Time series analysis based on multiscale Recurrence Quantification Analysis, Horizontal Visibility Graph and its applications on biological sequences

Author
Ioannis Sfakianakis

Supervisor
Dimitrios Katsaros

Co-supervisor
Pavlos Pavlidis

Department of Electrical and Computer Engineering
University of Thessaly
February 2020

Ανάλυση χρονοσειρών βασισμένη σε πολυεπίπεδη επαναληπτική ανάλυση ποσοτικοποίησης, οριζόντιας ορατότητας γραφήματα και οι εφαρμογές της σε βιολογικές ακολουθίες.

Συγγραφέας
Ιωάννης Σφακιανάκης

Επιβλέπων Καθηγητής
Κατσαρός Δημήτριος

Επιβλέπων Καθηγητής
Παυλίδης Παύλος

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Φεβρουάριος 2020

Abstract

Στη βιβλιογραφιά έχει προταθεί μια πληθώρα μεθόδων για την πρόβλεψη της δομής της πρωτεΐνης. Στην παρούσα διπλωματική εργασιά θα επιτευχθεί η ανάλυση χρονοσειρών, μέσα από την επαναληπτική ανάλυση ποσοτικοποίησης και τα οριζόντιας ορατότητας γραφήματα. Αρχικά, η θεωριά αναπαράστασης του χάους χρησιμοποιείται για να χαρτογραφίσει την αλληλουχία της πρωτεϊνικής δομής σε δύο χρονοσειρές. Στην συνέχεια, συνδιάζοντας την επαναληπτική ανάλυση ποσοτικοποίησης και τα οριζόντιας ορατότητας γραφήματα παράγεται ένα διάνυσμα με 30 χαρακτηρηστικά. Ο σκοπός της παρούσας διπλωματικής εργασιάς είναι να δημιουργηθεί μια εφαρμογή, η οποία θα συμπεριλαμβάνει τις προαναφερόμενες μεθόδους και η οποία θα είναι διακριτά πιο γρήγορη απο τις ήδη υπάρχουσες.

Λέξεις Κλειδιά: Ανάλυση Χρονοσειρών, Θεωριά Αναπαράστασης του Χάους, Επαναληπτική Ανάλυση Ποσοτικοποίησης, Οριζόντιας Ορατότητας Γραφήματα , Γλώσσα Προγραμματισμού Julia

Abstract

In the literature, a variety of methods are suggested for predicting protein structure. In this thesis, time series analysis is achieved, through recurrence quantification analysis and horizontal visibility graph. First, the chaos game representation is used to map the protein structure sequence into two time series. Then a 30 dimensional feature vector is acquired by combining recurrence quantification analysis and horizontal visibility graph. The main objective of this thesis, is to create an application, that combines the aforementioned methods and is notably faster than the existing ones.

Keywords: Time Series Analysis, Chaos Game Representation, Rerecurrence Quantification Analysis, Horizontal Visibility Graphs, Julia Programming Language

Contents

Chapter 1

Introduction

1.1 Time series analysis

Time series ideas appear virtually to all activities. Time series are used by humans for communication, description, and visualization. Because time is a physical concept, parameters and other characteristics of mathematical models for time series can have real-world interpretations. This is beneficial for the analysis and synthesis of time series. Time series are basic to scientific investigations. Basic questions of scientific concern are formulated in terms of time series concepts such as predicting values, describing models and introducing new models. *Because of the tremendous variety of possibilities*, substantial simplifications are needed in many time series analyses. *The subject of time series analysis would be important if for no other reason than that it provides means of examining the basic assumption of statistical independence invariably made in ordinary statistics* Brillinger (2000).

Data in business, economics, engineering, environment, medicine, and other fields of scientific investigations are often collected in regular time intervals. For example, many familiar time series occur in the field of economics, where we are continually exposed to daily stock market quotations or monthly unemployment figures. Social scientists follow population series, such as birthrates or school enrollments. In medicine, blood pressure measurements traced over time could be useful for evaluating drugs used in treating hypertension. Time series data tend to exhibit patterns such as trends, seasonal fluctuations, irregular cycles, and occasional shifts in level or variability. The data are, therefore, dynamically or serially correlated. In practice, the main objectives of analyzing time series are:

(a) to ascertain and extrapolate the dynamic pattern in the data for forecasting future observations

(b) to assess the effect of exogenous interventions known to have occurred at given time points

(c) to detect unknown and unsuspected interventions.

The first step in the analysis of a time series is usually to plot the series and look for features that indicate a specific pattern. These are 4 different types of time series patterns Brillinger (1981):

1. Trend: reflects the long-term progression of the series. A trend exists when there is a persistent increasing or decreasing direction in the data. The trend component does not have to be linear.

2. Cyclic: reflects repeated but non-periodic fluctuations.

3. Seasonal: reflects seasonality present in the time series data, like demand for flip flops, will be highest during the summer season. Seasonality occurs at a fixed period of time could be weekly, monthly, quarterly, etc.

4. Random: reflects random or irregular influences. This is residual after we have removed all other components from time-series data.

In this work, we focus on processing of biological sequences such as (i) sequence alignment and assembly and (ii) gene and protein prediction, using time series analysis methods.

1.2 Protein overview

Proteins are large biomolecules, or macromolecules, consisting of one or more long chains of amino acid residues. Proteins perform a vast array of functions within organisms, including catalysing metabolic reactions, DNA replication, responding to stimuli, providing structure to cells, and organisms, and transporting molecules from one location to another. Proteins differ from one another primarily in their sequence of amino acids, which is dictated by the nucleotide sequence of their genes, and which usually results in protein folding into a specific three-dimensional structure that determines its activity.

A polypeptide is a linear chain of amino acid residues and at least one long polypeptide is contained in a protein. Short polypeptides, containing less than 20–30 residues, are rarely considered to be proteins. The individual amino acid residues are bonded together by peptide bonds and adjacent amino acid residues. The sequence of amino acid residues in a protein is defined by the sequence of a gene, which is encoded in the genetic code. In general, the genetic code specifies 20 standard amino acids. Shortly after or even during synthesis, the residues in a protein are often chemically modified by post-translational modifications, which may alter the physical and chemical properties, folding, stability, activity, and ultimately, the function of the proteins. Sometimes proteins have non-peptide groups attached, which can be called prosthetic groups or cofactors. Proteins can also work together to achieve a particular function, and they often associate to form stable protein complexes.

Once formed, proteins only exist for a certain period and are then degraded and recycled by the cell's machinery through the process of protein turnover. A

protein's lifespan is measured in terms of its half-life and covers a wide range. They can exist for minutes or years with an average lifespan of 1–2 days in mammalian cells. Abnormal or misfolded proteins are degraded more rapidly either due to being targeted for destruction or due to being unstable Nelson et al. (2008).

1.2.1 Protein Structure

The majority of proteins fold into unique 3-dimensional structures. The shape into which a protein naturally folds is known as its native conformation. Although many proteins can fold unassisted, simply through the chemical properties of their amino acids, others require the aid of molecular chaperones to fold into their native states. The functions of proteins are maintained because of their ability to recognize and interact with a variety of molecules. The three-dimensional structural conformation provides and maintains the functional characteristics. The three-dimensional structure, in turn, is dependent on the primary structure. So, any difference in the primary structure may produce a protein which cannot serve its function.

There are four distinct aspects of a protein's structure (visually represented at Figure 1.1):

- **Primary structure:** the amino acid sequence. A protein is a polyamide.
- **Secondary structure:** regularly repeating local structures stabilized by hydrogen bonds. The most common examples are the α-helix, β-sheet and turns. Because secondary structures are local, many regions of different secondary structure can be present in the same protein molecule.
- **Tertiary structure:** the overall shape of a single protein molecule; the spatial relationship of the secondary structures to one another. Tertiary structure is generally stabilized by nonlocal interactions, most commonly the formation of a hydrophobic core, but also through salt bridges, hydrogen bonds, disulfide bonds, and even posttranslational modifications. The term "tertiary structure" is often used as synonymous with the term fold. The tertiary structure is what controls the basic function of the protein.
- **Quaternary structure:** the structure formed by several protein molecules (polypeptide chains), usually called protein subunits in this context, which function as a single protein complex.
- **Quinary structure:** the signatures of protein surface that organize the crowded cellular interior. Quinary structure is dependent on transient, yet essential, macromolecular interactions that occur inside living cells.

Proteins are not entirely rigid molecules. In addition to these levels of structure, proteins may shift between several related structures while they perform their functions. In the context of these functional rearrangements, these tertiary or quaternary structures are usually referred to as "conformations", and

transitions between them are called conformational changes. Such changes are often induced by the binding of a substrate molecule to an enzyme's active site, or the physical region of the protein that participates in chemical catalysis. In solution proteins also undergo variation in structure through thermal vibration and the collision with other molecules.

Proteins can be informally divided into three main classes, which correlate with typical tertiary structures: globular proteins, fibrous proteins, and membrane proteins. Almost all globular proteins are soluble and most of them are enzymes. Fibrous proteins are often structural, such as collagen, the major component of connective tissue, or keratin, the protein component of hair and nails. Membrane proteins often serve as receptors or provide channels for polar or charged molecules to pass through the cell membrane.

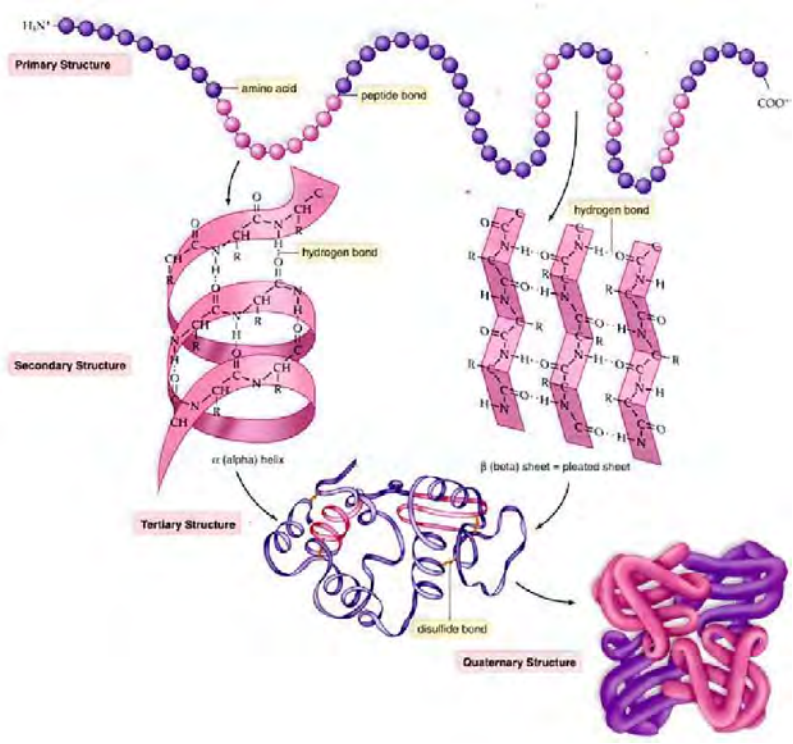

Figure 1.1: *Cartoon representation of the structural levels of proteins.*

1.2.2 Protein Folding Problem

According to the Protein Folding Problem, scientists try to predict 3D structure of proteins based on their amino acid sequence. Although it is known that a given sequence of amino acids almost always folds into a 3D structure with certain functions, it is impossible to predict, with high precision, the exact

folding pattern. Understanding the speed of proteins folding, which occurs extremely quickly, has also become a challenge to scientists. To be able to understand any type of biochemical reaction requires isolation and structure determination of reactants, intermediates and products. In protein folding, the isolation of reactants, intermediates and products is complicated because most interactions in proteins are non-covalent and weak interactions which lead to rapid rates of interconversion between each reaction state. Therefore, the isolation of intermediates is not easily achieved Finkelstein (2018). Based on the complexity of protein folding, there are 3 major problems of protein folding:

- What is the folding code?
- What is the folding mechanism?
- Can we predict the native structure of a protein from its amino acid sequence?

In the late 1980s, scientists discovered that there is a sequence of amino acid code that folds proteins in a particular way. The starting point of protein folding is indeed the primary structure (the sequence of amino acids), also known as denatured state of the protein. Even the smallest amount of the denatured state can activate nucleation and proliferation carried out through protein folding pathways. Characterization of these denatured states of proteins at physiological conditions is very difficult because it is necessary to unfold the proteins to their denatured states without the presence of denaturants. Recent research has allowed the study of denatured states to reach new heights using the single-molecule approach. Also there have been advancements to study intermediates in protein folding. Nowadays, researchers predict the structure of a protein by inputting the amino acid sequence into a computer. The advanced technology and modeling software allow scientists and researchers to form a predicted structure. However, the structure is not accurate, as there is always a small degree of errors present. Nevertheless, this can speed up discovery of new medications since the digital structure can be manipulated.

In 1968, Levinthal (1968) pointed out that protein folding, with precision, happens in microseconds, which seems unrealistic and impossible. This is also known as the Levinthal's paradox. Nowadays, we have advanced methods such as mutational methods and hydrogen exchange methods, which allow us to see structural folding events. However, the dynamics and mechanism of protein folding still require additional research and understanding. There are two different classical mechanisms that have been used to describe folding of single domain proteins. The first of the mechanisms is called the **Diffusion-Collision Model**. Proteins that follow this mechanism fold in a stepwise manner that involves growing secondary structure elements. These elements then collide, combine and strengthen. The second mechanism is known as the **Nucleation-Condensation Model**. Proteins following this method have been seen to fold from an unstructured denatured state with simultaneous formation of secondary and tertiary structure. The inherent stability of individual protein segment is

a key factor in determining the folding mechanism of a given protein. Many times, cell's life relies on the ability of its constituent proteins to fold into 3D structures that are crucial for their function. The amount of folded functional protein in a cell depends on several factors such as, rate of protein biosynthesis and degradation.

The interactions and bonds of side chains within a particular protein determine its tertiary structure. There are several types of bonds and forces that hold a protein in its tertiary structure.

- Hydrophobic interactions greatly contribute to the folding and shaping of a protein. The "R" group of the amino acid is either hydrophobic or hydrophilic. The amino acids with hydrophilic "R" groups will seek contact with their aqueous environment, while amino acids with hydrophobic "R" groups will seek to avoid water and position themselves towards the center of the protein.

- Hydrogen bonding in the polypeptide chain and between amino acid "R" groups helps to stabilize protein structure by holding the protein in the shape established by the hydrophobic interactions.

- Due to protein folding, ionic bonding can occur between the positively and negatively charged "R" groups that come in close contact with one another.

- Folding can also result in covalent bonding between the "R" groups of cysteine amino acids. This type of bonding forms what is called a disulfide bridge. Interactions called van der Waals forces also assist in the stabilization of protein structure. These interactions pertain to the attractive and repulsive forces that occur between molecules that become polarized.

These forces contribute to the bonding that occurs between molecules Pauling et al. (1951).

As protein types continue to increase, there are more and more methods for predicting protein structure. Different types of proteins have different patterns of existence and according to their patterns, proteins can be classified as all-a, all-b, a/b, a+b. The all-a are to a large extent composed of a-helix, while all-b have mainly b-strands Levitt and Chothia (1976). These methods are implemented in various programming languages, but they are notably slow. Our objective is to create a tool that is significantly faster.

Chapter 2

Methods

In the literature one could find several algorithms to analyse biological sequences such as DNA, RNA or proteins. A pioneer in this scientific field was Jeffrey (1990), who presented a new method for representing DNA sequences, the chaos game representation. Several years later, Yang et al. (2009) intended to predict protein structural classes employing recurrence quantification analysis based on chaos game representation. Olyaee et al. (2016), expanded the work of Yang et al. (2009) using complex networks along with the recurrence quantification analysis. Motivated by the recently proposed work of Jiang et al. (2019) that achieve efficient protein structure classification, we utilised the following algorithms:

1. Chaos Game Representation
2. Recurrence Quantification Analysis
3. Horizontal Visibility Graph

for the analysis of protein tertiary structure. The classification of protein's structure is a time demanding process. Our goal is to create an application that is notably faster than the existing implementations. RQA is already implemented in Matlab, Python and Julia. CGR and HVG exist only in Matlab, so we intend to implement them in Julia.

2.1 Chaos Game Representation

The Chaos Game Representation (CGR) is a graphical representation of a sequence. It is a method of converting a one-dimensional sequence into a graphical form. This comprehensive approach provides a visual image of a sequence that provides an insightful route for pattern recognition. The method was first proposed for the DNA sequence by Jeffrey (1990). Afterwards due to the translation of DNA into amino acids, CGR was also used for protein structure classification.

In 1999 Jones (1999) using PSIPRED was directly applied on the secondary prediction of a protein structure. PSIPRED is a simple and accurate secondary structure prediction method, incorporating two feed-forward neural networks which perform an analysis on output obtained from PSI-BLAST (Position Specific Iterated - BLAST). PSI-BLAST is used to find related sequences and to build a position-specific scoring matrix. The prediction method or algorithm is split into three stages: generating a sequence profile, predicting initial secondary structure, and filtering the predicted structure. PSIPRED works to normalize the sequence profile generated by PSI-BLAST. Then, by using neural networking, initial secondary structure is predicted. A second neural network is used to filter the predicted structure of the first network. This network has one hidden layer and results in three output nodes (one for each secondary structure element: helix, sheet, coil). The three final output nodes deliver a score for each secondary structure element for the central position of the window. Using the secondary structure with the highest score, PSIPRED generates the protein prediction Jones (1999). In general, CGR can be extended in such a way that it becomes practicable for the visualization and the analysis of the amino acid sequence in a protein.

Mathematically, the chaos game is described by an iterated function system (IFS). An IFS is a set of pairs of linear equations, each pair of the form

$$\begin{cases} x = ax + by + e \\ y = cx + dy + f \end{cases}$$

Each pair of equations gives the formula for computing the new value of the x and y coordinates.

From a given sequence S, one can define trajectories in a bounded set conserving all its statistical properties. There is a one to one mapping between S and the points in the CGR, which means in particular that each point contains the whole history of the sequence. Also the CGR exhibits a property of self-similarity, a concept very important in the study of fractals and chaotic dynamics.

A CGR of a DNA sequence is plotted in a unit square, the four vertices of which are labelled by the nucleotides $A = (0, 0), C = (0, 1), G = (1, 1), T = (1, 0)$. The plotting procedure is as follows: the first nucleotide of the sequence is plotted halfway between the centre of the square and the vertex representing this nucleotide; successive nucleotides in the sequence are plotted halfway between the previous plotted point and the vertex representing the nucleotide being plotted Jeffrey (1990).

Our goal is to generalize the CGR method, so that can work for any arbitrary sequence with different number of symbols and as a result a different graph shape will be created.

2.2 Recurrence Quantification Analysis

Recurrence is a fundamental characteristic of many dynamical systems and was introduced by Poincaré (1890). It has been recognised that in a larger context recurrences are part of one of three broad classes of asymptotic invariants:

- growth of the number of orbits of various kinds and of the complexity of orbit families;
- types of recurrences; and
- asymptotic distribution and statistical behaviour of orbits.

Recurrence Plot

A recurrence plot (RP) is a plot showing, for each moment i in time, the times at which a phase space trajectory visits roughly the same area in the phase space as at time j. It is a graph of $\vec{x}(i) \approx \vec{x}(j)$, showing i on a horizontal axis and j on a vertical axis, where $\vec{x}$ is a phase space trajectory. Eckmann (1987) introduced recurrence plots, which provide a way to visualize the periodic nature of a trajectory through a phase space. Often, the phase space does not have a low enough dimension (two or three) to be pictured, since higher-dimensional phase spaces can only be visualized by projection into the two or three-dimensional sub-spaces. However, making a recurrence plot enables us to investigate certain aspects of the m-dimensional phase space trajectory through a two-dimensional representation Marwan et al. (2007).

A recurrence is the time when the trajectory returns to a location it has visited before. Let $x(i)$ be the i-th point on the orbit describing a dynamical system in d-dimensional space, for $i = 1, \ldots, N$. The recurrence plot is an array of dots in a $N \times N$ square, where a dot is placed at (i, j) whenever X_j is sufficiently close to X_i. In practice one proceeds as follows to obtain a recurrence plot from a time series u_i with length N. First, choosing an embedding dimension m, one constructs the m-dimensional orbit of X_i by using the time delay method:

$$X_i = (u_i, u_{i+\tau}, ..., u_{i+(m-1)\tau}), \tag{2.1}$$

for $i = 1, 2, ..., N_m$ and $N_m = N-(m-1)\tau$, where m is the embedding dimension and τ is the time delay. Next, one calculates the Euclidean distance between all vector pairs in the reconstructed phase space. By selecting the appropriate threshold ε, the recurrence matrix can be expressed as follows:

$$\mathbf{R(i,j)} = \Theta(\varepsilon - \|\vec{x}(i) - \vec{x}(j)\|), \quad i, j = 1, ..., N, \tag{2.2}$$

where N is the number of considered states x_i, ε_i is a threshold distance, $\|\|$ a norm, and $\Theta()$ the Heaviside function Marwan et al. (2007).

The RP has always a black main diagonal line, the line of identity (LOI), since by definition $R(i, i) \equiv 1, (i = 1, ..., N)$. Additionally, the RP is symmetric by definition with respect to the main diagonal, $R(i, j) \equiv R(j, i)$

In order to compute an RP, an appropriate norm has to be chosen. The most frequently used norms are the L_1-norm, the L_2-norm (Euclidean norm) and the L_∞-norm (Maximum or Supremum norm). Taking into account a fixed ε, the L_∞-norm finds the most, the L_1-norm the least and the L_2-norm an intermediate amount of neighbours. To compute RPs, the L_∞-norm is often applied, because it is computationally faster and allows to study some features in RPs analytically.

A deciding parameter of an RP is the threshold ε. If we choose a pretty small ε, there may be almost no recurrence points and we can not learn anything about the recurrence structure of the examined system. On the other hand, if ε is chosen too large, almost every point is a neighbour of every other point, which leads to a lot of artefacts. A too large ε includes also points into the neighbourhood which are simple consecutive points on the trajectory. This effect is called tangential motion and causes thicker and longer diagonal structures in the RP as they actually are. Hence, we have to find a compromise for the value of ε. Furthermore, the influence of noise can require choosing a larger threshold, because noise would distort any existing structure in the RP.
Different "rules of thumb" was recommended for the choice of threshold ε, for example a few per cent of the maximum phase space diameter was proposed.
Another possibility is to choose ε according to the recurrence point density of the RP by seeking a scaling region in the recurrence point density.
Another standard for the choice of ε takes into consideration that a measurement of a process is a composition of the real signal and some observational noise with standard deviation σ. In order to get similar results as for the noise-free situation, ε has to be chosen such that it is five times larger than the standard deviation of the observational noise, i.e. $\varepsilon > 5\sigma$.
For periodic processes, the diagonal structures within the RP can be used in order to determine an optimal threshold. For this purpose, the density distribution of recurrence points along the diagonals parallel to the line of identity is considered.

Recurrence Plot Structures

The initial purpose of RPs was to visualise trajectories in phase space, which is particularly beneficial in the case of high dimensional systems. RPs generate important insights into the time evolution of these trajectories, because typical patterns in RPs are related to a specific behaviour of the system. Recurrence Plots determined by their typology, can be classified as homogeneous, periodic, drift and disrupted:

- Homogeneous RPs are typical of stationary systems in which the relaxation times are short in comparison with the time spanned by the RP. An example of such an RP is that of a stationary random time series.

- Periodic and quasi-periodic systems have RPs with diagonal oriented, periodic or quasi-periodic recurrent structures (diagonal lines, checkerboard

structures). Irrational frequency ratios cause more complex quasi-periodic recurrent structures (the distances between the diagonal lines are different).

- A drift is caused by systems with slowly varying parameters, i.e. non-stationary systems. The RP pales away from the line of identity.
- Abrupt changes in the dynamics as well as extreme events cause white areas or bands in the RP. RPs allow finding and assessing extreme and rare events easily by using the frequency of their recurrences.

Although these plots can provide a qualitative assessment of systems, as recurrence and short line segments increase, the visual inspection becomes complicated. To overcome the subjectivity of the recurrence plot, definitions and procedures to quantify recurrence plot structures was introduced by Zbilut and Webber Jr (1992).

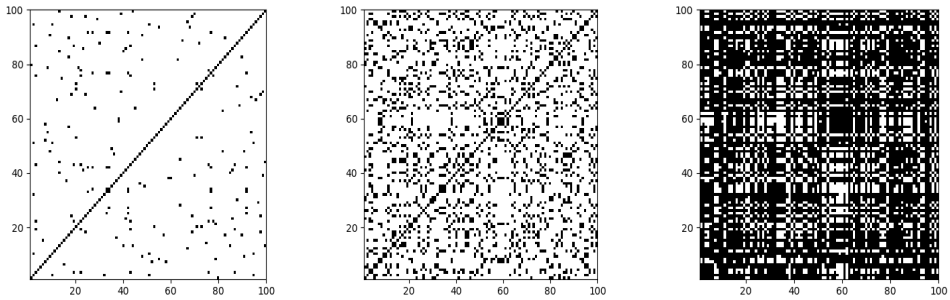

Figure 2.1: *Recurrence Plots with different thresholds ε for a random time series (left: $\varepsilon = 0.01$, middle: $\varepsilon = 0.1$, right: $\varepsilon = 0.5$)*

Quantification Analysis

In order to go beyond the visual impression derived by RPs, several measures of complexity which quantify the small-scale structures in RPs, have been introduced and are known as recurrence quantification analysis (RQA). These measures are based on the recurrence point density and the diagonal and vertical line structures of the RP. Computation of these measures in small windows (sub-matrices) of the RP moving along the line of identity yields the time dependent behaviour of these variables. Some studies based on RQA measures show that they are able to identify bifurcation points, especially chaos–order transitions. The vertical structures in the RP are related to intermittency and laminar states. Those measures quantifying the vertical structures enable also to detect chaos–chaos transitions.

Measures based on the recurrence density

- The simplest measure of the RQA is the recurrence **rate** or percent recurrences:

$$RR = \frac{1}{N^2} \sum_{i,j=1}^{N} R(i,j) \tag{2.3}$$

  which is a measure of the density of recurrence points in the RP. Note that it corresponds to the definition of the correlation sum, except that the line of identity is usually not included. Furthermore, in the limit $N \to \infty$, RR is the probability that a state recurs to its ε-neighbourhood in phase space. The value

$$N_n = \frac{1}{N} \sum_{i,j=1}^{N} R(i,j) \tag{2.4}$$

  is simply the average number of neighbours that each point on the trajectory has in its ε-neighbourhood.

Measures based on diagonal lines

Processes with uncorrelated or weakly correlated, stochastic or chaotic behaviour cause none or very short diagonals (l), whereas deterministic processes cause longer diagonals and less single, isolated recurrence points.

- The next measure is the percentage of recurrence points which form diagonal lines in the recurrence plot of minimal length l_{min}:

$$DET = \frac{\sum_{\ell=\ell_{\min}}^{N} \ell\, P(\ell)}{\sum_{\ell=1}^{N} \ell P(\ell)} \tag{2.5}$$

  where $P(\ell)$ is the frequency distribution of the lengths ℓ of the diagonal lines (i.e., it counts how many instances have length ℓ).

  This is introduced as a measure for **determinism** (or predictability) of the system. The threshold $\ell_{\min}$ excludes the diagonal lines which are formed by the tangential motion of the phase space trajectory. For $\ell_{\min} = 1$ the determinism is one.

- A diagonal line of length l means that a segment of the trajectory is rather close during l time steps to another segment of the trajectory at a different time; thus these lines are related to the divergence of the trajectory segments. The **average diagonal line length**:

$$L = \frac{\sum_{\ell=\ell_{\min}}^{N} \ell\, P(\ell)}{\sum_{\ell=\ell_{\min}}^{N} P(\ell)} \tag{2.6}$$

is the average time that two segments of the trajectory are close to each other, and can be interpreted as the mean prediction time.

- Another RQA measure considers the inverse of the length $L_{\max}$ of the longest diagonal line found in the RP and is called **divergence** or **linemax**:

$$DIV = \frac{1}{L_{\max}} \tag{2.7}$$

where $L_{\max} = \max(l_i; i = 1, ..., N_l)$.

These measures are related to the exponential divergence of the phase space trajectory. The faster the trajectory segments diverge, the shorter are the diagonal lines and the higher is the measure DIV. It was sometimes stated that the reciprocal of the maximal length of the diagonal lines would be an estimator for the positive maximal Lyapunov exponent of the dynamical system, but the divergence can only have the trend of the positive maximal Lyapunov exponent. As a result divergence can not reflect the maximal Lyapunov exponent.

- The measure **entropy** refers to the Shannon entropy of the probability $p(l) = P(l)/N_l$ to find a diagonal line of exactly length l in the RP:

$$ENT = -\sum_{\ell=\ell_{\min}}^{N} p(\ell) \ln p(\ell), \tag{2.8}$$

reflects the complexity of the deterministic structure in the system. However, this entropy depends sensitively on the bin number and, thus, may differ for different realisations of the same process.

Measures based on vertical lines

In continuous time systems discretised with sufficiently high time resolution and with an appropriate large threshold ε, a large part of these vertical lines usually correspond to the tangential motion of the phase space trajectory. However, not all elements of these sets belong to the tangential motion. For example, in systems with two different time scales, we might find vertical lines because of the finite size of the threshold ε, and not because of tangential motion.

- The amount of recurrence points which form vertical lines can be quantified in the same way as determinism:

$$LAM = \frac{\sum_{v=v_{\min}}^{N} vP(v)}{\sum_{v=1}^{N} vP(v)}, \tag{2.9}$$

where $P(v)$ is the frequency distribution of the lengths v of the vertical lines, which have at least a length of $v_{\min}$. This measure is called **laminarity** and represents the occurrence of laminar states in the system without

describing the length of these laminar phases.Laminarity will decrease if the RP consists of more single recurrence points than vertical structures.

- **Trapping time** measures the average length of the vertical lines,

$$TT = \frac{\sum_{v=v_{\min}}^{N} vP(v)}{\sum_{v=v_{\min}}^{N} P(v)} \tag{2.10}$$

  is related with the laminarity time of the dynamical system, i.e. how long the system remains in a specific state.

- Finally, the **maximal length** of the vertical lines in the RP

$$V_{\max} = \max(v_l; l = 1, ..., N_v) \tag{2.11}$$

  can be regarded, analogously to the standard measure $Lmax$.

In contrast to the RQA measures based on diagonal lines, these measures are able to find chaos–chaos transitions. Hence, they allow for the investigation of intermittency, even for rather short and non-stationary data series. Moreover, since for periodic dynamics the measures quantifying vertical structures are zero, chaos–order transitions can also be identified.

2.3 Horizontal Visibility Graph

The visibility algorithm has been introduced as a mapping between time series and complex networks. This process offers the possibility to apply complex networks procedures for time series characterization. It was first introduced by Lacasa et al. (2008). In my implementation, the Horizontal Visibility Graph algorithm was employed, an evolution of the initial visibility algorithm Luque et al. (2009) .
Let x_i, where $i = 1, ..., N$, be a time series of length N. The algorithm assigns each datum of the series to a node in the network. Two nodes i and j in the network are connected if one can draw a horizontal line in the time series joining x_i and x_j that does not intersect any intermediate data height. Thus, i and j are two connected nodes if the following geometrical criterion is fulfilled within the time series:

$$x_i, x_j > x_n \tag{2.12}$$

for all n such that $i < n < j$ (see Fig. 2.2).

The horizontal visibility graph associated to a time series is always:

1. Connected: each node sees at least its nearest neighbors (left-hand side and right-hand side).

2. Invariant under affine transformations of the series data: the visibility criterion is invariant under rescaling of both horizontal and vertical axis, as well as under horizontal and vertical translations.

3. Reversible/Irreversible character of the mapping: some information regarding the time series is inevitably lost in the mapping from the fact that the network structure is completely determined in the adjacency matrix.

4. Undirected/directed character of the mapping: Although this algorithm generates undirected graphs, note that one could also extract a directed graph (related to the temporal axis direction) in such a way that for a given node one should distinguish two different degrees: an ingoing degree k_{in}, related to how many nodes see a given node i, and an outgoing degree k_{out}, that is the number nodes that node i sees Luque et al. (2009).

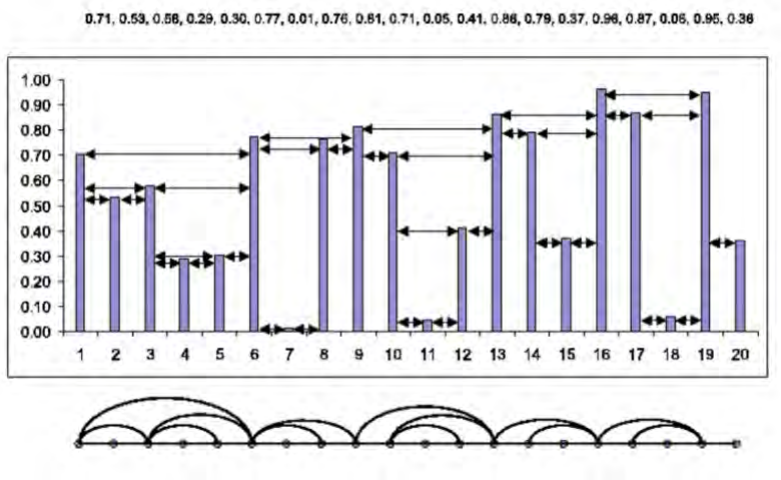

Figure 2.2: Example of a time series (20 data values) and the associated graph derived from the visibility algorithm. In the graph, every node corresponds, in the same order, to series data Lacasa et al. (2008).

Suppose each each sample point of time series x_i as a node n_i of graph $G = (V, E)$. G is an unweighed and undirected graph, where N and M are the number of nodes and the number of edges, respectively. Let A be the adjacency matrix of the graph G. Based on graph G, we can extract the following features:

- **Degree (k)** of any node i:

$$k_i = \sum_{j=1}^{N} A_{ij} \tag{2.13}$$

We choose the maximum value of degree k_{max} as our feature.

- **Average shortest path (L)** between the nodes:

$$L = \frac{1}{n(n-1)} \sum_{i \neq j} d_{i,j} \tag{2.14}$$

- **Diameter (D)**: is defined as the largest value of all the shortest path lengths in a network. Diameter is a measure of the compactness in a network and is computed by:

$$D = \max(d_{i,j}) \tag{2.15}$$

$\forall i, j$ pairs of shortest paths Emerson and Gothandam (2012).

- **Clustering Coefficient** (C): of any node i is the ratio between the total number of links actually connecting its neighbors and the total number of all possible links between these neighbors. It is given by

$$C_i = \frac{e_i}{k_i(k_i - 1)/2} \tag{2.16}$$

where e_i is the actual number of edges between the neighbors of node j.The clustering coefficient of the network is the average of C_i overall nodes Chang et al. (2008).

$$C = \frac{1}{N} \sum_{i} C_i \tag{2.17}$$

- **Energy** (E) of the graph is defined as:

$$E(G) = \sum_{i=1}^{n} |\lambda_i|, \tag{2.18}$$

where λ_i is the i^{th} eigenvalue of adjacency matrix A.

- **Laplacian Energy (LE)**. Let us define the Laplacian matrix as $L = D - A$, where D is a diagonal matrix containing the vertex degrees. It is defined as:

$$LE(G) = \sum_{i=1}^{n} |\mu_i - 2m/n|, \tag{2.19}$$

where μ_i is the i^{th} eigenvalue of the Laplacian.

Chapter 3

Implementation

All the algorithms mentioned in the previous chapter, have been already materialized in a plethora of programming languages, more notably in Matlab, Python and R. The problem with these implementations, especially for RQA which is computationally heavy, is that are time consuming. Our goal is to create an application that outperforms the existing ones and also be significantly faster. Based on the research done by Rosario (2019), the Recurrence Analysis package for Julia programming language is compared with other packages that have been developed for popular cross-platform programming languages, specifically:

- crqa version 1.0.7 for R
- CRP toolbox version 5.22 for Matlab
- pyunicorn version 0.5.1 for Python.

Figure 3.1 display a benchmark of the times taken by the four packages that are compared to compute the RQA of a time series (the first coordinate of a Rossler system with parameters $a = 0.25, b = 0.25, and c = 4$, starting at $(0, 0, 0)$ excluding the first 1000 points), taking samples of increasing length from $N = 250 to N = 3000$. The x-coordinate of the trajectories was embedded in three dimensions with a fixed delay of 6 samples; the recurrence matrix of the embedded time series was calculated using a fixed threshold $\varepsilon = 1.2$, and all RQA parameters were calculated. The first panel shows the times (milliseconds) in a natural scale, and the second in a logarithmic scale to see better the differences between Julia and Python. Both packages are outperformed by RecurrenceAnalysis and pyunicorn in Julia and Python, respectively, which show a similar speed in a natural scale, although pyunicorn is significantly faster, as clearly seen in the logarithmic scale: it is is about 5 times faster than RecurrenceAnalysis, between 20 and 50 times faster than crqa, and hundreds of times faster than the CRP Toolbox. An advantage of RecurrenceAnalysis with respect to all the other packages is that is is entirely written in Julia (without C or C++ code), so it is easy to inspect, extend and improve.

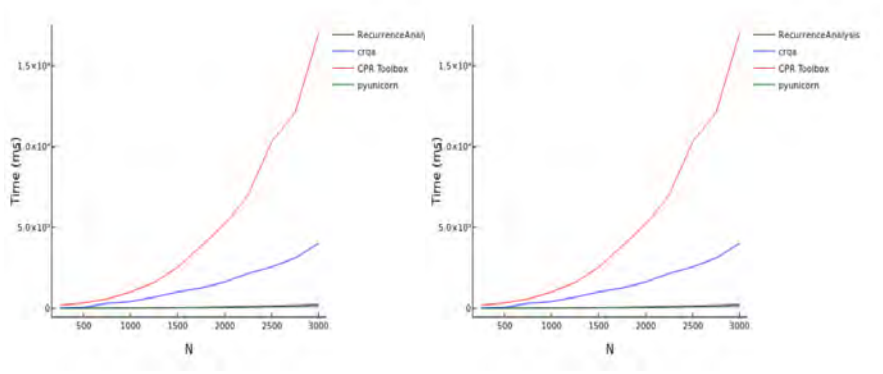

Figure 3.1: Median computation speeds of RQA by the different packages. Rosario (2019)

We choose Julia programming language for our implementation because it surpasses the existing ones and it is also a relatively new language so there is nothing similar already implemented. Julia is a high-level, high-performance, dynamic programming language. Julia supports concurrent, parallel and distributed computing, and direct calling of C and Fortran libraries without glue code. Julia shines is in its ability to balance speed and performance. It has the efficiency of a language like C, a language that requires you to specify variables and their corresponding actions, allowing the CPU to figure out what to do quickly and efficiently.

For the CGR algorithm, we decided to expand from the classic approaches that focuses only on protein and genome sequences and we adapted our approach, so it can work from the simplest approach of a triangle to higher polygons depending on the number of different characters we need to process.

At first, we calculate how many different symbols are contained in the sequence, so that we can find the proper angle θ to divide our polygon. If our sequence contains 3 symbols (e.g. protein sequence), an equilateral triangle with its centre at (0,0) will be created, if it has 4 symbols (e.g. DNA or RNA sequence) a square, if it has 5 a pentagon and so on. Afterwards, starting from (0,0) we locate the position of the first symbol by multiplying the rotation matrix of θ with the column vector $(1, 1)$. Then we locate the coordinates of the remaining symbols by multiplying the rotation matrix of θ with the previously calculated vector each time. Now that we have acquired the initial coordinates of all symbols, we are capable of calculating our timeseries x, y, using the following

commands:

```
Data: v= Dictionary with initial coordinates for each symbol in
       sequence, keys = Array with the keys of sequence
for j ← 1 to (length of sequence-1) do
    for i ← 1 to (count of symbols in sequence) do
        point in timeseries = v[i];
        if sequence[i]==keys[i] then
            if j==1 then
                x[j] = 0.5 * (0 + point[1]);
                y[j] = 0.5 * (0 + point[2]);
            end
            x[j+1] = 0.5 * (x[j] + point[1]);
            y[j+1] = 0.5 * (y[j] + point[2]);
        end
    end
end
```

Algorithm 1: Calculating timeseries x, y with CGR

Now that we have converted our sequence to two different time series x, y, we are capable of extracting the desired features using RQA and HVG algorithms.

In the 3.2 figure, we demonstrate some CGR examples of different symbols count.

Carrying on with the RQA described in 2.2, our work for this part was a lot easier, because the RecurrenceAnalysis package for Julia had all the required features already implemented. Our focus was centered in finding the most accurate time delay τ and embedding dimension m for our time series. For the time delay τ, we applied the uniform multivariate average mutual information method. To obtain coordinates for time delayed phase-space embedding that are as independent as possible, we are using the first minimum of mutual information as the optimal value of τ. If we can not find a local minimum, but may, e.g., be a monotonically decreasing function of τ, the lowest value of τ for which the AMI function drops below the value $1/\varepsilon$ is employed Wallot and Mønster (2018).

Concerning the embedding dimension m estimation, the False Nearest Neighbors (FNN) was utilised Kennel et al. (1992). FNN make the assumption that if two data points in the one-dimensional time series are close together, then they are neighbors. Their difference in magnitude provides us with the distance of those neighbors. If we embed the time series once (i.e., into two dimensions) using some time delay τ, then we can use the coordinates of those data points to examine whether the distance between them has changed appreciably. In particular, the optimal m is reached when i) FNN drops to 0, or ii) subsequent embeddings have the same number of false neighbors, or iii) the point before which the number of FNNs starts to increase again, or lastly iv) the point where the difference between the number false neighbors is less than a threshold $Ttol = 3\%$.

Having settled with the appropriate τ and m, we are able to reconstruct our

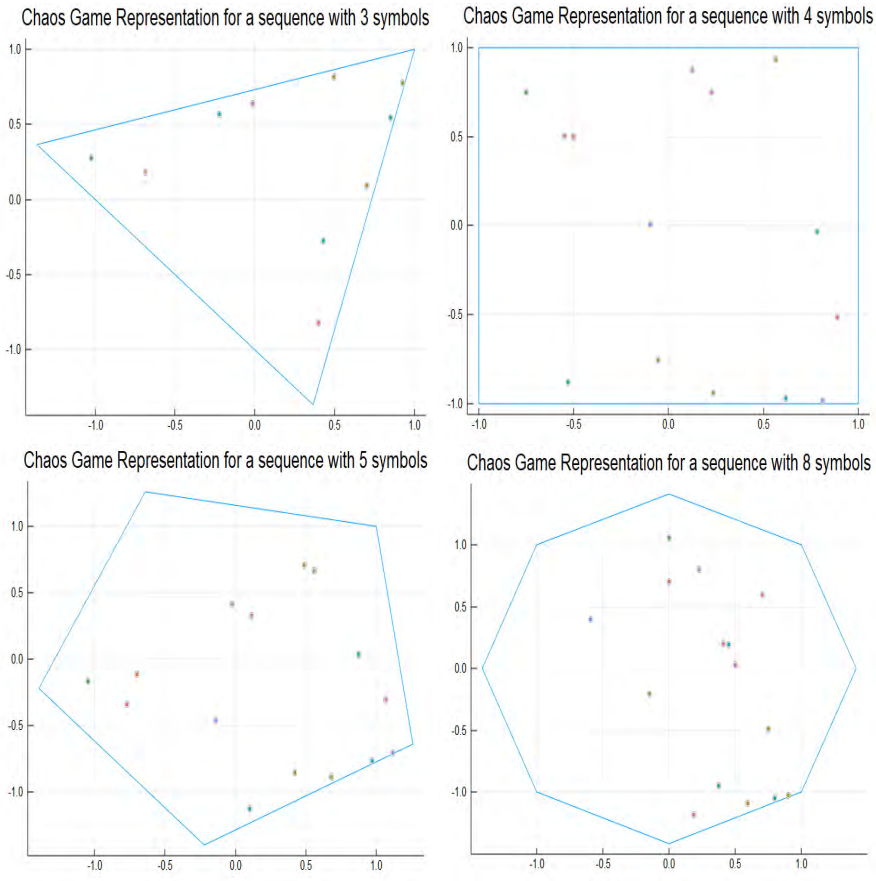

Figure 3.2: Chaos Game Representation for different sequences

initial time series x, y with julia's function *embed*. Afterwards, with $RecurrenceMatrix$ function, we calculate the recurrences in our time embedded time series, with threshold $\varepsilon = 0.1$ and the euclidean as a metric.

Finally, from the Recurrence Matrix, we extract the 8 features that are needed, namely: Recurrence Rate, Determinism, Laminarity, Trapping Time, Maximal Length, Entropy, Average Diagonal Line Length and Line Max.

As explained in section 2.3, for the creation of a horizontal visibility graph, we need to build the adjacency matrix that will contain the connections between the elements of the time series. This will be achieved, using the following

algorithm:

```
Data: sequence = the time series, mat = the adjacency matrix
for i ← 1 to (length of sequence) do
    for j ← i to (length of sequence-1) do
        if sequence[i] =>sequence[i+1] and i <length of sequence then
            j=j+1;
            mat[i,j]=1;
            mat[j,i]=1;
        end
        else
            j=j+1;
            mat[i,j]=1;
            mat[j,i]=1;
            break;
        end
    end
end
```

Algorithm 2: Calculation of the adjacency matrix

Now that we have established the connections between the nodes, with the function *graph*, the adjacency matrix is converted to a horizontal visibility graph. Just as RQA, the features that we want to deduce from our network is already implemented in julia, just like the majority of high level programming languages. The six features are some of the most common metrics that can describe a network. More specifically:

1. Degree: The maximum degree of the graph is retained.
2. Clustering Coefficient
3. Diameter
4. Average Shortest Path: That is calculated with the help of Bellman-Ford algorithm for shortest paths in a network.
5. Energy: Using the *eigenvals* function of the Linear Algebra Package for julia.
6. Laplacian Energy: Using the function *laplacianspectrum* that returns the eigenvalues of the Laplacian matrix for a graph.

Chapter 4

Experimental Results

In this work, in order to compare with the existing methods, the 25PDB low homologous protein was selected as benchmark dataset. The 25PDB dataset consists of 1,673 proteins with a similarity of approximately 25%. This dataset has 443 class $all - a$ proteins, 443 class $all - b$ proteins,346 class a/b proteins, and 441 class $a + b$ proteins.

Computation times

To test the performance of our implementation in Julia, another approach from a different programming language was needed. Thankfully, all the algorithms that we use was already utilised in Matlab, which was the answer to the aforementioned problem. Firstly, we tested the execution time of our implementation for different delays τ, while keeping the same embedding dimension m. The results are presented in the following matrix:

m	τ	execution time(seconds)
2	2	96.068564
2	3	96.729757
2	4	95.389177
3	2	95.018416
3	3	99.112987
3	4	94.857073
4	2	95.913025
4	3	97.840857
4	4	94.261608

We conclude that the execution time is remotely affected by the the embedding of the time series, as there is only a deviation of a few seconds.

For the comparison between the two different implementations, we decided to use $m = 3$ as embedding dimension while having a delay $\tau = 5$, for our two time series. The recurrence matrix of the embedded time series was calculated using

a fixed threshold $\varepsilon = 0.1$ and the euclidean as metric. The computations have been done on a machine running 64-bit Windows 10 Home operating system with system specifications: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz and 8GiB DDR4 Ram Memory. The software versions that we run the 2 different implementation were: Julia 1.2.0 and Matlab R2018a.

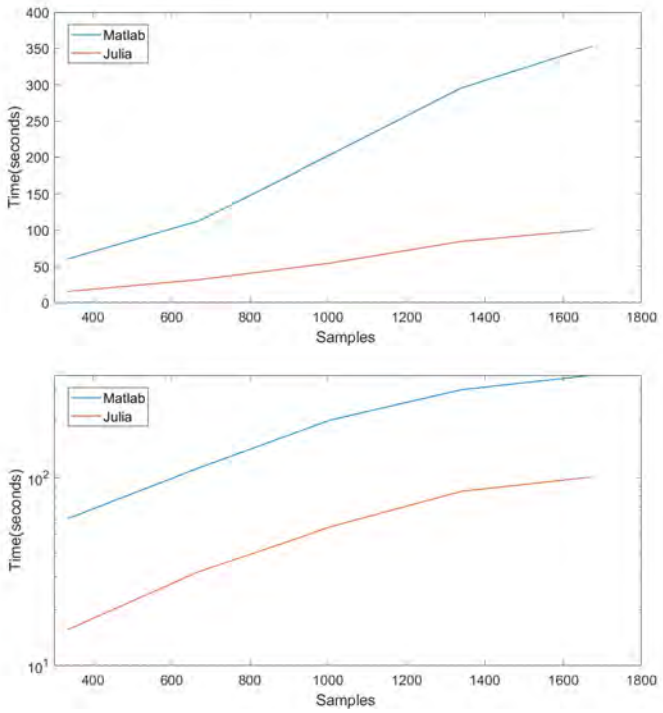

Figure 4.1: Computation speeds for the 2 implementations.

The first panel in 4.1 shows the time (seconds) in a natural scale, and the second in a logarithmic scale to see better the differences between Julia and Matlab.

It is obvious that Julia outperforms Matlab, it is 3 times faster.

We also compared each algorithm's execution time separately, where julia is also significantly faster.

	Julia	Matlab
CGR	1.724406	18.779072
HVG	38.863741	71.188770

Chapter 5

Future Work

In this work, a biological sequence is mapped by CGR into two time series. Then the feature extraction is based on RQA combined with complex networks. As a result, a vector with 30 features is obtained. The objective to create a faster implementation for analysing biological sequences, seems to have been accomplished.

My suggestions for further experimental work are the following: The 30-D vector can be used to predict the protein tertiary structure using the Support Vector Machine algorithm for the classification. Support Vector Machine (SVM) is a supervised machine learning algorithm for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems Cortes and Vapnik (1995). Furthermore, the optimal time delay tau and embedding dimension m for each time series can be found, using the AMI and FNN algorithms described in chapter 3. In this way, the features extracted from the time series will be more accurate and a higher accuracy can be achieved from SVM. The inclusion of these features that are currently missing, and possible improvements in the performance of the code, can contribute to a further computation speed gain and achieve the accuracy suggested in the paper of Jiang et al. (2019).

Bibliography

D. R. Brillinger. *Time series: data analysis and theory*, volume 36. Siam, 1981.

D. R. Brillinger. Time series: general. *Int. Encyc. Social and Behavioral Sciences*, 2000.

S. Chang, X. Jiao, C.-h. Li, X.-q. Gong, C.-x. Wang, et al. Amino acid network and its scoring application in protein–protein docking. *Biophysical chemistry*, 134(3):111–118, 2008.

C. Cortes and V. Vapnik. Support vector machine. *Machine learning*, 20(3): 273–297, 1995.

J. Eckmann. Recurrence plots of dynamical systems. *Europhysics Letters*, 5: 973–977, 1987.

I. A. Emerson and K. Gothandam. Network analysis of transmembrane protein structures. *Physica A: Statistical Mechanics and its Applications*, 391(3): 905–916, 2012.

A. Finkelstein. 50+ years of protein folding. *Biochemistry (Moscow)*, 83(1): S3–S18, 2018.

H. J. Jeffrey. Chaos game representation of gene structure. *Nucleic acids research*, 18(8):2163–2170, 1990.

H. Jiang, A. Zhang, Z. Zhang, Q. Meng, and Y. Li. Protein tertiary structure prediction based on multiscale recurrence quantification analysis and horizontal visibility graph. In *International Symposium on Neural Networks*, pages 531–539. Springer, 2019.

D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology*, 292(2):195–202, 1999.

M. B. Kennel, R. Brown, and H. D. Abarbanel. Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Physical review A*, 45(6):3403, 1992.

L. Lacasa, B. Luque, F. Ballesteros, J. Luque, and J. C. Nuno. From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, 105(13):4972–4975, 2008.

C. Levinthal. Are there pathways for protein folding? *Journal de chimie physique*, 65:44–45, 1968.

M. Levitt and C. Chothia. Structural patterns in globular proteins. *Nature*, 261 (5561):552–558, 1976.

B. Luque, L. Lacasa, F. Ballesteros, and J. Luque. Horizontal visibility graphs: Exact results for random time series. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 80:046103, 10 2009. doi: 10.1103/PhysRevE.80.046103.

N. Marwan, M. C. Romano, M. Thiel, and J. Kurths. Recurrence plots for the analysis of complex systems. *Physics reports*, 438(5-6):237–329, 2007.

D. L. Nelson, M. M. Cox, and A. L. Lehninger. *Principles of biochemistry*. Freeman New York:, 2008.

M. H. Olyaee, A. Yaghoubi, and M. Yaghoobi. Predicting protein structural classes based on complex networks and recurrence analysis. *Journal of theoretical biology*, 404:375–382, 2016.

L. Pauling, R. B. Corey, and H. R. Branson. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 37(4):205–211, 1951.

H. Poincaré. Sur le problème des trois corps et les équations de la dynamique. *Acta mathematica*, 13(1):A3–A270, 1890.

H. D. Rosario. Comparison of software packages for rqa. https://github.com/JuliaDynamics/RecurrenceAnalysis.jl/wiki/Comparison-of-software-packages-for-RQA, 2019.

S. Wallot and D. Mønster. Calculation of average mutual information (ami) and false-nearest neighbors (fnn) for the estimation of embedding parameters of multidimensional time series in matlab. *Frontiers in psychology*, 9:1679, 2018.

J.-Y. Yang, Z.-L. Peng, Z.-G. Yu, R.-J. Zhang, V. Anh, and D. Wang. Prediction of protein structural classes by recurrence quantification analysis based on chaos game representation. *Journal of Theoretical Biology*, 257(4):618–626, 2009.

J. P. Zbilut and C. L. Webber Jr. Embeddings and delays as derived from quantification of recurrence plots. *Physics letters A*, 171(3-4):199–203, 1992.