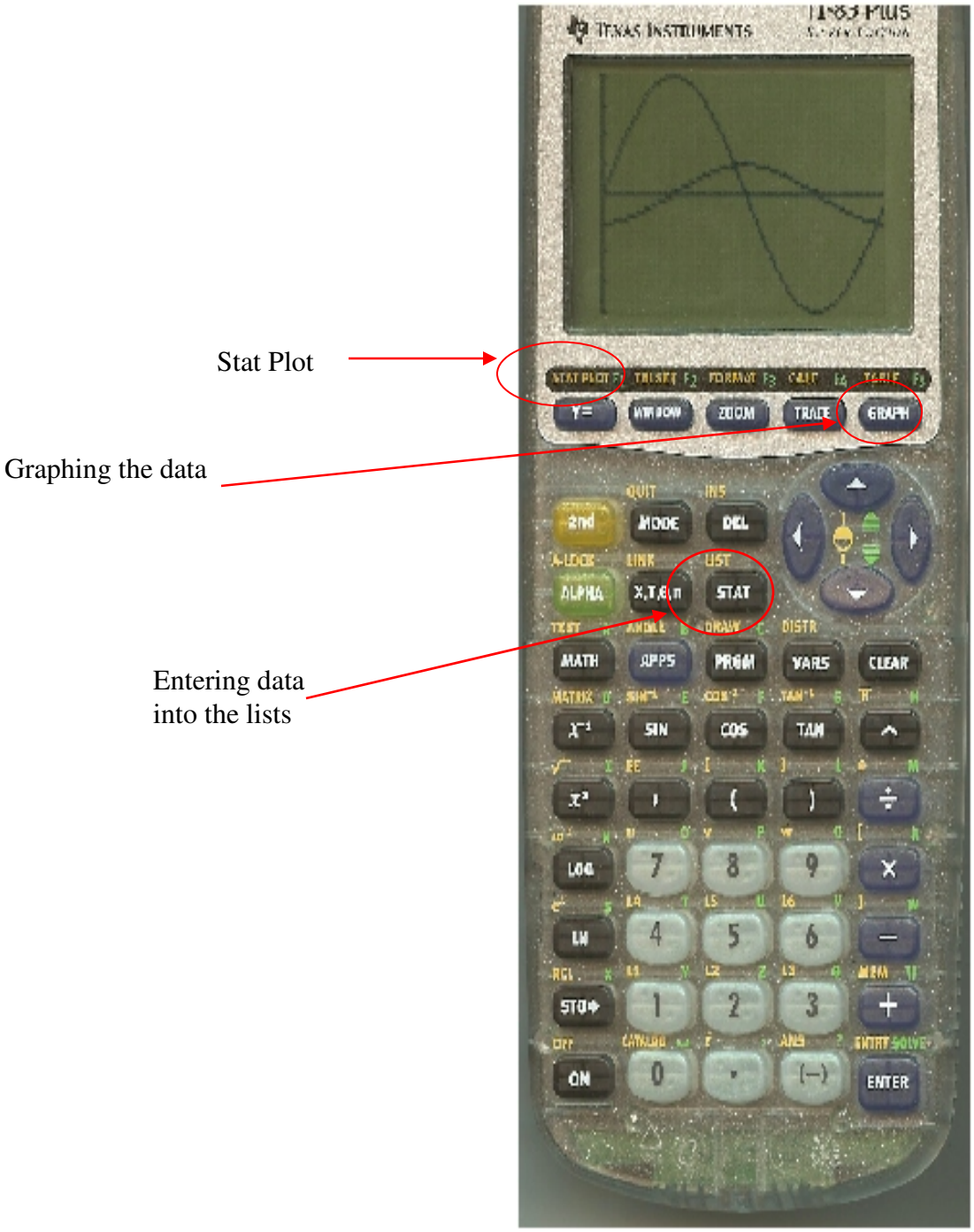






d) Plotting with Ti-83

One other program was used: the TI-83 Plus calculator. We used Stat Plot to enter data that we got by using the equations derived in Excel, and then graph them as a scatter plot. We did not use the linked connection to the computer program, because we felt that it was very similar to Excel, and that we would rather show how the same results could be formulated from an instrument available to most high school and several middle school students.



d) Impact of SmartBoard

The SmartBoard was helpful and playful at the same time. It took out the stress related to the competition but also helped in certain ways. By using the smartboard we were able to work on the computer together as compared to sitting around a laptop and trying to get the one person in control to follow their directions. With the smartboard all the team members could freely access the files. Instead of having to make one person sit through the meeting clicking off on the computer, we were able to cooperatively share the board and reduce the time spent in persuading the person-in-charge to implement their idea.

Summary

CMST was the best challenge that we have taken. It not only demanded what we knew but also made us learn new aspects of science. It opened our eyes. It has taught us the infinite expanse of science and physics. We are never limited in science. We can keep going in depth till infinity or keep zooming out to infinity. We are not restricted but by underestimation of science. For us, CMST showed us aspects of science, which were not only limited to physics, but also the science of computer modeling. CMST not only challenged us in thought and but also psychologically. It made us put aside the difference we had and cooperate as a team. Working in a team was also an enjoyable experience. Aside from the group planning meetings, we had fun, cracked jokes and made good friends. CMST was a challenge that none of us will forget. It was a discovery in its own.

Computational science, as we realized, has influenced the world so greatly. Its plays a key role not only in mathematics and science, but also in removing uncertainty. Computational science has allowed complex equations and solutions to be reduced into a simulation program, which can be used by an 8 year old. The original goal of computational science when it was founded was to help in the simulation of science and math principles through the use of computers, and it has aided this area greatly. Through our project, we learnt that computational science isn't just another job field in the sea of many others, but is actually one, which influences us more so than others.

In simulation programs and in computational science there is always a need for greater accuracy. To get more accuracy, or less uncertainty, you need smaller increments of the function. When computational science first started out, the accuracy we have right now in small programs could only be achieved through massive computers. However, now the same simulations that required a computer as big as a room can be simulated in a small program on an equally small computer. In striving to get less uncertain and more accurate, we know that we have to have smaller increments of the function. However, there is no limit to this as you can go infinitely smaller, except for the power of computers. This is the biggest revelation that we made through our project. Our project not only made us realize the expanse of computational science and the ease it brings, but it also gave us an appreciation of tools such as the CMST tools and more.

This project has been a moving target and certainly a challenge. There were times we did not know whether we could accomplish our initial objective of simulating the Heisenberg Uncertainty Principle and here we are with tools and results to show something even bigger about which we are definitely not "uncertain." Use of technology obviously enhanced our ability. Once we had a model in place, it served as a springboard to launch more ideas and tasks. Once we passed a certain threshold, it became possible to do more. While we were at some point trying to just complete this project and forget about it, it now became evident that we want to do more and more with it. Most of us now want to take an AP class in the coming years. We may even consider taking a CMST course if it is offered to high school kids.

References

1. Mr. Steve Whitman, BHS Physics Teacher
2. http://www.kettering.edu/~drussell/Demos/SHO/mass.html
3. http://theory.uwinnipeg.ca/physics/shm/node2.html
4. http://www.aip.org/history/heisenberg/p08.htm
5. Book: Brighton Highschool textbook, Physics: Principles and Problems, P. Zitzewitz, page 181.