DISCOVERING A REGULATORY NETWORK TOPOLOGY BY MARKOV CHAIN MONTE-CARLO ON GPGPUS WITH SPECIAL REFERENCE TO THE BIOLOGICAL CLOCK OF *NEUROSPORA CRASSA*

by

AHMAD MANSOUR AL-OMARI

(Under the Direction of Jonathan Arnold)

ABSTRACT

Bioinformatics in its interdisciplinary aspects comprises sciences of computers, medicine, biology, mathematics, and statistics. In essence, Bioinformatics uses computers to find causes of diseases and medical solutions. This dissertation addresses all of these sciences to solve one of the most important problems in system biology: solving large systems of ordinary differential equations (ODEs) describing how genetic networks behave using Markov Chain Monte-Carlo (MCMC) and parallel algorithms on General Purpose Graphical Processing Units (GPGPU). We used in this research *Neurospora crassa*, whish is a model organism that is widely explored and studied[1-5], due to its simplicity and its relatedness to the human beings. We predicted and understood the dynamics and the products of all of 2,418 genes that are believed to be under the control of the biological clock in *Neurospora crassa*. A genetic network that explains mechanistically how the biological clock functions in the filamentous fungus *Neurospora crassa* has been built and validated against over 31,000 data points from microarray experiments by harnessing the power of the GPGPU and exploiting the hierarchical

structure of that genetic network. Various mathematical models, statistical models, and numerical algorithms, such as Galerkin's method, in conjunction with Finite Element Method (FEM) piecewise hat functions, Adaptive Runge Kutta method (ARK), and Gauss-Legendre quadrature method are proposed and used on the GPU to accomplish the purpose of this thesis.

INDEX WORDS: Bioinformatics, Biological Clock, *Neurospora crassa*, Systems Biology, Genetic Network, Adaptive Runge Kutta, Ordinary differential equations Graphical Processing Unit, GPGPU, Galarkin, Finite Element Method, Piecewise Elements, Parallelization Strategies, supernet

DISCOVERING A REGULATORY NETWORK TOPOLOGY BY MARKOV CHAIN MONTE-CARLO ON GPGPUS WITH SPECIAL REFERENCE TO THE BIOLOGICAL CLOCK OF *NEUROSPORA CRASSA*

by

AHMAD MANSOUR AL-OMARI

B.Sc., Hijjawi for Engineering Tech. College, Yarmouk University, Jordan, 2004

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2015

DISCOVERING A REGULATORY NETWORK TOPOLOGY BY MARKOV CHAIN MONTE-CARLO ON GPGPUS WITH SPECIAL REFERENCE TO THE BIOLOGICAL CLOCK OF *NEUROSPORA CRASSA*

by

AHMAD MANSOUR AL-OMARI

Major Professor:	Jonathan Arnold
Committee:	Heinz-Bernd Schüttler
	Thiab Taha
	Suchendra Bhandarkar

Electronic Version Approved:

Julie Coffield
Interim Dean of the Graduate School
The University of Georgia
May 2015

DEDICATION

This dissertation is lovingly dedicated first and foremost to my beloved respective parents ***Mansour*** and ***Zakiah*** who have been my constant source of inspiration, love, and encouragement, second to my beloved wife ***Reem*** who has given me the drive and discipline to tackle any task with enthusiasm and determination, third to my beloved daughter ***Rand*** and son ***Hashim*** who are my future, fourth to all of my respective **sisters** and **brothers**, fifth to my lovely back-home country **Jordan**, and finally to my great universities **The University of Georgia** and **Yarmouk University**. Without every single one of them, this project would not have been made possible. May GOD bless amply all of them.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank ***THE ALMIGHTY GOD (ALLAH).*** Who gave me the gift of life, health, this opportunity to pursue my study of Ph.D. and gain much more of knowledge. Then my special gratitude and grateful goes to my advisor **Prof. Jonathan Arnold** who advises me to stay focused, structured and motivated. Without his patience, comments, and reviewing after me, I would not reach my goal. I found my inspiration in his words of encouragement. Thanks so much. I would like to thank greatly my committee members including; **Prof. Heinz-Bernd Schüttler** for not getting tired of my intensive and extensive questions and taking a long time to teach and correct me where and when I was short. Thanks so much. **Prof. Thiab Taha,** thank you for his valuable advices by showing me always the best ways that helped me to achieve my goals in addition to involving me in different workshops and lectures to strengthen my research knowledge. Thanks so much. **Prof. Suchendra Bhandarkar** for his valuable advices and comments during my study. Thanks so much. Finally, I would like to thank the CUDA Research and Teaching Centers in the Computer Science Department at UGA for allowing me to carryout my calculations using GPUs. UGA, through the efforts of Prof. Taha, was selected by Nvidia as a 2011 CUDA Teaching center and in 2014-2015 as a CUDA Research Center and many thanks to the people in the Georgia Advanced Computing Resource Center (GACRC) for providing the needed help and equipment (GPGPUs), especially **Prof. Shan-ho Tsai.** May GOD bless you all abundantly

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGUERS

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

Systems biology or systeomics concerns the biochemical reactions that happen in the biological systems components consisting of numerous and complex life processes such as protein-protein interactions, signal transduction, enzymes and substrates, and other cell reactions. Understanding and discovering these biochemical reactions underlying a complex process requires the use of computational and mathematical modeling of biological systems.

Genetic networks as a part of the systems biology deal with complex biochemical interactions and signaling and describe time-dependent concentrations of molecular species, such as genes, their RNAs, and their proteins as well as their substrates. These networks can be expressed as a system of coupled nonlinear ordinary differential equations (ODEs). Understanding such networks enables us to discover the biochemistry and genetic activity of a cell and how the cell evolves as a function of time (including its metabolic networks, signal transduction, and cell cycle models). Many problems could be solved and understood once these ODEs are solved. For example, human diseases like prostate cancer, and the phenotype of other complex traits, such as the development of an organ, and the biological clock of an organism could be so described. The problem facing biologists working to understand a genetic network is that the model parameters are mostly unknown, and the experimental data are noisy and limited for molecular

quantitative studies. Most genetic networks, such as that for the biological clock, are part of much larger modules controlling fundamental processes in the cell, such as metabolism, development, or response to environmental signals. As an example, the biological clock controls the circadian rhythms of about 2,418 distinct genes in the genome (with 11,000 genes) of a model system, the filamentous fungus, *Neurospora crassa[5]*. Predicting and understanding the dynamics of all of these genes and their products in a genetic network describing how the clock functions is a challenge and beyond the current capability of the fastest serial computers.

In the ensemble method, MCMC methods are used to generate random samples of models, including the initial conditions and parameters values, that actualize how the clock and 2,418 genes behave. For each model sampled, we need to solve a large system of ODEs describe the genetic network millions of times. Solving all of these models needs years for one simulation. We developed novel parallel algorithms on the General Purpose Graphical Processing Unit (or GPGPU) and ensemble methods, which use Markov Chain Monte Carlo, to fit and discover a regulatory network of unknown topology composed of 2,418 genes describing the entire clock circadian network, a network that is found from bacteria to humans, by harnessing the power of the GPGPU [6]and utilizing the hierarchical structure of that genetic network.

The organization for the rest of this dissertation as the following;

1) CHAPTER 2, we developed and propose a new numerical technique to solve nonlinear systems of initial value problems for nonlinear first-order differential equations that model genetic networks in systems biology using the Galerkin finite element method with

piecewise hat functions. The accuracy of this algorithm is high as adaptive Runge Kutta with a potential of parallelization in the ensemble method.

2) CHAPTER 3, we developed a parallelized version of the Adaptive Runge Kutta (ARK) method on the GPGPU that helps to solve very large systems of ODEs. For example, ODEs that belongs to a large genetic network describing clock can be used to determine genome-level dynamics.

3) CHAPTER 4, We implemented a novel "variable–topology supernet" ensemble method[4, 7-9] using MCMC simulations to fit and discover a regulatory network of unknown topology comprising 2,418 genes describing the entire clock circadian network, a network that is found from bacteria to humans, and we improved the parallelized ARK method on the GPU[6] and developed a faster algorithm on the GPU with an accuracy enough for the first-order linear ODEs of the biological problems, such as the genetic network that describing the clock of *Neurospora crassa,* using Gauss-Legendre quadrature method on the GPU[10]. For brevity, two novel approaches are used here to overcome the long time simulation problem. The first approach is to parallelize the classic ARK ODE solver on the GPGPU designated for solving general ODEs problems including linear and non-linear systems of ODEs where the achieved speed up over an extremely fast CPU and optimized C++ code is shown in Figure (4.5), which is about 237 fold for a single GPU, and the second approach is to parallelize the numerical exact integral (EI) solution formula on the GPGPU using Gauss Quadrature rule designated specifically for solving the first order linear ODEs that describes our network where the

achieved speed up ~142 fold over ARK method on the GPU described above. The goal in the MCMC methods is to select parameters that make the predicted solution fit to the experimental data as measured by some figure of merit, such as the chi-squared statistic with respect to the experimental data and the predicted solution. A fast ODE solver is critical to implementing MCMC methods on large networks.

This novel software[10] enabled us to discover a broad array of functions for *clock-controlled genes* as well as the gene regulators for those genes [10]as it is shown in Figure (4.12) within a few days instead of years using a GPGPU. We were able to infer the function of each regulator in Figure (4.12). We were able to discover how the *clock-controlled genes* were organized into a regulatory network. We found for the first time an explicit regulatory connection between the clock and ribosome biogenesis, which can now be tested. Each of these advancements were made possible by a new computational approach using GPUs.

4) CHAPTER 5, Conclusion summarizes our discovered findings and future work that can be done to improve our findings and algorithms.

CHAPTER 2

SOLVING NONLINEAR SYSTEMS OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS USING GALERKIN FINITE ELEMENT METHOD

Ahmad Al-Omari, Heinz-Bernd Schüttler, Jonathan Arnold, Thiab Taha

IEEE ACCESS Journal

Reprinted here with permission of publisher

Received April 5, 2013, accepted May 27, 2013, date of publication June 17, 2013, date of current version July 1, 2013.

Digital Object Identifier 10.1109/ACCESS.2013.2269192

ABSTRACT

Abstract- A new numerical technique to solve nonlinear systems of initial value problems for nonlinear first-order differential equations (ODEs) that model genetic networks in systems biology is developed. This technique is based on finding local Galerkin approximations on each sub-interval at a given time grid of points using piecewise hat functions. Comparing the numerical solution of the new method for a single nonlinear ODE with an exact solution shows that this method gives accurate solutions with relative error $1.88\text{x}10^{-11}$ for a time step $1\text{x}10^{-6}$. The new method is compared with the Adaptive Runge Kutta (ARK) method for solving systems of ODEs, and the results are comparable for a time step $2\text{x}10^{-4}$. It is shown that the relative error of the Galerkin method decreases approximately linearly with the log of the number of hat functions used. Unlike the ARK method, this new method has the potential to be parallelizable and to be useful for solving biological problems involving large genetic networks. A NSF commissioned video illustrating how systems biology helps us understand a fundamental process in cells is included.

INDEX WORDS: biological clock, Galerkin Method, Finite Element Method, Hat Function, Newton-Raphson Method, ordinary differential equation, toggle switch, systems biology.

2.1 INTRODUCTION

In the new cross-disciplinary field of systems biology merging genomics, bioinformatics and engineering the focus is on using networks of genes and their products to predict fundamental processes in the cell[11]. The field began in the 1990s with the assembly of biochemical pathways to describe the functioning of entire cells[12-14]. The field was transformed with the development of new genomics technologies[11, 15, 16]to measure how many genes and proteins behave simultaneously in cells. We are now poised to describe the cellular dynamics of an entire cellular network[17, 18]. The challenge is to be able to simulate such large networks. The dynamics of these cellular networks are often described by very large systems of ordinary differential equations[9]. One of the major problems in systems biology is solving large systems of ordinary differential equations describing how genetic networks behave[19], a challenge arising in other areas of science and engineering as well[20]. The Galerkin method has been employed for solving different kinds of ordinary differential equations[21-28]. Here we show how Galerkin's method can be used in conjunction with Finite Element Method (FEM) piecewise hat functions to solve systems of nonlinear first-order ordinary differential equations (ODEs). Here our method is applied to systems of ODEs describing several genetic networks[29, 30]. The importance of these networks to our daily lives is summarized in an NSF commissioned video attached[31]. The idea behind the method is to find local Galerkin approximations to the solutions of the ODEs on each sub-interval of a given mesh using a collection of hat functions. In addition to the fact that this method is a new method for solving any nonlinear system of ODEs with high accuracy and stability that is comparable with the ARK method, it has the potential to be parallelizable

and to be useful for solving biological problems that depend on solving systems of nonlinear ODEs modeling genetic networks [19]**.** Since the data to identify such networks are sparse and noisy (error being 10% of values measured or larger), such biological problems can be solved quickly and with acceptable accuracy and high stability when a small number of hat functions is used as shown in Figure (2.5). The high accuracy of the ARK method, as an example, is not needed for these biological problems[8]. Our new approach achieves the required biological accuracy and if so desired, gives results as accurate as the ARK method. The basic idea of the new approach is to approximate each element in the solution of a system of nonlinear first-order ODEs by a piecewise hat function on one subinterval at a time. In this paper, this method is illustrated by solving an initial value problem of: 1) a single nonlinear first-order differential equation; 2) a system of nonlinear first-order differential equations for a genetic network describing the toggle switch[32]; and 3) a system of nonlinear first-order differential equations for a genetic network for the biological clock of the model fungal system, *Neurospora crassa*[4] described in the video. The latter two initial value problems are central to systems biology. It is to be noted that parallelizing the Jacobian matrix and the integration functions in the Galerkin method can speed up the numerical computations of a system of nonlinear first-order differential equations.

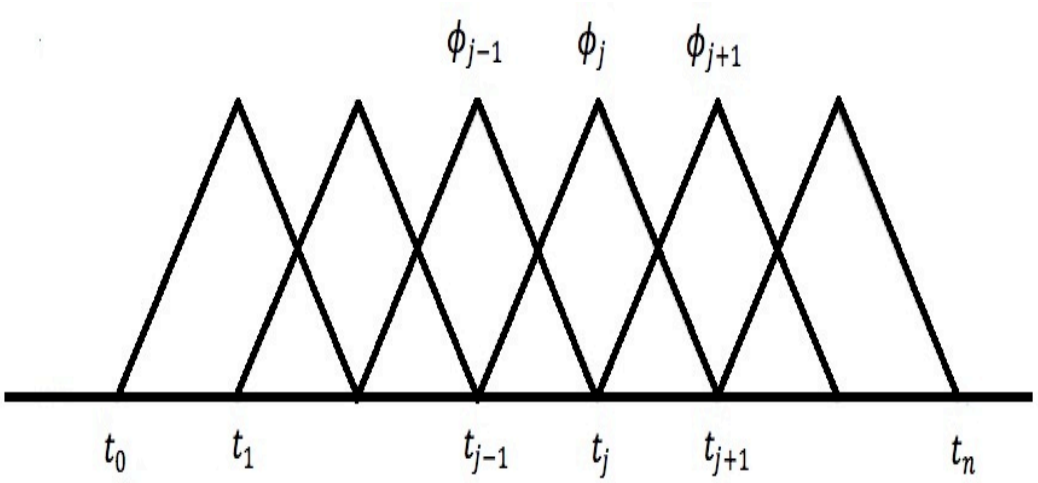

Figure 2.1:Display of Several hat functions

2.2 NUMERICAL METHODS

The Galerkin method is a very popular method for finding numerical solutions to partial differential equations. As an example for the Finite Element Method (FEM), we use the Galerkin method to approximate the solution of ordinary differential equations with a piecewise linear function as a sum of basis functions (Hat Functions). By using FEM and a weak formulation of the approximation method, which transfers the problem from a system of ODEs to a system of algebraic equations, we find the solution for the ODEs by solving these algebraic equations using the Newton-Raphson method. Three initial value problems are considered to show the accuracy of our method. The first problem involves solving only a single nonlinear first-order differential equation, and the other two cases involve solving two systems of nonlinear first-order differential equations. In the first example the new method is as accurate as the ARK method, and the other two examples solutions by the new method are comparable with the ARK method.

2.2.1 GALERKIN ALGORITHM FOR SOLVING SYSTEMS OF ODES

A system of initial-value problems for nonlinear first order ODEs over a solution's interval [0 L] can be defined as

y'=V(y,t);

where $y = \begin{bmatrix} V_1 \\ V_2 \\ . \\ . \\ . \\ V_{S-1} \\ V_S \end{bmatrix}$ $V = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_{S-1} \\ y_S \end{bmatrix}$

y(a)=b;

where S is the number of variables in a system of ODEs and in particular, the number of molecular species in a genetic network.

Note that a single nonlinear first order ODE problem considered above can be solved as a special case of the above system.

An approximate solution is expanded in terms of basis functions $\phi_j(t)$ as

$$y_n(t) = \sum_{j=0}^{N} p_{n,j}\, \phi_j(t) \qquad (1)$$

N is the number of hat functions; p_{nj} is a vector of unknowns expansion amplitudes that we are solving for; and n labels the different molecular species; and the $\phi_j(t)$ is a finite-element basis function (hat function) defined on a grid of time points tj by

$$\phi_j(t) = \begin{cases} \frac{t-t_{j-1}}{t_i-t_{j-1}}, & t_{j-1} \le t \le t_j \\ \frac{t_{j+1}-t}{t_{j+1}-t_j}, & t_j \le t \le t_{j+1} \\ 0\,, \text{ otherwise} \end{cases}$$

For example, the initial condition of the first species is given by $y_1(0)= p_{1,0}$, and the solution for a specific species (n) at a specific time point (j) is given by $y_n(t_j)= p_{n,j}$. Since $\phi_j(t_j) = 1$ and $\phi_k(t_j) = 0$ for k=j.

An alternative to hat functions is using compactly supported wavelets[33],or other types of finite-element basis functions, such as Hermite finite elements[8].

Using the residual form

$$R_n(t)=\sum_j^N p_{n,j}\phi_j'(t) - V_n(y_n(t),t) \tag{2}$$

we impose a weak Galerkin formulation of an approximate solution to solve for p_{nj} as weight variables

$$0=\int_{t_{k-1}}^{t_{k+1}} \phi_k(t)R_n dt \tag{3}$$

2.2.2 THE ALGORITHM

$$0=\int_{t_{k-1}}^{t_{k+1}} \phi_k(t)R_n dt \tag{4}$$

For arbitrary p_{nj}

$$F_{k,n}(p_{n,j}) = \int_{t_{k-1}}^{t_{k+1}} \phi_k(t)R_n(t)dt \tag{5}$$

$j = k-1,k,k+1; k = 1,...,L-1; n = 1,...,S$ (S is the number of species).

Define for a given fixed k; p^k ,$p^{k-1}$and p^{k+1}; where $p=p^{k+1}$

$$f(p)=\begin{bmatrix} F_{k,1}(p^{k-1},p^{k},p) \\ F_{k,2}(p^{k-1},p^{k},p) \\ . \\ . \\ . \\ F_{k,S}(p^{k-1},p^{k},p) \end{bmatrix} \equiv \begin{bmatrix} f_1(p) \\ f_2(p) \\ . \\ . \\ . \\ f_S(p) \end{bmatrix};$$

where $p_{1,k}$ are the solution points for the first ODE (species f_1), $p_{S,k}$ is the solution points for the last ODE (species f_S); k= 1, …, L-1.

Solving for f(p)=0 is done by using the Newton-Raphson method[34]. The procedure for a Newton-Raphson scheme for solving this system of nonlinear algebraic equations can be described by

a) Setting the initial iteration value to zero and assigning initial values for each variable,

b) Calculating the Jacobian matrix J,

$$J(p)=\begin{bmatrix} \frac{\partial f_1}{\partial p_1} & \frac{\partial f_1}{\partial p_2} & \cdots & \frac{\partial f_1}{\partial p_N} \\ \frac{\partial f_2}{\partial p_1} & \frac{\partial f_2}{\partial p_2} & \cdots & \frac{\partial f_2}{\partial p_N} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial f_S}{\partial p_1} & \frac{\partial f_S}{\partial p_2} & \cdots & \frac{\partial f_S}{\partial p_N} \end{bmatrix}$$

c) Using the Newton-Raphson scheme to solve the system of algebraic equations, which is obtained from f(p)=0, the solution is defined by

$$p_{i+1} = p_i - G_i \tag{6}$$

$$G_i = J^{-1}(p_i) f(p_i) \tag{7}$$

$$J(p_i) G_i = f(p_i) \tag{8}$$

The matrix G_i can be obtained by using the Gaussian elimination method with scale partial pivoting [34].

A central finite-difference formula has been used to find an approximation to the partial derivatives of the Jacobian matrix.

For example, calculating a given value in the Jacobian is done by

$$\frac{\partial f_r}{\partial p_u} \cong \frac{\Delta_p f_r}{\Delta p_u} = \frac{f_T(p_u + \delta) - f_T(p_u - \delta)}{2\delta} \tag{9}$$

$$;T = 0...S, u = 0...N$$

The δ has been assigned a small value such as $1x10^{-4}$, so that the final ODEs solutions for the above system do not change much by further reducing the δ value.

d) Calculating $f_T(p_u + \delta), f_T(p_u - \delta)$ by using the Gauss quadrature rule[34] since f_T is an integration function in this approach.

e) Iterating until the convergence of all variables is achieved. Tolerance of $1x10^{-6}$ is sufficient for solving the above system by using the Newton-Raphson method.

f) Note that solving such a system of algebraic equations is obtained sequentially. For example, the solution of the vector $f(p^{k+1})$ at fixed a time is obtained from the solution of the vectors $f(p^{k-1})$ and $f(p^{k})$.

What makes this Galerkin approach so attractive is the stability properties of the algorithm and the ability to control rigorously the error[23, 24, 27, 28].

In Figure (2.2) the solution of such systems is shown for the first initial value problem described below using our proposed method, and the solution oscillates around the exact solution. Therefore, to achieve a reasonably accurate solution with the lowest possible number of hat functions, we propose that the initial guess for the Newton-Raphson method on the oscillation time to be the average of p^{k-1} and p^{k} instead of just p^{k}. We check for an oscillation on p^{k-2}, p^{k-1} and p^{k} and we assign the average of p^{k-1} and p^{k} to p^{k} if the oscillation happens on these points. The accuracy of the final solution is based on the number of hat functions used for the solution (as the number of hat functions increases, the accuracy increases).

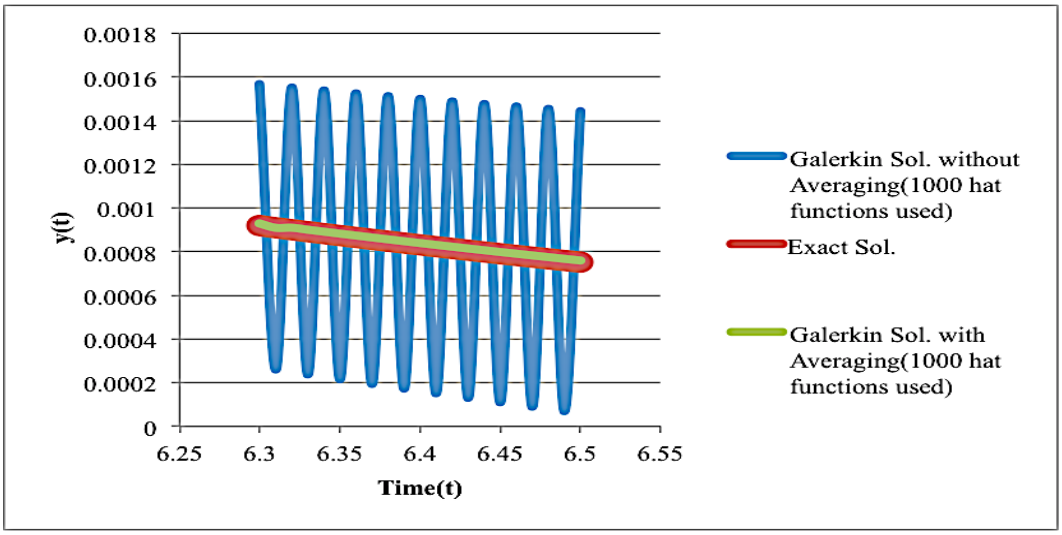

Figure 2.2: The solutions for the single nonlinear ODE and the effect of averaging on the numerical solution.

Figure (2.2) shows part of the solution for the nonlinear single ODE during the interval [0 10] using 1000 hat functions. The numerical solution without averaging is shown in blue, the exact solution in red, and the solution with averaging in green.

2.2.3 THE THREE INITIAL VALUE PROBLEMS CONSIDERED ARE

a) An initial value problem of a single nonlinear first ODE

$$y'=-y-y^2; \tag{10}$$

with intial condition y(0)=1

that has the exact solution:

$$y(t)=1/(-1+2e^{t}) \tag{11}$$

b) An initial value problem of a system of nonlinear first order ODEs for a genetic network of the toggle switch[32] as specified by

$$du/dt=\alpha_1/(1+v^{\beta})-u \tag{12}$$

$$dv/dt=\alpha_2/(1+u^{\gamma})-v \quad (13)$$

where; u(0)=0; v(0)=0 ; $\alpha_1=2,\alpha_2=4,\beta=2,\gamma=2$

$\propto_1 = 2, \propto_2 = 4, \beta = 2, \gamma = 2$

c) An initial value problem of a system of nonlinear first order ODEs for a genetic network of the biological clock of *Neurospora crassa*[4] as specified by

$$f_l'=A(f_G-f_l)w^n-A' f_l$$

$$f_r'=S_3(f_G-f_l)+S_4 f_l-D_3 f_r$$

$$f_p'=L_3 f_r-D_6 f_p$$

$$w'=E_2 u_p-D_8 w-nA(f_G-f_l)\, w^n+nA' f_l-Pwf_p^m$$

$$u_p'=L_1 u_{r1}-D_4 u_p-E_2 u_p$$

$$u_{r1}'=C_1\, u_{r0}\, f_p-D_7\, u_{r1}$$

$$u_{r0}'=V_1-D_1 u_{r0}-C_1 u_{r0} f_p$$

All of the parameter values of this clock network problem are given in Table (2.1) and Table (2.2) [4].

2.3 RESULTS AND DISCUSSIONS

The new method yields solutions for fixed and specific time steps, and the accuracy is as high as the ARK method if a large number of hat functions are considered as shown in Table (2.3) and Figure (2.3). On the other hand, the accuracy of the solution is still acceptable for biological problems if a fewer number of hat functions is considered as shown in Figure (2.4) and Figure (2.5).

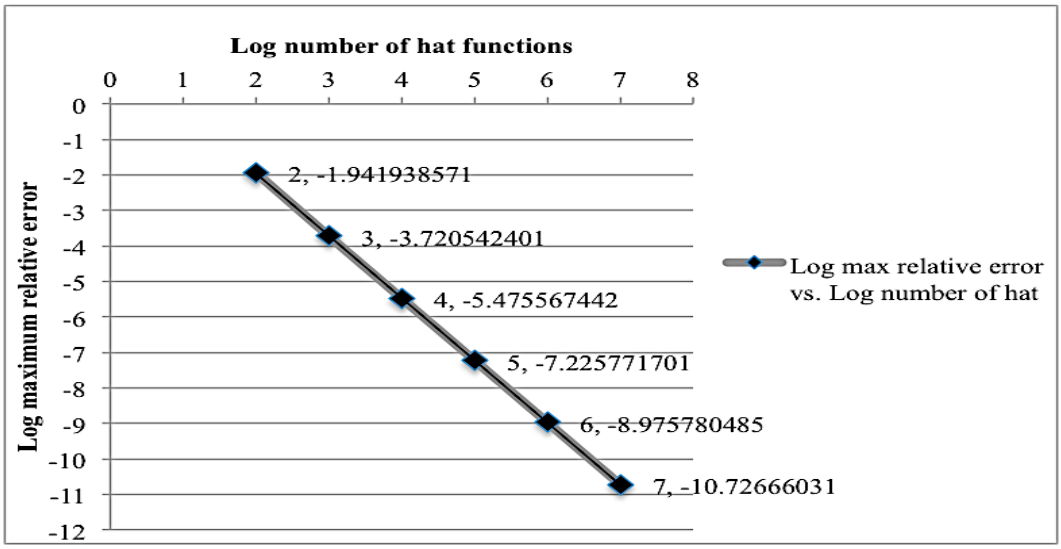

Figure 2.3: Maximum relative error (on a log scale) decreases linearly with the log of the number of hat functions for the single nonlinear ODE compared with the exact solution. Different numbers of hat functions ($1x10^2$, $1x10^3$, $1x10^4$, $1x10^5$, $1x10^6$, and $1x10^7$) over the solution interval [0 10] are used.

The maximum global relative error has been computed for the single nonlinear first ODE, for the genetic network of the toggle switch, and for the genetic network of the biological clock of *Neurospora crassa* using the following formula:

$$\text{Max global relative error} = Max_k \frac{2Max_n |Y_k^{Aprx.}(t_n) - Y_k^{Exact}(t_n)|}{Max_n |Y_k^{Aprx.}(t_n)| + Max_m |Y_k^{Exact}(t_n)|} \quad (14)$$

;where k=1,2,…,S = number of species.

The single nonlinear first ODE has an exact solution, and for the other two cases we have considered the approximate solutions of ARK with absolute and relative errors equal to $1x10^{-13}$ as exact solutions for them.

Table 2.1: (Initial Conditions at t=0)

Species	Initial Condition
f_l	0.00400782
f_r	0.181388
f_p	1.37307
w	0.0000663227
u_p	0.0000362815
u_{r1}	0.212505
u_{r0}	0.0000000252030

Table 2.2: (Parameters Values)

Parameters	Value	Parameters	Value
A	0.0000462010	D_8	0.00285475
A'	0.566108	C_2	1.66501
S_1	9.22739	P	3.55829
S_2	0.00353803	A_c	5.57336
S_3	0.000000136553	B_c	1.82043
S_4	9.07295	S_c	0.0149985
D_1	1.35911	L_c	0.0111332
D_2	2.77832	D_{cr}	0.268920
D_3	0.223231	D_{cp}	0.269409
C_1	0.0545178	vp	0.120699
L_1	59.7062	u_1	0.0124268
L_2	35.3755	f_0	0.692213
L_3	0.798222	n	4
D_4	0.00000947792	m	4
D_5	0.00000179706	E_2	v_p*C_2
D_6	0.159737	f_G	f_0+f_1
D_7	0.192918	V_1	S_1*u_1

Table 2.3: Errors in the solution over the time period [0 10] are shown as a function of the number of hat functions for the single ODE (example 1). The maximum relative errors have been calculated using different numbers of hat functions and the solutions compared with the ARK method (with absolute and relative errors for ARK equal to $1x10^{-13}$)

# of hat functions	Max relative error
1.0 E+02	0.0114304
1.0 E+03	0.000190308
1.0 E+04	3.34528064273916E-06
1.0 E+05	5.94604646865157E-08
1.0 E+06	1.05735181433986E-09
1.0 E+07	1.87646163866028E-11

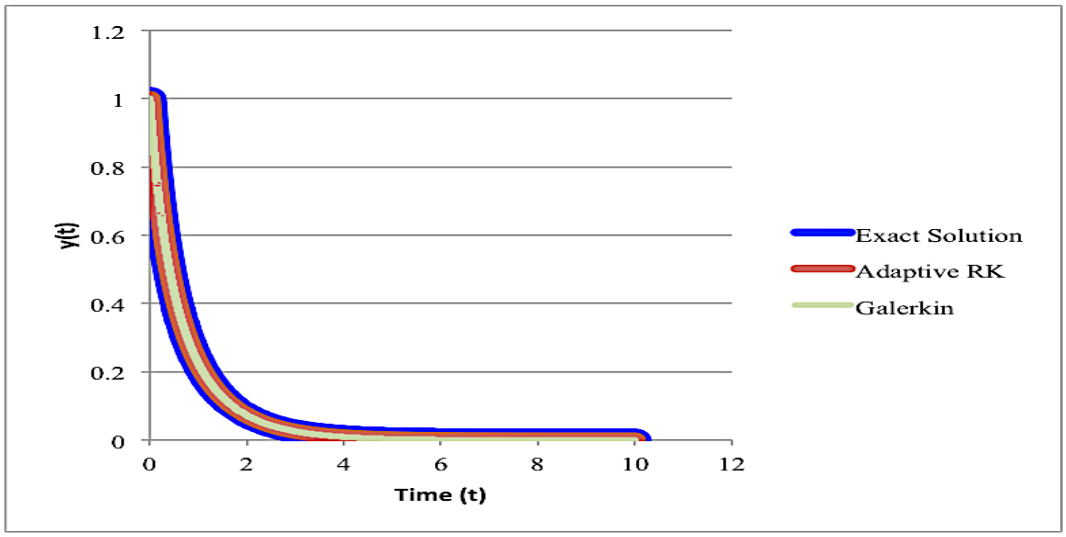

Figure 2.4: Solutions of the exact, Galerkin, and the ARK methods using $1x10^2$ hat functions over the time period [0 10] ODE

2.3.1 THE THREE CASES THAT HAVE BEEN CONSIDERED TO SHOW THE ACCURCY OF OUR METHOD

a) Solving an initial value problem of a single nonlinear first order ODE

We used the proposed algorithm to find solutions for the single nonlinear ODE with various numbers of functions in the fixed intervals [0 10]. It has been found that the accuracy increases approximately two orders of magnitude as the number of hat functions increases by one order of magnitude as is shown in Table (2.3), Figure (2.3), and Figure (2.4).

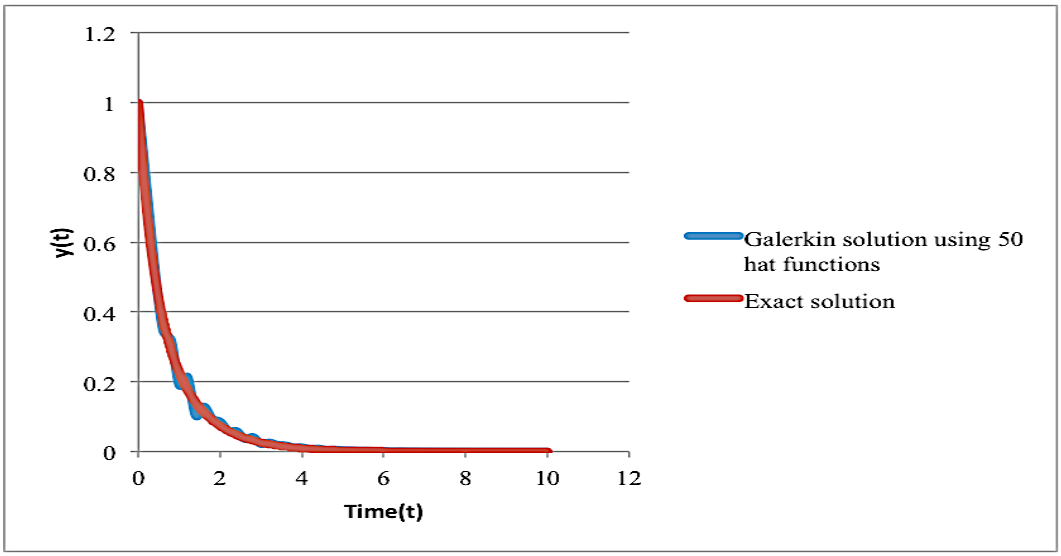

Figure 2.5: The solutions of the exact and Galerkin methods using relatively few number of hat functions (50 hat functions) over time period [0 10] for the single nonlinear ODE. Such a solution is sufficient for biological problems with acceptable accuracy and high stability using few hat functions.

b) Solving ODEs of the genetic network of the toggle switch

We used the proposed algorithm to find solutions for the toggle switch genetic network with various numbers of hat functions and time steps over the solution interval [0 10]. The solution is comparable with the ARK method (with absolute and relative errors for ARK being $1x10^{-13}$) as is shown in Table (2.4), Figure (2.6), Figure (2.7), and Figure (2.8).

From Table (2.4), we found that the errors coming from solving the v species in the toggle switch are larger than the ones coming from the u species with varying numbers of hat functions. Thus, the maximum global relative error equals to the maximum local relative error of the v species. The Galerkin and the ARK methods give comparable solutions for both variables u(t) and v(t) as shown in Figure (2.7) and Figure (2.8). The two solutions by different methods are virtually indistinguishable.

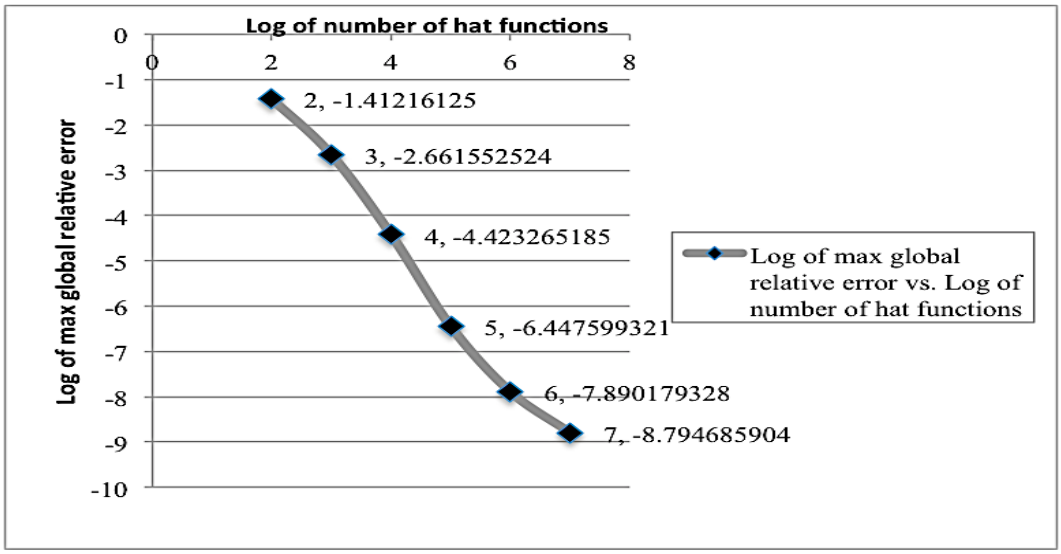

Figure 2.6: The maximum global relative error (on a log scale) among all of the species (global error of two species) for ODEs of the genetic network of the toggle switch decreases approximately linearly with the log of the number of hat functions. These results are based on the comparison with the ARK method (with absolute and relative errors for ARK are equal to $1x10^{-13}$). Different numbers of hat functions ($1x10^{2}$, $1x10^{3}$, $1x10^{4}$, $1x10^{5}$, $1x10^{6}$, and $1x10^{7}$) over the solution interval [0 10] are used.

Table 2.4: Errors in the solution over the time period [0 10] are shown as a function of number of hat functions for the toggle switch (example 2). The max local and global relative errors have been calculated using different number of hat functions and the solutions compared with the ARK method (with absolute and relative errors for ARK equal to $1x10^{-13}$)

Number of Hat Functions	Maximum Local Relative Error (u Species)	Maximum Local Relative Error (v Species)	Maximum Global Relative Error
1.0 E+02	2.82661552086085E-02	3.87113886066362E-02	3.87113886066362E-02
1.0 E+03	7.088438084833E-04	2.17995473439916E-03	2.17995473439916E-03
1.0 E+04	1.09812819646773E-05	3.77341709775271E-05	3.77341709775271E-05
1.0 E+05	1.12067920569916E-07	3.56779968791140E-07	3.56779968791140E-07
1.0 E+06	4.01110985582986E-09	1.28771772065253E-08	1.28771772065253E-08

c) Solving ODEs of the genetic network of the biological clock of *Neurospora crassa*.

The proposed Galerkin algorithm yields the solution for the biological clock of *Neurospora crassa* genetic network with various numbers of hat functions and time steps over a larger solution interval [0 200]. The solution is comparable with the ARK method (with absolute and relative errors for ARK equal to $1x10^{-13}$) as it is shown in Table (2.5) and Figure (2.9), Figure (2.10), Figure (2.11), Figure (2.12), Figure (2.13), Figure (2.14), Figure (2.15), and Figure (2.16).

Note that the accuracy is less than the second case 2) because we use the same number of hat functions over a larger solution interval [0 200] for the genetic network of the biological clock of *Neurospora crassa* instead of [0 10] for the former two cases. On the other hand, we still can see in Table (2.5) that a total of 10,000 hat functions is sufficient to obtain a relative error that is 0.07 or 7% or less. Again in (Fig. 9) there is a linear relation between the maximum global relative error and the number of hat functions on a log-log plot.

The solutions for this dynamical system in Figure (2.10), Figure (2.11), Figure (2.12), Figure (2.13), Figure (2.14), Figure (2.15), and Figure (2.16) using the Galerkin and ARK methods are indistinguishable using 1000 hat functions. Although the max global relative error using 1000 hat functions over the interval [0 200] is of order of 30% as it is shown in Table (2.5), in the figures we show that the solution is sufficiently good for biological problems.

Table 2.5: Maximum local relative errors of the Galerkin solution over the time period [0 200] for varied numbers of hat functions for the clock model (example 3). The maximum local and global relative errors have been calculated using different numbers of hat functions and the solutions, compared with the ARK method (with absolute and relative errors for ARK equal to $1x10^{-13}$)

# of Hat Functions	Max Local Rel. Error f1	Max Local Rel. Error fr	Max Local Rel. Error fp	Max Local Rel. Error w	Max Local Rel. Error up	Max Local Rel. Error ur1	Max Local Rel. Error ur0	Max Global Error
2.00E+2	1.12E+0	9.30E-1	7.28E-1	1.01E+0	1.25E-1	9.35E-2	4.53E-2	1.11551250069628E+0
1.00E+3	2.34E-1	1.91E-1	1.00E-1	3.08E-1	1.21E-2	7.47E-3	5.60E-3	3.07789282736857E-1
1.00E+4	5.14E-2	3.71E-2	1.86E-2	6.92E-2	2.31E-3	1.17E-3	9.91E-4	6.91539734511926E-2
1.00E+5	6.20E-3	4.55E-3	2.28E-3	8.49E-3	2.86E-4	1.44E-4	1.22E-4	8.48562558931092E-3
1.00E+6	6.77E-4	5.00E-4	2.52E-4	9.33E-4	2.76E-5	1.43E-5	1.35E-5	9.33489762577023E-4
1.00E+7	5.94E-5	4.38E-5	2.20E-5	8.18E-5	2.49E-6	1.28E-6	1.18E-6	8.17589819877030E-5

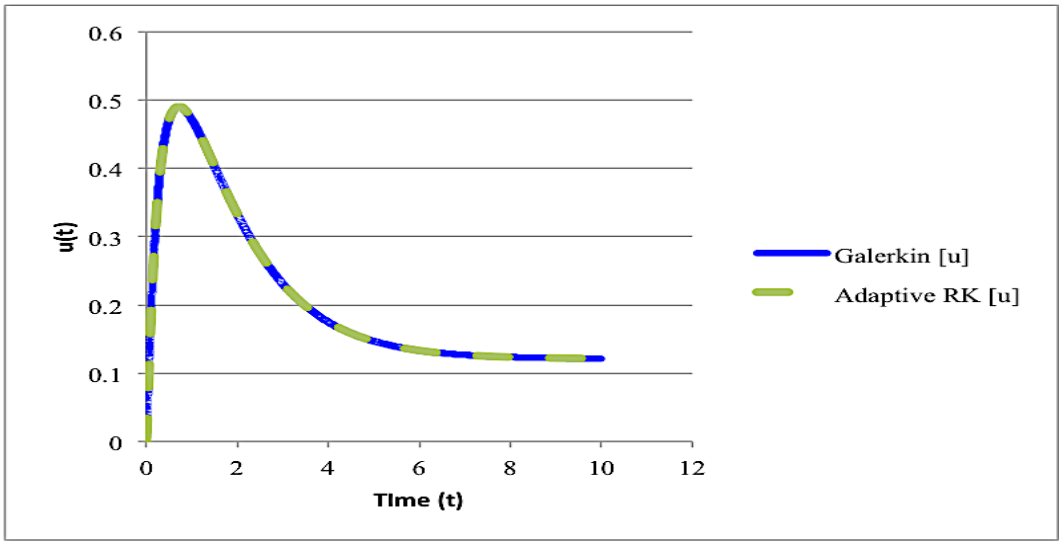

Figure 2.7: The solutions of u(t) using the Galerkin method with $1x10^2$ hat functions and the ARK method over the time period [0 10] for the genetic network of the toggle switch

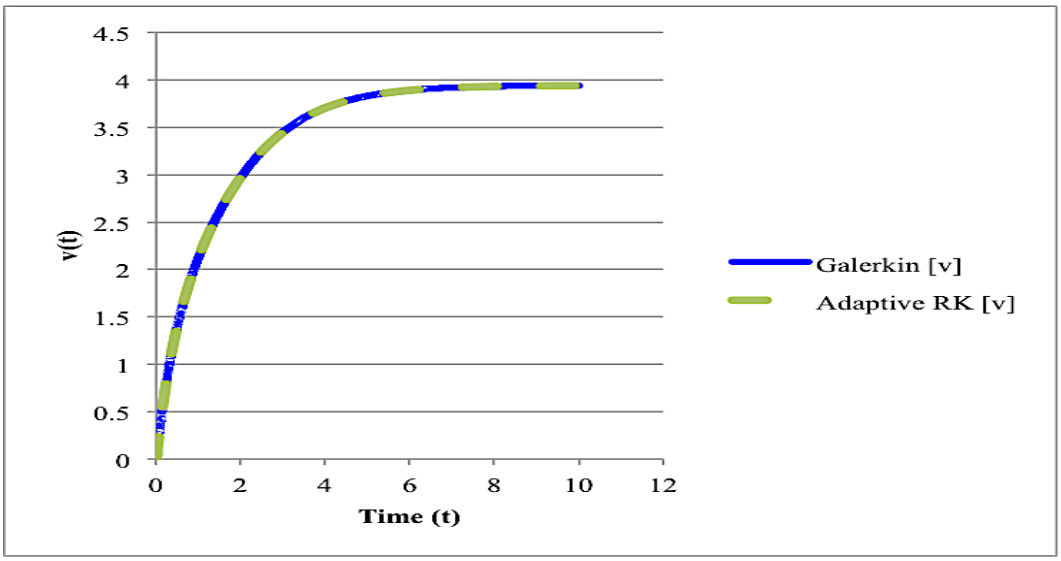

Figure 2.8: The solutions of v(t) using the Galerkin method with$1x10^2$ hat functions and the ARK method over the time period [0 10] for the genetic network of the toggle switch.

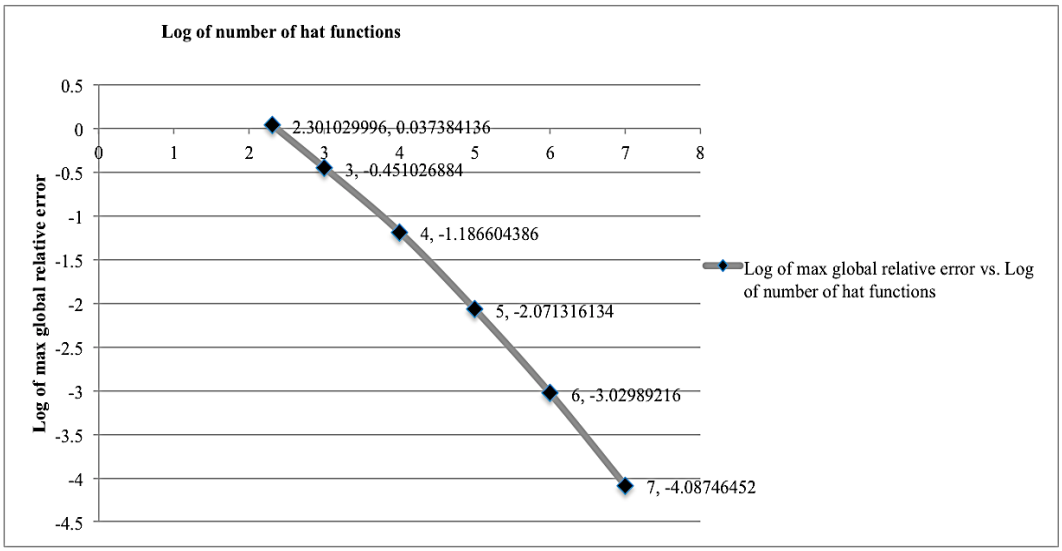

Figure 2.9: The maximum global relative error (on a log scale) among the whole species (seven species) for ODEs of the genetic network of the biological clock of *Neurospora crassa* decreases approximately linearly with the log of the number of hat functions compared with the ARK method (with absolute and relative errors for ARK equal to $1x10^{-13}$). Different numbers of hat functions ($2x10^2$, $1x10^3$, $1x10^4$, $1x10^5$, $1x10^6$, and $1x10^7$) over solution interval [0 200] are used.

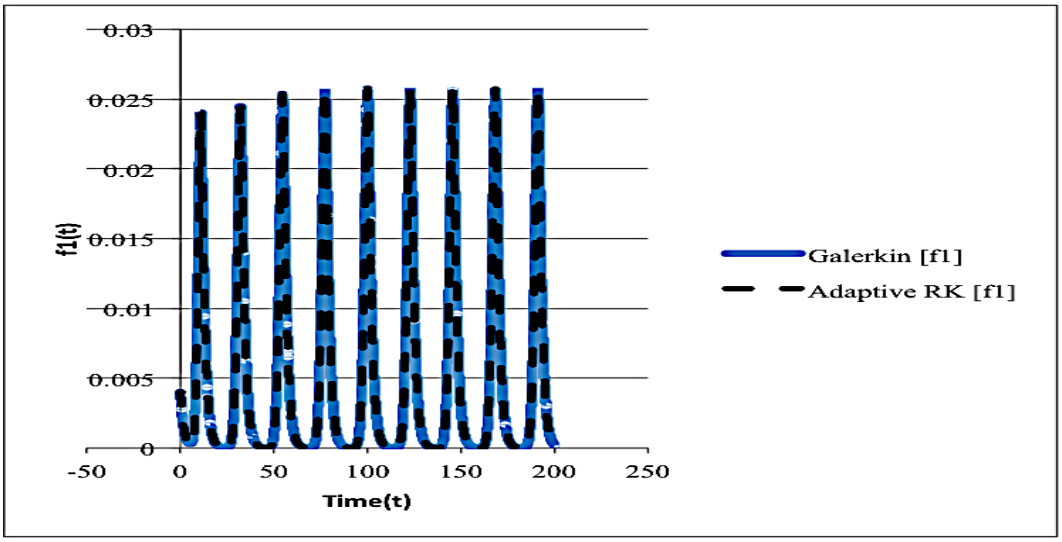

Figure 2.10: The solution of f1(t) using the Galerkin and the ARK method using $1x10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa.*

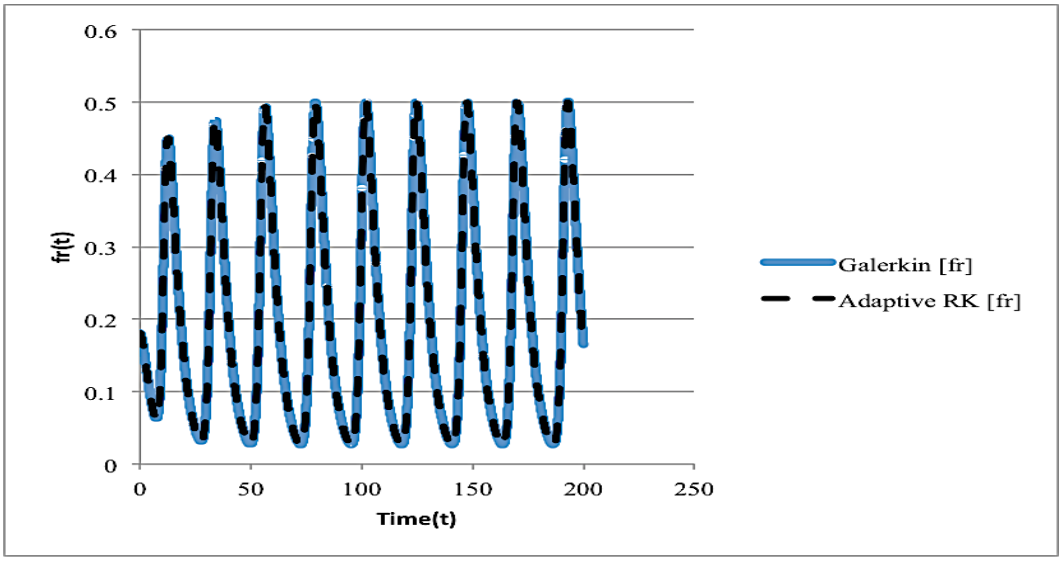

Figure 2.11: The solution of fr(t) using the Galerkin and the ARK methods using $1x10^3$ hat functions over the time period [0 200]for the genetic network of the biological clock of *Neurospora crassa*

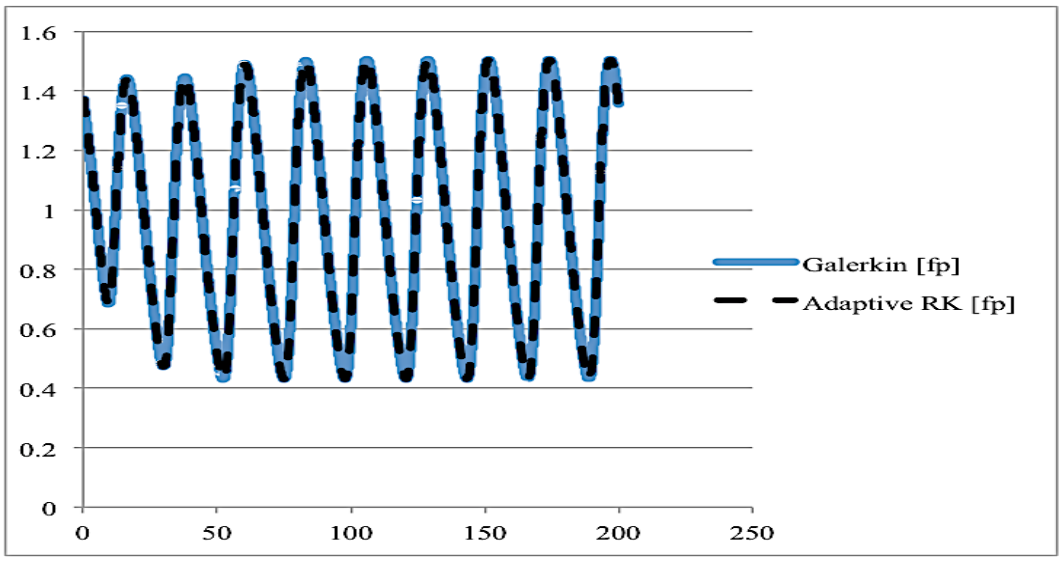

Figure 2.12: The solution of fp(t) using the Galerkin and the ARK methods using $1x10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*

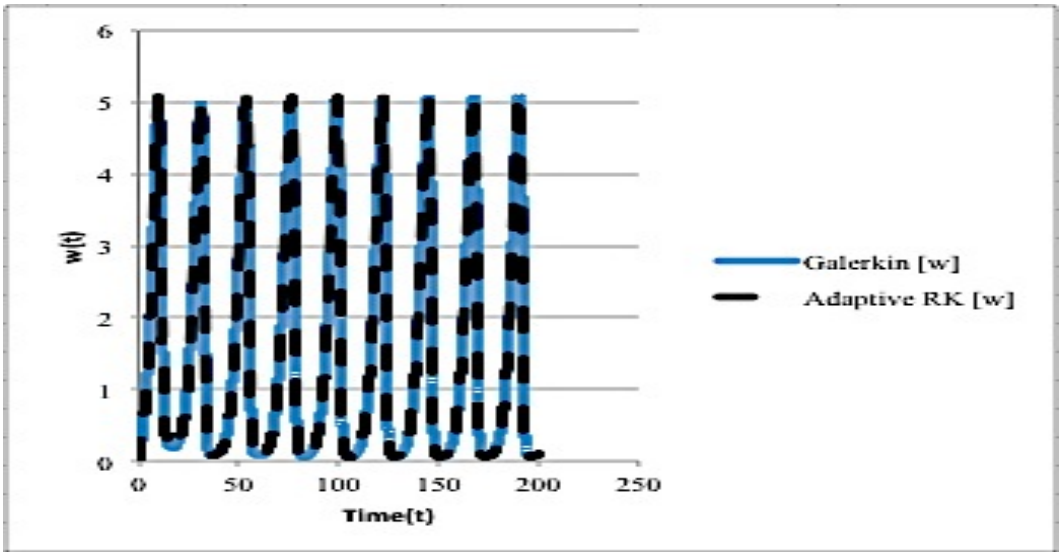

Figure 2.13: The solution of w(t) using the Galerkin and the ARK methods using $1x10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*.

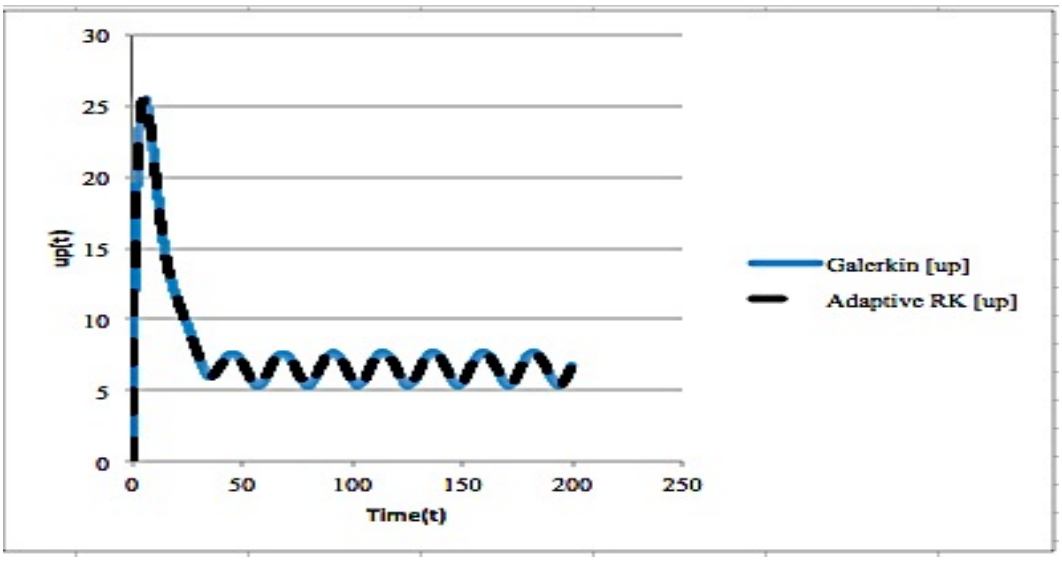

Figure 2.14: The solution of up(t) using the Galerkin and the ARK methods using $1x10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*

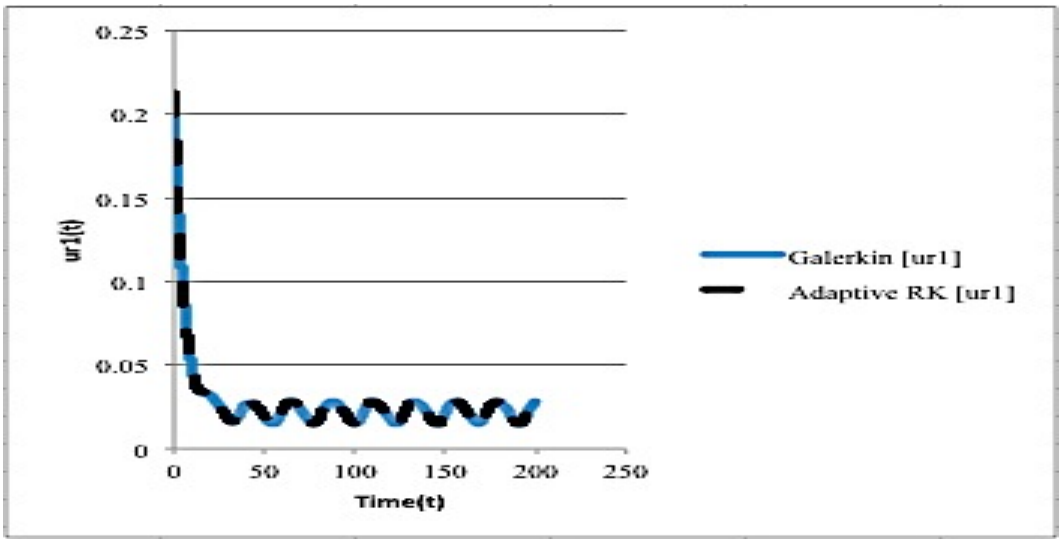

Figure 2.15: The solution of ur1(t) using the Galerkin and the ARK methods using $1x10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*.

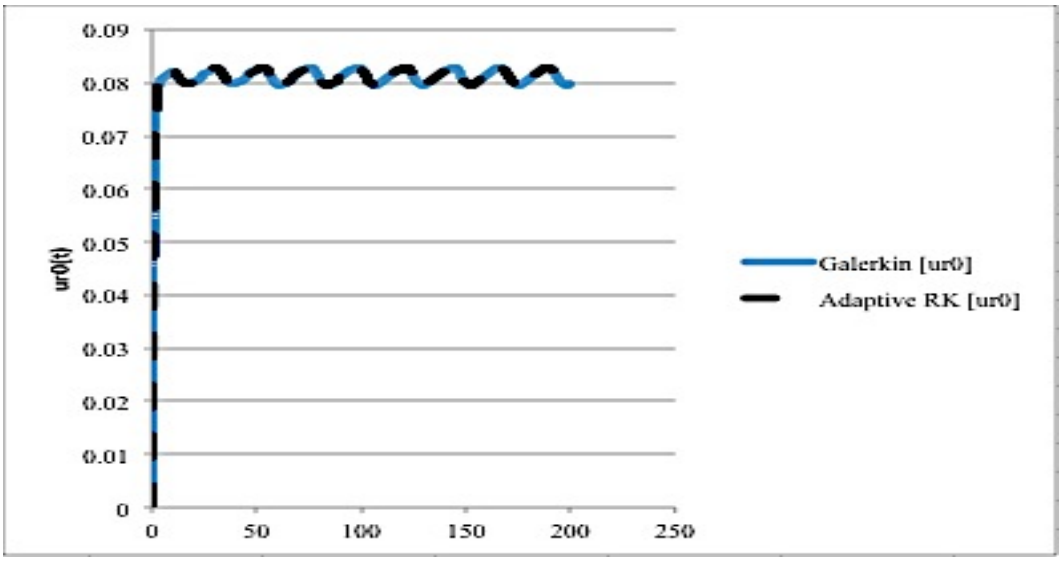

Figure 2.16: The solution of ur0(t) using the Galerkin and the ARK methods using $1x10^3$ hat functions over the time period [0 200] for the genetic network of the biological clock of *Neurospora crassa*.

2.3.2 A POTENTIAL PARALLELIZATION SCHEME FOR THE GALERKIN METHOD

Unlike the ARK method, which is inherently sequential for solving systems of ODEs, the Galerkin method as stated before can be parallelized by parallelizing the Jacobian matrix's calculation and the integration functions. This parallelization will speed up the numerical method of solving a system of nonlinear first-order differential equations. Moreover, this new method allows us to parallelize the ensemble method[4] for identifying genetic networks from real data on each variable (or species). Briefly, the ensemble method suggests that instead of identifying one unique parameterization of the model, we aim to identify an ensemble of models consistent with available experimental data and use Monte Carlo simulation techniques to generate random samples of model parameterizations (an ensemble) that represent the data well. This sampling process is captured in an animation within the associated NSF commissioned video[31]. In other

words, in the ODE solving scenario a unique solution will be found by specifying the initial conditions. In contrast, in the ensemble method since we don't know the initial conditions and other parameters values, which are required for solving systems of ODE, a Monte Carlo procedure is used to generate several initial conditions and parameters values, and while the Monte Carlo runs, it finds parameters that make the predicted solution closer to the experimental data. Finding the parameters could be done by using the Metropolis procedure[4], which minimizes the Chi-squared statistic comparing the experimental data and the predicted solution. Mainly, there are two stages in the ensemble method: the equilibration stage that is used to find parameters values that make the ODE solution converge to the experimental data and the accumulation stage, which is used to accumulate many sets of these parameters (i.e., the ensemble) that fit the experimental data well. Averaging over the ensemble allows an assessment of fit to the experimental data. Thus, averaging several solutions of the ODEs with different initial conditions that fit the experimental data will be found from a random sample of parameters that reproduce the experimental data.

Using ARK in the ensemble method implies that the system of ODEs should be re-solved for each proposed ensemble Monte Carlo updating step, and solving for the time step t+h requires the solution at the prior t. To apply a parallelized Galerkin method version instead of the ARK method version, suppose there are (n) hat functions subintervals with (k) test grid points. The purpose of these test grid points is to sample the quality of the solution, for each of the subintervals as shown in Figure (2.17). On the one hand, Monte Carlo simulation will propose a set of parameters values and hat

function amplitudes, which are required for solving the system of ODEs using FEM explained in this paper, for each subinterval. On the other hand, one or more subinterval(s) could be assigned to one slave processor that will solve for the system of ODEs given these parameters on its subinterval(s). Then for each of these test grid points within this subinterval(s), the method evaluates the left hand side of the differential equation and the right hand side of the differential equation and from the difference between the left hand side and the right hand side will find the residual, which is given in Equation 13. After that each processor will calculate its chi-squared statistic and send the result back to a master processor. The master processor adds the resulting chi-squared statistics up and either accepts or rejects the proposed parameters and amplitudes based on the Metropolis procedure. The potential parallelizing procedures for the Galerkin method are either through Message Passing Interface (MPI) or MPI with Graphics Processing Units (GPUs). The result is a new parallel ensemble method, which we call the super-ensemble method[8] because it combines the Monte Carlo search for parameters with an approximation to the ODE solution (by the Galerkin Method).

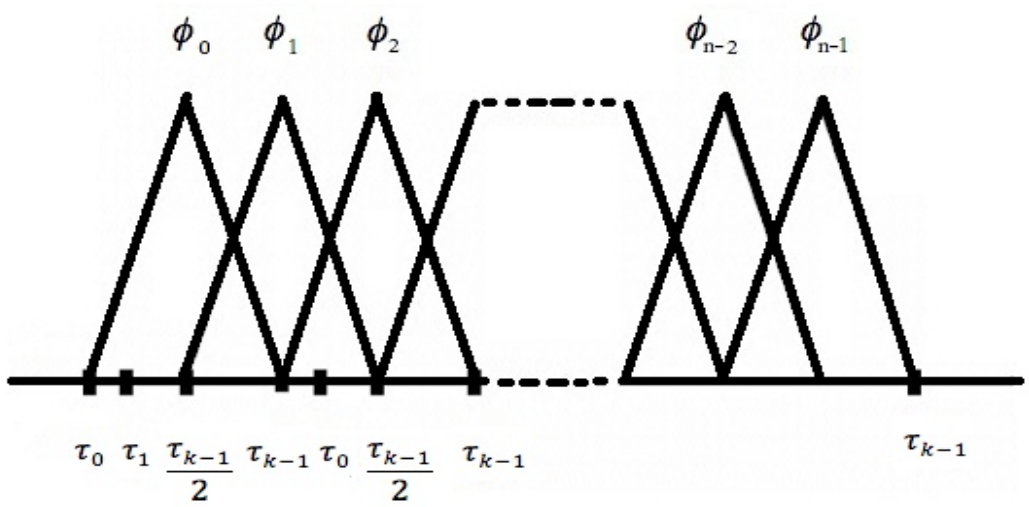

Figure 2.17: n subinterval with a hat function for each of these subintervals divided into k test grid points used to estimate the quality of the solution. One or more hat function(s) could be assigned to one slave processor.

2.4 CONCLUSION

A new method for solving systems of initial-value problems for nonlinear First-order Ordinary Differential Equations using the Galerkin finite elements method piecewise hat functions has been developed that gives as accurate a solution as the Adaptive Runge Kutta method when a large number of hat functions are used, and acceptable accuracy for the biological problems when a fewer number of hat functions is used. On the other hand, unlike the adaptive Runge Kutta method, this method has the potential to be parallelizable and to be useful for solving biological problems that depend on solving large systems of nonlinear ODEs describing genetic networks and other systems in engineering. Moreover, this method yields solutions not for arbitrary time steps but for desirable fixed time steps.

As shown above we produce trajectories from the Galerkin method comparable with the adaptive Runge Kutta method (with relative and absolute errors for the ARK are

equal to $1x10^{-13}$) by using a low number of hat functions (100 hat functions for the first two cases over the interval [0 10] and 1000 hat functions for the last case over the interval [0 200]). Developing this method to be faster than the traditional ODE solvers is a potential study in the future especially when biological problems with large networks are considered. Identifying large networks is complicated by having many parameters and limited data[35]. One solution to this problem is the use of ensemble methods[4]. A parallelized ODE solver enables faster sampling of the parameter space in ensemble methods to identify what we know (i.e., is supported across the ensemble) and what we do not know (i.e., is not supported across the ensemble) about a large system of ODEs.

2.5 ACKNOWLEDGEMENTS

This work was supported in part by the NSF under Grants NSF QSB-0425762 and the NSF DBI-1062213 and the Department of Systems Engineering and Medical Bioinformatics, Yarmouk University, Irbid, Jordan.

CHAPTER 3

SOLVING LARGE NONLINEAR SYSTEMS OF FIRST-ORDER ORDINARY DIFFERENTIAL EQUATIONS WITH HIERARCHICAL STRUCTURE USING MULTI-GPGPUS AND AN ADAPTIVE RUNGE KUTTA ODE SOLVER

Ahmad Al-Omari, Jonathan Arnold, Thiab Taha, Heinz-Bernd Schüttler

IEEE ACCESS Journal

Reprinted here with permission of publisher

Received October 7, 2013, accepted November 5, 2013, date of publication November 12, 2013, date of current version November 21,2013

Digital Object Identifier 10.1109/ACCESS.2013.2290623

ABSTRACT

The Adaptive Runge Kutta Method (ARK) on multi-General-Purpose Graphical Processing Units (GPGPUs or GPUs for short) is used for solving large nonlinear systems of first-order ordinary differential equations (ODEs) with over ~10,000 variables describing a large genetic network in systems biology for the biological clock. To carry out the computation of the trajectory of the system, a hierarchical structure of the ODEs is exploited, and an ARK solver is implemented in Compute Unified Device Architecture/C++ (CUDA/C++) on GPUs. The result is a 75-fold speedup for calculations of 2436 independent modules within the genetic network describing clock function relative to a comparable CPU architecture. These 2436 modules span one-quarter of the entire genome of a model fungal system, *Neurospora crassa*. The power of a GPU can in principle be harnessed by using warp-level parallelism, instruction level parallelism or both of them. Since the ARK ODE solver is entirely sequential, we propose a new parallel processing algorithm using warp-level parallelism for solving ~10,000 ODEs that belong to a large genetic network describing clock genome-level dynamics. A video is attached illustrating the general idea of the method on GPUs that can be used to provide new insights into the biological clock through single cell measurements on the clock.

INDEX WORDS: Bioinformatics, Biological clock, General-purpose graphical processing unit, finite element method, ordinary differential equation, adaptive Runge-Kutta integration, systems biology, warp-level parallelism.

3.1 INTRODUCTION

In a systems biology approach bridging genomics, bioinformatics, and engineering our goal is to explain the behavior of traits controlled by many genes, such as carbon metabolism, the biological clock, development, and cancer in terms of biochemical pathways found within living cells[11]. Since the 1990s, a variety of teams have assembled large maps of biochemical pathways in a variety of organisms with this goal in mind[12-14]. At the turn of the millennium it became possible to measure the dynamics of genomic-scale pathways spanning a whole living system[11, 15, 16]. We are now poised to describe the dynamics of an entire cell[17, 18]. A video is attached describing how this can be achieved through the integration of genomics, bioinformatics, and engineering[36].

Genetic networks describe time-dependent concentrations of molecular species, such as genes, their RNAs, and their proteins as well as their substrates[20]. These networks can be expressed as a system of coupled nonlinear first-order ordinary differential equations (ODEs). Understanding such networks enables us to discover the biochemistry and genetic activity of a cell and how the cell evolves as a function of time (including its metabolism, signal transduction, and cell cycle). Many problems could be solved and understood once these ODEs are identified. For example, human diseases like prostate cancer, the phenotype of other complex traits such as development[37], and the biological clock of an organism [5] could also be described. The most widely used approach to modeling these biochemical pathways are nonlinear systems of first-order ordinary differential equations[9].

GPUs have been used recently for solving computationally-intensive problems for many applications[38-41] including those in Bioinformatics[42, 43], numerical computations[44, 45], ray tracing[46],volume ray casting[47], computational fluid dynamics[48], and weather modeling[49]. Here we harness this new computing approach to develop new ODE solver methods employing Adaptive Runge Kutta Method (ARK)[34] on GPUs to simulate large genetic networks and ultimately identify these networks from available genomics data[4, 5, 19, 50].

A major proving ground for the new tools of systems biology has been the study of the molecular basis of the biological clock[51]. The key problem is linking the model identification of the clock to guiding expensive genomics experiments designed to identify the underlying network[9]. This model-guided discovery process, which we call computing life[5], requires the ability to simulate large nonlinear systems of first-order ODEs.

There are particular challenges to solving these ODEs. The system of ODEs is usually large. The experimental data are noisy and limited from molecular quantitative studies. More importantly, designing a new experiment is very expensive in terms of money (using genomics experiments) and time. To overcome the problem of many parameters and limited noisy data, new methods were developed for fitting these ODEs called ensemble methods[4, 9, 50]. The ensemble approach overcomes the limited genomics data on a particular network with many parameters by giving up on finding one best model. Instead, the search in the ensemble approach is for an ensemble of 40,000+

models consistent with the data. Averaging is then done over the ensemble to make predictions about the time-dependent behavior of the system. In order to implement these ensemble methods the ODE solver must be very fast!

Using the ARK method in the ensemble approach implies that the system of ODEs should be re-solved for each proposed ensemble Monte Carlo updating step, and solving for the time step t+h requires the solution at the prior t. For example, solving a genetic network as the one shown in Figure (3.1) [4] for the clock and constructing the ensemble of the unknown parameters that fit the experimental data needs a very large amount of time (i.e., 30 days on older processors). The diagram in Figure (3.2) specifies a much larger system of ODEs with hierarchical structure. There is a master clock module controlling 2436 slave modules each with 4 variables representing molecular species concentrations. We need a new approach to solve problems on this genomic scale.

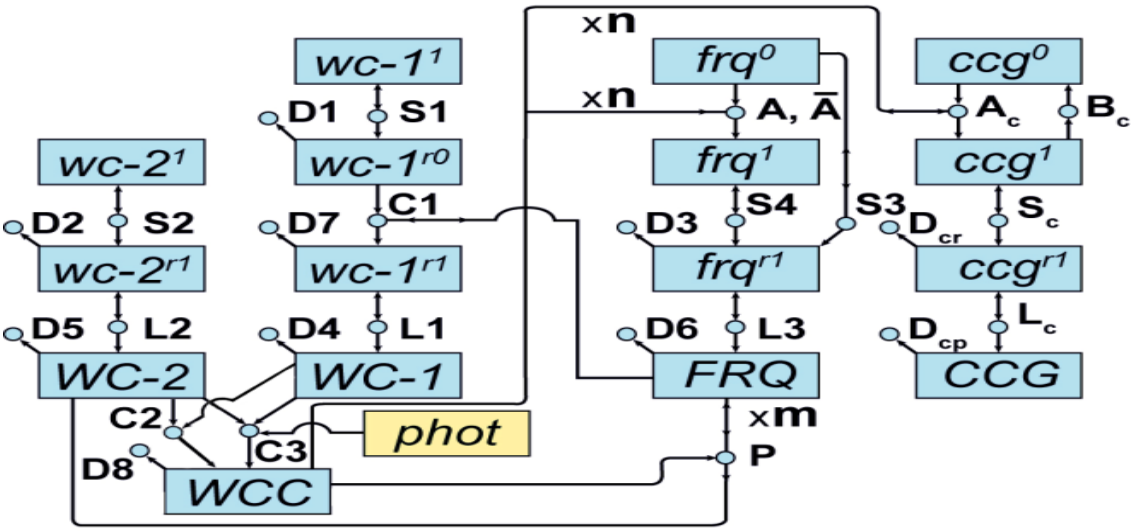

Figure 3.1: A genetic network for the biological clock from[4]. Molecular species (i.e., reactants or products) in the network are represented by boxes. The white-collar-1 (wc-1), white-collar-2 (wc-2), frequency (frq), and clock controlled gene (ccg) gene symbols are

sometimes superscripted 0, 1, r0, r1, indicating, respectively, a transcriptionally inactive (0) or active (1) gene or a translationally inactive (r0) or active (r1) mRNA. Associated protein species are indicated with capitals. A phot (in yellow) symbolizes the photon species. Reactions in the network are represented by circles. Arrows pointing to circles identify reactants; arrows leaving circles identify products; and bi-directional arrows identify catalysts. The labels on each reaction, such as S4, also serve to denote the rate coefficients for each reaction. Reactions labeled with an S, L, or D denote transcription, translation, or degradation reactions, respectively. Reactions without products, such as D7, are decay reactions. From[5].

Mainly, besides making these ensemble methods broadly available, our goal is to solve a genetic network shown in Figure (3.2) that consists of a master module (clock) and 2436 slave modules (subunits). Solving such a genetic network using a CPU implies that all of these subunits should be solved simultaneously and each subunit solved many times sequentially. This makes the process of finding the unknown parameters in the network using the ensemble method massively time consuming. In some cases where the network consists of 2436 subunits[5], the ensemble method is beyond the capability of the fastest serial computers. We developed an algorithm using the concept of warp-level parallelism[52] with a GPU and ARK method that makes possible simulating 2436 subunits under clock control with a speed up of about 75-fold relative to a solution of serial version on a CPU architecture. The code (see supplement for code + input file) is written in C++/CUDA computer language for the GPU and is written in C++ and compiled with g++ using –O2 and –O3 optimization flags for the CPU. What makes our approach attractive is that as more subunits and ODEs are added, the speed up achieved increases, if we consider the availability of the GPUs. The strategy we describe here for solving large nonlinear systems of first-order ODEs is an alternative to another ODE solver recently developed[53].

3.2 METHODS

The Warp-level parallelism concept is used to exploit and harness the power of a GPU for solving 2436 systems of ODEs using the ARK method for a large genetic network shown in Figure (3.2) describing the biological clock in *N. crassa*[5]. Since such a genetic network consists of many subunits and all of these subunits have the same mathematical form (as ODEs) as shown below but with different parameters, solving all of these systems of ODEs once in parallel suits the SIMD (single instruction, multiple data) and warp-level parallelism concepts (warp size for current NVIDIA GPUs is 32 threads). A common parallelization strategy in this category is to increase the number of warps and consequently the number of thread blocks (TBs) per streaming multiprocessor (SMX) on a GPU and decrease a TBs size (number of threads per block). In addition to the fact that this optimization strategy increases the number of thread blocks assigned to each SM, it provides more independent warps from other thread blocks when one warp is stalled [54]. Figure (3.2) shows 2436 systems of nonlinear ODEs (slave modules) that are needed to be solved to enable the implementation of the ensemble method with an ARK ODE solver[4]. The independence of these slave modules enabled us to suggest an algorithm to solve all of these modules in a parallel fashion using the ARK method and multi-GPGPUs.

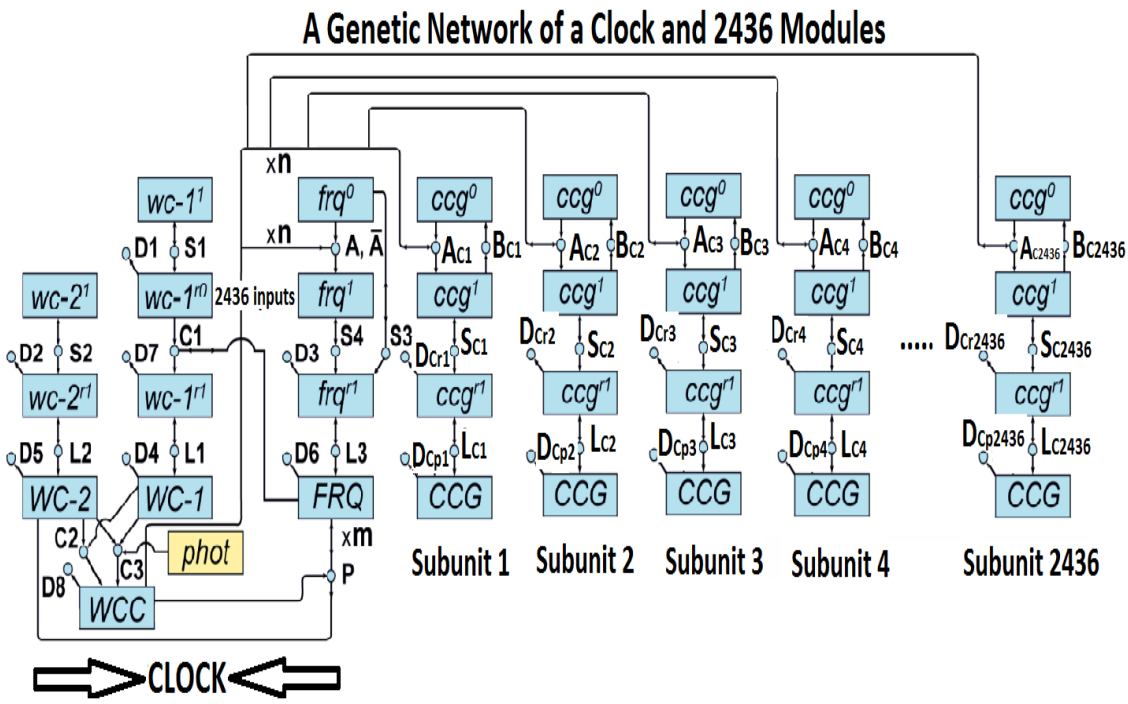

Figure 3.2:The whole genetic network consisting of 2436 slave modules (subunits) to be solved by the GPUs. The subunits are independent from each other but depend on the clock master module consisting of genes, wc-1, wc-2, frq, and their products. The subunits have the same mathematical form (ODEs) but different parameters. Identifying this huge genome scale network is beyond the fastest serial computer in the existence. Thus, the GPUs are necessary for solving such a network. This figure shows a modified genetic network for the biological clock from[4] in the genome. The notation to describe this network is the same as in Figure (3.1). An abbreviation of the notation for the clock controlled genes is now given: g0 = [ccg0] = concentration of ccg0; g1 = [ccg1] D concentration of ccg1; gr = [ccgr1] = concentration of ccgr1; gp = [CCG] = concentration of CCG.

In the Warp-level parallelism GPU(s) execute many warps concurrently. For example, on the Kepler K20x GPU, the maximum number of warps per SM equals to 64 warps, and the maximum number of TBs per SM equals to 16 TBs. Increasing the number of TBs and decreasing the block size is a well considered optimization strategy especially when the instruction level parallelism, i.e., thread code consists of multiple independent instructions in sequence, is hard to implement in some algorithms[54]. For example, the ARK method is in essence a sequential algorithm, and it is very hard to be parallelized by

instructional level parallelism because ARK doesn't have independent instructions in sequence and because the time taken by each warp is unpredictable[44]. To maximize the usage of warp level parallelism we use a warp per block to solve the dynamics of the slave module consisting of systems of nonlinear ODEs using the ARK method. The pressure of using a large number of blocks to solve our genetic network (2436 blocks) leads us to use multi-GPUs to increase the speed up as is shown in Figure (3.3) and Figure (3.4).

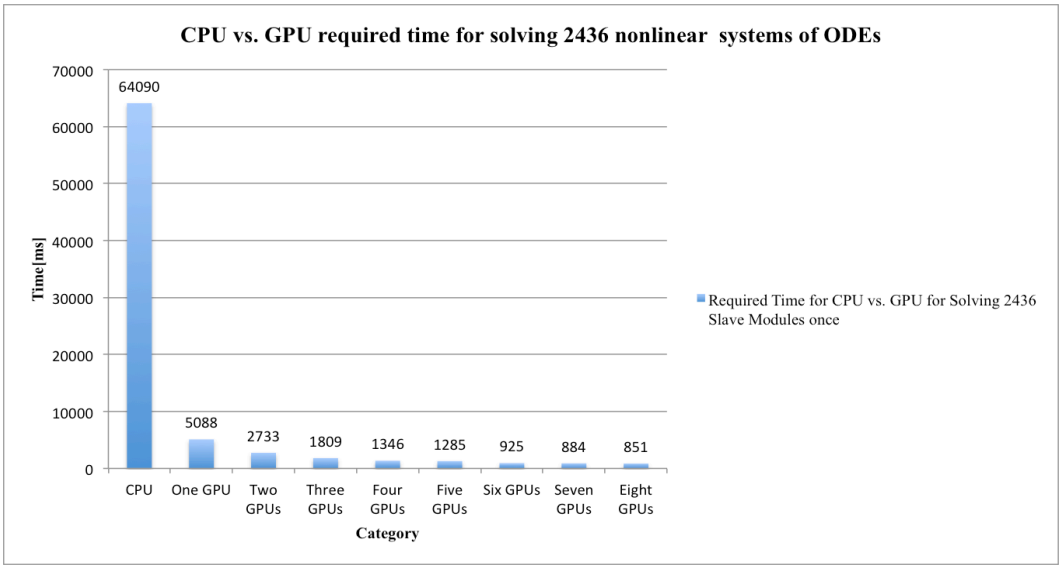

Figure 3.3: The time required for solving 2436 slave modules just one time using a NVIDIA GPU(s) [Kepler K-20x Tesla] over an extreme edition of optimized CPU [Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz]. Using four of the GPUs to solve our target genetic network 800,000 times shown in the Figure (3.2) to fit the observed data requires just twelve days while using the CPU requires a year and six months.

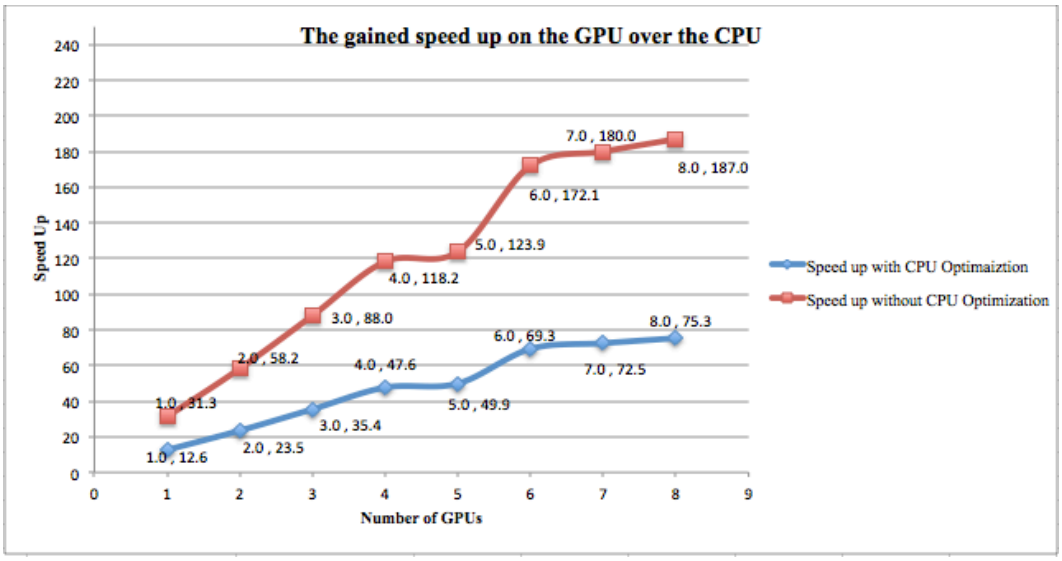

Figure 3.4: The achieved speed up using Multi-GPU [NVIDIA Kepler K-20x Tesla] over an extreme edition of CPU [Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz]. Red line shows the speed up without using the –O2 and -O3 flags for CPU optimization(C++ code/g++ compiler) and blue line shows the speed up with using –O2 and -O3 flags for CPU optimization.

3.2.1 THE ALGORITHM

Each slave module can be described as an initial value problem of a system of nonlinear first ODEs for a genetic network as is shown in Figure (3.2) and is specified by

$$\frac{dg_0}{dt} = B_c g_1 - A_c g_0 w(t)$$

$$\frac{dg_1}{dt} = A_c g_0 w(t) - B_c g_1$$

$$\frac{dg_r}{dt} = S_c g_1 - D_{cr} g_r$$

$$\frac{dg_p}{dt} = L_c g_r - D_{cp} g_p$$

The variables in this subsystem are the concentrations of the *clock-controlled genes* (g_0 and g_1 in the inactive and active state, respectively), their mRNAs (g_r), and proteins (g_p). The testing was done on Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz] Extreme Edition processor and a Tesla GPU [Kepler K-20x] to measure the speed up of our approach. A Kepler K-20x GPU handles double precision numbers and consists of 15 streaming multiprocessors (SMX), each (SMX) consisting of 192 SIMD cores and handling up to 16 TBs with restriction for 2048 threads per SMX. The idea of the algorithm is to assign each slave module to one TB consisting of 32 threads (a warp). The load of 2436 slaves modules (systems of nonlinear ODEs equations) are distributed equally across Multi-GPUs system with a potential for a slight decrease or increase in the number of slave modules for the last GPU. For example, using four GPUs implies that to launch a kernel (a function to be executed on the GPU) on each GPU involves configurations of grid size equals to 609 thread blocks and a block size of 32 threads with a total number of threads equals to 19,488 threads per GPU. Each warp is responsible for solving a system of equations for one slave module. The number of slave modules (292) that are strongly supported to be under clock control was determined by a series of model-guided experiments[5].

3.2.2 THE PROCEDURE

a) Copying a file to the constant memory of each GPU that consists of a 200 time point solution to interpolate for the variable w(t) appearing in the equations above. Those time points come from the master clock module and are passed to each slave module as is shown in Figure (3.2). All of the TBs need to access the same file in the constant memory.

b) Each thread block executes a kernel, which contains the ARK ODE solver.

c) All threads in the same block hold the same data (i.e. initial conditions and parameters values), and execute the same system of equations for an assigned slave module.

d) ARK's constants are declared in the constant memory of the GPU and are seen by all of the threads.

e) Kernel configuration consists of a grid size equal to the number of slave modules (609 slave modules per GPU) and a block size equals to a warp size.

Parameter values of this clock network problem are given in Table (3.1); all of the slave modules have the same set of parameters for the sake of simplicity and accuracy, and calculating performance of a CPU and GPU.

Table 3.1: Initial conditions at t=0 and parameters values

Species	Initial Conditions	Parameters	Values
g_0	13.4271	Ac	0.3005
g_1	13.4348	Bc	37.2048
g_r	1.2208	Sc	0.0086
g_p	2.0982	Lc	11.4377
		Dcr	0.4105
		Dcp	0.3589

3.3 RESULTS

Identifying a large clock genetic network is beyond the capabilities of the fastest CPU ever manufactured and needs a much more expensive super computer than the GPUs that we use with high capabilities to solve such a network. The proposed parallel procedure uses NVIDIA Kepler K-20x GPU(s) to solve a genetic network shown in

Figure (3.2) within a very short period of time compared with the time required on a CPU as is shown in Figure (3.3). The Ensemble method mentioned before needs this genetic network to be solved 40,000+ sweeps for an equilibration stage (each sweep is equivalent to solving the genetic network 10 times so that on average each variable out of ten variables in a slave module is updated once) and 40,000+ sweeps for an accumulations stage. The total number of sweeps to identify the genetic network of 2436 slave modules is shown in Figure (3.2) is 80,000+ sweeps. From Table (3.2), Figure (3.3), and Figure (3.4), the solution for 2436 slave modules from over 80,000 sweeps needs a CPU time of about one year and 6 months (considering 64090 milliseconds (ms) is needed to solve the 2436 slave modules once), while solving the same number of slave modules using just 4 GPUs needs about 12 days (considering 1346 ms is needed to solve the 2436 slave modules once), which is feasible and doable. Algorithm performance appears to plateau for 2436 slave modules somewhere between 4 and 6 GPUs in Figure (3.4). The genome dynamics of 295 *clock-controlled genes* over a 48 hour window are displayed in the attached video[36].

Table 3.2: A comparison of the simulation of a large clock genetic network on Kepler K-20x GPUs vs. a Quad-cores CPU [Intel(R) Core(TM) i5-2400 CPU@ 3.10GHz] Extreme Edition Processor with 2436 clock-controlled genes providing 2436 slave modules to the system of ODEs. The time required for solving the whole 2436 slave modules once using –O2 and –O3 optimization flags on the CPU equals to 64090 ms while without optimization flags equals to 159150 ms

#of GPUs	GPU Time [ms]	Speed up With –O2, -O3 Flags	Speed up Without –O2, –O3 Flags
1	5088	12.59630503	31.27948113
2	2733	23.45042078	58.23271131
3	1809	35.42841349	87.97678275
4	1346	47.61515602	118.2392273
5	1285	49.87548638	123.8521401
6	925	69.28648649	172.0540541
7	884	72.5	180.0339367
8	851	75.31139835	187.0152761

3.3.1 ALGORITHM PERFORMANCE AS A FUNCTION OF THE NUMBER OF SLAVE MODULES IN THE CLOCK NETWORK

In previous work, we identified a total of 2436 genes that were circadian in the *N. crassa* genome[5], which is considerably more than the 292 reported *clock-controlled genes*[5]. The simplest null hypothesis to be investigated is that all 2436 genes that have a WCC binding site and are circadian in expression are in fact all *clock-controlled genes*[55]. To test this hypothesis with the ensemble method would involve being able to simulate the clock mechanism +2436 slave modules. We now do this and examine the computational time of the algorithm as a function of the number of slave modules up to 2436 such modules.

In Figure (3.5) we depict the relationship between different numbers of slave modules and time required for solving them using a fixed number of GPUs. Providing a brief introduction for the Kepler K-20x Tesla architecture and the programming model helps to elucidate the results in Figure (3.5) and Figure (3.6). The Kepler K-20x consists of 15 SMX with a maximum of TBs per streaming multiprocessor equals to 16 TBs. Thus,

theoretically the device (GPU) can hold and run about 240 TBs simultaneously, given sufficient hardware resources, such as a register file and shared memory. Considering the sufficient resources, TBs do not leave a streaming multiprocessor until its execution is finished [56], and once TBs finish their time span, the scheduler keeps launching new TBs on these vacated SMXs for this kernel until all of the TBs have executed. For example, from Figure (3.5), on one hand, the time required on a single GPU for solving 200 or (800/4) slave modules or TBs (each slave module assigned to exactly a TB) equals to 462ms. On the other hand, the time required for solving 250 TBs (slave modules) needs 823ms, almost double the amount of time. This jump in the time is due to the fact that the device should solve theoretically 240 slave modules simultaneously until they finish their execution, and then it should start a new pass by solving the next 240 TBs or the rest of the available TBs simultaneously. Therefore, in this case, the 10 TBs difference need almost the same amount of time that is required for solving of 240 slave modules. As a matter of fact, although the theoretical number of TBs that can be run simultaneously on the device equals to 240, in our algorithm and as it is shown in Figure (3.5) and Figure (3.6), the number of blocks that can run simultaneously on the device is about 224 TBs, and after this number of blocks a jump in the time occurs for even one thread block more. The above clarifications should explain the scenario occurring on six GPUs as is shown in blue Figure (3.5) and Figure (3.6). Based on the fact that the time of the CPU is monotonically increasing with respect to the number of slave modules to be solved serially, then the drop in the speed up is shown in Figure (3.6) from time to time should be due to the jump in the time as is shown in Figure (3.5) given that the speed up=Time used by the GPU(s)/Time used by the CPU.

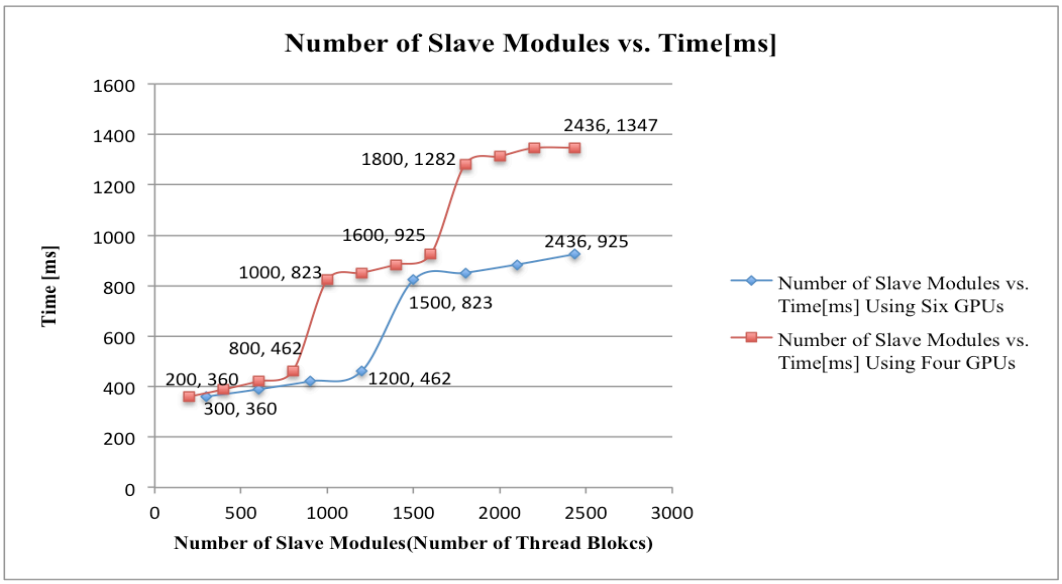

Figure 3.5: The relationship between number of thread blocks (slave modules) to be solved on the device and the required time. The jump in time is due to exceeding the maximum number of TBs running simultaneously on the device. If GPU is capable of running several N blocks simultaneously. Then from 1 to N blocks takes same time to complete.

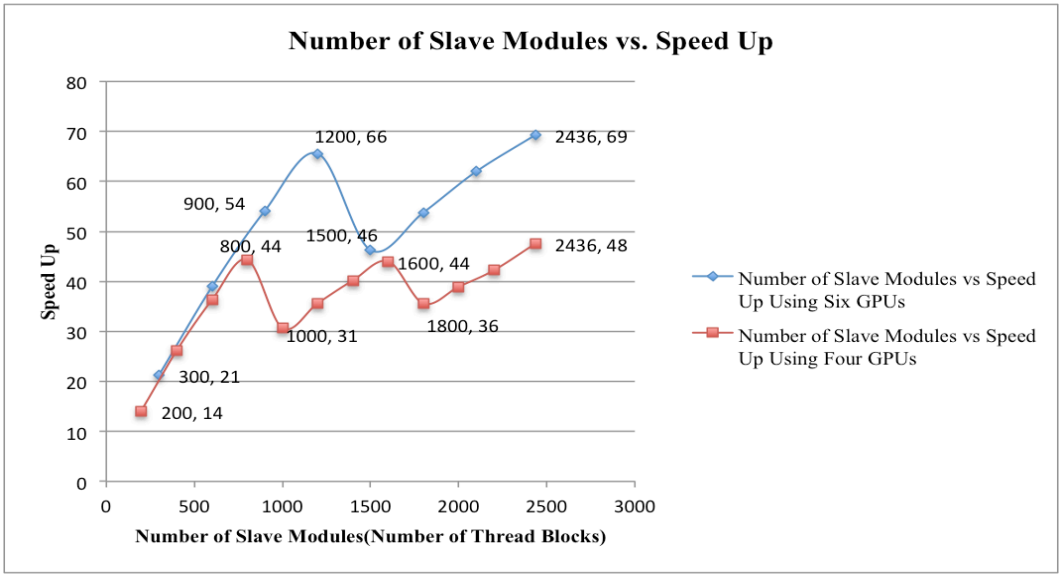

Figure 3.6: The relationship between number of thread blocks (slave modules) to be solved on the device and the speed up. The drop in the speed up is due to exceeding the maximum number of TBs running simultaneously on the device, given that the time of the CPU is monotonically increasing.

3.4 DISCUSSION

This new parallelization strategy for solving large systems of ODEs on GPUs opens up the possibility of simulating genome dynamics. The parallelization strategy means that it is now feasible potentially to use ensemble methods for fitting genome-scale genetic networks when they have hierarchical structure. The need for such methods stems from genomics data being noisy and sparsely distributed across the genome[4]. The key to this parallelization strategy is to identify hierarchical structure in the genetic network. There is some experimental evidence that this hierarchical structure may be widespread[57], and has been argued to be a feature of the clock network[55]. This approach can be coupled with classic well-performing solvers, such as ARK, to make possible model-guided discovery about genetic networks on a genomic scale[5].

The GPU is used in [58] to solve a system of ODEs for another oscillatory system, and the maximum achieved speed up is 47-fold using a LSODA solver[59]. This system of ODEs consists of 3 species, 6 reactions, and 8 parameters. In this paper, our proposed parallel algorithm uses an ARK ODE solver to solve systems of ODEs and operates on a much larger genome scale, consisting of 9744 species, 14616 reactions, and 24360 parameters. Our GPU implementation also leads to a higher speed up reaching up to 75-fold.

Our approach can also be compared with an adaptive step size GPU ODE solver for simulating electric cardiac activity[60]. Garcia et al.[60] developed a method for solving systems of ~300 ODEs describing cardiac activity with a single precision accuracy and a

speed up of 9.07-fold while in our work here we solved a system of ~10,000 ODEs using a double precision and a speed up of 75-fold.

An inherent limitation of the parallelization strategy here is the use of sequential ODE solvers, such as the ARK method. In order to calculate the trajectory at time t+h, it is necessary to have solved with high accuracy the trajectory at time t first. The ARK method is then inherently sequential. We recently developed an alternative parallelization strategy using a Galerkin Finite Element Method with Hat Functions as ODE solver[53], which is as accurate as the ARK method. It will be interesting to see how this alternative compares with parallelization using the hierarchical structure of the network. In comparison to [53], here in this paper another procedure using the ARK method along with the power of GPUs are functioning to solve the large genetic network using warp level parallelism, while in the Galerkin approach[53] is using instruction level parallelism alongside warp level parallelism in the GPUs. The only caveat is that the serial version of the Galerkin ODE solver is slower than ARK method and it is harder to implement on the GPUs.

A third parallelization strategy might involve a parallel implementation on multiple CPUs with MPI to narrow the gap in performance between CPUs and GPUs in Figure (3.4). We think the GPU is the preferred approach here for four reasons. One, using MPI across multiple CPUs needs hundreds of CPU cores, a more expensive strategy than multiple GPUs. Two, the thread on a GPU is very "lightweight" if it is compared with the thread on a CPU. Three, sometimes launching a certain number of

threads on a CPU degrades the overall performance especially if the number of threads exceeds the number of cores. Finally, the best practice for implementation of ensemble methods has been serial implementations of solvers, and so we want to ascertain how the use of a GPU based ODE solver changes the speed of the ensemble method relative to best current practice.

This methodology of the ARK method on a GPU is enabling a new approach to understanding the kinetics of the cell on a genome scale. As captured in the attached video, new approaches in nanotechnology are enabling the measurement of the clock in single cells[36]. This will open a whole new area of inquiry about the clock. We can begin to ask if the clock is truly stochastic with variation in the oscillators from cell to cell and whether or not there is any cell-to-cell communication of oscillators in different cells. To address these questions will require the solution to three methodological challenges. New engineering approaches will be needed to make single cell measurements[61]. As indicated in the video, this approach involves capturing individual cells with microfluidics technology. New models will be needed to incorporate stochastic behavior in the clock models [62]. While stochastic clock models have been proposed, they have no empirical basis and have not been tested. Third, new parallelization strategies will be needed to understand the large networks describing clock behavior. The clock network in a single cell could involve potentially 2436 distinct genes responding to the clock mechanism or equivalently, ¼ of the genome[5]. In this paper, we have introduced the second of two strategies to address the fitting of genome scale networks by the ensemble method. The time estimates in Table (3.1) implies that it

is now feasible to fit a genome-scale network to genome dynamics within a single cell, like that of the clock, with ensemble methods needed to overcome noisy data that is sparsely distributed across the genome. A 75-fold speedup of the simulation of a hierarchical network on GPUs was achievable Table (3.1). This speedup is sufficient to fit the entire clock network to the genome dynamics of a single cell.

3.5 CONCLUSION

In this paper we harness the power of multi-GPGPUs to solve many systems of nonlinear ordinary differential equations that belongs to a large genetic network describing clock genome-level dynamics using an Adaptive Runge Kutta ODE solver. Implementing the proposed algorithm opens up a door to utilize the ensemble approach to overcome the problem of many parameters and limited noisy genomics data on a particular network. Consequently, understanding such networks enables us to discover the biochemistry and genetic activity of a cell and how the cell evolves as a function of time (including its metabolism, signal transduction, and cell cycle).

3.6 ACKNOWLEDGEMENT

This work was supported in part by the NSF under Grants NSF QSB-0425762 and the NSF DBI-1062213 and the Department of Systems Engineering and Medical Bioinformatics, Yarmouk University, Irbid, Jordan.

CHAPTER 4

DISCOVERING REGULATORY NETWORK TOPOLOGIES USING ENSEMBLE METHODS ON GPGPUS WITH SPECIAL REFERENCE TO THE BIOLOGICAL CLOCK OF *NEUROSPORA CRASSA*

Ahmad Al-Omari, James Griffith, Michael Judge, Thiab Taha, Jonathan Arnold, &H-Bernd Schüttler

IEEE ACCESS Journal

Reprinted here with permission of publisher

Received Jan 16, 2015, accepted February 1, 2015, date of publication February 3, 2015, date of current version February 18, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2399854

ABSTRACT

Most genetic networks, such as that for the biological clock, are part of much larger modules controlling fundamental processes in the cell, such as metabolism, development, or response to environmental signals. For example, the biological clock is part of a much larger network controlling the circadian rhythms of about 2,418 distinct genes in the genome (with 11,000 genes) of the model system, *Neurospora crassa*. Predicting and understanding the dynamics of all of these genes and their products in a genetic network describing how the clock functions is a challenge and beyond the current capability of the fastest serial computers. We have implemented a novel "variable–topology supernet" ensemble method using Markov Chain Monte Carlo (MCMC) simulations to fit and discover a regulatory network of unknown topology composed of 2,418 genes describing the entire clock circadian network, a network that is found in organisms ranging from bacteria to humans, by harnessing the power of the GPGPU and exploiting the hierarchical structure of that genetic network. The result is the construction of a genetic network that explains mechanistically how the biological clock functions in the filamentous fungus *N. crassa* and is validated against over 31,000 data points from microarray experiments. Two transcription factors are identified targeting ribosome biogenesis in the clock network.

INDEX WORDS: Biological clock, General-purpose graphical processing unit, ensemble method, supernet, systems biology, and regulatory network topologies.

4.1 INTRODUCTION

Systems biology provides a pathway-centered approach to understanding complex traits, such as carbon metabolism[11], development[63], and cancer[64]. A major problem in systems biology is reconstructing such pathways on a genomic scale[65]. Genome-wide reconstruction of networks is particularly limited by three difficulties: (1) genomics data are sparse and noisy with respect to the trait of interest; (2) the underlying network is large; (3) the network topology is usually unknown. To overcome the first problem, ensemble methods were introduced to identify genetic networks even in the presence of sparse and noisy data[4, 50]. The key innovation was relinquishing finding one best model consistent with the data, but instead, identifying an ensemble of models consistent with the data available to predict systems behavior over time. To overcome the second problem new parallel computing strategies were developed for ensemble methods on General-Purpose Graphics Processing Units (GPGPUs)[66, 67]. The models being fitted were described by systems of nonlinear ordinary differential equations (ODEs) as a first approximation, and the solution to the second problem involved finding new ways to solve large (~10,000 variables) first-order nonlinear ODEs. Here we introduce a novel computational approach of network ensemble discovery that addresses all three problems including identifying a network of unknown topology on a genomic scale and illustrate its application to the large network of genes and their products associated with circadian rhythms[66]. The associated biological question is: By what mechanisms do such a small module as the clock mechanism, comprising only a single transcription factor (TF) complex, exert control over nearly one quarter of the entire genome? In the example below, the genomic scale network for the clock has ~38,000 parameters and ~31,000 data

points. Yet by averaging over the 40,000 members of the ensemble insights can be obtained into how the circadian network is organized, and predictions can be made about the whole system of 2,418 genes[5]. A video summary of the solution to this central problem in systems biology is attached.

4.1.1 MODEL

Our model below has two components, an unknown regulatory component (circles and ellipses in Figure (4.1) and Figure (4.2)) and the modules that control the expression of individual genes under clock control (boxes in Figure (4.1) and Figure (4.2)) with partially unknown regulatory topology. Our approach here rests on the key idea that any network we may wish to reconstruct, hence referred to as the "true net", can be represented as a particular special case of a much more general network model, referred to as the "variable-topology supernet". This is illustrated in Figure (4.1) and Figure (4.2) for a simple hierarchical transcriptional network, in which the clock transcription factor WCC regulates 5 other transcription factors which in turn may regulate hypothesized clock-controlled genes (*ccgs*) or be regulated by WCC directly. Such genes will be referred to as putative *ccgs*.

The hypothetical true net in Figure (4.2) consists of specifying how each putative *ccg* is regulated. The genetic network thus has an unknown true topology. Its supernet in Figure (4.1) consists of the six regulatory species regulating each putative *ccg.* The network kinetics, i.e., how fast each gene is converted into its product(s), is then governed by the reaction rate coefficients assigned to each reaction link, both in the true net and in the supernet. Clearly, the kinetics of the supernet will become identical to that

of the true net if we set to zero all supernet rate coefficients for those supernet reaction links which are absent in the true net; and if we set the supernet rate coefficients to the corresponding true net values for the subset of links that are present in the true net. We refer to this situation by saying that the true net of Figure (4.2) is "embedded" in the supernet in Figure (4.1).

The crucial point is that any network involving 2,418 genes can be embedded in the supernet of Figure (4.1). While the true net is initially unknown, we can perform a supernet ensemble simulation to find a MCMC sample of rate coefficients[4, 7] that is consistent with the data. The simplest hypothesis is shown in Figure (4.3): the clock transcription factor, WCC, alone is hypothesized to regulate all 2,418 genes.

Figure (4.1) and Figure (4.2) show the whole genetic network consists of 2,418 slave modules and a master module for the clock mechanism. There are 5 transcriptional regulators under clock control of the clock transcription factor, WCC, to be solved on the GPU[4, 66].

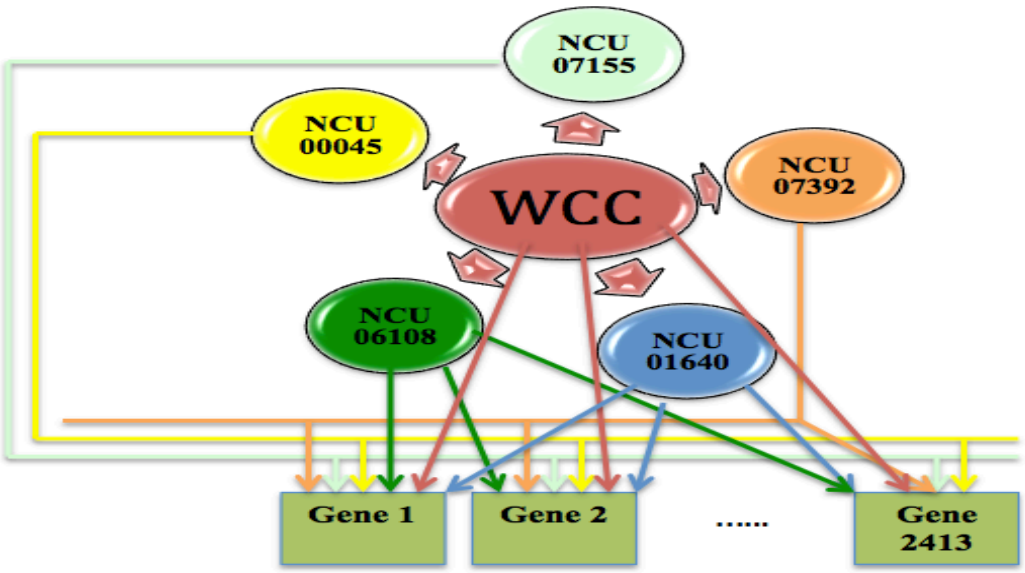

Figure 4.1: Supernet. Where each of the 2,418 genes is hypothesized to be regula4ted, potentially, by all of the six active transcription factors.

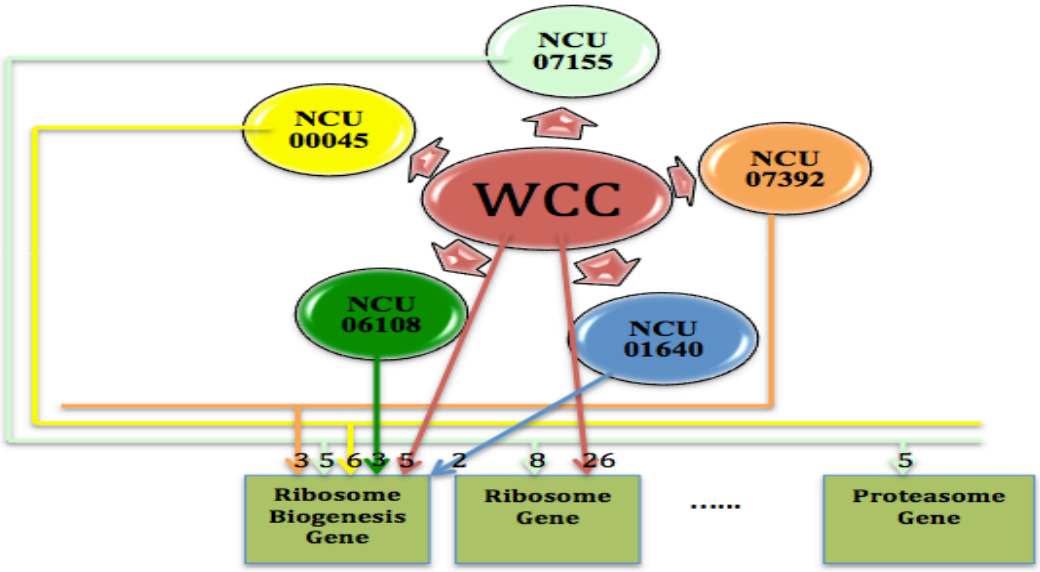

Figure 4.2: True net. The inferred regulation from fitting the supernet to available data by the ensemble method. Each of the 2,418 genes is inferred to be regulated by some of the six transcription factors.

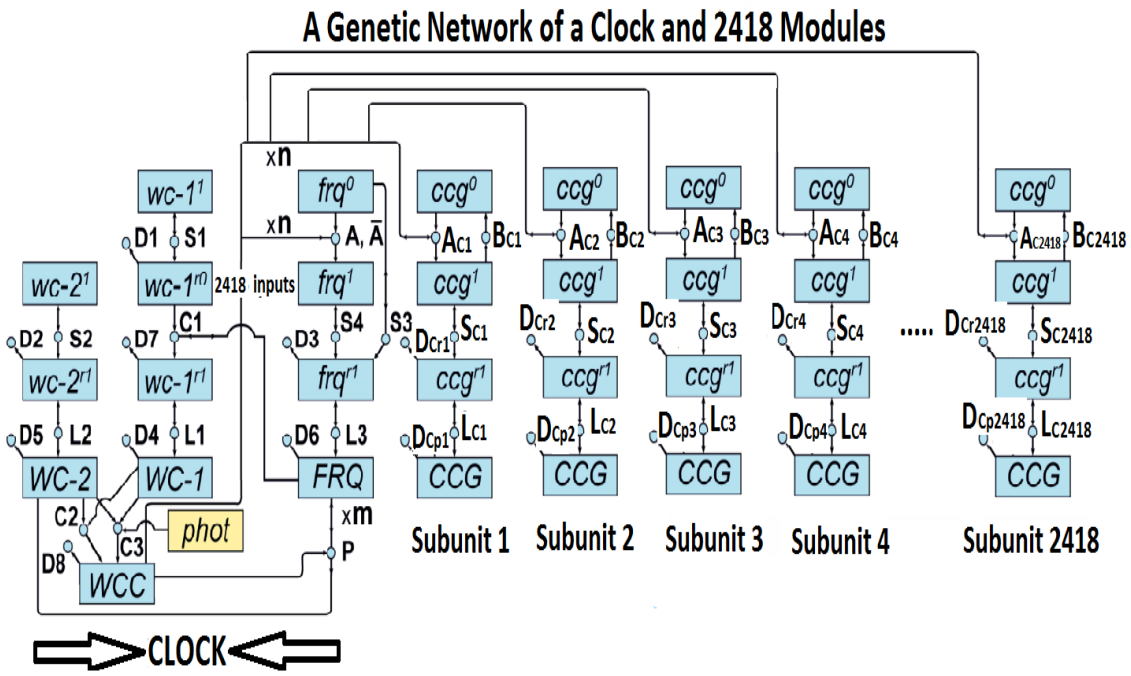

Figure 4.3: The simplest model is one in which all 2,418 genes are regulated by one transcription factor, WCC. Molecular species (i.e., reactants or products) in the network are represented by boxes. The white-collar-1 (wc-1), white-collar-2 (wc-2), frequency (frq), and clock controlled gene (ccg) gene symbols are sometimes superscripted 0, 1, r0, r1, indicating, respectively, a transcriptionally inactive (0) or active (1) gene or a translationally inactive (r0) or active (r1) mRNA. The notational convention for protein species is all capitals. A phot (box in yellow) symbolizes the photon species. Reactions in the network are represented by circles. Arrows pointing to circles identify reactants; arrows leaving circles identify products; and bi-directional arrows identify catalysts. The labels on each reaction, such as S4, also double as the rate coefficients for each reaction. Reactions with an A or B label are either activation or deactivation reactions. Reactions labeled with an S, L, or D represent transcription, translation, or degradation reactions, respectively. Reactions without products, such as D7, are used to indicate decay reactions. Reaction labeled C1 produces an alternative mRNA for the wc-1 gene. Reactions labeled C2 or C3 form complexes (WCC) with or without light. All of these reaction labels double as rate constants. In the more realistic model studied in this paper, an additional five proteins, shown as "ellipses" in Figure (4.1), are hypothesized as potential alternative regulators for each ccg. Redrawn from Al-Omari et al.[66]

4.1.2 THE SUPERNET

The genetic network exemplified in Figure (4.3) is one example of a genetic network consisting of a master clock module controlling 2,418 slave modules (putative

ccgs and their products). This network specifies a system of nonlinear first order differential equations under mass action kinetics. The rate coefficients are the labels on the reactions in Figure (4.3). A gene, such as frq^0, in the off state is activated into the on state, frq^1. This active gene is transcribed into a messenger RNA (mRNA), frq^r. The mRNA is translated into a protein, FRQ. The FRQ protein is the digital readout to the cell on the time of day. IF FRQ is high, it is dusk; if FRQ is low, it is dawn. The FRQ protein is the oscillator of the system. In turn the genes *wc-1* and *wc-2* are transcribed, and their mRNAs, translated, to produce the activator (WCC), which starts the oscillator. One of the functions of the FRQ protein is to destroy the activator WCC. The result is a negative feedback loop that contributes to oscillations. The nonlinear component of the entire network in Figure (4.3) is the clock mechanism in Figure (4.3) leading to oscillations.

All *ccg* slave modules have the same mathematical form of ODEs shown below[4], but each has its own independent values, treated as ensemble MC variables[4, 8, 9, 50], for rate coefficients and initial conditions, as follows:

$$\frac{dg_0}{dt} = B_c g_1 - A_c g_0 S(t) \quad (1)$$

$$\frac{dg_1}{dt} = A_c g_0 S(t) - B_c g_1 \quad (2)$$

$$\frac{dg_r}{dt} = S_c g_1 - D_{cr} g_r \quad (3)$$

$$\frac{dg_p}{dt} = L_c g_r - D_{cp} g_p \quad (4)$$

The dynamical variables here are the concentrations of the putative *clock-controlled gene* (g_0 and g_1 in the inactive and active state, respectively), its mRNAs (g_r), and its protein (g_p). See Figure (4.3) legend for parameters A_c, B_c, S_c, L_c, D_{cr}, and D_{cp} descriptions.The supernet allows for possible regulation of the putative *ccg* by any one of the six possible *ccg*-regulators, with S(t) denoting the weighted average of activator protein signals from all six regulators:

$$S(t)= \mu_0[\mathrm{Reg}_0(t)]^m + r_0^m \sum_{k=1}^{5} \left(\left[\frac{1}{r_k}\right]^m \mu_k[\mathrm{Reg}_k(t)]^m\right) \quad (5);$$

The "relative binding strength", μ_k, is the weight of the k^{th} regulator's contribution to the targeted ccg's activation signal, with $\mu_k \geq 0$ and $\sum_{k=0}^{5} \mu_k = 1$; $[\mathrm{Reg}_k]$ is the concentration of regulator protein k; r_k is the time-average of $\mathrm{Reg}_k(t)$ or the initial conditions defined below for k=0,…,5 (see Materials and Methods); and m is the Hill coefficient for the regulators. Regulators k=1-5 are themselves ccg products, assumed to be regulated by WCC, i.e., having a fixed μ_0=1. For any other, non-regulatory *ccg*, the μ_k, are ensemble MC variables, randomly varied to fit the data.

4.1.3 THE PARALLEL ALGORITHM FOR IMPLEMENTING THE ENSEMBLE METHOD ON A GENOMIC-SCALE GENETIC NETWORK OF UNKNOWN TOPOLOGY

An ensemble method was used previously[4] to identify the clock mechanism embedded in Figure (4.3) and took 60 days of simulation using older CPUs[4]. The ensemble method was used herein to describe and discover the other larger part Figure (4.3) (labeled subunits 1 through 2,418), consisting of a suite of 2,418 *circadian genes*.

We previously designed a method to solve a large genetic network consisting of systems of ODEs with ~10,000 dynamical variables, using warp-level parallelism in which each

block has a warp (32 threads) and in which Adaptive Runge Kutta[34] **(**ARK) is used to solve the system of ODEs with shared memory techniques. Here, we have developed significantly improved and more efficient algorithms that are applicable for simulating genetic networks, where putative *ccgs* are regulated independently of each other and each *ccg* maintains its own data (not using shared memory). Figure (4.4) shows a comparison between the previous algorithm[66] (in red), which uses ARK with a block of 32 threads and shared memory to solve for a slave module, and the current algorithm(in blue) ,which uses ARK with a single thread to solve for a slave module. The speed up of the current algorithm using a single GPU for solving 2,418 systems of ODEs once in Figure (4.4)is equal to 252-fold over a comparable CPU described [66], whereas the speed up of the previous algorithm to solve the same number of ODEs (~2,418) using 8 GPUs is equal to 75-fold over the same CPU. Although the current ARK algorithm is much faster, the previous one has an advantage over the current one in that it is applicable for a genetic network that has a dependency between genes due to the use of the shared memory. These two ARK algorithms can be applied to solve for both linear ODEs (the putative ccg genes) and non-linear ODEs (the clock mechanism) of genetic networks in Figure (4.3), implemented on GPUs, are now described.

A. THE ADAPTIVE RUNGE KUTTA ALGORITHM FOR SOLVING GENERAL ODE SYSTEMS ON THE GPU

The algorithm uses a common parallelization strategy that increases the number of thread blocks (TBs) per streaming multiprocessor and decreases thread blocks size (number of threads per block); for instance, we use a block of 32 threads. This provides more independent warps from other thread blocks when one warp is stalled[54]. In essence,

this new algorithm implies that each thread is responsible for solving a system of ODEs (linear or non-linear) with different parameters and initial conditions using a modified ARK algorithm that accommodates the GPU architecture. This algorithm utilizes registers instead of shared memory since there is no dependency in the data among slave modules. The achieved speed up is shown Figure (4.4) and Figure (4.5) with the GPU time as function of the number of slave modules (putative ccgs). An ensemble solution routinely requires 80,000 sweeps[4], each sweep being equal to the number of unknown parameters (16 parameters for each system of ODEs x 2,418 slave modules). Solving for 2,418 slave modules once on a CPU took 59,515 milliseconds as shown in Figure (4.5). The predicted CPU time for one ensemble run is 2.4 years. In contrast, solving the same number of slave modules using a single GPU and this new parallel algorithm needed about four days (considering 236.5 ms is needed to solve the 2,418 slave modules once as shown in Figure (4.5)). However, the time of 236.5 ms varies because MCMC will generate for every proposed update a different set of parameters for the ODEs being solved. Hence the number of iterations in the ARK method will increase or decrease based on an ODEs' set of parameters. We identified this genetic network in Figure (4.1) using the ARK parallelized algorithm and the MCMC algorithm in about 4 months. The algorithm is detailed in Materials and Methods.

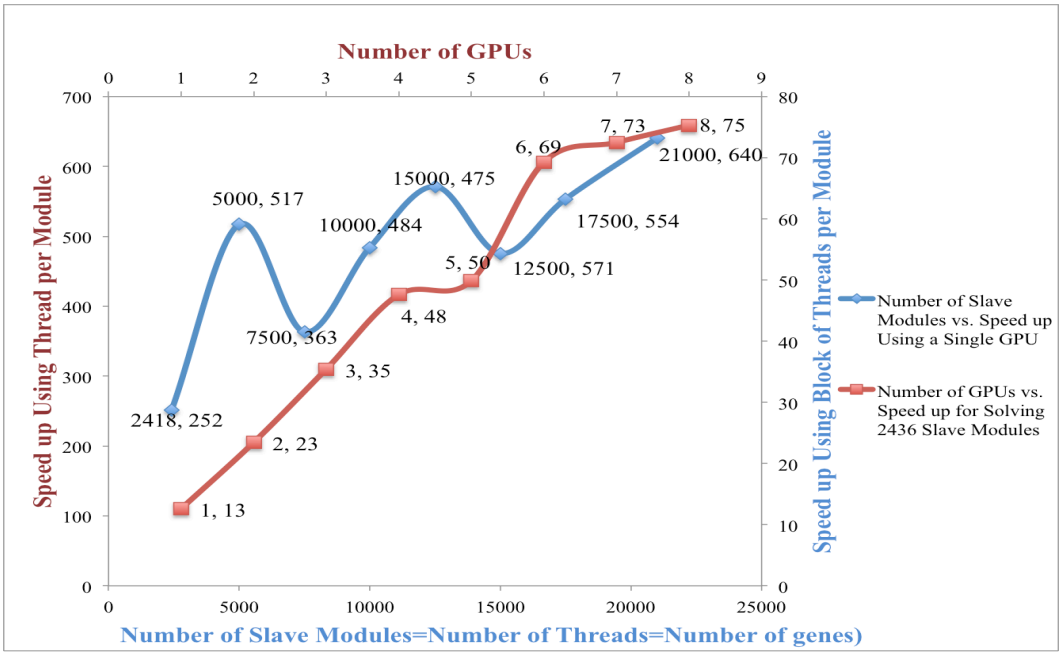

Figure 4.4: Our new algorithm with no shared memory and one slave module per thread (in Blue) and with a speedup of 250 to 650-fold outperforms our published algorithm with shared memory (in red) and a block of 32 threads per slave module and with a 13-75 fold speedup. The performance of each algorithm is given both as a function of the number of GPUs and the number of slave modules. These algorithms compute the dynamics of genomic scale networks on GPU(s) and make tractable ensemble methods on genomic scale networks. The dips in the speed up are due to exceeding the maximum number of thread blocks running simultaneously on the device, given that the time of the CPU is monotonically increasing as in Figure (4.5).

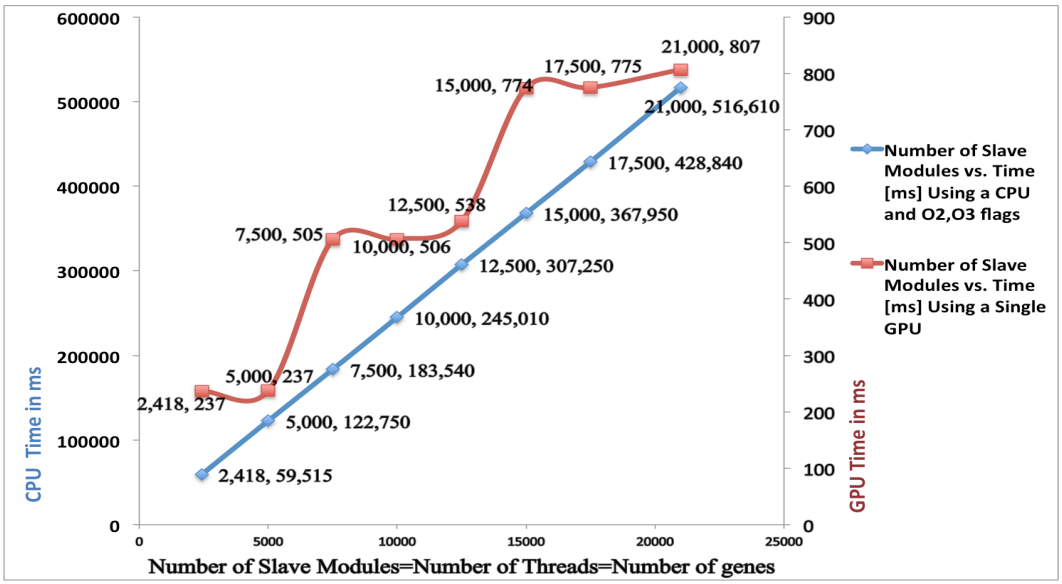

Figure 4.5: The time to solve a genome scale network makes tractable ensemble methods for genome scale networks on a GPU. The number of thread blocks (slave modules) to be solved on the device (Red) and on the CPU (Blue) determines in part the required computational time. The left Y-axis shows the CPU time while the right Y-axis shows the GPU time. The dips in the GPU time are due to exceeding the maximum number of thread blocks running simultaneously on the device. If a GPU is capable of running N blocks simultaneously, then from 1 to N blocks takes the same time to complete. The time of the CPU is monotonically increasing as function of slave modules.

B. THE NUMERICAL EXACT INTEGRAL SOLUTION ALGORITHM FOR SOLVING LINEAR ODE SYSTEMS ON THE GPU.

Since the ARK method still needs significant time for solving this large system of ODEs and since an alternative and a control to the ARK method is desirable, an independent numerical exact integral (EI) solution for solving the first order linear ODEs in Eqs. (1-4) was implemented using Gauss-Legendre quadrature[34] which fits exactly our ODEs system described above in this paper, (see Materials and Methods). We found no difference in the resulting solutions between the two methods as shown in Figure (4.6), where the max absolute error was 10-4 using just 32 Gauss-Legendre quadrature points. This solution's accuracy is sufficient for these biological problems[8]. The speed of the

numerical ODE solution did not depend on the set of parameters in a MCMC update while it does in the case of ARK method. With the numerical exact integral solution algorithm we solved the whole problem on the GPU within ~12 hours per MCMC, instead of 4 months by ARK: a 240-fold speed up over ARK on the same GPU. However, with both methods being highly accurate, the EI method applies only to first-order linear ODEs (putative ccg part in Figure (4.3)), while ARK applies to any ODE system as mentioned before.

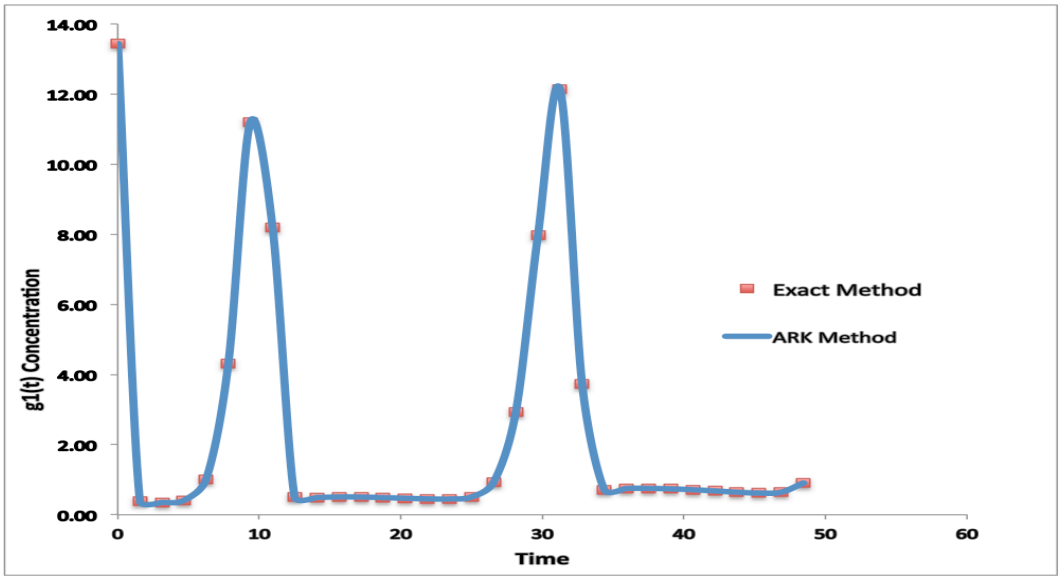

Figure 4.6: The solution of g1(t) using the numerical exact and the ARK methods using NG =32 Gauss-Legendre quadrature points over the time period [0,48] h for the genetic network of the biological clock of *N. crassa* agree. NG=6 between the red dots and the max absolute error is 10^{-4}

4.2 MATERIALS AND METHODS

All microarray data used for analysis in this paper came from Accession 13 entitled "cycle 1" in the public database FFGED[68], and the description of the data collected are described in Dong et al. [9]. In the cycle 1 dataset there are 2,436 features. Genes that are QA-responsive were removed so as not confound results using later datasets

involving a QA-responsive promoter, leaving 2,418 genes for analysis. The "cycle 1" data are also attached in the supplement.

4.2.1 OBTAINING THE REGULATOR [REG_K] PROTEIN CONCENTRATIONS

A) The protein concentration [Reg0(t)] ≡ [WCC(t)] can be obtained from the earlier biological clock ensemble simulations[4] on the simulation time interval [T0, T1] = [0, 48]h.

B) Due to lack of CCG protein data for the 5 regulatory ccg modules, k=1,…,5 their regulatory protein product concentrations, [Regk(t)] ≡ gp,k(t), are obtained as follows:

1) Obtain an estimate for rate coefficients L_c and D_{cp}, for the regulatory ccg-module from the clock parameter set of the earlier clock ensemble simulations [4]

2) Obtain, for a sufficiently dense t-grid on simulation time interval [T0, T1], the ensemble averages for the messenger RNA concentrations mk(t) ≡ gr,k(t), for the 5 regulatory ccg modules, k=1,…,5. Here, S(t) is replaced by [WCC(t)] in the respective Eqs. (1,2) for each of the respective modules, k=1,…,5, under the regulatory model assumption that the regulatory ccgs are activated by WCC only (see Figure (4.1)). Also, [WCC(t)] is again taken from the earlier ensemble simulation results for the biological clock[4] .

3) Given as inputs the rate coefficient estimates, L_c and D_{cp}, and the mRNA concentration, mk(t) ≡ gr,k(t), for each regulatory ccg module k=1…5, we can solve Eq. (4) for the module's protein product [Regk(t)] ≡ gp,k(t). That is, written in terms of mk(t) and [Regk(t)], Eq. (4) becomes

$$\frac{d[Reg_k]}{dt} = L_c m_k(t) - D_{cp}\,[Reg_k(t)];$$

$$k = 1, \ldots, 5$$

which is solved for [Regk(t)] on simulation interval [T0, T1] = [0, 48] h with initial condition rk by

$$[\mathrm{Reg_k(t)}] = \mathrm{r_k e^{-D_{cp}t}} + \int_{t=T_0}^{t} \mathrm{e^{-D_{cp}(t-t')}}\ \mathrm{L_c m_k(t')dt'};$$

k=1,…,5

To choose a value for the initial condition rk which is comparable in magnitude to typical values of [Regk(t)] over the observation time interval [T0, T1], we require that rk matches the time average of [Regk(t)]:

$\mathrm{r_k} = \frac{1}{T_1 - T_0} \int_{t=T_0}^{t=T_1} [\mathrm{Reg_k(t)}]\ \mathrm{dt}$; k=1,…,5

Inserting the foregoing ODE solution equation for $[\mathrm{Reg_k(t)}]$ into the latter equation for $\mathrm{r_k}$, we obtain a simple linear algebraic equation for $\mathrm{r_k}$ which is easily solved, given the inputs $\mathrm{L_c}$, $\mathrm{D_{cp}}$ and $\mathrm{m_k(t)}$ on time interval $[T_0, T_1]$, These time-averaged, and also initial concentration values serve as the $\mathrm{r_k}$ - input values in Eqs. (5) and (10-12).

4.2.2 ENSEMBLE METHOD FOR DISCOVERING A GENOMIC-SCALE NETWORK OF UNKNOWN TOPOLOGY:

There are two phases in the ensemble method: in the equilibration phase parameters values are found that allow the ODEs solution to fit to the experimental data, and in the accumulation phase many models or equivalently, sets of parameters, that represent the experimental data well are accumulated. Averaging over these models captured in the

accumulation phase allows prediction about how the described genetic network will behave. Thus, averaging many solutions of the ODEs (obtained by a MCMC method utilizing the Metropolis algorithm) with different initial conditions and parameters values allows an assessment of fit to the experimental data.

For best performance, unlike the equilibration phase which runs on a single GPU, the accumulation phase can be run in two GPUs, one for solving ODE solutions that are used in the χ^2-calculation for the Metropolis Monte Carlo update; and the other for calculating the results for the "MC Scores", *i.e.*, the ODE model solutions actually included in the ensemble MC sample to average and compare to the experimental data. The former ODE solutions (for χ^2-calculation) need to be calculated only for the 13 time-grid points at which observations were made; the latter (for inclusion in the ensemble MC sample) are required, less frequently, but on a much denser grid of time points. Moreover, overlapping the CPU and GPU jobs was considered and implemented in our simulation code whenever possible. A detailed description of the ensemble MC method on GPUs for the supernet is given right below.

The following steps describe the algorithm for the ensemble method[4, 50] for fitting and discovering the 2,418 genes regulatory network.

A) Metropolis Monte Carlo updating algorithm for θ- and μ-variables, in model Eqs. (1-5) where all six regulators, k=0,1,2,3,4,5, are activators:

   I. Proposal step (Monte Carlo): A Mersenne Twister algorithm[69] residing on the CPU is employed to draw all required random numbers $u \in [0,1]$ from a uniform distribution. In the proposal step we first decide randomly for each of the 2,418 *ccg* slave modules, with 50% probability, to either

(a) update one of the module's 10 so-called θ-variables[4] [initial conditions and rate coefficient values in Eqs. (1-5) or in Eqs. (3,4,10,11)] or (b) to update a pair of the module's 6 weights, μ_k [*CCG*-Regulators binding strengths]

a) The new θ-vector, θ', for each *ccg* module is generated from the old $\theta \equiv [\theta_1, \ldots, \theta_{10}]$ according to $\theta'_j = \theta'_j + SW_j\,(2u-1)$ where $u \in [0,1]$ is a uniformly distributed random number drawn for each module; the updated θ-component number, j, is likewise drawn with uniform probability from $j \in \{1,2, \ldots,10\}$; and SW_j is a step width variable assigned to every initial condition and rate coefficient MC variable, θ_j. Each SW_j is to be continually adjusted for optimal MC equilibration.

b) The proposed new regulator weights, μ_k', are from old μ_k in a total-weight-preserving pairwise updating step, according to the following procedure, for each *ccg* module:

1- A pair of weight indices, i and j, each from {0,1,…,5}, is selected with uniform probability, with $i \neq j$.

2- The new weights are calculated as $\mu'_i = \mu_i - \Delta\mu$ and $\mu'_j = \mu_j + \Delta\mu$; whereas $\mu'_k = \mu_k$ for all other k, with $k \neq i$ and $k \neq j$, and the random change Δμ is generated as follows:

3- A step-width factor $g_{ji} = r\, f_{ji}$ is calculated where r is a uniform random number, $r \in [0,1]$; and f_{ij} is the max step-width value, assigned to each weight-preserving μ-updating pair and continually adjusted for optimal MC equilibration. Each f_{ij} must be chosen to obey $0<f_{ij}<1$ and $f_{ij} = f_{ji}$;

4- Then set $\Delta\mu = g_{ji}\min(\mu_i, 1-\mu_j)$ which ensures that $\Delta\mu \leq g_{ji}\mu_i \leq \mu_i$ and $\Delta\mu \leq g_{ji}(1-\mu_j) \leq 1-\mu_j$. This then also ensures that the proposed new weights obey $\sum_{k=0}^{5} \mu_k{}' = 1$ and $0 \leq \mu_k{}' \leq 1$;

5- For the 5 regulatory *ccg* modules set the weights for their own regulation to $\mu_0=1$ and $\mu_{1,2,3,4,5}=0$ for the $Reg_{1,2,3,4,5}$ based on the model assumption that all 5 regulators are active regulatory *ccg*s are regulated only by $Reg_0 \equiv$ WCC

II. The CPU sends the 10 θ's and the 6 μ's for each *ccg* module to the GPU. An ARK ODE solver resides on the kernel function (executed on the GPU) and solves the 2,418 systems of ODEs of the slave *ccg* modules in parallel and sends the solution back to the CPU.

III. Accept/Reject step (Metropolis): The ODE system, Eqs. (1-5) or (10-12) for each non-regulatory *ccg* depends only on the six regulator proteins, but not on any other non-regulatory *ccg* modules. Furthermore, the regulator proteins are assumed to be independent of all non-regulatory *ccg* modules. As a result, the θ- and μ-variables of each non-regulatory *ccg* are also statistically independent of the θ- and μ-variables of all other

non-regulatory *ccg* modules in joint the ensemble likelihood function[4] for the whole system of all *ccg* modules. In other words, the whole-system ensemble likelihood function[4], Q, factorizes into independent single-*ccg* likelihood functions, $Q^{(n)}$, for each non-regulatory *ccg* module *n*. For that reason we can then perform the accept/reject steps of the θ- and μ-variables separately and independently for each non-regulatory *ccg* module. We thus apply the standard Metropolis criterion separately and independently to each non-regulatory *ccg*, each with the Metropolis acceptance probability $P_{acc}^{(n)}=\min[1,R^{(n)}]$, as defined below. By drawing a uniform random number $r^{(n)} \in [0, 1]$, the proposed change of the θ- or μ- variables, for a given *ccg n,* is accepted when $r^{(n)}<P_{acc}^{(n)}$, else rejected. Here, the probability ratio $R^{(n)}$ is given in terms of the single-*ccg* ensemble likelihood function $Q^{(n)}(\theta, \psi, \mu) = (1/\Omega^{(n)}) \exp[-H^{(n)}(\theta, \psi, \mu)]$ by [4]

- $R^{(n)}=\frac{Q^{(n)}(\theta',\psi,\mu')}{Q^{(n)}(\theta,\psi,\mu)} = \frac{\exp(-H^{(n)}(\theta',\psi,\mu'))}{\exp(-H^{(n)}(\theta,\psi,\mu))}=e^{-\Delta H^{(n)}}$

- $\Delta H^{(n)} = H^{(n)}(\theta',\psi,\mu') - H^{(n)}(\theta,\psi,\mu) = \frac{1}{2}[\chi_n^2(\theta',\psi,\mu') - \chi_n^2(\theta,\psi,\mu)]$

- $\chi_n^2(\theta,\psi,\mu) = \sum_{m=1}^{M}\left(\frac{\ln(y_{n,m}^{exp.}) - \ln(F_m(\theta,\mu)) - \psi_{c(n,m)}}{\left(\sigma_{n,m}^{exp.}/y_{n,m}^{exp.}\right)}\right)^2$

Here n=1,2,…,N and N=2418-5=2413 is the total number of non-regulatory *ccg* modules. M=13 is number of observation time points, t_m, at which experimental *ccg* mRNA concentration data, $y_{n,m}^{exp.}$ have been taken for *ccg*-module *n*. These experimental mRNA data are compared in the χ^2- function to the corresponding ODE model solutions for the mRNA concentration, $F_m(\theta,\mu) \equiv g_r(t_m;\theta,\mu)$, obtained at time t_m for *ccg* module *n,* given the module's ensemble MC variables (θ , μ). We use log-concentration difference residuals in the χ^2-function because it is "scale-factor free", *i.e.*, it assigns the weight to each data point in χ^2 independent of scale factor fluctuations. Data points (*n,m*) are assigned to scale factor classes, *c*, such that two data points belong to the same class if they both share the same unit conversion factor from experimental concentration units (fluorescent photon counts) to model concentration units; and *c(n,m)* denotes the class to which data point (*n,m*) has been assigned [4]. The log of the unknown unit conversion factor, ψ_c, for each class *c* is treated as an ensemble MC variable[4], on the same footing as θ and μ. The updates of the ψ-variables are performed separately and with a different procedure than θ- and μ- updates, as described below. In our experimental data sets, each *ccg* slave module actually has the same log unit conversion ψ_c , i.e., there is only one scale factor class. However, our approach applies generally to data sets requiring multiple scale factor classes.

B) Müller-Box updates of unit conversion factor variables ψ .

At fixed θ and μ for all *ccg* modules, the whole-system ensemble likelihood function, $Q=\prod_{n=1}^{N} Q^{(n)}$, results in an independent 1D Gaussian distribution for each log unit conversion factor variable ψ_c. We therefore update each ψ_c by drawing it from the appropriate 1D Gaussian distribution, without Metropolis, using the Müller-Box algorithm[70]. Since all θ and μ are fixed, the draw from this Gaussian distribution does not require a new solution of the model ODE system. Also, using Müller-Box, we achieve100% acceptance for each ψ_c-update. The ψ_c-updates are therefore very fast and are performed after each single-θ and each pairwise μ-variable update.

C) Metropolis Monte Carlo updating algorithm for θ- and μ-variables, in model Eqs.(10-12) where only five regulators, k=0,1,2,3,5, are activators and regulator k=4 (CSP-1) is a suppressor:

For this model, all six regular binding strengths, μk , are still constrained by positivity $\mu k \geq 0$ for k=0,1,…,5. However, only the weights of the five activators, k=0,1,2,3,5, but not k=4, are subject to the normalization condition, $\sum_{k=0,k\neq 4}^{5} \mu k=1$, while the weight of the suppressor (CSP-1) $\mu 4 \geq 0$ has no upper bound. (To avoid numerical problems in the ODE solution, in the MC simulations a very large upper bound, $\mu 4 < 100$, was actually imposed.) This modification on the weight constraints will allow both the suppressor and one of the five activators to fully bind to the ccg at the same time and thus compete in the ccg's regulation.

In this case, only the five activator weights, μ0, μ1, μ2, μ3, μ5, are updated by the pairwise MC updating procedure b) described above, while weight μ4 is treated like a θ-variable using the single-variable Metropolis MC updating procedure a). The single-variable Metropolis updating procedure is of course also used again for the θ-variables of each ccg module.

4.2.3 AN ENSEMBLE METHOD USING THE ADAPTIVE RUNGE KUTTA ON THE GPU:

The ARK algorithm can be used for solving system of linear and non-linear ODEs. In this paper, the ARK method is implemented on the GPU by assigning a thread to solve for a single system of ODEs out of 2,418 systems, so in essence, we have 2,418 threads solving the same system of ODEs, but each has a different set of parameters. We used the registers on the GPU as they are the fastest memory holder to define any variable that is required by the ARK method. On the other hand all of constant data, for instance, the interpolation files, which are six files one for each regulator, and all of the ARK's constants to be defined on the constant memory of the GPU. This code organization shows the best performance among other code organizations that were tested. Algorithms were coded in CUDA/C++ and are attached as a supplement and in sourceforge.net under the keyword vtens_ARK_clock1.

4.2.4 AN ENSEMBLE METHOD USING THE NUMERICAL EXACT SOLUTION ON THE GPU:

The system of ODEs described above can be solved using a numerical exact integral solution formula for solving the first order linear ODEs described above on the GPU using CUDA/C++ (code attached) and in sourceforge.net under the keyword vtens_EI_clock1.

1. The general first order linear ODE

$$\frac{dy(t)}{dt} + p(t)y(t) = q(t) \tag{6}$$

is solved subject to initial condition $y(t) = y_0$ at time $t = t_0$ by:

$$y(t) = y_0 e^{-J(t)} + e^{-J(t)} \int_{t_0}^{t} e^{J(t')}\, q(t')\, dt' \tag{7}$$

; where $J(t) = \int_{t_0}^{t} p(t')\, dt'$.

To avoid numerical overflows of the exponential function in the integrand, $e^{J(t')}$, we rewrite and numerically implement Eq. (7) as follows:

$$y(t) = y_0 e^{-J(t)} + \int_{t_0}^{t} e^{J(t')-J(t)}\, q(t')\, dt' \tag{8}$$

2. Use the foregoing formula to solve the ODE for $g_1(t)$, with $g_0(t)$ replaced by $g_0(t)= g_{tot}(t) - g_1(t)$; where quantity

   $g_{tot}(t)$ is the total gene concentration which is constant in time t, and given by: $g_{tot}(t) = g_0(t_0) + g_1(t_0)$; where $g_0(t_0)$ and $g_1(t_0)$; are the initial conditions for $g_0(t)$ and $g_1(t)$, imposed at time t_0=0

3. Use the solution for $g_1(t)$ (tabulated and interpolated) and the same formula, Eq. (7), to solve the ODE for the RNA concentration, $g_r(t)$

4. Use the solution for $g_r(t)$ (tabulated and interpolated) same formula, Eq.(7), to solve the ODE for the protein concentration, $g_p(t)$.
5. The Gauss-Legendre (GL) quadrature method[71] approximates the integral for a function $f(t)$ with t ∈[a, b] in terms of GL weights ($w_k^{(a,b)}$), GL roots ($t_k^{(a,b)}$), for a given number of integration points, N_G , with k∈{1,2,…, N_G} as follows:

$$I_G = \int_a^b f(t) \approx \sum_{k=1}^{N_G} w_k^{(a,b)} f(t_k^{(a,b)}) \quad (9)$$

In this paper, we used N_G =32 points to compute each $J(t)$ for the *t*-grid used in Eq.(8) and N_G =6 points in between each of these 32 t-grid points to compute $J(t')$ for the *t'*-grid used in Eq. (8). The algorithm used to compute the GL weights and roots is previously described[72].

4.2.5 REGULATORY NETWORK WITH CSP-1 (NCU00045) AS A REPRESSOR:

Fitting of the five distinct models in Figure (4.11) by MCMC using the ensemble method with the numerical exact solution was replicated twenty times for a total of 100 MCMC simulations. As can be seen in Figure (4.11) the χ^2 distributions are approximately normal, which implies that the mean of each of the χ^2 distributions (χ^2_{ave}) is going to be normal. We performed a one-way analysis of variance on the average χ^2_{ave} with the data in Table (4.1).

A one-way analysis of variance was performed on the average chi-squared values in Table (4.1)[73]. The analysis of variance is reported in Table (4.2). There is a significant difference at less than 0.0001 level in the average chi-squared values in Table (4.1). We then used the highly conservative Scheffe multiple comparison test to conclude

that only the ensembles with all activators and m=1 was significantly worse than the ensemble with a repressor (m=4) and 4 activators at the 0.05 level[73]. The remaining four model ensembles (other than the ensemble with all activators and m = 1) could not be distinguished from the latter model ensemble with a repressor (m =1) and 4 activators (m=4). There is too much overlap in the distributions of the chi-squared statistics across the respective ensembles to distinguish them as reported in the body of this work and Figure (4.11).

Table 4.1. Average χ^2ave in 20 replicates of each of 5 model ensembles with CSP-1 as activator or repressor and varying Hill coefficients. Models tend to perform progressively worse from left to right. In the second and third columns the CSP-1 protein is hypothesized to be a repressor with Hill coefficient for the repressor being 2 or 4 while the remaining regulators are hypothesized to be activators. In succeeding columns all regulators are hypothesized to be activators with varying Hill coefficients.

Model (Replicate)	Repressor(m=4) and 4 activators(m=4)	Repressor(m=2) and 4 activators(m=4)	All activators (m = 4)	All activators (m = 2)	All activators (m = 1)
1	65969	66134	65513	66141	68527
2	64729	64525	64317	64897	67155
3	65437	65779	66185	66883	69246
4	63095	63260	63694	64190	66787
5	63273	63457	63706	64251	66702
6	66287	65673	65774	66356	68758
7	65168	65330	65946	66515	69026
8	67354	66783	65820	66426	68857
9	65032	65263	66122	66640	68944
10	63606	63875	64065	64974	67863
11	65025	65222	66124	66631	69180
12	65078	65235	66085	66775	69109
13	62969	63317	63438	64526	66478
14	64868	65042	66077	66481	68803
15	63541	63610	66077	64792	67042
16	63832	64065	64849	65101	67635
17	63146	63323	63473	64114	66530
18	75717	75282	68817	69517	72172
19	66710	71261	69293	69351	72234
20	66879	66713	67618	68453	71100
Average	**65386**	**65657**	**65650**	**66151**	**68607**

Table 4.2 One-way analysis of variance on average χ^2_{ave} in Table (4.1) across 5 model ensembles of regulation of a circadian network. The 5 model ensembles are listed in Table (3.1).

Source of Variation	Degrees of Freedom	Sum Of Squares	Mean Squares	F
Between Models	4	140351655.14	35087913.79	7.22
Between replicates within Models	95	461653940.25	4859515.16	
Corrected Total	99	602005595.39		

Table 4.3 Primer pairs for target genes (NCU00685, NCU09843, NCU00476, NCU04166, and NCU08903) and endogenous controls (NCU05995 and rDNA).

Locus	Primer ID	Strand	Primer Sequence
NCU00685	685.PB.3i-F	Forward	5'- ACG ACG TTG AGC TGC ATT T -3'
	685.PB.3i-R	Reverse	5'- TTT GTA AAC GGT CGT CGC AG -3'
NCU09843	9843-3F	Forward	5'- AGA TGG CGA TTA TCA CGA ATG G -3'
	9843-3R	Reverse	5'- TTC CAT TCC CTT TCC CTT CC -3'
NCU00476	476-3F	Forward	5'- CTA CAA AGT CCC TAC CCA TCT G -3'
	476-3R	Reverse	5'- GTA ATC TCA TCC TCG CCC TG -3'
NCU04166	4166-3F	Forward	5'- CCT GCG AGT CGA TGA GTT G -3'
	4166-R	Reverse	5'- CAA TGA GAG CGT TGA TGG TG -3'
NCU08903	8903-3F	Forward	5'- GTC ACC GCA TCA CTC TCC -3'
	8903-3R	Reverse	5'- ACA AAA GAC GGG TGG CAG -3'
NCU05995 (polyubiquitin)	ub-1F	Forward	5'- CCG TGG CGG CCA GTA A -3'
	ub-1R	Reverse	5'- TCT GAT TCT TGA TGA CGA GCA AG -3'
(rDNA)	rD-F	Forward	5'-TCA AGC CGA TGG AAG TTT GAG-3'
	rD-R	Reverse	5'-TGC GGC CCA GAA CAT CTA A-3'

4.2.6 REAL-TIME QUANTITATIVE POLYMERASE CHAIN REACTION (RT-QPCR) OF REGULATOR TARGETS FOR CIRCADIAN RHYTHM TABLE (4.5):

Knockout strains were obtained for NCU01640 and NCU06108[74] and crossed to a *bd* strain (Fungal Genetics Stock Center Strain 1858 or 1859, respectively) to generate the double mutants, *bd*, $NCU01640^{KO}$ and *bd*, $NCU06108^{KO}$. Each of these double mutants were assayed for circadian rhythms in the 5 target genes in Table (4.5), using ubiquitin and rDNA as controls over a 48 h window. Replicate cultures of each strain were grown in liquid culture and synchronized by an average of 26 h of light (7- micromoles per Liter per second per meter squared) and then transferred to the dark to assay circadian rhythms

of target genes, a cycle 1 experiment[5]. In these experiments the total growth time (50 h) of 13 replicate cultures was kept constant, and one flask was harvested for cells every four hours over the 48 h observation period in the dark (D/D).

Total RNA was harvested from the 13 cultures of each strain, each at a different time point using a Spectrum Plant Total RNA kit (Sigma-Aldrich, St. Louis, MO, USA, Inc.). All RNA samples were then treated with DNase (# EN0525, Thermo Scientific, Pittsburgh PA, USA) according to manufacturer's protocol (but omitting the supplied buffer). The quality of the 26 RNA samples was assessed using an Agilent Technologies RNA 6000 Nano LabChip (#5067-1511, Agilent Technologies, Inc., Santa Clara, CA, USA) on an Agilent Technologies, Inc. 2100 Bioanalyzer , yielding RNA Integrity Numbers (RIN) between 6.0-6.3. cDNA synthesis was carried out with a Superscript III Ist Strand cDNA Synthesis Kit (Invitrogen, Inc., Grand Island, NY USA 18080-051). RT-qPCR was carried out on an ABI-Prism 7500 with a Brilliant III Ultra-fast SYBR Green qPCR Master Mix(#600882, Agilent Technologies, Inc.) or Brilliant II SYBR Green qPCR Master Mix(#600828, Agilent Technologies, Inc.).

For each target gene in Table (4.5) three primer pairs were tried and one selected based on amplification plots and disassociation curves Table (4.3). Two genes were considered for endogenous controls, 18S rDNA and ubiquitin. Both endogenous controls had expression levels that were aperiodic, but ubiquitin had RNA levels more comparable to those being measured in the targets and was selected as the endogenous control. The reference time point was 48 h for relative change in expression. All selected primers had

a single amplification product from the disassociation curves, but the peaks for the NCU04166 and ubiquitin primers in the disassociation curve were broader than the rest. The five primers selected for the target genes and endogenous controls were validated by 5, 4-fold dilution series with correlations of at least 0.97[75]. All reactions were compared to a control reaction only missing reverse transcriptase. These -RT controls differed by ~2-5 cycles from the start of amplification from those with reverse transcriptase. No-template controls tended to amplify after cycle 34, if at all, suggesting minimal random contamination during qPCR plate setup. Relative gene expression (RQ) was quantified by the $\Delta\Delta C_T$ method, and the average threshold cycle for each time point was normalized to ubiquitin at 48h for each strain. Periods of RQ series were estimated by fitting a sinusoid by the method of maximum likelihood as described previously[76]. The parameters, namely y-intercept, amplitude, period, and phase, were computed by maximum likelihood scoring initialized by a grid search.

4.3 RESULTS AND DISCUSSION

Comparison to Profiling Experiments: The ARK solver on GPUs to implement MCMC methods was sufficient to describe and explain published profiling data on 2,418 putative *ccgs* as it is shown in Figure (4.7) and Figure (4.8). Five model ensembles were identified by varying the Hill coefficients of the transcriptional regulator in Eq. (5), with m=4, 2, or 1 and by varying the NCU00045 encoded regulator from an activator to a repressor as shown below in Eqs. (10-12). The ensembles of the described genetic networks predicted the mRNA levels for most of the genes, and as is shown Figure (4.7) and Figure (4.8), the overall fit of the RNA profiling data on 2,418 genes was consistent

with ensemble predictions. The average contribution of a data point to the χ^2 was 2.05 (and 2.02 with CSP-1 as a repressor, which was a little larger than the working model (1.54)[5].

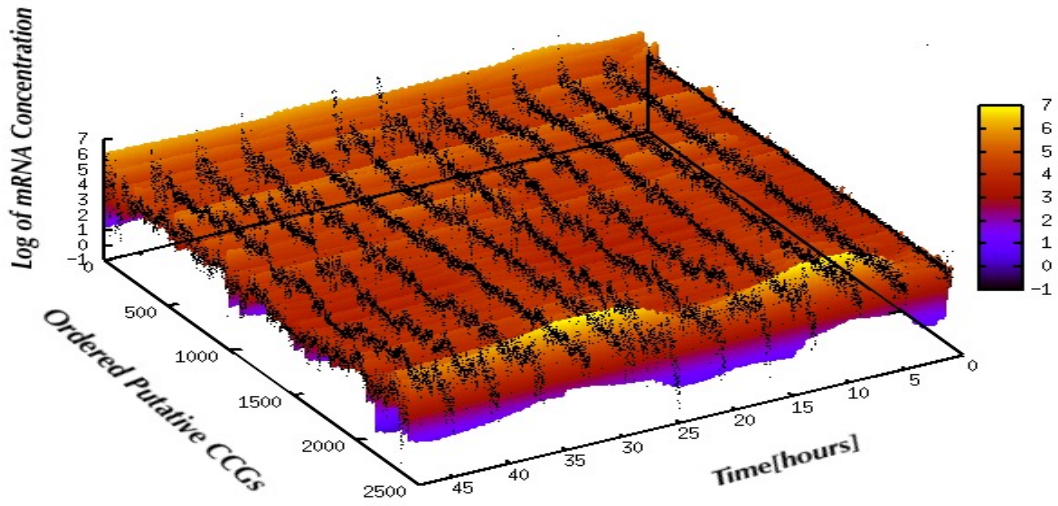

Figure 4.7: The network of 2,418 putative clock-controlled genes fits to experimental data using the ensemble method very well. The predictions (orange and purple) and the observation data (black dots) are shown in 3 dimensions. The modules in Figure (4.3) are ordered based on the similarity of their profiles.

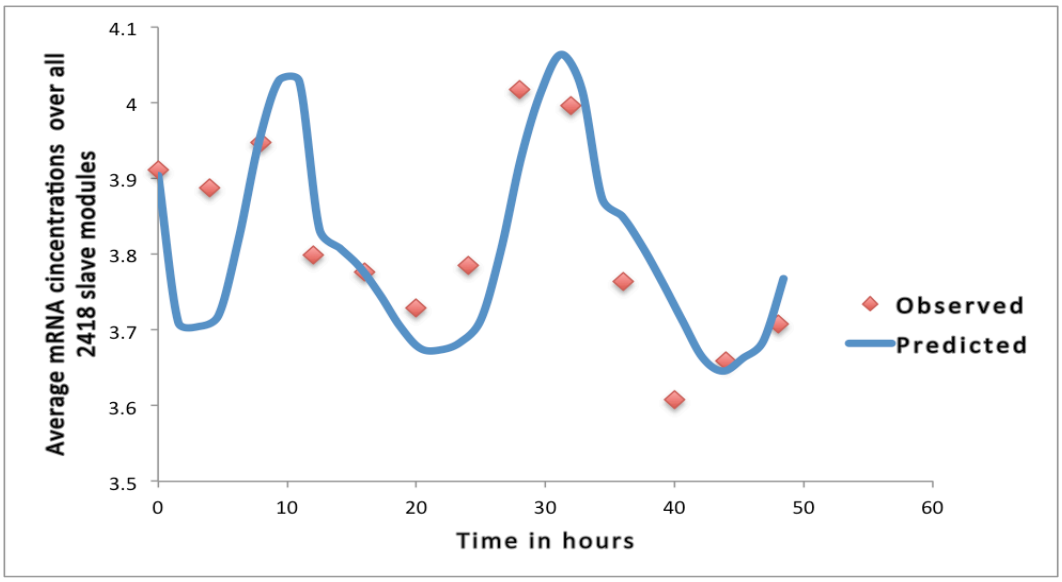

Figure 4.8: An ensemble of genetic networks predicts the mRNA levels overall of 2,418 putative clock-controlled genes (model used here where m=4 and all of regulators are

activators). The predictions fit the experimental data within a standard error. The observed data computed using $=\ln(y_{k,m}^{exp.})-<\psi_{ck}>$ vs. time and the predictions computed using $<\ln\left(F_{k,m}(t,\theta,\mu)\right)>$vs. time. The figures predictions for the other models are very close to this figure.

4.3.1 DISCOVERING THE REGULATORY NETWORK OF THE PUTATITVE CLOCK-CONTROLLED GENES.

A variety of biological functions for the putative *ccgs* were identified previously[5]. By fitting the profiling data to the supernet in Eq. (5) and given μ_k (the regulator binding strengths), k=0,…,5 we have been able to assign target genes and their functions to the corresponding regulators, WCC, ADV-1 (NCU07392), RPN-4 (NCU01640) in transcriptional control, product of NCU06108 in transcriptional control, repressor CSP-1 (NCU00045)[77], and product of NCU07155 (in regulation of nitrogen and sulfur metabolism). Each regulator has a corresponding binding strength (μ_k) to a target gene. For example, the binding strengths, μ_0 and μ_5, correspond to WCC and NCU07155, respectively. Having the ensemble average of binding strengths (μ's) for the 2,418 targets, the highest μ average for each gene indicated the candidate-binding regulator for that gene or slave module. In Figure (4.9), the number of genes assigned to each regulator is displayed.

As shown in the video, the supernet reconstruction of the network via MCMC converges quite quickly to the assignment of putative *ccgs* to regulators, i.e. to the estimates of the binding strengths. In the video the dynamic assignment of putative *ccgs* to regulators is shown during the equilibration stage of MCMC over the first 250 sweeps (a sweep being a visit on average to each parameter in the model during MCMC). The assignment(s) of 2,418 – 6 putative *ccgs* to 6 regulators is made in less than 250 sweeps.

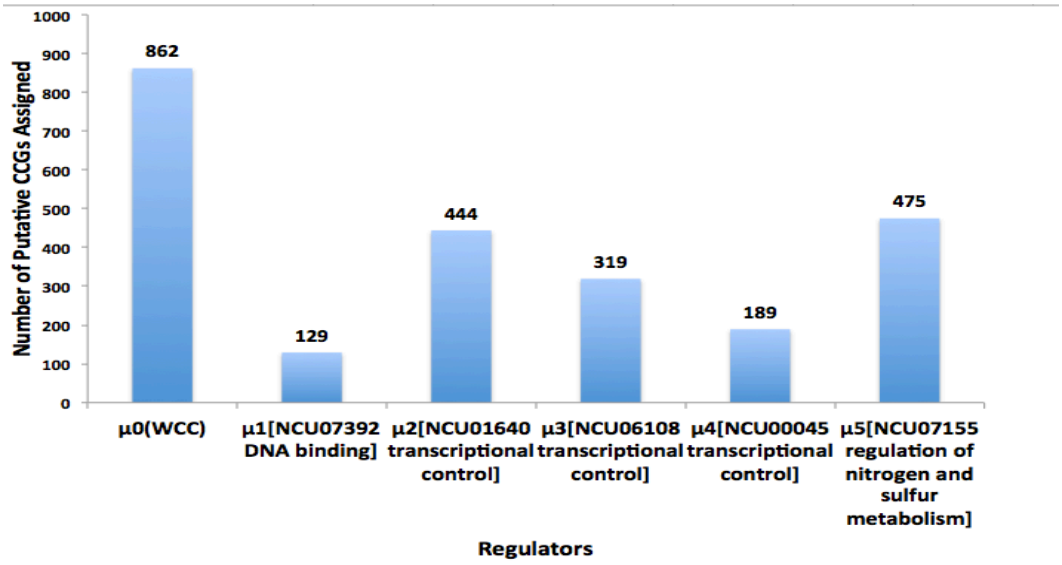

Figure 4.9: Putative ccgs are assigned to each of the six regulators (WCC, NCU07392, NCU01640, NCU06108, NCU00045, NCU07155). The highest µ average over the 40,000 accumulation sweeps of the 2,418 genes indicates the candidate-binding regulator for that gene.

We tested whether or not the genetic network hierarchy could be simplified in Table (4.4) and asked whether or not the regulation could simply be by WCC or simplified by dropping one of the 5 other regulators. At least one activator (namely *rpn-4* (NCU01640)) needed to be retained, and the data strongly supported at least one other transcription factor under WCC control in the hierarchy in Figure (4.1) and Figure (4.2).

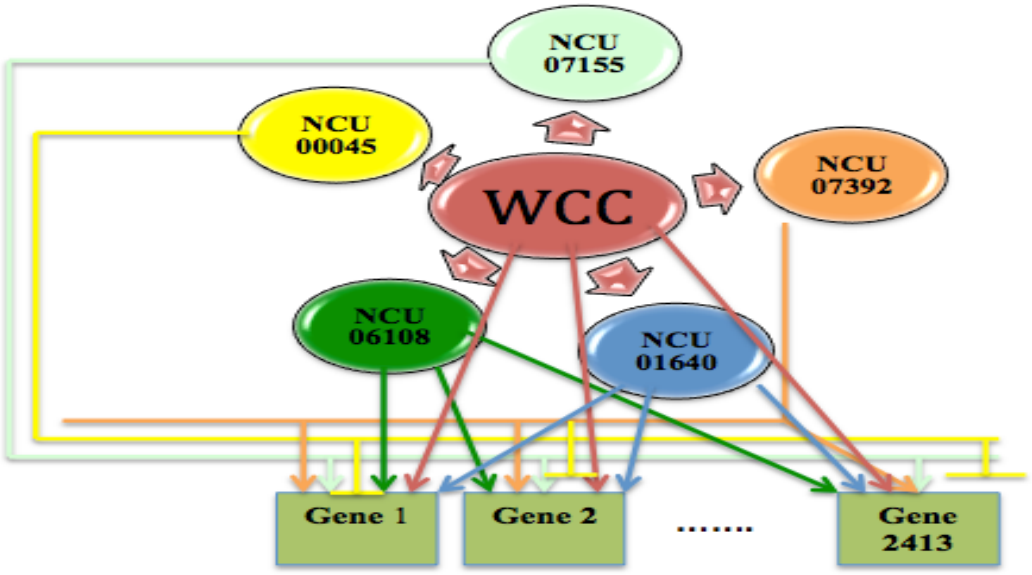

Figure 4.10: A genetic network consisting of 2,418 slave modules and a master module for the clock mechanism with repressor (NCU00045). There are 4 positive activators (NCU07392, NCU01640, NCU06108, NCU07155) and a repressor (NCU00045) under control of WCC, to be identified by the ensemble simulation[4, 66] .

4.3.2 REGULATORY NETWORK WITH CSP-1(NCU00045) AS A REPRESSOR.

Sancar et al.[77] suggested that CSP-1 may be a repressor as a opposed to an activator in Figure (4.10). We tested this hypothesis. Involving CSP-1 as a repressor instead of an activator implies changing Eqs. (1,2 and 5) respectively to the following, with k=4 being the CSP-1 protein:

$$\frac{dg_0}{dt} = B_c g_1 - A_c g_0 S(t) +$$

$$A_c g_1(\mathrm{t})\ \mu_4 \left[\frac{r_0}{r_4}\right]^m [\mathrm{Reg}_4(\mathrm{t})]^m \qquad (10)$$

$$\frac{dg_1}{dt} = A_c g_0 S(t) -$$

$$B_c g_1 - A_c g_1(t)\ \mu_4 \left[\frac{r_0}{r_4}\right]^m [\mathrm{Reg}_4(\mathrm{t})]^m \qquad (11)$$

$$\mathrm{S(t)} = \mu_0[\mathrm{Reg}_0(\mathrm{t})]^{\mathrm{m}} + r_0^m \sum_{k=1,k\neq 4}^{5} \left(\left[\frac{1}{r_k}\right]^m \mu_{\mathrm{k}}[\mathrm{Reg}_{\mathrm{k}}(\mathrm{t})]^{\mathrm{m}}\right) \qquad (12);$$

For this model, the weights μ_k are constrained only by $\mu_k \geq 0$ and $\sum_{k=0,k\neq 4}^{5} \mu_k=1$, while $\mu_4 \geq 0$ has no upper bound, as described in Materials and Methods. Figure (4.10) shows that the supernet in Figure (4.1) can be changed to involve the CSP-1 as a repressor instead of an activator while Figure (4.11) shows $\chi 2$ values for a model ensemble with different Hill coefficients value and with CSP-1 as an activator or repressor.

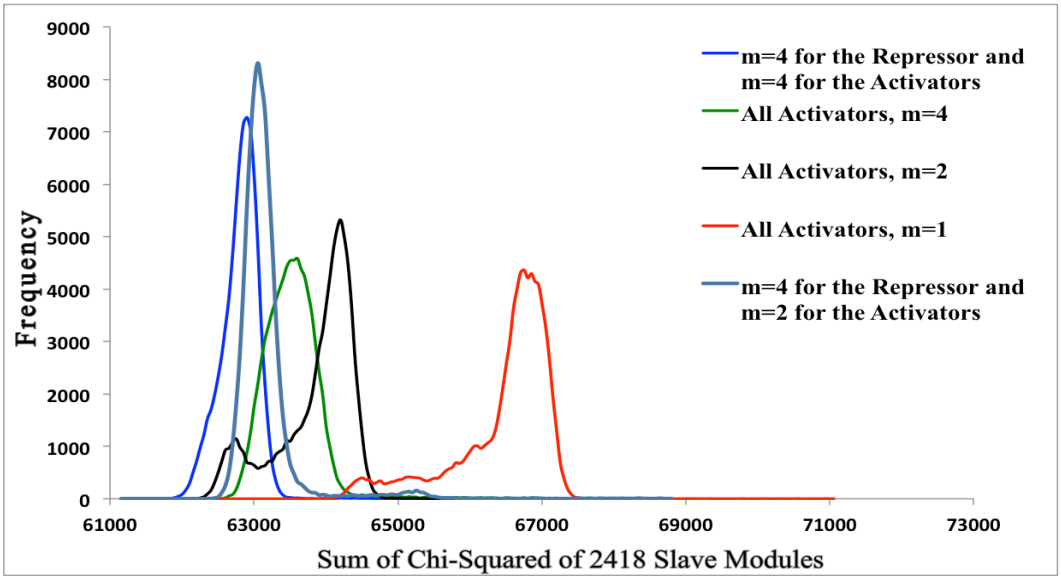

Figure 4.11: The best model ensemble (histogram of $\chi 2$ values most shifted to the left) has Hill coefficient m = 4 for the activators and m=4 for the repressor CSP-1. The histograms of $\chi 2$ values are computed for model ensembles with different Hill coefficients and with/without a repressor using Eqs.(5,10,11) (m=1,2,and 4).

From Figure (4.11), we observe that for the three model ensembles with activators only the ensemble without cooperativity (m = 1) has χ^2 values substantially larger (and worse) than those of the two ensembles with cooperativities, m=4 and 2. The χ^2 values of the model ensembles, with cooperativity m=4 and 2, overlapped each other, and the best fits were achieved with Hill coefficients of m = 4 as shown in Figure (4.7) and Figure (4.8). These results were consistent with earlier results[4] in that the model ensembles overlapped with cooperativities of 2 and 4 and in that the χ^2 values were lower with m=4.

However, treating CSP-1 (from NCU00045) as a repressor (see below) with Hill coefficient of 2 or 4 yielded lower χ^2-values but similar fit to the case with 5 activators each with Hill coefficient of 4 (WCC, NCU07392, NCU01640, NCU06108, NCU07155). Nonetheless, by replicating the ensemble fitting twenty times for each model, with a different random MC initial for ensemble model parameter variables at the start of each replica simulation (for a total of 100 MCMC runs), we find overlapping χ^2 distributions as in Figure (4.11) for all of the models except for the activator model ensemble with m=1 (red) in Figure (4.11) (See Materials and Methods). We conclude one model ensemble can be rejected, the models (in red) involving all activators and a Hill coefficient of m=1.

4.3.3 DISCOVERING A BROAD ARRAY OF FUNCTIONS FOR CLOCK-CONTROLLED GENES.

Putative *ccgs* found from the averages of the binding strengths (μ's) in the model ensemble and shown in Figure (4.9) were classified by their pathways and functions using KEGG Mapper software[78]. Figure (4.12) shows the number of annotated genes regulated by a particular regulator and participating in a particular pathway or function. In Figure (4.13) we show the binding strength of a particular regulator associated with a particular group of genes. There are distinct profiles for the regulators, and the binding strengths are quite high for at least one regulator to each target. An additional test was carried using the 862 genes that were inferred here to be regulated by WCC by asking how many genes were identified out of 292 genes[4, 5][4, 5][4, 5] [4, 5][6, 9] that are known to be regulated by WCC. The test showed that there were 153 *clock-controlled*

genes out of the 292 genes identified correctly while one sixth of the 292=~ 48 genes are expected to be regulated by each regulator including WCC if target genes were assigned at random to regulators.

Table 4.4. At least one regulator rpn-4 (NCU01640) cannot be excluded from the hierarchy in Figure (4.1) and Figure (4.2) without a highly significant loss of goodness of fit. The χ^2_{WCC} is for the model in Figure (4.3), in which WCC is the only regulator hypothesized. Average χ^2_{Model} is for a model in which the named transcription factor in column 1 is removed. The χ^2_{ALL} is for a model which there are 4 activators and 1 repressor, namely in Figure (4.10). All starred (*)χ^2 differences are significant at less than the 0.001 level.

Excluding Regulator	Average χ^2_{WCC}	Average χ^2_{Model}	Average χ^2_{ALL}	χ^2_{ALL} - χ^2_{Model}	df
NCU07392	68659.9	63984.7	63542.6	442.1	2418
NCU01640	68659.9	66621.3	63542.6	3078.7*	2418
NCU06108	68659.9	64209.5	63542.6	666.9	2418
NCU00045	68659.9	64469.0	63542.6	926.4	2418
NCU07155	68659.9	64571.5	63542.6	1028.9	2418

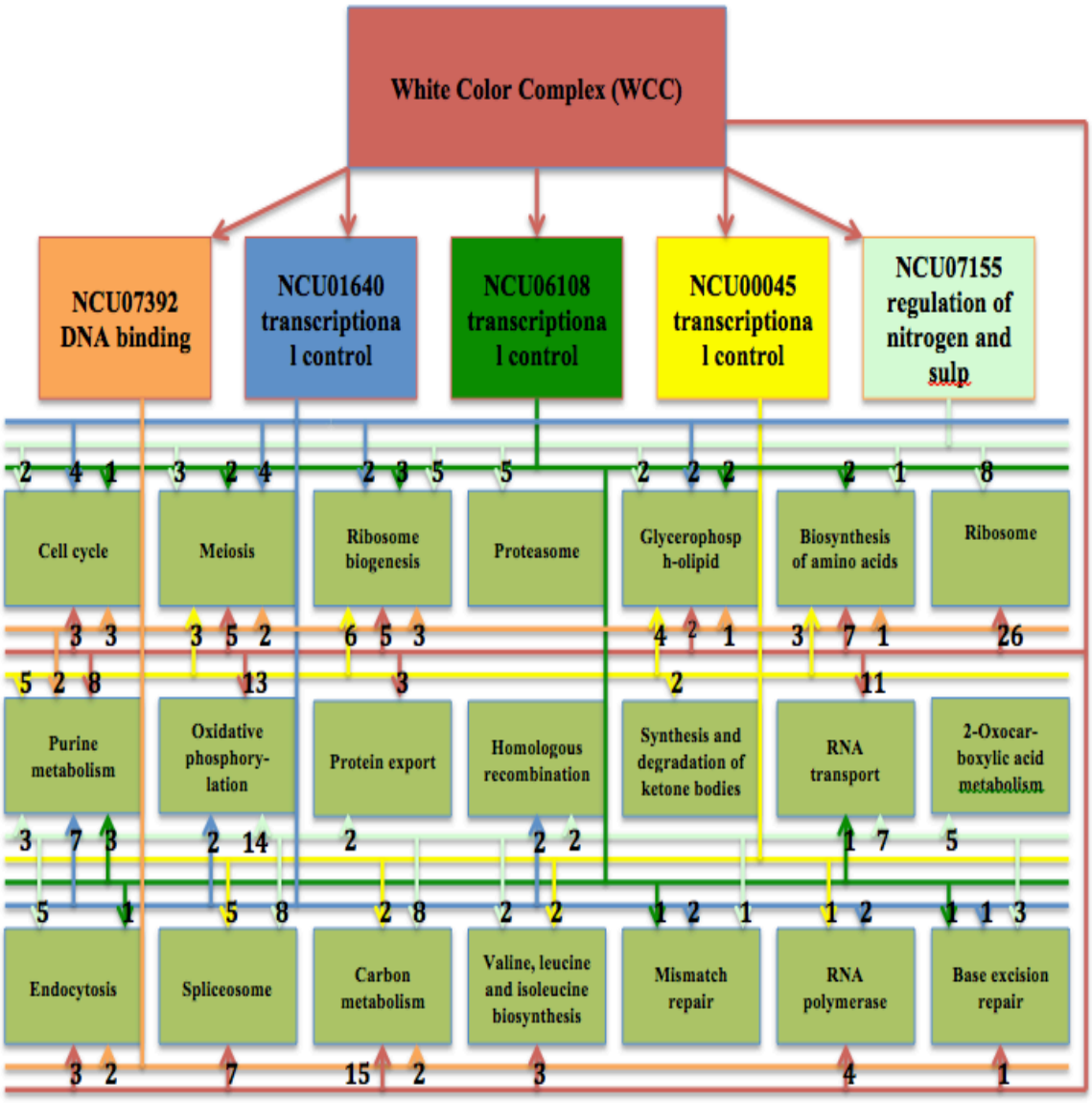

Figure 4.12: A regulatory genetic network for the six regulators (WCC, NCU07392, NCU01640, NCU06108, NCU00045, NCU07155) and the putative clock-controlled genes. The number on the arrow indicates how many annotated genes that are regulated by a particular regulator and participating in a particular pathway or function (small green boxes).

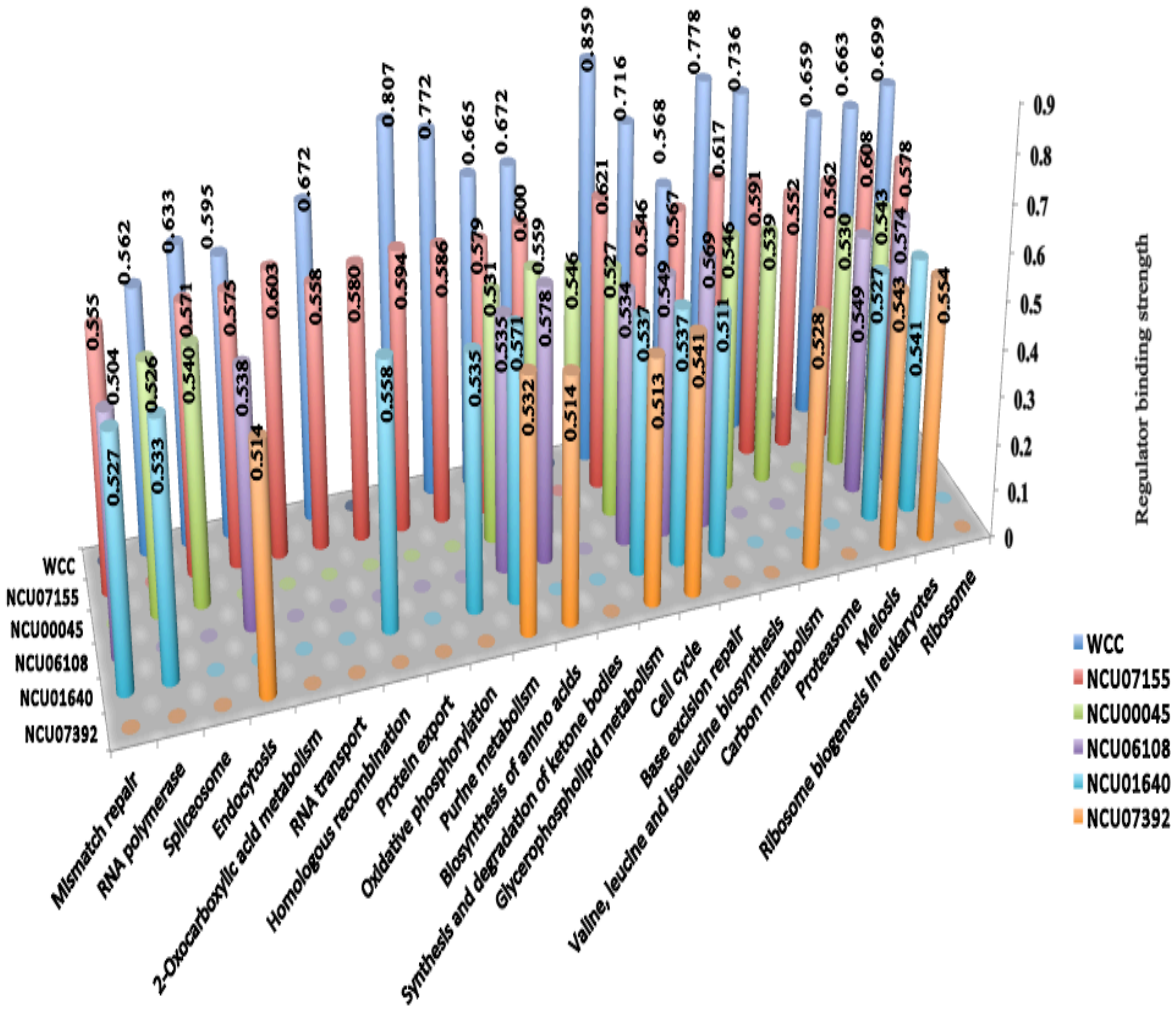

Figure 4.13: Regulator binding strength and target gene functions. The strength of regulator binding is computed by asking: what is the average of μ's across the 40,000 accumulation sweeps that is assigned to a group of genes that have the same function or pathway?

Table 4.5 :Averages of binding strengths (μ's) (over 40,000 sweeps) of each of 6 transcription factors to 5 targets in ribosome biogenesis in Eq. (5) with ± standard errors. Numbers in green highlight predicted targets of NUC01640 and NCU06108 cognate proteins.

	Target Genes				
Regulator	NCU00685	NCU09843	NCU00476	NCU04166	NCU08903
WCC	0.027±0.0028	0.061±0.0043	0.034±0.0029	0.032±0.0030	0.047±0.0036
NCU07392	0.105±0.0046	0.109±0.0054	0.100±0.0046	0.105±0.0049	0.098±0.0050
NCU01640	0.572±0.0077	0.509±0.0069	0.081±0.0042	0.076±0.0043	0.098±0.0040
NCU06108	0.110±0.0049	0.107±0.0053	0.57±0.0074	0.574±0.0080	0.574±0.0074
NCU00045	0.094±0.0054	0.119±0.0054	0.101±0.0047	0.105±0.0049	0.106±0.0049
NCU07155	0.091±0.0044	0.095±0.0047	0.110±0.0046	0.108±0.0049	0.112±0.0053

4.3.4 CRITICAL TEST OF GENETIC NETWORK PREDICTIONS USING RT-QPCR

As a critical test of the network in Figure (4.12), we focused on five annotated genes in ribosome biogenesis[5] whose binding strengths suggested regulation by the two of the six transcription factors, namely NCU01640 (*rpn-4*) or NCU06108. Genes NCU00685 (or *ck-1a*) and NCU09843 are predicted targets of RPN-4, and NCU00476, NCU04166, and NCU08903 are predicted targets of the cognate protein of NCU06108 (as seen from the highlighted binding strengths in green in Table (4.5)). The gene NCU00685 also appears to be a target of WCC. All circadian genes had periods between 16 and 30 h[5][9] before introducing a knockout. The prediction was that knockouts of *rpn-4* or NCU06108 should alter expression of all five genes, although the target gene NCU00685 should retain some rhythm. The binding strengths of the transcription factors to the five targets are summarized in Table (4.5), as estimated by the ensemble method.

The casein-kinase-1a (*ck-1a* or NCU00685) target gene is a *ccg*, directly regulated by WCC, as determined previously[5]. Hence, a knockout of NCU01640 should not prevent *ck-1a* from being circadian. Indeed, its circadian rhythm in Figure (4.14) is estimated here at 25 ± 2 h by RT-qPCR. In contrast, a gene free from circadian control should have

an abnormally long 'period' when tested for periodicity, making it indistinguishable from a non-oscillating system. For example, rDNA levels followed a 48 h period here (over a 48 h observation window) when ubiquitin was used as the internal control in RT-qPCR (see Materials and Methods). Accordingly, the periods of NCU04166 and NCU08903 shown in Figure (4.14) were 35 ± 4 h and 39 ± 2 h, consistent with loss of rhythm; the peaks for these two genes were far apart in Figure (4.14). The remaining two genes, NCU09843 and NCU00476, were found to have periods of 22 ± 2 and 20 ± 2 h consistent with those observed in race tubes to assay circadian rhythms[5], but a periodicity test based on amplitude was not significant for NCU00476 (P=0.0519)[76]. Results of the same periodicity test for the remaining 4 targets were significant at the 0.05 level, although the periodicity test based on amplitude for NCU09843 was barely significant (P=0.0355). Retention of a circadian rhythm in NCU09843 could be due to another transcription factor (*e.g.*, WCC) transmitting the circadian signal (see Table (4.5)), mis-assignment of the target to its predicted transcription factor by our algorithm, or other regulatory mechanisms at work on circadian genes. Interestingly, the binding strength of WCC to NCU09843 is stronger than to the other targets in Table (4.5).

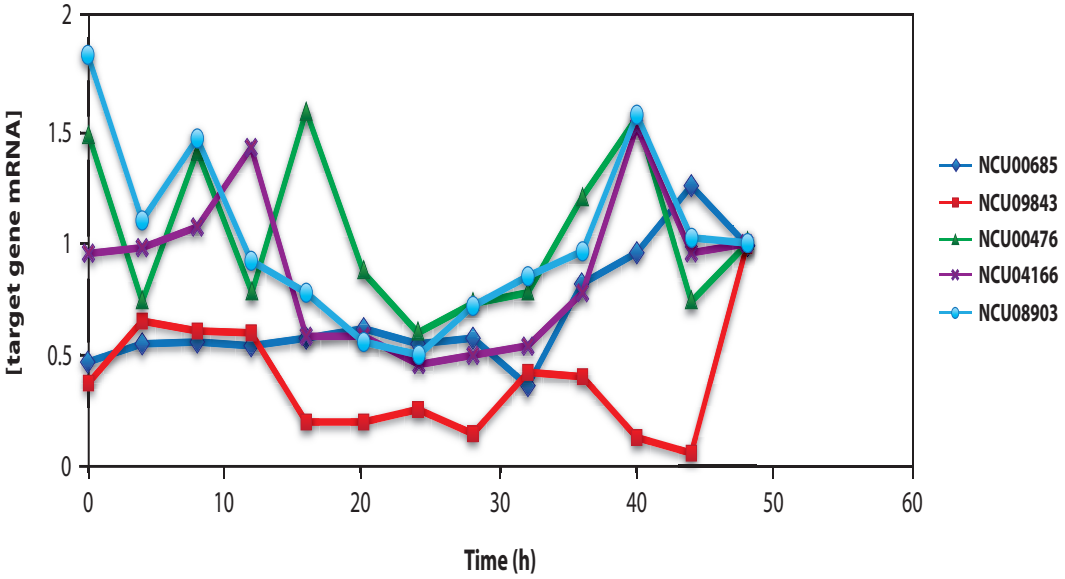

Figure 4.14: Change in Relative expression (RQ) of 4 of 5 target genes involved in ribosome biogenesis were correctly predicted as circadian or not circadian by the network model under knockout of NCU001640 (blue in Figure (4.10)) and NCU06108 (green in Figure (4.10)). RQ was determined by RT-qPCR (See Materials and Methods) using ubiquitin as endogenous control.

4.4 CONCLUSION

New algorithms for solving large systems of ODEs in parallel on the general-purpose graphical processing units (GPUs) allowed us to identify the dynamics of a genome-scale genetic network of unknown regulatory topology Figure (4.1) and Figure (4.2) using a supernet that consisted of 2,418 putative *ccgs* as shown in Figure (4.3). In over 40 years of clock biology, a set of 295 genes that are circadian, light-responsive, and under WCC-control have been identified and span a broad array of functions[5, 79]. To date these are likely to be *clock-controlled genes*. In this paper, we successfully fitted the dynamics and the rhythms of all 2,418 genes that are circadian in Figure (4.7) by ensemble methods implemented on a GPU and assigned each of them a place in a larger regulatory hierarchy. In addition to that, the simulation identified these genes' regulators and

assigned putative functions in Figure (4.12). The final best network identified involved a hierarchical regulatory network in Figure (4.10) with CSP-1 acting as a repressor. In this network *rpn-4* (NCU01640) and NCU06108 were assigned to regulate ribosome biogenesis in Figure (4.12). This connection between the clock and ribosome biogenesis was previously reported[5] and has been recently confirmed in mouse[80]. This connection of the clock to ribosome biogenesis raises the possibility that there are other regulatory mechanisms at work beyond transcriptional control[81] in Figure (4.10).

Strengths of Supernet Method. One of the features of the model in Figure (4.1) was the independent regulation of the target gene modules. Some of the target genes are likely to interact with the clock module in Figure (4.3) or with each other. The supernet method can be generalized by replacing the regulators with other kinds of regulatory modules involving post-transcriptional regulation mechanisms. One example of alternative regulatory modules would be the RNA operon[82]. Such alternative regulatory modules have the effect of linking together certain clusters of target genes by new kinds of post-transcriptional regulators and hence weakening the assumption of independence in target gene modules. This approach is completely feasible to implement on GPUs.

Another strength of ensemble methods in systems biology is data integration. Ideker et al. [11] integrated RNA profiling data as used here with protein profiling data. For example, under materials and methods we simplified the reconstruction problem by assuming the initial concentration of the regulatory proteins was a time average of the regulatory protein concentration because this is the kind of data most people have. This

assumption allowed us to calculate the trajectory of each regulatory protein. If one were to have the time and resources, it would be useful to carry out Westerns on all of the regulator proteins in Figure (4.10) to determine their protein profiles. We predict this will allow us to differentiate the ensemble hypotheses in Figure (4.10) with protein profiling data on the regulators.

A third strength of the Supernet Method is the possibility of generalization to multiple roles for regulators. In equations (10)-(12) we assumed each regulator had one of two roles as activator or repressor. In the same way that there could be different regulatory modules, there could also be a partition of the role of one regulator as activator or repressor depending on the targets. The supernet method allows us the possibility of generalizing equations (10)-(12) and to test a regulatory structure with multiple roles for each regulator, depending on the target module.

CHAPTER 5

CONCLUSION AND FUTURE WORK

We discovered and proposed new parallel solvers for large systems of ordinary differential equations on the general purpose graphical processing unit that help to discover and solve an important biological problem of genetic network identification with special reference to *Neurospora crassa's* biological clock, including: (A) Galerkin finite element method using piecewise hat functions explained in chapter 2, (B) two parallel algorithms for the Adaptive Runge Kutta method explained in chapter 3 and 4 and (C) and Gauss-Legendre quadrature method explained in chapter 4.

These solvers gave us the ability to discover a broad array of functions for *clock-controlled genes* as well as the gene regulators for those genes [10]as it is shown Figure (4.12) within few days instead of years using a GPGPU. We were able to infer the function of each regulator in Figure (4.12). We were able to discover how the *clock-controlled genes* were organized into a regulatory network. We found for the first time an explicit regulatory connection between the clock and ribosome biogenesis, which can now be tested. Each of these advancements were made possible by a new computational approach using GPUs.

Moreover, we proposed various statistical, mathematical and numerical methods throughout this dissertation. For examples, we showed for the first time how you could

use the Newton method for solving systems of algebraic equations along with the Galerkin finite element method; we showed how to use Monte Carlo metropolis and ensemble methods for solving large genetic network consisting of large systems of ODEs using the heterogeneity of the CPU and the GPU; we described how to 1) use Müller-Box updates for unit conversion factor variables, 2) obtain the regulator protein concentrations, 3) use Metropolis Monte Carlo updating algorithm for θ-variable (initial concentrations and parameters) and μ-variables(regulator binding strengths), 4) use for the first time a statistical mathematical model for six active regulators that regulate 2413 genes and the change on that model when one of those active regulators becomes a repressor, and we showed the procedure of the real-time quantitative polymerase chain reaction (RT-qPCR) of regulator targets for circadian rhythm.

In the future, we wish to improve the method of Galerkin by applying alternative numerical methods such as Quasi Newton method, supported wavelets or other types of finite element basis functions, such as Hermite finite elements and compare the accuracy and the speed of these alternative numerical methods with each other and with adaptive Runge Kutta method. Moreover, we need to implement the Galerkin method on the GPU and try to make it the fastest and the most accurate ODEs solver.

In addition to that we need to compare the impact of the three proposed ODE solvers on the genetic network solution including the regulators' genes assignment and the data fitting. We need to show the difference in the results by involving CSP-1 (NCU00045) as an active regulator and as a repressor. By the same token, we will apply Wang-Landau

algorithm as an alternative to the Metropolis Algorithm and see which will fit the data better and have the results over average of hundreds of best simulations instead of one simulation.

Finally, the supernet method can be generalized by replacing the regulators with other kinds of regulatory modules involving post-transcriptional regulation mechanisms. One example of alternative regulatory modules would be the RNA operon[82]. Such alternative regulatory modules have the effect of linking together certain clusters of target genes by new kinds of post-transcriptional regulators and hence weakening the assumption of independence in target gene modules. This approach is completely feasible to implement on GPUs.

REFERENCES

[1] C. Cogoni, and G. Macino, "Gene silencing in *Neurospora crassa* requires a protein homologous to RNA-dependent RNA polymerase," *Nature,* vol. 399, no. 6732, pp. 166-169, 1999.

[2] H. Tamaru, and E. U. Selker, "A histone H3 methyltransferase controls DNA methylation in *Neurospora crassa*," *Nature,* vol. 414, no. 6861, pp. 277-283, 2001.

[3] J. E. Galagan, S. E. Calvo, K. A. Borkovich, E. U. Selker, N. D. Read, D. Jaffe, W. FitzHugh, L.-J. Ma, S. Smirnov, and S. Purcell, "The genome sequence of the filamentous fungus *Neurospora crassa*," *Nature,* vol. 422, no. 6934, pp. 859-868, 2003.

[4] Y. Yu, W. Dong, C. Altimus, X. Tang, J. Griffith, M. Morello, L. Dudek, J. Arnold, and H. B. Schüttler, "A genetic network for the clock of *Neurospora crassa*," *Proceedings of the National Academy of Sciences,* vol. 104, no. 8, pp. 2809-2814, 2007.

[5] W. Dong, X. Tang, Y. Yu, R. Nilsen, R. Kim, J. Griffith, J. Arnold, and H. B. Schüttler, "Systems biology of the clock in *Neurospora crassa*," *PloS one,* vol. 3, no. 8, pp. e3105, 2008.

[6] A. Al-Omari, J. Arnold, T. Taha, and H. Schuttler, "Solving Large Nonlinear Systems of First-Order Ordinary Differential Equations With Hierarchical Structure Using Multi-GPGPUs and an Adaptive Runge Kutta ODE Solver."

[7] D. Battogtokh, D. Asch, M. Case, J. Arnold, and H.-B. Schüttler, "An ensemble method for identifying regulatory circuits with special reference to the qa gene cluster of *Neurospora crassa*," *Proceedings of the National Academy of Sciences,* vol. 99, no. 26, pp. 16904-16909, 2002.

[8] Y. Huang, "Parameter estimation of chemical reaction networks.A Master thesis, Physics and Astronomy Department, The University of Georgia.," 2007.

[9] A. K. Chakraborty, and J. Das, "Pairing computation with experimentation: a powerful coupling for understanding T cell signalling," *Nat Rev Immunol,* vol. 10, no. 1, pp. 59-71, Jan, 2010.

[10] A. Al-Omari, J. Griffith, M. Judge, T. Taha, J. Arnold, and H.-B. Schüttler, "Discovering Regulatory Network Topologies Using Ensemble Methods on GPGPUs with Special Reference to the Biological Clock of *Neurospora crassa*," *IEEE , Access,* vol. 3, pp. 27 - 42, 2015.

[11] T. Ideker, V. Thorsson, J. A. Ranish, R. Christmas, J. Buhler, J. K. Eng, R. Bumgarner, D. R. Goodlett, R. Aebersold, and L. Hood, "Integrated genomic and proteomic analyses of a systematically perturbed metabolic network," *Science,* vol. 292, no. 5518, pp. 929-934, May 4, 2001.

[12] I. M. Keseler, J. Collado-Vides, A. Santos-Zavaleta, M. Peralta-Gil, S. Gama-Castro, L. Muniz-Rascado, C. Bonavides-Martinez, S. Paley, M. Krummenacker, T. Altman, P. Kaipa, A. Spaulding, J. Pacheco, M. Latendresse, C. Fulcher, M. Sarker, A. G. Shearer, A. Mackie, I. Paulsen, R. P. Gunsalus, and P. D. Karp, "EcoCyc: a comprehensive database of Escherichia coli biology," *Nucleic Acids Res,* vol. 39, no. Database issue, pp. D583-D590, Jan, 2011.

[13] S. Okuda, T. Yamada, M. Hamajima, M. Itoh, T. Katayama, P. Bork, S. Goto, and M. Kanehisa, "KEGG Atlas mapping for global analysis of metabolic pathways," *Nucleic Acids Res,* vol. 36, no. Web Server issue, pp. W423-W426, Jul 1, 2008.

[14] R. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. S. Jr, N. Kyrpides, M. Fonstein, N. Maltsev, and E. Selkov, "WIT:integrated system for high-throughput genome sequence analysis and metabolic reconstruction," *Nucleic Acids Res,* vol. 28, pp. 123-125, 13 October, 2000.

[15] J. L. DeRisi, V. R. Iyer, and P. O. Brown, "Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale," *Science,* vol. 278, pp. 26-37, October 24, 1997.

[16] S. J. Maerkl, and S. R. Quake, "A systems approach to measuring the binding energy landscapes of transcription factors," *Science,* vol. 315, no. 5809, pp. 233-237, Jan 12, 2007.

[17] J. R. Karr, J. C. Sanghvi, D. N. Macklin, M. V. Gutschow, J. M. Jacobs, B. Bolival, Jr., N. Assad-Garcia, J. I. Glass, and M. W. Covert, "A whole-cell computational model predicts phenotype from genotype," *Cell,* vol. 150, no. 2, pp. 389-401, Jul 20, 2012.

[18] M. W. Covert, E. M. Knight, J. L. Reed, M. J. Herrgard, and B. O. Palsson, "Integrating high-throughput and computational data elucidates bacterial networks," *Nature,* vol. 429, no. 6987, pp. 82-96, May 6, 2004.
[19] X. Tang, W. Dong, J. Griffith, R. Nilsen, A. Matthes, K. B. Cheng, J. Reeves, H. B. Schüttler, M. E. Case, and J. Arnold, "Systems Biology of the qa Gene Cluster in *Neurospora crassa*," *PloS ONE,* vol. 6, no. 6, pp. e20671, 2011.
[20] S. H. Strogatz, "Exploring complex networks," *Nature* vol. 410, pp. 268-276, 8 March, 2001.
[21] M. Delfour, F. Trochu, and W. Hagar, "Discontinuous Galerkin methods for ordinary differential equations," *Mathematics of Computation,* vol. 36, no. 154, pp. 455-473, 1981.
[22] M. Delfour, and F. Dubeau, "Discontinuous polynomial approximations in the theory of one-step, hybrid and multistep methods for nonlinear ordinary differential equations," *Math. Comp,* vol. 47, pp. 169-189, 1986.
[23] C. Johnson, "Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations " *Society for Industrial and Applied Mathematics(SIAM),* vol. 25, no. 4, pp. 908-926, 1988.
[24] F. Dubeau, A. Ouansafi, and A. Sakat, "Galerkin methods for nonlinear ordinary differential equation with impulses," *Numerical Algorithms,* vol. 33, no. 1-4, pp. 215-225, 2003.
[25] H. Temimi, S. Adjerid, and M. Ayari, "Implementation of the Discontinuous Galerkin Method on a Multi-Story Seismically Excited Building Model," *Engineering Letters,* vol. 18, no. 18-27, 2010.
[26] B. L. Hulme, "One-step piecewise polynomial Galerkin methods for initial value problems," *Math. Comp,* vol. 26, pp. 415-426, 1972.
[27] D. Estep, "A posteriori error bounds and global error control for approximation of ordinary differential equations," *Society for Industrial and Applied Mathematics(SIAM),* vol. 32, no. 1, pp. 1-48, 1995.
[28] K. Bottcher, and R. Rannacher, "Adaptive error control in solving ordinary differential equations by the discontinuous Galerkin Method, Technical Report, University of Heidelberg," pp. 1-31, 1997.
[29] J. Arnold, R. T. Thiab, and L. Deligiannidis, "GKIN: a tool for drawing genetic networks," *Network Biology,* vol. 2, pp. 26-37, 2012.
[30] B. Aleman-Meza, Y. Yu, H. B. Schüttler, J. Arnold, and T. R. Taha, "KINSOLVER: A simulator for computing large ensembles of biochemical and gene regulatory networks," *Computers & Mathematics with Applications,* vol. 57, no. 3, pp. 420-435, 2009.
[31] M. O'Brien, and M. Walton, "Biological Clocks.," *Science Nation,* http://www.nsf.gov/news/special_reports/science_nation/biologicalclocks.jsp, 2010.
[32] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in Escherichia coli," *Nature,* vol. 403, pp. 339-342, 2000.
[33] I. Daubechies, "Orthonormal bases of compactly supported wavelets II. variations on a theme," *SIAM J. Math. Anal.,* vol. 24, no. 2, pp. 499-519, 1993.
[34] E. W. Cheney, and D. R. Kincaid, *Numerical mathematics and computing*: Brooks/Cole Publishing Company, 2012.
[35] H. de Jong, and D. Ropers, "Strategies for dealing with incomplete information in the modeling of molecular interaction networks," *Briefings in Bioinformatics,* vol. 7, no. 4, pp. 354-363, 2006.
[36] T. R. Dickey, "Single Cell Measurements on the biological clock by microfluidics," *Video attachment*.
[37] J. Jaeger, S. Surkova, M. Blagov, H. Janssens, D. Kosman, and K. N. Kozlov, "Dynamic control of positional information in the early Drosophila embryo," *Nature,* vol. 430, no. 6997, pp. 368-371, 2004.
[38] D. Luebke, M. Harris, N. Govindaraju, A. Lefohn, M. Houston, J. Owens, M. Segal, M. Papakipos, and I. Buck, "GPGPU: general-purpose computation on graphics hardware." p. 208.
[39] S. Ohshima, K. Kise, T. Katagiri, and T. Yuba, "Parallel processing of matrix multiplication in a CPU and GPU heterogeneous environment," *High Performance Computing for Computational Science-VECPAR 2006*, pp. 305-318: Springer, 2007.
[40] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A Survey of General‐Purpose Computation on Graphics Hardware." pp. 80-113.

[41] C. J. Thompson, S. Hahn, and M. Oskin, "Using modern graphics architectures for general-purpose computing: a framework and analysis." pp. 306-317.
[42] W. Liu, B. Schmidt, and W. Müller-Wittig, "Performance analysis of general-purpose computation on commodity graphics hardware: A case study using bioinformatics," *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology,* vol. 48, no. 3, pp. 209-221, 2007.
[43] M. Charalambous, P. Trancoso, and A. Stamatakis, "Initial experiences porting a bioinformatics application to a graphics processor," *Advances in Informatics*, pp. 415-425: Springer, 2005.
[44] L. Murray, "GPU acceleration of Runge-Kutta integrators," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 23, no. 1, pp. 94-101, 2012.
[45] G. Khanna, "High-Precision Numerical Simulations on a CUDA GPU: Kerr Black Hole Tails," *Journal of Scientific Computing*, pp. 1-15, 2013.
[46] C. Gribble, and A. Naveros, "GPU ray tracing with rayforce." p. 98.
[47] T. L. Falch, J. B. Floystad, D. W. Breiby, and A. C. Elster, "GPU Accelerated Visualization of Scattered Point Data," *IEEE Access,* vol. 1, pp. 564-576, 2013.
[48] S. Park, and H. Shin, "Efficient generation of adaptive Cartesian mesh for computational fluid dynamics using GPU," *International Journal for Numerical Methods in Fluids,* vol. 70, no. 11, pp. 1393-1404, 2012.
[49] I. Demeshko, N. Maruyama, H. Tomita, and S. Matsuoka, "Multi-GPU implementation of the NICAM atmospheric model." pp. 175-184.
[50] D. Battogtokh, D. K. Asch, M. E. Case, J. Arnold, and H. B. Schüttler, "An ensemble method for identifying regulatory circuits with special reference to the qa gene cluster of *Neurospora crassa*," *Proceedings of the National Academy of Sciences,* vol. 99, no. 26, pp. 16904-16909, 2002.
[51] H. Ukai, and H. R. Ueda, "Systems biology of mammalian circadian clocks," *Annual review of physiology,* vol. 72, pp. 579-603, 2010.
[52] J. Passerat-Palmbach, J. Caux, P. Siregar, and D. R. C. Hill, "Warp-level parallelism: Enabling multiple replications in parallel on GPU." pp. 24-26.
[53] A. Al-Omari, H. B. Schuttler, J. Arnold, and T. Taha, "Solving Nonlinear Systems of First Order Ordinary Differential Equations Using a Galerkin Finite Element Method," *Access, IEEE,* vol. 1, pp. 408-417, 2013.
[54] S. Ryoo, *Program optimization strategies for data-parallel many-core processors*: ProQuest, 2008.
[55] C.-H. Chen, C. S. Ringelberg, R. H. Gross, J. C. Dunlap, and J. J. Loros, "Genome-wide analysis of light-inducible responses reveals hierarchical light signalling in Neurospora," *The EMBO journal,* vol. 28, no. 8, pp. 1029-1042, 2009.
[56] L. Chen, O. Villa, S. Krishnamoorthy, and G. R. Gao, "Dynamic load balancing on single-and multi-GPU systems." pp. 1-12.
[57] D. Segre, A. DeLuna, G. M. Church, and R. Kishony, "Modular epistasis in yeast metabolism," *Nature genetics,* vol. 37, no. 1, pp. 77-83, 2004.
[58] Y. Zhou, J. Liepe, X. Sheng, M. P. H. Stumpf, and C. Barnes, "GPU accelerated biochemical network simulation," *Bioinformatics,* vol. 27, no. 6, pp. 874-876, 2011.
[59] A. C. Hindmarsh, "ODEPACK, A Systematized Collection of ODE Solvers, RS Stepleman et al.(eds.), North-Holland, Amsterdam,(vol. 1 of), pp. 55-64," *IMACS transactions on scientific computation,* vol. 1, pp. 55-64, 1983.
[60] V. M. Garcia, A. Liberos, A. M. Climent, A. Vidal, J. Millet, and A. Gonzalez, "An adaptive step size GPU ODE solver for simulating the electric cardiac activity." pp. 233-236.
[61] G. M. Whitesides, "The origins and the future of microfluidics," *Nature,* vol. 442, no. 7101, pp. 368-373, 2006.
[62] D. Gonze, J. Halloy, and A. Goldbeter, "Robustness of circadian rhythms with respect to molecular noise," *Proceedings of the National Academy of Sciences,* vol. 99, no. 2, pp. 673-678, 2002.
[63] J. Jaeger, S. Surkova, M. Blagov, H. Janssens, D. Kosman, K. N. Kozlov, Manu, E. Myasnikova, C. E. Vanario-Alonso, M. Samsonova, D. H. Sharp, and J. Reinitz, "Dynamic control of positional information in the early Drosophila embryo," *Nature,* vol. 430, no. 6997, pp. 368-371, Jul 15, 2004.

[64] E. Lee, J. de Ridder, J. Kool, L. F. Wessels, and H. J. Bussemaker, "Identifying regulatory mechanisms underlying tumorigenesis using locus expression signature analysis," *Proceedings of the National Academy of Sciences,* vol. 111, no. 15, pp. 5747-5752, 2014.

[65] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins, "Inferring genetic networks and identifying compound mode of action via expression profiling," *Science,* vol. 301, no. 5629, pp. 102-5, Jul 4, 2003.

[66] A. Al-Omari, J. Arnold, T. Taha, and H. Schuttler, "Solving Large Nonlinear Systems of First-Order Ordinary Differential Equations With Hierarchical Structure Using Multi-GPGPUs and an Adaptive Runge Kutta ODE Solver," *IEEE , Access,* vol. 1, pp. 770-777, 2013.

[67] A. Al-Omari, H.-B. Schuttler, J. Arnold, and T. Taha, "Solving Nonlinear Systems of First Order Ordinary Differential Equations Using a Galerkin Finite Element Method," *Access, IEEE,* vol. 1, pp. 408-417, 2013.

[68] Z. Zhang, and J. P. Townsend, "The filamentous fungal gene expression database (FFGED)," *Fungal Genetics and Biology,* vol. 47, no. 3, pp. 199-204, 2010.

[69] M. Matsumoto, and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation (TOMACS),* vol. 8, no. 1, pp. 3-30, 1998.

[70] G. E. Box, and M. E. Muller, "A note on the generation of random normal deviates," *The annals of mathematical statistics,* vol. 29, no. 2, pp. 610-611, 1958.

[71] G.W. Snedecor and W. G. Cochran, Statistical Met*hods,* 7th ed. Ames, IA,USA: Iowa State Univ. Press, 1980, pp. 215_237

[72] G. von Winckel, http://www.mathworks.com/matlabcentral/fileexchange/4540-legendre-gauss-quadrature-weights-and-nodes, 2004.

[73] G. W. Snedecor, and W. G. Cochran, "Statistical Methods, 7nth Edition," pp. 215-237, 1980.

[74] H. V. Colot, G. Park, G. E. Turner, C. Ringelberg, C. M. Crew, L. Litvinkova, R. L. Weiss, K. A. Borkovich, and J. C. Dunlap, ""A high-throughput gene knockout procedure for Neurospora reveals functions for multiple transcription factors" (vol 103, pg 10352, 2006)," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 103, no. 44, pp. 16614-16614, Oct 31, 2006.

[75] S. A. Bustin, V. Benes, J. A. Garson, J. Hellemans, J. Huggett, M. Kubista, R. Mueller, T. Nolan, M. W. Pfaffl, and G. L. Shipley, "The MIQE guidelines: minimum information for publication of quantitative real-time PCR experiments," *Clinical chemistry,* vol. 55, no. 4, pp. 611-622, 2009.

[76] M. E. Case, J. Griffith, W. Dong, I. L. Tigner, K. Gaines, J. C. Jiang, S. M. Jazwinski, and G. C. Study, "The aging biological clock in *Neurospora crassa*," *Ecology and Evolution,* vol. 4, pp. 3494-3507, 2014.

[77] G. Sancar et al "A Global Circadian Repressor Controls Antiphasic Expression of Metabolic Genes in Neurospora ," *Molecular cell,* vol. 44, no. 5, pp. 687-697, 2011.

[78] Online, "KEGG Mapper," http://www.genome.jp/kegg/tool *map_pathway1.html*, 2014.

[79] R. M. de Paula, Z. A. Lewis, A. V. Greene, K. S. Seo, L. W. Morgan, M. W. Vitalini, L. Bennett, R. H. Gomer, and D. Bell-Pedersen, "Two circadian timing circuits in *Neurospora crassa* cells share components and regulate distinct rhythmic processes," *Journal of biological rhythms,* vol. 21, no. 3, pp. 159-168, 2006.

[80] C. l. Jouffe, G. Cretenet, L. Symul, E. Martin, F. Atger, F. Naef, and F. d. r. Gachon, "The circadian clock coordinates ribosome biogenesis," *Plos Biology,* vol. 11, no. 1, pp. e1001455, 2013.

[81] J. M. Hurley, A. Dasgupta, J. M. Emerson, X. Zhou, C. S. Ringelberg, N. Knabe, A. M. Lipzen, E. A. Lindquist, C. G. Daum, and K. W. Barry, "Analysis of clock-regulated genes in Neurospora reveals widespread posttranscriptional control of metabolic potential," *Proceedings of the National Academy of Sciences*, pp. 201418963, 2014.

[82] J. Keene, B. Stillman, D. Stewart, and T. Grodzicker, "Biological Rhythms Workshop IA: molecular basis of rhythms generation," in Cold Spring Harbor symposia on quantitative biology, Woodbury, NY, 2007, pp. 157-165.

APPENDIX A

Biography

AHMAD AL-OMARI received the B.Sc. degree in Electrical and Computer Engineering from Yarmouk University, Irbid, Jordan, in 2004. He was a Teaching Assistant and Lab Engineer with the Department of Electrical and Computer Engineering, Yarmouk University, from 2004 to 2010. He was involved in more than 30 different projects on computers, communication, and electronic engineering. He received a grant of over $150 000 from Yarmouk University to pursue the Ph.D. degree in bioinformatics and in the same theme he received a Teaching Assistant grant of over $133 000 from The University of Georgia. He is currently a Research and Teaching Assistant and the Ph.D. degree in bioinformatics under the supervision of Prof. Jonathan Arnold with the University of Georgia, Athens, GA, USA, on systems biology and genetic networks. His current research interests include parallel computation, systems biology, finite elements method, machine learning and pattern recognition, biological circuits and gene networks, and numerical analysis.