Scope of Optimization in Identification of Gene Regulatory Network

By

Dipanjan Biswas

Registration No.: 137304 of 2016-2017

Examination Roll No.: M6IAR19014

Under the guidance of
Dr. Pratyusha Rakshit

This thesis is submitted in the partial fulfillment for the Degree of Master of Technology in Intelligent Automation and Robotics under Electronics and Telecommunication Engineering

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Jadavpur University
Kolkata – 700032

May 2019

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled "Scope of Optimization in Identification of Gene Regulatory Network" has been carried out by Dipanjan Biswas (University Registration No. 137304 of 2016-2017) under my guidance and supervision and be accepted in partial fulfillment of the requirement for the Degree of Master of Technology in Intelligent Automation and Robotics of Jadavpur University.

Dr. Pratyusha Rakshit
(Project Guide)
Intelligent Automation and Robotics
Department of Electronics &
Telecommunication Engineering
Jadavpur University

Prof. Amit Konar
(Course Coordinator)
Intelligent Automation and Robotics
Department of Electronics &
Telecommunication Engineering
Jadavpur University

Dr. Sheli Sinha Chaudhuri
(Head of the Department)
Department of Electronics &
Telecommunication Engineering
Jadavpur University

Prof. Chiranjib Bhattacharjee
Dean. Faculty Council of
Engineering and Technology
Jadavpur University

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

CERTIFICATE OF APPROVAL

The forgoing thesis is hereby approved as a creditable study of engineering subject and presented in a manner satisfactory to warrant acceptance as prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn there in but approve the thesis only for which it is submitted.

Dipanjan Biswas

Examination Roll No.: M6IAR19014

Committee on final examination for evaluation of the thesis

External Examiner

Dr. Pratyusha Rakshit
(Supervisor)

FACULTY OF ENGINEERING AND TECHNOLOGY
JADAVPUR UNIVERSITY

DECLARATION OF ORIGINALITY OF COMPLIANCE
OF
ACADEMIC THESIS

I hereby declare that this thesis entitled "**Scope of Optimization in Identification of Gene Regulatory Network**" contains literature survey and original research work by the undersigned candidate, as part of his Degree of Master of Technology in Intelligent Automation and Robotics.

All information have been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by these rules and conduct, I have fully cited and referenced all materials when required and none of the work represented in this thesis is fabricated.

Name: Dipanjan Biswas

Examination Roll No.: M6IAR19014

Thesis Title: Scope of Optimization in Identification of Gene Regulatory Network

Date:
Place: Kolkata

Signature of the candidate

ACKNOWLEDGEMENTS

First and foremost, I would like to express my earnest gratitude and heartfelt indebtedness to my advisor, Dr. Pratyusha Rakshit, Department of Electronics and Telecommunication Engineering, for the privilege of allowing me to work under her towards my Degree of Master of Electronics & Telecommunication Engineering. This work would not have been materialized, but for her whole-hearted help and support. Working under her has been a great experience. I sincerely thank my supervisor, particularly for all the faith she had in me. I am thankful to Dr. Sheli Sinha Chaudhuri who acted as Head of the Department of Electronic and Telecommunication Engineering during the tenure of my studentship. I would also like to show my gratitude to the respected professors of the Department of Electronics and Telecommunication engineering particularly Prof. Amit Konar for his constant guidance and valuable advices.

I am indebted to my classmates and friends for the good wishes they have provided. Lastly, I would like to thank my family for their love and support.

Date:
Place: Kolkata

Dipanjan Biswas
Examination Roll No.: M6IAR19014
Jadavpur University

TABLE OF CONTENTS

CHAPTER 4: CONCLUSION & FUTURE WORK

CHAPTER 1

INTRODUCTION

1.1. GENE

A **gene** is a sequence of nucleotides in DNA or RNA that codes for a molecule that has a function. A gene is the basic physical and functional unit of heredity. Some genes act as instructions to make molecules called proteins. However, many genes do not code for proteins. In humans, genes vary in size from a few hundred DNA bases to more than 2 million bases. The Human Genome Project estimated that humans have between 20,000 and 25,000 genes.

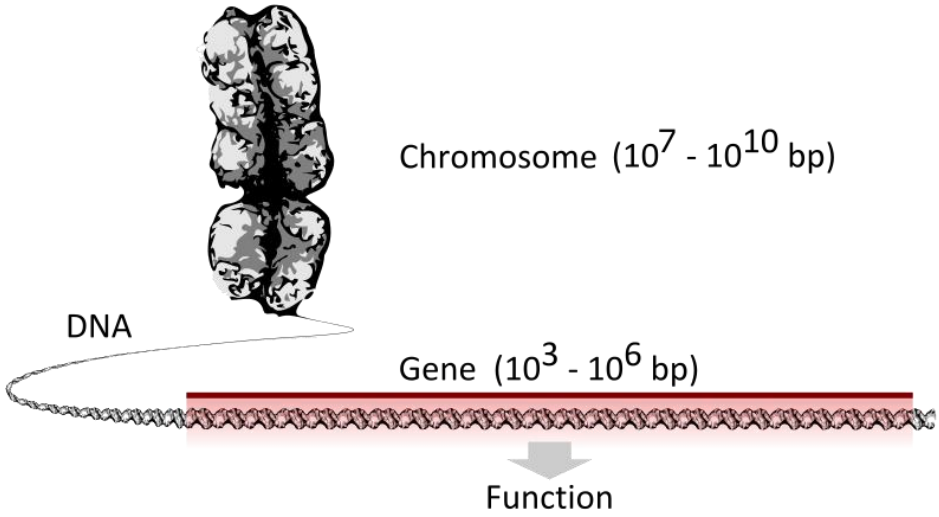


Fig 1.1. Figure shows gene is a region of DNA

Every person has two copies of each gene, one inherited from each parent. Most genes are the same in all people, but a small number of genes (less than 1 percent of the total) are slightly different between people. Alleles are forms of the same gene with small differences in their sequence of DNA bases. These small differences contribute to each person's unique physical features.

During gene expression, the DNA is first copied into RNA. The RNA can be directly functional or be the intermediate template for a protein that performs a function. The transmission of genes to an organism's offspring is the basis of the inheritance of phenotypic trait. These genes make up different DNA sequences called genotypes. Genotypes along with environmental and developmental factors determine what the phenotypes will be. Most biological traits are under the influence of polygenes (many different genes) as well as gene–environment interactions. Some genetic traits are instantly visible, such as eye color or number of limbs, and some are not, such as blood type, risk for specific diseases, or the thousands of basic biochemical processes that constitute life.

Genes can acquire mutations in their sequence, leading to different variants, known as alleles, in the population. These alleles encode slightly different versions of a protein, which cause different phenotypical traits. Usage of the term "having a gene" (e.g., "good genes," "hair color gene") typically refers to containing a different allele of the same, shared gene. Genes evolve due to natural selection / survival of the fittest and genetic drift of the alleles.

The structure of a gene consists of many elements of which the actual protein coding sequence is often only a small part. These include DNA regions that are not transcribed as well as untranslated regions of the RNA.

Flanking the open reading frame, genes contain a regulatory sequence that is required for their expression. First, genes require a promoter sequence. The promoter is recognized and bound by transcription factors that recruit and help RNA polymerase bind to the region to initiate transcription. The recognition typically occurs as a consensus sequence like the TATA box. A gene can have more than one promoter, resulting in messenger RNAs (mRNA) that differ in how far they extend in the 5' end. Highly transcribed genes have "strong" promoter sequences that form strong associations with transcription factors, thereby initiating transcription at a high rate. Others genes have "weak" promoters that form weak associations with transcription factors and initiate transcription less frequently. Eukaryotic promoter regions are much more complex and difficult to identify than prokaryotic promoters.

Additionally, genes can have regulatory regions many kilobases upstream or downstream of the open reading frame that alter expression. These act by binding to transcription factors which then cause the DNA to loop so that the regulatory sequence (and bound transcription factor) become close to the RNA polymerase binding site. For example, enhancers increase transcription by binding an activator protein which then helps to recruit the RNA polymerase to the promoter; conversely silencers bind repressor proteins and make the DNA less available for RNA polymerase.

The transcribed pre-mRNA contains untranslated regions at both ends which contain a ribosome binding site, terminator and start and stop codons. In addition, most eukaryotic open reading frames contain untranslated introns which are removed before the exons are translated. The sequences at the ends of the introns dictate the splice sites to generate the final mature mRNA which encodes the protein or RNA product.

Many prokaryotic genes are organized into operons, with multiple protein-coding sequences that are transcribed as a unit. The genes in an operonare transcribed as a continuous messenger RNA, referred to as a polycistronic mRNA. The term cistron in this context is equivalent to gene. The transcription of an operon's mRNA is often controlled by a repressor that can occur in an active or inactive state depending on the presence of specific metabolites. When active, the repressor binds to a DNA sequence at the beginning of the operon, called the operator region, and represses transcription of the operon; when the repressor is inactive transcription of the operon can occur. The products of operon genes typically have related functions and are involved in the same regulatory network.

1.2. Functions of Gene

The chromosomes within our cells contain an enormous amount of information. It is estimated that humans have somewhere around 30,000 genes. Each gene codes for an RNA molecule that is either used directly or used as a guide for the formation of a protein such as the insulin shown earlier. Information in our cells generally flows in a predictable order from the storage form of the information (DNA) through the working form (RNA) into the final product (protein).

The process in which particular sections of DNA (genes) are used to produce RNA is known as transcription. The set of genes that are 'on' at any given time is critical. The variable environment in which we live means that different genes need to be 'on' at different times. For example, if a meal contains large amounts of lactose, a sugar found in milk, then our bodies respond by turning on (transcribing) the genes that lead to the production of enzymes that break down lactose. If a different sugar or nutrient is present, the correct genes need to be turned on to process it.

1.3. Central Dogma

The central dogma of molecular biology describes the two-step process, transcription and translation, by which the information in genes flows into proteins: DNA → RNA → protein. It was first proposed in 1958 by Francis Crick, discoverer of the structure of DNA.

The central dogma suggests that DNA contains the information needed to make all of our proteins, and that RNA is a messenger that carries this information to the ribosomes. The ribosomes serve as factories in the cell where the information

is translated from a code into the functional product. The process by which the DNA instructions are converted into the functional product is called gene expression.

The central dogma states that the pattern of information that occurs most frequently in our cells is:

- From existing DNA to make new DNA (DNA replication)
- From DNA to make RNA (transcription)
- From RNA to make new proteins (translation)

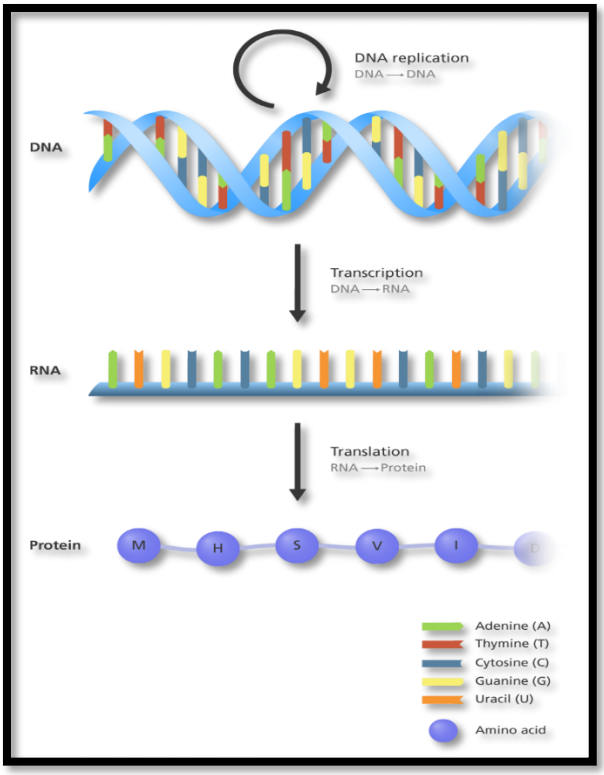


Fig.1.3. Flow of information between DNA, RNA, protein

Transcription

The goal of transcription is to make an RNA copy of a gene. This RNA can direct the formation of a protein or be used directly in the cell. All cells with a nucleus contain the same exact genetic information. As discussed, only a small percentage of the genes are actually being used to make RNA at any given time in a particular cell. The transcription process is very tightly regulated in normal cells.

- Genes must be transcribed at the correct time.
- The RNA produced from a gene must be made in the correct amount.

- ONLY the required genes should to be transcribed.
- Turning transcription off is just as important as turning it on.

You can picture this as a sophisticated production line, like you would find in a factory. You would want the assembly line working when you needed the product and shut down when you no longer needed the product.

Translation

After the messenger RNA (mRNA) is produced through the transcription process just described, the mRNA is processed in the nucleus and then released into the cytosol.

The mRNA is then recognized by the ribosomal subunits present in the cytosol and the message is 'read' by the ribosome to produce a protein. The information for the direction of protein formation is encoded in the sequence of nucleotides that make up the mRNA. Groups of three nucleotides (called codons) are 'read' by the ribosome and lead to the addition of a particular amino acid into the growing polypeptide (protein).

After the protein is formed it acquires its active folded state and is able to perform its functions in the cell. The proper folding, transportation, activity and eventual destruction of proteins are all highly regulated processes.

Replication

Finally, as the final step in the Central Dogma, to transmit the genetic information between parents and progeny, the DNA must be replicated faithfully. Replication is carried out by a complex group of proteins that unwind the super helix, unwind the double-stranded DNA helix, and, using DNA polymerase and its associated proteins, copy or replicate the master template itself so the cycle can repeat DNA → RNA → protein in a new generation of cells or organisms.

Exceptions to the central dogma

The central dogma is not really a dogma in the traditional sense of the word, like all scientific theories it is modified as we learn more details of the processes. The biggest revolution in the central dogma was the discovery of retroviruses, which transcribe RNA into DNA through the use of a special enzyme called reverse transcriptase has resulted in an exception to the central dogma; RNA → DNA → RNA → protein. Also, some virus species are so primitive that they use only RNA

→ proteins, having not developed DNA. With the discovery of prions, a new exception to the central dogma has been discovered, Protein → Protein. That is, proteins directly replicating themselves by making conformational changes in other proteins. Although retroviruses, certain primitive viruses, and prions may violate the central dogma, they are technically not considered "alive", and thus the rule that "all cellular life follows the central dogma" still holds true.

1.4. Gene Regulatory Network (GRN)

A gene regulatory network (GRN) is a collection of molecular regulators that interact with each other and with other substances in the cell to govern the gene expression levels of mRNA and proteins. These play a central role in morphogenesis, the creation of body structures, which in turn is central to evolutionary developmental biology.

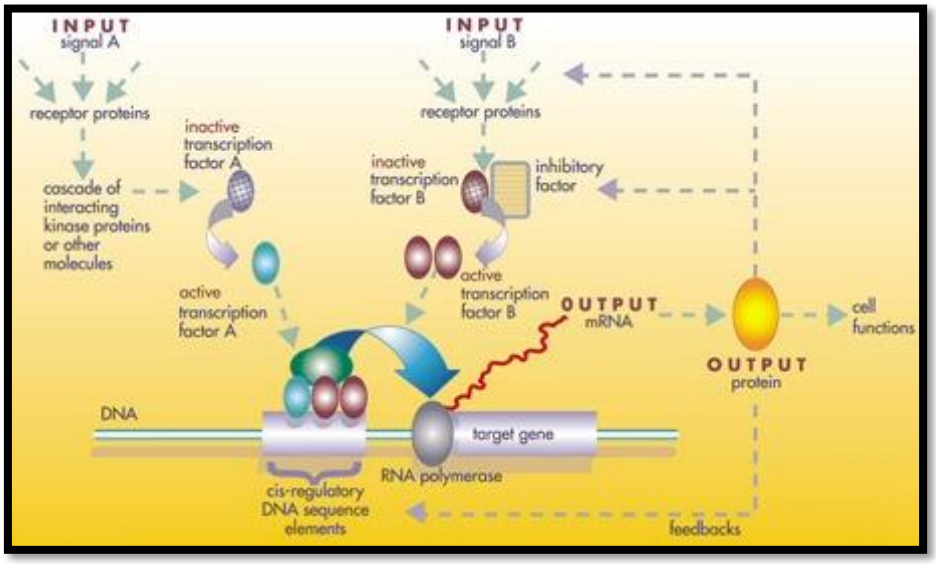


Fig. 1.4. Structure of a gene regulatory network

The regulator can be DNA, RNA, protein and complexes of these. The interaction can be direct or indirect (through transcribed RNA or translated protein). In general, each mRNA molecule goes on to make a specific protein (or set of proteins). In some cases this protein will be structural, and will accumulate at the cell membrane or within the cell to give it particular structural properties. In other cases the protein will be an enzyme, i.e., a micro-machine that catalyzes a certain reaction, such as the breakdown of a food source or toxin. Some proteins though serve only to activate other genes, and these are the transcription

factors that are the main players in regulatory networks or cascades. By binding to the promoter region at the start of other genes they turn them on, initiating the production of another protein, and so on. Some transcription factors are inhibitory.

In single-celled organisms, regulatory networks respond to the external environment, optimizing the cell at a given time for survival in this environment. Thus a yeast cell, finding itself in a sugar solution, will turn on genes to make enzymes that process the sugar to alcohol. This process, which we associate with wine-making, is how the yeast cell makes its living, gaining energy to multiply, which under normal circumstances would enhance its survival prospects.

In multicellular animals the same principle has been put in the service of gene cascades that control body-shape. Each time a cell divides, two cells result which, although they contain the same genome in full, can differ in which genes are turned on and making proteins. Sometimes a 'self-sustaining feedback loop' ensures that a cell maintains its identity and passes it on. Less understood is the mechanism of epigenetics by which chromatin modification may provide cellular memory by blocking or allowing transcription. A major feature of multicellular animals is the use of morphogen gradients, which in effect provide a positioning system that tells a cell where in the body it is, and hence what sort of cell to become. A gene that is turned on in one cell may make a product that leaves the cell and diffuses through adjacent cells, entering them and turning on genes only when it is present above a certain threshold level. These cells are thus induced into a new fate, and may even generate other morphogens that signal back to the original cell. Over longer distances morphogens may use the active process of signal transduction. Such signaling controls embryogenesis, the building of a body plan from scratch through a series of sequential steps. They also control and maintain adult bodies through feedback processes, and the loss of such feedback because of a mutation can be responsible for the cell proliferation that is seen in cancer. In parallel with this process of building structure, the gene cascade turns on genes that make structural proteins that give each cell the physical properties it needs.

1.5. Importance of Gene Regulatory Network (GRN)

GRNs provide the fundamental control mechanism directing developmental process. Gene expression is regulated sequence-specifically by the interaction of transcription factors with *cis*-regulatory DNA modules. Thus, the control operations which assign diverse cellular functions are those determining when and where transcription factor encoding genes will be expressed. By encoding the *cis*-regulatory inputs of every regulatory gene, GRNs specify the interactions among regulatory genes that are responsible for the expression of particular sets of transcription factors. These transcription factors in turn also control cohorts of genes encoding many other kinds of protein, here referred to as effector genes, that is, differentiation genes and morphogenesis genes. Cells manifest their fates in development by the programmed activation of distinct suites of effector genes, directly determining their biological properties, the final specific readout of developmental GRNs. Thus ultimately the expression of all genes in the genome is linked by interactions within GRNs.

Regulatory genes have the special feature that they play dual roles in the GRN, in that their expression is at once the output of the upstream regulatory genes which provide their transcriptional inputs, and at the same time they provide inputs to other target genes within the same network. Thus, the set of transcription factors present in a given time and place determines the new set of transcription factors to be expressed, which then in turn establishes the expression of another regulatory condition. The continuous changes of regulatory gene expression in developmental time can be regarded as the major driver of developmental progression. Development is powered by changes in states of regulatory gene activity and as a consequence of these changes, new cell fates are established in the construction of the body plan. Development is ultimately controlled by GRNs, and these constitute the primary machinery of control in Metazoa.

1.6. GRN as an optimization problem

Estimating the true gene regulatory network (GRN), when the number of genes is much greater than the number of samples, it has aroused considerable interest in the computational biology community. Several scientists and researchers have presented different approaches to this difficult problem and have advanced the field. However, many unsolved tasks in this area remain, including identifying

high correlated covariates, noisy data, and reasonable prior knowledge necessary to accomplish GRN inferring and model estimation.

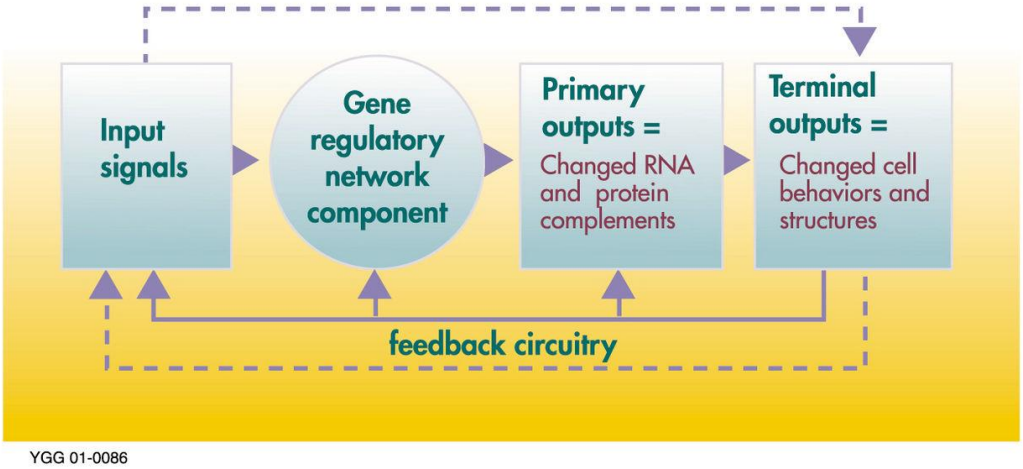


Fig.1.5. Process of gene regulatory network

The past 30 years has seen several developments concerned the learning structure of GRNs. The complex relationships between such components motivated us to identify a multivariate approach. Inferring gene networks is usually known as the process of identifying gene interactions from gene expression data through computational analysis. The entire inferring process can be summarized as a task of predicting connectivity among genes. Essentially, it involves learning the structure of a graph. However, in many domains, problems such as the large numbers of variables, small samples sizes, and possible presence of unmeasured causes, remain major impediments to practical applying these developments.

To accomplish this task and infer the structure of GRNs from high throughput microarray data, several techniques have been developed for the mathematical modeling of GRNs from expression data, notably differential equation, the vector autoregressive, dynamic Bayesian networks, Boolean networks, and the information-theoretic method. In addition, other methods based on regression, such as least absolute shrinkage and selection operator (Lasso), or partial least squares (PLS) and supervised learning, have shown positive results. Although standard linear modeling approaches enable analysis of a modeled system, they are not effective in large-scale network discovery. This is because the number of candidate parameters and models is extremely high, and thus searching efficiently and reliably with tight control on many false positives is difficult. By contrast, by

using ordinary differential equations to model transcriptional changes in terms of environmental and transcription factor influence, time-series network identification (TSNI) constructs a local regulatory network of genes that are affected by an external perturbation. Dynamic regulatory events miner (DREM) uses a Hidden Markov Model (HMM) based algorithm to identify transcription factors that control divergence points in gene expression profiles in order to reconstruct dynamic regulatory networks. Friedman et al. (Friedman, 2004) were among the first to suggest using dynamic Bayesian networks (DBNs) to model regulatory networks that change over time, as such models can capture time dependent structures such as feedback loops that are impossible to express using traditional probabilistic networks.

1.7. Organization of Thesis

The paper is organized as follows. In Chapter 1 we provides definitions of Gene, functions of gene, GNR, importance of GNR. Chapter 2 provides overview of Differential Evolution (DE) algorithm and pseudo code, important steps, applications of DE. In Chapter 3 we used DE algorithm with different activation functions such as Sigmoid, tanh, ReLU to optimize Gene Regulatory Network. And experiments and results are done in Chapter 3. Finally, the thesis is summarized and concluded in Chapter 4.

References

[1]. https://www.cancerquest.org/cancer-biology/gene-function

[2]. https://en.wikipedia.org/wiki/Gene

[3]. https://www.yourgenome.org/facts/what-is-the-central-dogma

[4]. http://groups.inf.ed.ac.uk/bionlp/links/central-dogma.html

[5]. https://en.wikipedia.org/wiki/Gene_regulatory_network

[6]. Davidson, E. H., & Peter, I. S. (2015). *Gene Regulatory Networks. Genomic Control Process*

[7]. Li, Y & Li, G. (2017). An optimization approach of the reconstructing gene regulatory networks and simulation based on dynamic Bayesian networks. Revista de la Facultad de Ingenieria. 32. 82-90.

CHAPTER 2

DIFFERENTIAL EVOLUTION

2.1. Introduction

Differential evolution (DE) is a stochastic, population-based search strategy developed by Storn and Price in 1995.[2][3] While DE shares similarities with other evolutionary algorithms (EA), it differs significantly in the sense that distance and direction information from the current population is used to guide the search process. Furthermore, the original DE strategies were developed to be applied to continuous-valued landscapes.

DE is used for multidimensional real-valued functions but does not use the gradient of the problem being optimized, which means DE does not require the optimization problem to be differentiable, as is required by classic optimization methods such as gradient descent and quasi-newton methods. DE can therefore also be used on optimization problems that are not even continuous, are noisy, change over time, etc.[1]

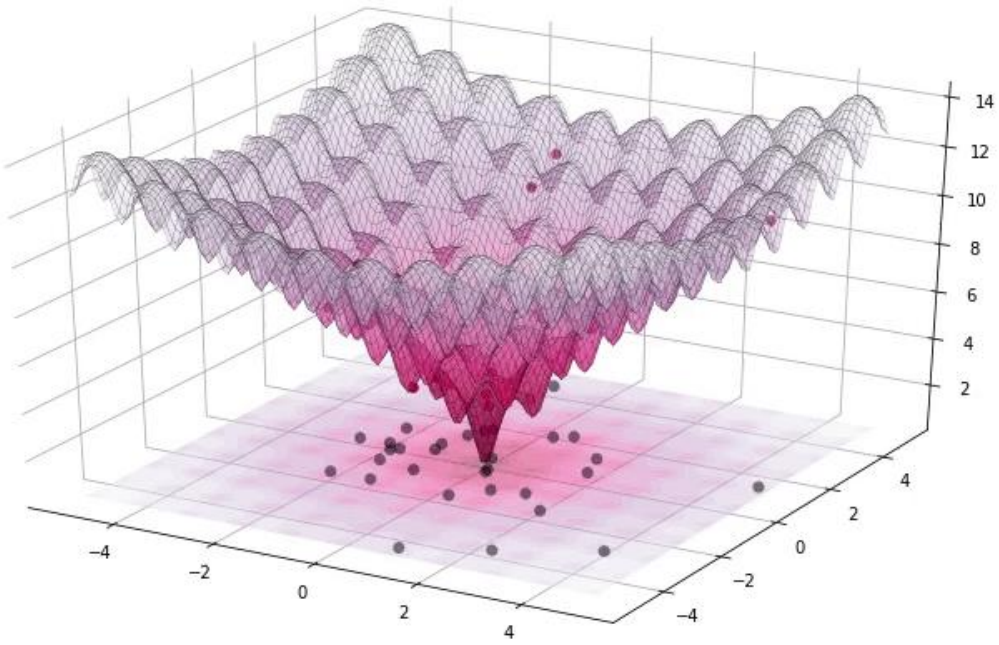


Fig.2.1. Differential evolution optimizing 2-D Ackley function

DE optimizes a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimization problem at hand. In this way the optimization

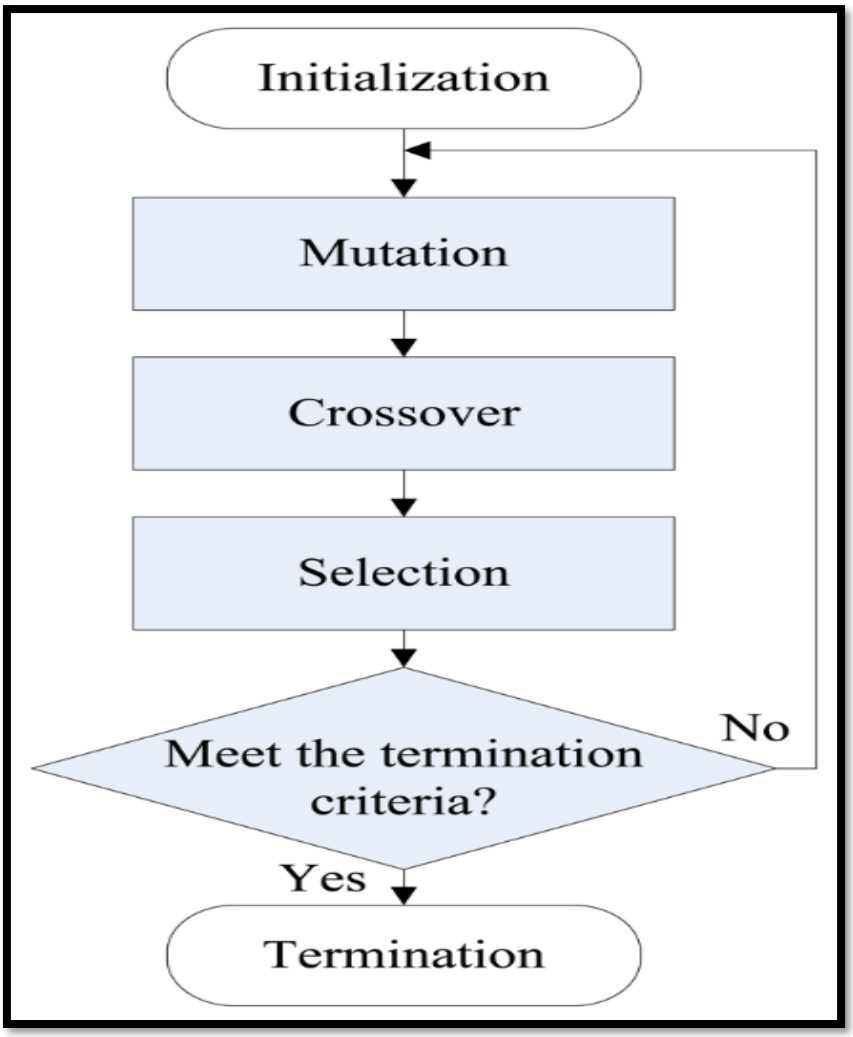
problem is treated as a black box that merely provides a measure of quality given a candidate solution and the gradient is therefore not needed.

A basic variant of the DE algorithm works by having a population of candidate solutions (called agents). These agents are moved around in the search-space by using simple mathematical formulae to combine the positions of existing agents from the population. If the new position of an agent is an improvement then it is accepted and forms part of the population, otherwise the new position is simply discarded. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

As in genetic algorithms, design parameters in a d-dimensional search space are represented as vectors, and various genetic operators are operated over their bits of strings. However, unlike genetic algorithms, differential evolution carries out operations over each component (or each dimension of the solution). Almost everything is done in terms of vectors. For example, in genetic algorithms, mutation is carried out at one site or multiple sites of a chromosome, whereas in differential evolution, a difference vector of two randomly chosen population vectors is used to perturb an existing vector. Such vectorized mutation can be viewed as a more efficient approach from the implementation point of view. This kind of perturbation is carried out over each population vector and thus can be expected to be more efficient. Similarly, crossover is also a vector-based, component-wise exchange of chromosomes or vector segments.
Apart from using mutation and crossover as differential operators, DE has explicit updating equations. This also makes it straightforward to implement and to design new variants.

2.2. Different steps of Differential Evolution

Variation from one generation to the next is achieved by applying crossover and/or mutation operators. If both these operators are used, crossover is usually applied first, after which the generated offspring are mutated. For these algorithms, mutation step sizes are sampled from some probability distribution function. DE differs from these evolutionary algorithms in that mutation is applied first to generate a trial vector, which is then used within the crossover operator to produce one offspring, and mutation step sizes are not sampled from a prior known probability distribution function. In DE, mutation step sizes are influenced by differences between individuals of the current population.



Main stages of Differential Evolution

1. Initialization

A good uniform random initialization method is used to construct the initial population, the initial individuals will provide a good representation of the entire search space, with relatively large distances between individuals. Over time, as the search progresses, the distances between individuals become smaller, with all individuals converging to the same solution.

Evaluation of all the population and storing the fitness value of the same population in a vector.

2. Mutation

The DE mutation operator produces a trial vector for each individual of the current population by mutating a target vector with a weighted differential. This trial vector will then be used by the crossover operator to produce offspring. For each parent, $x_i(t)$, generate the trial vector, $u_i(t)$, as follows: Select a target

vector, $x_{i1}(t)$, from the population, such that $i \neq i_1$. Then, randomly select two individuals, x_{i2} and x_{i3}, from the population such that $i \neq i_1 \neq i_2 \neq i_3$ and $i_2, i_3 \sim U(1, n_s)$. Using these individuals, the trial vector is calculated by perturbing the target vector as follows:

$$u_i(t) = x_{i1}(t) + \beta*(x_{i2}(t) - x_{i3}(t))$$

Where $\beta \in (0, \infty)$ is the scale factor, controlling the amplication of the differential variation.

3. Crossover

The DE crossover operator implements a discrete recombination of the trial vector, $u_i(t)$, and the parent vector, $x_i(t)$, to produce offspring, $x'_i(t)$. Crossover is implemented as follows:

$$x'_{ij}(t) = \begin{cases} u_{ij}(t), & j \in J \\ x_{ij}(t), & otherwise \end{cases}$$

Where $x_{ij}(t)$ refers to the j^{th} element of the vector $x_i(t)$, and J is the set of element indices that will undergo perturbation (or in other words, the set of crossover points). Different methods can be used to determine the set, J, of which the following two approaches are the most frequently used:

- **Binomial crossover:** The crossover points are randomly selected from the set of possible crossover points, $\{1, 2, \ldots, n_x\}$, where n_x is the problem dimension. In this algorithm, p_r is the probability that the considered crossover point will be included. The larger the value of p_r, the more crossover points will be selected compared to a smaller value. This means that more elements of the trial vector will be used to produce the offspring, and less of the parent vector. Because a probabilistic decision is made as to the inclusion of a crossover point, it may happen that no points may be selected, in which case the offspring will simply be the original parent, $x_i(t)$. This problem becomes more evident for low dimensional search spaces. To enforce that at least one element of the offspring differs from the parent, the set of crossover points, J, is initialized to include a randomly selected point, $j*$

- **Exponential crossover:** inclusion of a crossover point, it may happen that no points may be selected, in which case the offspring will simply be the original parent, xi(t). This problem becomes more evident for low dimensional search spaces. To enforce that at least one element of the offspring differs from the parent, the set of crossover points, J , is initialized to include a randomly selected point, $j*$

4. Selection

Selection is applied to determine which individuals will take part in the mutation operation to produce a trial vector, and to determine which of the parent or the offspring will survive to the next generation. With reference to the mutation operator, a number of selection methods have been used. Random selection is usually used to select the individuals from which difference vectors are calculated. For most DE implementations the target vector is either randomly selected or the best individual is selected. To construct the population for the next generation, deterministic selection is used: the offspring replaces the parent if the fitness of the offspring is better than its parent; otherwise the parent survives to the next generation. This ensures that the average fitness of the population does not deteriorate.

2.3. Pseudo Code of DE

```
Begin
    Initialize the population with random values between (0 and 1)
    Denormalize the population

    Evaluate the fitness value of the population.
    Choose the best candidate from the population according to the fitness value and store it in best.
    Best_vector
    Best_fitness

    For i = 0 to num_its:
        For j = 0 to no_pop:
            Target_vector
            Target_fitness
            # Mutation
            Select randomly three candidates a, b, c such that a != b != c and not equal to target candidate.
            Create a mutant candidate using a, b, c.
            Mutant = a + mutation_factor*(b – c)

            #Crossover
            If rand() < Crossover_point then:
                Create a trial vector using mutant and target vector.
            End If
            Evaluate this trial vector and store the result in f.

            # Selection
            If f  < Target_fitness then:
                Target = trial
                Target_vector = f
                If f < Best_fitness then:
                    Best = trial
                End If
            End IF
        End For
    End For
    Return Best, Best_fitness
```

2.4. Applications of DE in Bioinformatics

Bioinformatics is an interdisciplinary field that develops and improves methods for storing, retrieving, organizing and analyzing biological data. A major activity in bioinformatics is to develop software tools to generate useful biological knowledge. As an interdisciplinary field of science, bioinformatics combines biology, computer science, information engineering and statistics to analyze and interpret biological data.

The term bioinformatics was coined by Paulien Hogeweg and Ben Hesper to describe "the study of informatic processes in biotic systems" and it found early use when the first biological sequence data began to be shared. Whilst the initial analysis methods are still fundamental to many large-scale experiments in the molecular life sciences, nowadays bioinformatics is considered to be a much broader discipline, encompassing modelling and image analysis in addition to the classical methods used for comparison of linear sequences or three-dimensional structures.

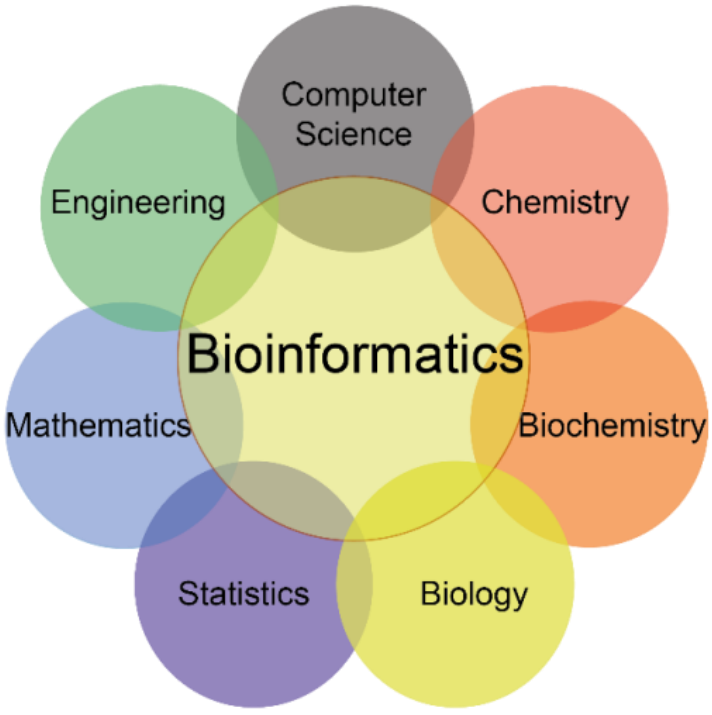


Fig.2.4. Different fields that make up Bioinformatics

The actual process of analyzing and interpreting data is referred to as computational biology. Important sub-disciplines within bioinformatics and computational biology include:

- Development and implementation of computer programs that enable efficient access to, use and management of, various types of information
- Development of new algorithms (mathematical formulas) and statistical measures that assess relationships among members of large data sets. For example, there are methods to locate a gene within a sequence, to predict protein structure and/or function, and to cluster protein sequences into families of related sequences.

The primary goal of bioinformatics is to increase the understanding of biological processes. What sets it apart from other approaches, however, is its focus on developing and applying computationally intensive techniques to achieve this goal. Examples include: pattern recognition, data mining, machine learning algorithms, and visualization. Major research efforts in the field include sequence alignment, gene finding, genome assembly, drug design, drug discovery, protein structure alignment, protein structure prediction, prediction of gene expression and protein–protein interactions, genome-wide association studies, the modeling of evolution and cell division/mitosis.

The Evolutionary Algorithms (EA) works on the principle of exploration and exploitation. The exploration starts from a random population. Later stages of process where the search is concentrated in a particular area, exploitation of search space yields the optimal solution. Compared to other global optimization techniques, evolutionary algorithms (EA) are easy to implement and very often they provide satisfactory solutions. A population of candidate solutions is initialized. New solutions are created by applying operators on the chosen parameters. The fitness or worthiness of the resulting solutions is evaluated and suitable selection strategy is then applied for the continuation of those solutions in the next iteration.

The general procedure for an evolutionary algorithm is as follows:

1. Given a population of individuals
2. Environmental pressure causes natural selection
3. Rise in fitness
4. For a fitness function, randomly create a set of candidate solutions
5. Based on fitness, some better candidates are chosen to seed the next generation.
6. Recombination – A procedure on two or more candidates which gives rise to two or more candidates.

7. Due to recombination, the newly formed candidates are entirely based on fitness.
8. The newly formed candidates replace the less fit individuals
9. The process is iterated.

In bioinformatics, if the problem is protein sequencing, the population would be set of proteins. The objective function would be maximizing the set of proteins that match with the DNA database.
Most of the bioinformatics applications are of multi-objective optimization problems. Algorithms like FASTLSA and FASTA can be easily applied with evolutionary procedure and they compare two sequences and the selection procedure can be decided the evolutionary way.
Most of the bioinformatics applications are of multi-objective optimization problems. Algorithms like FASTLSA and FASTA can be easily applied with evolutionary procedure and they compare two sequences and the selection procedure can be decided the evolutionary way.

2.5. Conclusion

Differential evolution (DE) is such a method that since its inception in 1995, DE has earned a reputation as a very effective global optimizer. While DE is not a panacea, its record of reliable and robust performance makes it one of the best optimizer.

While DE may not always be the fastest method, it is usually the one that produces the best result, although the number of cases in which it is also faster is significant. DE also proves itself to be robust, both in how control parameters are chosen and in the regularity with which it finds the true optimum.

In addition, when compared to one-point optimizers, DE is relatively immune to differences in initial populations. Because it is a direct search method, DE is versatile enough to solve problems whose objective functions lack the analytical description needed to compute gradients. DE is a good first choice when approaching a new and difficult global optimization problem is defined with continuous or discrete parameters.

References

[1] Rocca, P.; Oliveri, G.; Massa, A. (2011). "Differential Evolution as Applied to Electromagnetics". IEEE Antennas and Propagation Magazine. **53** (1): 38–49. doi:10.1109/MAP.2011.5773566.

[2] Storn, R.; Price, K. (1997). "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*. **11** (4): 341-359. Doi:10.1023/A:1008202821328

[3] Storn, R. (1996). "On the usage of differential evolution for function optimization". Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS). pp. 519–523.

[4] https://en.wikipedia.org/wiki/Differential_evolution

[5] https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/#

[6] https://www.ebi.ac.uk/training/online/course/bioinformatics-terrified-2018/what-bioinformatics

[7] http://www.nust.edu.pk/INSTITUTIONS/Centers/RCMS/ap/pg/MSBioinformatics/Pages/default.aspx

CHAPTER 3

SOLVING GRN USING DIFFERENTIAL EVOLUTION

3.1. Introduction

Inference of genetic regulatory networks from time series gene expression data has attracted attention, due to its importance in revealing fundamental cellular processes, investigating functions of genes, and understanding complex relations and interactions between genes. Large-scale gene expression data provide us with genome-wide information about the genetic regulatory networks that control basic biological processes such as development, disease, and the cell cycle. Time-series
gene expression data of such measurements allow us to visualize this dynamics directly as changing intensity patterns on gene chips. The goal of reverse engineering [1] techniques is to capture the pattern of regulatory excitation and inhibition amongst a set of genes and reconstruct their underlying genetic network.
Many types of linear or nonlinear mathematical models have been already proposed to infer gene regulatory networks from the time series microarray data i.e. a reverse engineering problem. Boolean networks [5], Dynamic Bayesian network [6], S-system [7], etc. were very popular methods to infer GRN. However in this project we have used Recurrent Neural Network (RNN) which is a closed loop Neural Network with a delayed feedback variable. It is very suitable to model the dynamics of genes and infer GRN from the temporal data. Generally, RNN along with an optimization technique is used to infer the GRN where the objective function of optimization is the training error. In a recurrent neural network, the current state of a neuron is determined by the previous states of all or most of the neurons in the network. As a result, the model provides dynamic aspect, which is most essential for the gene regulatory network. The weight between neurons gives the numeric interaction values between genes. Thereby, if it is possible to find those weight values between neurons from the time series data available for genes, then the real interaction between genes can be revealed.

We have implemented differential evolution algorithm (DE) for the reconstruction of GNR based on RNN. Though there are many techniques available to train recurrent neural network, such as back propagation through time, Genetic Algorithm (GA)[9], Particle Swarm Optimization (PSO)[10], K-means Population-Based Incremental Learning (KPBIL)[11], Bat Algorithm (BA)[12], hybridized Cuckoo Search (CS)-Flower Pollination Algorithm (FPA)[13], we have chosen DE because of its fast convergence rate and simplicity.

Traditional gradient descent-based methods are easily stuck in local minima and the computation of the derivatives is also not always possible. Here, the performance of the evolutionary computation technology-based method, known as differential evolution (DE) in training RNNs is investigated.

3.2. Proposed Methodology

A. Model used in the framework:

To the dynamic aspect of recurrent neural network, we have chosen the differential equation (1) as our model, where each gene expression is differentiated with respect to time.

$$\Gamma_i \frac{de_i}{dt} = \int \left(\sum_{j=1}^{N} w_{ji}\, e_j(t) - b_i \right) - d_i g_i \quad (1)$$

Let e_i is the expression of i^{th} gene, Γ_i is the time constant, w_{ji} be a weight from neuron *j* to neuron *i* in the neural net representation of a recurrent neural network ; w_{ji} can be positive, negative or zero depending on whether j^{th} gene is activating, inhibiting gene *i* or doesn't have any effect on it at all, 'b_i', 'd_i', and *'N'* represent the bias term for i^{th} gene, decay constant for e_i, and total number of genes present in the network, *f(z)=1/(1+e^{-z}), tanh and ReLU* are the nonlinear functions used to get the output of each gene. Here, *z* is the combined effect of all genes on i^{th} gene. To incorporate the discrete feature in our model with respect to time we have changed (1) as follows:

$$\Gamma_i \frac{de_i}{dt} = \int \left(\sum_{j=1}^{N} w_{ji}\, e_j(t) - b_i \right) - d_i g_i$$

$$\Gamma_i \frac{e_i(t+\Delta t) - e_i(t)}{\Delta t} = \int \left(\sum_{j=1}^{N} w_{ji}\, e_j(t) - b_i \right) - d_i g_i$$

$$e_i(t + \Delta t) = \frac{\Delta t}{\Gamma_i} \int \left(\sum_{j=1}^{N} w_{ji}\, e_j(t) - b_i \right) + e_i(t) \left(1 - \frac{\Delta t}{\Gamma_i} d_i \right) \quad (2)$$

Equation (2) demonstrates how expression of a particular gene changes with time in response to the other genes present in the network.

B. Synthesis of gene expression data

Here we attempted to generate artificial gene expression time series data to test the accuracy of our method. Using the model in equation (2), we have generated time series data using the parameter values of TABLE I of a 4-gene network.

TABLE - I

	Gene 1	Gene 2	Gene 3	Gene 4	b_i	Γ_i
Gene 1	20.0	-20.0	0.0	0.0	0.0	10.0
Gene 2	15.0	-10.0	0.0	0.0	-5.0	5.0
Gene 3	0.0	-8.0	12.0	0.0	0.0	5.0
Gene 4	0.0	0.0	8.0	-12.0	0.0	5.0

The interpretation of the above weight values are as follows. From TABLE I we can see that box (1, 2) = - 20.00, the meaning is that gene1 has -20.00 unit of effect
on gene2. We have chosen the same set of values as in paper [1] to compare the accuracy of our model. The generated time series data for the genes are shown in Fig.1

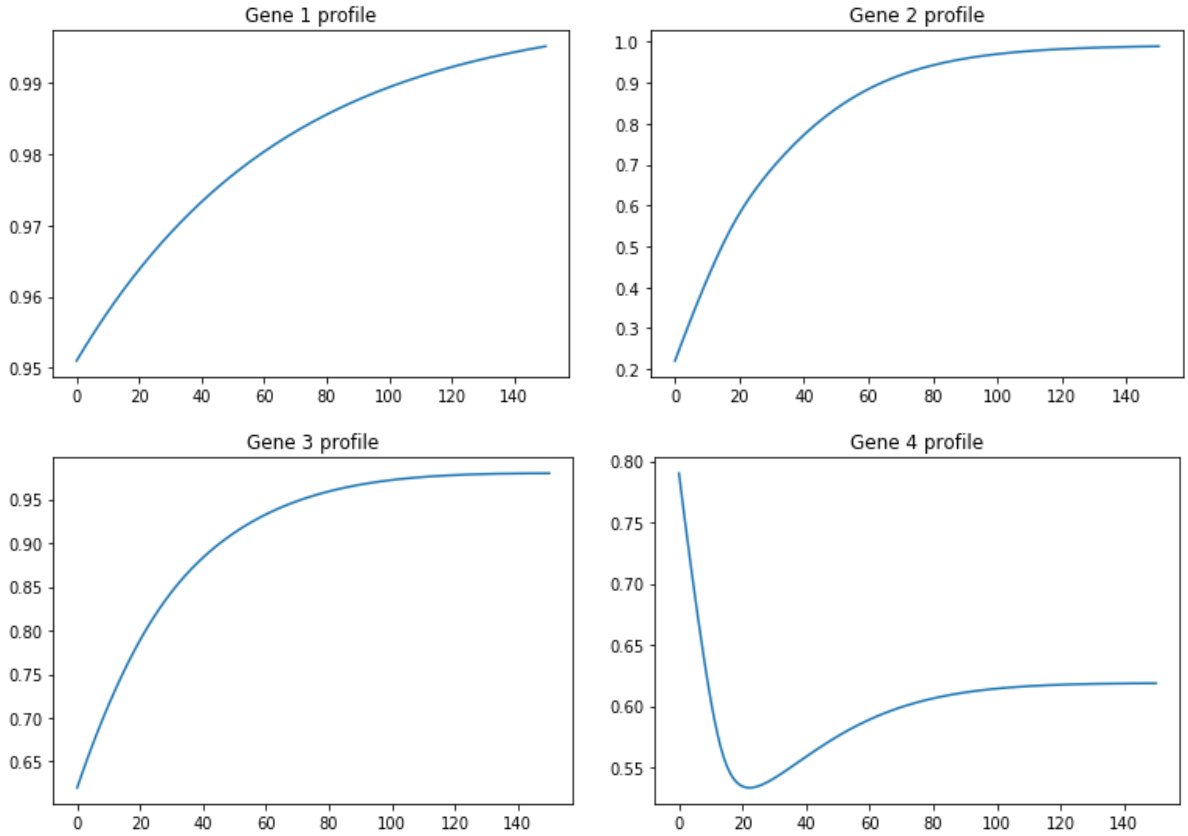


Fig.3.2. Expression profile of Gene1, Gene 2, Gene 3 and Gene 4 respectively.

It can be seen from Fig. 3.2 that nearly after150 points, the expression of all the genes get saturated, therefore this is the region from where we can extract maximum information. Because of this reason we have used 150 data points for each gene profile.

C. The cost function used

Accuracy of gene regulatory network (GRN) design mainly depends on two issues (i) how well we can measure the accuracy of the existing connection values of the network, and (ii) how well we can measure the accuracy of the skeletal structure (network topology) of the simulated network. Handling both issues simultaneously is a tough job, because we do not have any knowledge except the available gene expression time series data, which is also limited. To meet the first issue, we evaluated the accuracy of the produced gene expression of our simulated network by comparing it with the gene expression produced using the network parameters of TABLE I with the hope that if the network parameters of our simulated network is closer to the parameters of TABLE I then the difference (error) between these two set of gene expression will be less. That error has been calculated using the equation (3).

$$C_1 = \frac{1}{\mathrm{TNM}} \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{n}^{N} \left\{ \left[e_{org}^{n}(t) \right]_m - \left[e_{cal}^{n}(t) \right]_m \right\}^2 \quad (3)$$

Here M is the number of time series used; T is the number of data point in each time series data, and N is the number of gene present in the network. $\left[e_{org}^{n}(t) \right]_m$ is the original expression of nth gene at t^{th} time instance in m^{th} time series, and $[e_{cal}^{n}(t)]_m$ is the calculated expression of the same using our simulated network. The study of genetics reveals that in a gene regulatory network it is unlikely that all the genes interact with each other; rather few genes are involve in regulation of a gene. Considering this practical phenomenon, we designed the cost function given by equation (4).

$$C_2 = c \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\left| w_{ji} \right|}{\left| 1 + w_{ji} \right|} \quad (4)$$

Here w_{ji} is the connection value between gene j, and gene i, and c is a constant. Choosing proper value of c is also tricky. An appropriate value will lead to a good solution or it may mislead the system, its value should be such that C2 can't override C1. Our final cost function is shown in equation (5).

$$C = C_1 + C_2$$

$$C = \frac{1}{\text{TNM}} \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{n}^{N} \left\{ \left[e_{org}^{n}(t) \right]_m - \left[e_{cal}^{n}(t) \right]_m \right\}^2 + c \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{|w_{ji}|}{|1 + w_{ji}|} \quad (5)$$

Using this cost function, we will select the solution with the smallest cost value as the final solution i.e. if cost of solution1 is less than that of solution2 then solution1 is our final solution.

D. Differential Evolution (DE)

Consider a population of size N
The population matrix can be shown as:

$$x_{n,i}^{g} = \left[x_{n,1}^{g}, x_{n,2}^{g}, x_{n,3}^{g}, \ldots\ldots, x_{n,D}^{g} \right]$$

Where g is the generation, n = 1, 2,3, 4,……..,N and D parameters
The steps of Differential Evolution are as follows:

1. Initialization

Initial population is generated randomly between lower and upper bound.

$$x_{n,i} = x_{n,i}^{L} + rand() * \left(x_{n,i}^{U} - x_{n,i}^{L} \right) \quad i = 1, 2, 3,.., D \text{ and } n = 1, 2, 3,..,N$$

Where x_i^U is the upper bound of the variable x_i
Where x_i^L is the lower bound of the variable x_i

2. Mutation

From each parameter vector, select three other vectors $x_{r1n}^{g}, x_{r2n}^{g} \text{ and } x_{r3n}^{g}$ randomly.
Add the weighted difference of two of the vectors to the third

$$v_n^g = x_{r1n}^{g} + F\left(x_{r2n}^{g} - x_{r3n}^{g} \right) \qquad n = 1, 2, 3, \ldots, N$$

v_n^g is called donor vector

F is generally taken between 0 and 1

3. Recombination

A trial vector $U_{n,i}^{g+1}$ is developed from the target vector, $x_{n,i}^g$, and the donor vector, $V_{n,i}^{g+1}$

$$U_{n,i}^{g+1} = \begin{cases} V_{n,i}^{g+1} & if\ rand() \leq C_p\ or\ i = I_{rand} \\ x_{n,i}^g & if\ rand() > C_p\ and\ i \neq I_{rand} \end{cases} \quad \begin{matrix} i = 1, 2, 3, \dots, D \\ n = 1, 2, 3, \dots, N \end{matrix}$$

I_{rand} is a integer random number between [1, D]

C_p is the recombination probability

4. Selection

The target vector $x_{n,i}^g$ is compared with the trial vector $u_{n,i}^{g+1}$ and the one with the lowest function value is selected for the next generation.

$$x_n^{g+1} = \begin{cases} U_{n,i}^{g+1} & if\ f\left(U_n^{g+1}\right) < f\left(x_n^g\right) \\ x_n^g & otherwise \end{cases}$$

n = 1, 2, 3, …, N

3.3 Pseudo Code

INITIALIZE LIBRARIES
INITIALIZE PARAMETERS
Crossover ratio = CR
Mutation = mut
No of iteration = its
No of gene, No of parameters (NP), Upper and Lower bound
Initialize gene_cal

INITIALIZE POPULATION
Randomly initialize a population of NP individuals with each individual uniformly distributed in range [Upper, Lower]

FUNCTION DE (population):
Fitness = array of fitness of all population
Best = best individual having lowest fitness
Fitness(best) = fitness of the best individual

FOR I in range(NP): **do**
    I. MUTATION:
    select a target vector
    a, b , c = randomly select three individuals without replacement from the population
    IF(selected individuals are different): **THEN**
        Mutant vector = a + mutation factor*(b – c)
    END IF
    ELSE:
        Select again
    END ELSE

    II. CROSSOVER
    Generate trial vector
    IF crossover point > CR: **THEN**
        Replace elements between target vector and mutant vector and create a trial vector
    END IF

Evaluate the trial vector and calculate the fitness

III. SELECTION
IF fitness(trial_vector) < fitness(target vector): **THEN**
fitness(target vector) = fitness(trial vector)
target vector = trial vector
IF fitness(trial vector) < fitness(best): **THEN**
best = trial
END IF
END IF

RETURN best, fitness(best)

FUNCTION fitness(vector):
Call generate_gene_data(weight, bias, time_constant)
Fitness = $(gene\_original - gene\_calculated)^2$
Fitness / (no_of_gene)*time_point
RETURN fitness

FUNCTION calculate_gene_data(weight, bias, time_constant):
Calculate sum, apply sum to sigmoid function, subtract bias,
add gene(j)*weight
Calculate gene_org

3.4. Actual Code

```
#------------------------ Importing libraries -------------------------#
import numpy as np
import random
import matplotlib.pyplot as plt
np.random.seed(30)
random.seed(30)

#------------------------ Parameters initialization -------------------------#
no_gene = 4
no_param = 24
no_pop = 70
time_point = 150
delta = 0.01
CR = 0.8
mut = 0.4
its = 500
bounds = [(-20, 20)]*no_param

#------------------------ Gene Expression Data -------------------------#

gene_org = original gene data

#-----------------------Initialization-------------------------#

gene_org = np.array(gene_org)
gene_cal = np.zeros(gene_org.shape)

gene_cal[0][0] = 0.951
gene_cal[1][0] = 0.22
gene_cal[2][0] = 0.62
gene_cal[3][0] = 0.79

# Initializing the population with random values
pop = np.random.rand(no_pop, no_param)
```

```
#-----------------------Differential Evolution-----------------------#

def de(pop):

    dimensions = len(bounds)
    min_b, max_b = np.asarray(bounds).T
    diff = np.fabs(min_b - max_b)

    # Population Normalization
    pop_denorm = min_b + pop * diff

    # Calculation fitness of the population
    fitness = np.asarray([func(ind) for ind in pop_denorm])
    best_idx = np.argmin(fitness)
    best = pop_denorm[best_idx]

    for i in range(its):

        print('iteration-> {}'.format(i),'cost-> {}'.format(func(best)))

        for j in range(no_pop):

            # Mutation
            idxs = [idx for idx in range(no_pop) if idx != j]
            a, b, c = pop[np.random.choice(idxs, 3, replace = False)]
            mutant = np.clip((a + mut * (b - c)), 0, 1)

            # Crossover
            cross_points = np.random.rand(dimensions) < CR
            if not np.any(cross_points):
                cross_points[np.random.randint(0, dimensions)] = True

            trial = np.where(cross_points, mutant, pop[j])
            trial_denorm = min_b + trial * diff

            # Evaluation
            f = func(trial_denorm)
```

```
            # Selection
            if f < fitness[j]:
                fitness[j] = f
                pop[j] = trial
                if f < fitness[best_idx]:
                    best_idx = j
                    best = trial_denorm

        yield best, fitness[best_idx]


#---------------------------- Cost Function ----------------------------#
def func(pop):

    w = np.array(pop[0:16])
    b = np.array(pop[16:20])
    T = np.array(pop[20:24])
    total_cost1 = 0

    calculate_gene_exp(w, b, T)

    total_cost1 = np.sum(np.power((gene_org - gene_cal),2))

    total_cost1 /= no_gene*time_point

    return total_cost1


#---------------------------- Calculating gene data ----------------------------#
def calculate_gene_exp(w, b, T):

    for t in range(1, time_point):

        k = 0
        for i in range(no_gene):
```

```
        gene_cal[i][t] = gene_cal[i][t-1]*(1-(delta/T[i]))
        sum = 0
        for j in range(no_gene):
            sum += gene_cal[j][t-1]*w[k]
            k += 1

        sum -= b[i]

        fsum = 1/(1 + np.exp(-sum))

        gene_cal[i][t] += fsum*(delta/T[i])

a = list(de(pop))

m = np.linspace(0, 150, 150)

plt.figure(1)
plt.plot(m, gene_cal[0],label='calculated'), plt.plot(m, gene_org[0], '--', label =
'original')
plt.legend()
plt.title('Gene 1 profile')
plt.show()

plt.plot(m, gene_cal[1], label='calculated'), plt.plot(m, gene_org[1], '--', label =
'original')
plt.legend()
plt.title('Gene 2 profile')
plt.show()

plt.plot(m, gene_cal[2], label='calculated'), plt.plot(m, gene_org[2], '--', label =
'original')
plt.legend()
plt.title('Gene 3 profile')
plt.show()

plt.plot(m, gene_cal[3], label='calculated'), plt.plot(m, gene_org[3], '--', label =
'original')
```

```
plt.legend()
plt.title('Gene 4 profile')
plt.show()

m = np.linspace(0, 150, 150)

plt.plot(m, gene_cal[0])
plt.title('Gene 1 profile')
plt.show()

plt.plot(m, gene_cal[1])
plt.title('Gene 2 profile')
plt.show()

plt.plot(m, gene_cal[2])
plt.title('Gene 3 profile')
plt.show()

plt.plot(m, gene_cal[3])
plt.title('Gene 4 profile')
plt.show()

print('parameters = {}'.format((np.round(a[-1][0]))), 'Cost = {}'.format((a[-
1][1])))

# cr = cumulative error of each gene
cr1 = np.sum(np.fabs(gene_org[0]-gene_cal[0]))
cr2 = np.sum(np.fabs(gene_org[1]-gene_cal[1]))
cr3 = np.sum(np.fabs(gene_org[2]-gene_cal[2]))
cr4 = np.sum(np.fabs(gene_org[3]-gene_cal[3]))
print(cr1, cr2, cr3, cr4)
```

3.5. Experiments and Results

The experiment was carried out on AMD E1-1200 1.4Ghz APU powered laptop using Jupyter Notebook with Python 3 kernel. The code is written in Python 3 language. We made use of the numpy, random and matplotlib libraries for computation, random number generation and graph plotting. We chose python because it is very easy to use yet very powerful.
We applied DE code to two different data set, one having 4 genes and 150 time point data, second, having 8 genes and 50 time point data and calculated the result.
We used three different activation functions in our code and applied to each dataset. The three activation functions used by us are 1. Sigmoid, 2. Hyperbolic Tangent (tanh), 3. Rectified Linear Unit (ReLU).

4 GENE 150 TIME POINT DATASET

Sigmoid activation function:
We applied differential evolution (DE) code with sigmoid activation function $f(x) = 1/(1 + e^{-x})$ on 4 gene 150 time point dataset and calculated the result. The results are as follows:
I. Network and Weight values of the network:

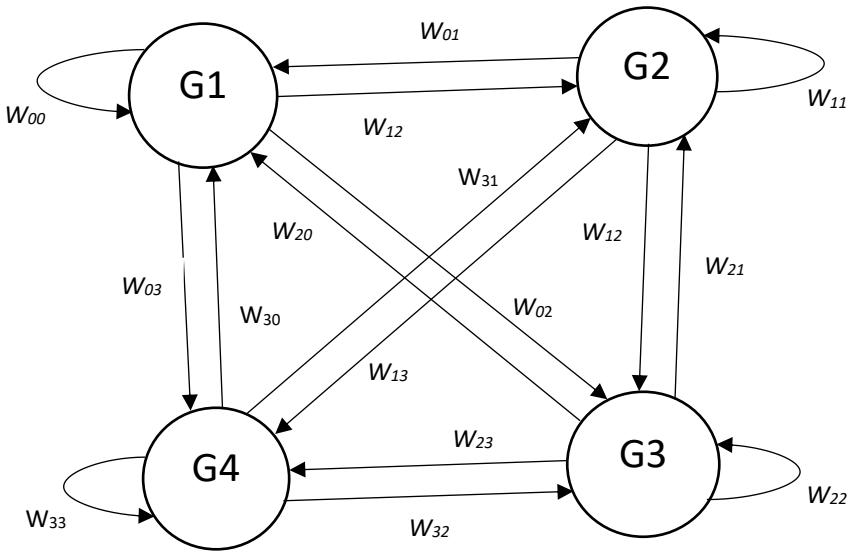
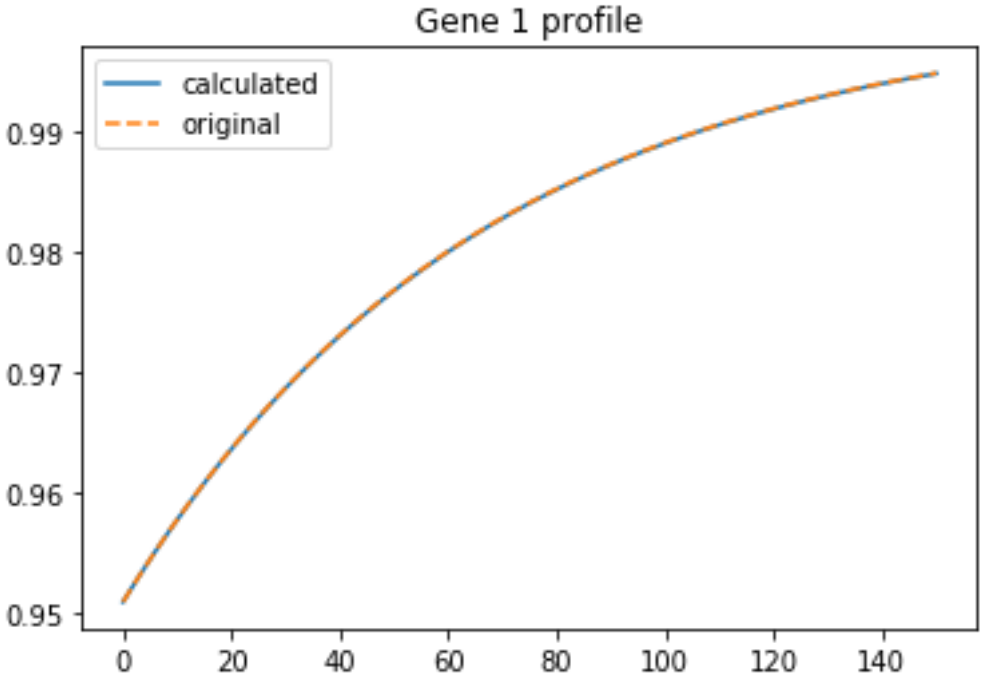


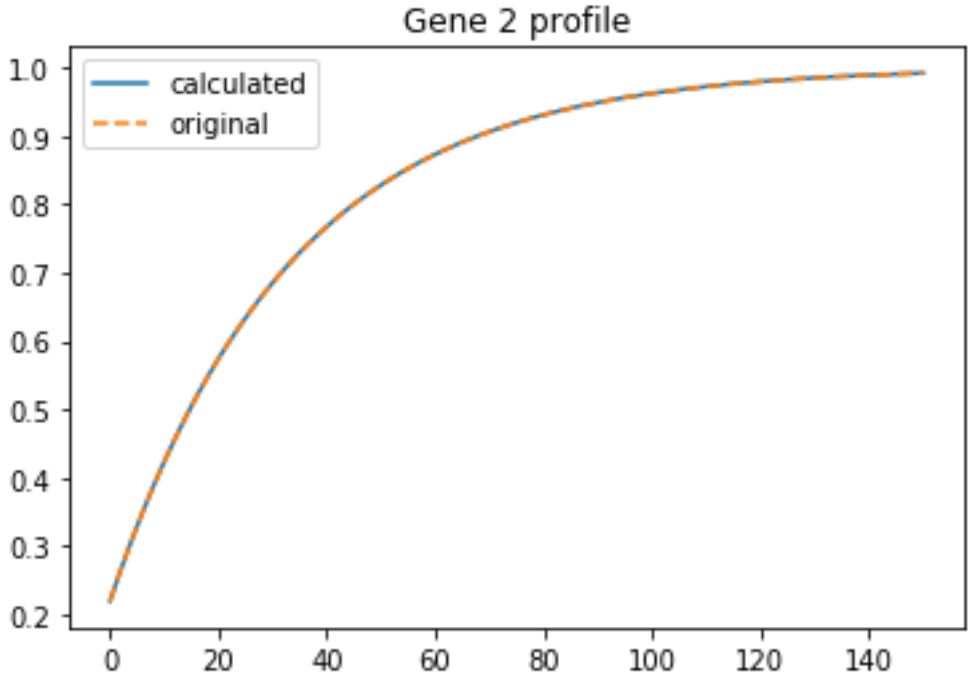
Fig. 3.4.1. Connections between genes with weights

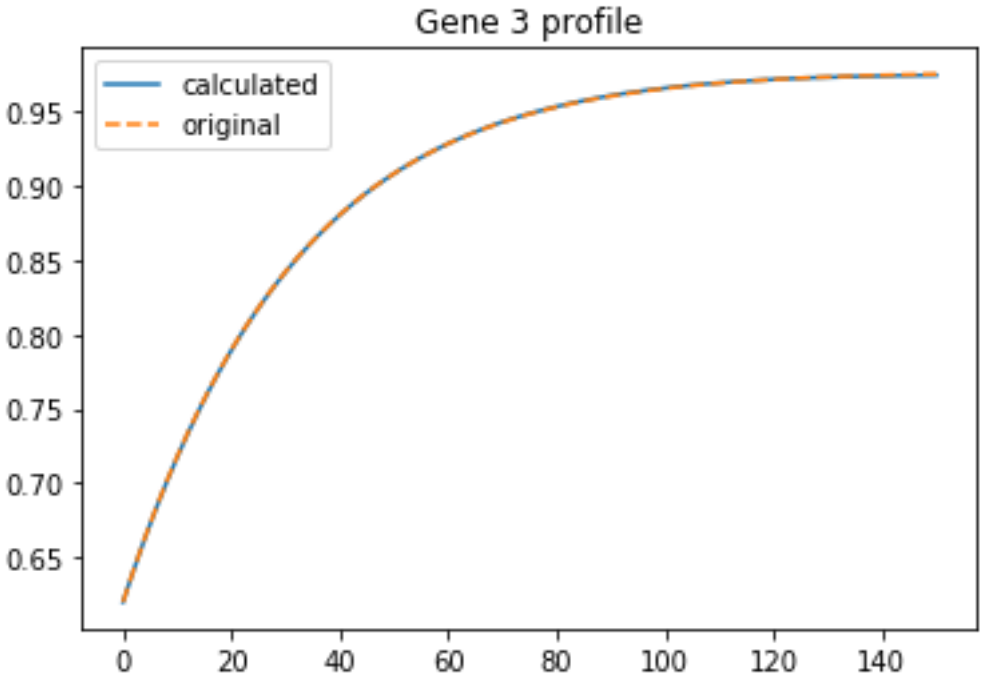
Table 1. Result after run using sigmoid function on 4 gene 150 time point data, 70 population

	Gene 1	Gene 2	Gene 3	Gene 4	b_i	Γ_i
Gene 1	-3.77	20.0	-7.61	3.08	-15.12	0.66
Gene 2	8.42	8.45	-1.98	7.00	-19.9	0.33
Gene 3	-18.26	-9.73	9.04	12.84	-14.83	0.34
Gene 4	-4.01	3.9	0.0	-12.25	-8.08	0.332

II. Gene profile graphs







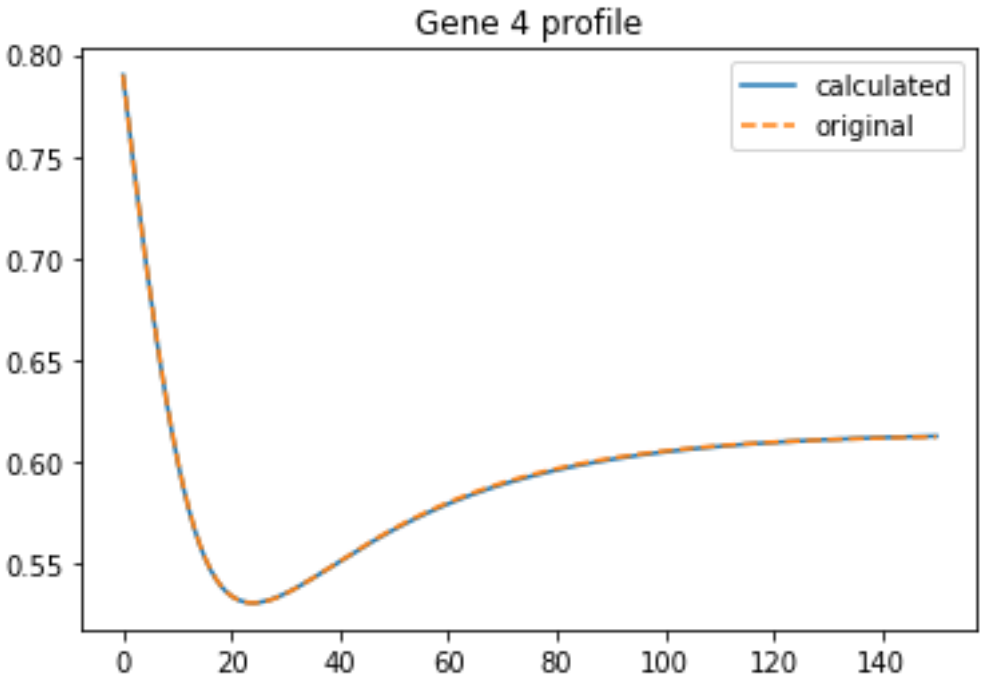


Fig. 3.4.1. Expression profile (obtained and original) of gene 1, gene 2, gene 3, gene 4 obtained using DE and sigmoid as an activation function

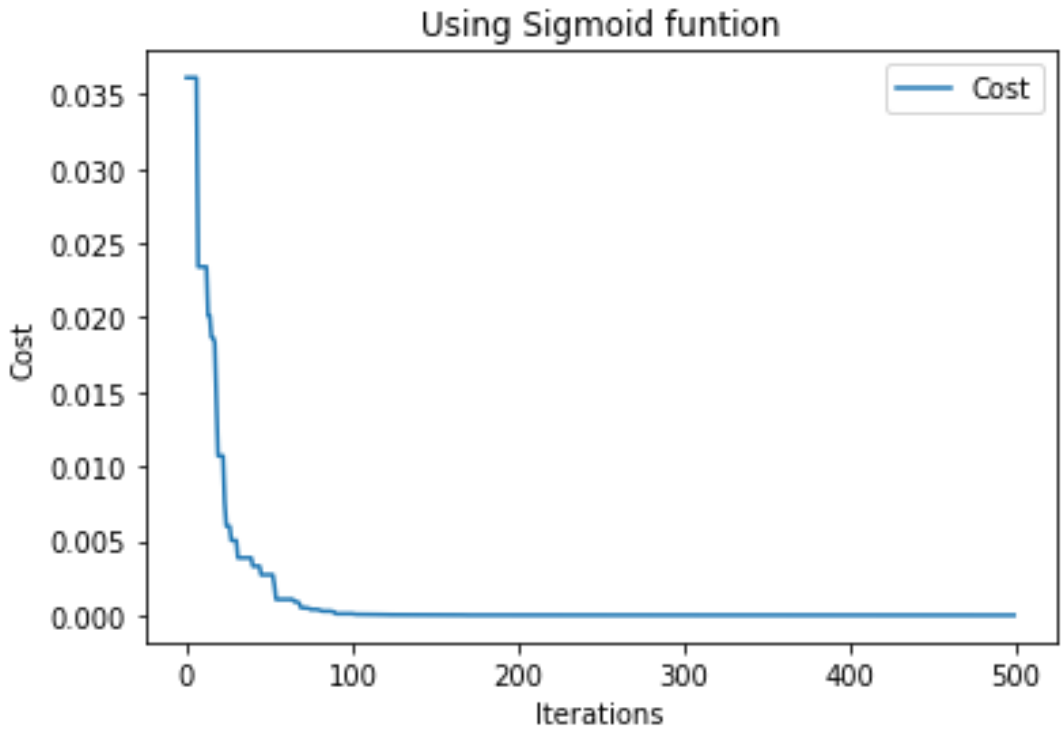


Fig. 3.4.2. Minimum cost function value in each step in DE using sigmoid activation function after 500 iterations

III. Cumulative error of each gene after the complete run of DE using sigmoid activation function.

Gene 1 = 0.005387648569680326
Gene 2 = 0.0024540599978749578
Gene 3 = 0.04065428762069223
Gene 4 = 0.05334962900011064

Tangent Hyperbolic (tanh) activation function:

Second, we applied differential evolution (DE) code with tangent hyperbolic (tanh) activation function $f(x) = {(e^x - e^{-x})}/{(e^x + e^{-x})}$ on 4 gene 150 time point dataset and calculated the result.

The results are as follows:

I. Network and Weight values of the network:

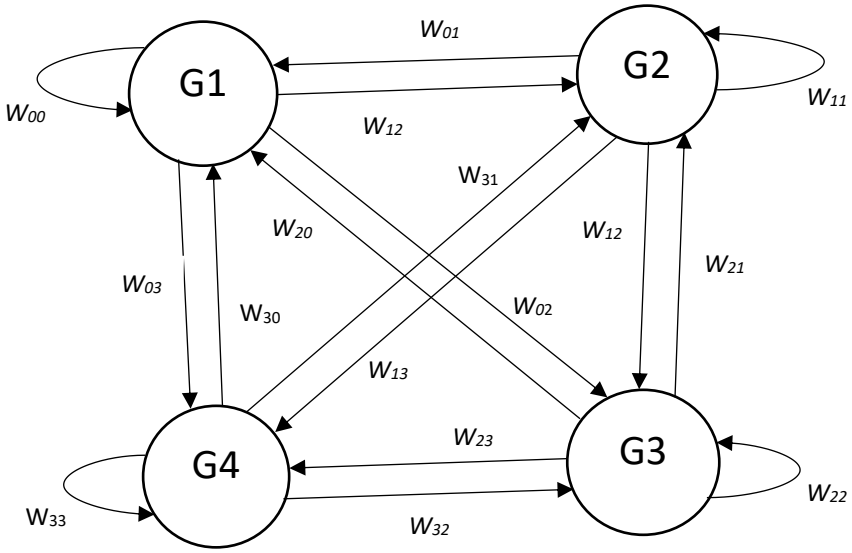
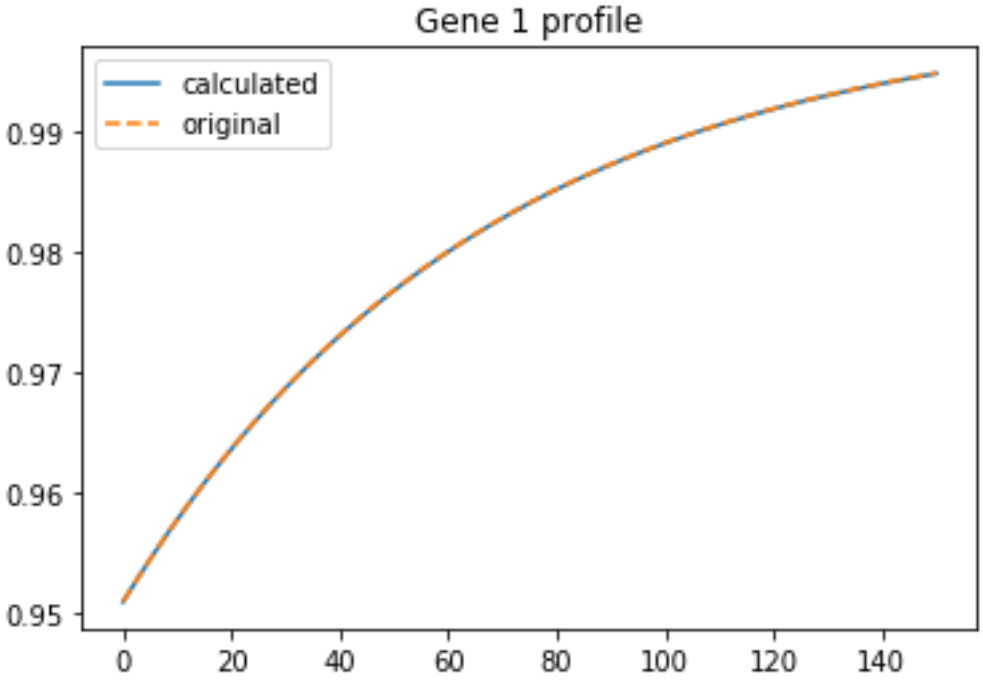


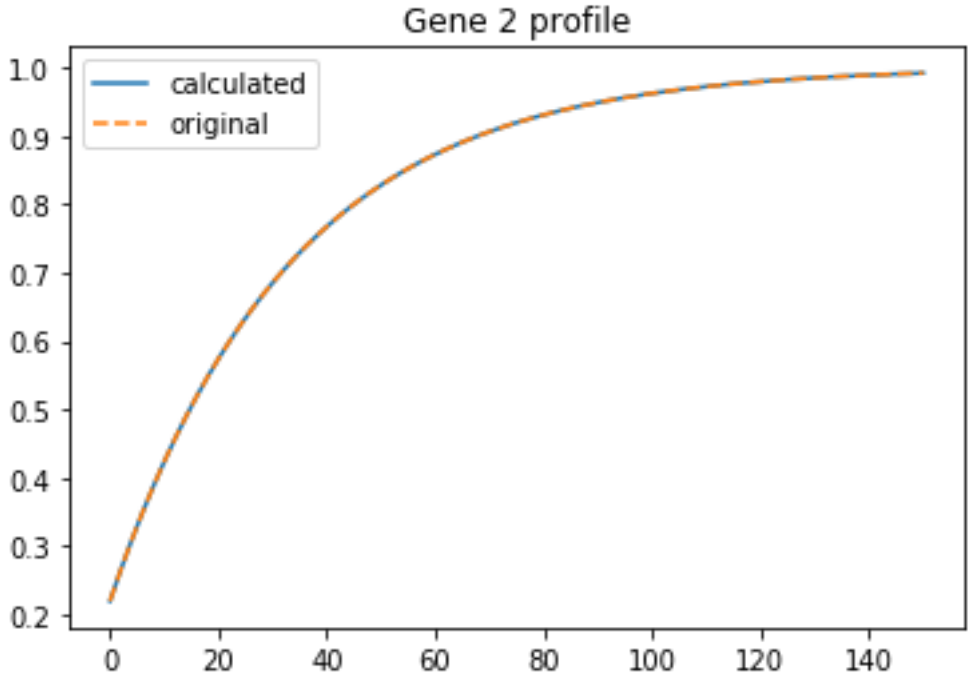
Fig. 3.4.1. Connections between genes with weights

Table 3. Result after run using tanh function on 4 gene 150 time point data, 70 population

	Gene 1	Gene 2	Gene 3	Gene 4	b_i	Γ_i
Gene 1	19.0	-16.0	-15.29	14.33	-20.0	0.65
Gene 2	8.77	20.0	17.83	-5.83	-18.37	0.33
Gene 3	17.47	4.48	2.78	-7.79	18.30	-1.97
Gene 4	7.715	4.20	-0.831	-20.0	-1.91	0.90

II. Gene profile graphs





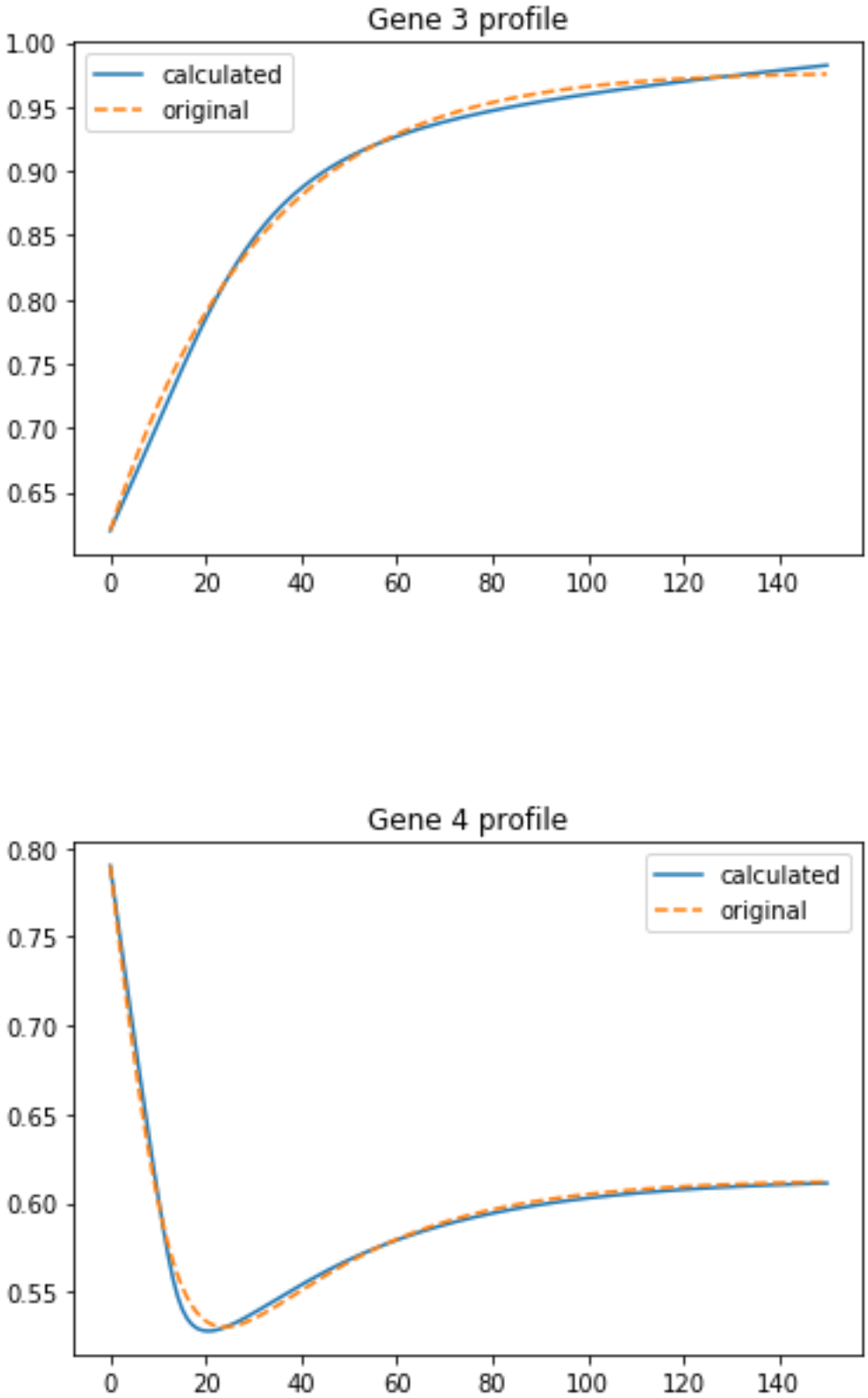


Fig. 3.4.1. Expression profile (obtained and original) of gene 1, gene 2, gene 3, gene 4 obtained using DE and tanh as an activation function

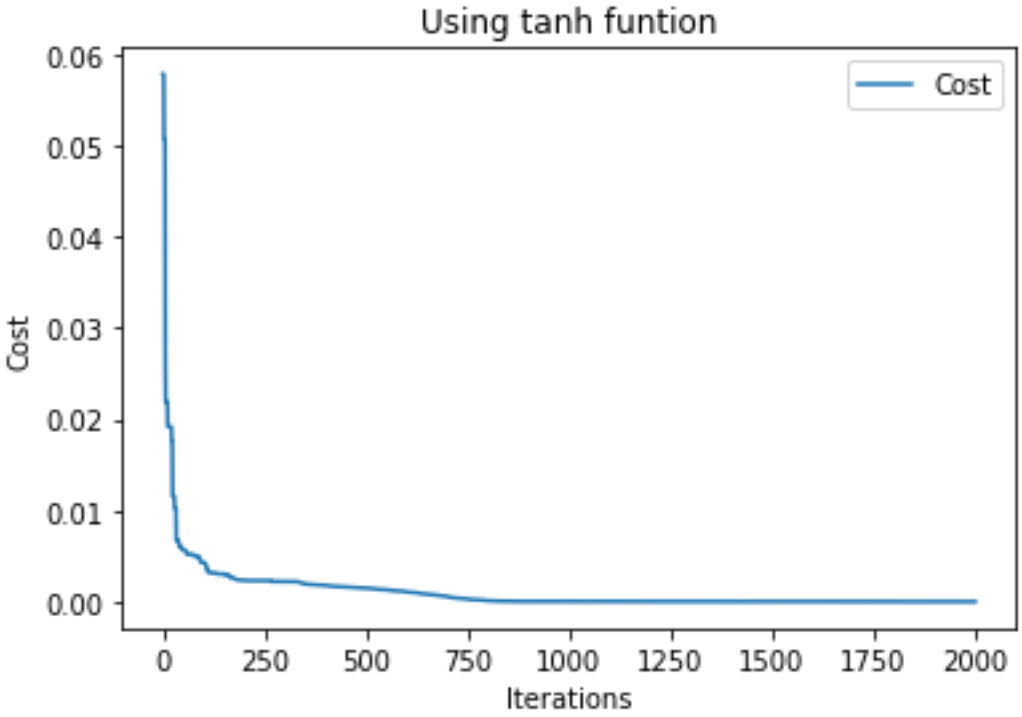


Fig. 3.4.2. Minimum cost function value in each step in DE using tanh activation function after 2000 iterations

III. Cumulative error of each gene after the complete run of DE using tanh activation function.

Gene 1 = 0.005049140885144965
Gene 2 = 0.009772150626521758
Gene 3 = 0.7326788134764473
Gene 4 = 0.39660406731284326

Rectified Linear Unit (ReLU) activation function:

Third, we applied differential evolution (DE) code with rectified linear unit (ReLU) activation function $f(x) = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$ $or\ max(0,\ x)$ on 4 gene 150 time point dataset and calculated the result.

The results are as follows:

I. Network and Weight values of the network:

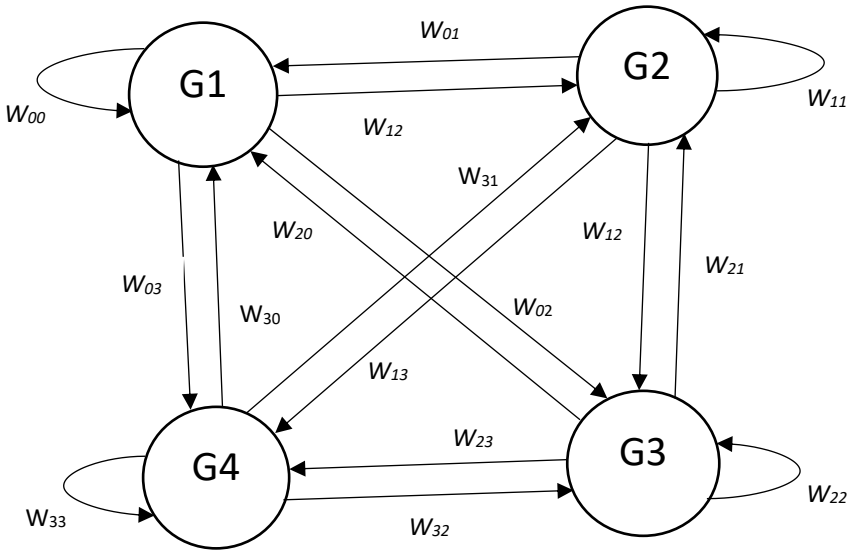
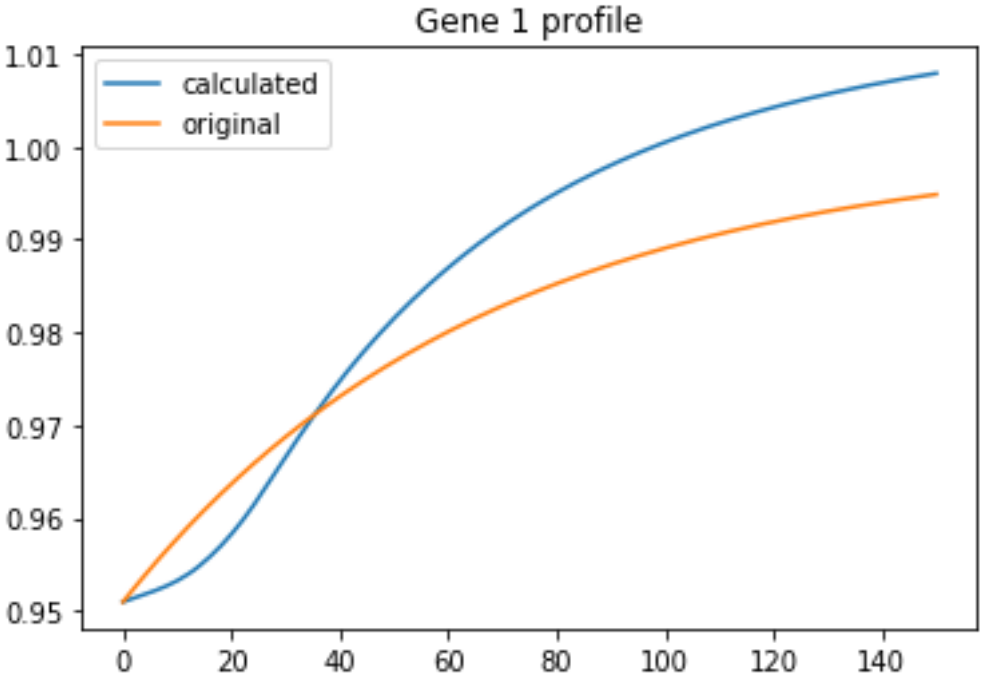


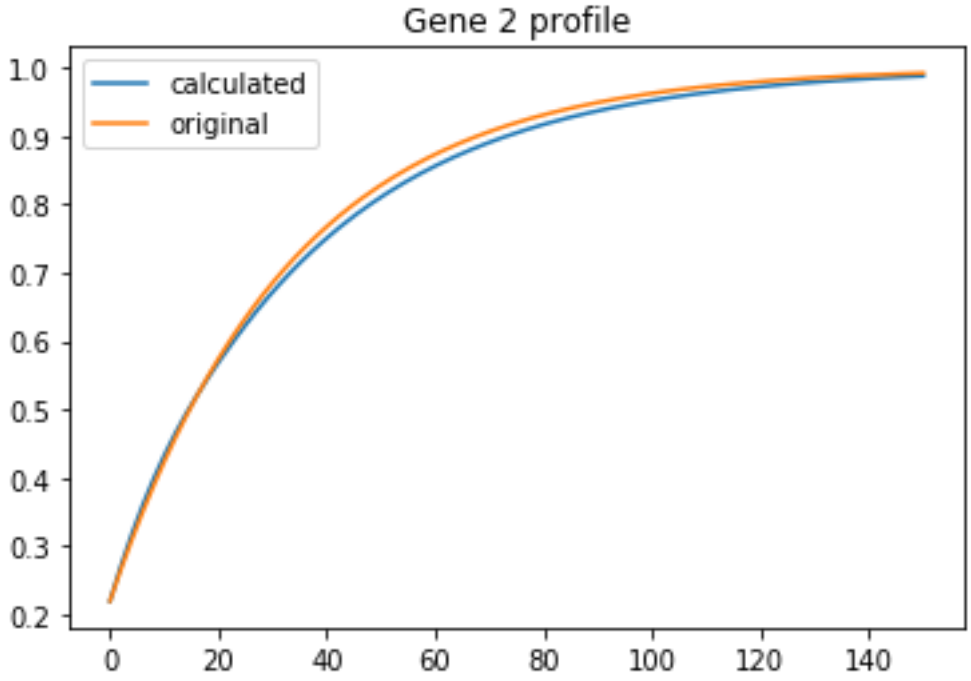
Fig. 3.4.1. Connections between genes with weights

Table 3. Result after run using ReLU activation function on 4 gene 150 time point data, 70 population

	Gene 1	Gene 2	Gene 3	Gene 4	b_i	Γ_i
Gene 1	-17	-9	19	-2	-9	16
Gene 2	12	-20	10	11	-17	6
Gene 3	5	-5	18	-4	14	-1
Gene 4	19	3	-5	-12	10	0.33

II. Gene profile graphs





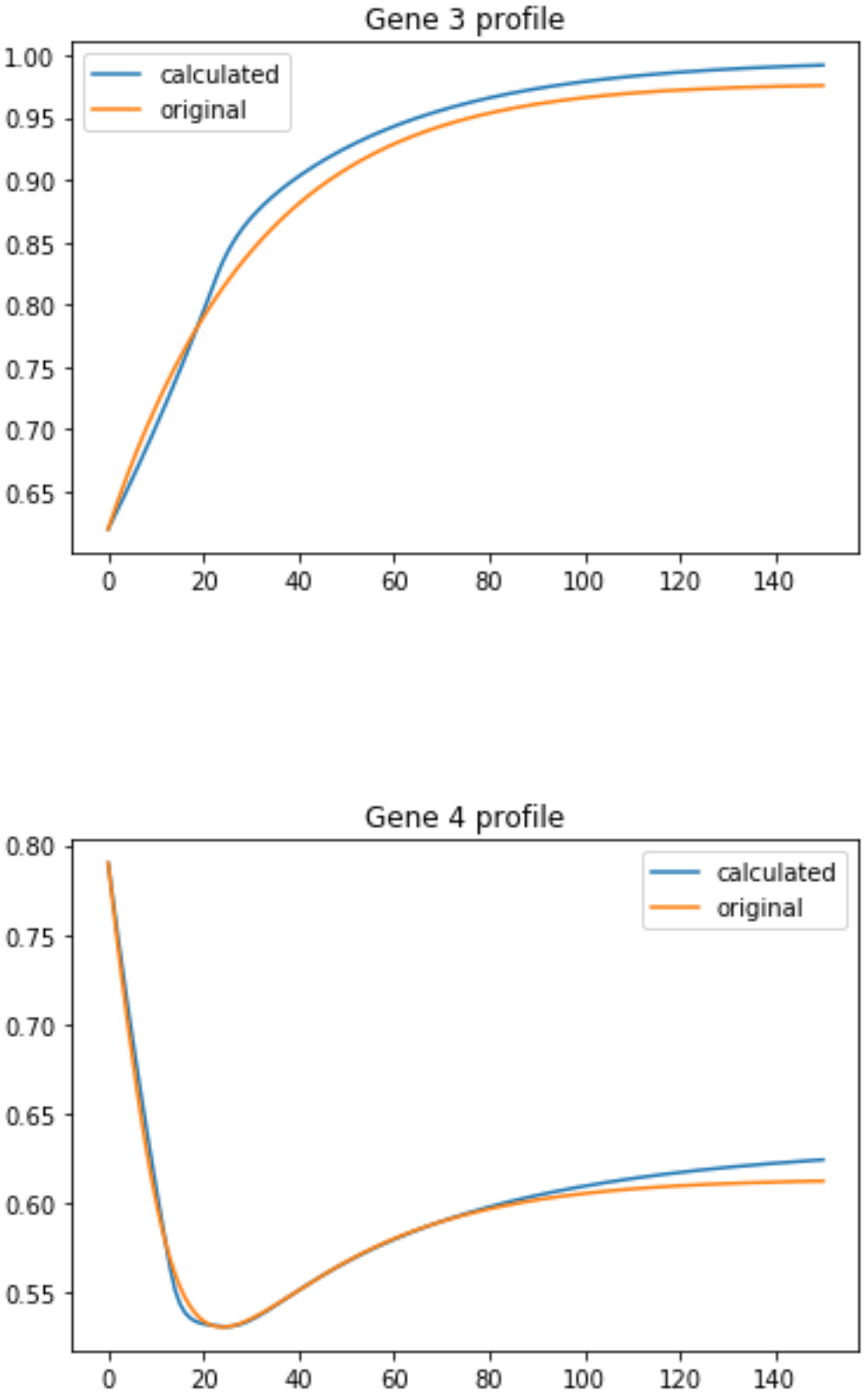


Fig. 3.4.1. Expression profile (obtained and original) of gene 1, gene 2, gene 3, gene 4 obtained using DE and ReLU as an activation function

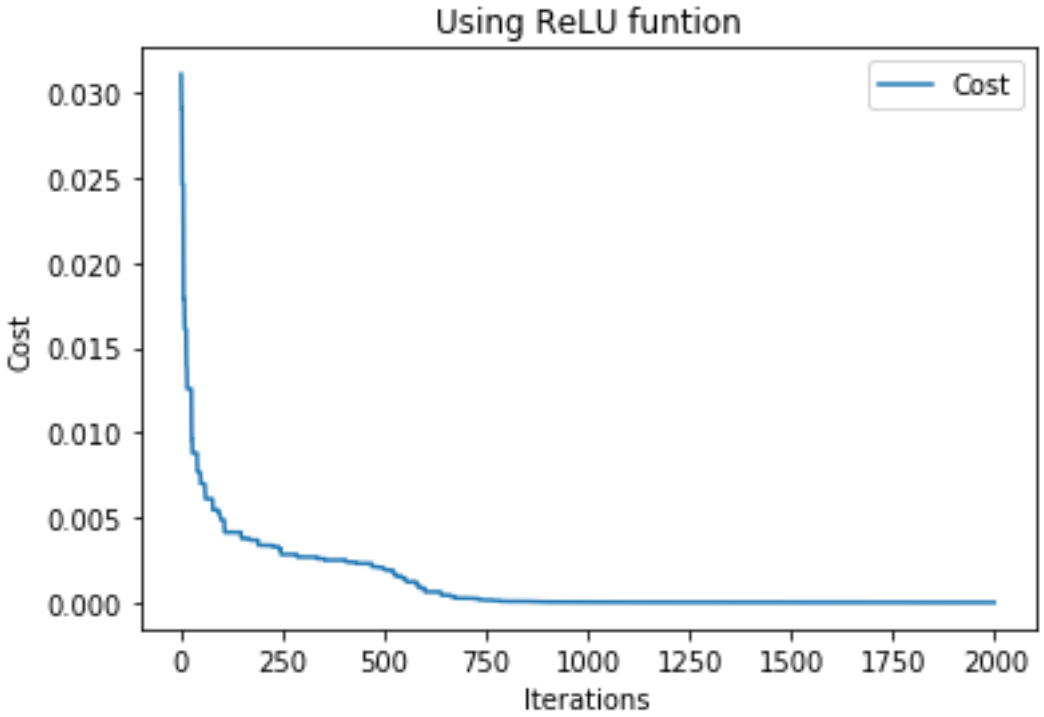


Fig. 3.4.2. Minimum cost function value in each step in DE using ReLU activation function after 2000 iterations

III. Cumulative error of each gene after the complete run of DE using sigmoid activation function.

Gene 1 = 1.219748937526846
Gene 2 = 1.620647139444918
Gene 3 = 2.2344797549153337
Gene 4 = 0.6278830194131982

8 GENE 50 TIME POINT DATASET

Sigmoid activation function:

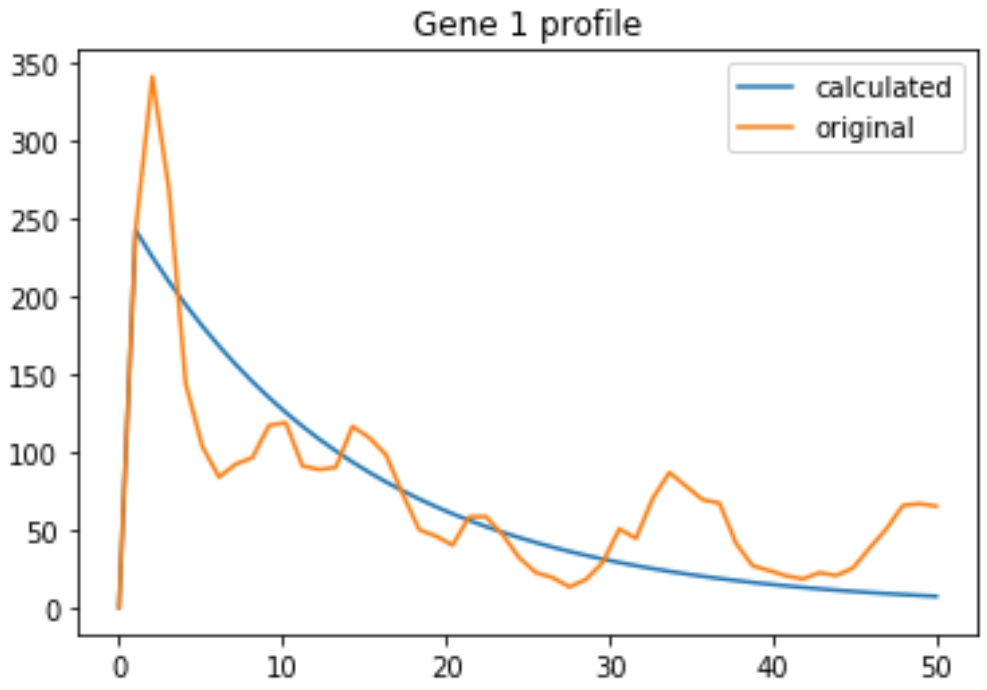
First, we applied differential evolution (DE) code with sigmoid activation function $f(x) = 1/(1 + e^{-x})$ on 8 gene 50 time point dataset and calculated the result.

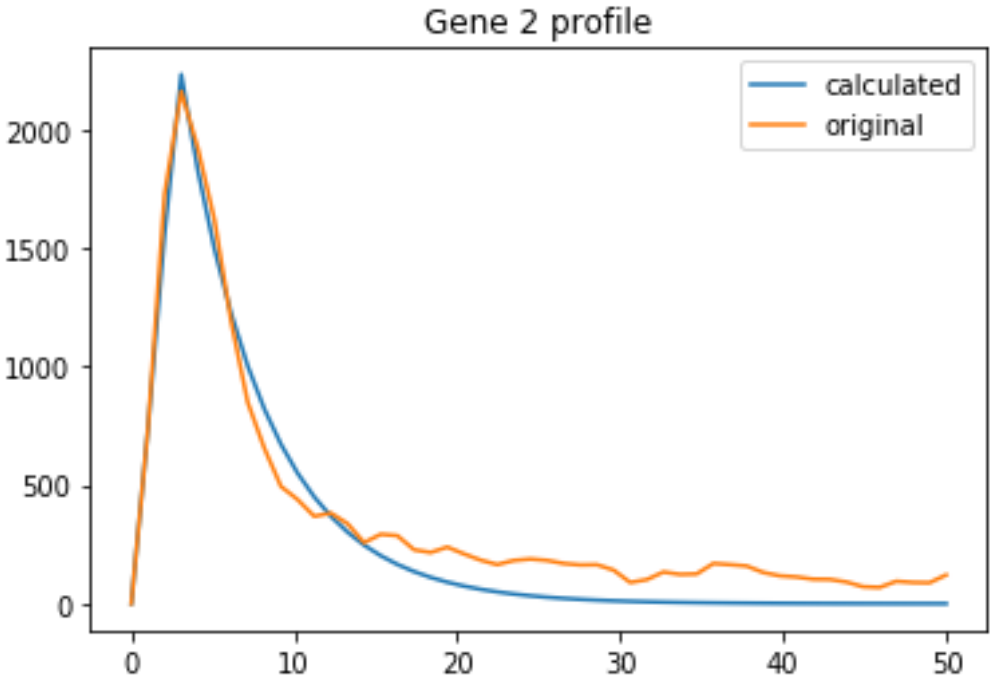
The results are as follows:

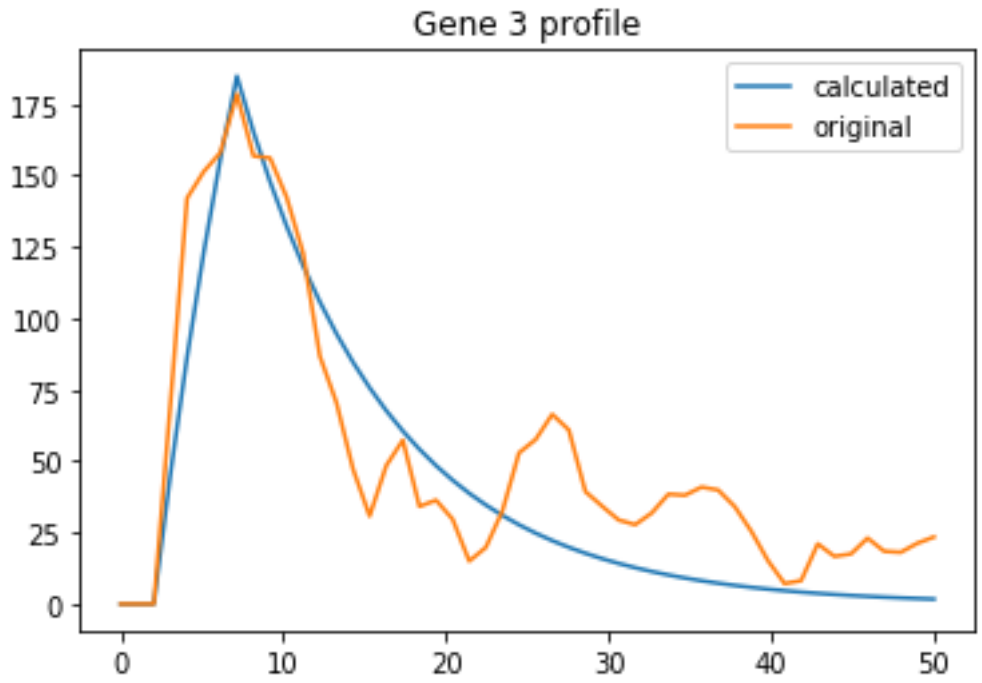
I. Weight values of the network:

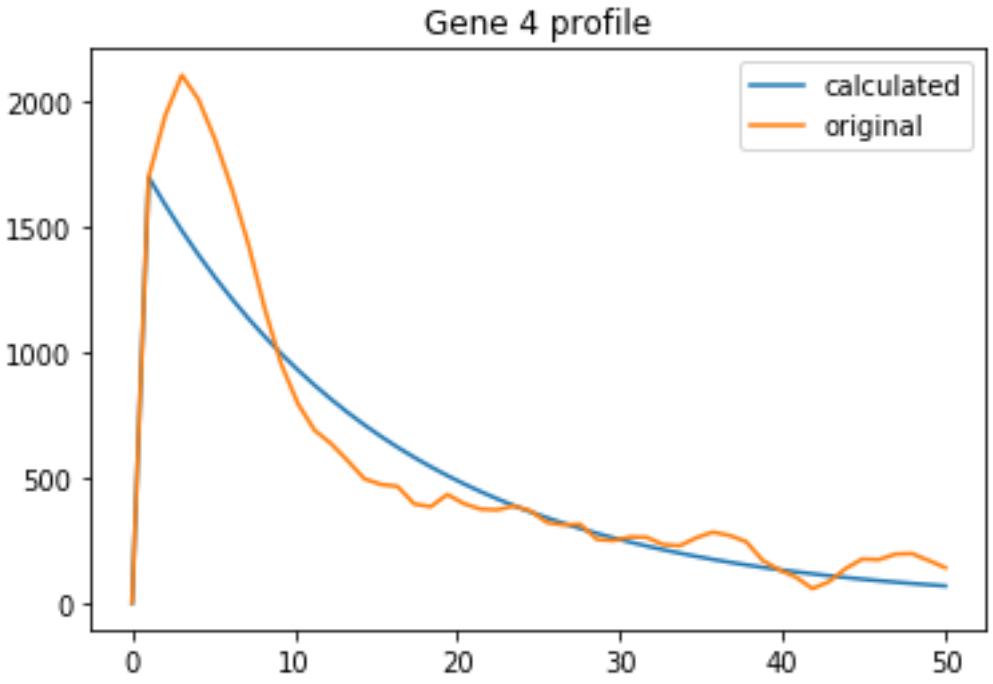
	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7	Gene 8	b_i	Γ_i
Gene 1	-27.9	5.60	-20.7	30.0	-17.5	-27.8	-30.0	30.0	28.71	-3.54
Gene 2	3.83	-1.31	12.55	-19.9	29.85	-26.9	-29.5	-10.0	-28.8	0.001
Gene 3	-27.5	25.1	-29.0	2.04	-29.2	29.23	-1.34	-27.4	-23.1	0.002
Gene 4	-26.5	1.55	-16.0	-22.9	-16.7	27.84	-27.9	3.58	-7.87	-4.40
Gene 5	4.61	22.48	-26.0	28.89	28.13	-19.9	22.44	-22.1	-28.8	2.53
Gene 6	-29.9	8.79	-13.5	-19.5	-27.7	27.7	-7.79	16.79	-28.2	-0.28
Gene 7	-20.8	3.48	26.87	19.45	18.54	15.46	-30.0	26.51	-4.78	0.001
Gene 8	-29.4	14.49	-0.21	-28.5	-26.4	-28.6	27.73	22.44	29.20	14.17

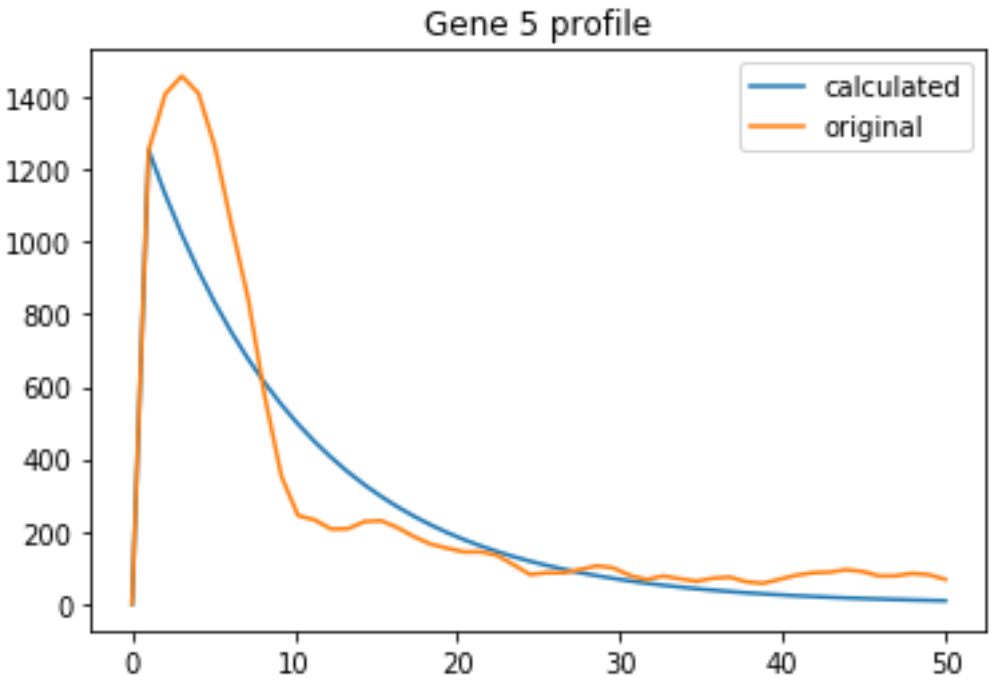
II. Gene profile graphs:

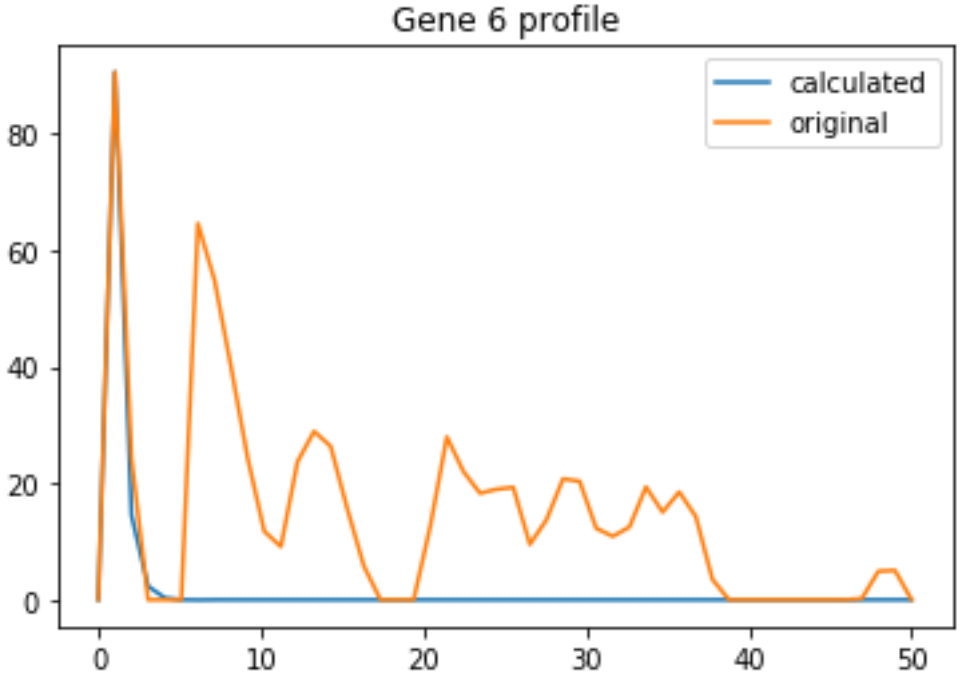





Gene 3 profile
calculated
original
175
150
125
100
75
50
25
0
0
10
20
30
40
50


Gene 4 profile
calculated
original
2000
1500
1000
500
0
0
10
20
30
40
50


Gene 5 profile
calculated
original
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50


Gene 6 profile
calculated
original
80
60
40
20
0
0
10
20
30
40
50

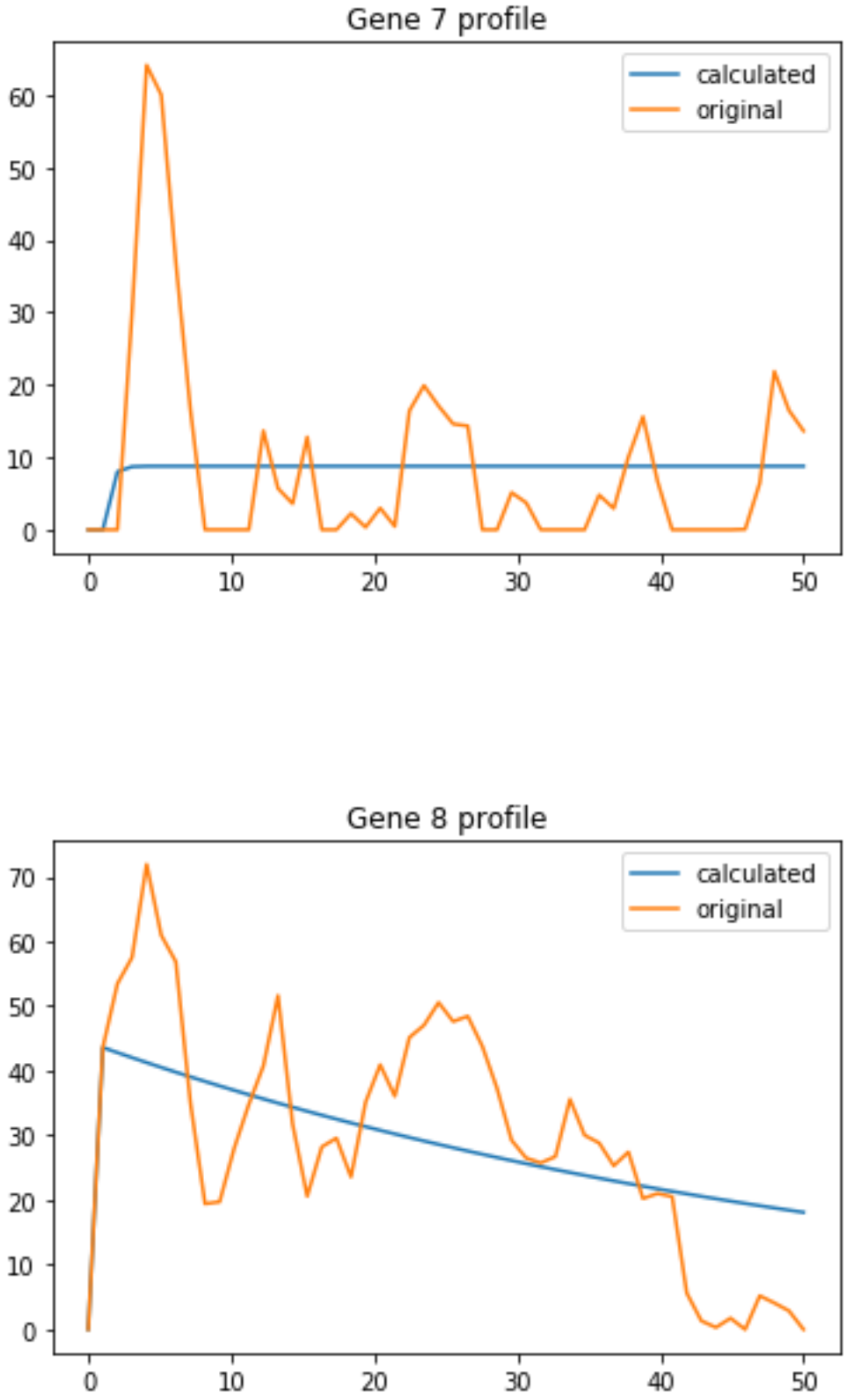


Fig. 3.4.1. Expression profile (obtained and original) of gene 1, gene 2, gene 3, gene 4, gene 5, gene 6, gene 7, gene 8 obtained using DE and sigmoid as an activation function

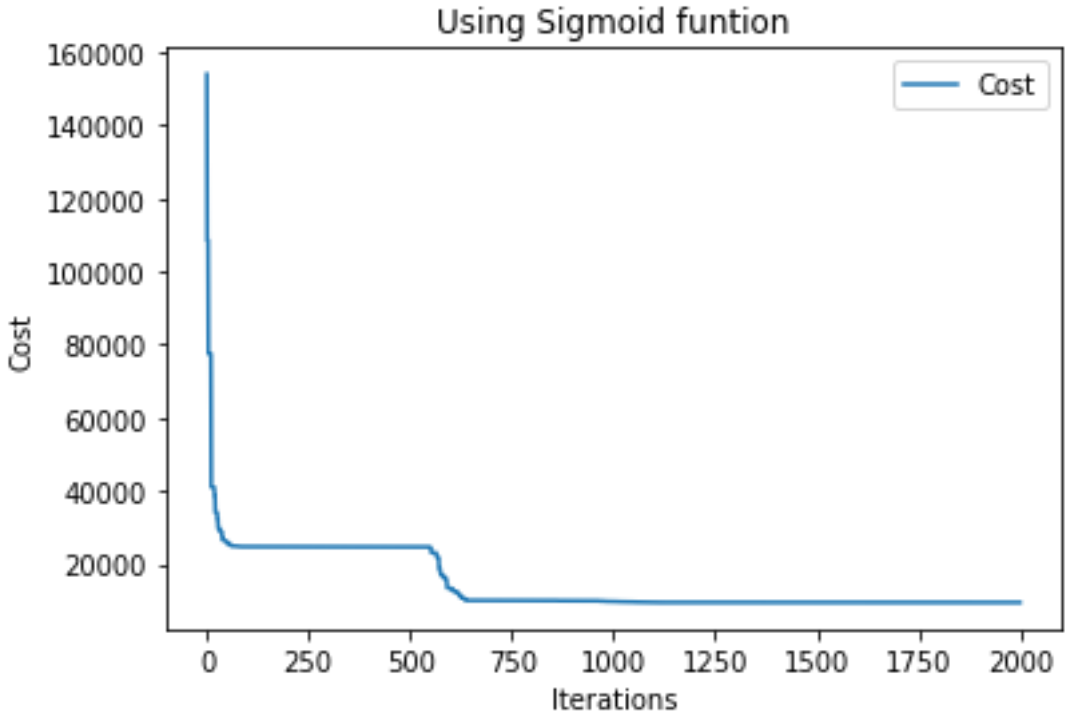


Fig. 3.4.2. Minimum cost function value in each step in DE using sigmoid activation function after 2000 iterations

III. Cumulative error of each gene after the complete run of DE using sigmoid activation function.

Gene 1 = 1444.6383413566189
Gene 2 = 5399.367133105187
Gene 3 = 942.014392573654
Gene 4 = 6195.347531207003
Gene 5 = 4690.470643825132
Gene 6 = 617.47679739405
Gene 7 = 471.1232009093766
Gene 8 = 536.7045559935261

Tangent Hyperbolic (tanh) activation function:

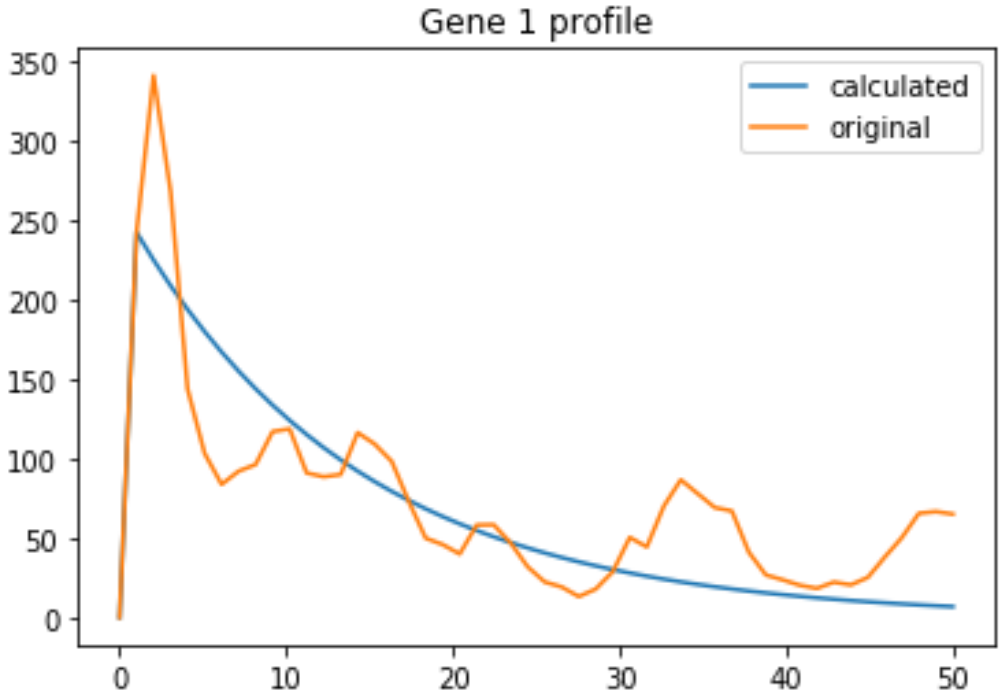
Second, we applied differential evolution (DE) code with tangent hyperbolic (tanh) activation function $f(x) = {(e^x - e^{-x})}/{(e^x + e^{-x})}$ on 8 gene 50 time point dataset and calculated the result.

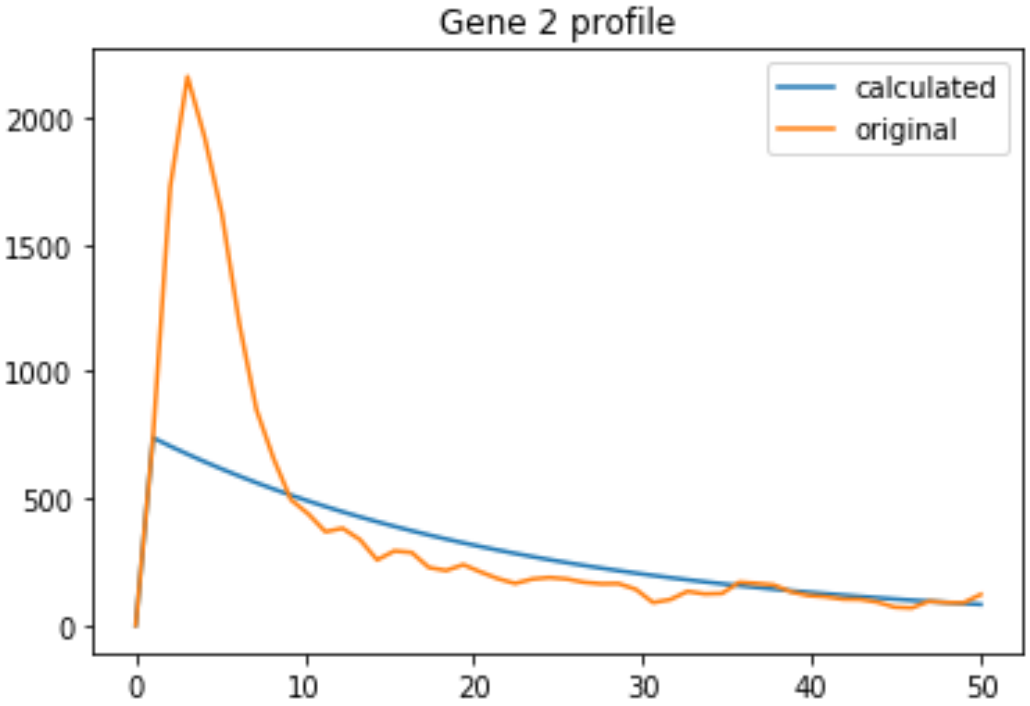
The results are as follows:

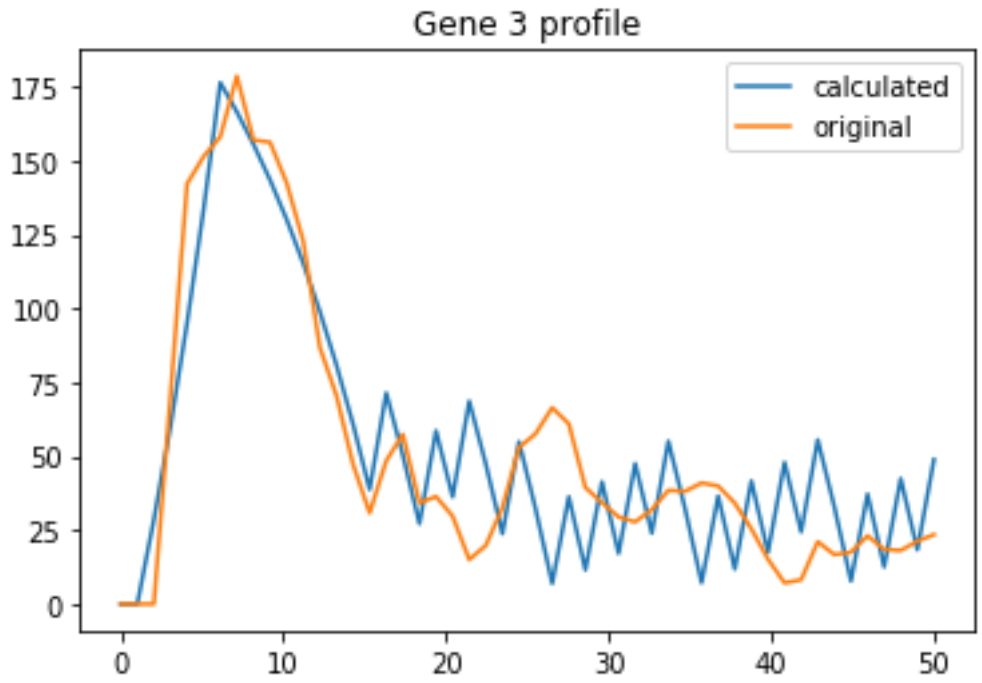
I. Weight values of the network:

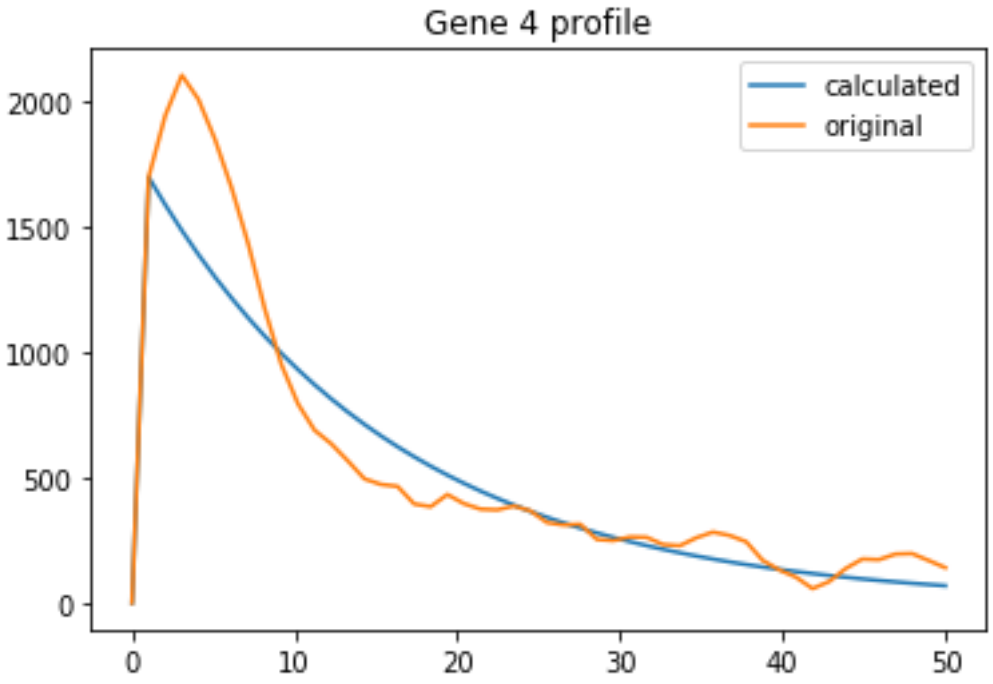
	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7	Gene 8	b_i	Γ_i
Gene 1	-21.3	-24.4	-4.76	-22.5	-18.3	-16.8	-13.5	22.40	-19.0	-2.48
Gene 2	6.0	-13.3	28.01	23.47	-27.1	-2.45	-17.1	-20.1	27.47	-5.91
Gene 3	-27.3	-18.5	26.04	21.19	-18.3	-24.8	3.507	-16.5	-24.9	-0.003
Gene 4	-6.12	-28.0	-0.78	19.02	-25.2	-29.0	-30.0	-29.9	21.95	-3.43
Gene 5	28.43	27.56	8.52	4.98	-23.0	-20.5	-28.0	7.74	-10.5	2.47
Gene 6	-23.8	8.147	26.14	29.28	-5.67	28.72	-8.20	12.20	-17.2	0.41
Gene 7	1.76	-5.24	13.00	-8.52	-1.77	29.07	-14.6	-25.4	24.03	-0.001
Gene 8	-24.9	23.85	-30.0	-11.5	-3.67	29.58	-18.6	14.84	27.23	-14.8

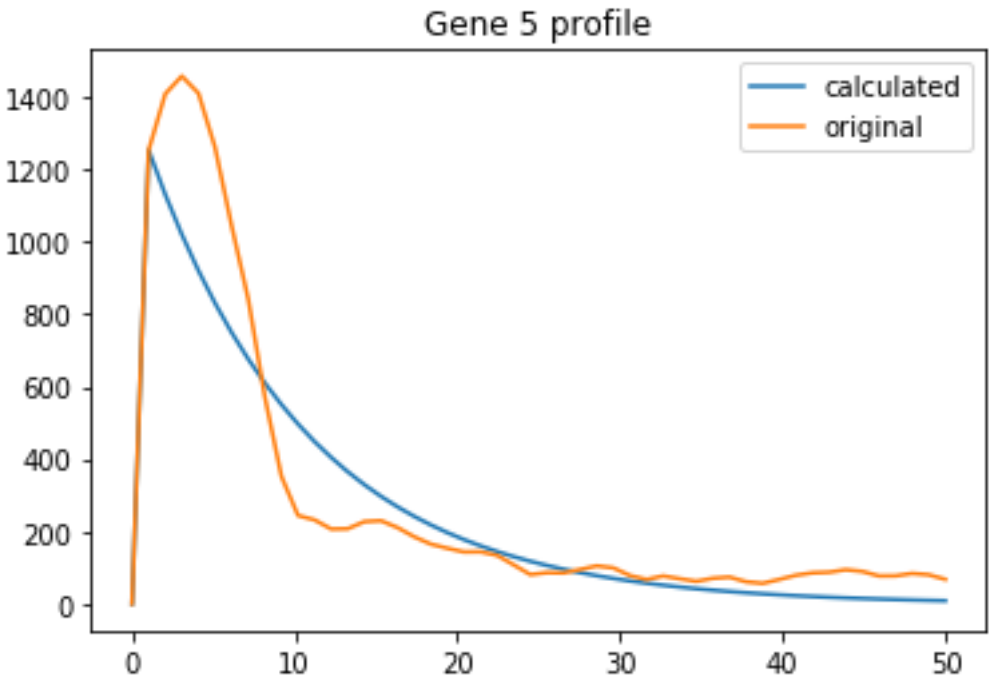
I. Gene profile graphs:

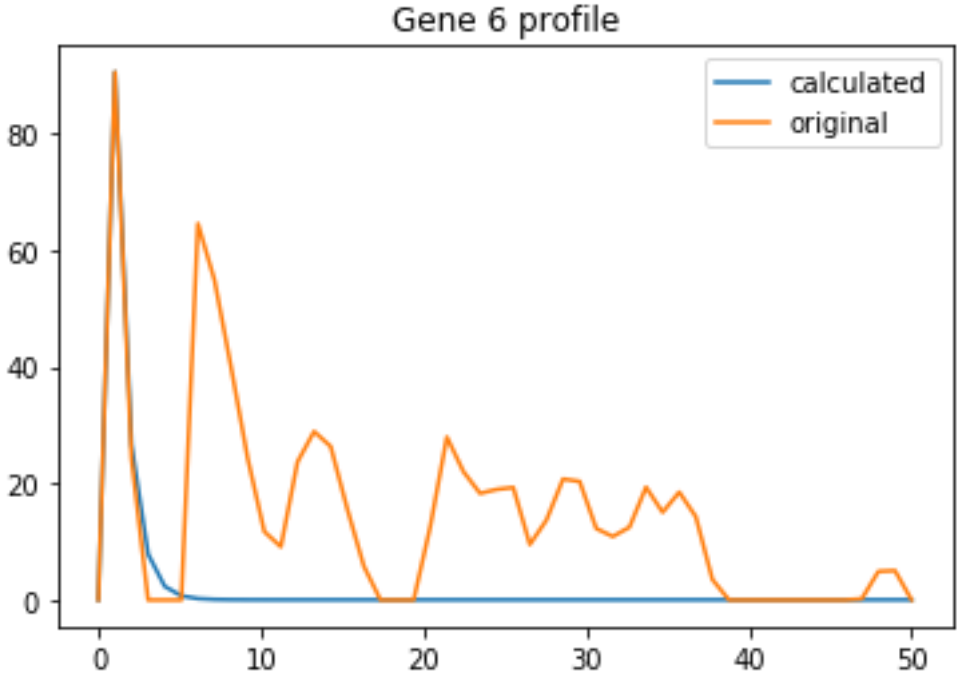





Gene 3 profile
calculated
original
175
150
125
100
75
50
25
0
0
10
20
30
40
50


Gene 4 profile
calculated
original
2000
1500
1000
500
0
0
10
20
30
40
50


Gene 5 profile
calculated
original
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50


Gene 6 profile
calculated
original
80
60
40
20
0
0
10
20
30
40
50

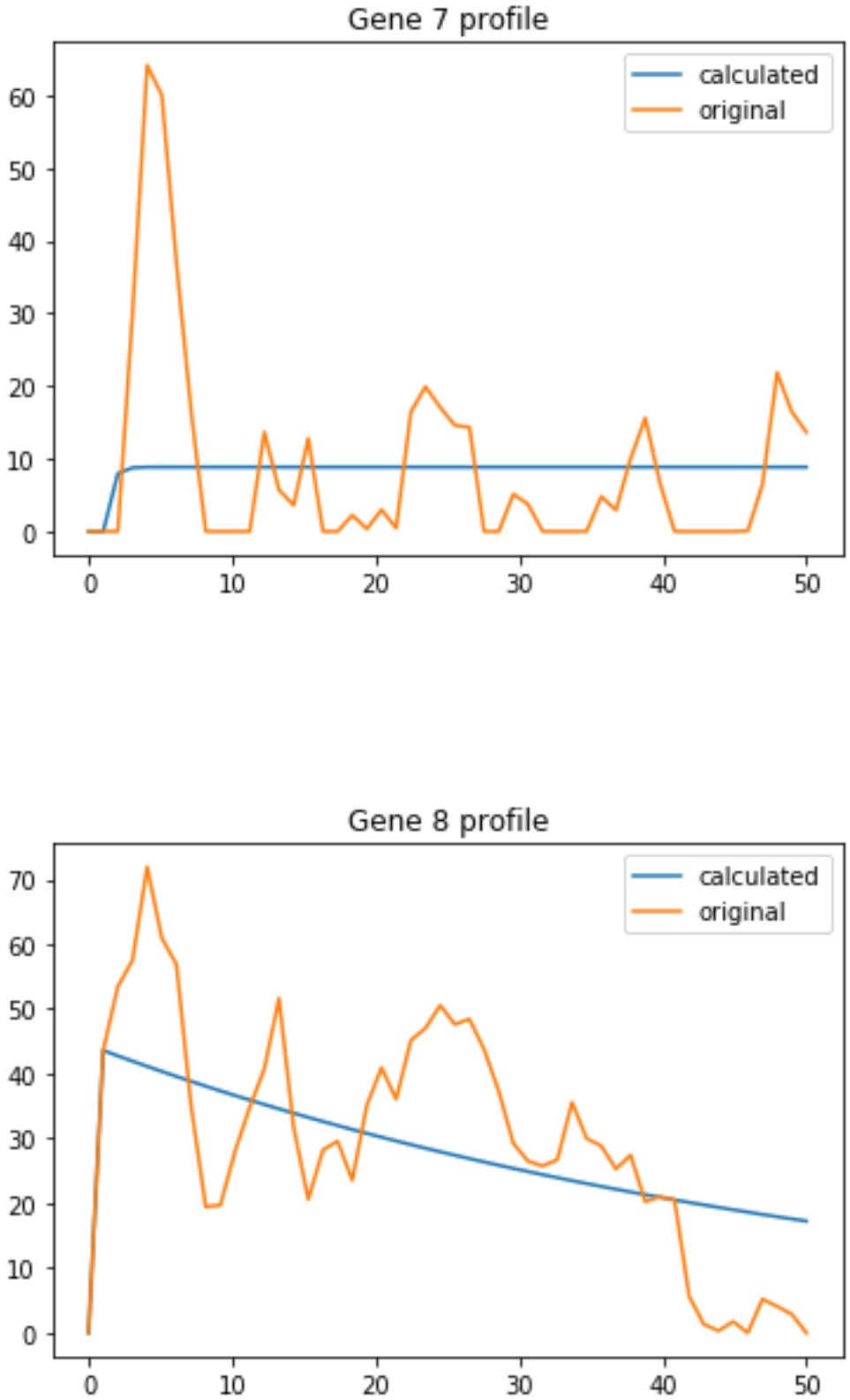


Fig. 3.4.1. Expression profile (obtained and original) of gene 1, gene 2, gene 3, gene 4, gene 5, gene 6, gene 7, gene 8 obtained using DE and tanh as an activation function

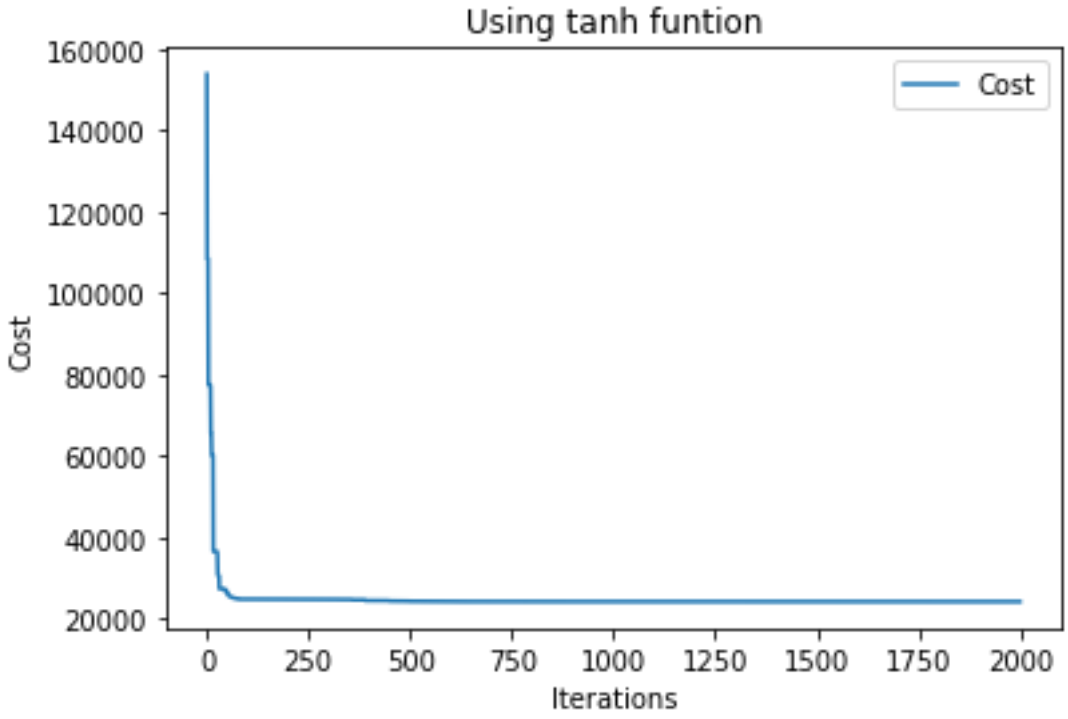


Fig. 3.4.2. Minimum cost function value in each step in DE using tanh activation function after 2000 iterations

III. Cumulative error of each gene after the complete run of DE using tanh activation function.

Gene 1 = 1439.4910149880902
Gene 2 = 8131.501435219638
Gene 3 = 844.267210270187
Gene 4 = 6207.163716012898
Gene 5 = 4690.7166396743705
Gene 6 = 617.3660965917309
Gene 7 = 472.0692689359811
Gene 8 = 537.707899707553

Rectified Linear Unit (ReLU) activation function:

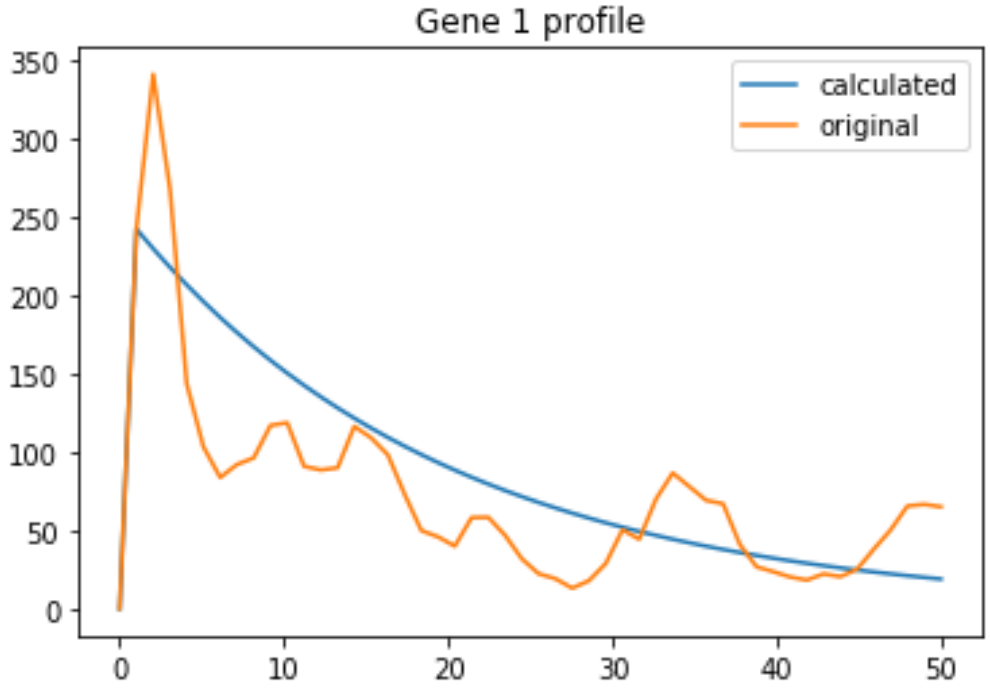
Third, we applied differential evolution (DE) code with rectified linear unit (ReLU) activation function $f(x) = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$ $or\ max(0,\ x)$ on 8 gene 50 time point dataset and calculated the result.

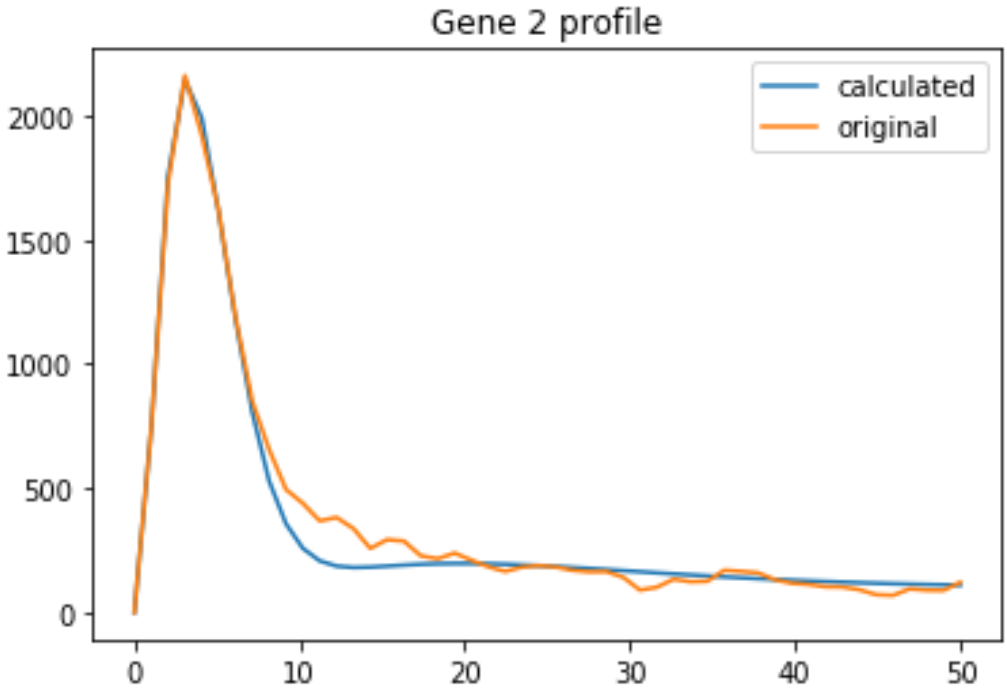
The results are as follows:

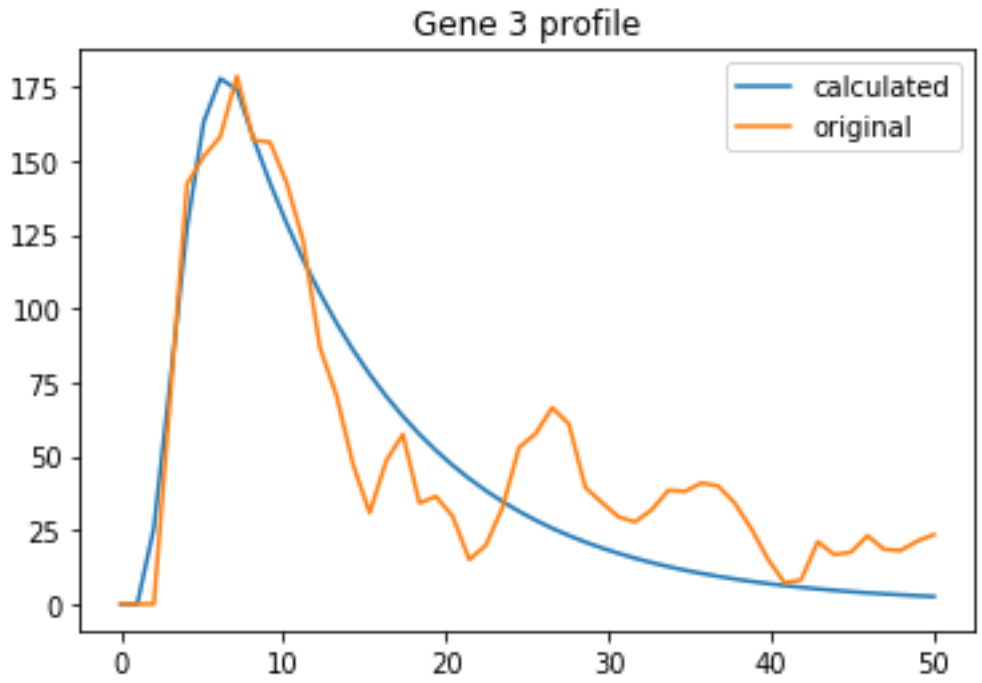
I. Weight values of the network:

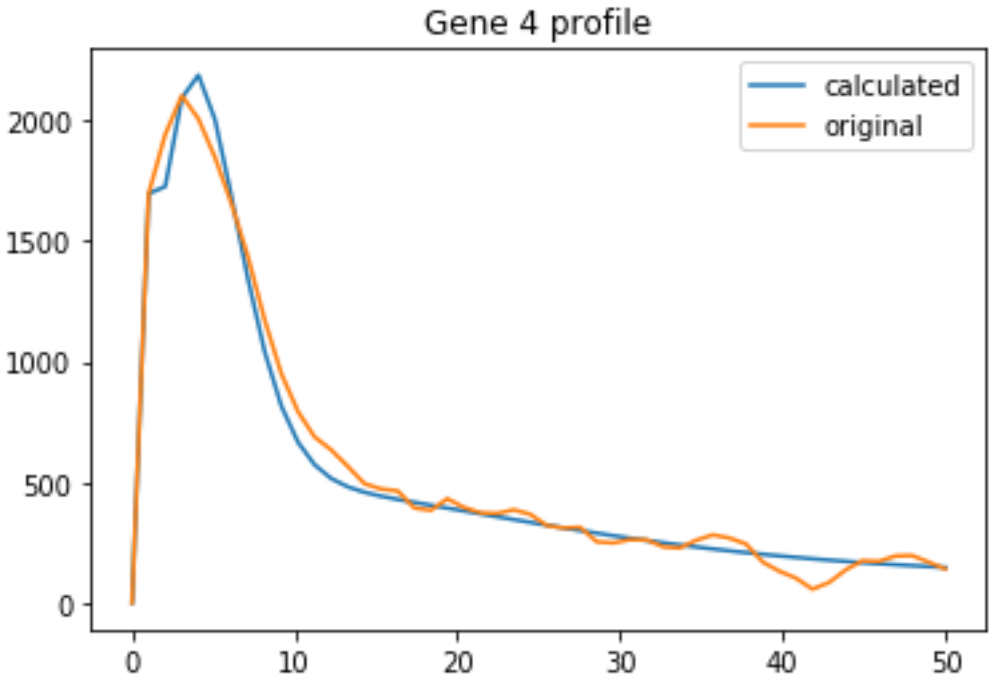
	Gene 1	Gene 2	Gene 3	Gene 4	Gene 5	Gene 6	Gene 7	Gene 8	b_i	Γ_i
Gene 1	29.93	-25.4	23.22	-29.0	2.21	-29.0	23.49	-25.1	-29.8	5.35
Gene 2	18.52	24.25	15.10	7.39	-30.0	29.95	-0.94	-29.9	-8.78	-0.10
Gene 3	-29.4	5.47	-27.3	-3.13	14.26	-1.99	-14.8	-26.8	-5.44	2.408
Gene 4	2.79	-0.67	3.30	29.10	-17.0	16.71	-27.1	-30.0	-23.1	-0.09
Gene 5	-26.9	3.04	28.3	21.3	2.03	-30.0	29.9	-29.2	27.40	-0.66
Gene 6	-9.09	-6.75	-23.1	-30.0	-6.68	20.76	16.15	15.03	-27.7	-0.92
Gene 7	-23.4	-23.3	8.75	30	-28.1	11.09	28.99	28.83	20.64	-22.8
Gene 8	3.01	-10.8	29.72	-6.21	-21.3	-21.4	28.63	-27.8	-28.9	-28.8

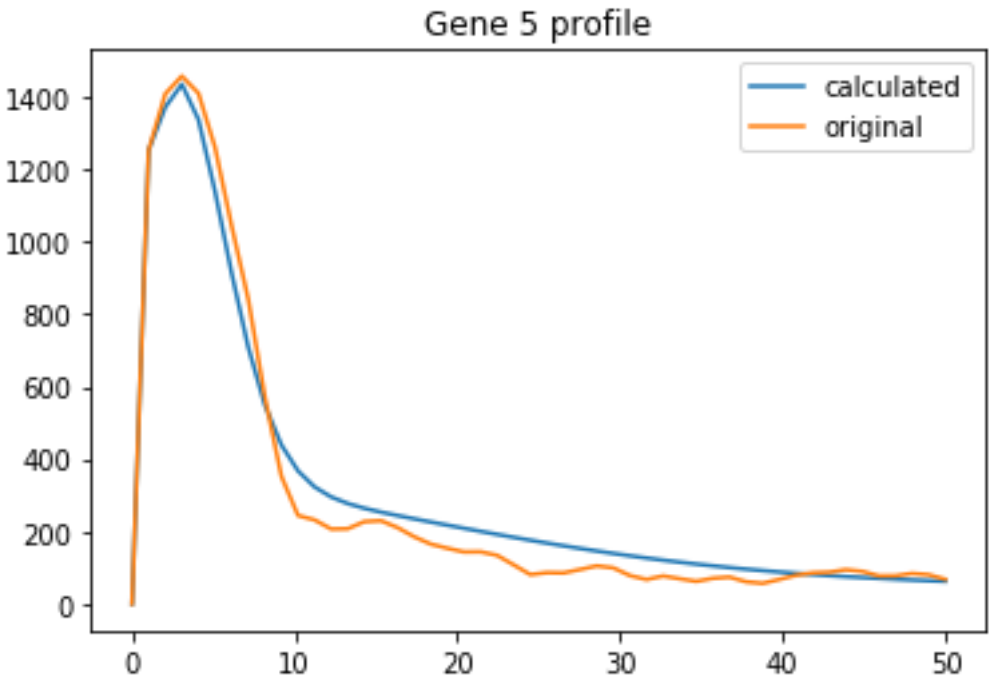
II. Gene profile graphs:

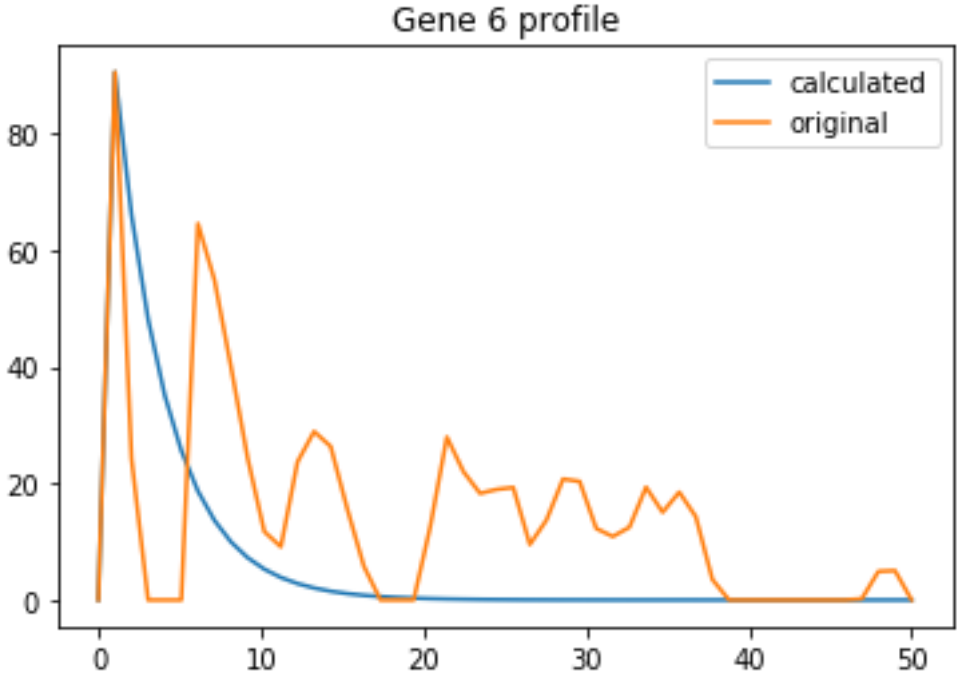





Gene 3 profile
calculated
original
175
150
125
100
75
50
25
0
0
10
20
30
40
50


Gene 4 profile
calculated
original
2000
1500
1000
500
0
0
10
20
30
40
50


Gene 5 profile
calculated
original
1400
1200
1000
800
600
400
200
0
0
10
20
30
40
50


Gene 6 profile
calculated
original
80
60
40
20
0
0
10
20
30
40
50

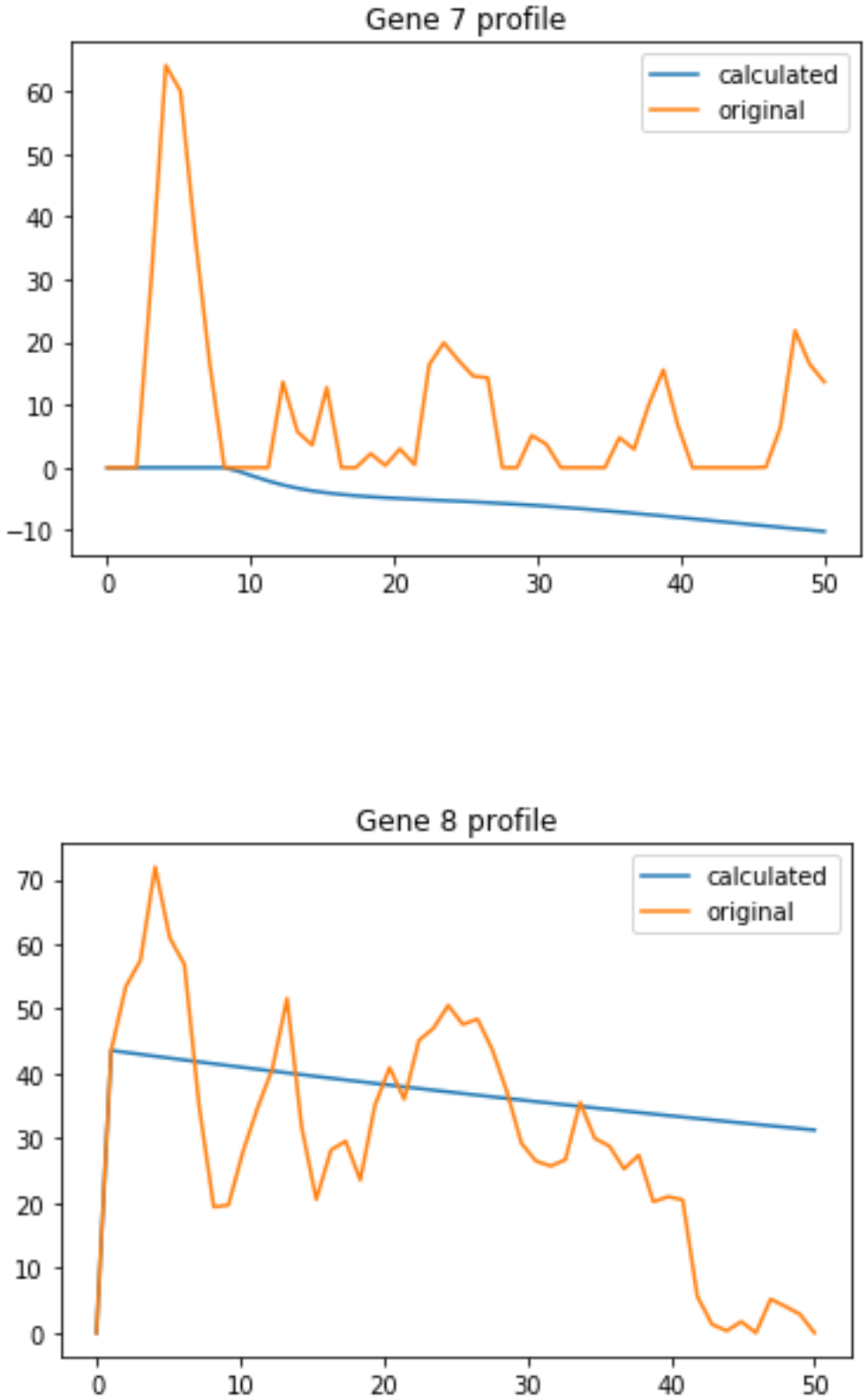


Fig. 3.4.1. Expression profile (obtained and original) of gene 1, gene 2, gene 3, gene 4, gene 5, gene 6, gene 7, gene 8 obtained using DE and ReLU as an activation function

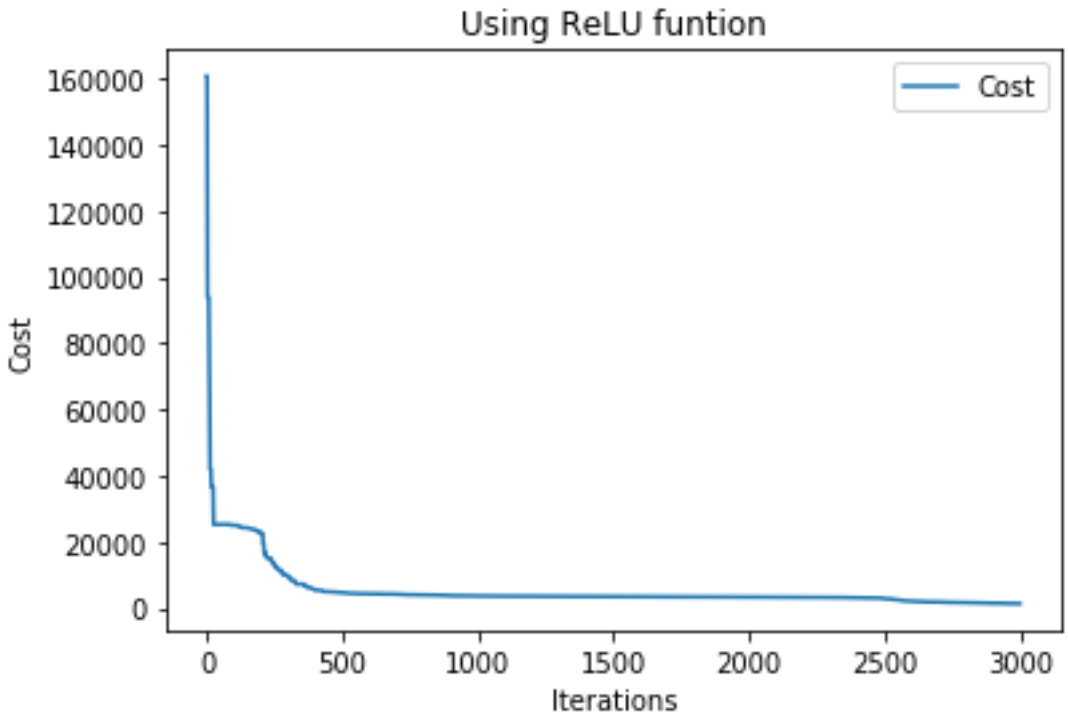


Fig. 3.4.2. Minimum cost function value in each step in DE using ReLU activation function after 3000 iterations

III. Cumulative error of each gene after the complete run of DE using ReLU activation function.

Gene 1 = 1704.2664858294645
Gene 2 = 2177.1760460114638
Gene 3 = 881.4329431142887
Gene 4 = 2537.2964561386193
Gene 5 = 2424.934737389529
Gene 6 = 689.3671569037579
Gene 7 = 691.4115267879777
Gene 8 = 668.2582379742862

3.6. Conclusion

Recurrent neural network-based models, using time series gene expression data from microarray experiments, provide a promising way to investigate gene regulatory mechanisms and understand gene interactions. The only information available for inference of gene regulatory network is the gene expression time series data, which are also erroneous, and there is no guideline regarding the structure of a particular gene regulatory network. Hence, the result obtained is not 100% accurate, but considering the challenges, and limited availability of information, our model provides a good result.

Due to the large number of model parameters and the small number of data sets available, the system of equations in GRN identification problem is highly underdetermined and ambiguous. Therefore, multiple solutions exist, which fit the given data, but show only little resemblance with the original target system.

References

[1] P. D'haeseleer, S. Liang, R. Somogyi, "Genetic network inference: from co-expression clustering to reverse engineering," *Bioinformatics* vol.16, no.8, pp.707- 726, 2000.

[2] Kozlov, K., & Samsonov, A. (2010). *DEEP—differential evolution entirely parallel method for gene regulatory networks. The Journal of Supercomputing, 57(2), 172–178.*doi:10.1007/s11227-010-0390-6

[3] Xu, R., Venayagamoorthy, G. K., & Wunsch, D. C. (2007). *Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization. Neural Networks, 20(8), 917–927.*doi:10.1016/j.neunet.2007.07.002

[4] Briti Sundar Mondal, Arup Kumar Sarkar, Mahmudul Hasan, M., & Noman, N. (2010). *Reconstruction of Gene Regulatory Networks using Differential Evolution. 2010 13th International Conference on Computer and Information Technology (ICCIT).*doi:10.1109/iccitechn.2010.5723898

[5] Akutsu T, Miyano S, Kuhara S, Identi¯cation of genetic networks from a small number of gene expression patterns under the Boolean network model, Pac Symp Biocomput 4:17–28, 1999.

[6] Perrin BE, Ralaivola L, Mazurie A, Bottani S, Mallet J, D'Alche-Buc F, Gene networks inference using dynamic Bayesian networks, Bioinformatics 19(2):II138–II148, 2003.

[7] Liu LZ, Wu FX, Zhang WJ, Inference of biological S-system using the separable estimation method and the genetic algorithm, IEEE/ACM Trans Comput Biol Bioinform 9(4):955–965, 2012.

[8] Mandal, S., Saha, G., & Pal, R. K. (2017). *Recurrent neural network-based modeling of gene regulatory network using elephant swarm water search algorithm. Journal of Bioinformatics and Computational Biology, 15(04), 1750016.*doi:10.1142/s0219720017500160.

[9] Chiang JH, Chao SY, Modeling human cancer-related regulatory modules by GA-RNN hybrid algorithms, BMC Bioinformatics 8(91):1–13 2007.

[10] Xu R, Wunsch IID, Frank R, Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization, IEEE/ACM Trans Comput Biol Bioinform 4(4):681–692, 2007.

[11] Palafox L, Noman N, Iba H, Study on the use of evolutionary technique for inference in gene regulatory networks, Natural Computing and Beyond, Springer, Tokyo, pp. 82–92, 2013.

[12] Khan A, Mandal S, Pal RK, Saha G, Construction of gene regulatory networks using recurrent neural networks and swarm intelligence, Scienti¯ca 2016(1060843):1–14, 2016, doi: 10.1155/2016/1060843.

[13] Mandal S, Khan A, Saha G, Pal RK, Large scale recurrent neural network based modeling of gene regulatory network using Cuckoo Search-Flower Pollination Algorithm, Adv Bioinformatics 2016(5283937):1–9, 2016, doi: 10.1155/2016/5283937.

[14] Datta, D., Sinha Choudhuri, S., Konar, A., Nagar, A., & Das, S. (2009). A recurrent fuzzy neural model of a gene regulatory network for knowledge extraction using differential evolution. 2009 IEEE Congress on Evolutionary Computation. doi:10.1109/cec.2009.4983307

[15] Datta, D., Konar, A., & Janarthanan, R. (2009). Extraction of interaction information among genes from gene expression time series data. 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). doi:10.1109/nabic.2009.5393607

[16] Das, P., Rakshit, P., Konar, A., & Janarthanan, R. (2011). *A recurrent fuzzy neural model of a gene regulatory network for knowledge extraction using Artificial Bee Colony optimization algorithm. 2011 International Conference on Recent Trends in Information Systems*.doi:10.1109/retis.2011.6146837

CHAPTER 4

CONCLUSION AND FUTURE WORKS

4.1. Conclusion

The study of gene regulatory networks (GRNs) structure is important in understanding cellular function. GRNs are typically represented by graphs in which the nodes represent the genes and the edges show the regulatory or interaction between genes. There are many methods for inference of GRNS. Here, in this thesis we used model based method. We have chosen RNN model to infer Gene Regulatory Network (GRN), and to find the optimal weights and bias of the network we have used Differential Evolution as an optimization algorithm with different activation functions, such as Sigmoid, tanh and ReLU. Given the similarity between RNNs and gene networks, we believe that RNNs will play an important role in exploring the mystery of gene regulation relationships. As to the RNNs model, it can be further extended to include factors like time delay, which is an important property of genetic regulatory networks, but unfortunately, not well addressed yet. Also, RNNs can also take into account more complex interactions, such as interactions between triplets of variables rather than pairs. The performance of DE with the three activation function is compared. The result from all three activation functions shows that sigmoid function converge faster than the rest two and takes less iteration to provide close to accurate result. This method infer GRNs with high accuracy and can identify direction of interaction. However, this is time consuming and require many parameters to be set up, and thus cannot be used for large-scale networks. This method works well for small networks, but large scale networks are still out of the scope of the method in the current form because of the high dimensionality of the model.

4.2. Future Works

This study has demonstrated that Differential Evolution is able to infer RNN model of a gene regulatory network with ease provided the dataset is small. For future study we could use better optimization algorithms such as Particle swarm optimization, Bat algorithm etc., to provide better insight into gene networks.

Currently, one of the major limiting factors for genetic regulatory network inference is the paucity of reliable gene expression time series data, which restricts the applications of current computational methods to only synthetic data, or small-scale real networks, with only several genes or gene clusters. Also, it is equally
important to combine prior information about the regulatory networks of study into the model so as to remove some biologically impossible connections.

These results are generally simplified versions of the network. If more knowledge can be incorporated in the cost function, then the model will be able to provide more accurate results. Our next goal is to incorporate a few more constraints in our cost function to make it more accurate. Being able to predict gene expressions more accurately provides a way to explore how drugs affect a system of genes as well as for finding which genes are interrelated in a process. To increase the robustness and redundancy of current models and further improve the search capability of the training algorithms are also important and interesting directions for further research.