MODELLING FUNCTIONAL DYNAMICAL SYSTEMS BY PIECEWISE LINEAR SYSTEMS WITH DELAY

MUSTAFA KAHRAMAN

SEPTEMBER 2007

MODELLING FUNCTIONAL DYNAMICAL SYSTEMS BY PIECEWISE LINEAR SYSTEMS WITH DELAY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA KAHRAMAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF SCIENTIFIC COMPUTING

SEPTEMBER 2007

Approval of the Graduate School of Applied Mathematics

Prof. Dr. Ersan Akyıldız
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Bülent Karasözen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Marat Akhmet
Co-supervisor

Assist. Prof. Dr. Hakan Öktem
Supervisor

Examining Committee Members

Prof. Dr. Gerhard Wilhelm Weber	(METU,IAM)	
Assist. Prof. Dr. Hakan Öktem	(METU,IAM)	
Assoc. Prof. Dr. Ü. Erkan Mumcuoğlu	(METU,II)	
Assist. Prof. Ali Ulaş Özgür Kişisel	(METU,MATH)	

ABSTRACT

MODELLING FUNCTIONAL DYNAMICAL SYSTEMS BY PIECEWISE LINEAR SYSTEMS WITH DELAY

Kahraman, Mustafa
M.Sc., Department of Scientific Computing
Supervisor: Assist. Prof. Dr. Hakan Öktem
Co-supervisor: Prof. Dr. Marat Akhmet

September 2007, 130 pages

Many dynamical systems in nature and technology involve delays in the interaction of variables forming the system. Furthermore, many of such systems involve external inputs or perturbations which might force the system to have arbitrary initial function. The conventional way to model these systems is using delay differential equations (DDE). However, DDEs with arbitrary initial functions has serious problems for finding analytical and computational solutions. This fact is a strong motivation for considering abstractions and approximations for dynamical systems involving delay. In this thesis, the piecewise linear systems with delay on piecewise constant part which is a useful subclass of hybrid dynamical systems is studied. We introduced various representations of these systems and studied the state transition conditions. We showed that there exists fixed point and periodic stable solutions. We modelled the genomic regulation of fission yeast cell cycle. We discussed various potential uses including approximating the DDEs and finally we concluded.

Keywords: piecewise linear systems, delay systems, external input, hybrid dynamical systems with delay, regulatory gene networks.

ÖZ

FONKSİYONEL DİNAMİK SİSTEMLERİN GECİKMELİ PARÇALI DOĞRUSAL SİSTEMLER İLE MODELLENMESİ

Kahraman, Mustafa
Yüksek Lisans, Bilimsel Hesaplama Bölümü
Tez Yöneticisi: Assist. Prof. Dr. Hakan Öktem
Tez Yardımcısı: Prof. Dr. Marat Akhmet

Eylül 2007, 130 sayfa

Doğa ve teknoloji ile ilgili birçok dinamik sistem değişkenlerin etkileşiminde gecikmeler içerir. Üstelik bu sistemlerin birçoğu başlangıç fonksiyonunun gelişigüzel olmasına neden olan dış girdilere ve düzensizliklere sahiptir. Gecikmeli diferansiyel denklemler (DDE) ile böyle sistemlerin modellenmesi geleneksel yöntemdir. Ancak gelişigüzel başlangıç fonksiyonlu gecikmeli diferansiyel denklemlerin analitik ve sayısal çözümlerini bulmada ciddi sorunlar vardır. Bu durum gecikmeli dinamik sistemlerin soyutlanmasının ve yaklaşımının göz önünde bulundurulması için güçlü bir motivasyondur. Bu tezde, hibrid dinamik sistemlerin bir alt kolu olan parçalı sabit kısmında gecikme içeren parçalı doğrusal sistemler çalışılmıştır. Böyle sistemlerin değişik gösterimleri tanıtılmış ve hal değiştirme koşulları çalışılmıştır. Mayanın hücre bölünmesinin genetik düzenlenmesi modellenmiştir. DDE'lerin yaklaşımını kapsayan değişik uygulamalar ele alınmış ve son olarak sonuca varılmıştır.

Anahtar Kelimeler: parçalı doğrusal sistemler, gecikmeli sistemler, dış girdi, hibrid dinamik sistemler, düzenleyici gen ağları.

To my family

ACKNOWLEDGMENT

I would like to thank my supervisor Assist. Prof. Dr. Hakan ÖKTEM for patiently guiding, motivating, and encouraging me throughout this study.

I am thankful to my co-supervisor Prof. Dr. Marat AKHMET for his kindness and suggestions.

Though I know, it is not completely possible to express my gratitude, I would like to thank to a few special people.

I want to thank my dear friends Müşerref, Derya, Nurgül, Nilüfer for their endless support and being with me all the time.

Also, the environment of Institute of Applied Mathematics (IAM) itself has an important place in this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background to the Study

Constructing mathematical models of known dynamical processes is a fundamental method for many nature science and engineering problems. Once a mathematical model for a dynamical phenomena has been constructed, it can be used for predicting the future behavior from the initial conditions, anticipating the suitable interventions if required or designing a technological system exhibiting the desired behavior.

Recent developments like growing knowledge on genomic, neurophysiologic, ecological processes improved instrumentation for large scale control systems and increased computational power for simulating more complex dynamical system models stimulated a growing interest on developing the dynamical system model classes.

A promising alternative in this direction is the use of hybrid dynamical systems. Hybrid systems are systems formed by continuous and boolean variables regulating each other [32, 52, 78]. Hybrid systems might be one of the oldest control technologies humanity has used. However, their mathematical formalization is recent due to the developments on control systems. Nowadays, hybrid systems offer several advances for various modelling [18, 52, 27] and theoretical problems [19, 52] in nature science. A very useful subclass of hybrid systems is the piecewise linear systems. The most attractive feature of piecewise linear systems is their capability of substituting more complex systems into locally solvable systems. This is essentially important when simulation, solution or analysis of a first principle model of a dynamical process is not possible with reasonable computational sources. In these cases, approximations and abstractions of an exact model is much more useful and hybrid dynamical systems form a significant tool for this purpose.

Increasing effort for modelling history dependent systems like genomic regulation [39], tracking problems led researchers investigating functional dynamical systems. When time delays are introduced between the interactions of the variables forming the system, the behavior of the system depends not only to the initial state. The future of the system is determined by the initial functions of the variables for the delay duration. Especially, when delays coexist with multistationarity they result with history dependent responses of the systems which underlie many complicated adaptation and learning tasks in living organisms. Integral equations which can not be converted into differential equations, delay differential equations, delay equations of some more abstract functional differential equations are typical model of these systems.

Even thought theoretical methods for DDEs and various FDEs are developed for understanding the functional dynamical systems, their simulation requires abstractions and approximations. Especially, when the system is affected by external factors like environmental, control or initialization inputs, the initial functions might be arbitrary leading to the lack of analytical solutions.

1.2 Brief Description of the Study

In this thesis we studied piecewise linear systems with delay where the delay is introduced into the state transitions. By this approach we handle the functional part of the system in the piecewise constant part. Firstly, we studied the stability analysis of piecewise linear systems with delay. A Poincare map is constructed and existence of fixed point of this map is investigated using the contraction principle. After the stability study we introduced a more useful model to cell cycle of fission yeast model. The original model includes differential and algebraic equations. We constructed a new model by a piecewise linear system with delay preserving the basic dynamics and relations in the regulation of cell division cycle of fission yeast. Then, we introduced the idea of approximating delay differential equations by hybrid dynamical systems with delay. In this thesis, we approximated a simple delay differential equation by a piecewise linear system with delay. Finally, we added the other possible applications of hybrid dynamical systems with delay.

1.3 Purpose of the Study

The contribution of this work consists in modelling functional dynamical systems by piecewise linear systems with delay. We approximate the delay in functional dynamical systems by including the delay in state transitions of piecewise

linear system. This idea makes it possible to observe the behavior of functional dynamical systems in an easy way. We investigated the stability of piecewise linear systems with delay and observed that stable periodic solutions can be seen. Conditions to see stable periodic solutions are stated. Regulatory gene networks have multistationary nature and delayed responses. Therefore, history dependent responses are seen in these systems. Modelling such processes by systems with reduced complexity is important in terms of observing the behavior with reasonable computational sources. In this work, it is aimed to reconstruct a model to cell division cycle regulation of fission yeast using a piecewise linear system with delay. It is claimed that the process of cell cycle can be modelled including more known information related to the process. Therefore, a more realistic model can be obtained by using piecewise linear systems with delay. Additionally, the systems will be easier because we will deal with linear differential equations. The idea of approximating delay differential equations by piecewise linear systems with delay is introduced. The purpose of this introduction is to show that delay differential equations can be approximated more easily by using adaptive sampling which comes with the use of piecewise linear systems. Constructing chaos generators by piecewise linear systems with delay can be considered as a further research study. Additionally, memorization of external inputs in regulatory gene networks can be investigated in terms of piecewise linear systems with delay.

1.4 Significance of the Study

There are three crucial properties of this work. Firstly, we showed that the stable periodic solutions can be seen in piecewise linear systems with delay. A Poincare map can be constructed and stability analysis can be done in this way. Secondly, functional dynamical systems can be modelled by piecewise linear systems with delay. Simple models can be used to approximate the complex dynamics. Finally, regulatory gene networks can be divided into subsystems and the output of each system can be the external input of other systems. Whole details of the regulatory gene networks can be included in piecewise linear systems with delay model by this way.

CHAPTER 2

BACKGROUND

This section contains background material for the study. Background material for the cell cycle regulation of fission yeast is included in the chapter named "Fission Yeast Cell Cycle Model".

2.1 Dynamical Systems

2.1.1 A Dynamical View of the World

Dynamical systems are aimed at describing the principles of nature in terms of mathematical relations. Natural sciences like mechanics, electricity, magnetism, and thermodynamics use dynamical systems formalism to describe behaviors of processes and to express laws of discipline. Dynamical systems describe the evolution of system in time. Some systems evolve in continuous time, and some others evolve in discrete time steps.

For example, the simple differential equation used to describe the population dynamics in continuous time is

$$\tfrac{dx}{dt} = kx,$$

where k represents the (fixed) relative growth rate of the population. Here, a differential equation is used to capture the dynamics of the system.

The differential equation can be solved by separating the variables

$$(\tfrac{1}{x})dx = kdt$$

and, integrating,

$$\log|x| = \textstyle\int(\tfrac{1}{x})dx = \int kdt = kt + C,$$

so

$$|x(t)| = e^C e^{kt} \qquad \text{with } e^C = |x(0)|,$$

$$x(t) = x(0)e^{kt}.$$

This solution gives the population in continuous t. This model can be used for species that can reproduce continuously. On the other hand, the discrete-time model for population dynamics is more appropriate when generations do not overlap [30] as observed in butterflies whose existence and reproduction is strictly seasonal. For this case, denoting the present population by x, next year's population is $f(x) = kx$ for some positive constant k, which is the average number of offspring per butterfly. If the population dynamics is evolved in discrete time step for each year, x_i denotes the population in year i. Therefore, $x_{i+1} = f(x_i) = kx_i$ and $x_1 = kx_0, x_2 = kx_1 = k^2x_0$, and so on. The population grows exponentially like in the continuous model.

2.1.2 Linear Maps

Maps are important for dynamical systems as seen in the discrete time model of population dynamics of butterfly. In this work, contraction on linear maps is used. Therefore, introductory information is presented related to scalar linear maps and contractions in Euclidean space for case of one variable and several variables.

Scalar Linear Maps and Linearization

Definition 2.1: A map is a way of associating unique objects to every point in a given set. So a map from $A \rightarrow B$ is an object f such that for every $a \in A$, there is a unique object $f(a) \in B$. The terms function and mapping are synonymous with map [7, 37, 72].

Maps are good mathematical tools for constructing models. Scalar linear map $x_{i+1} = f(x_i) = kx_i$ with $k > 0$ describes the dynamics of the primitive discrete time population model. This model diverges if $k > 1$ and goes to 0 if $k < 1$ for any initial value $x_0 \neq 0$. For this model, asymptotic behavior is independent of the initial condition.

Most interesting dynamical systems are not linear. However, by the help of linearization nonlinear systems can be approximated. Differentiability, means there exist a good linear approximation near any given point. Such linear approximations can sometimes be useful for dynamics when the orbits of a nonlinear map stay near enough to the reference point for the linear approximation to be relevant [30].

The proposition below gives relation of asymptotic behavior of maps and linearization of maps.

Proposition 2.1: Suppose F is a differentiable map of the line and $F(b) = b$. If all orbits of the linearization of F at b are asymptotic to b, then all orbits of F that start near enough to b are asymptotic to b as well.

Contraction in Euclidean Space

Firstly, define Euclidean space [29, 72].

Definition 2.2: Euclidean n-space is the space of all n-tuples of real numbers $(x_1, x_2, ..., x_n)$ and is denoted $\mathbb{R}^n \times \mathbb{R}^n$ is a vector space and has lebesque covering dimension n. Elements of $\mathbb{R}^n$ are called n-vectors, $\mathbb{R}^1 = \mathbb{R}$ is the set of real numbers (i.e. the real line), and $\mathbb{R}^n$ is called the Euclidean plane. In Euclidean space, covariant and contravariant quantities are equivalent, so $\overrightarrow{e}^j = \overrightarrow{e_j}$.

Now define contracting maps [30] with respect to the Euclidean distance

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

Definition 2.3: A map f of a subset X of Euclidean space is said to be Lipschitz continuous with Lipschitz constantλ, or λ-Lipschitz if

$$d(f(x), f(y)) \leq \lambda d(x,y)$$

for any $x, y \in X$. The map f is said to be a contraction or a λ $-$contraction if $\lambda < 1$. If a map is Lipschitz-continuous, then

$$Lip(f) = \sup_{x \neq y} \frac{d(f(x), f(y))}{d(x,y)}.$$

Example 2.1: The function $f(a) = \sqrt{a}$ is a contraction on $[1, \infty)$.

For two points $a, b \in [1, \infty)$ check

$$d(f(a), f(b)) \leq \lambda d(a, b);$$

for $f(a) = \sqrt{a}$

$$\sqrt{(\sqrt{a} - \sqrt{b})^2} \leq \lambda \sqrt{(a-b)^2}$$

is satisfied for $\lambda < 1$. So the function is a contraction.

The Case of One Variable

Derivative test [30] is an easy way of checking the contraction condition.

Proposition 2.2: Let I be an interval and $f : I \to \mathbb{R}$ a differentiable function with $|f'(x)| \leq \lambda$ for all $x \in I$. Then f is λ - Lipschitz.

Proof: By Mean-Value Theorem, for any two points $x, y \in I$ there exists a point c between x and y such that

$$d(f(x), f(y)) = |f(x) - f(y)| = |f'(c)(x-y)| = |f'(c)| d(x, y) \leq \lambda d(x, y).$$

Example 2.2: The function $f(a) = \sqrt{a}$ is a contraction on $[1, \infty)$.

Using Proposition 2.2 it is easy to find $|f'(x)| \leq \frac{1}{2x} \leq 0,5$ for $x \geq 1$ which defines contraction on $[1, \infty)$.

Proposition 2.3 gives stronger condition [30] for contraction.

Proposition 2.3: Let I be a closed bounded interval and $f : I \to I$ a continuously differentiable function with $|f'(x)| \leq 1$ for all $x \in I$. Then, f is a contraction.

Proof: The maximum λ of $|f'(x)|$ is attained at some point x_0, because f' is continuous. It is less than 1, since $|f'(x)| \leq 1$.

The evolution of dynamical process is represented as $f^n(x)$, in case of maps, where $f^n(x)$ means applying the function f to initial state n times. The terms orbit, initial value, fixed point, periodic point are defined in Definition 2.4.

Definition 2.4: Let x be a point and let f be a map. The orbit of x under f is the set of points $\{x, f(x), f^2(x), ..., f^n(x)\}$. The starting point x for the orbit is called the initial value of the orbit. A point p is a fixed point of the map f if $f(p) = p$. A periodic point is a point x such that $f^n(x) = x$ for some $n \in N$, that is a point in $Fix(f^n)$. Such an n is said to be a period of x. The smallest such n is called the prime period of x [5, 30].

Example 2.3: Consider the map $g(x) = 2x(1-x)$ from the real line to itself. The orbit of $x = 0.2$ under g is $\{0.2, 0.32, 0.4352....\}$ and the points $x = 0$ and $x = 0.5$ are the fixed point of g.

The convergence of sequence generated by orbits is determined by the Proposition 2.4 [30].

Proposition 2.4: (Contraction Principle) Let $I \subset \mathbb{R}$ be a closed interval, possibly infinite on one or both sides, and $f : I \to I$ a λ-contraction. Then f has a unique fixed point x_0 and $|f^n(x) - x_0| \leq \lambda^n |x - x_0|$ for every $x \in \mathbb{R}$, that is, every orbit of f converges to x_0 exponentially [30].

Proof: By iterating $|f(x) - f(y)| \leq \lambda |x - y|$, it is seen that

$$(1) \qquad |f^n(x) - f^n(y)| \leq \lambda^n |x - y|$$

for $x, y \in \mathbb{R}$ and $n \in \mathbb{N}$; so for $x \in I$ and $m \geq n$ the triangular inequality can be used to show

$$(2) \qquad |f^m(x) - f^n(x)| \leq \textstyle\sum_{k=0}^{m-n-1} |f^{n+k+1}(x) - f^{n+k}(x)|$$

$$\leq \textstyle\sum_{k=0}^{m-n-1} \lambda^{n+k}|f(x) - x| \leq \frac{\lambda^n}{1-\lambda}|f(x) - x|.$$

The right - hind side of (2) became arbitrarily small as n gets large. This shows that $(f^n(x))_n$ is a Cauchy sequence. Thus for any $x \in I$ the limit of $f^n(x)$ as $n \to \infty$ exists because Cauchy sequences converges. The limit is in I because I is closed. This limit is the same for all x. Denote this limit by x_0 and show that x_0 is a fixed point for f. If $x \in I$ and $n \in \mathbb{N}$, then

$$|x_0 - f(x_0)| \leq |x_0 - f^n(x_0)| + |f^n(x) - f^{n+1}(x)| + |f^{n+1}(x) - f(x_0)|$$

$$\leq (1 + \lambda)|x_0 - f^n(x)| + \lambda^n |x - f(x)|$$

Since $|x_0 - f^n(x)| \to 0$ and $\lambda^n \to 0$ as $n \to \infty$, $f(x_0) = x_0$ is obtained. That $|f^n(x) - x_0| \leq \lambda^n |x - x_0|$ for every $x \in \mathbb{R}$ follows from (1) with $y = x_0$.

Here, the familiar fact

$$\textstyle\sum_{k=0}^{n} r^k = \frac{1-r^{n+1}}{1-r}$$

and, as $n \to \infty$,

$$\textstyle\sum_{k=0}^{\infty} r^k = \frac{1}{1-r}$$

is used.

The Case of Several Variables

Same contraction principles can be applied in higher dimensions as stated in Proposition 2.5 [30]. Euclidean distance is used instead of absolute value in this case.

Proposition 2.5: (Contraction Principle) Let $X \subset \mathbb{R}^n$ be closed and $f : X \to X$ a λ-contraction. Then f has a unique fixed point x_0 and $d(f^n(x), x_0) = \lambda^n d(x, x_0)$ for every $x \in X$ [30].

Proof: Iterating $d(f(x), f(y)) \leq \lambda d(x, y)$ shows

$$(1) \qquad d(f^n(x), f^n(y)) \leq \lambda^n d(x, y)$$

for $x, y \in X$ and $n \in \mathbb{N}$. Thus, $(f^n(x))_n$ is a Cauchy sequence, because

$$(2) \qquad d(f^m(x), f^n(x)) \leq \textstyle\sum_{k=0}^{m-n-1} d(f^{n+k+1}(x), f^{n+k}(x))$$

$$\leq \textstyle\sum_{k=0}^{m-n-1} \lambda^{n+k} d(f(x), x) \leq \frac{\lambda^n}{1-\lambda} d(f(x), x)$$

for $m \geq n$, $\lambda^n \to 0$ as $n \to \infty$. Thus, $\lim_{n\to\infty} f^n(x)$ exists (because Cauchy sequences in $\mathbb{R}^n$ converges) and is in X because X is closed. By (1) it is the same for all x. If x_0 denotes this limit, then

$$d(x_0, f(x_0)) \leq d(x_0, f^n(x)) + d(f^n(x), f^{n+1}(x)) + d(f^{n+1}(x), f(x_0))$$

$$\leq (1 + \lambda) d(x_0, f^n(x)) + \lambda^n d(x, f(x))$$

for $x \in X$ and $n \in \mathbb{N}$. Now $f(x_0) = x_0$ because $d(x_0, f^n(x)) \to 0$ $(n \to \infty)$.

The Derivative Test

Here contraction property is verified for the case of several variables [30], using the derivative.

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a map with continuous partial derivatives. Then, at each point one can define the derivative or differential of $f = (f_1, f_2, ..., f_n)$ as the linear map defined by the matrix of partial derivatives

$$Df = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x2} & . \ . \ . & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x2} & . \ . \ . & \frac{\partial f_2}{\partial x_n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & . \ . \ . & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Norm of the differential is the norm of the matrix Df.

Given a square matrix A, a matrix norm $||A||$ is a nonnegative number associated with A having the properties [28, 72]

1. $||A|| > 0$ when $A \neq 0$ and $||A|| = 0$ if and only if $A = 0$,

2. $||kA|| = |k|||A||$ for any scalar k,

3. $||A + B|| \leq ||A|| + ||B||$,

4. $||AB|| \leq ||A||||B||$.

Spectral norm is the natural norm induced by the L$_2$-norm [28, 67, 72]. Let A† be the adjoint of the square matrix A, so that A$^\dagger = a^*_{ji}$, then

$$||A||_2 = (\text{maximum eigenvalue of } \mathrm{A}^\dagger \mathrm{A})^{0.5}$$

$$= \max_{||x||_2 \neq 0} \frac{||A||_2}{||x||_2}$$

The definition of convex set [14, 72] is important because it is needed in the derivative test.

A set in Euclidean space R^d is convex if it contains all the line segments connecting any pair of its points. If the set does not contain all the line segments, it is called concave as seen in Figure 2.1.

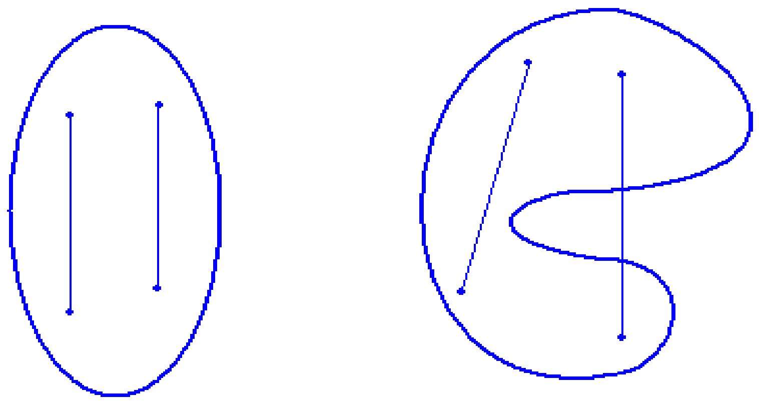


Figure 2.1: A convex (left) and a concave (right) set.

Finally, derivative test [30] can be defined for several variables. Since Lemma 2.1 [30] is used in Theorem 2.1 and Theorem 2.2 it is presented firstly.

Lemma 2.1: If $g : [a, b] \to \mathbb{R}^m$ is continuous and differentiable on (a, b), then there exists $t \in [a, b]$ such that

$$||g(b) - g(a)|| \leq ||\tfrac{d}{dt} g(t)||(b - a).$$

Proof: Let $v := g(b) - g(a)$, $\phi(t) := v(v, g(t))$. By Mean Value Theorem for one variable there exists a $t \in (a,b)$ such that $\phi(b) - \phi(a) = \phi'(t)(b-a)$, and so

$$(b-a)||v||||\tfrac{d}{dt}g(t)|| \geq (b-a)(v, \tfrac{d}{dt}g(t)) = \tfrac{d}{dt}\phi(t)(b-a) = \phi(b) - \phi(a)$$

$$= (v, g(b)) - (v, g(a)) = (v, v) = ||v||^2$$

divide by $||v||$ to finish the proof.

Theorem 2.1: If $C \subset \mathbb{R}^n$ is convex and open and $f : C \to \mathbb{R}^m$ is differentiable with $||Df(x)|| \leq M$ for all $x \in C$, then $||f(x) - f(y)|| \leq M||x - y||$ for $x, y \in C$.

Proof: The line segment connecting x and y is given by $c(t) = x + t(y - x)$ for $t \in [0, 1]$, and it is contained in C by convexity. Let $g(t) = f(c(t))$. Then by the chain rule

$$||\tfrac{d}{dt}g(t)|| = ||Df(c(t))\tfrac{d}{dt}c(t)|| = ||Df(c(t))(y-x)|| \leq M||y-x||.$$

By Lemma 2.1, this implies $||f(y) - f(x)|| = ||g(1) - g(0)|| \leq M||y - x||$.

Corollary 2.1: If $C \subset \mathbb{R}^n$ is a convex open set, $f : C \to C$ a map with continuous partial derivatives, and $||Df(x)|| \leq \lambda < 1$ at every point $x \in \mathbb{R}^n$, then f is a $\lambda-$contraction [30].

An open set may not contain the limits of Cauchy sequences in it. Therefore, the following theorem [30] considers the closure of a set.

Theorem 2.2: If $C \subset \mathbb{R}^n$ is an open strictly convex set, $\overline{C}$ its closure, $f : \overline{C} \to R^n$ differentiable on C and continuous on $\overline{C}$ with $||Df|| \leq \lambda < 1$ on C, then f has a unique fixed point $x_0 \in \overline{C}$ and $d(f^n(x), x_0) \leq \lambda^n d(x, x_0)$ for every $x \in \overline{C}$.

Proof: For $x, y \in \overline{C}$ we parameterize the line segment connecting x and y by $c(t) = x + t(y - x)$ for $t \in [0, 1]$ and let $g(t) = f(c(t))$. Then $c((0, 1))$ is contained in C by strict convexity and

$$||\tfrac{d}{dt}g(t)|| = ||Df(c(t))\tfrac{d}{dt}c(t)|| = ||Df(c(t))(y-x)|| \leq \lambda||y-x||.$$

By Lemma 2.1 this implies $||f(y) - f(x)|| \leq \lambda||y - x||$. Thus, f is a $\lambda-$contraction and has a unique fixed point x_0. Furthermore, $d(f^n(x), x_0) = \lambda^n d(x, x_0)$ for every $x \in \overline{C}$.

2.1.3 Stability of Fixed Points of Maps

The stability of fixed points of one - dimensional maps is mentioned here. The idea is also applicable for maps in $\mathbb{R}^n$.

The idea of stability of fixed points of one-dimensional maps can be described as in [5]. Assuming that the discrete-time system exists to model real phenomena, not all fixed points are alike. A stable fixed point has the property that points near it are moved even closer to the fixed point under the dynamical system. For an unstable fixed point, nearby points move away as time progresses. A good analogy is that a ball at the bottom of a valley is stable, while a ball balanced at the tip of a mountain is unstable.

The question of stability is significant because a real-world system is constantly subject to small perturbations. Therefore, a steady state observed in a realistic system must correspond to a stable fixed point. If the fixed point is unstable, small errors or perturbations in the state would cause the orbit to move away from the fixed point, which would then not be observed.

The derivative of the map at a fixed point p is a measure of how the distance between p and a nearby point is magnified or shrunk by f. The concept of "near" is made precise by referring to all real numbers within a distance ϵ of p as the epsilon neighborhood $N_\epsilon(p)$. Denote the real line by $\mathbb{R}$. Then $N_\epsilon(p)$ is the interval of numbers $\{x \in\mathbb{R}: |x-p| < \epsilon\}$.

Definition 2.5: Let f be a map on $\mathbb{R}$ and let ρ be a real number such that f(ρ) = ρ. If all points sufficiently close to ρ are attracted to ρ, then ρ is called a *sink* or an *attracting fixed point*. More precisely, if there is an ϵ >0 such that for all x in the epsilon neighborhood $N_\epsilon(\rho)$, $\lim_{k\to\infty} f^k(x) = \rho$, then ρ is a sink. If all points sufficiently close to p are repelled from ρ, then ρ is called a *source* or a *repelling fixed point*. More precisely, if there is an epsilon neighborhood $N_\epsilon(\rho)$ such that each x in $N_\epsilon(\rho)$ except for ρ itself eventually maps outside of $N_\epsilon(\rho)$, then ρ is a source.

Theorem 2.3: Let f be a (smooth) map on $\mathbb{R}$, and assume that p is a fixed point of f.

1. If $|f'(\rho)| < 1$, then ρ is a sink.

2. If $|f'(\rho)| > 1$, then ρ is a source.

Proof: PART 1. Let a be any number between $|f'(\rho)|$ and 1; for example, a could be chosen to be $(1 + |f'(\rho)|)/2$. Since

$\lim_{x\to\rho} \frac{|f(x)-f(\rho)|}{|x-\rho|} = |f'(\rho)|$,

there is a neighborhood $N_\epsilon(\rho)$ for some ϵ >0 so that

$\frac{|f(x)-f(\rho)|}{|x-\rho|}$ <a

for x in N$_\epsilon(\rho)$, $x\neq \rho$.

In other words, $f(x)$ is closer to ρ than x is, by at least a factor of a (which is less than 1). This implies two things: First, if $x \in N_\epsilon(\rho)$, then $f(x) \in N_\epsilon(\rho)$; that means that if x is within of ρ, then so is $f(x)$, and by repeating the argument, so are $f^2(x)$, $f^3(x)$, and so forth. Second, it follows that

$|f^k(x)$ - $p| \leq$ a$^k|x - p|$

for all $k \geq 1$. Thus ρ is a sink.

2.1.4 Dynamical System Classification According to Resolution

Dynamical systems can be classified based on the resolution of their state variables and the resolution of their time sets over which the state evolves as John Lygeros mentioned [42].

Based on type of their states, dynamical systems can be classified into:

1. Continuous: if the state take values in Euclidean space $\mathbb{R}^n$ for some $n \geq 1$. $x \in \mathbb{R}^n$ is used to denote the state of a continuous dynamical system.

2. Discrete: if the state takes values in a countable or finite set $\{q_1, q_2, ...\}$. q is used to denote the state of a discrete system. For example, a light switch is a dynamical system whose state takes on two values, $q \in \{ON, OFF\}$. A computer is also a dynamical system whose state takes on a finite (albeit very large) number of values.

3. Hybrid: if some of the state variables take values from $\mathbb{R}^n$ while some other state variables take values from a finite set. For example, the closed loop system we obtain when we use a computer to control an inverted pendulum is hybrid: part of the state (namely the state of the pendulum) is continuous, while another part (namely the state of the computer) is discrete.

Based on the set of times over which the state evolves, dynamical systems can be classified as:

1. Continuous time: if the set of times is a subset of the real line. The symbol $t \in \mathbb{R}$ is used to denote continuous time. Typically, the evolution of the state of a continuous time system is described by an ordinary differential equation (ODE). For example, continuous time population dynamics explained in "A Dynamical View of the World" section

$$\tfrac{dx}{dt} = kx.$$

2. Discrete time: if the set of times is a subset of the integers than $k \in \mathbb{Z}$, is used to denote discrete time. Typically, the evolution of the state of a discrete time system is described by a difference equation. For example, discrete time population dynamics explained in "A Dynamical View of the World" section

$$x_{k+1} = kx_k.$$

3. Hybrid time: if the evolution is over continuous time but there are also discrete "instants" where something "special" like state transitions or jumps happens.

2.1.5 Examples of Dynamical Systems

Given examples based on classification of dynamical systems according to resolution will be beneficial to clarify the subject. Pendulum, manufacturing machine, and thermostat are good examples, as John Lygeros mentioned [42].

A Continuous System: (Pendulum)

Consider the pendulum given in Figure 2.2. Here, l is the length of weightless solid rod, m is mass of the ball, θ denotes angle the pendulum makes with the downward vertical.

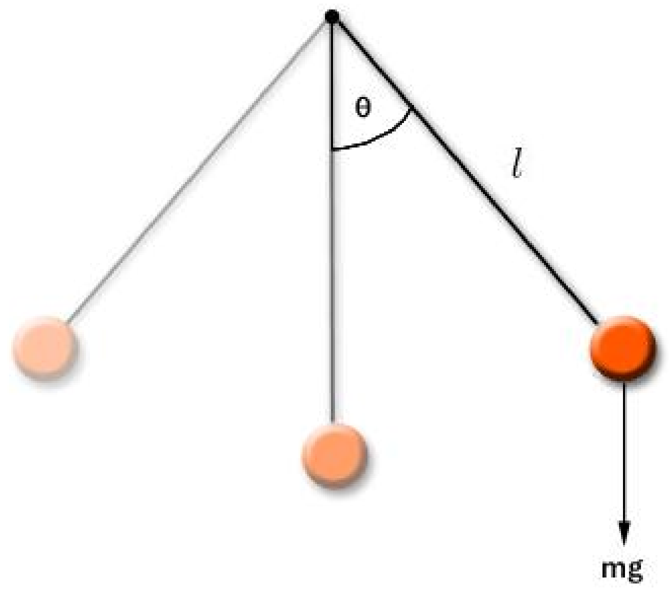


Figure 2.2: The pendulum.

The evolution of θ is governed by

$$ml\frac{d^2\theta}{dt^2} + dl\frac{d\theta}{dt} + mg\sin(\theta) = 0,$$

where d is the dissipation constant.

Finding θ as a function of time will explain the dynamics of the pendulum. This means finding the solution of ODE. At $t = 0$ pendulum starts with initial position θ_0 and initial velocity $\frac{d\theta}{dt}$.

This function must satisfy

$$\theta(0) = \theta_0,$$

$$\frac{d\theta(0)}{dt} = \frac{d\theta_0}{dt},$$

$$ml\frac{d^2\theta(t)}{dt^2} + dl\frac{d\theta(t)}{dt} + mg\sin(\theta(t)) = 0, \text{ for all } t \in \mathbb{R}$$

Such a function is known as a trajectory of the system.

To simplify the notation typically, ODE is written is state space form

$$\frac{dx}{dt} = f(x)$$

where

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \frac{d\theta}{dt} \end{bmatrix},$$

which gives rise to the state space equation

$$\frac{dx}{dt} = \begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{g}{l}\sin(x_1) - \frac{d}{m}x_2 \end{bmatrix} = f(x).$$

The vector $x \in \mathbb{R}^2$ is the state of the system.

Solving the ODE for θ is equivalent to finding a function

$$x(0) = \begin{bmatrix} x_1(0) \\ x_2(0) \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \frac{d\theta_0}{dt} \end{bmatrix}$$

$$\frac{dx(t)}{dt} = f(x(t)), \text{ for all } t \in \mathbb{R}$$

Figure 2.3 shows the trajectory of the system for $l = 1$, $m = 1$, $d = 1$, $g = 9.8$, $\theta(0) = 0.75$ and $\frac{d\theta}{dt}(0) = 0$.

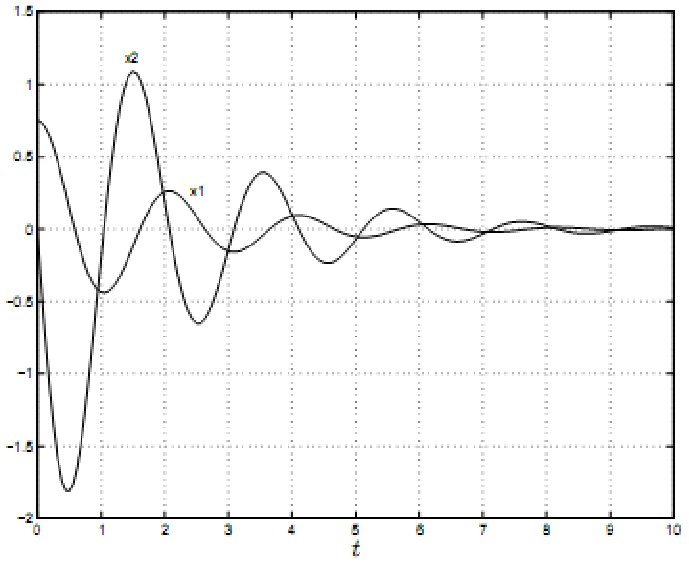


Figure 2.3: Trajectory of the pendulum.

In order to visualize the solutions of pendulum system phase plane plots (plots of $x_1(t)$ vs $x_2(t)$) are convenient. Figure 2.4 shows the phase plan plot of the trajectory of Figure 2.3.

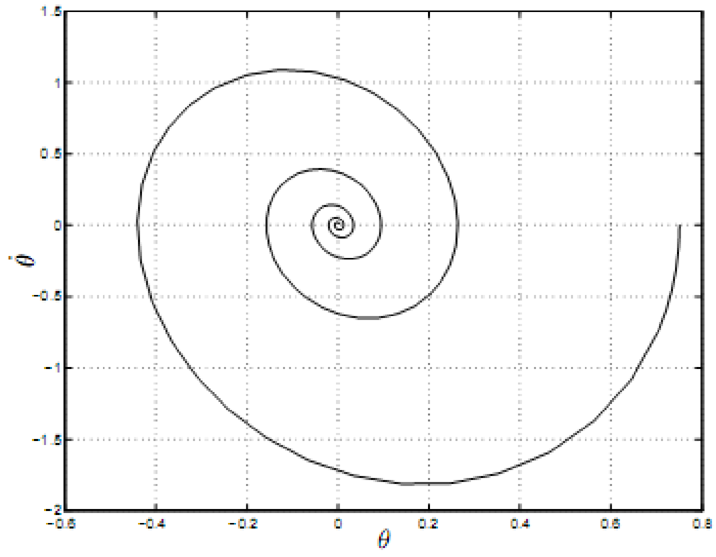


Figure 2.4: Phase plane plot of the trajectory of Figure 2.3.

A Discrete System: (Manufacturing Machine)

A manufacturing machine is defined as a machine processing parts of type p one at a time.

This machine can be in one of three states: Idle (I), working (W), and down (D). Additionally, depending on some certain events machine can transition between these states.

If this machine is modelled as a dynamical system it will have three discrete states

$$q \in Q = \{I, W, D\}$$

and some certain events which cause state transitions

$$\sigma \in \Sigma = \{p, c, f, r\},$$

where

$p =$ a part in any type arrives,

$c =$ complete processing,

$f =$ failure,

$r =$ repair.

The state after event occurs is given by a state transition relation

$$\partial : Q \times \Sigma \rightarrow Q$$

For example, if the machine is idle state and part p is arrived, then it turns to working state.

$$\partial(I, p) = W$$

A Hybrid System: (Thermostat)

Consider a thermostat controlled radiator heats a room as seen in Figure 2.5.

The radiator can be in one of two states: ON, OFF. If the radiator is in the OFF state, the temperature of the room decreases exponentially towards 0 degree (0 degree is assumed as environmental temperature) according to the differential equation

$$\tfrac{dT}{dt} = -aT$$

for some $a > 0$.

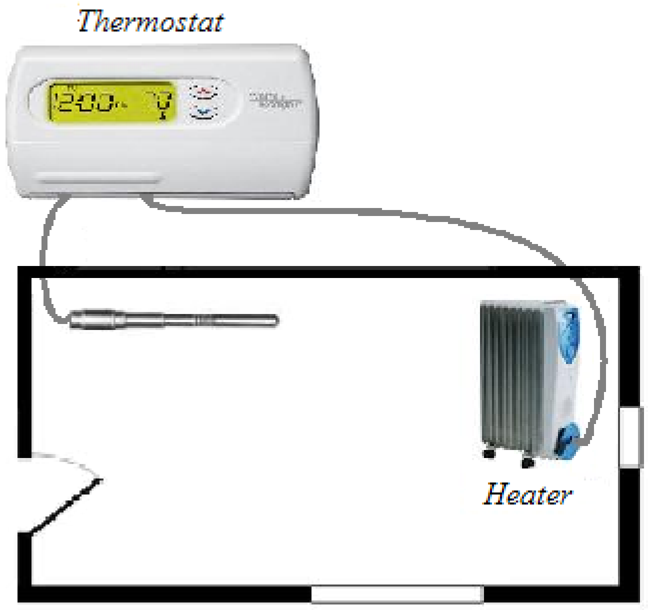


Figure 2.5: Thermostat controlled heater.

When the thermostat turns the hater on a state transition occurs from OFF to ON state. In ON state, temperature of the room increase exponentially towards 30 degrees according to the differential equation

$$\frac{dT}{dt} = -a(T - 30).$$

Assume that the aim is to keep the room temperature at around 20 degrees. Then, the discrete states are

$$q \in Q = \{ON, OFF\}$$

and the events are

$$\sigma \in \Sigma = \{T \leq 19, T \geq 21\}$$

A trajectory of thermostat system is seen in Figure 2.6.

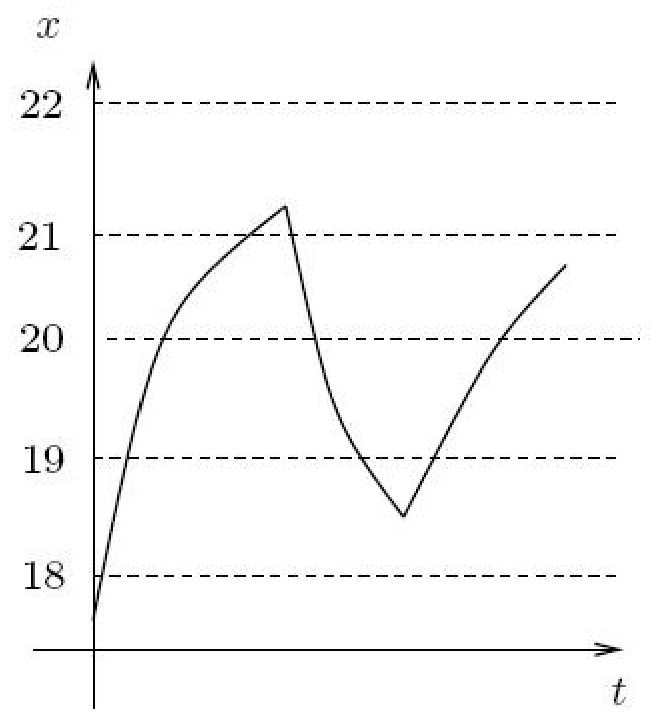


Figure 2.6: A trajectory of the thermostat system.

Thermostat system has both continuous and discrete state.

2.2 Hybrid Dynamical Systems

In this section, hybrid dynamical systems and related topics are covered at an introductory level. The basics of graph theory are presented first, then information related to automata is given. After piecewise linear systems are introduced an example related to hybrid dynamical systems is included. Finally, the graph representation of hybrid dynamical systems is mentioned by water tank example.

2.2.1 Basics of Graph Theory

A hybrid dynamical system can be seen as a graph whose nodes are states and edges are possible state transitions between states. Therefore, it will be beneficial to start with graph theory and present the basics.

A graph is a collection of vertices, certain unordered pairs of which are called its edges [73]. A graph is described by its vertices and by edges between vertices. A sample graph and its vertices and edges are seen in Figure 2.7.

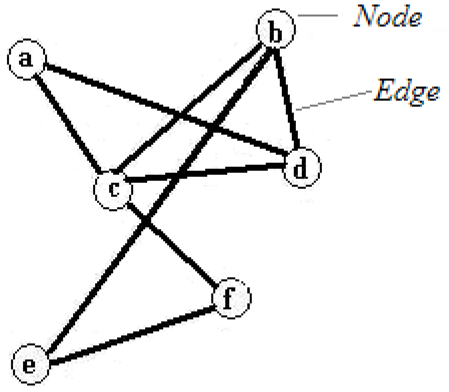


Figure 2.7: A simple graph.

A graph can be represented as a pair $G = (V, E)$ of sets satisfying $E\subseteq[\mathrm{V}]^2$ where V is the set of vertices; thus, the elements of E are 2-element subsets of V [15]. By this representation V represents the vertices of graph and E represents the edges of the graph. For example, if the graph in Figure 2.7 is considered,

$V = \{a, b, c, d, e, f\}$ and $E = \{\{a, c\}, \{a, d\}, ...\}$.

The degree of a graph vertex v of a graph G is the number of graph edges which contain v [64]. A vertex of degree 0 is isolated. $d(V)$ represents the degree of the vertex v. We denote $\delta(G) := \min\{d(V)|v \in V\}$ and $\Delta(G) := \max\{d(V)|v \in V\}$.

A path in a graph is a non-empty graph $P = (V, E)$ of the form

$$V = \{x_0, x_1, ..., x_k\} \quad , E = \{x_0x_1, x_1x_2, x_2x_3, ..., x_{k-1}x_k\},$$

where the x_i are all distinct [15]. A sample path of the graph presented in Figure 2.7 is seen in Figure 2.8.

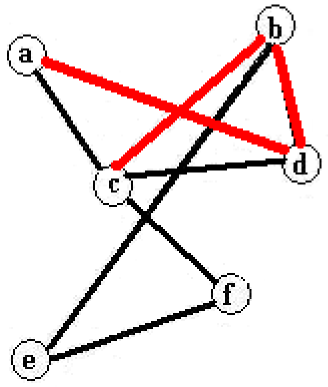


Figure 2.8: A sample path.

If two ends of a path are connected this is called a cycle (if $P = x_0...x_{k-1}$ is a path and $k \geq 3$, then the graph $C = P + x_{k-1}x_0$ is a cycle [15]). It will useful in later of this work to introduce the Proposition 2.6 [15].

Proposition 2.6: Every graph G conforming $\delta(G) \geq 2$ contains a path of length $\delta(G)$ and a cycle of length at least $\delta(G) + 1$.

Proof: Let $x_0 ... x_k$ be a longest path in G. Then, all the neighbors of x_k lie on this path. Hence, $k \geq d(x_k) \geq \delta(G)$. If $i < k$ is minimal with $x_i x_k \in E(G)$, then $x_i ... x_k x_i$ is a cycle of length at least $\delta(G) + 1$.

A directed graph is a graph $G = (V, E)$ consists of a set of vertices V and a set of edges E that are ordered pairs of elements of V [58]. Additionally, a directed multigraph is a graph $G = (V, E)$ consists of a set of vertices V and a set of edges E, and a function f from $\{(u, v) | u, v \in V\}$ to E. The edge e_1 and e_2 are multiple edges if $f(e_1) = f(e_2)$. Figure 2.9 gives examples, to a directed graph and to a directed multigraph.

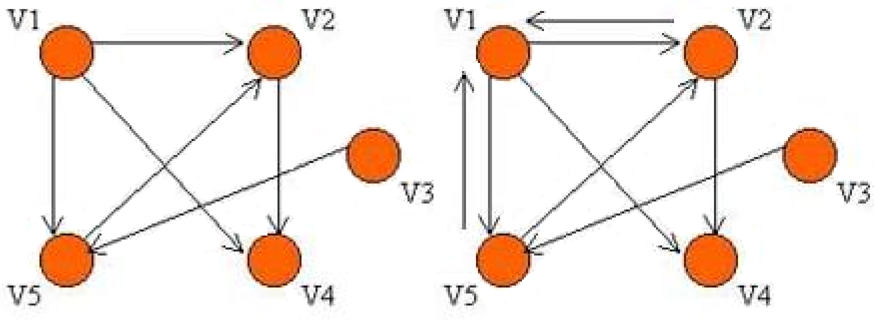


Figure 2.9: Directed graph (left) and directed multigraph (right).

2.2.2 Discrete Event Systems (Automata)

Discrete event systems (DES) are important to manage modern world, most of which are man-made systems such as multicomputer systems, communication networks, traffic systems and manufacturing systems. In this section, regular languages and regular expressions, graph representation of regular expressions, finite automaton, and hybrid automaton are presented.

Regular Languages and Regular Expressions

A string which is a finite set of finite sequence of symbols is the basic object in automata and language theory [16]. The length of a string is denoted by $|x|$, and means the number of symbols contained in string. For example, $|abcd| = 4$. The empty string is denoted by ϵ and $|\epsilon| = 0$. The basic operation on strings is concatenation (”.”) which means joining two strings. To illustrate, $x.y = xy$, x string and y string are concatenated to form the xy string. Additionally, since languages are sets, usual set operations like union, intersection, difference and complement are applicable to languages. Limited set of symbols used to form strings is called an alphabet. If Σ represents an alphabet Σ^* denotes the set of all possible strings over Σ. A language is a set of strings and a language L over Σ is just a subset of Σ^*.

The concept of regular languages (or, regular sets) over an alphabet Σ is defined recursively as follows [16]:

(1) The empty set is a regular language.

(2) For every symbol $a \in \Sigma$, $\{a\}$ is a regular language.

(3) If A and B are regular languages, then A, AB, and A^* are all regular languages.

(4) Nothing else is a regular language.

For example, the set $\{10, 01\}$ is a regular language over the binary alphabet: $\{10, 01\} = (\{1\}\{0\}) \cup (\{0\}\{1\})$.

To simplify the representations for regular languages, the notion of regular expressions over alphabet C is defined [16] as follows:

(1) $\emptyset$ is a regular expression which represents the empty set.

(2) ϵ is a regular expression which represents language $\{\epsilon\}$.

(3) For $a \in \Sigma$, a is a regular expression which represents language $\{a\}$.

(4) If r_A and r_B are regular expressions representing languages A and B, respectively, then $(r_A) + (r_B), (r_A)(r_B)$, and $(r_A)^*$ are regular expressions representing A, AB, and A^*, respectively.

(5) Nothing else is a regular expression over Σ.

Graph Representation for Regular Expressions

There is an interesting way to represent regular expressions by labelled digraphs with labels from $\Sigma \cup \{\epsilon\}$. For any regular expression r, its labelled digraph representation can be obtained [16] as follows:

1. Initially, we start with two special vertices, the initial vertex and the final vertex, and draw an edge between them with label r (see Figure 2.10(l)). (In Figure 2.10(l), an arrow with no starting point to the initial vertex is drawn and double circles are used to denote the final vertex.)

2. Repeat the following until every edge has a label that does not contain oper-

ation symbols $+, ., $ or *:

Replace each edge with label $f + g$ by two edges with labels f and g, as shown in Figure 2.10(2).

Replace each edge with label fg by an additional vertex and two edges with labels f and g, as shown in Figure 2.10(3).

Replace each edge with label f^* $(f.g)$ by an additional vertex and three edges with labels ϵ, f, and ϵ, as shown in Figure 2.10(4).

3. Delete all edges with label $\emptyset$.

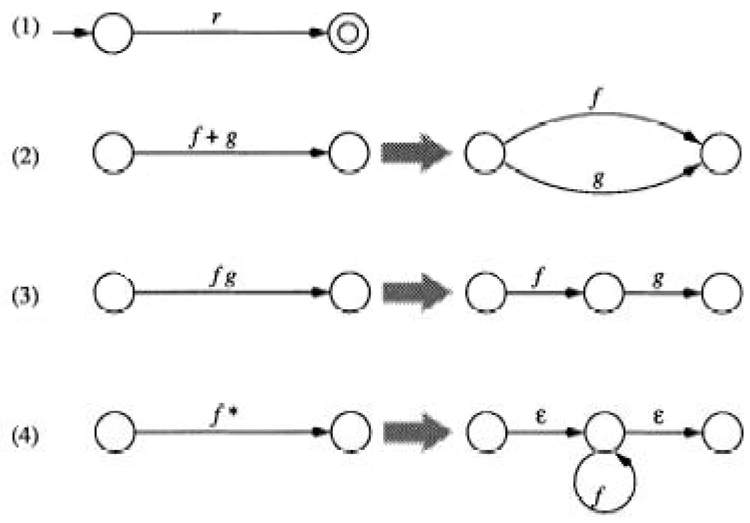


Figure 2.10: Graph $G(r)$ for regular expression r.

For a regular expression r, let $G(r)$ be its graph representation constructed as above. Clearly, each edge in $G(r)$ has a label in $\Sigma \cup \{\epsilon\}$. Every path in $G(r)$ is associated with a string which is obtained by concatenating all symbols labelling the edges in the path. This representation has the following property [16]:

Theorem 2.4: Let r be a regular expression. A string x belongs to the language $L(r)$ if and only if there is a path in $G(r)$ from the initial vertex to the final vertex whose associated string is x.

Proof: Let v_1 be the initial vertex and q be the final vertex in $G(r)$. Consider the following statement:

S: $x \in L(r)$ if and only if there is a path $(v_1, v_2, ..., v_k = v_f)$ in digraph G such that $x \in L(r_1)L(r_2)...L(r_{k-1})$, where r_i is the label of the edge $(v_i, v_{i+1}), i = 1, ..., k-1$.

It is claimed that statement S holds with respect to the graph G at any stage of the above construction of $G(r)$. First, it is clear that S holds with respect to the graph G at the beginning of the step 2. Next, we observe that, because of the way the edges are replaced, if S holds with respect to a graph G, then it still holds after an edge replacement performed at step 2. Thus, the statement S holds with respect to the graph G at the end of step 2. At step 3, we delete edges with the label 0, and this does not affect statement S, since $L(\emptyset) = \emptyset$. Therefore, at the end of step 3, each edge of $G(r)$ is labelled by exactly one symbol from $\Sigma \cup \{\epsilon\}$, and statement S implies that $x \in L(r)$ if and only if there is a path in $G(r)$ from v_1 to v_f whose associated string is exactly x.

Finite Automaton

Automata theory is important for implementing systems with a fixed set of resources. If we use finite number of states for a system, the entire history of the system could not be remembered but it will be designed carefully to represent the important states of the system. Therefore, systems could be implemented as circuits or as a program. For example, the simplest system on/off switch is nontrivial finite automaton [16].

A finite automaton (or FA) can be formally defined as a 5-tuple (Q, Σ, T, q_0, T) [16], where

• Q is a finite set of states,

• Σ is the alphabet (defining what set of input strings the automaton operates on),

• $T : Q \times \Sigma \rightarrow P(Q)$ is the transition function,

• $q_0 \in Q$ is the starting state,

• $F \subset Q$ and is a set of final (or accepting) states.

Operation of the FA begins at q_0 , and movement from state to state is governed by the transition function T.

FAs represent regular languages and a FA can be represented visually by using directed graphs.

Hybrid Automaton

One of the most common forms of representing a hybrid system is the hybrid automaton [9, 41, 56]:

Definition 2.6: (Hybrid Automaton) A hybrid automaton H is a collection $H = (Q, X, f, Init, D, E, G, R)$ where,

- Q is a finite set of discrete variables.
- $X \subseteq \mathbb{R}^n$.
- X is a finite dimensional set of continuous variables.
- $f : Q \times X \rightarrow TX$ is a vector field.
- $Init \subseteq Q \times X$ is a set of initial states.
- $D : Q \rightarrow P(X)$ is a domain.
- $E \subseteq Q \times Q$ subset of set of edges over Q.
- $G : E \rightarrow P(X)$ is the guard condition.
- $R : E \times X \rightarrow P(X)$ is the reset map.

Here, Q denotes the set of al possible valuations of $q \in Q$, X denotes a smooth manifold for X, TX denotes the tangent bundle of X and $P(X)$ is the power set of X.

$(q, x_q) \in Q \times X$ is referred to as the state h of the hybrid automaton H.

2.2.3 Piecewise Linear Systems

In this section, piecewise linearity is introduced and switching differential equations are presented.

Piecewise Linearity

Let the state space $U \subseteq \mathbb{R}^p$ of a dynamical system is formed by z disjoint subspaces such that [52]

$U = U1 \cup U2 \cup ... \cup Uz$ and

$U_i \cap U_j = \emptyset$ if $i \neq j$.

Consider three different initial states which are elements of the same subspace U_i. This means

$y_0 \in U_i, y_1 \in U_i, y_2 \in U_i,$

which satisfies the following equality:

$y_2 - y_0 = r(y_1 - y_0) \qquad (r \in \mathbb{R}).$

Let y_0, y_1, y_2 yield $y_0(t)$, $y_1(t)$, $y_2(t)$, respectively. This means: If the system started with the initial state $y(t_0) = y_0$ then the function representing its temporal evolution for $t > t_0$ is denoted by $y_0(t)$.

The system is piecewise linear in U_i if

$y_2[t_0, t_i] - y_0[t_0, t_i] = r(y_1[t_0, t_i] - y_0[t_0, t_i]) \qquad (r \in \mathbb{R}),$

and $y_0(t)$, $y_1(t)$, $y_2(t) \in U_i$ for all $t_0 < t < t_i$ where $[t_0, t_i]$ stands for the time interval that is spent in subspace U_i. Herewith, the system itself is said to be piecewise linear if it is piecewise linear in all subspaces of its state space [52].

Switching Differential Equations

A piecewise linear system can be represented by the switching differential equations as follows [52]:

$\frac{dy}{dt} = M_{s(t)}y(t) + N_{s(t)}x_e(t) + k_{s(t)}$

$s(t) = s_i$ if $y(t) \in U_i$,

where

$y(t) \in \mathbb{R}^n$ is a column vector representing the continuous variables,

$x_e \in \mathbb{R}^n$ is a column vector representing the external inputs,

$s(t) \in (1, 2, ..., p)$ is a symbolic variable representing the state of the system,

$M : s \longrightarrow \mathbb{R}^{n \times n}$ is a switching matrix whose elements are determined by the state of the system,

$N : s \longrightarrow \mathbb{R}^{n \times n}$ is a switching matrix whose elements are determined by the state of the system

$k : s \longrightarrow \mathbb{R}^n$ is a switching vector whose elements are determined by the state of the system and

$U_i \subset \mathbb{R}^n$ is a subspace of the systems state space.

Subscript i denotes ith element of the corresponding vector [52].

The following simplified switching structure is mostly considered in this work:

$\frac{dy}{dt} = M_{s(t)} y(t) + k_{s(t)}$,

$s_i(t) = F(Q([y_1(t), y_2(t), ..., y_n(t)]))$,

$$Q_i(y(t)) = \{ \begin{array}{l} 1 \text{ if } y_i(t) > h_i \\ 0 \text{ if } y_i(t) \leq h_i \end{array},$$

where

$s(t) \in (0,1)^n$ is a column vector representing the states of the variables,

$M : s^n \rightarrow \mathbb{R}^{n \times n}$ is a switching matrix,

$k : s^n \rightarrow \mathbb{R}^n$ is a switching vector whose elements are determined by the states of the variables,

$F : (0,1)^n \rightarrow (0,1)^n$ is a Boolean function [52].

2.2.4 Example

For illustration, following example is presented:

$\frac{dy}{dt} = M_{s(t)} x(t) + k_{s(t)}$,

$s_i(t) = F(Q([y_1(t), y_2(t)]))$,

$$Q_i(y(t)) = \{ \begin{array}{l} 1 \text{ if } x_i(t) > h_i \\ 0 \text{ if } x_i(t) \leq h_i \end{array},$$

$$M_{0,0} = M_{0,1} = M_{1,0} = M_{1,1} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$k_{0,0} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, k_{0,1} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, k_{1,0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, k_{1,1} = \begin{bmatrix} 2 \\ 0 \end{bmatrix},$$

$$h_1 = h_2 = 1.$$

The threshold divide the system into 4 different regions as seen in Figure 2.11.

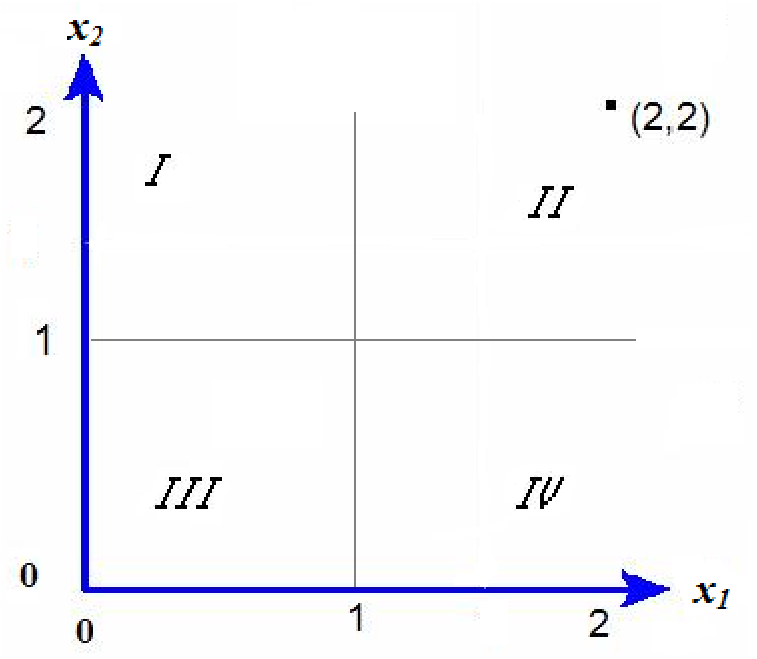


Figure 2.11: State space, thresholds of the example.

If the initial state is in the region IV ($x_1(t) > 1$, $x_2(t) \leq 1$), the governing differential equation is:

$$\frac{dy}{dt} = M_{1,0}y(t) + k_{1,0},$$

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

whose solution is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1(0)e^{-t} \\ x_2(0)e^{-t} \end{bmatrix}.$$

All points in region IV will exponentially converge to $(0, 0)$.

If the initial state is in the region II ($x_1(t) > 1,\ x_2(t) > 1$) the governing differential equation is:

$$\tfrac{dx}{dt} = M_{1,1}x(t) + k_{1,1},$$

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix},$$

whose solution is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 - (2 - x_1(0))e^{-t} \\ x_2(0)e^{-t} \end{bmatrix}.$$

All points in region II will exponentially converge to $(2, 0)$.

If the initial state is in the region I ($x_1(t) \leq 1,\ x_2(t) > 1$) the governing differential equation is:

$$\tfrac{dx}{dt} = M_{0,1}x(t) + k_{0,1},$$

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix},$$

whose solution is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 - (2 - x_1(0))e^{-t} \\ 2 - (2 - x_2(0))e^{-t} \end{bmatrix}.$$

All points in region I will exponentially converge to $(2, 2)$.

If the initial state is in the region III ($x_1(t) \leq 1$, $x_2(t) \leq 1$) the governing differential equation is:

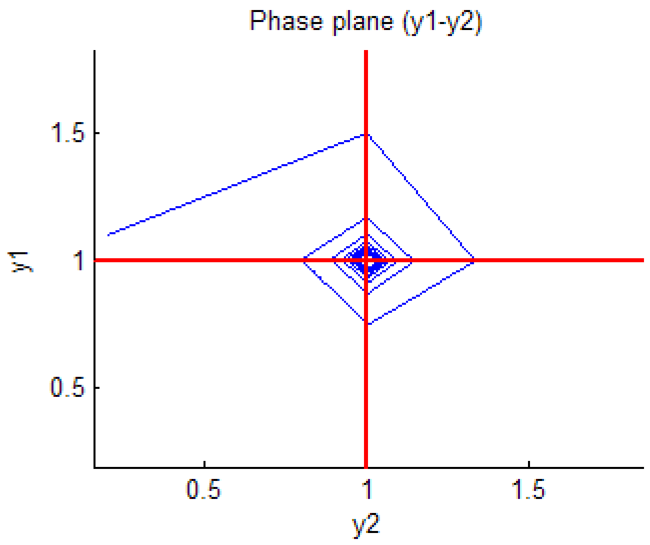


Figure 2.12: The behavior of the system for initial point $y_0 = [1.1\ 0.2]'$.

$$\frac{dx}{dt} = M_{0,0}x(t) + k_{0,0},$$

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix},$$

whose solution is

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1(0)e^{-t} \\ 2 - (2 - x_2(0))e^{-t} \end{bmatrix}.$$

All points in region III will exponentially converge to $(0, 2)$.

The system is numerically simulated. Here, the dynamics of the system for two initial points is presented. If the initial point is $y_0 = \begin{bmatrix} 1.1 \\ 0.2 \end{bmatrix}$. The behavior of the system is as seen in Figure 2.12 and Figure 2.13.

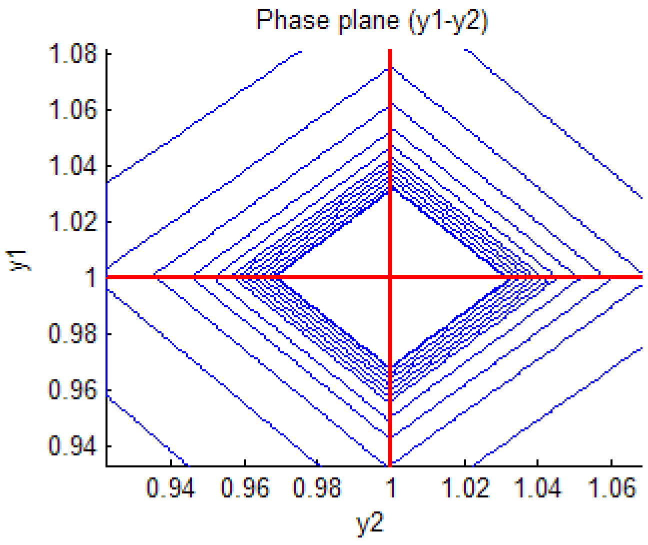


Figure 2.13: A closer look of the Figure 2.12.

However, if the $y_0 = \begin{bmatrix} 1.002 \\ 0.9999 \end{bmatrix}$ is chosen for initial point the behavior observed is as in Figure 2.14.

2.2.5 Graph Representation of HDS

As discussed in "Hybrid Automaton" section directed graphs can be used for representation of HDS.

At the beginning it will be useful to give the definitions of hybrid time trajectory, execution, and zeno hybrid automata based on [34, 40].

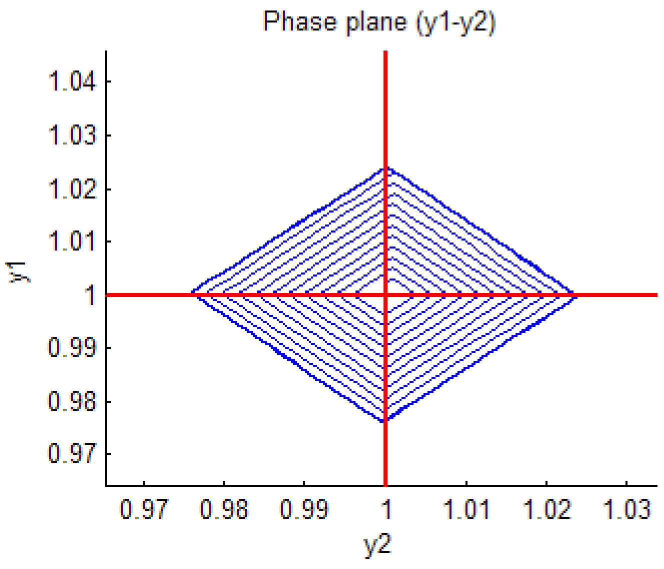


Figure 2.14: The behavior of the system for $y_0 = [1.002\ 0.9999]'$.

A hybrid system will involve continuous evolution as well as instantaneous transitions. To distinguish the times at which discrete transitions take place the notion of a hybrid time trajectory is introduced.

Definition 2.7: (Hybrid time trajectory). A hybrid time trajectory $\tau = \{I_i\}_{i=0}^{N}$ is a finite or infinite sequence of intervals of the real line, such that

for all $0 \leq i < N$, $I_i = [\tau_i, \tau_i']$ with $\tau_i \leq \tau_i' = \tau_{i+1}$;

if $N < \infty$, either $I_N = [\tau_N, \tau_N']$ with $\tau_N \leq \tau_N' < \infty$, or $I_N = [\tau_N, \tau_N')$ with $\tau_N \leq \tau_N' \leq \infty$.

The interpretation is that τ_i are the times at which discrete transitions take place; notice that multiple transitions may take place at the same time (if $\tau_i = \tau_i' = \tau_{i+1}$). Hybrid time trajectories can extend to infinity either if τ is an infinite sequence, or if it is a finite sequence ending with an interval of the form $[\tau_N, \infty)$. Υ denotes the set of all hybrid time trajectories.

Definition 2.8: (Execution). An execution χ of a hybrid automaton H is a collection $\chi = (\tau, q, x)$ with $\tau \in \Upsilon$, $q : \tau \to Q$, and $x : \tau \to X$, satisfying

$(q(\tau_0), x(\tau_0)) \in Init$ (set of possible initial conditions);

for all i such that $\tau_i < \tau_i'$, $x(t)$is continuously differentiable and $q(t)$ is constant for $t \in [\tau_i, \tau_i']$, and $x(t) \in I(q(t))$ and $\frac{dx(t)}{dt} = f(q(t), x(t))$ for all $t \in [\tau_i, \tau_i')$ (continuous evolution); and

for all $i, e = (q(\tau_i'), q(\tau_{i+1})) \in E$, $x(\tau_i') \in G(e)$, and $x(\tau_{i+1}) \in R(e; x(\tau_i'))$ (discrete evolution).

Here, Q and X are defined with ”Hybrid Automaton” Section 2.2.2.

For an execution $\chi = (\tau, q, x)$, $(q_0, x_0) = (q(\tau_0), x(\tau_0))$ is used to denote the initial state of χ.

Unlike conventional continuous dynamical systems, the interpretation is that an automaton H accepts an execution $\chi = (\tau, q, x)$ (as opposed to generates). This conceptual difference allows one to consider hybrid automata that accept no executions for some initial states, accept multiple executions for the same initial states, or do not accept executions over arbitrarily long time horizons. An execution $\chi = (\tau, q, x)$ is called finite if τ is a finite sequence ending with a closed interval, infinite if τ is an infinite sequence or if $\sum_i(\tau_i' - \tau_i) = \infty$, Zeno if it is infinite but $\sum_i(\tau_i' - \tau_i) < \infty$, and maximal if it is not a strict prefix of any other execution of H. For an infinite execution we denote the Zeno time as

$\tau_\infty = \sum_i(\tau_i' - \tau_i)$. Clearly, $\tau_\infty < \infty$ if the execution is Zeno and $\tau_\infty = \infty$ otherwise. $\hat{H}(q_0, x_0)$ is used to denote the set of all executions of H with initial condition $(q_0, x_0) \in Init$, $\hat{H}^M(q_0, x_0)$ to denote the corresponding maximal executions, and $\hat{H}^\infty(q_0, x_0)$ to denote the infinite executions.

Definition 2.9: (Zeno hybrid automaton). A hybrid automaton H is called Zeno if there exists $(q_0, x_0) \in Init$ such that all executions in $\hat{H}^\infty(q_0, x_0)$ are Zeno executions.

In graph representation of hybrid systems, each partition of state space can be represented by a node and possible transitions between them can be represented by edges. Nodes generally represent discrete variables $Q = \{q_1, q_2, ..., q_n\}$ in a hybrid dynamical system. Transitions between states are represented by directed edges. The guard conditions $G(edge_i)$ which cause state transition are generally placed on directed edges. Additionally, reset relations $R(edge_i, x)$ are represented on directed edges. In a node, which represent the state q_i in a hybrid dynamical systems, the governing vector field $f(q_i, x)$ for the state and evolution set $I(q_i)$ are given.

To illustrate the concept water tank system of Alur and Henzinger [34, 6] is presented here. The system and graph representation of the system is given in Figure 2.15 and Figure 2.16.

For $i = 1, 2$, let x_i denote the volume of water in Tank i, and $v_i > 0$ denote the (constant) flow of water out of Tank i. Let the constant flow of water into the system is denoted by w, directed exclusively to either Tank 1 or Tank 2 at each point in time. The objective is to keep the water volumes above r_1 and r_2,

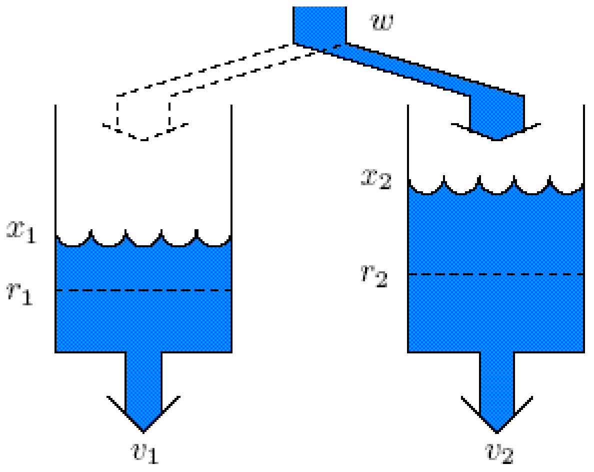


Figure 2.15: The water tank system.

respectively (assuming that $x_1(0) > r_1$ and $x_2(0) > r_2$). This is to be achieved by a switched control strategy that switches the inflow to Tank 1, whenever $x_1 \leq r_1$, and to Tank 2, whenever $x_2 \leq r_2$. More formally, the water tank automaton is a hybrid automaton, denoted by WT, with

$Q = \{q_1, q_2\}$ and $X = R^2$,

$Init = Q \times \{x \in X : (x_1 \geq r_1)\Lambda(x_2 \geq r_2), r_1, r_2 > 0\}$,

$f(q_1, x) = (w - v_1, -v_2)$ and $f(q_2, x) = (-v_1, w - v_2), v_1, v_2 w > 0$,

$I(q_1) = \{x \in X : x_2 \geq r_2\}$ and $I(q_2) = \{x \in X : x_1 \geq r_1\}$,

$E = \{(q_1, q_2), (q_2, q_1)\}$,

$G(q_1, q_2) = \{x \in X : x_2 \leq r_2\}$ and $G(q_2, q_1) = \{x \in X : x_1 \leq r_1\}$ and

$R((q_1, q_2), x) = R((q_2, q_1), x) = \{x\}$.

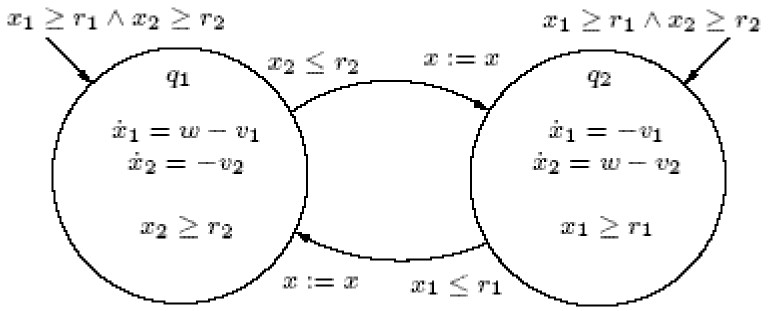


Figure 2.16: Graphical representation of the water tank hybrid system.

It is straightforward to show that WT accepts a unique infinite execution for each initial state [72]. Moreover, if $\max\{v_1, v_2\} < w < v_1 + v_2$, then WT is Zeno with Zeno time $\tau_\infty = (x_1(0) + x_2(0) - r_1 - r_2)/(v_1 + v_2 - w)$, where $(x_1(0), x_2(0))$ is the continuous part of the initial state.

2.3 Hybrid Dynamical Systems with Delay on Piecewise Constant Part

Hybrid dynamical systems with delay can be defined as

$$\frac{dy}{dt} = M_{s(t)}y(t) + N_{s(t)}x_e(t) + k_{s(t)},$$

$$s_i(t) = F_i(Q([y_1(t - \tau_{1i}), y_2(t - \tau_{2i}), ..., y_n(t - \tau_{ni})])),$$

$$Q_j(y(t)) = \{ \begin{array}{c} 1 \text{ if } y_j(t) \geq h_j \\ 0 \text{ if } y_j(t) < h_j \end{array},$$

where, $f_s : Y \times \ X_e \rightarrow \mathbb{R}^n$ is a switching function, $s(t) \in (0,1)^n$ is a column vector representing the state of the system, $F_i : (0,1)^k \rightarrow (0,1)^k$ is a set of Boolean functions for the state variable and $\tau_{ji} \in R$ is the time necessary for the threshold crossing of the continuous variable j to affect the state of the Boolean variable i. Here we consider 2^n states and each variable's crossing of its threshold causes a guard condition. In fact, neither all variables need to cause a guard condition nor a variable need to have a single threshold for using a hybrid system to approximate a dynamical system involving delays. However, this can be adjusted by increasing the number of thresholds and assigning same parameters M_s,N_s and k_s to different states. The system forms a mapping from the combination of initial function space of its states and state space to its future function space, i.e., $H_d : S_{s,t_0-<t<t_0} \times Y \times X \rightarrow S_{y,t>t_0}$. If the system is Zeno free (verifiable), the piecewise constant structure of $s(t)$ allows representing it with finite number of variables. Thus, hybrid systems in the form of H_d offer tractable abstractions of dynamical systems involving delay [52].

CHAPTER 3

ANALYSIS OF PERIODIC SOLUTIONS OF PIECEWISE LINEAR SYSTEMS WITH DELAY ON PIECEWISE CONSTANT PART

In this section periodic solutions of piecewise linear systems with delay on piecewise constant part is presented first. Then, the stability analysis of an example system is studied.

3.1 Periodic Solutions of Piecewise Linear Systems with Delay on Piecewise Constant Part

The problem of the asymptotic representation of solutions of differential equations with impulse action on surfaces, and of systems of differential equations with discontinuous right hand side is investigated in [1]. This is possible by piecewise continuous vector-valued functions as stated in [1]. These results can be applied to piecewise linear systems with delay on piecewise constant part. The problem of the existence of periodic solutions of differential equations with pulse effect on the surfaces and to differential equations with discontinuous right hand sides close to arbitrary nonlinear ones is studied in [2]. In [2] it is seen that, under some certain conditions differential equations with pulse effect on the surfaces and differential equations with discontinuous right hand sides close to arbitrary nonlinear ones admit ω - periodic solutions. Benefitting from this work, the existence of periodic solutions of PLS with delay on piecewise linear part can be studied. Additionally, [24] presents differential equations with discontinuous righthand sides.

In this part of the study, the state transition conditions are studied. Let us consider the following PLS with delay on piecewise constant part,

$$\tfrac{dy}{dt} = M_{s(t)}y(t) + k_{s(t)},$$

$$s(t) = s_i \text{ if } y(t-\tau) \in U_i,$$

where

$y(t) \in \mathbb{R}^n$ is a column vector representing the continuous variables,

$s(t) \in (1, 2, ..., p)$ is a symbolic variable representing the state of the system,

$M : s \longrightarrow \mathbb{R}^{n\times n}$ is a switching matrix whose elements are determined by the state of the system,

$k : s \longrightarrow \mathbb{R}^{n}$ is a switching vector whose elements are determined by the state of the system and

$U_i \subset \mathbb{R}^n$ is a subspace of the systems state space,

τ is the constant delay in state transitions.

At a particular state Q$_j$ the system evolves according to:

$$y(t) = -M_j^{-1}k_j + (y_0 + M_j^{-1}k_j)e^{M_j(t-t_0)} \text{ for } t_0 < t < t_e$$

where -$M_j^{-1}k_j$ is the focal point of the system at the state Q_j, y_0 is the state of the system when it entered into the state Q_j, t_0 time at y_0, t_e is the exit time from Q_j if it exists.

We can say that the system at the state Q_j converges to or diverges from the coordinates of the corresponding focal point along eigenvectors. Where, convergence happens along eigenvectors with negative eigenvalues and divergence happens along eigenvectors with positive eigenvalues.

A linear system $x'(t) = Ax(t) + k$, where A matrix is not in diagonal form but is diagonalizable, can be converted in diagonal form as

$$T^{-1}AT = \Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n),$$

$$z(t) = T^{-1}x(t) \iff x(t) = Tz(t),$$

$$x'(t) = T^{-1}x'(t) = T^{-1}Ax(t) + T^{-1}k = T^{-1}ATz(t) + T^{-1}k,$$

$$z'(t) = \Lambda z(t) + T^{-1}k.$$

For simplicity we considered only diagonal M matrices.

$$M = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & ... & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix}.$$

A hybrid system can be represented as an n-dimensional hypercube. Each axis represents a variable y$_i$ and each node on an axis represents a state Q_j of this variable. A state transition due to the threshold crossing of a particular variable y_i can be represented by a directed edge from an initial node to the target node along y_i axis. Figure 3.1 is a graphical representation of a system consisting of 3 variables.

From each state Q_j a transition (an edge) to a neighboring state Q_k separated by $y_i = h_i$ hyperplane can happen if for λ_{ji} (ith diagonal element of λ_j, where λ_j is λ matrix belonging to state Q_j) following is satisfied:

- for $Re(\lambda_{ji}) < 0$:

  $-\lambda_{ji}^{-1}k_{ji} < h_i < y_e$,

  or $-\lambda_{ji}^{-1}k_{ji} > h_i > y_e$,

- for $Re(\lambda_{ji}) > 0$:

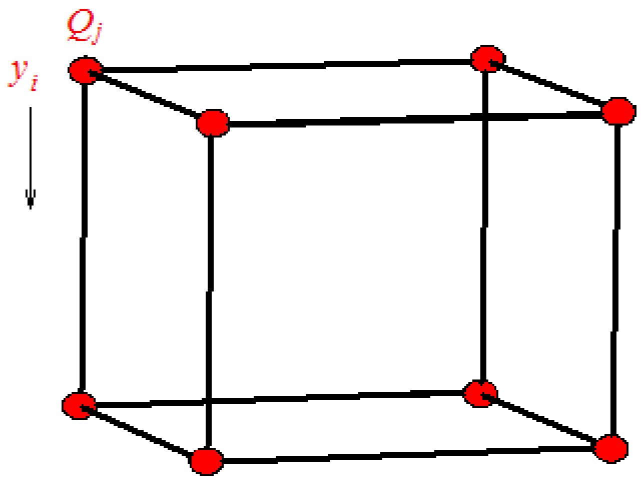


Figure 3.1: An example to network representation of state space of piecewise linear systems.

$$-\lambda_{ji}^{-1}k_{ji} < y_e < h_i,$$

$$\text{or } -\lambda_{ji}^{-1}k_{ji} > y_e > h_i.$$

where $y_e \in U_j$, U_j is the subspace (invariant set) of Q_j, and if $y(t-\tau) \in U_j$, then $\frac{dy}{dt} = \lambda_j y(t) + k_j$ is the governing differential equation.

If none of the transition conditions described above occur, then the system will not leave Q_j.

Assume each state Q (the corners of the hypercube) is represented as a vertex and possible transition as an edge. Then the system can be represented as a graph. An edge exists on this graph if one of the transition conditions is satisfied. Any circuit of the graph represents a cycle. And a cycle is a possible periodic solution of piecewise linear system.

Let $y_e \in U_j$ and possible transitions, satisfying transition conditions, from U_j are $\{(U_j, U_{k1}), (U_j, U_{k2}), ..., (U_j, U_{km})\}$ $(m \leq n)$ and assume the reaching times to corresponding thresholds be $(T_{k1}, T_{k2}, ..., T_{km})$ then y_e evolves into the state U_{ki}, where $k_i = \arg\min(T_{k1}, T_{k2}, ..., T_{km})$ $(i < m)$.

3.2 Stability Analysis of Piecewise Linear Systems with Delay on Piecewise Constant Part

The stable states of dynamical systems are either fixed points or periodic ones. From the graph of the described system in previous section we can derive the following results:

• Any state Q_j which has a convergent (all λ_{ij} less than zero) focal point within its invariant set has a fixed point stable state which is the focal point of the state. The verification of this statement is trivial. Any locally linear system with a convergent focal point will converge to the focal point in infinite time. If the focal point is within the initial state, the system will not leave the state and converges to focal point. The basin of attraction of such a fixed point stable state include the invariant set of Q_j and the set of points of other states which make a transition into Q_j.

• Any circuit on the graph of the system represents a cycle. Whether the corresponding cycle has a periodic attractor (stable state) or not is not so trivial. A widely used method for analyzing the stability of such system is selecting a convex manifold in the state space and checking whether the consequent crossings of the trajectory with the manifold form a contraction map. In the following parts of this section, we will continue the stability analysis applied on an example.

• Since a graph might have multiple number of circuits and a multiple number of subspaces might have a convergent focal point within its invariant set this class of systems can exhibit multistationary behavior.

The analysis is focused on a piecewise linear system on the plane with a constant delay. In this study the example introduced previously is revisited with a constant delay in state transition. The system includes a cyclic trajectory and the stability of this trajectory is investigated.

The system used for stability analysis is as seen in Figure 3.2.

Exemplary systems can be described as:

$$\frac{dx}{dt} = M_{s(t)}x(t) + k_{s(t)},$$

$$s_i(t) = F(Q([x_1(t-\tau), x_2(t-\tau)])),$$

$$Q_i(x(t)) = \{ \begin{array}{c} 1\ if\ x_i(t) > h_i \\ 0\ if\ x_i(t) \leq h_i \end{array} \text{ for } i = 1, 2,$$

$$M_{0,0} = \lambda_I = \begin{bmatrix} \lambda_{11} & 0 \\ 0 & \lambda_{12} \end{bmatrix}, M_{0,1} = \lambda_{II} = \begin{bmatrix} \lambda_{21} & 0 \\ 0 & \lambda_{22} \end{bmatrix},$$

$$M_{1,0} = \lambda_{III} = \begin{bmatrix} \lambda_{31} & 0 \\ 0 & \lambda_{32} \end{bmatrix}, M_{1,1} = \lambda_{IV} = \begin{bmatrix} \lambda_{41} & 0 \\ 0 & \lambda_{42} \end{bmatrix}, \lambda_{ij} \in \mathbb{R},$$

$$k_{0,0} = k_I = \begin{bmatrix} k_{11} \\ k_{12} \end{bmatrix}, k_{0,1} = k_{II} = \begin{bmatrix} k_{21} \\ k_{22} \end{bmatrix},$$

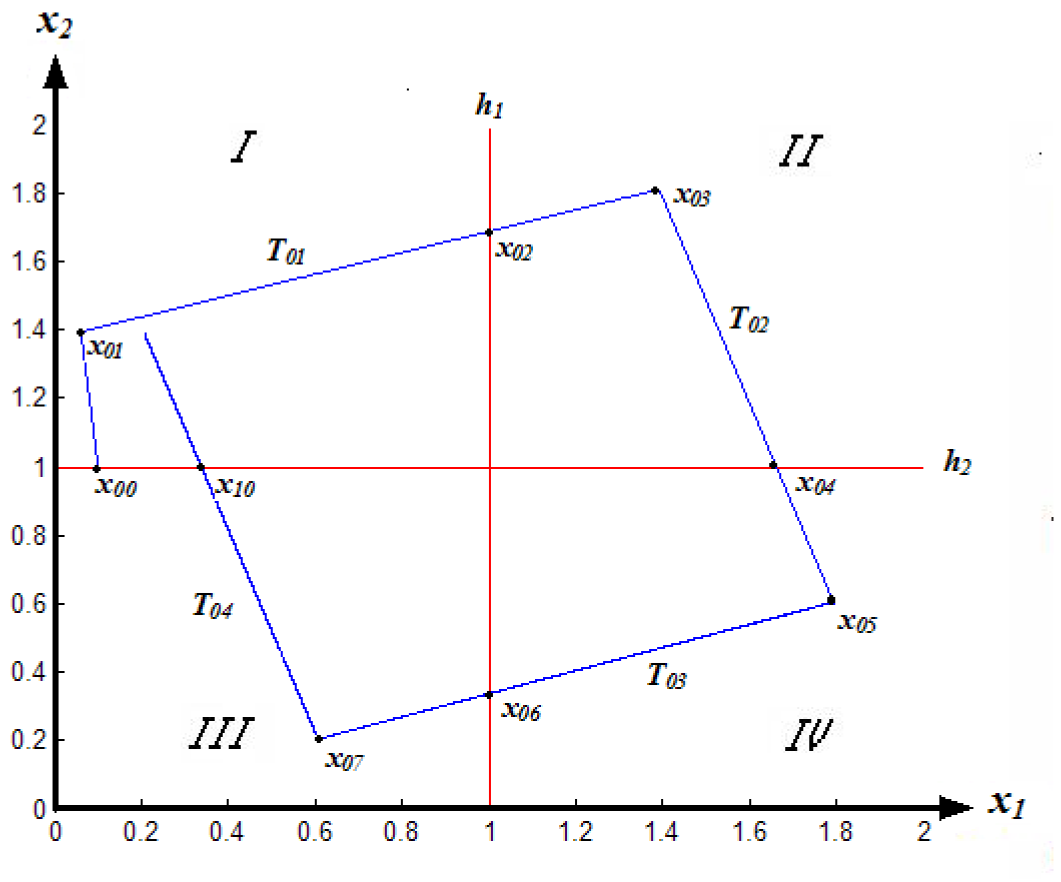


Figure 3.2: Trajectory of the system.

$$k_{1,0} = k_{III} = \begin{bmatrix} k_{31} \\ k_{32} \end{bmatrix}, \; k_{1,1} = k_{IV} = \begin{bmatrix} k_{41} \\ k_{42} \end{bmatrix}, \; k_{ij} \in \mathbb{R},$$

h_1 is the threshold for x_1 and h_2 is the threshold for x_2.

taking $Re(\lambda_{ij}) < 0$ and assuming that the focal point of state I is in state II, the focal point of state II is in state IV, the focal point of state IV is in state III, and the focal point of state III is in state I.

If the system is in state I (Q_I):

$\frac{dx}{dt} = \lambda_I x(t) + k_I$ is the governing differential equation for which it holds:

if $x^e = [x_1^e \; x_2^e]' \in U_I$, where U_I is a subspace (invariant set) of Q_I,

the following transition condition is satisfied:

$-\lambda_{I1}^{-1} k_{I1} > h_1 > x_1^e.$

Therefore, there will be a state transition $I \rightarrow II$

If the system is in state II (Q_{II}):

$\frac{dx}{dt} = \lambda_{II} x(t) + k_{II}$,is the governing differential equation for which,

if $x^e = [x_1^e \; x_2^e]' \in U_{II}$ where U_{II} is a subspace (invariant set) of Q_{II}

the following transition condition is fulfilled:

$-\lambda_{II2}^{-1} k_{II2} < h_2 < x_2^e.$

Therefore, there will be a state transition $II \rightarrow IV$

If the system is in state IV (Q_{IV}):

$\frac{dx}{dt} = \lambda_{IV} x(t) + k_{IV}$,is the governing differential equation for which,

if $x^e = [x_1^e \; x_2^e]' \in U_{IV}$ where U_{IV} is a subspace (invariant set) of Q_{IV}

the following transition condition is satisfied:

$-\lambda_{IV1}^{-1} k_{IV1} < h_1 < x_1^e.$

For this reason, there will be a state transition $IV \rightarrow III$.

If the system is in state III (Q_{III}), then

$\frac{dx}{dt} = \lambda_{III}x(t) + k_{III}$, is the governing differential equation for which,

if $x^e = [x_1^e \; x_2^e]' \in U_{III}$ where U_{III} is a subspace (invariant set) of Q_{III}

the coming transition condition is satisfied:

$-\lambda_{III2}^{-1}k_{III2} > h_2 > x_1^e$.

Therefore, there will be a state transition from $IV \rightarrow I$

For this reason, at the end, a cycle will be completed for such a system. This section will be concluded by an exemplary system with values as seen below will be considered.

$\frac{dx}{dt} = M_{s(t)}x(t) + k_{s(t)}$,

$s_i(t) = F(Q([x_1(t-\tau), x_2(t-\tau)]))$,

$$Q_i(x(t)) = \{ \begin{matrix} 1 \; if \; x_i(t) > h_i \\ 0 \; if \; x_i(t) \leq h_i \end{matrix} \quad \text{for } i = 1, 2,$$

$$M_{0,0} = M_{0,1} = M_{1,0} = M_{1,1} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$k_{0,0} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, k_{0,1} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, k_{1,0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, k_{1,1} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \text{ and}$$

$h_1 = h_2 = 1$ and $\tau = 0.5$.

The aim is to construct a Poincare map on the threshold h_2. The map will be $n-1$ dimensional. The system is 2-dimensional so the map will be 1-dimensional with variable x_1. Then, the fixed point of this map is found and the stability is investigated as described previously in Section 2.1.3.

Here τ represents the constant delay in state transitions. In the Figure 3.2 x_{ij} are the points where state transition or threshold crossing occurs. In the analysis work one complete cycle is investigated. Furthermore, T_{01} is the time required to reach to x_{02} from x_{01}, T_{02} is the time required to reach to x_{04} from x_{03}, T_{03} is the time required to reach to x_{06} from x_{05}, T_{04} is the time required to reach to x_{10}(on h_2) from x_{07}. The time required to reach to x_{01} from x_{00}, to x_{03} from x_{02}, to x_{05} from x_{04}, to x_{07} from x_{06} is τ.

Let us take a point $x_{00} = \begin{bmatrix} x_{00}^1 \\ x_{00}^2 \end{bmatrix}$ on h_2 (the line [0 1] on h_2 is taken as a cross section) as seen in Figure 3.1 and try to construct the map in terms of system defining constants and the initial point x_{00}.

Firstly, let us define the times $T_{01}, T_{02}, T_{03}, T_{04}$ in terms of system parameters and the initial point $x_{00} = \begin{bmatrix} x_{00}^1 \\ x_{00}^2 \end{bmatrix}$.

We want to define T_{01}, in terms of system parameters and initial point. In the work on T_{01}, consider the evolution of x_1 as seen in the Figure 3.3.

Here, the equation

$$x_{02}^1 = h_1 = e^{T_{01}\lambda_{11}}(x_{01}^1 + \lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11}$$

is obtained and from this equation,

$$e^{T_{01}\lambda_{11}} = \tfrac{h_1+\lambda_{11}^{-1}k_{11}}{x_{01}^1+\lambda_{11}^{-1}k_{11}}, \text{ and}$$

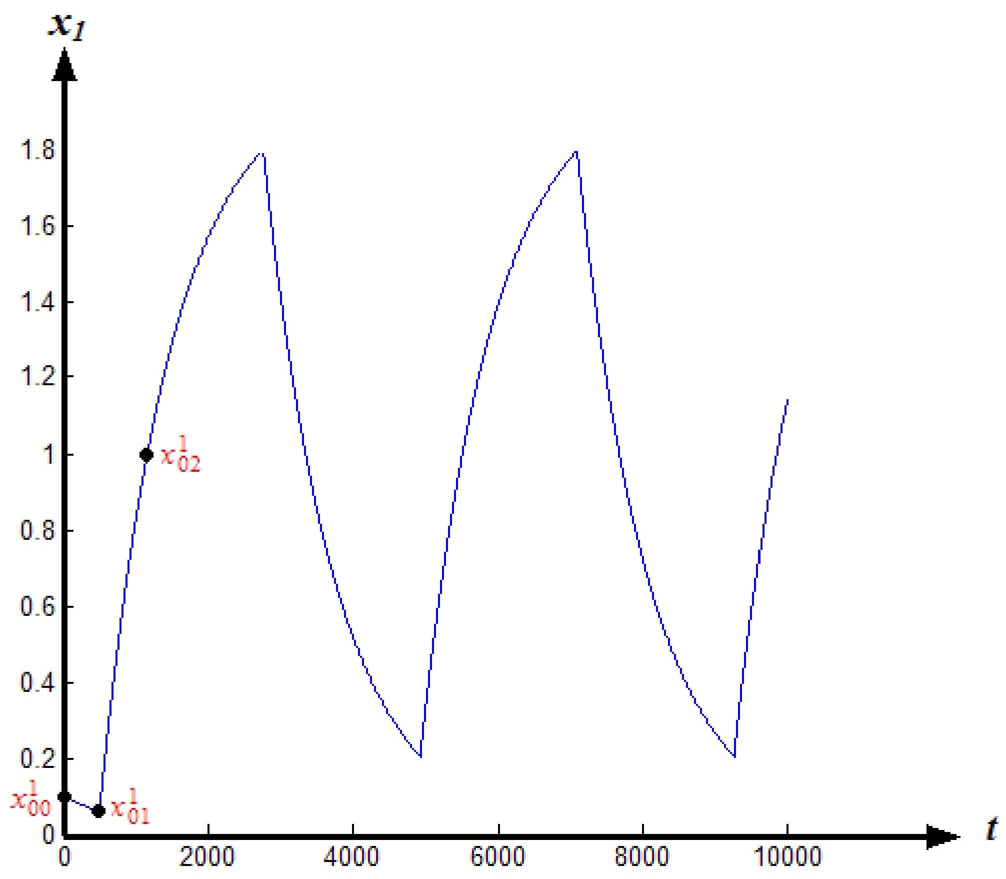


Figure 3.3: the evolution of x_1 during T_{01}.

$$T_{01} = \ln\left(\frac{h_1+\lambda_{11}^{-1}k_{11}}{x_{01}^1+\lambda_{11}^{-1}k_{11}}\right)/\lambda_{11} \qquad (1)$$

are found. Additionally,

$$x_{01}^1 = e^{\tau\lambda_{31}}(x_{00}^1 + \lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31} \qquad (2)$$

By substituting (2) in (1) we obtain

$$T_{01} = \ln((h_1 + \lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1+$$

$$\lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11},$$

where $\exp(a)$ means e^a.

Then, we try to obtain a function consisting of system parameters and initial point defining T_{02} so in the work on T_{02} consider the evolution of x_2 component as seen in the Figure 3.4.

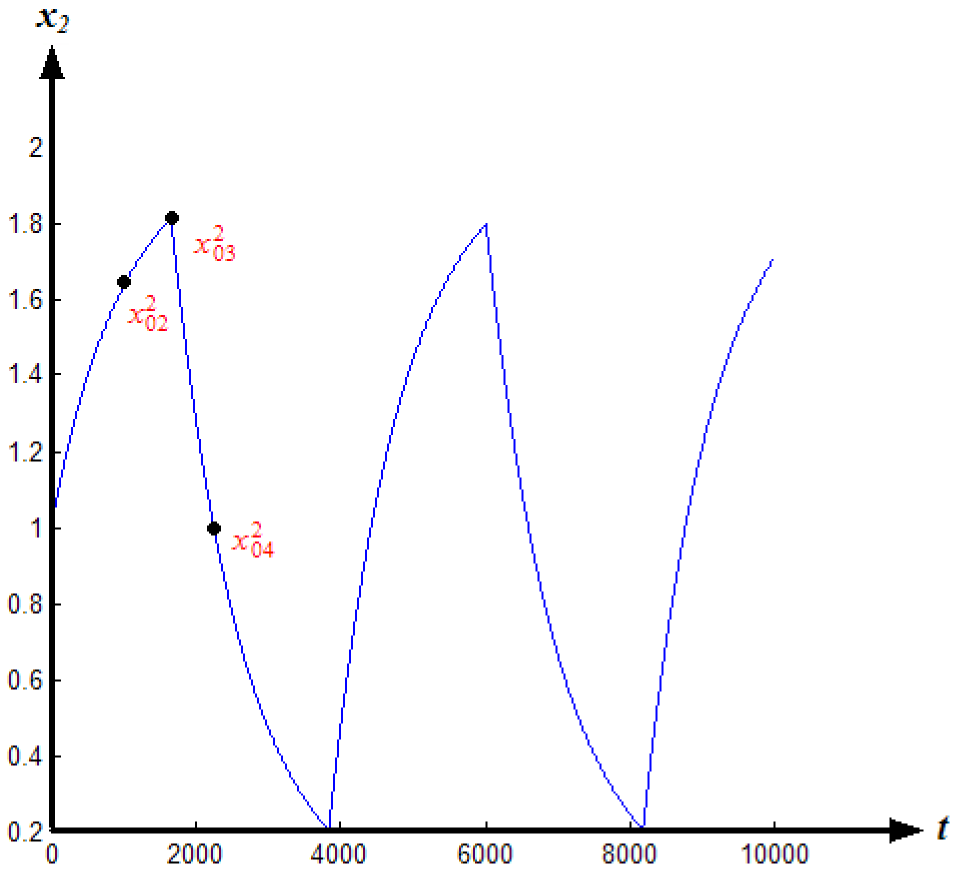


Figure 3.4: the evolution of x_2 during T_{02}.

Here, the equation

$$x_{04}^2 = h_2 = e^{T_{02}\lambda_{22}}(x_{03}^2 + \lambda_{22}^{-1}k_{22}) - \lambda_{22}^{-1}k_{22}$$

is obtained, and from this equation we get

$e^{T_{02}\lambda_{22}} = \frac{h_2+\lambda_{22}^{-1}k_{22}}{x_{03}^2+\lambda_{22}^{-1}k_{22}}$, and

$$T_{02} = \ln(\tfrac{h_2+\lambda_{22}^{-1}k_{22}}{x_{03}^2+\lambda_{22}^{-1}k_{22}})/\lambda_{22}. \qquad (3)$$

Additionally, we find

$$x_{03}^2 = e^{\tau\lambda_{12}}(x_{02}^2 + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12} \qquad (4.1)$$

$$x_{02}^2 = e^{T_{01}\lambda_{12}}(x_{01}^2 + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12} \qquad (4.2)$$

$$x_{01}^2 = e^{\tau\lambda_{32}}(x_{00}^2 + \lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32}$$

$$x_{01}^2 = e^{\tau\lambda_{32}}(h_2 + \lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32}. \qquad (4.3)$$

By substituting (4.3) in (4.2), (4.2) in (4.1), and finally (4.1) in (3), we obtain

$$T_{02} = \ln((h_2 + \lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(log((h_1+$$

$$\lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1 + \lambda_{31}^{-1}k_{31})-$$

$$\lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$$

$$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12})-$$

$$\lambda_{12}^{-1}k_{12} + \lambda_{22}^{-1}k_{22}))/\lambda_{22}.$$

Then, in the work on T_{03} consider the evolution of x_1 component as seen in the Figure 3.5.

Here, the equation

$$x_{06}^1 = h_1 = e^{T_{03}\lambda_{41}}(x_{05}^1 + \lambda_{41}^{-1}k_{41}) - \lambda_{41}^{-1}k_{41}$$

is obtained and from this equation, we get

$e^{T_{03}\lambda_{41}} = \frac{h_1+\lambda_{41}^{-1}k_{41}}{x_{05}^1+\lambda_{41}^{-1}k_{41}}$,

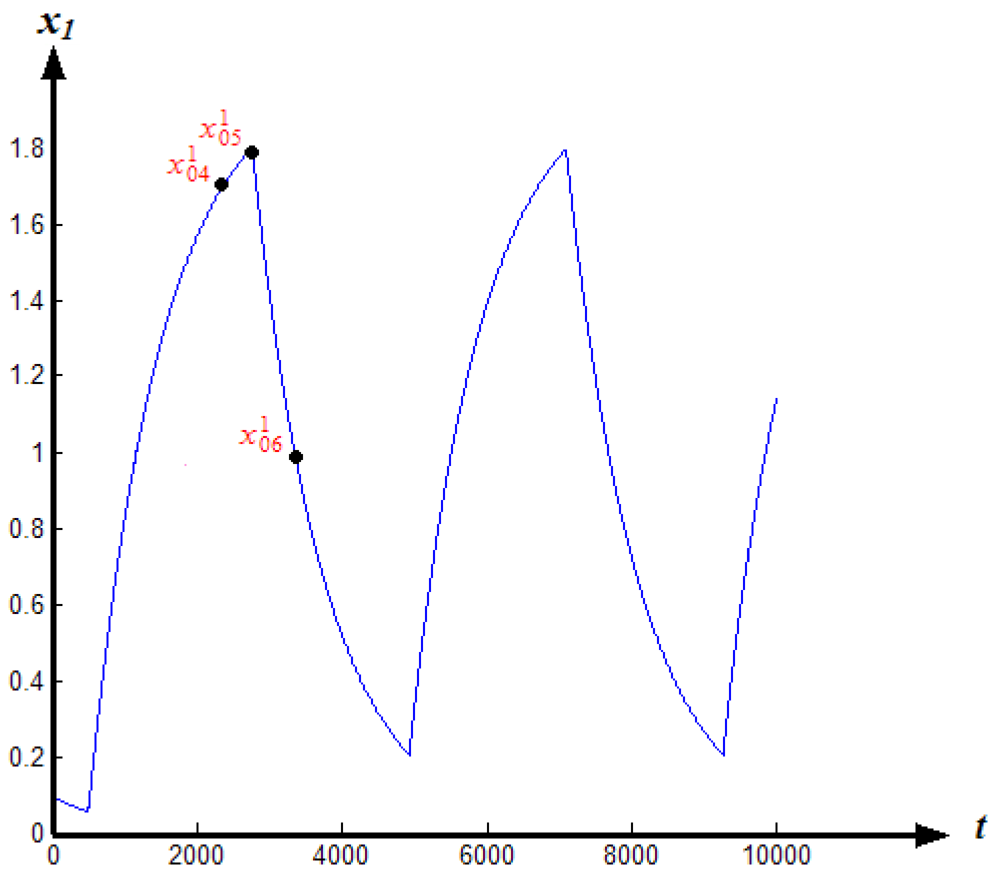


Figure 3.5: the evolution of x_1 during T_{03}.

$$T_{03} = \ln\left(\frac{h_1+\lambda_{41}^{-1}k_{41}}{x_{05}^1+\lambda_{41}^{-1}k_{41}}\right)/\lambda_{41}. \tag{5}$$

Additionally,

$$x_{05}^1 = e^{\tau\lambda_{21}}(x_{04}^1 + \lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21}. \tag{6.1}$$

$$x_{04}^1 = e^{T_{02}\lambda_{21}}(x_{03}^1 + \lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21}. \tag{6.2}$$

$$x_{03}^1 = e^{\tau\lambda_{11}}(x_{02}^1 + \lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11}$$

$$x_{03}^1 = e^{\tau\lambda_{11}}(h_1 + \lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11}. \tag{6.3}$$

By substituting (6.3) in (6.2), (6.2) in (6.1), and finally (6.1) in (5), we receive

$$T_{03} = ln((h_1 + \lambda_{41}^{-1}k_{41})/(exp(r\lambda_{21})exp(\ln((h_2+$$

$$\lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1+$$

$$\lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1 + \lambda_{31}^{-1}k_{31})-$$

$$\lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$$

$$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12}+$$

$$\lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1+$$

$$\lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11} + \lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21}+$$

$$\lambda_{41}^{-1}k_{41}))/\lambda_{41}$$

Finally, in the work on T_{04} consider the evolution of x_2 component of x_{06}^2, x_{07}^2,*and* x_{10}^2 as seen in the Figure 3.6.

Here, the equation

$$x_{10}^2 = h_2 = e^{T_{04}\lambda_{32}}(x_{07}^2 + \lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32}$$

is obtained, and from this equation we gets

$$e^{T_{04}\lambda_{32}} = \frac{h_2+\lambda_{32}^{-1}k_{32}}{x_{07}^2+\lambda_{32}^{-1}k_{32}},$$

$$T_{04} = \ln(\frac{h_2+\lambda_{32}^{-1}k_{32}}{x_{07}^2+\lambda_{32}^{-1}k_{32}})/\lambda_{32}. \quad (7)$$

Additionally,

$$x_{07}^2 = e^{\tau\lambda_{42}}(x_{06}^2 + \lambda_{42}^{-1}k_{42}) - \lambda_{42}^{-1}k_{42}, \quad (8.1)$$

$$x_{06}^2 = e^{T_{03}\lambda_{42}}(x_{05}^2 + \lambda_{42}^{-1}k_{42}) - \lambda_{42}^{-1}k_{42}, \quad (8.2)$$

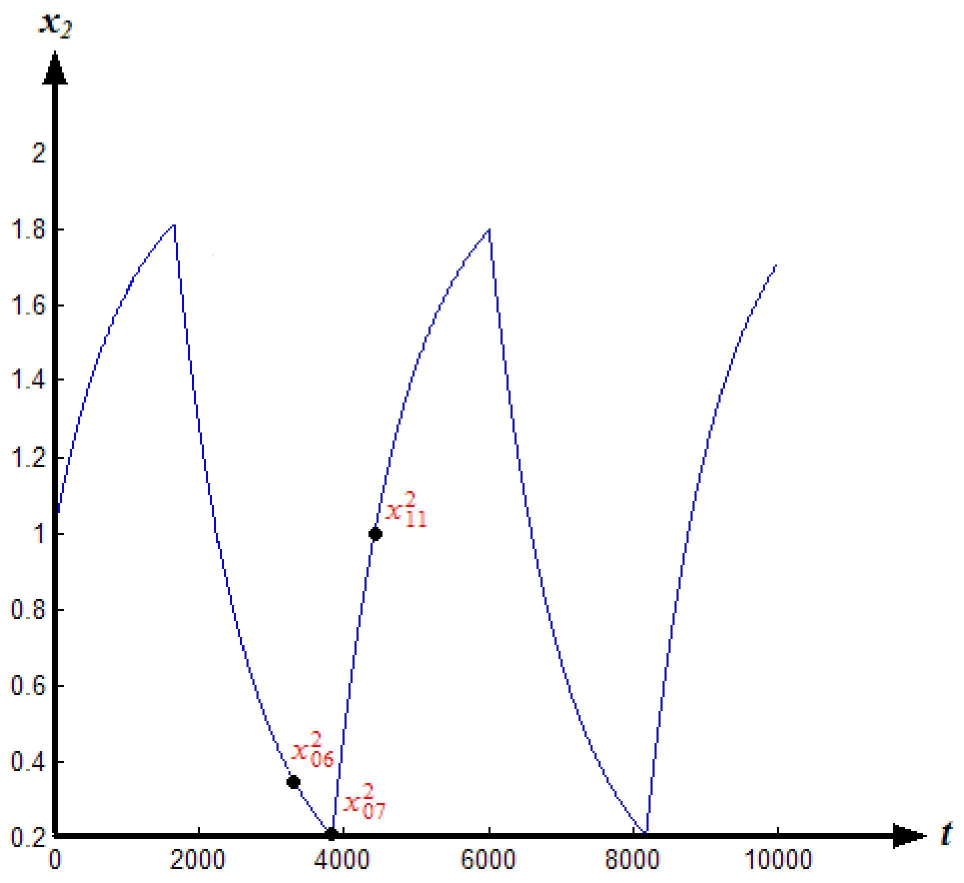


Figure 3.6: the evolution of x_2 during T_{04}.

$$x_{05}^2 = e^{\tau\lambda_{22}}(x_{04}^2 + \lambda_{22}^{-1}k_{22}) - \lambda_{22}^{-1}k_{22}$$

$$x_{05}^2 = e^{\tau\lambda_{22}}(h_2 + \lambda_{22}^{-1}k_{22}) - \lambda_{22}^{-1}k_{22}. \tag{8.3}$$

By substituting (8.3) in (8.2), (8.2) in (8.1), and finally (8.1) in (7) we obtain

$$T_{04} = \ln((h_2 + \lambda_{32}^{-1}k_{32})/(exp(r\lambda_{42})exp(\ln((h_1+$$

$$\lambda_{41}^{-1}k_{41})/(exp(r\lambda_{21})exp(\ln((h_2+$$

$$\lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1+$$

$\lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1 + \lambda_{31}^{-1}k_{31})-$

$\lambda_{31}^{-1}k_{31}+ \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$

$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12}+$

$\lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1+$

$\lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11} + \lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21}+$

$\lambda_{41}^{-1}k_{41}))/\lambda_{41}\lambda_{42})(exp(r\lambda_{22})(h_2+$

$\lambda_{22}^{-1}k_{22}) - \lambda_{22}^{-1}k_{22} + \lambda_{42}^{-1}k_{42}) - \lambda_{42}^{-1}k_{42}+$

$\lambda_{32}^{-1}k_{32}))/\lambda_{32}$

Now, consider the map by x_1 - component. The starting point is x_{00}^1 and after the cycle $I \rightarrow II \rightarrow IV \rightarrow III$ as seen in Figure 3.2, the trajectory returns to the point x_{11}^1. This point will be represented by the map f such that

$$f(x_{00}^1) = x_{10}^1.$$

It is obvious that

$$x_{10}^1 = e^{T_{04}\lambda_{31}}(x_{07}^1 + \lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31}$$

$$x_{07}^1 = e^{\tau\lambda_{41}}(x_{06}^1 + \lambda_{41}^{-1}k_{31}) - \lambda_{41}^{-1}k_{41}$$

$$x_{07}^1 = e^{\tau\lambda_{41}}(h_1 + \lambda_{41}^{-1}k_{31}) - \lambda_{41}^{-1}k_{41},$$

so

$$x_{10}^1 = e^{T_{04}\lambda_{31}}(e^{\tau\lambda_{41}}(h_1 + \lambda_{41}^{-1}k_{41}) - \lambda_{41}^{-1}k_{41} + \lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31},$$

and substituting T_{04} which is found in (7), we get

$x_{10}^1 = f(x_{00}^1) = exp(\ln((h_2+$

$\lambda_{32}^{-1}k_{32})/(exp(r\lambda_{42})exp(\ln((h_1+$

$\lambda_{41}^{-1}k_{41})/(exp(r\lambda_{21})exp(\ln((h_2+$

$\lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1+$

$\lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1 + \lambda_{31}^{-1}k_{31})-$

$\lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$

$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12}+$

$\lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1 + \lambda_{11}^{-1}k_{11})-$

$\lambda_{11}^{-1}k_{11} + \lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21}+$

$\lambda_{41}^{-1}k_{41}))/\lambda_{41}\lambda_{42})(exp(r\lambda_{22})(h_2+$

$\lambda_{22}^{-1}k_{22}) - \lambda_{22}^{-1}k_{22} + \lambda_{42}^{-1}k_{42}) - \lambda_{42}^{-1}k_{42}+$

$\lambda_{32}^{-1}k_{32}))/\lambda_{32}l31)(e^{\tau\lambda_{41}}(h_1 + \lambda_{41}^{-1}k_{41})-$

$\lambda_{41}^{-1}k_{41} + \lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31}.$

Using numerical simulation the value 0.3379 is found to be the fixed point of this map for numeric exemplary system given at the beginning of this section.

The derivative of the map $f(x_{00}^1)$ is found by the help of MatLab as

$f'(x_{00}^1) = \frac{df(x_{00}^1)}{dx_{00}^1} = 1/(exp(r\lambda_{42})exp(\ln((h_1+$

$\lambda_{41}^{-1}k_{41})/(exp(r\lambda_{21})exp(\ln((h_2+$

$\lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1 + \lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1+$

$\lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$

$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12}+$

$\lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1 + \lambda_{11}^{-1}k_{11})-$

$\lambda_{11}^{-1}k_{11} + \lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21}+$

$\lambda_{41}^{-1}k_{41}))/\lambda_{41}\lambda_{42})(exp(r\lambda_{22})(h_2 + \lambda_{22}^{-1}k_{22})-$

$\lambda_{22}^{-1}k_{22} + \lambda_{42}^{-1}k_{42}) - \lambda_{42}^{-1}k_{42}+$

$\lambda_{32}^{-1}k_{32})exp(r\lambda_{42})/(exp(r\lambda_{21})exp(\ln((h_2+$

$\lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1 + \lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^{1}+$

$\lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$

$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12}+$

$\lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1 + \lambda_{11}^{-1}k_{11})-$

$\lambda_{11}^{-1}k_{11} + \lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21}+$

$\lambda_{41}^{-1}k_{41})exp(r\lambda_{21})/(exp(r\lambda_{12})exp(\ln((h_1+$

$\lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^{1} + \lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31}+$

$\lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2 + \lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32}+$

$\lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12} + \lambda_{22}^{-1}k_{22})exp(r\lambda_{12})/(exp(r\lambda_{31})(x_{00}^{1}+$

$\lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11})exp(r\lambda_{31})/\lambda_{11}\lambda_{12}exp(\ln((h_1+$

$\lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^{1} + \lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31}+$

$\lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2 + \lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32}+$

$\lambda_{12}^{-1}k_{12})/\lambda_{22}\lambda_{21}exp(\ln((h_2 + \lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1+$

$\lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1 + \lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31}+$

$\lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2 + \lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32}+$

$\lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12} + \lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1+$

$\lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11} + \lambda_{21}^{-1}k_{21})/\lambda_{41}\lambda_{42}exp(\ln((h_1+$

$\lambda_{41}^{-1}k_{41})/(exp(r\lambda_{21})exp(\ln((h_2+$

$\lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1 + \lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1+$

$\lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$

$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12}+$

$\lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1 + \lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11}+$

$\lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21} + \lambda_{41}^{-1}k_{41}))/\lambda_{41}\lambda_{42})(exp(r\lambda_{22})(h_2+$

$\lambda_{22}^{-1}k_{22}) - \lambda_{22}^{-1}k_{22} + \lambda_{42}^{-1}k_{42})/\lambda_{32}\lambda_{31}exp(\ln((h_2+$

$\lambda_{32}^{-1}k_{32})/(exp(r\lambda_{42})exp(\ln((h_1+$

$\lambda_{41}^{-1}k_{41})/(exp(r\lambda_{21})exp(\ln((h_2+$

$\lambda_{22}^{-1}k_{22})/(exp(r\lambda_{12})exp(\ln((h_1 + \lambda_{11}^{-1}k_{11})/(exp(r\lambda_{31})(x_{00}^1+$

$\lambda_{31}^{-1}k_{31}) - \lambda_{31}^{-1}k_{31} + \lambda_{11}^{-1}k_{11}))/\lambda_{11}\lambda_{12})(exp(r\lambda_{32})(h_2+$

$\lambda_{32}^{-1}k_{32}) - \lambda_{32}^{-1}k_{32} + \lambda_{12}^{-1}k_{12}) - \lambda_{12}^{-1}k_{12}+$

$\lambda_{22}^{-1}k_{22}))/\lambda_{22}\lambda_{21})(exp(r\lambda_{11})(h_1 + \lambda_{11}^{-1}k_{11}) - \lambda_{11}^{-1}k_{11}+$

$\lambda_{21}^{-1}k_{21}) - \lambda_{21}^{-1}k_{21} + \lambda_{41}^{-1}k_{41}))/\lambda_{41}\lambda_{42})(exp(r\lambda_{22})(h_2+$

$\lambda_{22}^{-1}k_{22}) - \lambda_{22}^{-1}k_{22} + \lambda_{42}^{-1}k_{42}) - \lambda_{42}^{-1}k_{42}+$

$\lambda_{32}^{-1}k_{32}))/\lambda_{32}\lambda_{31})(exp(r\lambda_{41})(h_1 + \lambda_{41}^{-1}k_{41}) - \lambda_{41}^{-1}k_{41} + \lambda_{31}^{-1}k_{31}).$

For numeric exemplary system given on page 58, $|f'(x)| < 1$ where $x \in [0, 1]$, is found according to numerical simulations. Here, the interval $[0, 1]$ is considered because the function $f(x)$ will evolve in this interval for the given numeric exemplary system. Therefore, $f(x)$ is a contraction on this interval and by contraction principle $f(x)$ has a unique fixed point on this interval. This fixed point is $x = 0.3379$ as found. $|\mathrm{f}'(0.3379)| = 0.0001699$ and this point is stable according to Theorem 2.3.

For n-dimensional systems the stability analysis can be done by obtaining an n-1 dimensional map on hyperplane threshold and investigating the stability of fixed point of this map.

The sample system has an stable periodic attractor. The stability analysis on periodic attractors of piecewise linear systems with delay can be done in this way, and stable periodic attractors can be obtained.

CHAPTER 4

FISSION YEAST CELL CYCLE MODEL

In this chapter, we present a piecewise linear systems with delay on piecewise constant part for modelling the cell cycle regulation of fission yeast. In first section, background for fission yeast cell cycle is presented. In the second section, the well-known ODE model and our piecewise linear model with delay are included. Finally, a brief discussion about the advantages of our model is given.

4.1 Background for Fission Yeast Cell Cycle

The regulatory mechanism that order and coordinate the progress of the eukaryotic cell cycle have been intensively studied in recent years. Yeast have proved invaluable in unravelling the major control elements of the eukaryotic cell cycle and research with two species, in particular, has significantly advanced the understanding of cell cycle regulation: the budding yeast, Saccharomyces cerevisiae and the fission yeast, Schizosaccharomyces pombe. Such an understanding is important in the field of human cancer, which is fundamentally a disease of the cell cycle [71].

4.1.1 Cell Cycle of Fission Yeast

Cells reproduce and duplicate their contents and then divide in two. This cell-division cycle is the fundamental means by which all living things are propagated. In unicellular species, such as bacteria and yeast, each cell division produces an additional organism. In multicellular species, many rounds of cell division are required to make a new individual, and cell division is needed in the adult body, too, to replace cells that are lost by wear and tear or by programmed cell death [3].

The details of the cell cycle may vary, but certain requirement are universal. First and foremost, to produce a pair of genetically identical daughter cells, the DNA must be faithfully replicate, and the replicated chromosomes must be segregated into two separate cells as seen in Figure 4.1. Additionally, recent experiments have provided a new and simpler perspective, revealing a cell-cycle control system that coordinates the cycle as a whole. The proteins of this control system first appeared over a billion years ago and have been so well conserved in evolution that many of them function perfectly when transferred from a human cell to a yeast cell [3].

Genetic studies of the cell cycle in the yeasts Saccharomyces cerevisiae and Schizosaccharomyces pombe illuminated important principles underlying the logic of cell cycle control and identified many of its key players. Screens were performed for conditional temperature-sensitive mutations, causing cells to arrest quickly

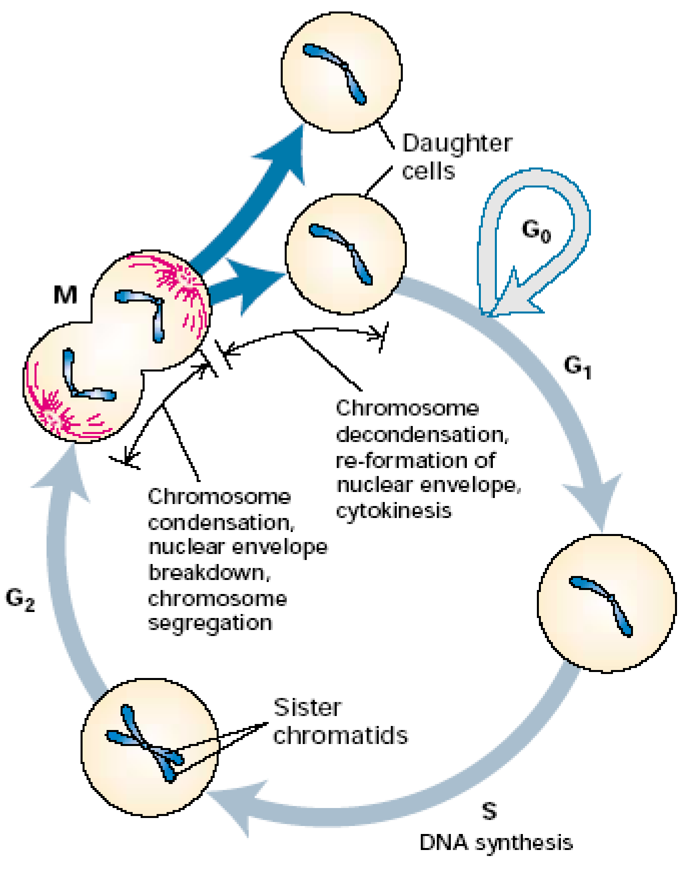

Daughter cells
G₀
M
G₁
Chromosome decondensation, re-formation of nuclear envelope, cytokinesis
Chromosome condensation, nuclear envelope breakdown, chromosome segregation
G₂
Sister chromatids
S
DNA synthesis

Figure 4.1: Summary of major events in the eukaryotic cell cycle [38].

with uniform morphologies, indicative of defects in executing individual cell cycle steps. Analysis of a large collection of such cdc (cell division cycle) mutants from S. cerevisiae placed them into series of dependent and independent pathways [13].

The most important components of the eukaryotic cell cycle engine are cyclin-dependent protein kinases, heterodimers consisting of a catalytic subunit (a Cdk) and a regulatory subunit (a cyclin). Cdks, which are active only in complex with a cyclin partner, exert their action by phosphorylating other proteins [47, 50]. Their protein-kinase activity is required to start both DNA replication and mitosis. Lower eukaryots use only one essential Cdk subunit (generally called Cdk1), while higher eukaryotes use many. Cdk1 is often called Cdc2, in recognition of the gene (cdc2) that encodes this protein in fission yeast [50, 51]. In fission yeast, complexes between Cdk1 and B-type cyclins play the major roles in cell cycle regulation [25, 50]. Cdc13 is the only essential B-type cyclin [25, 50]. Cdc2/Cdc13 activity is called M-phase promoting factor (MPF).

Figure 4.2 illustrates the functions of four proteins that regulate the protein kinase activity of the fission yeast CDK. First is Cdc13, the mitotic cyclin of fission yeast, which associates with the CDK to form MPF with extremely low activity. Second is the Wee1 protein-tyrosine kinase, which phosphorylates an inhibitory tyrosine residue (Y15) in the CDK subunit. Third is another kinase, designated CDK-activating kinase (CAK), which phosphorylates an activating threonine residue (T161). When both residues are phosphorylated, MPF is inactive. Finally, the Cdc25 phosphates removes the phosphate from Y15, yielding highly active MPF [38].

The destruction of mitotic CDK activity at anaphase allows cells to divide and enter G1 phase of the next cell cycle [10, 69]. Exit from mitosis is controlled by the anaphase promoting complex (APC), which initiates the degradation of cohesions and mitotic cyclins [69, 76]. Hence, to understand the molecular control of cell reproduction is to understand the regulation of CDK and APC activities [66, 69, 76].

MPF is controlled by antagonistic interactions with its enemies [49, 50]. The enemies have negative effects on MPF, but MPF can down-regulate all of its

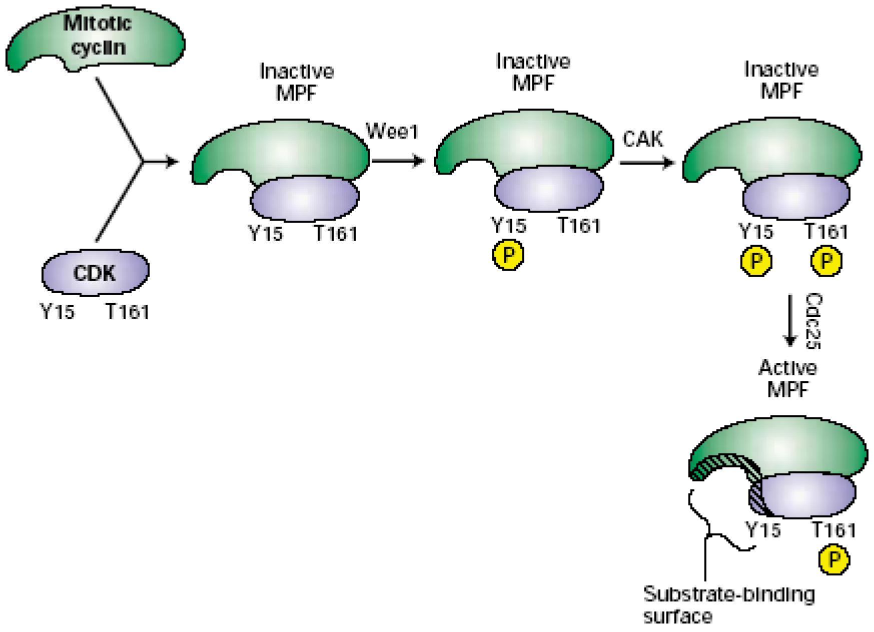


Figure 4.2: Regulation of the kinase activity of fission yeast mitosis-promoting factor (MPF) [38].

enemies [50] as visualized in Figure 4.3 [69].

Two of these enemies these enemies are active in G1 phase, while a different group regulates the G2/M transition [50].

The first G1 enemy, Ste9 (also called Srw1), targets Cdc13 to the APC core and promotes its degradation in G1 [50, 74]. On the other hand, phosphorylation of Ste9 by MPF inhibits its association with the APC core, rendering it inactive [50, 75].

The other G1 enemy of MPF is a stoichiometric inhibitor, called Rum1 [46, 50], which can bind to Cdc2/Cdc13 complexes and inhibit their activity [12, 50]. However, phosphorylation of Rum1 by MPF promotes its ubiquitination (by a

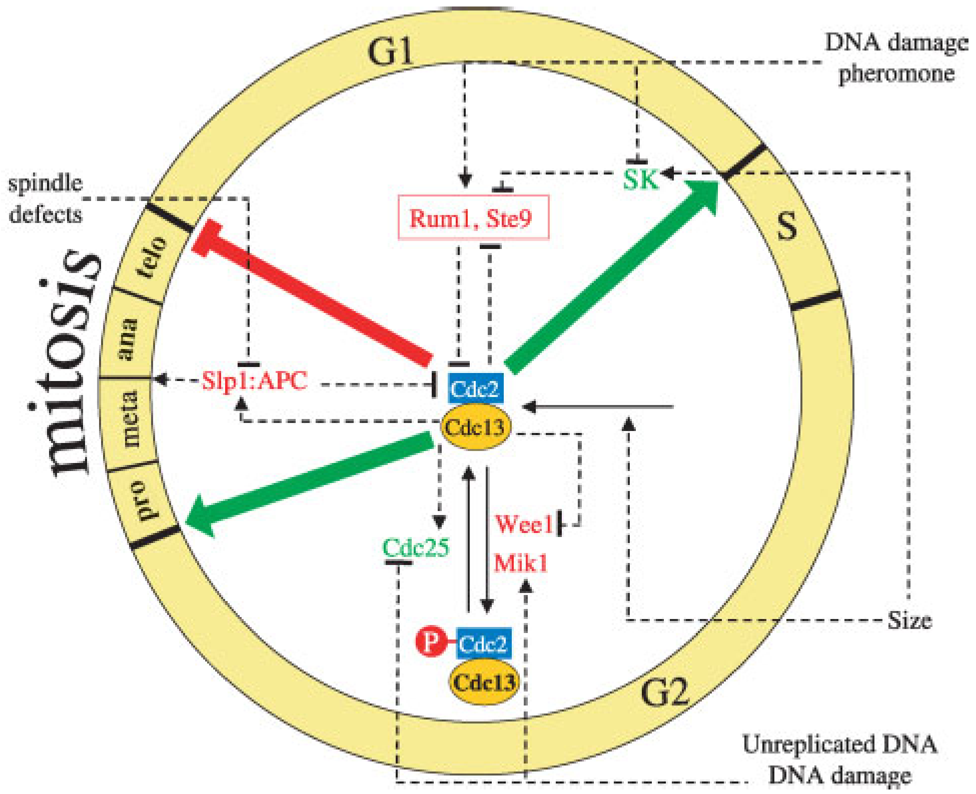


Figure 4.3: The cell cycle engine in fission yeast [69].

different complex than APC) and repair degradiation [8, 50]. Hence, there is antagonism between MPF and Rum1, as well as between MPF and Ste9 [50].

Because of these antagonistic relationships, MPF and its G1 enemies con not coexist. Either the enemies win and the cell is in G1 phase (with low MPF activity), or MPF wins and the cell is in S/G2/M phase of the cell [49, 50]. The fight between MPF and its enemies is modulated by helper molecules, which shift the balance in one direction or the other [50].

The helper molecule for the start transition (G1→S) is a "starter" kinase, a group of Cdk/cyclin complexes (Cdc2 with Cig1, Cig2, and Puc1 cyclins), which help MPF to get the upper hand by phosphorylating Rum1 [43, 50] and Ste9. The starter kinases can help MPF because they are less sensitive to Rum1 inhibition and Ste9-dependent ubiquitination. The helper molecule for the finish transition (M→G1) is the Slp1/APC complex, which promotes the degradation of Cdc13 and activates Ste9 (possibly by activating the phosphates that activates Ste9). Slp1 can help the enemies because it is not inactivated by MPF phosphorylation, as is Ste9. In fact, Slp1 seems to be activated in an MPF-dependent manner.

The duration of G2 phase is regulated by a different mechanism, namely enzymatic inhibition of MPF activity. The active site of Cdc2 contains a phosphorylatable tyrosine residue, and its tyrosine-phosphorylated form is inactive [50, 51]. Two tyrosine kinases can inactivate Cdc2 in this way, Wee1 and Mik1 [36, 50, 61]. In returns, MPF can also phosphorylate and inactivate them [4, 50]. So there is another case of mutual antagonism and alternative steady states: an S/G2 state (plenty of tyrosine-phosphorylated Cdc2/Cdc13, with enough activity to support DNA synthesis but not mitosis) and an M state (inactive Wee1 and Mik1, lots of highly active Cdc2/Cdc13 cell in mitosis).

The G2/M transition is accelerated by a direct positive feedback loop. The inhibitory phosphate group of Cdc2 is removed by a specific phosphates, called Cdc25 [45, 50]. Cdc25 is also phosphorylated by MPF, but the phosphorylated form of Cdc25 is more active [45, 50]. In this case, MPF helps its friend, Cdc25 [50]. The activation - inactivation relationship between MPF, Wee1 and Cdc25 is visualized as discussed before in Figure 4.2.

The proteins that modulate Cdc2 activity are themselves modulated by Cdc2:Cdc13, through a set of feedback loops [68, 69].

- Rum1 inhibits Cdc2:Cdc13, but Cdc2:Cdc13 phosphorylates Rum1, thereby targeting Rum1 for degradation.

- Ste9:APC labels Cdc13 for degradation, but Cdc2:Cdc13 can phosphorylate Ste9, thereby downregulating its activity and targeting it for degradation.

- Wee1 phosphorylates and inactivates Cdc2:Cdc13, but, at the same time, Cdc2: Cdc13 is trying to phosphorylate and inactivate Wee1.

- Cdc25 takes the inactivating phosphate group off PCdc2: Cdc13, and Cdc2: Cdc13 returns the favor by phosphorylating and thereby activating Cdc25.

- Slp1:APC, which also labels Cdc13 for degradation, is itself activated by Cdc2: Cdc13 by an indirect pathway.

The first three feedback loops are examples of mutual antagonism. Under appropriate conditions, the antagonists cannot coexist, i.e., the feedback loop works like a toggle switch. Either Cdc2:Cdc13 has the upper hand and its antagonist (Rum1 or Ste9 or Wee1) is suppressed, or vice versa. The fourth interaction is a positive feedback loop: Cdc2 and Cdc25 activate each other in a mutually amplifying fashion. The last interaction is a time-delayed negative feedback loop, which, under appropriate conditions, can generate oscillations (as Cdc2:Cdc13 concentration rises, it turns on Slp1, which targets Cdc13 for degradation, causing Cdc2:Cdc13 concentration to fall, and Slp1 to turn off).

The state of these feedback loops responds to cell size. Small cells tend to be in G1 phase (with little Cdc2 activity); medium-sized cells tend to be in S–G2 phase; large cells tend to be in M phase, with Cdc25 active and Wee1 inactive [60, 62, 69]. It is this responsiveness of the Cdc2 control system to cell size that coordinates the chromosome cycle to cell growth. To model these effects, John J. Tyson, Attila Csikasz-Nagy, and Bela Novak assumed that Cdc13 is synthesized at a rate proportional to cell mass (i.e., number of ribosomes), and then it combines with Cdc2 and moves into the nucleus, where its effective nuclear concentration increases steadily as the cell grows. Hence, an important determinant of the state of the Cdc2 control system is the mass/nucleus ratio [68, 69].

4.2 Modeling Cell Cycle Regulation of Fission Yeast

4.2.1 ODE Model

The molecular mechanism described in previous section can be summarized in a schematic wiring diagram Figure 4.4 according to Olivia Eriksson, Yishao Zhou and Jesper Tagner [50].

To keep the model simple, a number of dynamic variables are assumed to be in pseudosteady state: (1) the TF for synthesis of the starter kinase (SK), (2) the trimeric complexes of Rum1 and Cdc13/Cdc2, and (3) the tyrosine modifying enzymes (Wee1 and Cdc25). It is assumed that Rum1 binds to both types of Cdc2/Cdc13 dimers: unphosphorylated (MPF) and phosphorylated (preMPF) (see Figure 4.5).

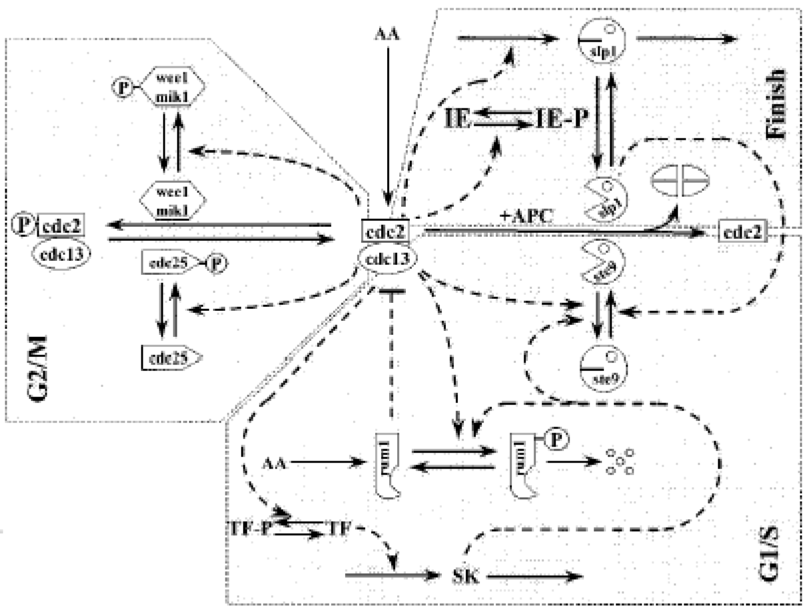


Figure 4.4: The wiring diagram of the fission-yeast cell-cycle engine. In the middle of the diagram is Cdc2/Cdc13 ˜MPF!, which is regulated by proteolysis of the Cdc13 component, phosphorylation of Cdc2 subunit, and stoichiometric inhibition of the complex. These processes are arranged according to the cell cycle transitions in which they are involved [50].

The cell cycle mechanism of fission yeast is converted to algebraic and differential equations [50]. These algebraic and differential equations are presented in Table 4.1.

The parameter values of Table 4.1 for a wild type cell is as presented in Table 4.2.

The cell mass increases exponentially from one to two between birth and cell division. The cell mass is divided by two at the end of mitosis, when MPF decreases through 0.01, although daughter cells do not physically separate from

Table 4.1: The differential and algebraic equations describing the cell cycle mechanism of fission yeast [50].

$$\frac{d[Cdc13_T]}{dt} = k_1 M - (k_2' + k_2''[Ste9] + k_2'''[Slp1])[Cdc13_T],$$

$$\frac{d[preMPF]}{dt} = k_{wee}([Cdc13_T] - [preMPF]) - k_{25}[preMPF] - (k_2' + k_2''[Ste9] + k_2'''[Slp1])[preMPF],$$

$$\frac{d[Ste9]}{dt} = (k_3' + k_3''[Slp1])\frac{1-[Ste9]}{J_3+1-[Ste9]} - (k_4'[SK] + k_4[MPF])\frac{[Ste9]}{J_4+[Ste9]},$$

$$\frac{d[Slp1_T]}{dt} = k_5' + k_5''\frac{[MPF]^4}{J_5^4+[MPF]^4} - k_6[Slp1_T],$$

$$\frac{d[Slp1]}{dt} = k_7[IEP]\frac{[Slp1_T]-[Slp1]}{J_7+[Slp1_T]-[Slp1]} - k_8\frac{[Slp1]}{J_8+[Slp1]} - k_6[Slp1],$$

$$\frac{d[IEP]}{dt} = k_9[MPF]\frac{1-[IEP]}{J_9+1-[IEP]} - k_{10}\frac{[IEP]}{J_{10}+[IEP]},$$

$$\frac{d[Rum1_T]}{dt} = k_{11} - (k_{12} + k_{12}'[SK] + k_{12}''[MPF])[Rum1_T],$$

$$\frac{d[SK]}{dt} = k_{13}[TF] - k_{14}[SK],$$

$$\frac{dM}{dt} = \mu M,$$

$$[Trimer] = \frac{2[Cdc13_T][Rum1_T]}{\Sigma + \sqrt{\Sigma^2 - 4[Cdc13_T][Rum1_T]}},$$

$$[MPF] = \frac{([Cdc13_T]-[preMPF])([Cdc13_T]-[Trimer])}{[Cdc13_T]},$$

$$[TF] = G(k_{15}M, k_{16}' + k_{16}''[MPF], J_{15}, J_{16}),$$

where

$$k_{wee} = k_{wee}' + (k_{wee}'' - k_{wee}')G(V_{awee}, V_{iwee}[MPF], J_{awee}, J_{iwee}),$$

$$k_{25} = k_{25}' + (k_{25}'' - k_{25}')G(V_{a25}[MPF], V_{i25}, J_{a25}, J_{i25}),$$

$$\Sigma = [Cdc13_T] + [Rum1_T] + K_{diss},$$

$$G(a,b,c,d) = \frac{2ad}{b-a+bc+ad+\sqrt{(b-a+bc+ad)^2-4ad(b-a)}}.$$

Table 4.2: Parameter values for wild-type cells [50].

Cdc13 synthesis and degradation:

$k_1 = 0.03, k_2^{'} = 0.03, k_2^{''} = 1, k_2^{'''} = 0.1.$

Ste9 activation and inactivation:

$k_3^{'} = 1, k_3^{''} = 10, J_3 = 0.01, k_4^{'} = 2, k_4 = 35, J_4 = 0.01.$

Slp1 synthesis, degradation, activation and inactivation:

$k_5^{'} = 0.005, k_5^{''} = 0.3, k_6 = 0.1, J_5 = 0.3,\ k_7 = 1, k_8 = 0.25, J_7 = 0.001,$
$J_8 = 0.001.$

IE activation and inactivation:

$k_9 = 0.1, k_{10} = 0.04, J_9 = 0.01, J_{10} = 0.01.$

Rum1 synthesis, degradation and inhibition:

$k_{11} = 0.1, k_{12} = 0.01, k_{12}^{'} = 1, k_{12}^{'} \quad ' = 3, K_{diss}0 = .001.$

SK synthesis and degradation:

$k_{13} = 0.1, k_{14} = 0.1.$

TF activation and inactivation:

$k_{15} = 1.5, k_{16}^{'} = 1, k_{16}^{''} = 2, J_{15} = 0.01, J_{16} = 0.01.$

Wee1 activation and inactivation:

$V_{awee} = 0.25, V_{iwee} = 1, J_{awee} = 0.01, J_{iwee} = 0.01.$

Cdc25 activation and inactivation:

$V_{a25} = 1, V_{i25} = 0.25, J_{a25} = 0.01, J_{i25} = 0.01.$

Rate of tyr-phosphorylation and dephosphorylation:

$k_{wee}^{'} = 0.15, k_{wee}^{''} = 1.3, k_{25} \quad ' = 0.05, k_{25}^{''} = 5.$

Growth rate:

$\mu = 0.005.$

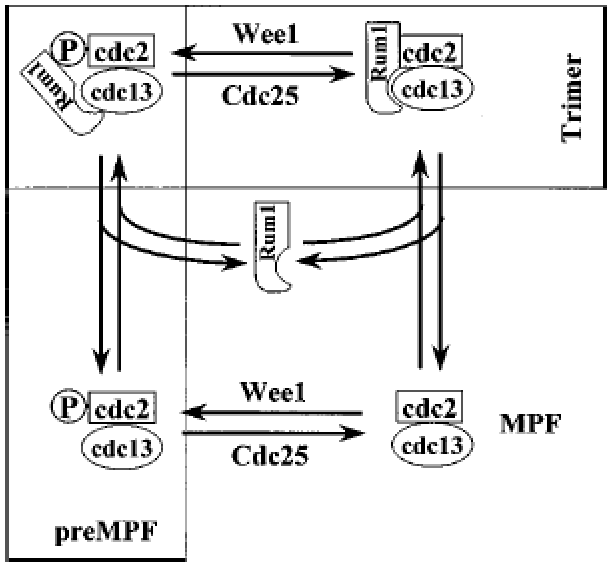


Figure 4.5: Rum1 binding to Cdc2/Cdc13 dimers. It is assumed that Rum1 binds to both active (MPF) and tyrosine-phosphorylated Cdc2/Cdc13 dimers. If association and dissociation of trimeric species is rapid, then (Eq.10 in orijinal) describes the equilibrium concentration of the total pool of trimers in terms of the total pools of Cdc13 and Rum1. The sum of the dimeric and trimeric tyrosine-phosphorylated forms is called preMPF [50].

one another until 15–20 minutes after exit from mitosis.

The MPF level fluctuates during the cycle among three different levels. Cells enter mitosis with high MPF activity. After a time delay, Slp1/APC is activated by the high MPF activity, initiating the degradation of Cdc13. As a consequence, MPF activity drops, Ste9/APC activates, and Cdc13 degradation accelerates. Loss of MPF relieves the inhibition on the TF responsible for the synthesis of the cyclin subunit of the SK. Because newborn wild-type cells are already large enough to pass Start, they activate the TF for SK after a very short G1 phase. Consequently, the level of SK increases abruptly and the G1 enemies of MPF

(Ste9/APC and Rum1) cannot stay. Actually, G1 is so short that Rum1 does not have time to come up, which is consistent with experimental observations [12, 50]. As soon as Ste9 gets inactivated, the Cdc13 level rises and the cell passes the G1/S transition. However, SK does not inactivate the third enemy, Wee1, which phosphorylates Cdc2/Cdc13. The phosphorylated form has reduced protein-kinase activity, which seems to be enough to initiate S phase but not mitosis. When the cell reaches a critical size, the positive feedbacks for G2/M transition turn on. Abrupt activation of MPF by Cdc25 drives the cell into mitosis [50].

To achieve the coordination, the cell-cycle control system has specific size checkpoints where the control system halts and waits until the cell has reached a critical size [3].

The cell cycle control of fission yeast is accomplished by the change of concentration of some molecules as discussed. The concentration and activation of these molecules will change in different ways as the cell gets bigger. If, for example, Wee1 protein were to become diluted relative to Cdc25 protein as a result of cell growth, growth would swing the regulatory balance in favour of MPF activation, and growth beyond a critical size would trigger an autocatalytic MPF explosion. [3, 17, 21, 23, 35, 48]. The differential equations seen in Table I are numerically simulated. The code of this work is presented in Appendix. In Figures 4.6, 4.7, 4.8, 4.9, and 4.10 figures obtained by numerical simulation of this ODE model can be seen.

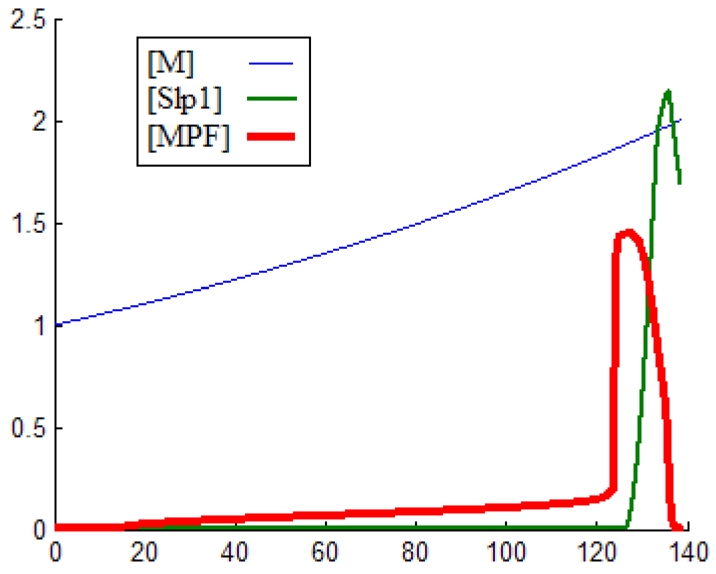


Figure 4.6: Numerical simulation of M, Slp1, and MPF in ODE model.

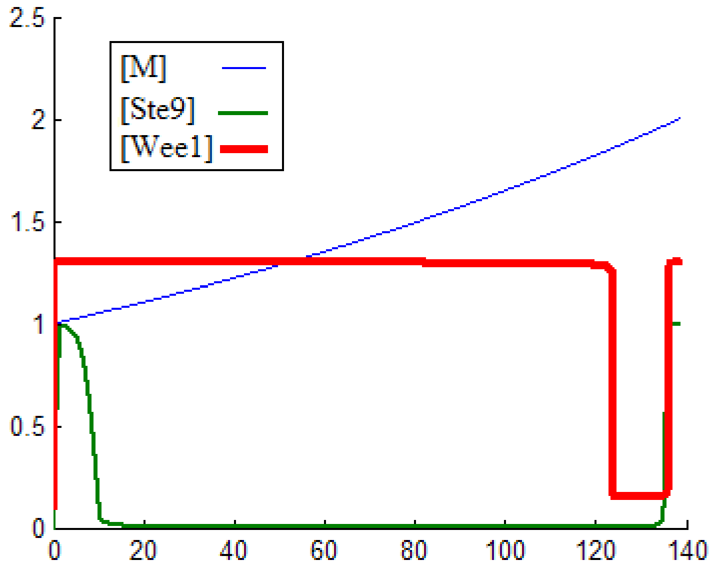


Figure 4.7: Numerical simulation of M, Ste9, and Wee1 (kwee) in ODE model.

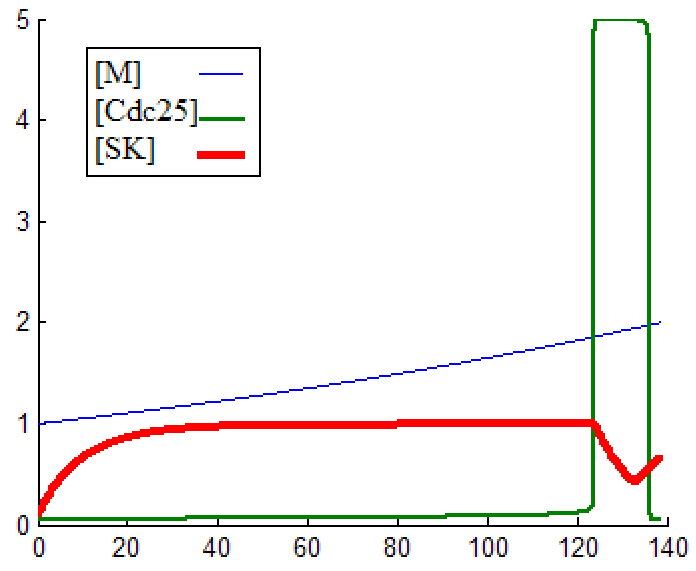


Figure 4.8: Numerical simulation of M, Cdc25 (k25), and SK in ODE model.

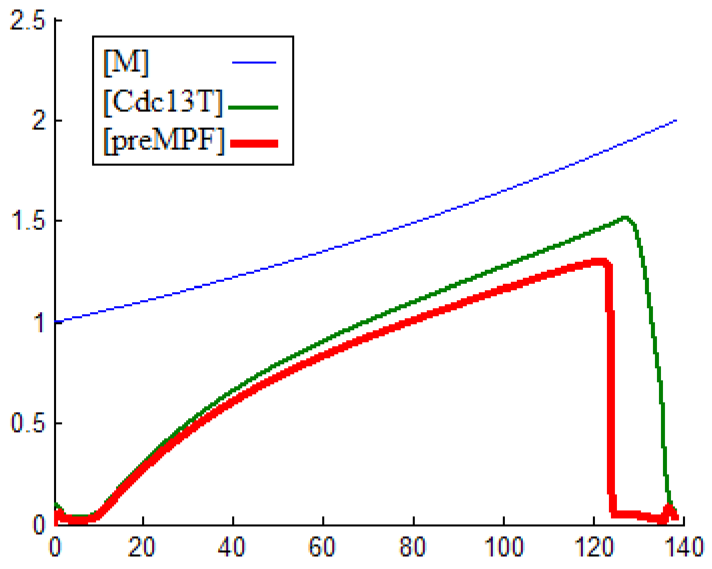


Figure 4.9: Numerical simulation of M, $Cdc13_T$, and preMPF in ODE model.

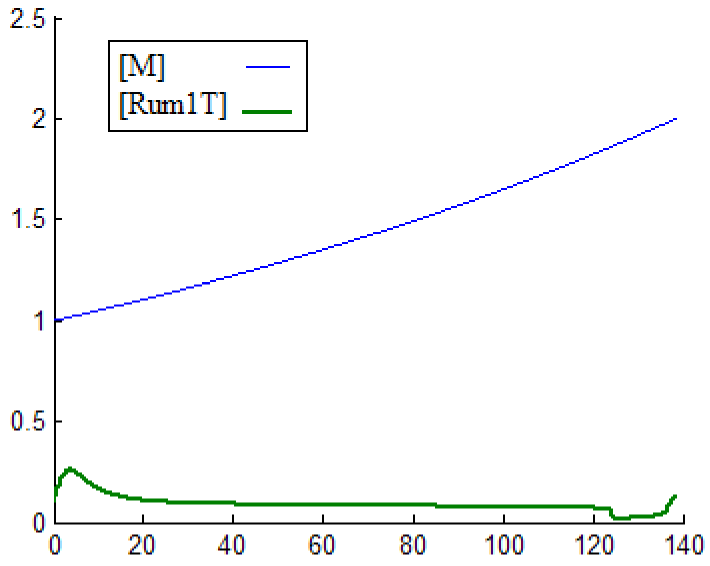


Figure 4.10: Numerical simulation of M and Rum1_T in ODE model.

4.2.2 Piecewise Linear Model with Delay

In this work, a piecewise linear system with delay on piecewise constant part model is used to model the fission yeast cell cycle control. The evolution of concentrations of molecules governing cell cycle of wild-type fission yeast is approximated by piecewise linear system with delay modules. Each module approximates the change of concentration of a molecule. The external input of each module is coming from an other module or modules. The state transitions occurs when external variables exceeds a threshold. The piecewise linear model is obtained according to the assumption: the subsystems can be approximated by an action ("when x goes down y goes up", etc.), and possibly a time delay for this action as Olivia Eriksson, Yishao Zhuo and Jesper Tegner assumed in their model [20]. Delays in the system can be approximated by delays in state transitions.

For illustration, if A, B, and C are variables and the evolution of variable A is dependent on the states of B and C a module for A could be constructed as

$$\tfrac{d[A]}{dt} = \mu^A_{s(t)}[A] + 0[B] + 0[C] + k^A_{s(t)}$$

$$s(t) = F(Q_B([B](t)), Q_C([C](t)))$$

$$Q_B([B](t)) = i \text{ if } h^B_i < [B](t) \leq h^B_{i+1}$$

$$Q_C([C](t)) = i \text{ if } h^C_i < [B](t) \leq h^C_{i+1} \text{ for } \ i = 1, 2, ..., n$$

If variable B has n discrete states and variable C has n discrete states then module A will have n^2 states. In general, number of states = $2^{(n_2)}3^{(n_3)}...m^{(n_m)}$ where n_i is the number of variables having i states ($i = 2, 3, ..., m$). In this work $\mu_{s(t)}$ for all modules is take to be -1 at each state. Therefore, only $k_{s(t)}$ is changing. By using such a modular system the cell cycle of fission yeast could be approximated. The values k and h are obtained according to numerical simulations of ODE model presented.

Firstly, cell mass (M) is approximated. In this approximation an external variable named $[EC]$ is included, denoting Environmental Conditions. A cell's growth rate is a mercy of the environment varying according to the supply of nutrients and other factors [3, 35]. Therefore, a cell's growth rate is proportional to its environment. This, information is used in this approximation and $[EC]$ is included as an external input to the system M. In this work, $[EC]$ is assumed to be a real variable between 0 and 2. Additionally, a threshold for $[EC]$ is assumed to be 1. If $[EC]$ is below its threshold 1, the environmental conditions are assumed to be in scarcity, and if $[EC]$ is above its threshold 1, the environmental conditions are assumed to be well enough for cell to grow with a normal growth rate. If more thresholds are included for $[EC]$ the approximation will be more realistic. However, in this work only one threshold is included to keep simplicity. The graph of the evolution of M in time is presented in Figure 4.12. For each discrete state q_1 and q_2 parameters μ_1,μ_2,n_1, and n_2 are used as seen in Figure 4.11. The parameters n_1 and n_2 are taken to be zero in this model. It means, the variable $[EC]$ does not have an effect on the evolution of the cell mass but this variable is used to determine the state of the system. According to state of the variable $[EC]$ the state of the system is determined. If $[EC] \geq 1$, the system is in the state q_1 and if $[EC] < 1$, the system is in the state q_2. The parameter μ_1 is the multiplier of state q_1 in which the evolution is in a nutrient rich environment so $\mu_1 = 0.005$ and the parameter μ_2 is the multiplier of state q_2 in which the evolution is in a nutrient scarce environment so $\mu_2 = 0.0025$. These values are assumed according to the work of Bela Novak, Zsuzsa Pataki, Andrea Ciliberto, and John J. Tyson on wild-type fission yeast cell division cycle model [50]. In the state q_2 the cell's growth rate decreases due to environmental conditions. Additionally, $[M] \geq 1$ is necessary for beginning of the cell division. The system could be approximated with:

$Q = \{q_1, q_2\}$ and $[\mathbf{M}] = \mathbf{R}, [\mathbf{EC}] \in [\mathbf{0}, \mathbf{2}]$;

$Init = Q \times \{[M] \in [\mathbf{M}] : [\mathbf{M}] \geq \mathbf{1}\} \times \{[\mathbf{EC}] \in [\mathbf{EC}] : ([\mathbf{EC}] \geq \mathbf{0})$
$\wedge([EC] \leq 2)\}$;

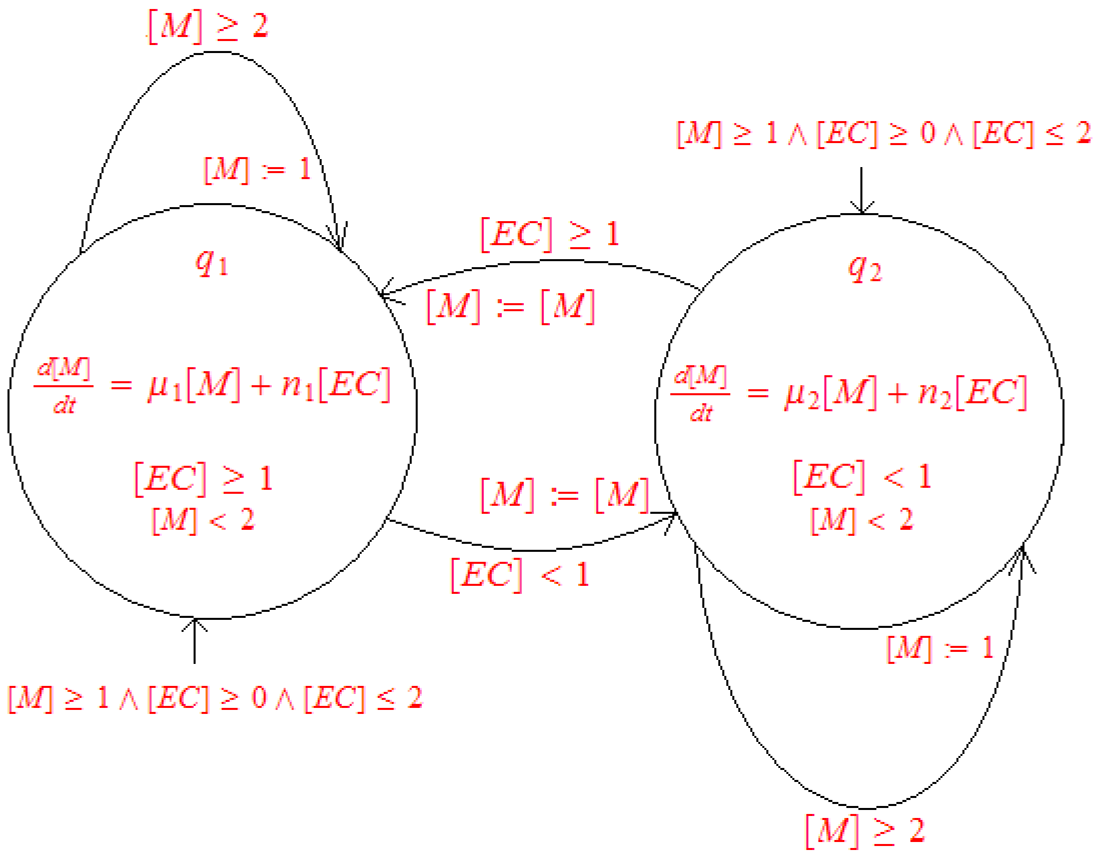
$f(q_1, [M]) = (\mu_1[M] + n_1[EC])$ and
$f(q_2, [M]) = (\mu_2[M] + n_2[EC]), \mu_1, n_1, \mu_2, n_2 \geq 0$;

$I(q_1) = \{[M] \in [\mathbf{M}],[\mathbf{EC}] \in [\mathbf{EC}] : ([\mathbf{M}] < \mathbf{2}) \wedge ([\mathbf{EC}] \geq \mathbf{1})\}$ and

$I(q_2) = \{[M] \in [\mathbf{M}], [EC] \in [\mathbf{EC}] : ([\mathbf{M}] < \mathbf{2}) \wedge ([\mathbf{EC}] < \mathbf{1})\}$;

$E = \{(q_1, q_2), (q_2, q_1), (q_1, q_1), (q_2, q_2)\}$, $G(q_1, q_1) = \{[M] \in [\mathbf{M}] : [\mathbf{M}] \geq \mathbf{2}\}$, $G(q_2, q_2) = \{[M] \in [\mathbf{M}] : [\mathbf{M}] \geq \mathbf{2}\}$, $G(q_1, q_2) = \{[EC] \in [\mathbf{EC}] : [\mathbf{EC}] \geq \mathbf{1}\}$, $G(q_2, q_1) = \{[EC] \in [\mathbf{EC}] : [\mathbf{EC}] < \mathbf{1}\}$

$R((q_1, q_1), [M]) = R((q_2, q_2), [M]) = \{1\}$ and $R((q_1, q_2), [M]) = R((q_2, q_1), [M]) = \{[M]\}$.



$[M] \geq 2$
$[M] := 1$
q_1
$\frac{d[M]}{dt} = \mu_1[M] + n_1[EC]$
$[EC] \geq 1$
$[M] < 2$
$[M] \geq 1 \wedge [EC] \geq 0 \wedge [EC] \leq 2$
$[EC] \geq 1$
$[M] := [M]$
$[M] := [M]$
$[EC] < 1$
$[M] \geq 1 \wedge [EC] \geq 0 \wedge [EC] \leq 2$
q_2
$\frac{d[M]}{dt} = \mu_2[M] + n_2[EC]$
$[EC] < 1$
$[M] < 2$
$[M] := 1$
$[M] \geq 2$

Figure 4.11: Network representaton of M.

The model M is numerically simulated and the graph obtained by this simulation is as seen in Figure 4.12. As seen in Figure 4.12, after the external variable $[EC]$ drops to 0.5 at $t = 200$, the cell's growt rate decreases.

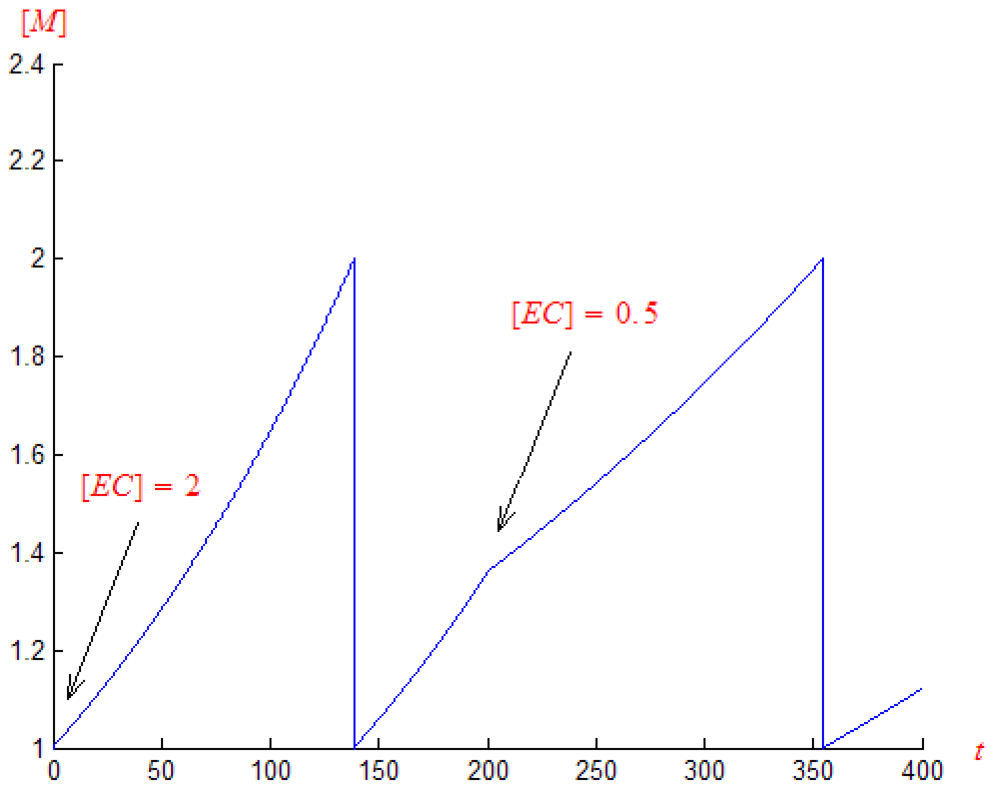


Figure 4.12: The M-model simulation.

The $[Cdc25]$ depends on the state of $[MPF]$. Here, $[Cdc25]$ is represented as $k25$ in the ODE model. According to numerical simulations, the dynamics of $[k25]$ could be approximated as

$$\frac{d[k25]}{dt} = \mu_{s(t)}^{k25}[k25] + 0[MPF] + k_{s(t)}^{k25},$$

$$s(t) = \{ \begin{array}{c} s_1 \text{ if } [MPF] < 0.25 \\ s_2 \text{ if } [MPF] \geq 0.25 \end{array}, \text{ and}$$

$k_{s_1}^{k25} = 0, k_{s_2}^{k25} = 5$, and $\mu_{s_1,s_2}^{k25} = -1$.

Figure 4.13 is the simulation of the k25-module.

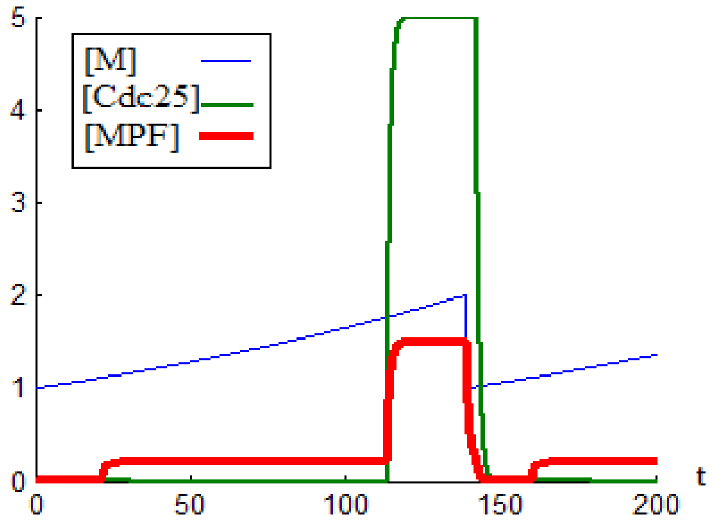


Figure 4.13: Numerical simulation of k25-module.

Additionally, $[Slp1]$ also depends only on the state of $[MPF]$. However, there is a delay in this module as mentioned before. The delay is found by numerical simulations. The Figure 4.14 represents the step response of $[Slp1]$ to $[MPF]$. It is obvious in this figure $[Slp1]$ changes depending on $[MPF]$ after a time delay.

According to the Figure 4.14. the delay approximated to be 15. According to numerical simulations the dynamics of $[Slp1]$ could be approximated as

$$\frac{d[Slp1]}{dt} = \mu_{s(t)}^{Slp1}[Slp1] + 0[MPF] + k_{s(t)}^{Slp1},$$

$$s(t) = \{ \begin{array}{c} s_1 \text{ if } [MPF](t-15) < 0.25 \\ s_2 \text{ if } [MPF](t-15) \geq 0.25 \end{array}, \text{ and}$$

$k_{s_1}^{Slp1} = 0, k_{s_2}^{Slp1} = 2.5$, and $\mu_{s_1,s_2}^{Slp1} = -1$.

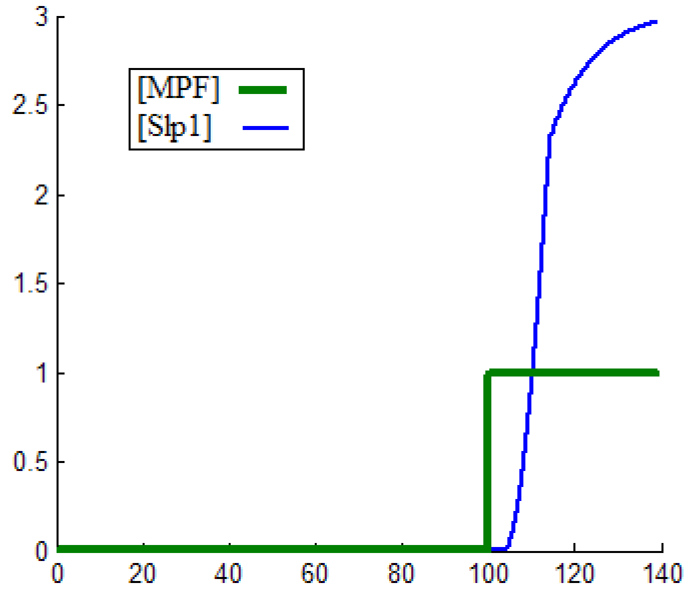


Figure 4.14: The step response of the $[Slp1]$ to $[MPF]$.

Figure 4.15 is obtained after the numerical simulation of Slp1-module.

$[Wee1]$ which is represented by kwee in ODE module only depends on $[MPF]$. Therefore, it can be approximated as

$$\frac{d[kwee]}{dt} = \mu^{kwee}_{s(t)}[kwee] + 0[MPF] + k^{kwee}_{s(t)},$$

$$s(t) = \{ \begin{array}{c} s_1 \text{ if } [MPF] < 0.25 \\ s_2 \text{ if } [MPF] \geq 0.25 \end{array}, \text{ and}$$

$$k^{kwee}_{s_1} = 1.3, k^{kwee}_{s_2} = 0.2, \text{ and } \mu^{kwee}_{s_1,s_2} = -1.$$

Figure 4.16 is obtained after the numerical simulation of kwee-module.

The variable $[SK]$ depends on the state of $[M]$ and $[MPF]$. An approximation to evolution of this variable can be constructed as

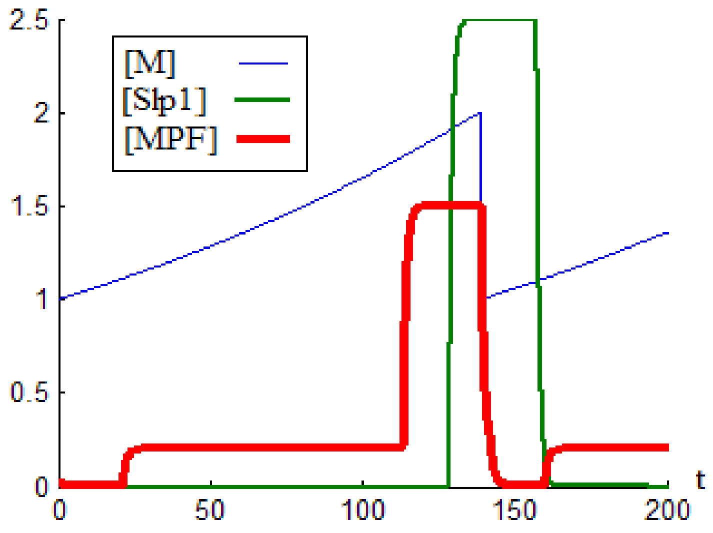


Figure 4.15: Numerical simulation of Slp1-module.

$$\frac{d[SK]}{dt} = \mu^{SK}_{s(t)}[SK] + 0[MPF] + 0[M] + k^{SK}_{s(t)},$$

$$s(t) = \{ \begin{array}{l} s_1 \text{ if } [MPF] < 0.25 \wedge [M] < 1.1 \\ s_2 \text{ if } [MPF] < 0.25 \wedge [M] \geq 1.1 \\ s_3 \text{ if } [MPF] \geq 0.25 \wedge [M] < 1.1 \\ s_4 \text{ if } [MPF] \geq 0.25 \wedge [M] \geq 1.1 \end{array} \text{, and}$$

$$k^{SK}_{s_1} = 0.4, k^{SK}_{s_2} = 0.4, k^{SK}_{s_3} = 1.1, k^{SK}_{s_3} = 0.4, \text{ and } \mu^{SK}_{s_1,s_2,s_3,s_4} = -1.$$

Figure 4.17 is the numerical simulation of SK-module.

Here, $[Ste9]$ depends on $[Slp1]$, $[SK]$, and $[MPF]$. As mentioned before, $[Slp1]$ has positive effect on $[Ste9]$ while $[SK]$ and $[MPF]$ has negative effect on $[Ste9]$. The dynamics of $[Ste9]$ can be described as

$$\frac{d[Ste9]}{dt} = \mu^{Ste9}_{s(t)}[Ste9] + 0[MPF] + 0[SK] + 0[Slp1] + k^{Ste9}_{s(t)},$$

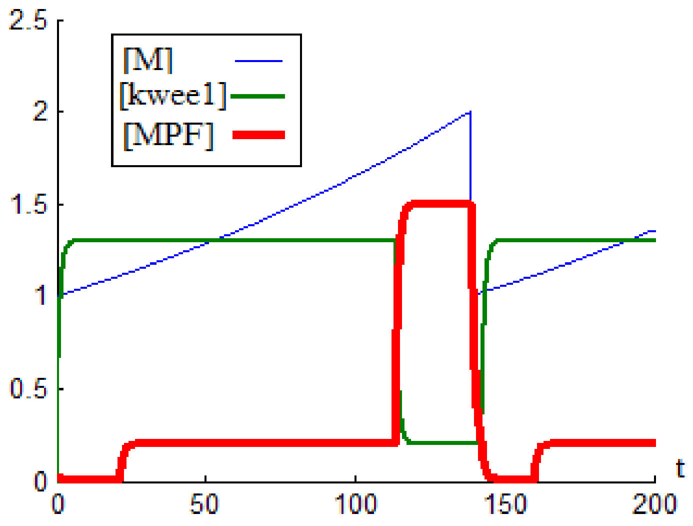


Figure 4.16: Numerical simulation of kwee-module.

$$s(t) = \begin{cases} s_1 \text{ if } [Slp1] \geq 0.1 \wedge [MPF] \geq 0.3 \wedge [SK] \geq 0.7 \\ s_2 \text{ if } [Slp1] \geq 0.1 \wedge [MPF] \geq 0.3 \wedge [SK] < 0.7 \\ s_3 \text{ if } [Slp1] \geq 0.1 \wedge [MPF] < 0.3 \wedge [SK] \geq 0.7 \\ s_4 \text{ if } [Slp1] \geq 0.1 \wedge [MPF] < 0.3 \wedge [SK] < 0.7 \\ s_5 \text{ if } [Slp1] < 0.1 \wedge [MPF] \geq 0.3 \wedge [SK] \geq 0.7 \\ s_6 \text{ if } [Slp1] < 0.1 \wedge [MPF] \geq 0.3 \wedge [SK] < 0.7 \\ s_7 \text{ if } [Slp1] < 0.1 \wedge [MPF] < 0.3 \wedge [SK] \geq 0.7 \\ s_8 \text{ if } [Slp1] < 0.1 \wedge [MPF] < 0.3 \wedge [SK] < 0.7 \end{cases}, \text{ and}$$

$k_{s_1}^{Ste9} = 0, k_{s_2}^{Ste9} = 1, k_{s_3}^{Ste9} = 1, k_{s_4}^{Ste9} = 1, k_{s_5}^{Ste9} = 0, k_{s_6}^{Ste9} = 1,$
$k_{s_7}^{Ste9} = 0, k_{s_8}^{Ste9} = 1$, and $\mu_{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8}^{Ste9} = -1$.

Figure 4.18 is obtained after the numerical simulation of Ste9-module.

The variable $[Rum1_T]$ depends on $[SK]$ and $[MPF]$. Evolution of $[Rum1_T]$

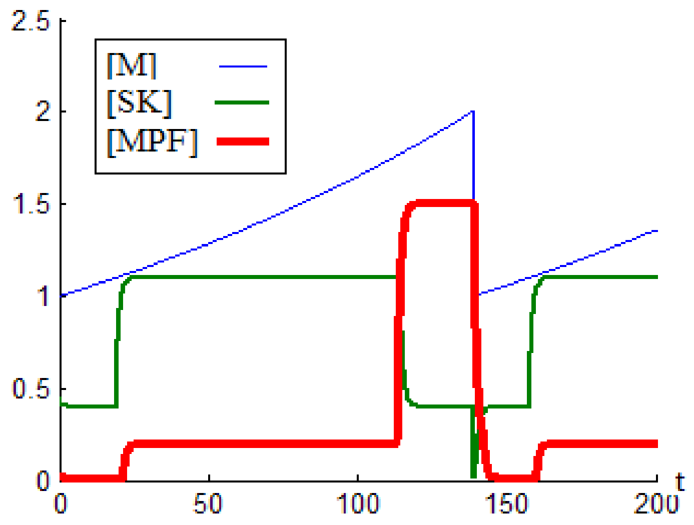


Figure 4.17: Numerical simulation of SK-module.

can be approximated as

$$\frac{d[Rum1_T]}{dt} = \mu_{s(t)}^{Rum1_T}[Rum1_T] + 0[MPF] + 0[SK] + k_{s(t)}^{Rum1_T},$$

$$s(t) = \{ \begin{array}{l} s_1 \text{ if } [MPF] \geq 0.3 \wedge [SK] \geq 0.8 \\ s_2 \text{ if } [MPF] \geq 0.3 \wedge [SK] < 0.8 \\ s_3 \text{ if } [MPF] < 0.3 \wedge [SK] \geq 0.8 \\ s_4 \text{ if } [MPF] < 0.3 \wedge [SK] < 0.8 \end{array},$$

$k_{s_1}^{Rum1_T} = 0.02, k_{s_2}^{Rum1_T} = 0.02, k_{s_3}^{Rum1_T} = 0.11, k_{s_4}^{Rum1_T} = 0.3$, and

$\mu_{s_1,s_2,s_3,s_4}^{Rum1_T} = -1.$

Figure 4.19 is the numerical simulation of Rum1$_T$-module.

The variable $[Cdc13_T]$ depends on $[M]$, $[Ste9]$, and $[Slp1]$. The evolution can be approximated as

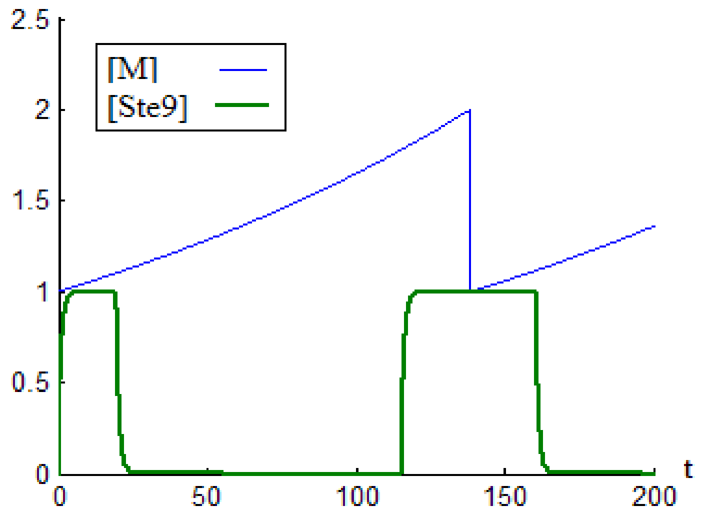


Figure 4.18: Numerical simulation of Ste9-module.

$$\frac{d[Cdc13_T]}{dt} = \mu_{s(t)}^{Cdc13_T}[Cdc13_T] + 0[M] + 0[Ste9] + 0[Slp1] + k_{s(t)}^{Cdc13_T},$$

$$s(t) = \{ \begin{array}{l} s_1 \text{ if } [M] \geq 1.75 \wedge [Ste9] \geq 0.5 \wedge [Slp1] \geq 1 \\ s_2 \text{ if } [M] \geq 1.75 \wedge [Ste9] \geq 0.5 \wedge [Slp1] < 1 \\ s_3 \text{ if } [M] \geq 1.75 \wedge [Ste9] < 0.5 \wedge [Slp1] \geq 1 \\ s_4 \text{ if } [M] \geq 1.75 \wedge [Ste9] < 0.5 \wedge [Slp1] < 1 \\ s_5 \text{ if } [M] < 1.75 \wedge [Ste9] \geq 0.5 \wedge [Slp1] \geq 1 \\ s_6 \text{ if } [M] < 1.75 \wedge [Ste9] \geq 0.5 \wedge [Slp1] < 1 \\ s_7 \text{ if } [M] < 1.75 \wedge [Ste9] < 0.5 \wedge [Slp1] \geq 1 \\ s_8 \text{ if } [M] < 1.75 \wedge [Ste9] < 0.5 \wedge [Slp1] < 1 \end{array},$$

$k_{s_1}^{Cdc13_T} = 0.1, k_{s_2}^{Cdc13_T} = 0.3, k_{s_3}^{Cdc13_T} = 0.6, k_{s_4}^{Cdc13_T} = 0.5,$
$k_{s_5}^{Cdc13_T} = 1.5, k_{s_6}^{Cdc13_T} = 1.5, k_{s_7}^{Cdc13_T} = 1.5, k_{s_8}^{Cdc13_T} = 1.5$, and

$\mu_{s_1,s_2,s_3,s_4,s_5,s_6,s_7,s_8}^{Cdc13_T} = -1.$

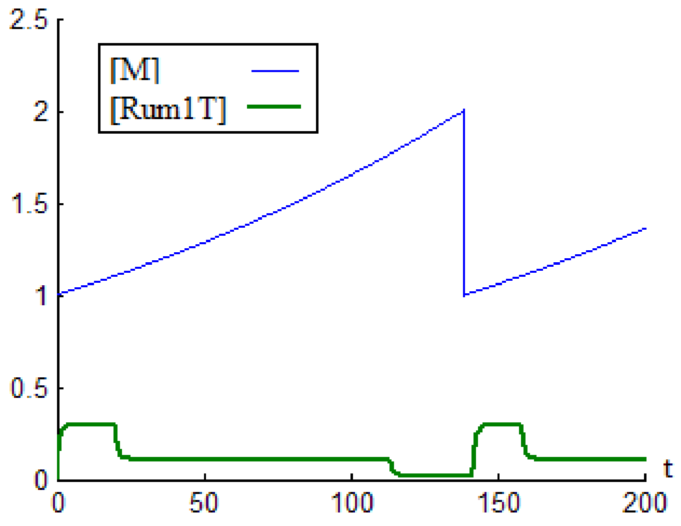


Figure 4.19: Numerical simulation of $\mathrm{Rum1_T}$-module.

Figure 4.20 is the numerical simulation of $Cdc13_T$-module.

The variable $[preMPF]$ depends on $[Cdc13_T]$, $[kwee]$, $[k25]$, $[Ste9]$, and $[Slp1]$. The evolution can be approximated as

$\frac{d[preMPF]}{dt} = \mu_{s(t)}^{preMPF}[preMPF] + 0[Cdc13_T] + 0[kwee] + 0[k25] + 0[Ste9] + 0[Slp1] + k_{s(t)}^{preMPF}$

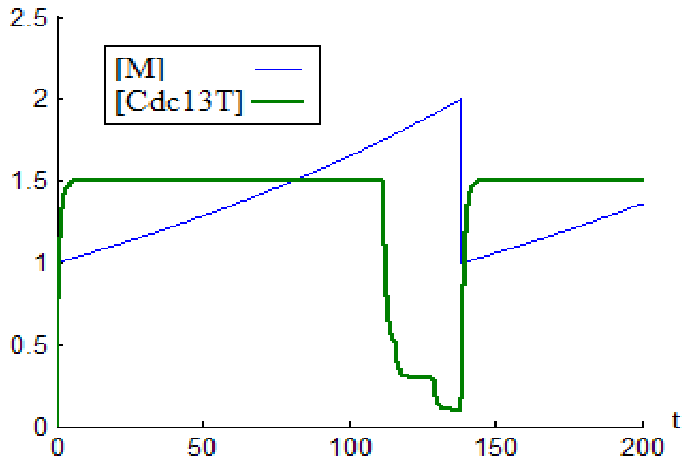


Figure 4.20: Numerical simulation of $\mathrm{Cdc13_T}$-module.

$$s(t) = \begin{cases}
s_1 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_2 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_3 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_4 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4 \\
s_5 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_6 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_7 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_8 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4 \\
s_9 \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_{10} \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_{11} \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_{12} \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4 \\
s_{13} \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_{14} \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_{15} \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_{16} \text{ if } [Cdc13_T] \geq 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4 \\
s_{17} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_{18} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_{19} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_{20} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4 \\
s_{21} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_{22} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_{23} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_{24} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] \geq 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4 \\
s_{25} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_{26} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_{27} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_{28} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] \geq 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4 \\
s_{29} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] \geq 4 \\
s_{30} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] \geq 0.8 \wedge [k25] < 4 \\
s_{31} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] \geq 4 \\
s_{32} \text{ if } [Cdc13_T] < 0.5 \wedge [Ste9] < 0.5 \wedge [kwee] < 1 \wedge [Slp1] < 0.8 \wedge [k25] < 4
\end{cases}$$

Additionally, we have

$k_{s_1}^{preMPF} = 1.5, k_{s_2}^{preMPF} = 1.4, k_{s_3}^{preMPF} = 1.5, k_{s_4}^{preMPF} = 1.4,$
$k_{s_5}^{preMPF} = 1.5, k_{s_6}^{preMPF} = 1.4, k_{s_7}^{preMPF} = 1.5, k_{s_8}^{preMPF} = 1.4,$

$k_{s_9}^{preMPF} = 1.5, k_{s_{10}}^{preMPF} = 1.4, k_{s_{11}}^{preMPF} = 1.5, k_{s_{12}}^{preMPF} = 1.4,$
$k_{s_{13}}^{preMPF} = 1.5, k_{s_{14}}^{preMPF} = 1.4, k_{s_{15}}^{preMPF} = 1.5, k_{s_{16}}^{preMPF} = 1.4,$

$k_{s_{17}}^{preMPF} = 0.3, k_{s_{18}}^{preMPF} = 0.3, k_{s_{19}}^{preMPF} = 0.3, k_{s_{20}}^{preMPF} = 0.3,$
$k_{s_{21}}^{preMPF} = 0.5, k_{s_{22}}^{preMPF} = 0.5, k_{s_{23}}^{preMPF} = 0.6, k_{s_{24}}^{preMPF} = 0.6,$

$k_{s_{25}}^{preMPF} = 0.3, k_{s_{26}}^{preMPF} = 0.3, k_{s_{27}}^{preMPF} = 0.3, k_{s_{28}}^{preMPF} = 0.3,$
$k_{s_{29}}^{preMPF} = 0.6, k_{s_{30}}^{preMPF} = 0.6, k_{s_{31}}^{preMPF} = 0.8, k_{s_{32}}^{preMPF} = 0.8,$ and $\mu_{s_{1-32}}^{preMPF} = -1.$

Figure 4.21 is the numerical simulation of preMPF-module.

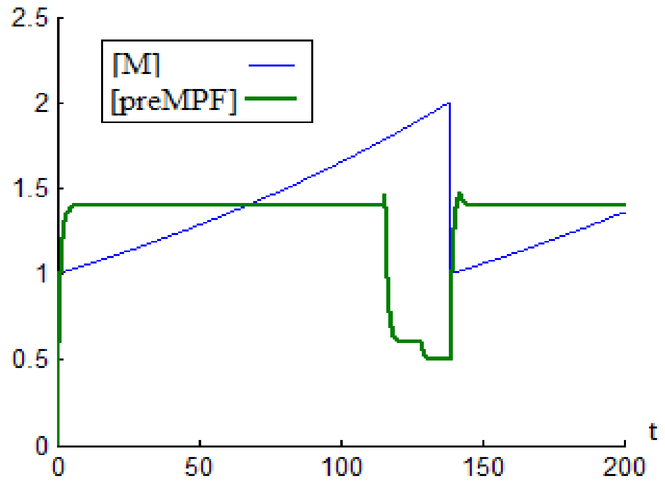


Figure 4.21: Numerical simulation of preMPF-module.

The variable $[MPF]$ depends on $[Cdc13_T]$, $[preMPF]$, and $[Rum1_T]$. The evolution can be approximated as

$$\frac{d[MPF]}{dt} = \mu_{s(t)}^{MPF}[MPF] + 0[Cdc13_T] + 0[preMPF] + 0[Rum1_T] + k_{s(t)}^{MPF},$$

$$s(t) = \{ \begin{array}{l} s_1 \text{ if } [Cdc13_T] \geq 0.7 \wedge [preMPF] \geq 0.5 \wedge [Rum1_T] \geq 0.15 \\ s_2 \text{ if } [Cdc13_T] \geq 0.7 \wedge [preMPF] \geq 0.5 \wedge [Rum1_T] < 0.15 \\ s_3 \text{ if } [Cdc13_T] \geq 0.7 \wedge [preMPF] < 0.5 \wedge [Rum1_T] \geq 0.15 \\ s_4 \text{ if } [Cdc13_T] \geq 0.7 \wedge [preMPF] < 0.5 \wedge [Rum1_T] < 0.15 \\ s_5 \text{ if } [Cdc13_T] < 0.7 \wedge [preMPF] \geq 0.5 \wedge [Rum1_T] \geq 0.15 \\ s_6 \text{ if } [Cdc13_T] < 0.7 \wedge [preMPF] \geq 0.5 \wedge [Rum1_T] < 0.15 \\ s_7 \text{ if } [Cdc13_T] < 0.7 \wedge [preMPF] < 0.5 \wedge [Rum1_T] \geq 0.15 \\ s_8 \text{ if } [Cdc13_T] < 0.7 \wedge [preMPF] < 0.5 \wedge [Rum1_T] < 0.15 \end{array},$$

$k_{s_1}^{MPF} = 0.01, k_{s_2}^{MPF} = 0.2, k_{s_3}^{MPF} = 0.01, k_{s_4}^{MPF} = 0.2,$ $k_{s_5}^{MPF} = 0.01,$ and

$k_{s_6}^{MPF} = 1.5, k_{s_7}^{MPF} = 0.01, k_{s_8}^{MPF} = 0.01,$, and $\mu_{s_{1-8}}^{MPF} = -1.$

Figure 4.22 is the numerical simulation of MPF-module.

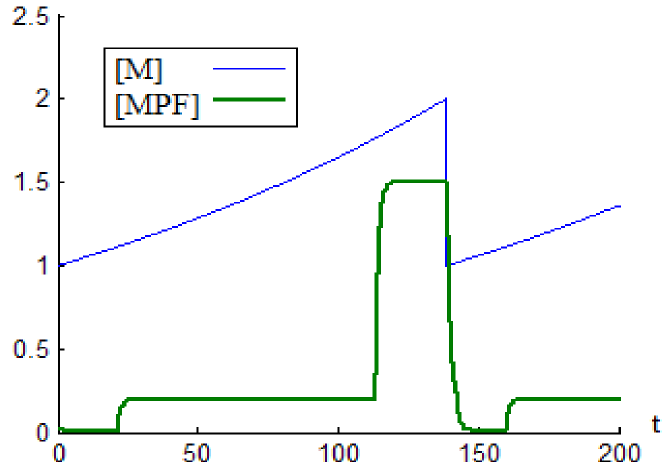


Figure 4.22: Numerical simulation of MPF-module.

Finally, all modules describe the dynamics of the cell cycle of fission yeast.

And the whole system could be written as

$\frac{d[M]}{dt} = \mu^M_{s(t)}[M] + 0[EC]$,

$\frac{d[y]}{dt} = M_{s(t)}y + X_{s(t)}y + k_{s(t)}$ where

$\mathbf{S} = \{\mathbf{s^{k25}(t)} \cup \mathbf{s^{Slp1}(t)} \cup \mathbf{s^{kwee}(t)} \cup \mathbf{s^{SK}(t)} \cup \mathbf{s^{Ste9}(t)} \cup \mathbf{s^{Rum1_T}(t)} \cup$
$s^{Cdc13_T}(t) \cup s^{preMPF}(t) \cup s^{MPF}(t)\}$,

$s(t) \in \mathbf{S}$,

$y = [k25\ Slp1\ kwee\ SK\ Ste9\ Rum1_T\ Cdc13_T\ preMPF\ MPF]'$,

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$M = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}.$$

4.3 Advantages of Piecewise Linear Model with Delay

In this section, a nonlinear system is approximated by a piecewise linear system by subdividing system and by looking at the steady state dynamics. The relations between variables can be used directly to construct the model. The model will be an approximation to the process. By such approximations the system behavior can be observed more easily. Additionally, the number of variables affecting the system can be reduced. Basic relations can be considered and the internal processes can be omitted. Systems described by delay differential equations can also be approximated by this way. The stability of the piecewise linear system model with delays can be analyzed as described in previous section.

CHAPTER 5

APPROXIMATING DDE BY PIECEWISE LINEAR SYSTEMS WITH DELAY

Delay-differential equations (DDEs) are a large and important class of dynamical systems. They often arise in either natural or technological control problems. In these systems, a controller monitors the state of the system, and makes adjustments to the system based on its observations. Since these adjustments can never be made instantaneously, a delay arises between the observation and the control action.

When we give initial conditions for finite-dimensional dynamical systems, we only need to specify a small set of numbers, namely, the initial values of the state variables, and perhaps the initial time in nonautonomous systems. In order to solve a delay equation, we need more: At every time step, we have to look back to earlier values of x. We therefore need to specify an initial function which gives the behavior of the system prior to time 0 (assuming that we start at t =0). This function has to cover a period at least as long as the longest delay since we will be looking back in time that far [59].

Piecewise linear systems could be used in the approximation of delay differential equations. In this section general idea is presented with a simple example. Consider the delay differential equation:

$$\frac{dx}{dt} = f(x(t), x(t-\tau)),$$

The initial function is the function $x(t)$ defined on interval M $[-\tau, 0]$. This delay differential equation could be approximated by

$$\frac{dx}{dt} = f(x(t), k_s),$$

where $x_0 = x(0)$ is the initial value (value of the initial function $x(t)$ at t=0) and

$$k_s = \frac{ah_{i+1} + bh_i}{a+b},$$

$$s(t) = i \text{ if } h_{i+1} \geq x(t-\tau) > h_i,$$

where h_i is the threshold for i^{th} state. Here, k_s is a linear combination of i^{th} state's thresholds h_i and h_{i+1}, and it is an approximation of the value of $x(t-\tau)$. Actually, the simple linear combination

$$k_s = \frac{h_{i+1} + h_i}{2},$$

$$s(t) = i \text{ if } h_{i+1} \geq x(t-\tau) > h_i.$$

could be used $(i = 1, 2, ..., n)$, where n is the number of thresholds. The number of discrete states will be $n-1$. And, the dynamical system described by delay differential equation will evolve in $(h_1, h_n]$. Therefore, delay differential equations with arbitrary initial function could be approximated by using such a piecewise

Table 5.1: Data for initial function.

t	x	t	x	t	x
-1.00	0.6	-0.65	1.4	-0.30	2.8
-0.95	2.2	-0.60	0.1	-0.25	0.8
-0.90	1.2	-0.55	1.6	-0.20	1.5
-0.85	2.6	-0.50	2.5	-0.15	2.5
-0.80	0.8	-0.45	1.2	-0.10	0.8
-0.75	1.5	-0.40	2.2	-0.05	1.3
-0.70	2.6	-0.35	1.3	0.00	2.0

linear system.

In this work, a simple example is presented to make the idea clear. Consider the following delay differential equation:

$$\frac{dx}{dt} = -x(t-1)$$

with an arbitrary initial function $x(t)$ on $[-1,0]$.To construct an arbitrary initial function, arbitrary data seen in Table 5.1 are used.

The method of linear least squares [31] is used for curve fitting and the function

$$f(t) = 1.8947 + 5.6899t + 44.9378t^2 + 119.1792t^3$$
$$+131.0523t^4 + 52.1560t^5$$

is obtained at the end.

This function is taken as an arbitrary initial function for $[-1,0]$. The graph of the function is presented in Figure 5.1.

Therefore, consider

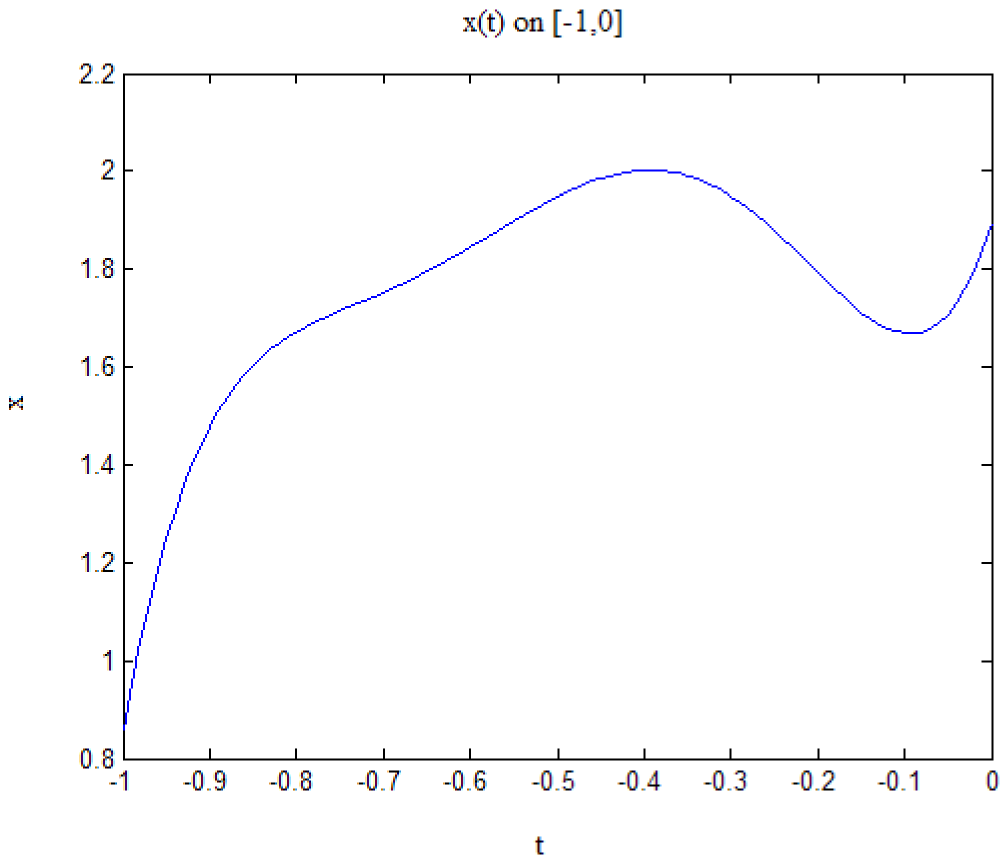


Figure 5.1: The initial function used in the example.

$\frac{dx}{dt} = -x(t-1)$ on $(0.5]$

with $x(t) = 1.8947 + 5.6899t + 44.9378t^2 + 119.1792t^3$

$+131.0523t^4 + 52.1560t^5$ on $[-1, 0]$.

Firstly, the MatLab function dde23 is used for solution. The code is presented at Appendix. The graph obtained after using dde23 is seen in Figure 5.2.

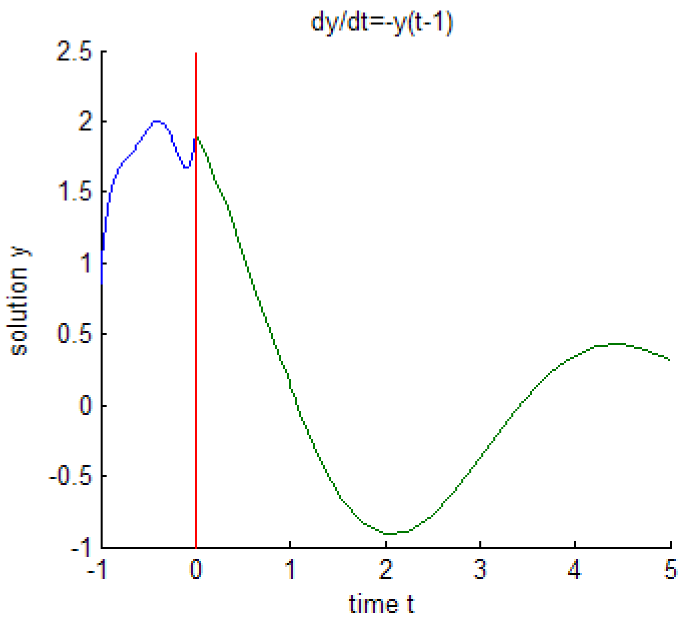


Figure 5.2: Solution obtained using dde23.

Then, the piecewise linear system approximation is used for the same problem. The code of piecewise linear system approximation is presented in the Appendix.

The thresholds are taken as:

$h_1 = -2$ and $h_n = 2.5$ where $n = 10$.

Therefore, 9 discrete states are obtained and $k_i = \frac{h_{i+1}+h_i}{2}$ for $i = 1, 2, ..., 9$.

The equation $\frac{dx}{dt} = -x(t-1)$ is approximated by $\frac{dx}{dt} = -k_{s(t)}$, where $s(t) = i$ if $h_{i+1} \geq x(t-1) > h_i$ for $i = 1, 2, ..., 9$. Here, the interval $(-2, 2.5]$ is decided for discrete state space by considering the real solution obtained by dde23 on interval $(0, 5]$. The interval for discrete states will change depending on the delay differential equation, initial function, and the time interval of the solution.

The graph obtained by piecewise linear system approximation is as seen in Figure 5.3.

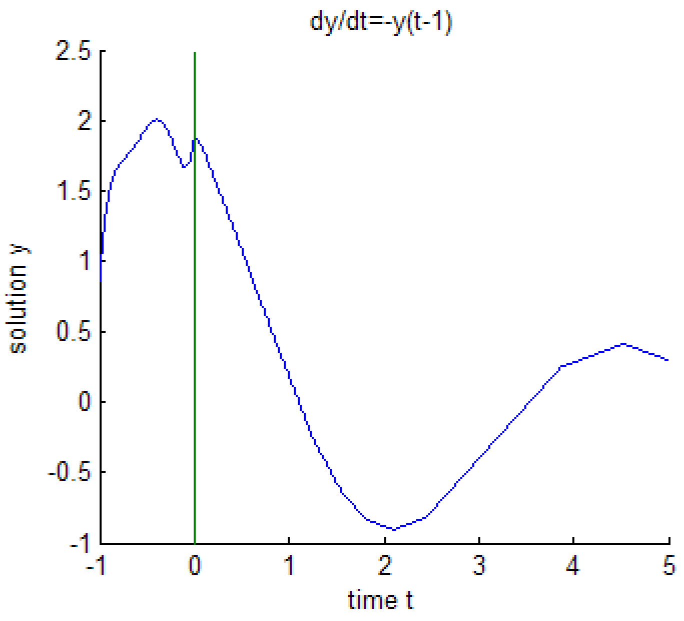


Figure 5.3: Solution obtained by piecewise linear system approximation.

This example illustrates the usage of piecewise linear system for the approximation of delay differential equations. It provides a simple numerical method for approximating delay differential equations. By this way, complex delay differential equations could be approximated. Additionally, adaptive sampling is an advantage for this numerical method.

CHAPTER 6

PROMISES AND CHALLENGES

Hybrid dynamical systems with delay can have different application areas and can provide advantages in terms of including the time delays. In this chapter, some possible applications are discussed briefly.

Firstly, using hybrid dynamical systems with delay in modelling traffic flow will bring some advantages. Microscopic vehicular traffic flow model can be described for continuous time change of kinematic variables of car i as

$$\frac{dv_i(t+T)}{dt} = a_i(j \geq i, t) + \xi_i(t), \; \frac{dx_i(t+T)}{dt} = v_i(t+T),$$

where a_i is a given acceleration function depending on kinematic variables of all leading cars $j > i$ and car i at t, ξ_i is a normal distributed random part of a_i , T reaction time [70]. The flow of each car depends to flow of leading cars. Therefore, a car's flow can be modelled as evolving in discrete states and continuously in each state. Governing differential equations are different in each state and the switching between these states is determined by an external input which is the states of leading cars in this case. Additionally, a time delay can be introduced to this systems for mechanical properties of cars. For example, a car needs time to change its velocity and acceleration. By this mean the efficiency of roads can be estimated and the problems in traffic can be determined. By using such a model, traffic can be simulated more realistically. Traffic simulation will bring us some advantages. It will help in improving the traffic systems. Which will improve capacity and the level-of-service, road safety, and the environmental impact of traffic. Traffic simulation softwares can use such a hybrid dynamical system with delay model. Figure 6.1 is a representation from a traffic simulation software [11].

Figure 6.1: A 3D view from a traffic simulation software (Aimsun urban demo [11]) .

Similarly, airspace systems can be modelled by the help of hybrid dynamical systems with delay. At its full extent, such systems are systems of overwhelming complexity. Thousands of aircrafts may be aloft at one time; hundreds of controllers are monitoring and directing them with the assistance of many communication and surveillance technologies [57]. In such a system, there are lots of entity affecting each other. It is a combination of continuous-time and discrete-event systems, their interactions, and possibly delays in these interactions. Therefore, modelling these systems by hybrid dynamical systems with delay will help to capture the whole system.

Another possible use of hybrid dynamical systems with delay is in modelling computer networks. Computer networks play an increasingly important role in our lives. These networks experience major problems due to traffic congestion. A lot of effort is spent on trying to reduce the problems with congestion. Congestion in computer networks is handled with various types of congestion control. The objective with research on control of networks is often to improve traffic throughput and to better accommodate different service demands. Today's congestion control is in most networks implemented as end-to-end protocols [22, 33, 55]. There is intensive research on modelling and simulation of the internet. It has been pointed out that classical network models from telecommunication based on Poisson modelling are not suitable for the internet [22, 54]. The general opinion in the network area is still that Internet modelling and simulation are open research problems [22, 26]. The standard modelling and simulation environment targeted at network research is the discrete-event simulator ns-2 [22, 79]. ns-2, which was originally developed at UC Berkeley, directly implements the Internet protocols and simulates individual packets. Another approach to network modelling is to use fluid models, i.e., to approximate packet transmission with a continuous flow and basically neglect the network protocols. Fluid models capture average transmission rates but ignore events such as packet drops. Fluid models are hence suitable for the study of steady-state behavior but not for evaluating transient phenomena. Therefore, hybrid models are preferred for better simulations. Additionally, hybrid dynamical systems with delay can be used for same purpose where delays in the system can be modelled in this case. Delays occurs because of the physical media in computer networks.

Sensor networks are another possible application area of hybrid dynamical systems with delay. Sensor networks (SNs) are gaining a role of importance in the research community. Embedded computers are well settled in our lives, in our houses, in our cars, and in our work environments. Embedded systems, by definition, interact with the physical world. They are sensors, actuators, and controllers which are programmed to perform specified functions. As the range of applications grows, the need arises to network several embedded systems to perform incrementally complex tasks. The automotive domain is an excellent example. Here, several embedded systems interact to provide a safe, comfortable driving experience [65]. PEGs can be seen as sensor networks. PEGs are a mathematical abstraction arising from numerous situations which addresses the problem of controlling a swarm of autonomous agents in the pursuit of one or more evaders. Typical examples are search and rescue operations, surveillance, localization and tracking of moving parts in a warehouse, and search and capture missions. In some cases, the evaders are actively avoiding detection, as in capture missions, whereas in other cases their motion is approximately random, as in rescue operations. Different versions of PEGs have been analyzed according to different frameworks and assumptions. Deterministic PEGs on finite graphs have been extensively studied [44, 53, 65]. In these games, the playing field is abstracted to be a finite set of nodes, and the allowed motions for the pursuers and evaders are represented by edges connecting nodes. An evader is captured if both the evader and one of the pursuers occupy the same node. A representation of usage of sensor networks is seen in Figure 6.2 and Figure 6.3 [65]. Sensor networks increase visibility in PEGs. Each pursuer shares what it sees to other pursuers. In these networks, communication delay is an important issue and so

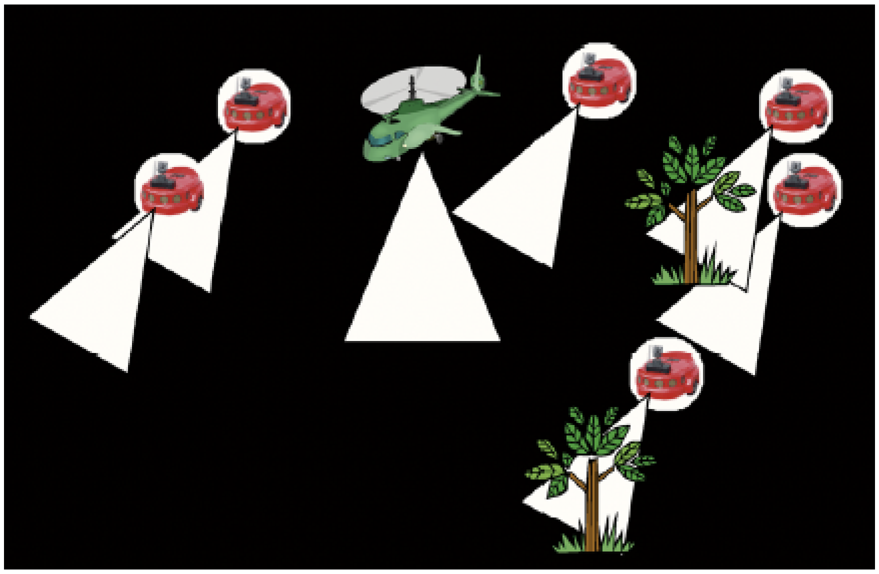


Figure 6.2: What pursuers really see [65].

hybrid dynamical systems with delay can be used to construct sensor network systems.

In conclusion, piecewise linear systems with delay on piecewise constant part can be used in a wide range of areas where continuous and discrete evolution exists with time delays. Traffic simulations, airspace systems, modelling computer protocols, and sensor network systems are some examples of application areas.

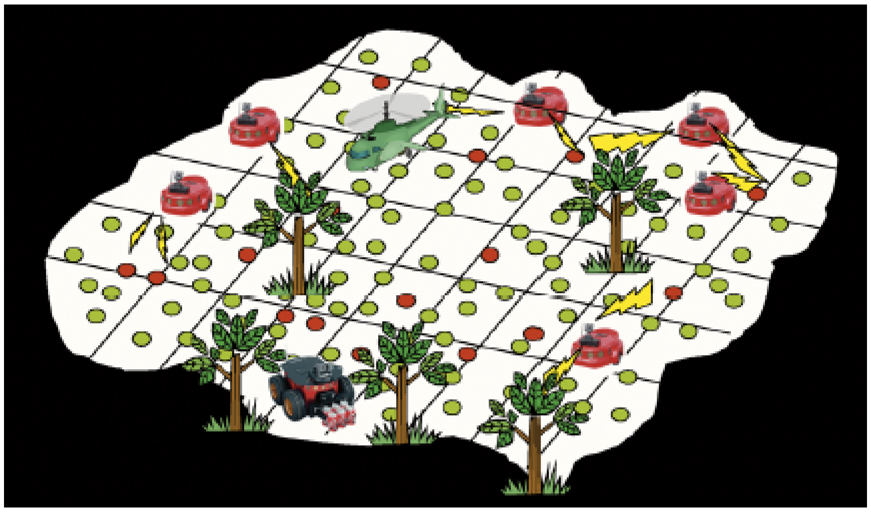


Figure 6.3: SN increases visibility [65].

CHAPTER 7

CONCLUSION AND DISCUSSION

In this study, piecewise linear systems with delay on the piecewise linear constant part which is a subclass of hybrid dynamical systems is investigated in terms of modelling functional dynamical systems. The stability analysis is performed first. It is seen that, asymptotically stable periodic solution can be obtained in these systems. Then, a model of cell cycle control of fission yeast is constructed. Piecewise linear systems with delay can be used especially in gene regulatory networks. Additionally, it is seen that piecewise linear systems with delay can be used in approximating the delay differential equations. Traffic simulations, airspace systems, modelling computer protocols, and sensor network systems are some other examples to application areas of hybrid dynamical systems with delay.

In this work, we showed that it is possible to analyze the stability of periodic attractors of piecewise linear system by constructing Poincare map and analyzing the stability of fixed point of this map. A simple 2-dimensional example is presented for stability analysis. For n - dimensional systems a map on $n-1$

dimensional hyperplane can be constructed and the stability of fixed point of this map can be investigated.

Complex dynamics of systems including delay can be observed in a simpler way and first principles models of natural phenomena can be approximated by piecewise linear systems. Being able to use such piecewise linear systems in modelling of complex dynamics comes with some advantages like handling perturbations and environmental factors as inputs. Complex systems can be analyzed with reasonable computational resources. Additionally, all the internal processes can be modelled which is not always possible with ODE models.

Constructing DDE models of gene regulatory networks and investigating the dependence of future behavior to the initial function by using these models is a promising challenge.

REFERENCES

[1] Akhmetov, M., *Asymptotic representation of solutions of regularly perturbed systems of differential equations with a non classical right-hand side*, Ukrainian Mathematical Journal, 43, no.10, pp. 1209 - 1214, (1991).

[2] Akhmetov, M., and Perestyuk, N., *Periodic solutions of strongly nonlinear systems with nonclassical right-hand side in the case of a family of generating solutions*, Ukrainian Mathematical Journal, 45, no.2, 215 - 222, (1993).

[3] Alberts, Bruce, Bray, Dennis, Lewis, Julian, Raff, Martin, Roberts, Kieth, Watson, James D., *Molecular Biology of the Cell*, Third ed. Garland Publishing, Inc. New York & London, (1994).

[4] Aligue, R., Wu, L., and Russell, P. J., Biol. Chem. 272, 13320 (1997).

[5] Alligood, Kathleen T., Sauer, Tim D., and Yorke, James A., Chaos: An Introduction to Dynamical Systems, Springer-Verlag New York, Inc., (1996).

[6] Alur, R., Henzinger, T.A., Modularity for timed and hybrid systems, in: A. Mazurkiewicz, J. Winkowski (Eds.), CONCUR 97: Concurrency Theory, Lecture Notes in Computer Science, vol. 1243, Springer, Berlin, pp. 74 - 88, (1997).

[7] Arfken, G., *Mapping 6.6 in Mathematical Methods for Physicists*, 3rd ed. Orlando, FL:Academic Press, pp. 384 - 392, (1985).

[8] Benito, J., Martin-Castellanos, C., and Moreno, S., EMBO J. 17, 482 (1998).

[9] Branicky, M., Borkar, V., and Mitter S., A unified framework for hybrid control: Model and optimal control. IEEE Transactions on Automatic Control, 43(1):31 - 45, (Jan. 1998).

[10] Cerutti, L., Simanis, V., Controlling the end of the cell cycle. Curr Opin Genet Dev, 10:65 - 69, (2000).

[11] Clark, Paul ATM Motorway Traffic Simulator, B.Sc. Computer Science, (23/03/2006).

[12] Correabordes, J., and Nurse, P., Cell 83, 1001 (1995); C. Martin-Castellanos, K. Labib, and S. Moreno, EMBO J. 15, 839 (1996).

[13] Creighton, Thomas E., Encyclopedia of Molecular Biology, European' Molecular Biology Laboratory London, England Volumes 1 - 4, (1999).

[14] Croft, H.T., Falconer,K.J., and Guy,R.K., Convexity, Ch. A in Unsolved Problems in Geometry. New York:Springer-Verlag, pp.6 - 47, (1994).

[15] Diestel Reinhard, Graph Theory, pp. 2- , New York, (2000).

[16] Ding-Zhu Du, Ker-I Ko, Problem Soluing in Automata, Languages, and Complexit, (2001).

[17] Edgar, B.A., O'Farrell, P.H., The three postblastoderm cell cycles of Drosophila embryogenesis are regulated in G2 by string. Cell 62: 469 - 480, 1990 O'Farrell, P.H. Cell cycle control: many ways to skin a cat. Trends Cell Biol. 2:159 - 163, (1992).

[18] Edwards, R., Glass, L., *Combinatorial explosion in model gene networks*, Journal of Chaos volume 10 number 3 pp. 691 - 704, (September 2000).

[19] Edwards, R., Siegelmann, H.T., Aziza, K., Glass, L., *Symbolic dynamics and computation in model gene networks*, Journal of Chaos volume 11 number 1 pp. 160 - 169, (March 2001).

[20] Eriksson, Olivia, Zhou, Yishao, and Tegne, Jesper, Modeling Complex Cellular Networks - robust switching in the cell cycle ensures a piecewise linear reduction of the regulatory network, 43rd IEEE Conference on Decision and Control, Atlantis, Paradise Island, Bahamas, (December 14 - 17, 2004).

[21] Fankhauser, G., Nucleo-cytoplasmic relations in amphibian development. Int. Rev. Cytol. 1:165 - 193, 1952 Henery C.C., Bard, J.B.L., Kaufman, M.H. Tetraploidy in mice, embryonic cell number, and the gain of the developmental map. Dev. Biol. 152:233 - 241, 1992 Prescott, D.M. Changes in nuclear volume and growth rate and prevention of cell division in Amoeba

proteus resulting from cytoplasmic amputations. Exp. Cell Res. 11:94 - 98, (1956).

[22] Färnqvist, Daniel, Modelica Modeling of Computer Networks, Master's Thesis, Royal Institute of Technology (KTH) Department of Signals, Sensors & Systems, (May 2002).

[23] Fesquet, D., et al., The MO15 gene encodes the catalytic subunit of a protein kinase that activates cdc2 and other cyclin dependent kinases (CDKs) through phosphorylation of Thr 161 and its homologues. EMBO J. 12:3111 - 3121, (1993) Gould, K.L.: Nurse, P. Tyrosine phosphorylation of the fission yeast cdc2$^+$ protein kinase regulates entry into mitosis. Nature 342:39 - 45, (1989) Kumagai, A.; Dunphy, W.G. Regulation of the cdc25 protein during the cell cycle in Xenopus extracts. Cell 70:139 - 151, (1992).

[24] Filippov, A.F., *Differential Equations with Discontinuous Righthand Sides*, (1988).

[25] Fisher, D., and Nurse, P. Semin., Cell Biol. 6,73 (1995); K. Nasmyth, Trends Genet. 12, 405 (1996).

[26] Floyd, S. and Paxson, V., Difficulties in Simulating the Internet. IEEE/ACM Trans. On Networking, 9(4):392 - 403, (2001).

[27] Gebert, J., Öktem H., Pickl, S.W., Radde, N., Weber, G.W., Yılmaz, F. B., Formulation An Algorithmic Approach to Local State Transition Matrices,

to appear in the proceedings of InterSymp' 2004 The International Institute for Advanced Studies in Systems Research and Cybernetics, Baden, (July 29th to August 5nd 2004).

[28] Gradshteyn, I.S., and Ryzhik, I.M., *Tables of Integrals, Series, and Products*, 5th ed. San Diego, CA: Academic Press, pp. 1114 - 1125, (1979).

[29] Gray, A., *Euclidean Spaces, 1.1 in Modern Differential Geometry of Curves and Surfaces*, Boca Raton, FL:CRC Press, pp. 2-4 (1993).

[30] Hasselblatt, B., and Katok, A., *A First Course in Dynamics with a Panorama of Recent Developments*, (2003).

[31] Heath Michael T., *Scientific Computing An Introductory Survey*, 2nd ed., (2001).

[32] Henzinger, T.A., *The theory of hybrid automata*, Proc. 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996).

[33] Jacobson, V., Congestion avoidance and control. In Proc. of SIGCOMM, volume 18.4, pages 314 - 329, (1988).

[34] Johanssona, Karl Henrik, Egerstedtb, Magnus, Lygerosa, J., Sastrya, S., On the regularization of Zeno hybrid automata, Systems & Control Letters 38 (1999) 141 - 150, (1999).

[35] Johnston, G.C., Pringle, J.R., Hartwell, L.H., Coordination of growth with cell division in the yeast Saccharomyces cerevisiae. Exp. Cell Res. 105:79-98, 1977 Nurse, P. Genetic control of cell size at cell division in yeast. Nature 256:547 - 551, (1975).

[36] Lundgren, K., et. al., Mik1 and Wee1 Cooperate in the Inhibitory Tyrosine Phosphorylation of Cdc2, Cell 64, 1111 (1991).

[37] Lee, X., Transformation of the Plane, http://www.best.com/˜xah/Math Graphics Galery dir/Transform 2d Plot dir/transform2dPlot.html.

[38] Lodish, Berk, Matsudaira, Kaiser, Krieger, Scot, Zipursky, Darnell, *Molecular Cell Biology*, Fifth ed., (2004).

[39] Luonan Chen, Senior Member, IEEE, and Kazuyuki Aihara, Stability of Genetic Regulatory Networks With Time Delay, IEEE Transections on Circuits and Systems—I: Fundemental Theory and Applications, VOL. 49, NO. 5, (May. 2002).

[40] Lygeros, J., Tomlin, C., Sastry, S., Controllers for reachability speci cations for hybrid systems, Automatica 35 (3) (1999) 349 - 370, (1999).

[41] Lygeros, J., Johansson, K., Simi´c, S., Zhang, J., and Sastry, S., Dynamical properties of hybrid automata, IEEE Transactions on Automatic Control, 48(1):2 - 16, (Jan. 2003).

[42] Lygeros John, Lecture Notes on Hybrid Systems, University of Patros, (2004).

[43] Martin-Castellanos, C., Blanco, M. A., de Prada, J. M., and Moreno, S., Mol. Biol. Cell 11, 543 (2000).

[44] Megiddo, M., Hakimi, S., Garey, M., Johnson, S., and Papadimitriou, C., "The complexity of searching a graph," J. ACM, vol. 35, no. 2, pp. 18 - 44, (1988).

[45] Millar, J., and Russell, P., Cell 68, 407 (1992).

[46] Moreno, S., and Nurse, P., Nature (London) 367, 236 (1994).

[47] Murray, A., and Hunt, T., The Cell Cycle, W. H. Freeman, New York, (1993).

[48] Nasmyth, K., Control of the yeast cell cycle by the Cdc28 protein kinase. Curr. Opin. Cell Biol. 5:166 - 179,1993 Solomon, M.J. Activation of the various cyclin/cdc2 protein kinase. Curr. Opin. Cell Biol. 5:180 - 186, (1993).

[49] Novak, B., Csikasz-Nagy, A., Gyorffy, B., K. Nasmyth, and Tyson, J. J., Philos. Trans. R. Soc. London, Ser. B 353, 2063 (1998).

[50] Novak, B., Pataki, Z., Ciliberto, A., and Tyson John J., Mathematical Model of the Cell Division Cycle of Fission Yeast, CHAOS Volume 11, Number 1, (March 2001).

[51] Nurse, P., Nature (London) 344,503 (1990).

[52] ÖKTEM, Hakan Hybrid Systems Lecture Notes , Ankara, (2006).

[53] Parsons, T. "Pursuit-evasion in a graph," in Lecture Notes in Mathmatics, Theory and Application of Graphs, Y. Alani and D. Lick, Eds: Springer-Verlag, vol. 642, pp. 426 - 441., (1976).

[54] Paxson, V., and Floyd, S., Wide-area traffic: the failure of Poisson modeling. IEEE/ACM Trans. on Networking, 3(3): 226 - 244, (1995).

[55] Peterson, L. L., and Davie, B. S., Computer networks: a systems approach. Morgan Kauffmann, 2nd edition, (2000).

[56] Piovesan, Jorge L.; Tanner, Herbert G.; and Abdallah, Chaouki T. "Discrete Asymptotic Abstractions of Hybrid Systems", Proceedings of the 45th IEEE Conference on Decision & Control Manchester Grand Hyatt Hotel San Diego, CA, USA, December 13 - 15, (2006).

[57] Pritchett, Amy R., Lee, Seungman, Huang, David, Goldsman, David Hybrid - System Simulation for National Airspace System Safety Analysis, Proceedings of the 2000 Winter Simulation Conference, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds., (2000).

[58] Rosen Kenneth H., *Discrete Mathematics and Its Applications*, pp. 438 - 439, 1999.

[59] Roussel, Marc R., Delay-differential equations, (July 22 2004).

[60] Russell, P, Nurse, P., cdc25$^+$ functions as an inducer in the mitotic control of fission yeast, Cell 1986;45:145 - 153, (1986).

[61] Russell, P., and Nurse, P., Cell 49, 559 (1987).

[62] Russell, P, Nurse, P., Negative regulation of mitosis by wee1$^+$, a gene encoding a protein kinase homolog. Cell, 49:559 - 567, (1987).

[63] Shampine, L.F., and Thompson, S., Solving DDEs in MATLAB, Applied Numerical Mathematics, Vol. 37, pp. 441 - 458, (2001).

[64] Skiena, S., *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Reading, MA: Addison-Wesley, (1990).

[65] Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., and Sastry, S. Shankar Distributed Control Applications Within Sensor Networks, Proceedings of the IEEE, Vol. 91, No. 8, (August 2003).

[66] Stern, B, Nurse, P., A quantitative model for the cdc2 control of S phase and mitosis in fission yeast. Trends Genet, 12:345 - 350, (1996).

[67] Strang, G., *Linear Algebra and Its Applications*, 4th ed., New York, Academic Press, (1980).

[68] Tyson, JJ, Chen, KC, Novak B., Network dynamics and cell physiology, Nature Rev Mol Cell Biol, 2:908 - 916, (2002).

[69] Tyson, John J., Csikasz-Nagy, Attila, and Novak, B., The Dynamics of Cell Cycle Regulation, BioEssays 24.12, (2002).

[70] Waldeer, K. T., Kinetic Theory in Vehicular Traffic Flow Modeling, 25th International Symposium on Rarefied Gas Dynamics, Saint-Petersburg, Russia, (2006).

[71] Walker, Graeme M., Yeast Physiology and Biotechnology, John Wiley & Sons Ltd., England, (2000).

[72] Weisstein, Eric W., *CRC Concise Encyclopedia of Mathematics*, (1999).

[73] Wilf Herbert S., *Algorithms and Complexity*, pp.24 - , Philadelphia, (1994).

[74] Yamaguchi, S., Murakami, H., and Okayama, H., Mol. Biol. Cell 8, 2475 (1997); K. Kitamura, H. Maekawa, and C. Shimoda, ibid. 9, 1065 (1998).

[75] Yamaguchi, S., Okayama, H., and Nurse, P., EMBO J. 19,3968 (2000); M. Blanco, A. Sanchez-Diaz, J. M. Prada, and S. Moreno, ibid. 19, 3945 (2000).

[76] Zachariae, W., Nasmyth, K., Whose end is destruction: cell division and the anaphase-promoting complex. Genes Dev, 13:2039 - 2058, (1999).

[77] MatLab 7.0.0.19920 (R14) Help "MATLAB Function Reference".

[78] http://wwwer.df.op.dlr.de/cacsd/hds/index.shtml - IEEE CACSD homepage.

[79] http://www.isi.edu/nsman/ns. The Network Simulator ns-2. Information Sciences Institute, University of Southern California.

Appendix A. MATLAB FUNCTIONS

A.1 dde23

dde23 Solve delay differential equations (DDEs) with constant delays [77, 63].

Syntax:

sol = dde23(ddefun,lags,history,tspan)

sol = dde23(ddefun,lags,history,tspan,options)

Arguments:

ddefun: Function that evaluates the right side of the differential equations $y'(t) = f(t, y(t), y(t-\tau_1), ..., y(t-\tau_k))$. The function must have the form

$dydt = ddefun(t, y, Z)$

where t corresponds to the current t, y is a column vector that approximates $y(t)$, and Z(:,j) approximates $y(t-\tau_j)$ for delay $\tau_j = lags(j)$. The output is a column vector corresponding to $f(t, y(t), y(t-\tau_1), ..., y(t-\tau_k))$.

lags: Vector of constant, positive delays $\tau_1, ..., \tau_k$.

history: Specify history in one of three ways:

A function of t such that $y = history(t)$ returns the solution $y(t)$ for $t \leq t_0$ as a column vector

A constant column vector, if $y(t)$ is constant

The solution *sol* from a previous integration, if this call continues that integration

tspan: Interval of integration as a vector $[t_0, t_f]$ with $t_0 < t_f$.

options: Optional integration argument. A structure you create using the ddeset function.

p1,p2,... Optional parameters that dde23 passes to ddefun, if it is a function, and any functions you specify in options.

Description

$sol = dde23(ddefun, lags, history, tspan)$ integrates the system of DDEs

$y'(t) = f(t, y(t), y(t - \tau_1), ..., y(t - \tau_k))$

on the interval $[t_0, t_f]$, where $\tau_1, ..., \tau_k$ are constant, positive delays and $t_0 < t_f$.

dde23 returns the solution as a structure sol. Use the auxiliary function "deval" and the output sol to evaluate the solution at specific points tint in the interval $tspan = [t_0, t_f]$. $yint = deval(sol, tint)$

The structure sol returned by dde23 has the following fields.

$sol.x$	Mesh selected by dde23
$sol.y$	Approximation to $y(x)$ at the mesh points in $sol.x$.
$sol.yp$	Approximation to $y'(x)$ at the mesh points in $sol.x$
$sol.solver$	Solver name, 'dde23'

sol = dde23(ddefun,lags,history,tspan,options) solves as above with default integration properties replaced by values in options, an argument created with ddeset.

Commonly used options are scalar relative error tolerance 'RelTol' (1e-3 by default) and vector of absolute error tolerances 'AbsTol' (all components are 1e-6 by default). Use the 'Jumps' option to solve problems with discontinuities in the history or solution. Set this option to a vector that contains the locations of discontinuities in the solution prior to t0 (the history) or in coefficients of the equations at known values of after t0. Use the 'Events' option to specify a function that dde23 calls to find where functions $g(t, y(t), y(t-\tau_1), ..., y(t-\tau_k))$ vanish. This function must be of the form

[value,isterminal,direction] = events(t,y,Z)

and contain an event function for each event to be tested. For the kth event function in events:

value(k) is the value of the kth event function.

isterminal(k) = 1 if you want the integration to terminate at a zero of this event function and 0 otherwise.

direction(k) = 0 if you want dde23 to compute all zeros of this event function, +1 if only zeros where the event function increases, and -1 if only zeros where the event function decreases.

If you specify the 'Events' option and events are detected, the output structure sol also includes fields:

sol.xe Row vector of locations of all events, i.e., times when an event function vanished

sol.ye Matrix whose columns are the solution values corresponding to times in sol.xe

sol.ie Vector containing indices that specify which event occurred at the corresponding time in sol.xe

Examples

This example solves a DDE on the interval $[0, 5]$ with lags 1 and 0.2. The function ddex1de computes the delay differential equations, and ddex1hist computes the history for t <= 0.

sol = dde23(@ddex1de,[1, 0.2],@ddex1hist,[0, 5]);

This code evaluates the solution at 100 equally spaced points in the interval [0,5], then plots the result.

tint = linspace(0,5);

yint = deval(sol,tint);

plot(tint,yint);

Algorithm

dde23 tracks discontinuities and integrates with the explicit Runge-Kutta (2,3) pair and interpolant of ode23. It uses iteration to take steps longer than the lags.

Appendix B. M FILES

B.1 MatLab Code of Piecewise Linear System Example

```
%**********************************
% (C) Mustafa KAHRAMAN (2007) *
%**********************************
function Simulate_Cycle_Without_Delay
%System parameters
M=[-1 0;0 -1];
k=[0 2 0 2;0 0 2 2];
treshold = 1;
%Initial value 1
%init=[1.002;0.9999];
%Initial value 2
init=[1.1;0.2];
%States
state_1 = false;
state_2 = false;
state_3 = false;
```

```
state_4 = false;

state = 1;

%Time Approximation

step = 0.001;

time = 0;

t_n = 20;

%Data

y = init;

values = transpose(init);

for i = 0:step:t_n

time = time + step;

% state 3 ********************

if(y(1)<1 & y(2)<1)

alues=[values;transpose(y)];

% state 2 ******************

elseif (y(1)>1 & y(2)>1)

diff_eqn = @(y)M*y+k(:,3);

y = Euler_Method(y,diff_eqn,step);
```

```
values=[values;transpose(y)];

% state 4 ********************

elseif (y(1)<=1 & y(2)>=1)

diff_eqn = @(y)M*y+k(:,1);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

% state 1 **********************

elseif (y(1)>=1 & y(2)<=1)

diff_eqn = @(y)M*y+k(:,4);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

end

end

hold on;

hold all;

%Plot y1 - y2

plot(values(:,2),values(:,1));

title('Phase plane (y1-y2)');
```

```
xlabel(’y2’);

ylabel(’y1’);

%The lines for thresholds.

plot([1 1],[0 2],’r’,’LineWidth’,2);

plot([0 2],[1 1],’r’,’LineWidth’,2);


%Evaluates the y’n+1’ vlue according to previous point informaton

%using Euler’s method

%Inputs:

% y_n –> previous y value

% fy –> differential equation (function input)

% h –> step size

%

%Output:

% y_n_1 –> next y value

function y_n_1 = Euler_Method(y_n,fy,h)

f_tn_yn =fy(y_n);

y_n_1 = y_n + f_tn_yn*h;

end
```

B.2 MatLab Code of Stability Analysis

```
%***********************************
% (C) Mustafa KAHRAMAN (2007) *
%***********************************
function Simulate_cycle_with_delay
M=[-1 0;0 -1];
k=[0 2 0 2;0 0 2 2];
threshold = 1;
init=[1;0.1];
const_delay = 0.5;
%State Transition Time
t12=0;
t24=0;
t43=0;
t31=0;
%Boolean variables for states
state_1 = false;
state_2 = false;
```

```
state_3 = false;

state_4 = false;

state = 1;

%Time Approximation

step = 0.001;

time = 0;

t_n = 10;

%Data

y = init;

values = transpose(init);

for i = 0:step:t_n

time = time + step;

% state 3 ********************

if(y(1)<1 & y(2)<1)

if(state_3 == false)

state_1 = false;

state_2 = false;

state_3 = true;
```

```
state_4 = false;

t43=time-step;

diff_eqn = @(y)M*y+k(:,1);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

else

if time <= t43 + const_delay

diff_eqn = @(y)M*y+k(:,1);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

else

diff_eqn = @(y)M*y+k(:,2);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

end

end

% state 2 ******************

elseif (y(1) >1 & y(2)>1)
```

```
if(state_2 == false)

state_1 = false;

state_2 = true;

state_3 = false;

state_4 = false;

t12=time-step;

diff_eqn = @(y)M*y+k(:,4);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

else

if time <= t12 + const_delay

diff_eqn = @(y)M*y+k(:,4);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

else

diff_eqn = @(y)M*y+k(:,3);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];
```

```
end

end

% state 4 *******************

elseif (y(1)<=1 & y(2)>=1)

if(state_4 == false)

state_1 = false;

state_2 = false;

state_3 = false;

state_4 = true;

t24=time-step;

diff_eqn = @(y)M*y+k(:,3);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

else

if time <= t24 + const_delay

diff_eqn = @(y)M*y+k(:,3);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];
```

```
else

diff_eqn = @(y)M*y+k(:,1);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

end

end

% state 1 *********************

elseif (y(1)>=1 & y(2)<=1)

if(state_1 == false)

state_1 = true;

state_2 = false;

state_3 = false;

state_4 = false;

t31=time-step;

diff_eqn = @(y)M*y+k(:,2);

y = Euler_Method(y,diff_eqn,step);

values=[values;transpose(y)];

else
```

```
if time <= t31 + const_delay
diff_eqn = @(y)M*y+k(:,2);
y = Euler_Method(y,diff_eqn,step);
values=[values;transpose(y)];
else
diff_eqn = @(y)M*y+k(:,4);
y = Euler_Method(y,diff_eqn,step);
values=[values;transpose(y)];
end
end
end
end
%Plot y1 - y2
plot(values(:,2));
title('y2-t');
xlabel('y2');
ylabel('y1');
%Plot y1 - y2
```

```
plot(values(:,2),values(:,1));

title('Phase plane (y1-y2)');

xlabel('y2');

ylabel('y1');

%Threshold lines

plot([0 2],[1 1],'r');

plot([1 1],[0 2],'r');


%Evaluates the y'n+1' values according to previous point

%using Euler's method

%Inputs:

% y_n -> previous y value

% fy -> differential equation (function input)

% h -> step size

%

%Output:

% y_n_1 -> next y value

function y_n_1 = Euler_Method(y_n,fy,h)

f_tn_yn =fy(y_n);
```

```
y_n_1 = y_n + f_tn_yn*h;

end
```

B.3 MatLab Code of HDS and ODE Models of Cell Division Cycle of Fission Yeast

Matlab Code Of ODE Model

```
%****************************************
%Simulation of Fission Yeast Cell Division
% (C) Mustafa KAHRAMAN
%****************************************
function result = simulation_fission_yeast_original
global Cdc13T;
global Ste9;
global Slp1;
global preMPF;
global kwee;
global k25;
global MPF;
global SK;
global IEP;
global Slp1T;
global Rum1T;

global TF;

global trimer;

global M;

global mu;

%Initial values of global variables

Cdc13T=0.1;

preMPF=0;

Ste9=0;

Slp1=0.01;

%Slp1=2.2;

Slp1T=1;

IEP=0.1;

Rum1T=0.1;

SK=0.1;

M=1;

mu=0.005;

MPF=0.0095;
```

```
kwee=1.2996;

k25=0.519;

TF=0.0098;

trimer=0.0905;

%Initial values of iteration variables

Cdc13T_i=Cdc13T;

preMPF_i=preMPF;

%Slp1=0;

Slp1_i=Slp1;

Slp1T_i=Slp1T;

IEP_i=IEP;

Rum1T_i=Rum1T;

SK_i=SK;

M_i=M;

Ste9_i=Ste9;

kwee_i=kwee;

k25_i=k25;

MPF_i=MPF;
```

```
TF_i=TF;

trimer_i=trimer;

%time

t_initial=0;

t=t_initial;

t_step=0.005;

t_final=150;

%Data arrays

Cdc13T_array=[0 Cdc13T];

preMPF_array=[0 preMPF];

Ste9_array=[0 Ste9];

Slp1_array=[0 Slp1];

Slp1T_array=[0 Slp1T];

IEP_array=[0 IEP];

Rum1T_array=[0 Rum1T];

SK_array=[0 SK];

MPF_array=[0 MPF];

t_array=t_initial;
```

```
M_array=[0 M];

kwee_array=[0 kwee];

k25_array=[0 k25];

for t=t_initial:t_step:t_final

if(M<=2)

%Cdc13T**********************************************

a = ode23t(@dCdc13T_dt,[t t+t_step],Cdc13T);

Cdc13T_i=a.y(11); Cdc13T_array=[Cdc13T_array;[t Cdc13T_i]];

%preMPF**********************************************

a = ode23t(@dpreMPF_dt,[t t+t_step],preMPF);

preMPF_i=a.y(11); preMPF_array=[preMPF_array;[t preMPF_i]];

%Ste9***********************************************

a = ode23t(@dSte9_dt,[t t+t_step],Ste9);

Ste9_i=a.y(11); Ste9_array=[Ste9_array;[t Ste9]];

%Slp1T**********************************************

a = ode23t(@dSlp1T_dt,[t t+t_step],Slp1T);

Slp1T_i=a.y(11); Slp1T_array=[Slp1T_array;[t Slp1T_i]];

%Slp1**********************************************
```

```
a = ode23t(@dSlp1_dt,[t t+t_step],Slp1);

Slp1_i=a.y(11); Slp1_array=[Slp1_array;[t Slp1_i]];

%IEP**************************************************

a = ode23t(@dIEP_dt,[t t+t_step],IEP);

IEP_i=a.y(11); IEP_array=[IEP_array;[t IEP_i]];

%Rum1T************************************************

a = ode23t(@dRum1T_dt,[t t+t_step],Rum1T);

Rum1T_i=a.y(11); Rum1T_array=[Rum1T_array;[t Rum1T_i]];

%SK***************************************************

a = ode23t(@dSK_dt,[t t+t_step],SK);

SK_i=a.y(11); SK_array=[SK_array;[t SK_i]];

%M****************************************************

M_i=Euler_Method(M,@dM_dt,t_step);

M_array=[M_array;[t M_i]];

%trimer**********************************************

trimer_i=Trimer_equation(Cdc13T,Rum1T);

%MPF**************************************************

MPF_i=MPF_equation(Cdc13T,preMPF,trimer);
```

```
MPF_array=[MPF_array;[t MPF_i]];

TF_i=G(1.5*M,1+2*MPF,0.01,0.01);

%kwee************************************************

kwee_i=Kwee_equation(MPF);

kwee_array=[kwee_array;[t kwee_i]];

%k25*************************************************

k25_i=K25_equation(MPF);

k25_array=[k25_array;[t k25_i]];

%Change global variables

Cdc13T=Cdc13T_i;

preMPF=preMPF_i;

Slp1=Slp1_i;

Slp1T=Slp1T_i;

IEP=IEP_i;

Rum1T=Rum1T_i;

SK=SK_i;

M=M_i;

Ste9=Ste9_i;
```

```
kwee=kwee_i;

k25=k25_i;

MPF=MPF_i;

TF=TF_i;

trimer=trimer_i;

end

end

figure;

hold all

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Slp1_array(:,1),Slp1_array(:,2),'LineWidth',2);

plot(MPF_array(:,1),MPF_array(:,2),'LineWidth',3);

figure;

hold all

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Ste9_array(:,1),Ste9_array(:,2),'LineWidth',2);

plot(kwee_array(:,1),kwee_array(:,2),'LineWidth',3);

figure;
```

```
hold all

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(k25_array(:,1),k25_array(:,2),'LineWidth',2);

plot(SK_array(:,1),SK_array(:,2),'LineWidth',3);

figure;

hold all

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Cdc13T_array(:,1),Cdc13T_array(:,2),'LineWidth',2);

plot(preMPF_array(:,1),preMPF_array(:,2),'LineWidth',3);

figure;

hold all

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Rum1T_array(:,1),Rum1T_array(:,2),'LineWidth',2);

end

%Evaluates the y'n+1' vlue according to previous point information

%using Euler's method

%Inputs:

% y_n -> previous y value
% fy –> differential equation (function input)

% h –> step size

%

%Output:

% y_n_1 –> next y value

function y_n_1 = Euler_Method(y_n,fy,h)

f_tn_yn =fy(y_n);

y_n_1 = y_n + f_tn_yn*h;

end

%Differential equation of Cdc13T

function result = dCdc13T_dt(param_Cdc13T)

global M;

global Ste9;

global Slp1;

k1=0.03;

kb2=0.03;

kbb2=1;

kbbb2=0.1;
```

```
result=k1*M-(kb2+kbb2*Ste9+kbbb2*Slp1)*param_Cdc13T;

end

%Differential equation of preMPF

function result = dpreMPF_dt(param_preMPF)

global kwee;

global Cdc13T;

global Slp1;

global Ste9;

global k25;

kb2=0.03;

kbb2=1;

kbbb2=0.1;

result=kwee*(Cdc13T-param_preMPF)-k25*param_preMPF-

          (kb2+kbb2*Ste9+kbbb2*Slp1)*param_preMPF;

end

%Differential equation of Ste9

function result = dSte9_dt(param_Ste9)

global Slp1;
global MPF;

global SK;

kb3=1;

kbb3=10;

J3=0.01;

kb4=2;

k4=35;

J4=0.01;

result = (kb3+kbb3*Slp1)*((1-param_Ste9)/(J3+1-param_Ste9))-

          (kb4*SK+k4*MPF)*(param_Ste9/(J4+param_Ste9));

end

%Differential equation of Slp1

function result = dSlp1_dt(param_Slp1)

global IEP;

global Slp1T;

k7=1;

J7=0.001;

k8=0.25;
```

```
k6=0.1;

J8=0.001;

result=k7*IEP*((Slp1T-param_Slp1)/(J7+Slp1T-param_Slp1))-

            k8*(param_Slp1/(J8+param_Slp1))-k6*param_Slp1;

end

%Differential equation of Slp1T

function result = dSlp1T_dt(param_Slp1T)

global MPF;

kb5=0.005;

kbb5=0.3;

J5=0.3;

k6=0.1;

result = kb5+kbb5*((MPF^4)/(J5^4+MPF^4))-k6*param_Slp1T;

end

%Differential equation of IEP

function result = dIEP_dt(param_IEP)

global MPF;

k9=0.1;
```

```
J9=0.01;

k10=0.04;

J10=0.01;

result=k9*MPF*((1-param_IEP)/(J9+1-param_IEP))-
k10*(param_IEP/(J10+param _IEP));

end

%Differential equation of Rum1T

function result=dRum1T_dt(param_Rum1T)

global MPF;

global SK;

k11=0.1;

k12=0.01;

kb12=1;

kbb12=3;

result=k11-(k12+kb12*SK+kbb12*MPF)*param_Rum1T;

end

%Differential equation of SK

function result=dSK_dt(param_SK)

global TF;
```

```
k13=0.1;

k14=0.1;

result=k13*TF-k14*param_SK;

end

%Differential equation of M

function result=dM_dt(param_M)

global mu;

result=mu*param_M;

end

%Algebraic equation of kwee

function result=Kwee_equation(param_MPF)

kbwee=0.15;

kbbwee=1.3;

Vawee=0.25;

Viwee=1;

Jawee=0.01;

Jiwee=0.01;

result=kbwee+(kbbwee-kbwee)*G(Vawee,Viwee*param_MPF,Jawee,Jiwee);
```

```
end

%Algebraic equation of k25

function result=K25_equation(param_MPF)

kb25=0.05;

kbb25=5;

Va25=1;

Vi25=0.25;

Ja25=0.01;

Ji25=0.01;

result=kb25+(kbb25-kb25)*G(Va25*param_MPF,Vi25,Ja25,Ji25);

end

%Algebraic equation of TF

function result=TF_equation(param_M,param_MPF)

k15=1.5;

kb16=1;

kbb16=2;

J15=0.01;

J16=0.01;
```

```
result=G(k15*param_M,kb16+kbb16*param_MPF,J15,J16);

end

%Algebraic equation of Trimer

function result = Trimer_equation(param_Cdc13T,param_Rum1T)

Kdiss=0.001;

result=(2*param_Cdc13T*param_Rum1T)/
(Sigma(param_Cdc13T,param_Rum1T)+
sqrt(Sigma(param_Cdc13T,param_Rum1T)^2-
4*param_Cdc13T*param_Rum1T));

end

%Algebraic equation of MPF

function result=MPF_equation(param_Cdc13T,param_preMPF,param_trimer)

result=((param_Cdc13T-param_preMPF)*(param_Cdc13T-param_trimer))/
param _Cdc13T;

end

%Sigma

function result=Sigma(param_Cdc13T,param_Rum1T)

Kdiss=0.001;

result=param_Cdc13T+param_Rum1T+Kdiss;

end
```

```
%G

function result = G(a,b,c,d)

result=(2*a*d)/(b-a+b*c+a*d+sqrt((b-a+b*c+a*d)^2-4*a*d*(b-a)));

end
```

MatLab Code of HDS Model

```
%*****************************

% (C) Mustafa KAHRAMAN (2007)

%*****************************

function Simulate_Fission_Yeast_HDS_ALL_new3

%M

M=1;

EC=2;%Environmental Conditions

global M_array;

M_array=[0 M];

global CS_G1;

CS_G1=1.1;%Critical Size for G2 size checkpoint

global CS_G2;

CS_G2=1.9;%Critical Size for G2 size checkpoint
```

```
%Cdc25

Cdc25=0.01;

global Cdc25_array;

Cdc25_array=[0 Cdc25];

%SK

SK=0.45;

global SK_array;

SK_array=[0 SK];

%Rum1T

Rum1T=0.01;

global Rum1T_array;

Rum1T_array=[0 Rum1T];

%Slp1

Slp1=0.01;

global Slp1_array;

Slp1_array=[0 Slp1];

%Wee1

Wee1=0;
```

```
global CS_G1;

global Wee1_array;

Wee1_array=[0 Wee1];

%Ste9

Ste9=0;

global Ste9_array;

Ste9_array=[0 Ste9];

%Cdc13T

Cdc13T=0.01;

global Cdc13T_array;

Cdc13T_array=[0 Cdc13T];

%preMPF

preMPF=0.01;

global preMPF_array;

preMPF_array=[0 preMPF];

%MPF

MPF=0.01;

global MPF_array;
```

```
MPF_array=[0 MPF];

%time

t_step=0.005;

t_initial=0;

t_final=200;

global t;

for t=t_initial:t_step:t_final

M = M_module(M,EC,t_step);

M_array=[M_array;[t M]];

SK = SK_module(SK,M,MPF,t_step);

SK_array=[SK_array;[t SK]];

Rum1T = Rum1T_module(Rum1T,SK,MPF,t_step);

Rum1T_array=[Rum1T_array;[t Rum1T]];

Ste9 = Ste9_module(Ste9,MPF,Slp1,SK,t_step);

Ste9_array=[Ste9_array;[t Ste9]];

Wee1 = Wee1_module(Wee1,MPF,t_step);

Wee1_array=[Wee1_array;[t Wee1]];

Cdc25 = Cdc25_module(Cdc25,MPF,t_step);
```

```
Cdc25_array=[Cdc25_array;[t Cdc25]];

Cdc13T = Cdc13T_module(Cdc13T,M,Ste9,Slp1,t_step);

Cdc13T_array=[Cdc13T_array;[t Cdc13T]];

preMPF = preMPF_module(preMPF,Cdc13T,Wee1,Cdc25,Ste9,Slp1,t_step);

preMPF_array=[preMPF_array;[t preMPF]];

Slp1 = Slp1_module(Slp1,MPF,SK,t_step);

Slp1_array=[Slp1_array;[t Slp1]];

MPF_array=[MPF_array;[t MPF]];

MPF = MPF_module(MPF,Cdc13T,preMPF,Rum1T,t_step);

end

figure;

hold on;

hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Cdc25_array(:,1),Cdc25_array(:,2),'LineWidth',2);

plot(MPF_array(:,1),MPF_array(:,2),'LineWidth',3);

figure;

hold on;
hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Slp1_array(:,1),Slp1_array(:,2),'LineWidth',2);

plot(MPF_array(:,1),MPF_array(:,2),'LineWidth',3);

figure;

hold on;

hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Wee1_array(:,1),Wee1_array(:,2),'LineWidth',2);

plot(MPF_array(:,1),MPF_array(:,2),'LineWidth',3);

figure;

hold on;

hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(SK_array(:,1),SK_array(:,2),'LineWidth',2);

plot(MPF_array(:,1),MPF_array(:,2),'LineWidth',3);

figure;

hold on;
hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Ste9_array(:,1),Ste9_array(:,2),'LineWidth',2);

figure;

hold on;

hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Rum1T_array(:,1),Rum1T_array(:,2),'LineWidth',2);

figure;

hold on;

hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(Cdc13T_array(:,1),Cdc13T_array(:,2),'LineWidth',2);

figure;

hold on;

hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(preMPF_array(:,1),preMPF_array(:,2),'LineWidth',2);
```

```
figure;

hold on;

hold all;

plot(M_array(:,1),M_array(:,2),'LineWidth',1);

plot(MPF_array(:,1),MPF_array(:,2),'LineWidth',2);

end

% Cell Mass Module*******

function result_M = M_module(M,EC,t_step)

if M>=2

result_M=1;

elseif M<2 & EC>=1

f=@(x)0.005*x;

result_M=Euler_Method(M,f,t_step);

elseif M<2 & EC$¡1

f=@(x)0.0025*x;

result_M=Euler_Method(M,f,t_step);

end

end
```

```
% SK Module**********

function result_SK = SK_module(SK,M,MPF,t_step)

global CS_G1;

if M>=CS_G1 & MPF<1

f=@(x)(-1)*x+1.1;

result_SK=Euler_Method(SK,f,t_step);

elseif M>=CS_G1 & MPF>=1

f=@(x)(-1)*x+0.4;

result_SK=Euler_Method(SK,f,t_step);

elseif M<CS_G1 & MPF<1

f=@(x)(0-1)*x+0.4;

result_SK=Euler_Method(SK,f,t_step);

elseif M<CS_G1 & MPF>=1

SK=0.01;

f=@(x)(0-1)*x+0.4;

result_SK=Euler_Method(SK,f,t_step);

end

end
```

```
% MPF Module**********

function result_MPF = MPF_module(MPF,Cdc13T,preMPF,Rum1T,t_step)

if Cdc13T>=0.7 & preMPF>=0.5 & Rum1T>=0.15

f=@(x)(-1)*x+0.01;

result_MPF=Euler_Method(MPF,f,t_step);

elseif Cdc13T>=0.7 & preMPF>=0.5 & Rum1T <0.15

f=@(x)(-1)*x+0.2;

result_MPF=Euler_Method(MPF,f,t_step);

elseif Cdc13T>=0.7 & preMPF<0.5 & Rum1T >=0.15

f=@(x)(-1)*x+0.01;

result_MPF=Euler_Method(MPF,f,t_step);

elseif Cdc13T>=0.7 & preMPF<0.5 & Rum1T <0.15

f=@(x)(-1)*x+0.2;

result_MPF=Euler_Method(MPF,f,t_step);

elseif Cdc13T<0.7 & preMPF>=0.5 & Rum1T >=0.15

f=@(x)(-1)*x+0.01;

result_MPF=Euler_Method(MPF,f,t_step);

elseif Cdc13T<0.7 & preMPF>=0.5 & Rum1T <0.15
```

```
f=@(x)(-1)*x+1.5;

result_MPF=Euler_Method(MPF,f,t_step);

elseif Cdc13T<0.7 & preMPF<0.5 & Rum1T >=0.15

f=@(x)(-1)*x+0.01;

result_MPF=Euler_Method(MPF,f,t_step);

elseif Cdc13T<0.7 & preMPF<0.5 & Rum1T <0.15

f=@(x)(-1)*x+0.01;

result_MPF=Euler_Method(MPF,f,t_step);

end

end

% Cdc13T Module**********

function result_Cdc13T = Cdc13T_module(Cdc13T,M,Ste9,Slp1,t_step)

if M>=1.75 & Ste9>=0.5 & Slp1>=1

f=@(x)(-1)*x+0.1;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);

elseif M>=1.75 & Ste9>=0.5 & Slp1<1

f=@(x)(-1)*x+0.3;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);
```

```
elseif M>=1.75 & Ste9<0.5 & Slp1>=1

f=@(x)(-1)*x+0.6;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);

elseif M>=1.75 & Ste9<0.5 & Slp1<1

f=@(x)(-1)*x+0.5;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);

elseif M<1.75 & Ste9>=0.5 & Slp1>=1

f=@(x)(-1)*x+1.5;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);

elseif M<1.75 & Ste9>=0.5 & Slp1<1

f=@(x)(-1)*x+1.5;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);

elseif M<1.75 & Ste9<0.5 & Slp1>=1

f=@(x)(-1)*x+1.5;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);

elseif M<1.75 & Ste9<0.5 & Slp1<1

f=@(x)(-1)*x+1.5;

result_Cdc13T=Euler_Method(Cdc13T,f,t_step);
```

```
end

end

%preMPF module

function result_preMPF =

preMPF_module(preMPF,Cdc13T,Wee1,Cdc25,Ste9,Slp1,t_step)

if Cdc13T>=0.5 & Ste9>=0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9>=0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9>=0.5 & Wee1>=1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9>=0.5 & Wee1>=1 & Slp1<0.8 & Cdc25<4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9>=0.5 & Wee1<1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;
```

```
result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9>=0.5 & Wee1<1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9>=0.5 & Wee1<1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9>=0.5 & Wee1<1 & Slp1<0.8 & Cdc25<4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1>=1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;
```

```
result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1>=1 & Slp1<0.8 & Cdc25<4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1<1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1<1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1<1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+1.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T>=0.5 & Ste9<0.5 & Wee1<1 & Slp1<0.8 & Cdc25<=4

f=@(x)(-1)*x+1.4;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+0.3;
```

```
result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+0.3;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1>=1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+0.3;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1>=1 & Slp1<0.8 & Cdc25<4

f=@(x)(-1)*x+0.3;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1<1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+0.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1<1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+0.5;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1<1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+0.6;
```

```
result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9>=0.5 & Wee1<1 & Slp1<0.8 & Cdc25<4

f=@(x)(-1)*x+0.6;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+0.3;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1>=1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+0.3;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1>=1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+0.3;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1>=1 & Slp1<0.8 & Cdc25<4

f=@(x)(-1)*x+0.3;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1<1 & Slp1>=0.8 & Cdc25>=4

f=@(x)(-1)*x+0.6;
```

```
result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1<1 & Slp1>=0.8 & Cdc25<4

f=@(x)(-1)*x+0.6;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1<1 & Slp1<0.8 & Cdc25>=4

f=@(x)(-1)*x+0.8;

result_preMPF=Euler_Method(preMPF,f,t_step);

elseif Cdc13T<0.5 & Ste9<0.5 & Wee1<1 & Slp1<0.8 & Cdc25<=4

f=@(x)(-1)*x+0.8;

result_preMPF=Euler_Method(preMPF,f,t_step);

end

end

% Rum1T Module**********

function result_Rum1T = Rum1T_module(Rum1T,SK,MPF,t_step)

if MPF>=0.3 & SK>=0.8

f=@(x)(-1)*x+0.02;

result_Rum1T=Euler_Method(Rum1T,f,t_step);

elseif MPF>=0.3 & SK<0.8
```

```
f=@(x)(-1)*x+0.02;

result_Rum1T=Euler_Method(Rum1T,f,t_step);

elseif MPF<0.3 & SK>=0.8

f=@(x)(-1)*x+0.11;

result_Rum1T=Euler_Method(Rum1T,f,t_step);

elseif MPF<0.3 & SK<0.8

f=@(x)(-1)*x+0.3;

result_Rum1T=Euler_Method(Rum1T,f,t_step);

end

end

% Ste9 Module**********

function result_Ste9 = Ste9_module(Ste9,MPF,Slp1,SK,t_step)

if Slp1>=0.1 & MPF>=0.3 & SK>=0.7

f=@(x)(-1)*x+0;

result_Ste9=Euler_Method(Ste9,f,t_step);

elseif Slp1>=0.1 & MPF>=0.3 & SK<0.7

f=@(x)(-1)*x+1;

result_Ste9=Euler_Method(Ste9,f,t_step);
```

```
elseif Slp1>=0.1 & MPF<0.3 & SK>=0.7

f=@(x)(-1)*x+1;

result_Ste9=Euler_Method(Ste9,f,t_step);

elseif Slp1>=0.1 & MPF<0.3 & SK<0.7

f=@(x)(-1)*x+1;

result_Ste9=Euler_Method(Ste9,f,t_step);

elseif Slp1<0.1 & MPF>=0.3 & SK>=0.7

f=@(x)(-1)*x+0;

result_Ste9=Euler_Method(Ste9,f,t_step);

elseif Slp1<0.1 & MPF>=0.3 & SK<0.7

f=@(x)(-1)*x+1;

result_Ste9=Euler_Method(Ste9,f,t_step);

elseif Slp1<0.1 & MPF<0.3 & SK>=0.7

f=@(x)(-1)*x+0;

result_Ste9=Euler_Method(Ste9,f,t_step);

elseif Slp1<0.1 & MPF<0.3 & SK<0.7

f=@(x)(-1)*x+1;

result_Ste9=Euler_Method(Ste9,f,t_step);
```

```
end

end

% Wee1 Module*************

function result_Wee1 = Wee1_module(Wee1,MPF,t_step)

if MPF>=0.25

f=@(x)(-1)*x+0.2;

result_Wee1=Euler_Method(Wee1,f,t_step);

else

f=@(x)(-1)*x+1.3;

result_Wee1=Euler_Method(Wee1,f,t_step);

end

end

% Cdc25 Module**********

function result_Cdc25 = Cdc25_module(Cdc25,MPF,t_step)

if MPF>=0.25

f=@(x)(-1)*x+5;

result_Cdc25=Euler_Method(Cdc25,f,t_step);

else
```

```
f=@(x)(-1)*x+0;

result_Cdc25=Euler_Method(Cdc25,f,t_step);

end

end

% Slp1 Module*************

function result_Slp1 = Slp1_module(Slp1,MPF,SK,t_step)

global MPF_array;

global t;

if t>15

if MPF_array(int32(t*200-3000),2)>=0.25

f=@(x)(-1)*x+2.5;

result_Slp1=Euler_Method(Slp1,f,t_step);

else

f=@(x)(-1)*x+0;

result_Slp1=Euler_Method(Slp1,f,t_step);

end

else

result_Slp1=0;
```

```
end

end
```

B.4 MatLab Code for Approximating DDE by ode23 and HDS with Delay

Matlab Code for Approximating DDE by ode23

```
%*************************************
%Simulation of x'(t)=-x(t-1) by dde23
% (C) Mustafa KAHRAMAN
%*************************************
function result = Simulation_DDE_dde23
sol = dde23(@ddex1dee,[1],@ddex1history,[0, 5]);
figure;
hold on;
hold all;
yy=ddex1history(-1);
for t=-0.99:0.01:0
yy=[yy;ddex1history(t)];
end
plot(transpose(-1:0.01:0),yy);
plot(sol.x,sol.y)
plot([0 0],[-1,2.5]);

title('dy/dt=-y(t-1)');

xlabel('time t');

ylabel('solution y');

end

function dydt = ddex1dee(t,y,Z)

ylag1 = Z(:,1);

dydt = [ -1*ylag1(1)];

end

function s = ddex1history(t)

x =[1.8947;5.6899;44.9378; 119.1792;131.0523;52.1560];

s = x(1)+x(2)*t+x(3)*t^2+x(4)*t^3+x(5)*t^4+x(6)*t^5;

end
```

MatLab Code for Approximating DDE by HDS with Delay

```
%***********************************

% Simulation of x'(t)=-x(t-1) by

% Hybrid Dynamical Systems with Delay

% (C) Mustafa KAHRAMAN
```

```
%*****************************************
function result = Simulation_DDE_HDS
%time**********************
delay=1;
t_step=0.001;
t_initial=0;
t_final=5;
%**************************
%initial function*********
y=ddex1history(t_initial-delay)
for t_i=(t_initial-delay+t_step):t_step:t_initial
y=[y;ddex1history(t_i)];
end
%**************************
index=int32(1/t_step)*delay+1;
s=y(index);
for t=t_initial+t_step:t_step:t_final
index=int32(t*(1/t_step));
```

```
y_delay = y(index);

k=0;

%Thresholds

if y_delay>=-2 & y_delay<-1.5

k=-1.75;

elseif y_delay>=-1.5 & y_delay<-1

k=-1.25;

elseif y_delay>=-1 & y_delay<-0.5

k=-0.75;

elseif y_delay>=-0.5 & y_delay<0

k=-0.25;

elseif y_delay>=0 & y_delay<0.5

k=0.25;

elseif y_delay>=0.5 & y_delay<1

k=0.75;

elseif y_delay>=1 & y_delay<1.5

k=1.25;

elseif y_delay>=1.5 & y_delay<2
```

```
k=1.75;

elseif y_delay>=2 & y_delay<2.5

k=2.25;

end

f=@(x)-1*k;

y_new=Euler_Method(s,f,t_step);

s=y_new;

y=[y;y_new];

end

figure;

hold on;

hold all;

plot(transpose(t_initial-delay:t_step:t_final),y);

plot([0 0],[-1,2.5]);

title('dy/dt=-y(t-1)');

xlabel('time t');

ylabel('solution y');

end
```

```
%Evaluates the y'n+1' value according to previous point information

%using Euler's method

%Inputs:

% y_n –> previous y value

% fy –> differential equation (function input)

% h –> step size

%

%Output:

% y_n_1 –> next y value

function y_n_1 = Euler_Method(y_n,fy,h)

f_tn_yn =fy(y_n);

y_n_1 = y_n + f_tn_yn*h;

end

function s = ddex1history(t)

x =[1.8947;5.6899;44.9378; 119.1792;131.0523;52.1560];

s = x(1)+x(2)*t+x(3)*t^2+x(4)*t^3+x(5)*t^4+x(6)*t^5;

end
```