Computational Physics Project: Lyapunov Exponents

Thomas Hollingsworth, Corpus Christi College

May 6, 2024

Abstract

Algorithms for calculating the largest lyapunov exponent and spectrum of lyapunov exponents for discrete and continuous time dynamical systems were implemented in Python and used to analyse the logistic map, the standard map and the Lorenz system. The implementation of each algorithm is detailed and justified accompanied by comments on the complexity and relevant computational physics. The largest lyapunov exponent of the Lorenz system (with standard chaotic parameters) was calculated as 0.899 ± 0.002 and the LLE for the logistic map with parameter $r = 3.7$ was calculated as 0.355 ± 0.001. The algorithm for determining the largest lyapunov exponent for the Lorenz system was shown to decrease in error as approximately $t^{\frac{1}{2}}$.

word count: 2993

1 Introduction

Lyapunov exponents (LEs) are a global property of dynamical systems that describe the exponential convergence or divergence of infinitesimally separated trajectories in phase space, both discrete and continuous time systems have an associated spectrum of LEs, one for each phase space dimension. In addition to uses in the calculation of quantities such as the Kaplan-Yorke dimension [4] and Kolmogorov-Sinai entropy [2], the LE spectrum can indicate and characterise deterministic chaos.

Deterministic chaos is observed in a variety of systems, from the simple double pendulum [8], to financial modelling [6],and heart arrythmia [3]. The associated phase spaces similarly span a vast range, from basic models that deal with single digit dimensions to weather forecasting simulations with $\mathcal{O}(10^9)$ dimensions. Consequently, developing and improving methods to accurately determine LE spectra has a broad impact.

In this paper, computational methods for determining the Largest Lyapunov Exponent (LLE) and the Lyapunov Spectrum were investigated and optimised. Section 2 gives a general theoretical background on phase space and lyapunov exponents. Sections 3,4 and 5 each cover: key theory and computational physics, algorithm implementation, and results and analysis, for the Logistic Map (1D discrete), Standard Map (2D discrete) and the Lorenz System (3D continuous). Section 6 gives a conclusion, and details of software, hardware and code structure are given in the appendices.

2 General Theoretical Background

The complete state of a dynamical system at any time can be fully represented by a point in phase space $\mathbf{x}$, and the evolution of the system is represented by a phase space trajectory $\mathbf{x}(t)$, calculated from the relevant equations of motion.

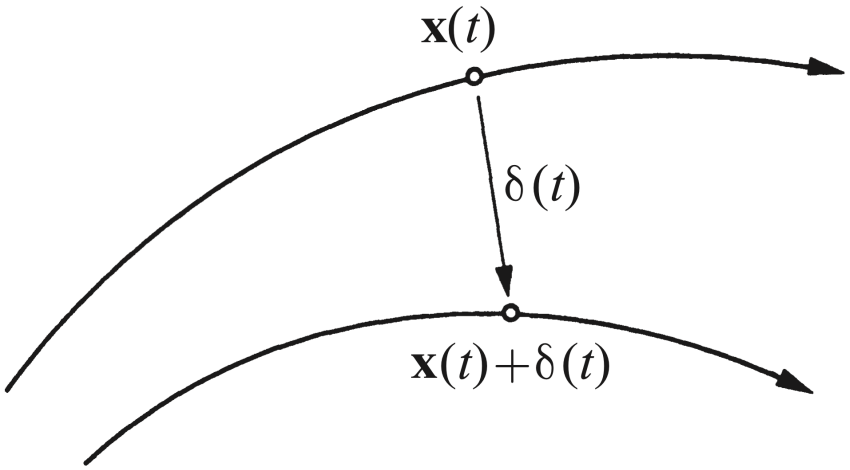

Figure 1: Diagram of nearby phase space trajectories **ref fun book**

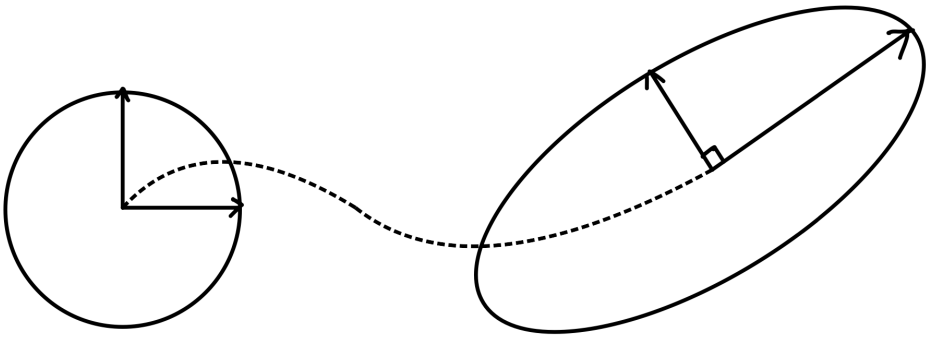

Figure 2: Initial n sphere of nearby trajectories distorts to an n ellipsoid

Lyapunov exponents describe the evolution of infinitesimal perturbations to a trajectory in phase space, denoted in Figure 1 by $\delta(t)$. For an n dimensional phase space, the spectra of n LEs, labelled λ_i, describe the long-term evolution of an infinitesimal n-sphere of perturbations into an n-ellipsoid, with principal axes that asymptotically grow as e^{λ_i} respectively [11].

As $t \to \infty$, the behaviour of all perturbations will be dominated by the (LLE) largest lyapunov exponent, λ_1.

$$|\delta(t)| \approx e^{\lambda_1 t}|\delta(0)| \tag{1}$$

Chaotic systems have a positive LLE and are therefore highly sensitive to initial conditions, with small differences growing exponentially.

The equations of motion for chaotic systems contain non-linearities and are non-integrable [5], consequently, numerical methods must be used to solve them. In practice, any numerical method will introduce error, even at a fundamental level the finite representation of floating points introduces error. The positive LLE in chaotic systems means these errors grow exponentially with time, making accurate numerical integration of the systems extremely challenging.

3 Logistic Map

3.1 Theory and Analysis of Computational Physics

The logistic map is a 1D discrete time map governed by the equation:

$$x_{n+1} = F(x_n) = rx_n(1 - x_n) \tag{2}$$

Where x_n is a number between 0 and 1. The initial condition x_0 only gives a transient contribution to the dynamics and the long term behaviour is completely determined by the single parameter r, analogous to trajectories converging on an attractor in phase space. Varying r gives rise to a variety of different behaviours, which will be examined in the following sections.

One suggested algorithm [9]for calculating the single LE of this system is as follows:

- Choose an initial point and iterate it until any transient behaviour has sufficiently decayed, label this x_0
- Choose a point very close to x_0, $x_0' = x_0 + \delta_0$.
- Iterate these two points, n times using the logistic map equation.
- Calculate $\frac{|x_n' - x_n|}{|x_0' - x_0|} = \frac{|\delta_n|}{|\delta_0|}$
- Use the discrete analogue of Eqn. 1, $\lambda_1 \approx \frac{1}{n} \ln(\frac{|\delta_n|}{|\delta_0|})$ and the above result to estimate λ_1

The accuracy of this algorithm improves as δ_0 decreases and n increases. In order to continually satisfy the small δ_n limit, this algorithm must be modified to periodically move x' closer to x.

Since the LLE dominates long term behaviour of any perturbation, this algorithm can be used to calculate the LLE of systems of any dimension, provided that x' is moved closer to x along the line joining them. For a system of dimension N, applying the map to all $2N$ coordinates gives a complexity of $\mathcal{O}(N^2)$, adjusting x' is a $\mathcal{O}(N)$ operation. Therefore, this algorithm scales with $\mathcal{O}(N^2 n)$ for dimension N and iterations n.

A simple but significant improvement to this method can be found by considering the dynamics of the perturbation itself rather than the two x values.

$$\delta_{n+1} = x_{n+1}' - x_{n+1} = F(x_n + \delta_n) - F(x_n) \tag{3}$$

Now taking the first order Taylor expansion, corresponding to $\delta_n \to 0$.

$$\delta_{n+1} = F'(x_n)\delta_n = r(1 - 2x_n)\delta_n \tag{4}$$

This new equation can be used to estimate λ_1. As with the previous algorithm, this easily extends to higher dimensional problems, where $F'(x_n)$ would be replaced by the Jacobian of F, an $N \times N$ matrix, and $\delta(n)$ would be an N dimensional vector. This approach would require iterating δ_n and x_n leading to the same $\mathcal{O}(N^2 n)$ scaling as the previous algorithm. The benefit of this approach is that the infinitesimal perturbation limit is built in, giving more accurate results than those from any finite perturbation and not requiring constant readjustment to maintain, leading to improved computational speed.

3.2 Implementation of Algorithm

The implemented algorithm mostly followed the structure above with some extra considerations to improve performance. The most prominent of these was the choice to construct the algorithm to simultaneously run any number of repeat trials, each with their own initial conditions and r values. This required replacing x_0 and r with numpy nd.arrays of size (m) where m is the number of trials to run. Doing this allowed the code to leverage numpys vectorisation and as a result the Lyapunov exponents for a multitude of different x_0 and r could be determined significantly faster than if they were calculated individually,one-by-one.

Another major difference is in the final step of calculating λ_1.Some simple manipulation shows:

$$\delta_{n+1} = F'(x_n)F'(x_{n-1})...F(x_0)\delta_0 \tag{5}$$

$$\ln|\delta_{n+1}| = \sum_{i=0}^{n} \ln|F'(x_i)| + \ln|\delta_0| \tag{6}$$

$$\lambda_1 = \lim_{n\to\infty} \frac{1}{n} \ln \frac{|\delta_{n+1}|}{|\delta_0|} = \lim_{n\to\infty} \langle \ln|F'(x)| \rangle \tag{7}$$

Where $F'(x)$ is averaged across all (post transient) iterations. This equation for λ_1 was used as it more effectively leveraged numpy vectorisation. The implementation was as follows:

- User provides an nd.array of the m x_0 and r values to be investigated, along with the number of iterations for the transient to decay, n_T, and the number of iterations to perform post-transient (n)..
- The logistic map is performed simultaneously on all initial conditions, the x values obtained are added to an array of size $(m, n + n_T)$, this is repeated $(n + n_T)$ times.
- The array of x values is indexed to size (m, n) to remove the transient.
- Numpy vectorised methods are used to generate an (m, n) array of $F'(x)$ values: $F'(x) = r(1 - 2x)$.
- The absolute value and natural log of this array is taken before taking the mean along the rows, again utilising numpy vectorised methods.
- The final array has dimension (m) and contains the estimates of the Lyapunov exponent for each run
- Option included to calculate the mean after each iteration to return an (m, n) array of the LE estimates for each sample after each iteration.

This algorithm was able to perform 1,000,000 post transient iterations for a single sample in around 2.15s. The major benefits of vectorisation are apparent as the algorithm could perform 100 samples of 1,000,000 iterations in a time of just 6.1s, saving significantly over having to re-run the algorithm. This algorithm still has complexity $\mathcal{O}(N^2 n)$ as $n \to \infty$, therefore, this result shows that, even for large n, most of the 2.15s processing time is attributed to overhead rather than the core algorithm.

3.3 Results and Analysis

An easy way to visually check that the algorithm is giving reasonable results is the construction of a bifurcation diagram [10]paired with a plot of λ_1 against r for λ_1 values calculated by the algorithm, shown in Figure 3.

A bifurcation diagram is a plot of the 'equilibrium' values reached by the map as a function of the parameter r. For $0.5 < r < 3$ the diagram is a single line, indicating that all trajectories converge to a

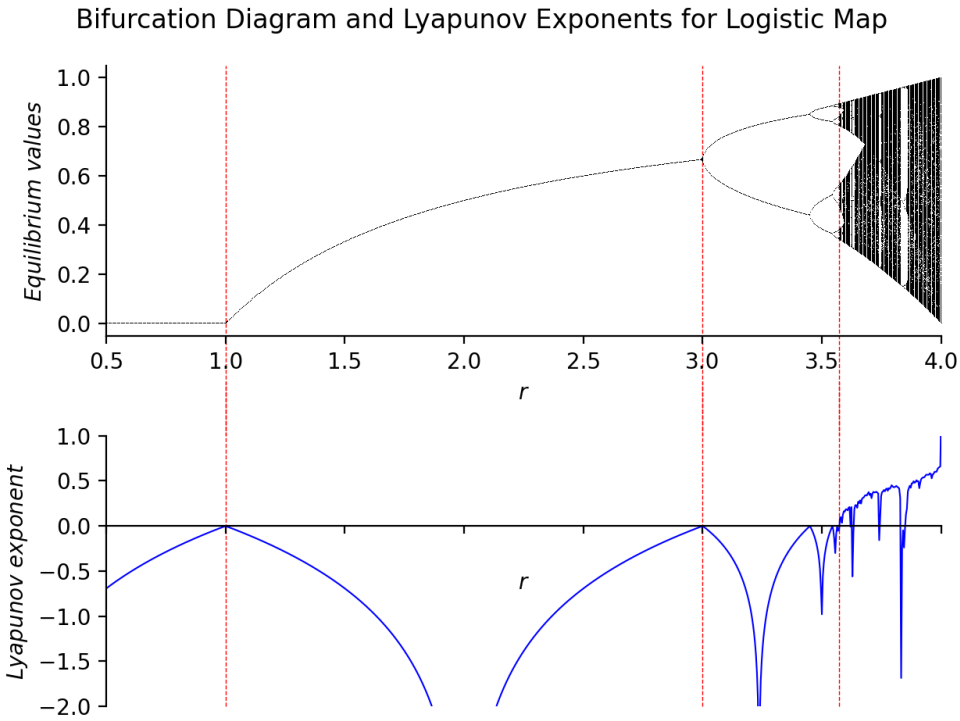

Figure 3: Bifurcation diagram and LLE Estimates

single point.For $3 < r < 3.5$ the diagram forks, indicating that trajectories converge on a periodic sequence of values. As expected, these regions correspond to a negative LE due to the convergence of trajectories. Beyond the value $r = 3.56995$, which is marked on the figure, the Lyapunov exponent is mostly positive with occasional negative spikes, this pairs well with the bifurcation diagram which shows chaotic behaviour interspersed with regions of periodicity.

These results show that the LLE algorithm gives qualitatively accurate values. As there are not well documented values for the LE of the logistic map to compare to, the quantitative accuracy of the algorithm will have to be assessed by looking at its convergence with iterations and robustness to variations in initial conditions.

The top plot in Figure 4 shows the LLE estimates after each iteration of the algorithm for a variety of runs with different initial conditions and (chaotic) $r = 3.7$. The lower plot shows the difference between subsequent estimates. It is clear that even after a small number of iterations the LE estimates begin to converge on a positive value independent of their initial conditions. For Figure 5, 1000 trials were performed with random x_0 and $r = 3.7$ and $n = 100,000$ iterations. Even after a relatively small number of iterations the algorithm produces results that are robust to different x_0 giving: $\lambda_1 = 0.355 \pm 0.001$.

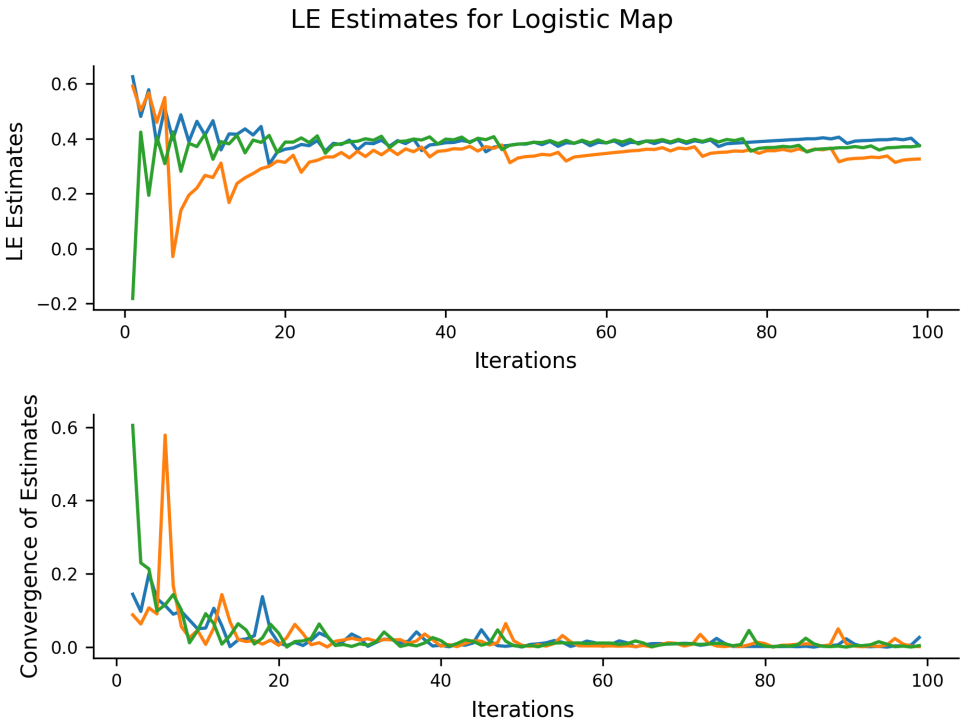

Figure 4: LLE estimates converging with iteration

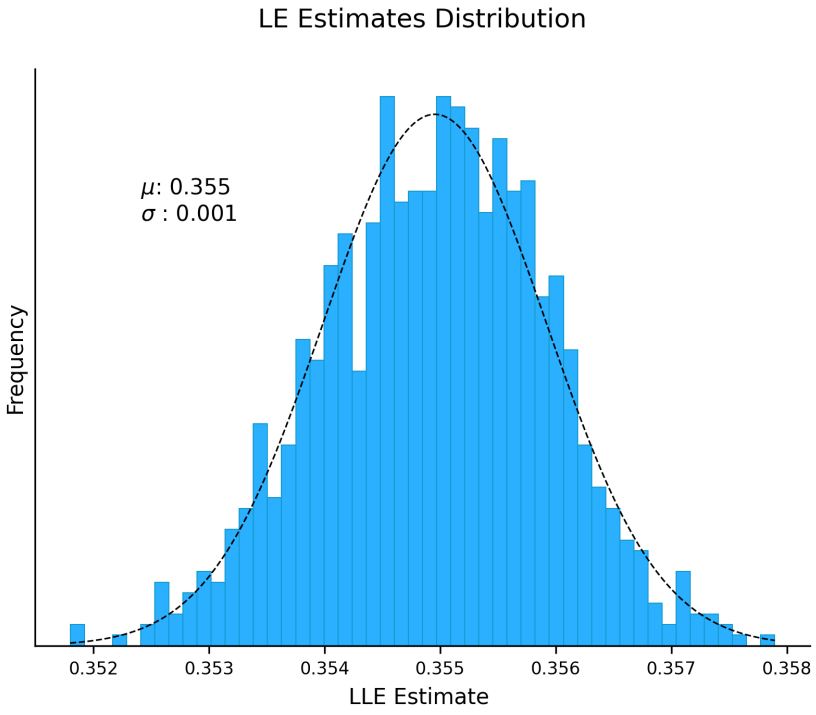

Figure 5: Variation of LLE estimates with initial condition

4 Standard Map

4.1 Theory and Analysis of Computational Physics

The standard map is a 2D discrete time map governed by the equations:

$$p_{n+1} = p_n + K\sin(\theta_n) \mod 2\pi,$$
$$\theta_{n+1} = \theta_n + p_{n+1} \mod 2\pi.$$

The standard map describes the Poincaré's surface of section for the 'kicked rotator', and provides a simple approximation to a variety of processes such as accelerator physics. For the purposes of this investigation, the most useful property of the standard map is that it is a Hamiltonian system and has a symplectic geometry.

For symplectic systems, phase space volumes are conserved, this has some useful consequences for the LEs of the standard map:

1. Total phase space volume is constant, therefore $\sum_i \lambda_i = 0$
2. LEs appear in symmetric pairs: $\lambda_k = -\lambda_{N+1-k}$ (for phase space dimension N). In this case $\lambda_1 = -\lambda_2$ [7].
3. There are no attractors in the phase space, consequently there are no transients.

These results can be used to help assess the accuracy of calculated LE values.

The algorithm used for the logistic map can be extended to higher-dimensional problems but can only calculate the largest LE and must be modified to allow calculation of the complete LE specturm. Calculating the LLE involves monitoring the exponential growth of a single perturbation, whereas determining the M largest Lyapunov exponents of a system requires calculating the growth of an infinitesimal MD subspace [7].

Similar to the previous LLE algorithm, the first step is to calculate a trajectory of $\vec{x_n}$ values by iterating an initial $\vec{x_0}$. These values can then be used to calculate the Jacobian at each iteration:

$$J_n = \begin{pmatrix} \frac{\partial p_{n+1}}{\partial p_n} & \frac{\partial p_{n+1}}{\partial \theta_n} \\ \frac{\partial \theta_{n+1}}{\partial \theta_n} & \frac{\partial \theta_{n+1}}{\partial p_n} \end{pmatrix} = \begin{pmatrix} 1 & K\cos(\theta_n) \\ 1 & 1 + K\cos(\theta_n) \end{pmatrix}$$

The single displacement vector of the LLE algorithm δ_0 is replaced by an orthogonal matrix Q_0 that represents a 2D subspace of perturbations.

This now involves a complete matrix multiplication at each iteration, giving a typical complexity of $\mathcal{O}(N^3)$ for dimension N.

The 2D subspace, which starts as a square, rapidly distorts to a parallelogram with an area that tends to 0 as both axes align with the direction of largest growth, this corresponds to the Q matrix becoming increasingly singular as a result of repetitive multiplications with the Jacobian matrices. To overcome this, the matrix Q can be periodically re-orthogonalised. This process , called QR decomposition deconstructs the matrix into an orthogonal matrix which will be labelled Q_1 and a right triangular matrix R_1 . This process is showed in Fig6. With reference to Fig6, the lyapunov exponents are calculated from the exponential growth of the projection of the red vectors on the re-orthogonalised blue vectors, which can be found from the diagonal elements of the R matrix.

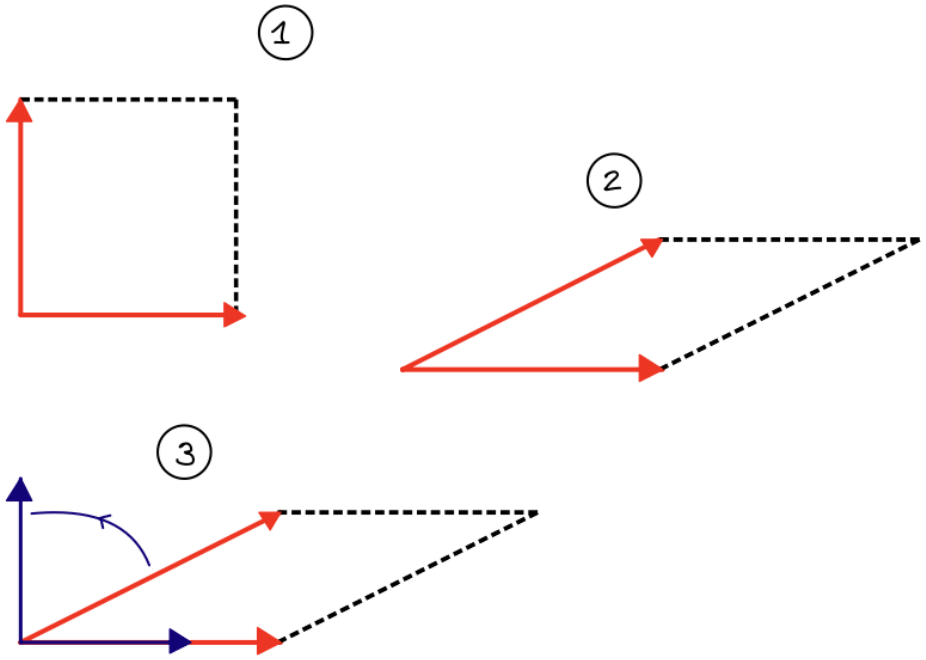

Figure 6: Depicting the evolution of the Q perturbation matrix

4.2 Implementation of Algorithm

This algorithm was also designed to run multiple samples simultaneously to take advantage of numpy vectorisation, the basic structure of the algorithm can be summarised as:

1. Simultaneously iterate all samples n times, storing the values $\vec{x_n}$ in a large array.
2. Calculate the J_n matrices at each iteration for all samples.
3. Split each trajectory of J matrices into steps with n_step matrices per step.
4. Calculate the product of the n_step matrices in each step to give matrices: S_1, S_2 etc.
5. For each sample, generate a random orthogonal matrix Q_0 to represent the initial perturbation subspace.
6. Pre-multiply Q_0 by S_1 then perform QR decomposition to get Q_1 and R_1.
7. Use the diagonal elements of the R_1 to calculate an estimate for the Lyapunov exponents.
8. Take the new Q_1 matrix and pre-multiply it by S_2.
9. Continue to iterate the previous 3 lines for all steps.
10. Take the average of the estimates for the Lyapunov exponents calculated at each step.

The key areas where the complexity of this algorithm deviates from the previous is in the multiplication of a series of matrices and the QR decomposition, these were investigated in more depth.

4.2.1 Matrix Multiplication

Matrix multiplication is typically an order $\mathcal{O}(N^3)$ process for square matrices of dimension N. The two easiest methods for performing a series of square matrix multiplications were looping the use of np.dot and using scipy.linalg.multidot. Multidot is a detailed function that takes advantage of the associativity of matrix multiplication to find the most efficient way to multiply a list of matrices together. For large matrices, long lists of matrices and matrices with different sizes, multidot is preferable to looping np.dot(). However, for this particular case, the simplicity of square matrices and the small number of matrices per step meant that the significant overhead required for multidot made it more time intensive. A rough time test showed that multidot only began to outperform the simple loop once the dimension of the problem reached 1000. As a further note, packages such as PyTorch and TensorFlow can significantly improve the efficiency of matrix multiplication by utilising the parallel processing power of GPUs [1], this would be an important consideration for massive systems such as weather prediction.

4.2.2 QR Decomposition

QR decomposition is also an order $\mathcal{O}(N^3)$ process. To optimise this process the speed and accuracy of a variety of algorithms were compared for different matrix sizes, Figures 7 and 8

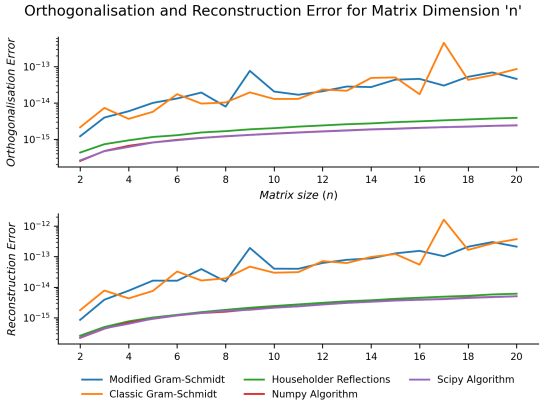

Figure 7: Error for different QR algorithms

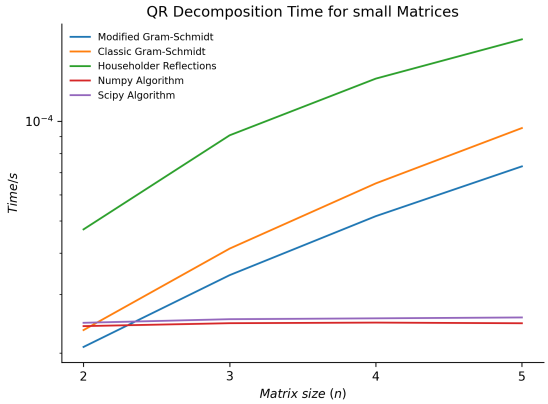

Figure 8: Runtime for different QR algorithms

It is clear that the scipy and numpy QR algorithms perform the best. The overhead associated with these methods mean they lose out in runtime to the more simplistic Gram-Schmidt algorithms for 2×2 matrices; however, they are far faster for larger matrices and have a significantly smaller error. A brief review of the scipy source code reveals that the QR decomposition runs an LAPACK code based on the Householder reflection method.

4.3 Results and Analysis

To interpret the results of the LE algorithm the phase space of the standard map was plotted for a variety of K parameters, shown in Figure 9.

For K=0.5, the majority of trajectories are periodic or quasi-periodic, corresponding to loops or fractal chain-like patterns, which will have both LEs equal to 0. As K increases chaotic trajectories begin to emerge, for K=2, phase space is dominated by chaotic trajectories with only a few islands of stability. The chaotic

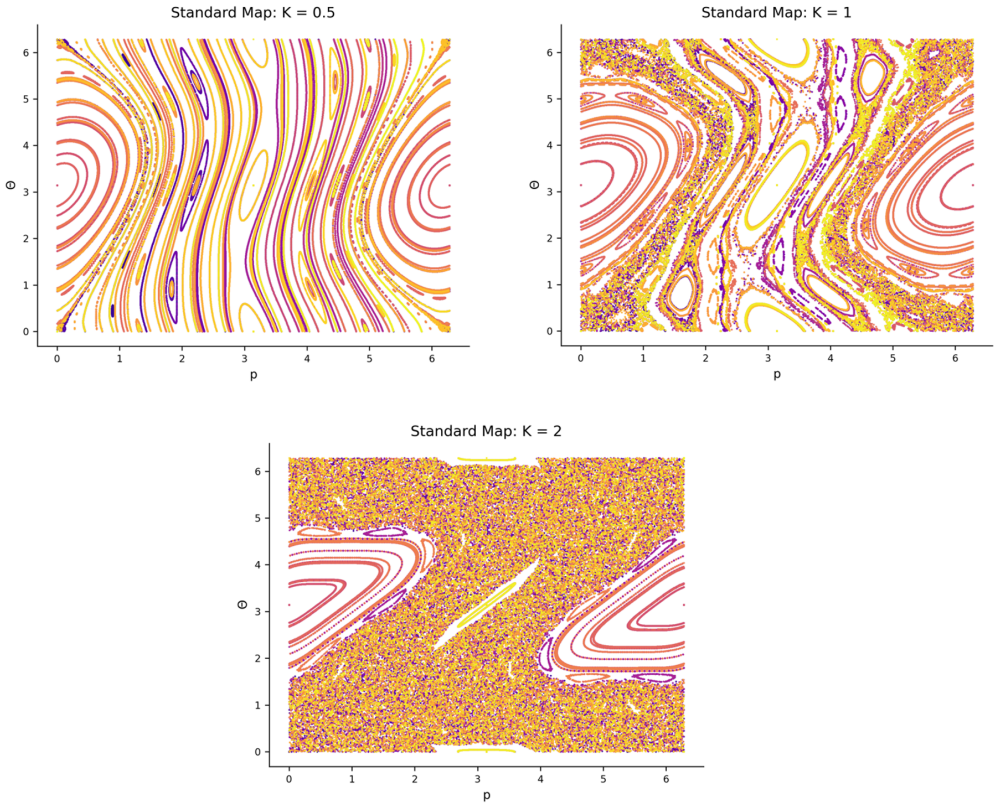

Figure 9: Standard Map phase diagrams for different K

trajectories should have a positive LLE and a mirrored negative second lyapunov exponent. The algorithm was tested on a variety of initial conditions for both chaotic and non-chaotic trajectories.

Figures 10 and 11 show that both Lyapunov exponents converge towards 0 (the correct value) for a variety of periodic orbits.

Similarly,Figures 12 and 13 demonstrate that the LEs for chaotic orbits converge to non-zero values and satisfy the expected symmetric properties. These results help to validate the accuracy of the algorithm.

The reliability of the algorithm was further tested by repeating samples with the same initial conditions but different Q_0 displacement subspaces, for both chaotic and non-chaotic orbits, shown in Figures 15 and 14. Despite initial variation, all samples rapidly converged towards a common value (zero for no-chaotic and non-zero for chaotic)

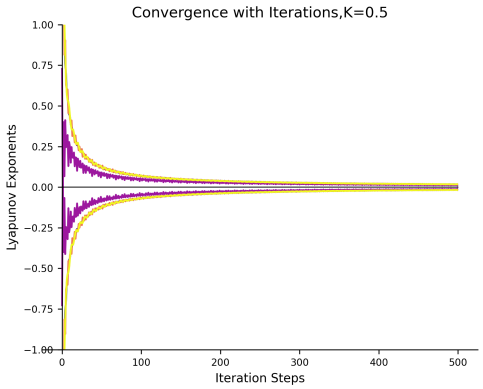

Figure 10: LE estimates vs number of re-orthogonalisation steps

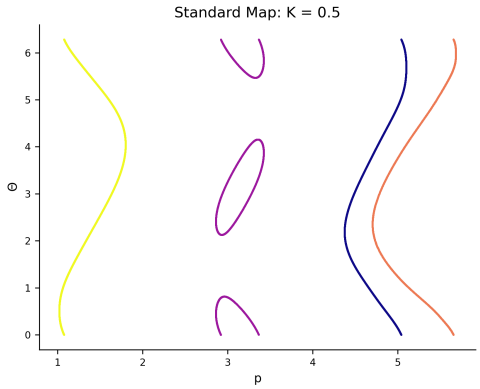

Figure 11: Corresponding phase diagram

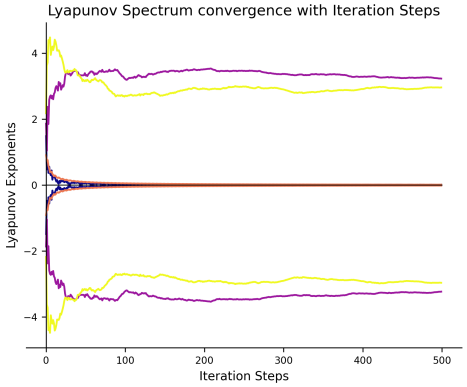

Figure 12: LE estimates vs number of re-orthogonalisation steps

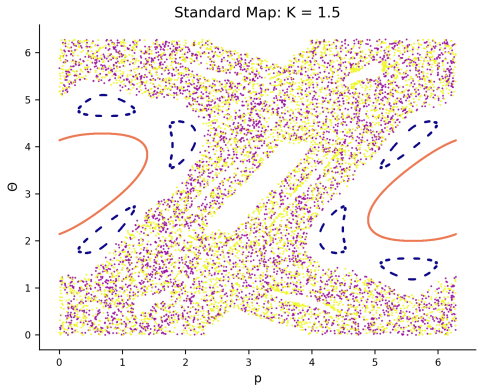

Figure 13: Corresponding phase diagram

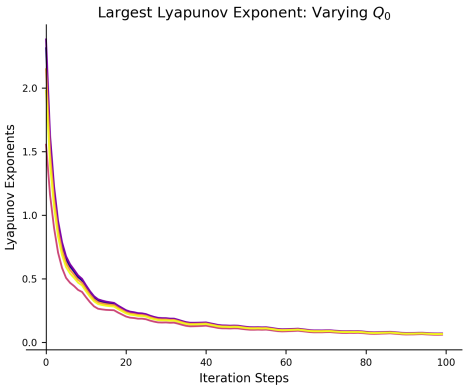

Figure 14: Convergence for different Q_0, non-chaotic trajectories.

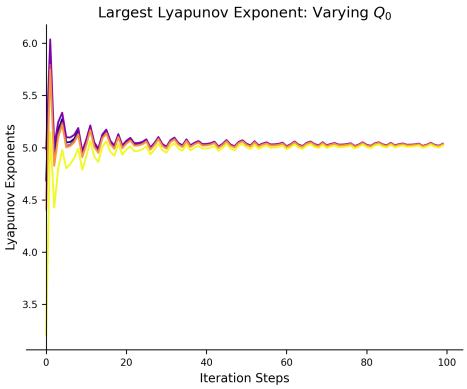

Figure 15: Convergence for different Q_0, chaotic trajectories.

5 Lorenz System

5.1 Theory and Analysis of Computational Physics

The Lorenz system is a 3D continuous time system governed by the equations:

$$\frac{dx}{dt} = \sigma(y - x), \tag{8}$$

$$\frac{dy}{dt} = x(\rho - z) - y, \tag{9}$$

$$\frac{dz}{dt} = xy - \beta z. \tag{10}$$

with standard parameters:

- $\sigma = 10$,
- $\rho = 28$,
- $\beta = \frac{8}{3}$.

For these parameters, the Lorenz system is chaotic and characterised by a strange attractor, a fractal 2D surface with an infinite surface area. The system is dissipative and therefore all trajectories, after an initial condition transient will end up on the strange attractor (see the attached animation).

The LLE algorithm used on the logistic map was altered to enable the calculation of the LLE for the Lorenz map. Due to the continuous nature of the system, the phase space trajectories and the infinitesimal perturbation are now described by coupled differential equations which must be simultaneously numerically integrated. This process was performed using scipy.solveivp due to its efficiency, easily changeable parameters and adaptive step size. The necessity of numerical integration introduces a cohort of errors due to the inability to perfectly follow chaotic trajectories in phase space. Using higher order integration methods and smaller step sizes will reduce this error at the expense of computational runtime.

Due to the exponential growth of the initial perturbation vector, it has to be periodically normalised so as to not encounter errors due to overflowing the limits of numpy 64 bit floating points.

5.2 Results and Analysis

Using the modified algorithm, LLE estimates were plotted against computational time, for a range of renormalisation time steps T, Figure 16. The algorithm was robust to changes in T and all samples showed convergence towards a positive LLE. As expected, the samples with larger T steps run slightly faster, as renormalisation doesn't have to be performed as often; however, it is clear that the runtime is dominated by the numerical integration.

The accuracy of the algorithm was assessed by comparing the distribution of LLE estimates from 200 samples with randomised initial conditions and perturbations. The 4th order RK45 integration method was used with a relative tolerance of $1e-3$ and an absolute tolerance of $1e-6$ and the trajectories were integrated for 6000s of simulation time. The result was an LLE value of 0.899 ± 0.002, shown in Figure 17.

Plotting the standard deviation of a distribution of LLE estimates against run-time gives an indication of the efficiency of the algorithm, shown in Figure 18. Performing a linear fit on a log-log plot, gives a gradient of -0.53 ± 0.02 which indicates that the error in the calculated LLE value approximately decreases with $t^{0.5}$, suggesting increasing runtime gives diminishing returns.

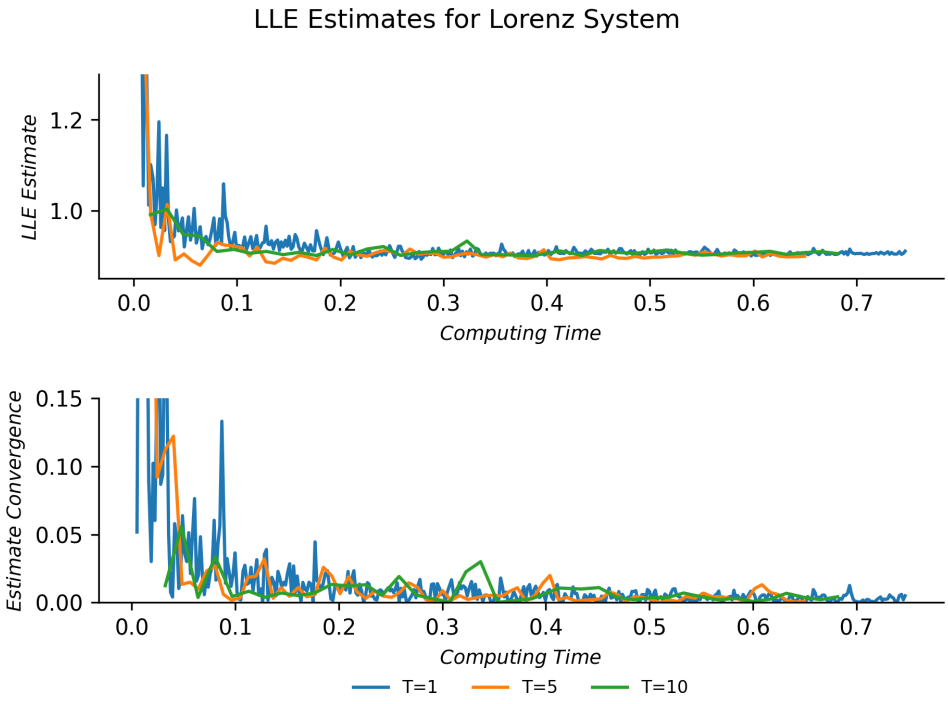

Figure 16: Depicting the evolution of the Q perturbation matrix

The main contributor to both runtime and error is the numerical integration, consequently, to try and optimise the efficiency of the algorithm, the error (defined by the standard deviation of a distribution of LLE estimates) and runtime were calculated for a variety of integration parameters, Figures 19 and 20. These plots show that runtime and error are relatively unaffected by changes to the absolute tolerance in the investigated range; however, runtime and error are both significantly affected by the relative tolerance. As expected, decreasing the relative tolerance decreases the error at the expense of longer runtime.

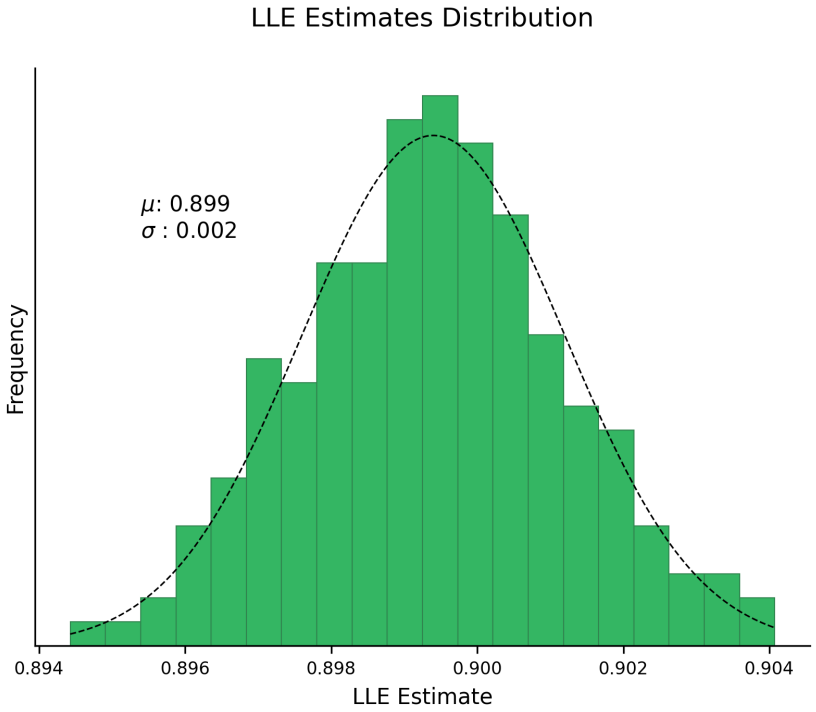

Figure 17: Variations in LLE estimate due to initial conditions

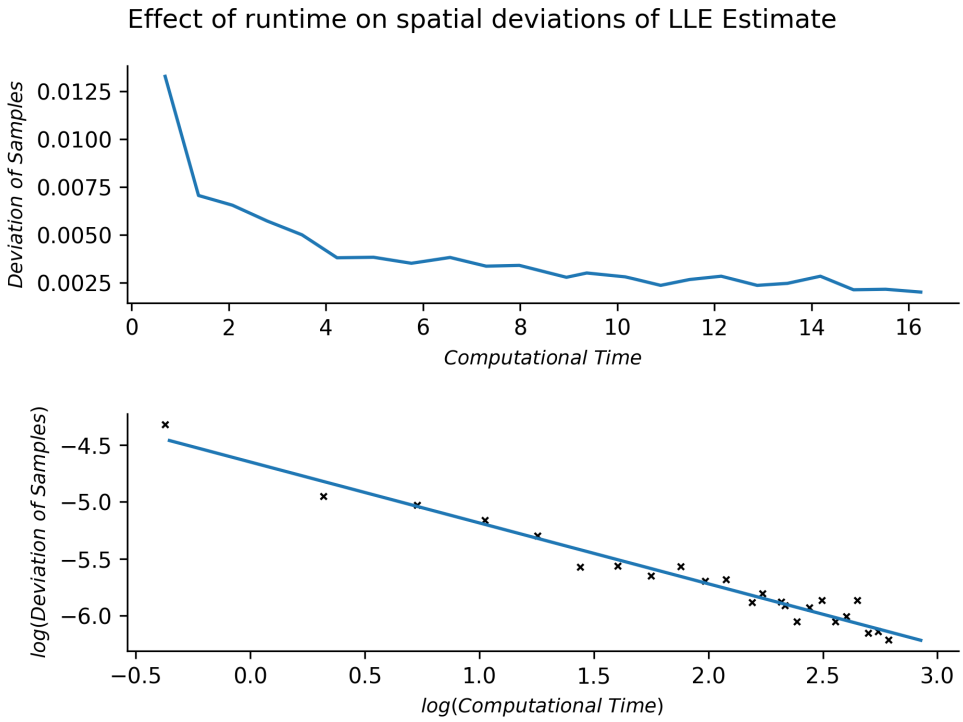

Figure 18: The convergence of LLE estimates with time

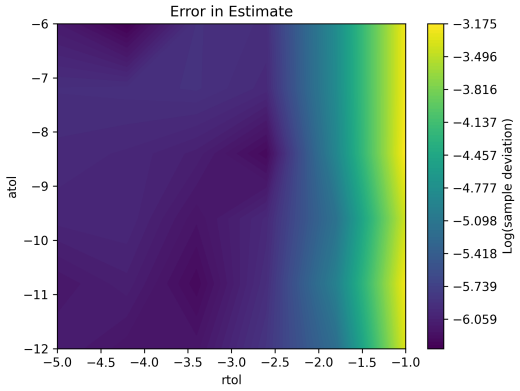

Figure 19: Affect of rtol and atol on error

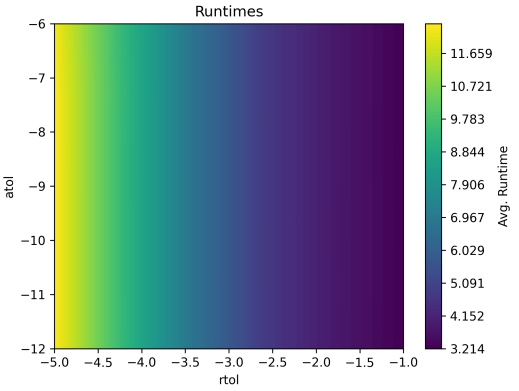

Figure 20: Affect of rtol and atol on runtime

6 Conclusions

Algorithms for calculating the LLE and lyapunov spectrum for discrete time systems and the LLE for continuous time systems were implemented in python, with analysis of their complexity and justification of specific code structure. The algorithms were robust with respect to altering initial conditions and gave qualitatively valid results. The quantitative accuracy was estimated by assessing the convergence of estimates with iterations and runtime. The LLE of the Lorenz system (for standard parameters) was calculated as 0.899 ± 0.002 and the LLE for the logistic map with r=3.7 was calculated as 0.355 ± 0.001.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] Luis Barreira and Ya B Pesin. *Lyapunov exponents and smooth ergodic theory*, volume 23. American Mathematical Soc., 2002.

[3] Timothy A Denton, George A Diamond, Richard H Helfant, Steven Khan, and Hrayr Karagueuzian. Fascinating rhythm: a primer on chaos theory and its application to cardiology. *American heart journal*, 120(6):1419–1440, 1990.

[4] J Doyne Farmer, Edward Ott, and James A Yorke. The dimension of chaotic attractors. *Physica D: Nonlinear Phenomena*, 7(1-3):153–180, 1983.

[5] Alain Goriely. *Integrability and nonintegrability of dynamical systems*, volume 19. World Scientific, 2001.

[6] Kazuo Nishimura and Makoto Yano. *Non-linear dynamics and chaos in optimal growth: An example*. Springer, 2012.

[7] Arkady Pikovsky and Antonio Politi. *Lyapunov exponents: a tool to explore complex dynamics*. Cambridge University Press, 2016.

[8] Troy Shinbrot, Celso Grebogi, Jack Wisdom, and James A Yorke. Chaos in a double pendulum. *American Journal of Physics*, 60(6):491–499, 1992.

[9] Julien Clinton Sprott. *Chaos and time-series analysis*. Oxford university press, 2003.

[10] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[11] Alan Wolf, Jack B Swift, Harry L Swinney, and John A Vastano. Determining lyapunov exponents from a time series. *Physica D: nonlinear phenomena*, 16(3):285–317, 1985.

A Software and Hardware Specifications

- All code developed in Visual Studio Code running Python version 3.9.13
- Formatting handled using Black Formatter, ms-python.black-formatter
- Model: MacBook Air (M1, 2020)
- Processor: Apple M1 chip, 8-core CPU (4 performance cores and 4 efficiency cores)

- Memory: 8 GB unified memory
- Storage: 256 GB SSD
- Operating System: macOS Monterey

B Code File Structure

- Folder: Logistic Map
  - LogisticMap.py: contains functions that perform the logistic map $x_n \to x_{n+1}$ on a 1D array of initial conditions, with options to vary r parameter for each initial condition and the number of iterations.
  - LogisticCalcs.py: contains a function that calculates the Lyapunov Exponent of the logistic map
  - LogisticAnalysis.py: Contains a variety of functions that test the LogisticCalcs algorithm and produces plots
  - BifurcationDiagram.py contains a function dedicated to creating a plot of the bifurcation diagram and corresponding Lyapunov exponents.
- Folder:Standard Map
  - StandardMap.py: contains functions that perform the standard map $x_n \to x_{n+1}$, also includes code for generating random initial conditions and a grid of initial conditions.
  - StandardMapCalcs.py: contains a function that calculates the spectrum of Lyapunov Exponents of the standard map
  - StandardMapAnalysis.py: Contains a variety of functions that test the StandardMapCalcs algorithm and produces plots
  - PhaseDiagram.py contains a function dedicated to creating phase diagrams for the Standard Map
- Folder:Lorenz
  - Lorenz.py: contains functions that return the derivatives for the lorenz equation and the associated derivatives for an infinitesimal perturbation, also contains functions to perform numerical integration on initial conditions and perturbation.
  - LorenzCalcs.py: contains a function that calculates the largest Lyapunov Exponent of the Lorenz system.
  - LorenzAnalysis.py: Contains a variety of functions that test the LorenzCalcs algorithm and produces plots
  - LorenzAnimation.py contains a function dedicated to creating an animation of the Lorenz system.
- Folder:QR
  - QRDecomposition.py: contains an assortment of different algorithms for performing QR decomposition
  - QRAnalysis.py: contains functions that analyse and compare the different QR algorithms, e.g. error and runtime