Finals Projects Symposium

ASU PHY494 Computational Methods in Physics (Spring 2016)

Arizona State University, Department of Physics
Room PSH 355

Thursday May 5, 2016, 11:30am–2pm

1 Posters

#	title	authors
1	ATLAS	Bobby Hackett, Jordan Pagni, Alex Warren
2	Billiard Simulation	Nathaniel Soderberg, Nik O'Brien, Kit Wing Fung
3	Blackjack Card Counting Simulation	Ryan Heilman, Layne Bradshaw, Jacob Dwyer
4	Digital Signal Processing Techniques in Python	Charles E. Fortune, Skyler J. Hugo, Trevor Van Engelhoven
5	Replicating results of 2016 March Madness Tournament	Bryan Katterman, Alejandro Martinez
6	Rendezvous with Rama: Ramageddon!	Jordan Boyd, Ashley Mascareno, Andy Winhold
7	Stern-Gerlach Simulation	Andrew Durkiewicz, Nate Simon, and Greg Vetaw

2 Procedures

Mount your poster on the provided poster board and attach the number assigned to you (see table above). At the end of the symposium, return all mounting materials and take your poster with you.

Note that the instructor might take pictures of the posters as an additional record for grading purposes.

2.1 Q&A

- Each member of the team will be asked to engage in an individual Q&A with the instructor in front of the poster of about 6 minute duration. The Q&A will be graded and be part of the final grade.
- The TAs will also ask questions and report their assessment to the instructor. However, they will only report positive evaluations, i.e., you cannot make your grade worse by talking to a TA.
- If you are not engaged in a Q&A then you are free (and encouraged) to look at other posters.

2.2 Poster prize

Participants will be able to vote for the best poster. Ballots will be provided: rank order the top three posters and drop your ballot into the collection container. The winning team will be awarded a prize.

3 Abstracts

Abstracts[1] are listed in the order of appearance in the program.

#1 • ATLAS • Bobby Hackett, Jordan Pagni, Alex Warren
`https://github.com/ASU-CompMethodsPhysics-PHY494/final-atlas-mission`

Our goal was to model the trajectory of a spacecraft, ATLAS, as it made its way to Saturn's moon Titan, on a journey of astrobiological exploration. The first step was to simulate a dynamic solar system of seven massive bodies that interact gravitationally. Each body is given an initial position and velocity–relative to the Sun at the origin–using values from real astronomical data. The positions of each body are calculated as a function of velocity, which itself depends on the force of gravity between bodies. Using a `for` loop, we iterate through a predetermined number of timesteps–updating the current positions, velocities, and gravitational forces. Once the solar system was complete, we added ATLAS. The spacecraft behaves just like any other body, but with an initial mass, position, and velocity, that are based on both real and hypothetical space exploration missions. Through careful analysis of many ATLAS trajectory simulations, we were able to project initial conditions that lead to a successful intercept with Titan. We didn't fulfill all of our hoped for

[1]The 200-word limit to the abstract text is indicated by graying out any text beyond the limit. For a real conference, your abstract would have been truncated or rejected by the submission system.

objectives, but with more time, the ATLAS mission would have: placed ATLAS in a stable orbit in the Titan-Saturn system, utilized gravity assist maneuvers, and calculated the fuel expenditure on route.

Code is available under the MIT LICENSE.

#2 • **Billiard Simulation** • Nathaniel Soderberg, Nik O'Brien, Kit Wing Fung

`https://github.com/ASU-CompMethodsPhysics-PHY494/final-billiard-simulation`

The Problem - Billiards is a game that has its roots buried in physics and geometry. However, human players are doomed to sometimes fail. Therefore the purpose of this simulation is to have a computer deduce the most efficient shot given an initial setup of balls. In the development of this code certain problems arose such as; implementing the holes, removing balls that fell in these holes, handling collisions, and coding the visuals.

The Approach - To simplify the simulation, python objects were created to easily call the balls and table in the program. Friction was also neglected and only five balls were observed, starting in predetermined positions. The program runs in a series of steps that starts by recording each ball's position. Balls are moved based on velocity, remove balls if needed, and then update positions based on collisions.

The Results - Because efficiency is the goal, the program gives amount of time and number of balls left on the table as outputs. If a certain shot angle results in all of the balls being holed in the shortest amount of time that angle is declared the most efficient angle.

Conclusion - With this program the computer can find the best shot given certain parameters.

#3 • **Why we are Dropping out of College: AKA Blackjack Card Counting Simulation** • Ryan Heilman, Layne Bradshaw, Jacob Dwyer

`https://github.com/ASU-CompMethodsPhysics-PHY494/final-card-counting-simulation`

The purpose of this project was to determine not only whether or not card counting in the game of blackjack is a viable strategy, but what a successful betting scheme would look like. The major hurdles that needed to be overcome for a success was the implementation of code that could simultaneously run a card game, while keeping the card count, and solving the math required given by the High-Lo counting system. This was accomplished via object oriented programming that would set up a game with players and determine whether or not to hit/stay and how much one should bet according to the cards and rules that were provided. The simulation was successful in the sense that we found betting patterns with net gains, while others produced large losses in a players bankroll. With over 1000 simulations run, we found, for the most part, that more conservative patterns led to larger overall gains. For example, a strategy with wagers ranging from \$145–\$165 led to an annual income between \$250,000 and \$350,000 per year.

Code is available under the MIT License.

#4 • **Digital Signal Processing Techniques in Python** • Charles E. Fortune,

Skyler J. Hugo, Trevor Van Engelhoven
https://github.com/ASU-CompMethodsPhysics-PHY494/final-digital-signal-processor

Digital signal processing written in Pure Python designed to generate and foremost manipulate waveforms for audio design purposes. The modern music industry revolves heavily around digital audio-mixing, and this project ventures into this realm by constructing what is colloquially known as an effects processor comparable to those seen in use across the music industry. Ideally, this system could be expanded upon given proper time and funding to increase overall fidelity quality and effect variety (in the form of improved hardware and code optimisation) Primary coding methods can be broken down into three immediate segments of design: function generation, effect processing in a Python environment, further output as audible noise utilising Raspberry Pi interface. The results are promising, with outcomes extending well beyond initial forecasted success projections. Working Python code (though not optimised) wrapped through a Raspberry Pi output-system produces desired effect-processing with little to no audio latency and justifiable loss in overall audio fidelity. This project serves as a stable proof-of-concept for the further development of Python as an audio effects-processor. The research team has concluded that further investment in project-development will likely yield further positive results.

#5 • Replicating results of 2016 March Madness Tournament • Bryan Katterman, Alejandro Martinez

https://github.com/ASU-CompMethodsPhysics-PHY494/final-march-madness

Correctly predicting the odds of all 63 games of the March Madness tournament if relegated to just pure random chance, are 1 in 9,223,372,036,854,775,808 (over one in nine quintillion). Our goal for this project was to create a Monte Carlo model to predict the outcome of the 2016 NCAA March-Madness tournament and compare our results with industry claimed predictive standards. We accomplished this by collected statistics on the outcomes of the previous 35 years of tournaments in conjunction with the statistics from the 2016 participating teams and created a Monte Carlo model using basic Bayesian inference. The two biggest hurdles for this project have been: a) the time consuming nature of gathering and analyzing statistics to build an accurate model and b) trying to find industry hints toward what accurate prediction algorithms look like. The latter is something the sports and gaming industries rarely disclose and as an outsider, we had to create our own predictive equations. Overall, we achieved our goal of creating a predictive model using team statistics, however our results were fairly inaccurate. Our model was able to correctly predict the winner of the tournament at a rate of 1 in 32, and the 2 teams that made it to the championship games at a rate of 1 in 270, which are odds about twice and four times better than chance respectively. Future work will include updating our model with more robust statistics to improve accuracy and research player stats to feed into the model.

Code is available under the Apache 2.0 License.

#6 • Rendezvous with Rama: Ramageddon! • Jordan Boyd, Ashley Mascareno, Andy Winhold

`https://github.com/ASU-CompMethodsPhysics-PHY494/final-rendezvous-with-ramageddon`

Near earth objects (NEOs) and the devastation they have caused Earth in the past continues to pose risks in the future. Should an extrasolar object following a hyperbolic trajectory graze too close to Earth, determining whether the Earth is on a potential collision course is vital. Inspired by Arthur C. Clarkes “Rendezvous With Rama”, this Python-based simulation reenact the hyperbolic orbit of extrasolar object Rama, a 50km long, cylindrical alien starship on a trajectory through the inner solar system passes. The simulation scenario imagines what kind of threat Rama may pose if it was a bit closer to Earth. Using Newtonian and Keplerian orbital mechanics coupled with the velocity verlet algorithm, the simulation computes the positions and velocities to make predictions about the orbits of Mercury, Venus, Earth, Mars, and Rama from a heliocentric frame of reference. The prediction for the velocity Rama needed to maintain a hyperbolic orbit was found to be 41.7 km/s, and the smallest distance between Earth and Rama for the duration of the trajectory was found to be 39,127 km. The eccentric anomalies for the inner planets were found to converge at tolerance 1e-5.

#7 • Stern-Gerlach Simulation • Andrew Durkiewicz, Nate Simon, and Greg Vetaw

`https://github.com/ASU-CompMethodsPhysics-PHY494/final-stern-gerlach-simulation`

In 1922 Otto Stern and Walther Gerlach confirmed that microscopic particles have a quantized intrinsic degree of freedom called spin angular momentum when they sent a beam of sliver atoms through an inhomogeneous magnetic field. Since silver atoms have an extra electron in the 5s orbital, the result showed that the spin of the electron is $\pm\hbar/2$. Thus, for this project we sought out to simulate the Stern-Gerlach experiment by using the finite difference method with a Gaussian wavepacket implemented in Python. We also wished to reconstruct the results of the experiment in the classical limit where the states of the electron were assumed to be a continuous band symmetric about the axis where the beam splits. The major hurtle that was encountered was trying to get the states to split and discretize in a manner that made sense for both the classical limit and the quantum limit. The results that were obtained in the semi-classical limit showed that when electrons exit the magnetic field they can populate any state that is symmetric about their initial axis of motion between $\pm\hbar$. The quantum result showed an interference distribution that was heavily peaked about the initial axis of motion which differed from the classical result. For future work we are anticipating simulating the experiment by using the Crank-Nicolson method.

Code is available under the CC0 license.