Modelling a Damped Static Pendulum Using Different Numerical Integration Methods

Jedidiah Buck McCready

SSU Physics Capstone

Outline

- What is it?
- Why model it?
- How is it modelled?
- Implementations
- Results
- What I learned in the process

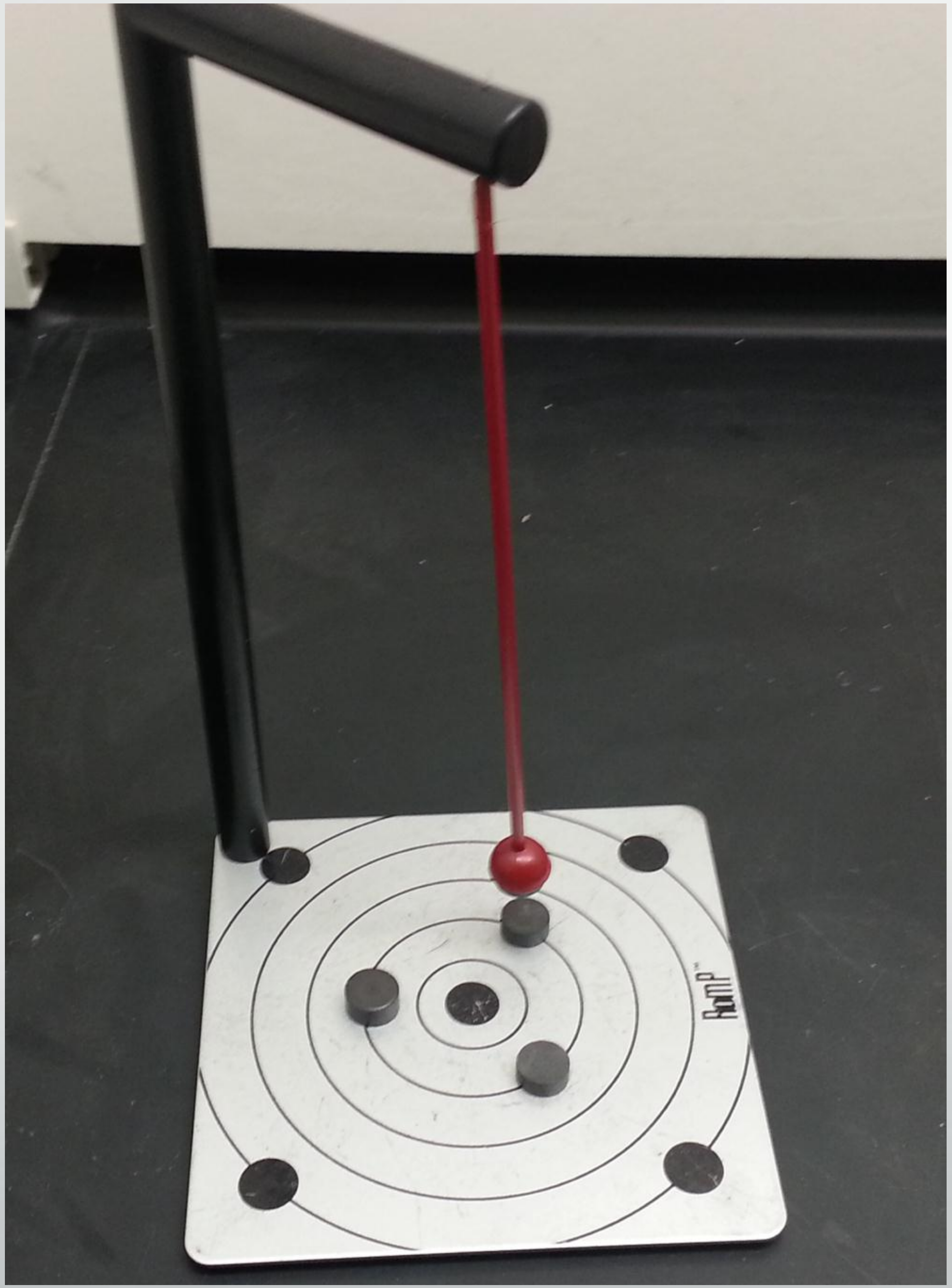
What is it?

- Pendulum that can freely move in x-y directions but is bound in the z direction by the pendulum length
- Three distinct forces: gravity, attractors, and air resistance (dampening)
- Attractors treated as fixed points that exert a force on the pendulum head proportional to $1/\text{distance}^2$
- Modeled using differential equations and numerical integration methods

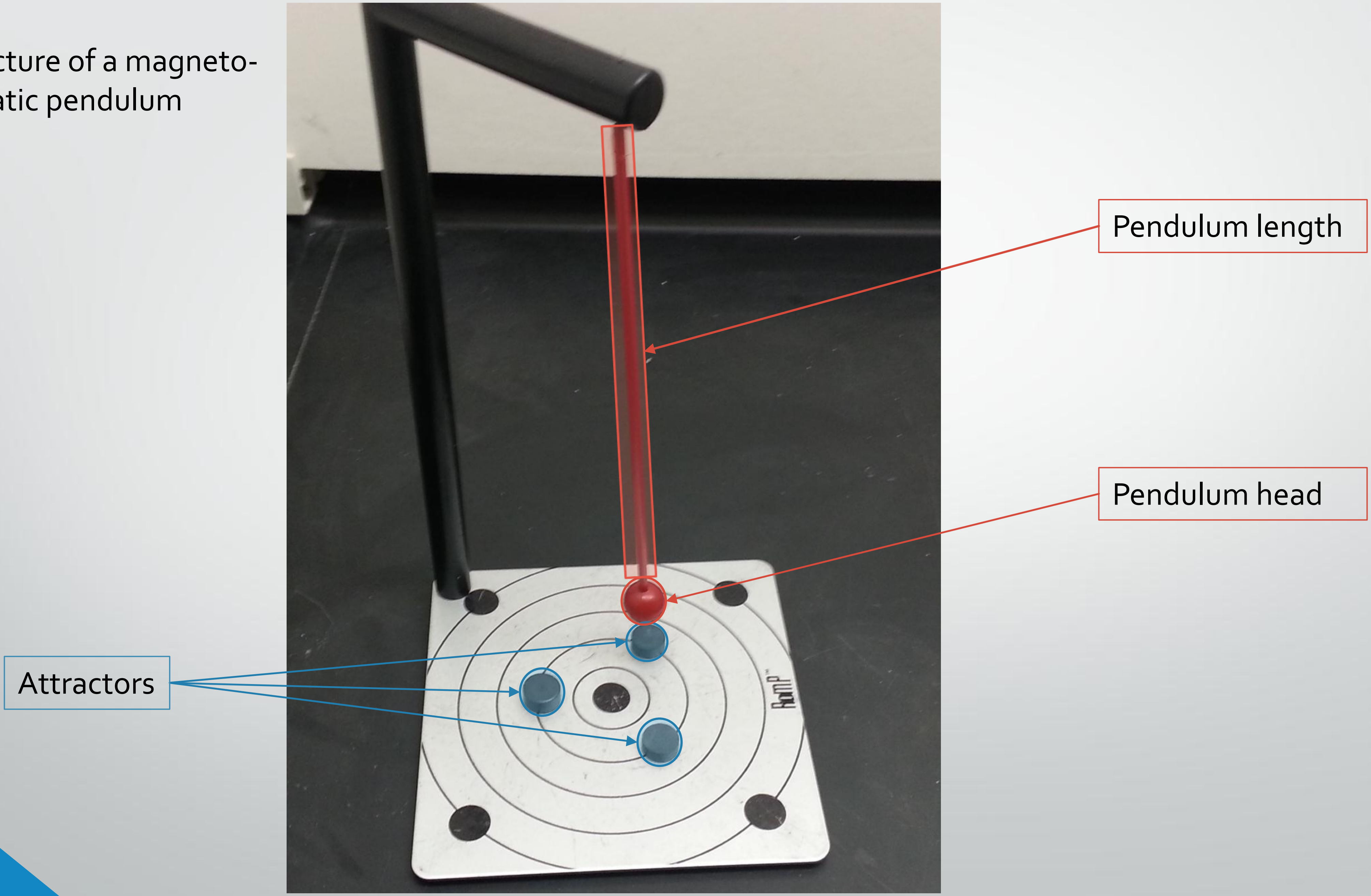
Why model it?

- The problem is relatively generic: the coding and mathematical knowledge required is applicable to many other problems
- The problem is computationally intensive making it interesting to optimize both numerically (underlying mathematics) and programmatically (code)
- The problem was the final project for Physics 381 when I took the class but the implementation was computationally slow (it took hours to compute a single image) and was impractical for system analysis
- A chaotic fractal pattern emerges from the system (it creates pretty images)

Picture of a magneto-static pendulum



Picture of a magneto-static pendulum



The differential equations

$$\vec{F}_g = -mg\sin\theta\cos\theta\hat{r} \tag{1}$$

$$\vec{F}_g = -mg\frac{\sqrt{1-\frac{x^2+y^2}{L^2}}}{L}(x\hat{i}+y\hat{j}) \tag{2}$$

$$\vec{F}_{m_n} = -\frac{k_1\vec{r}_n}{r_n^3} \tag{3}$$

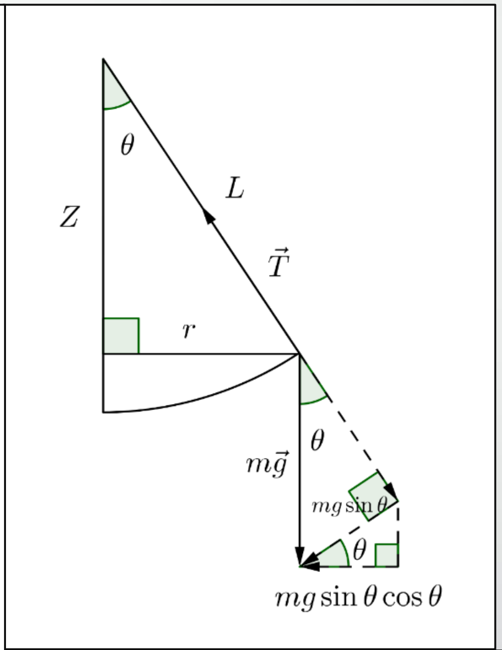
$$\vec{F}_{m_n(x_n,y_n)} = \frac{-k[(x-x_n)\hat{i}+(y-y_n)\hat{j}]}{\left[(x-x_n)^2+(y-y_n)^2+\left(d+L-\sqrt{L^2-(x^2+y^2)}\right)\right]^{3/2}} \tag{4}$$

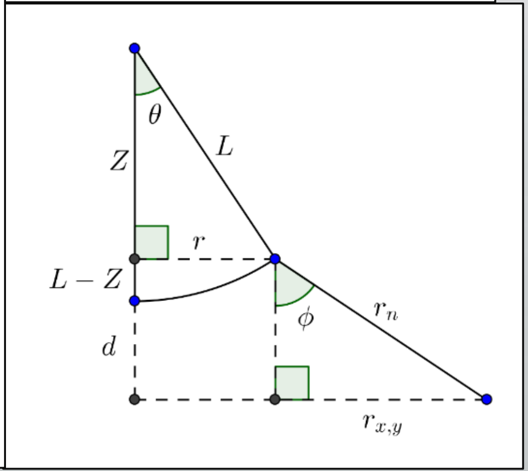
$$\vec{F}_d = -b(v_x\hat{i}+v_y\hat{j}) \tag{5}$$

$$\vec{F}_{total} = m\vec{a} \tag{6}$$

$$a_x = \frac{d^2x}{dt^2} = \left(\frac{F_{gx}+F_{m_1x}+F_{m_2x}+F_{m_3x}+F_{dx}}{m}\right)\hat{i} \tag{7}$$

$$a_y = \frac{d^2y}{dt^2} = \left(\frac{F_{gy}+F_{m_1y}+F_{m_2y}+F_{m_3y}+F_{dy}}{m}\right)\hat{j} \tag{8}$$





Adaptive Runge-Kutta method

$$\dot{y} = f(t, y), \quad y(t_0) = y_0$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i k_i$$

$$t_{n+1} = t_n + h$$

$$k_s = hf(t_n + c_s h, y_n + a_{s1}k_1 + a_{s2}k_2 + \cdots + a_{s,s-1}k_{s-1})$$

$$e_{n+1} = y_{n+1} - y^*_{n+1} = \sum_{i=1}^{s} (b_i - b^*_i) k_i$$

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & & \ddots & & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s \\
 & b^*_1 & b^*_2 & \cdots & b^*_{s-1} & b^*_s
\end{array}$$

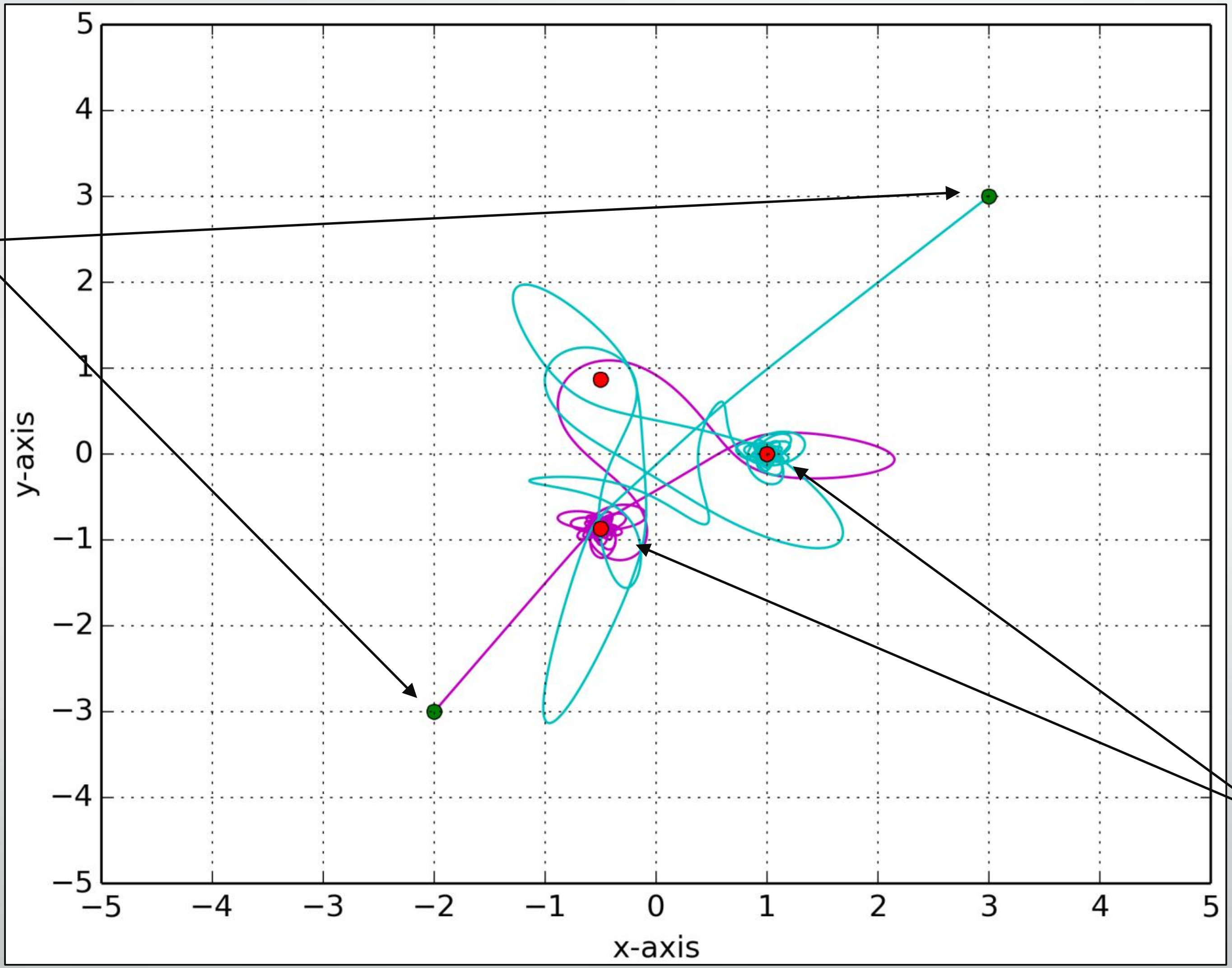
0						
1/5	1/5					
3/10	3/40	9/40				
3/5	3/10	-9/10	6/5			
1	-11/54	5/2	-70/27	35/27		
7/8	1631/55296	175/512	575/13824	44275/110592	253/4096	
	37/378	0	250/621	125/594	0	512/1771
	2825/27648	0	18575/48384	13525/55296	277/14336	1/4

Kash and Carp 5(4) Butcher Tableau

Plotting the system

1. Release the pendulum from a starting position (initial conditions)
2. Pendulum swings according to all the forces acting upon it (attractors, gravity, and air resistance)
3. Pendulum eventually converges and comes to rest at an attractor

Trajectory Paths



Starting points (-2,-3) and (3,3)

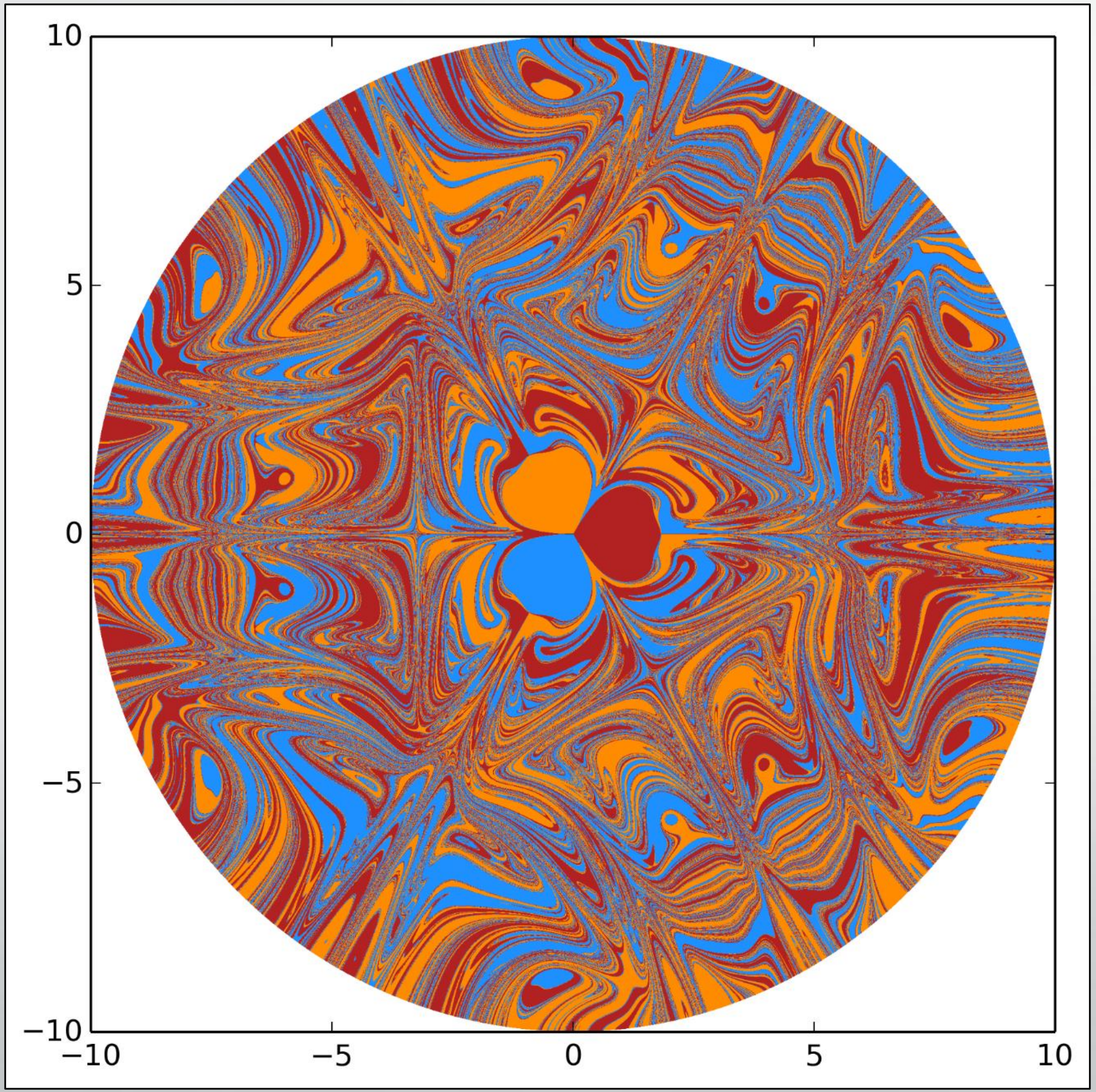
Convergence points

X-Y plane represents the base plate under which the pendulum swings. The red circles represent attractor positions.

Plotting a map of starting positions

1. Find the convergence point for a set of points equally spaced across the entire x-y plane (bounded only by the length of the pendulum)
2. Assign a color to each starting position according to the attractor it converges to
3. Decrease the spacing between starting points to form a more continuous image (increase resolution)

Map of starting point convergence



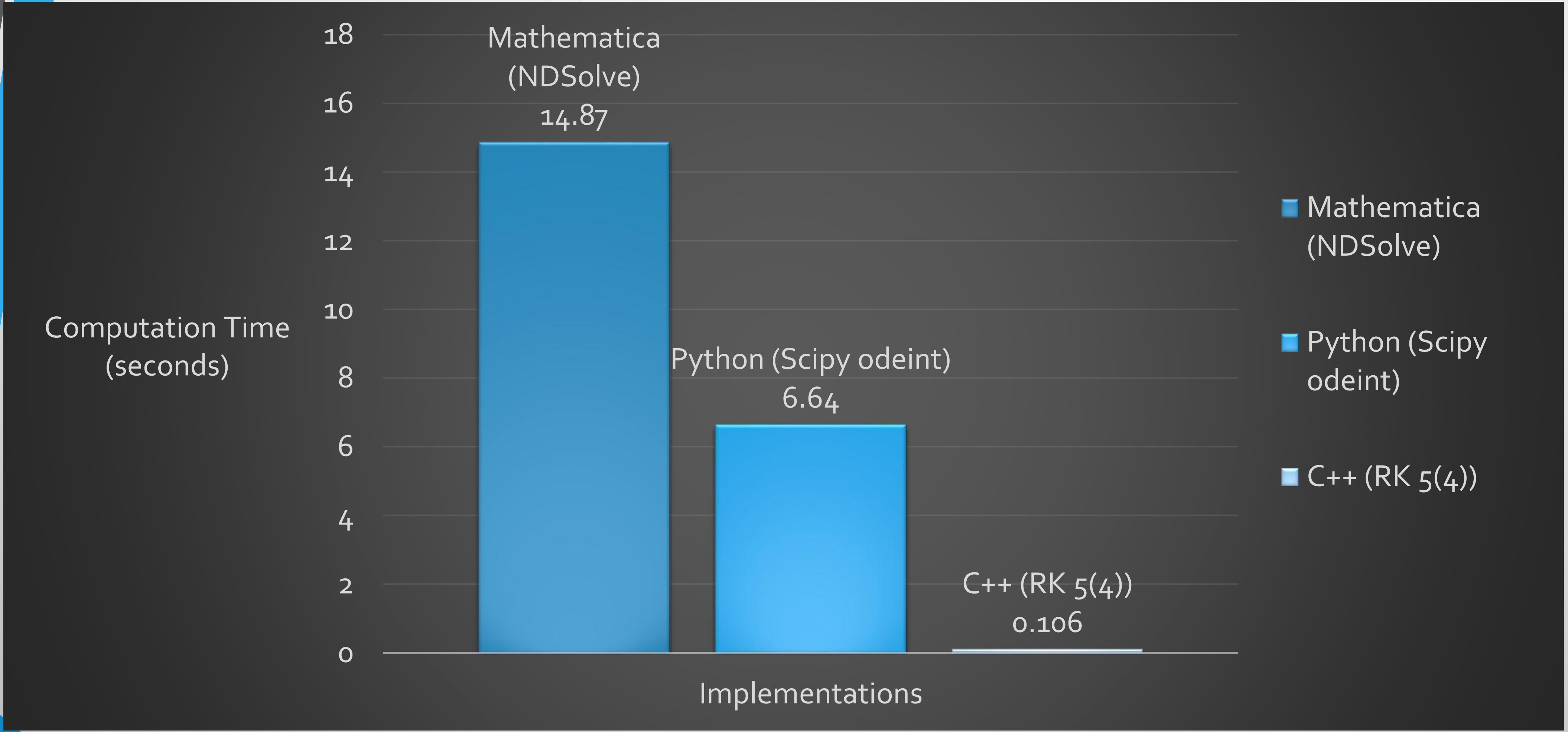
X-Y plane of many starting points colored according to the attractor they converge to.

The white is outside the bounds of the pendulum length (for this map it is 10 unit lengths).

Implementations

- Mathematica using “NDSolve,” this is the method that was implemented in Physics 381 (utilizes primarily the adaptive multistep LSODA method)
- Python with open source science and math libraries “NumPy” and “SciPy,” and using “numba” to accelerate the code with clang LLVM (also utilizes the LSODA method)
- C++11/14 written from the ground up using an embedded adaptive step size Runge-Kutta 5(4) method developed by Cash and Karp

Why three different implementations?



Computation times for a sample map of 841 starting points (29x29 pixel image) on an i7-3630QM at 2.4Ghz.

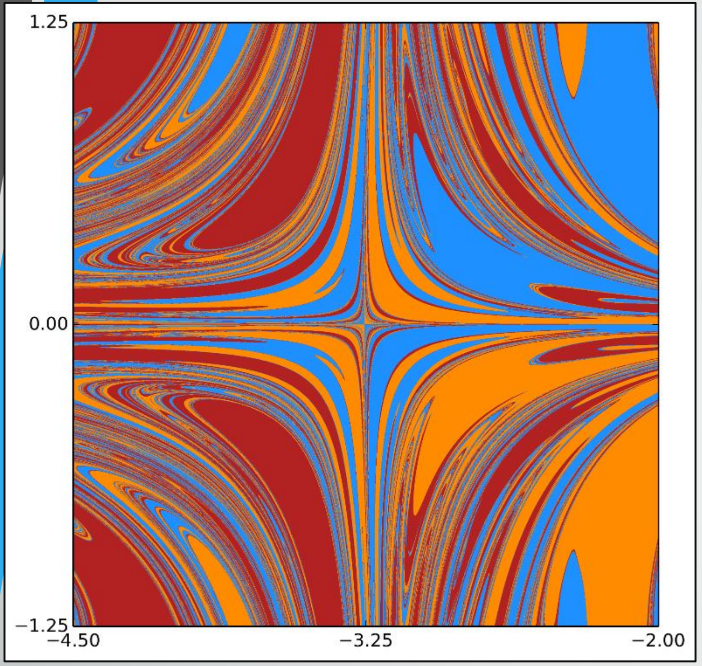
Why computation time matters

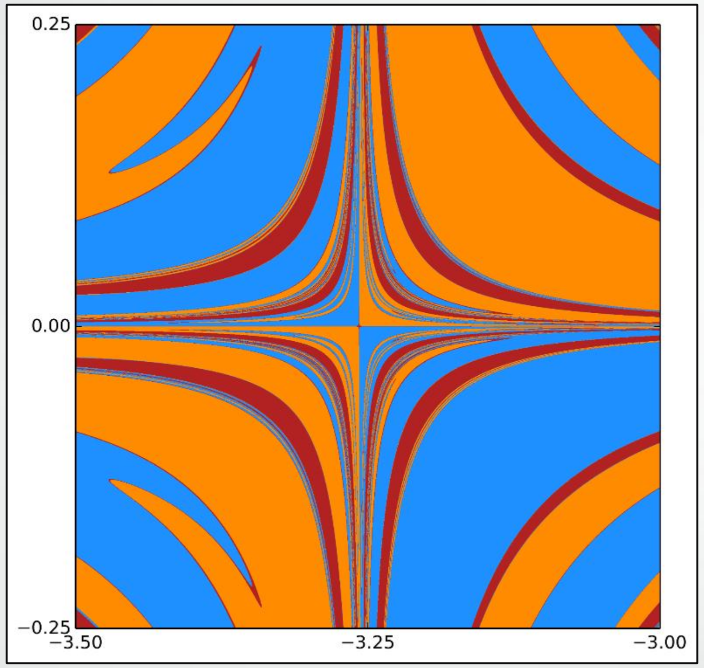
- Constructing many high resolution maps with varying system parameters (air resistance, attractor strength, gravitational strength, etc.) is useful for profiling the system
- An 800x800 pixel map takes about 2 hours and 30 minutes to integrate using Mathematica NDSolve
- This same 800x800 pixel map takes about 1 minute using the C++ implementation

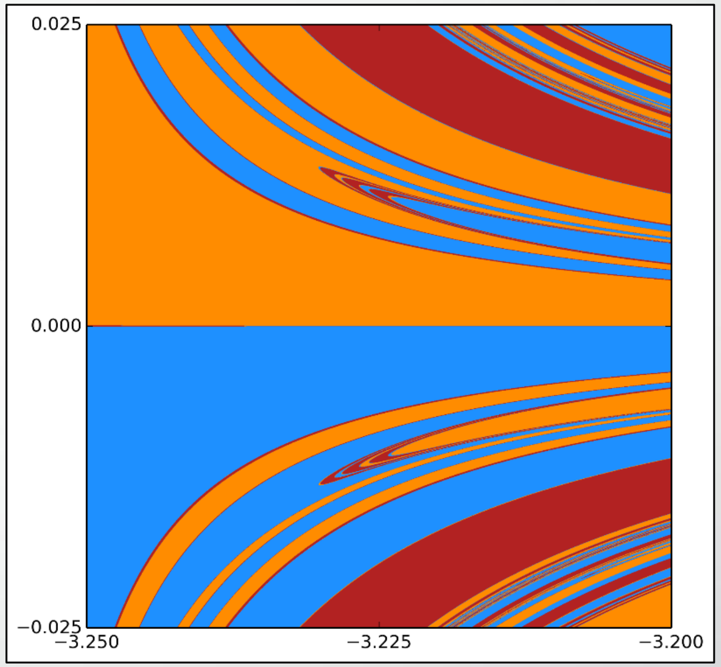
What about precision and numerical stability?

1. A fairly high resolution map (800x800 or 640,000 points) was compared between Mathematica and the C++ RK 5(4) implementation
2. About 1500 points disagreed between maps (~0.23% of the total points)
3. Reran the integration on the points that disagreed in Mathematica with a more strict error control
4. Only about 25 points disagreed (~0.0039% of the total points)
5. Reran the C++ implementation with a more strict error control
6. No points disagreed

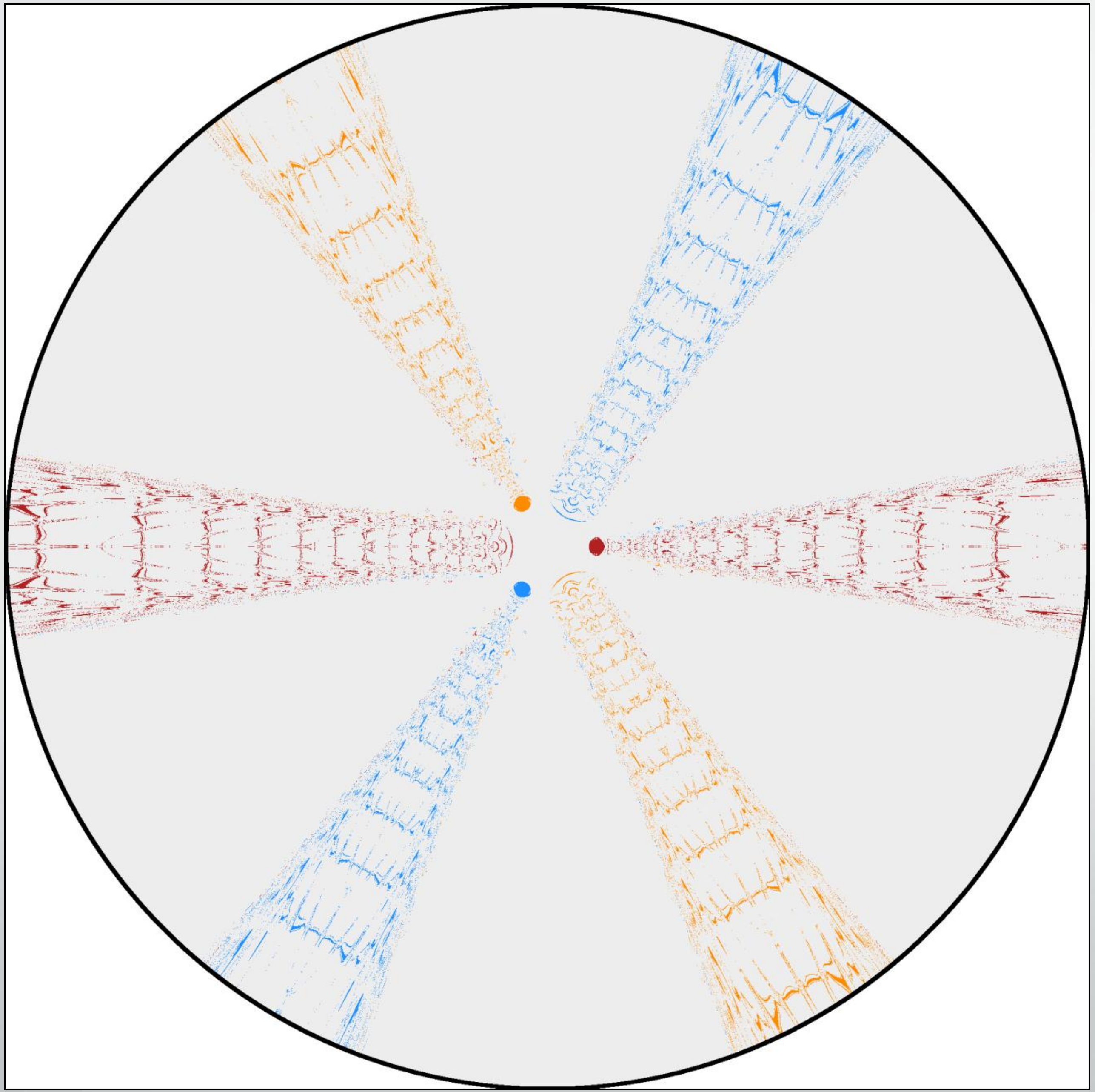
Zooming in shows fractal type image







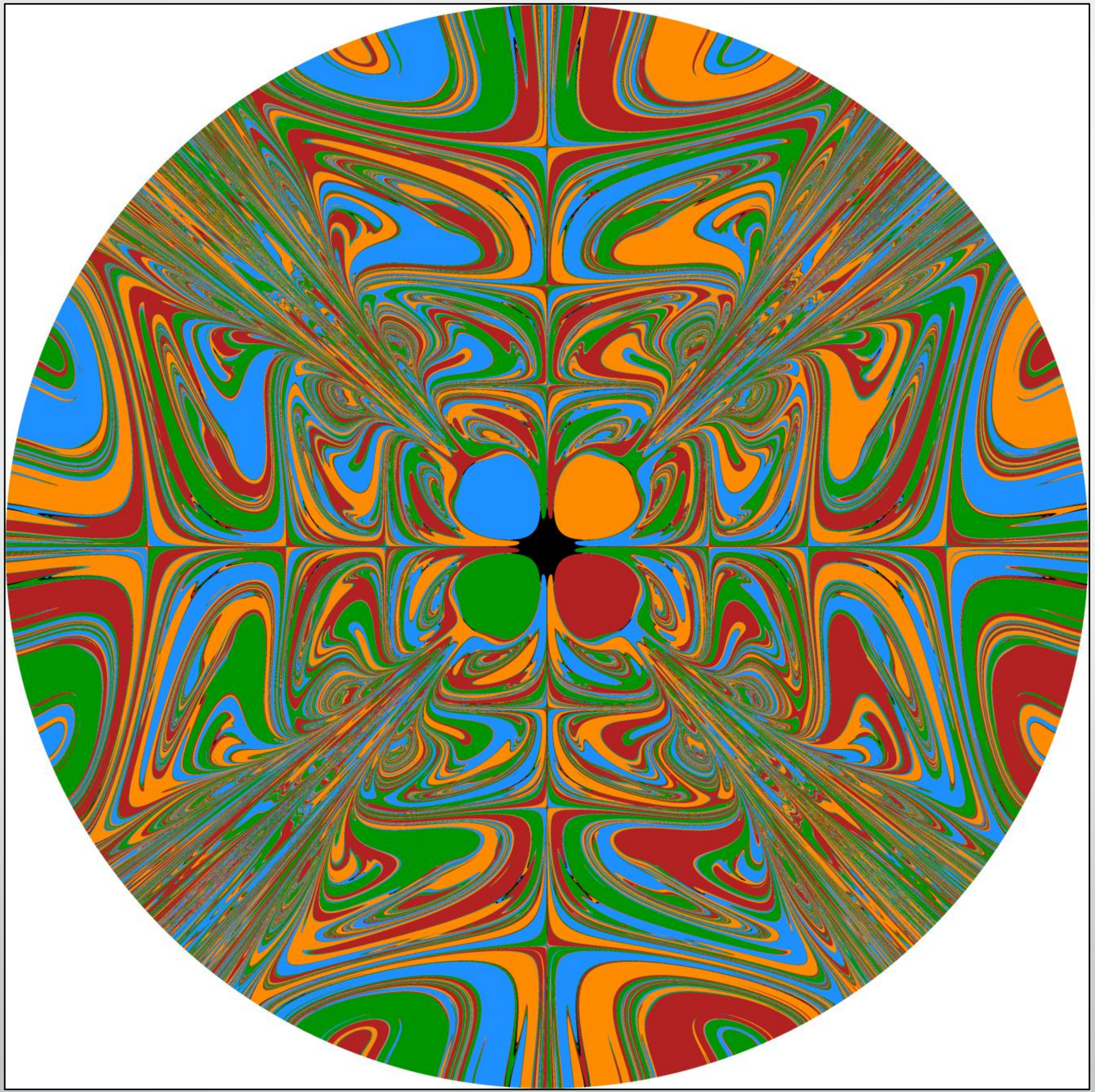
Zooming in shows the repeating pattern within the boundaries between basins. The pattern becomes increasingly sparse, and requires greater and greater magnification to resolve.



Same system as shown previously but with about 20 times stronger gravitational force.

The light grey represents a convergence to the origin.

The region outside the circle is outside the bounds of the pendulum length.



A system with 4 attractors placed in a square formation about the origin.

The black represents convergence to the origin.

Animation of parameter changes

- By integrating many maps with a slightly changing parameter we can stich the images together to form an animated video
- Five hundred 800x800 maps were integrated to form an animation (250 million points integrated)
- It would take weeks of continuous computation using Mathematica to integrate this many starting points on an i7-3630QM at 2.4Ghz; it took approximately 5 hours using the C++ implementation
- The animation starts with weak dampening (b=0.1, e.g. low density gas) and gradually increases to very high dampening (b=0.5, e.g. pendulum suspended in a viscous liquid)

The code is open source

- C++ implementation is freely available on GitHub at https://github.com/jbuckmccready/staticpendulum
- Code is completely modular and object oriented; the integrator, system and map are separate templated class objects
- For example to add another attractor to the system it is as simple as "mySystemObj.add_attractor(x_position, y_position, attraction_strength)"
- Easy to add more integration methods and other systems of differential equations

Concluding remarks

- I learned about differential equations, numerical integration methods, programming in different languages, and optimization techniques
- Successfully decreased the computation times by more than two orders of magnitude
- Utilized the improved computation times to create animations that give a visual understanding of how the system behaves
- Code can be expanded by adding statistical analysis tools, new physical systems, and more integration methods
- A group of physicists recently published (November 2013) a paper that quantitatively categorizes these damped chaotic systems as undergoing "doubly transient chaos"

References and acknowledgements

J. R. Cash, A. H. Karp. "A Variable Order Runge-Kutta Method for Initial Value Problems with Rapidly Varying Right-Hand Sides." ACM Transactions on Mathematical Software, Vol. 16, No. 3, 1990.

L. F. Shampine, A. Witt. "Control of Local Error Stabilizes Integrations." Computational and Applied Mathematics, Vol. 62, No. 3, 1995.

A. E. Motter, M. Grulz, et al. "Doubly Transient Chaos: Generic Form of Chaos in Autonomous Dissipative Systems." Phys Rev. Lett. 111, 194101.

Thanks to:

My advisor Hongtao Shi for giving me feedback on organizing a short presentation

Hunter Mills for the Geogebra free body diagrams

Michieal Jones for getting me originally interested in the project from Physics 381

Questions?