DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

REACHABILITY ANALYSIS OF LAC OPERON UNDER PARAMETER UNCERTAINTIES

by

Gökhan DEMİRKIRAN

June, 2012

İZMİR

REACHABILITY ANALYSIS OF LAC OPERON UNDER PARAMETER UNCERTAINTIES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Electrical and Electronics Engineering Program

by

Gökhan DEMİRKIRAN

June, 2012
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"REACHABILITY ANALYSIS OF LAC OPERON UNDER PARAMETER UNCERTAINTIES"** completed by **GÖKHAN DEMİRKIRAN** under supervision of **PROF.DR. CÜNEYT GÜZELİŞ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof.Dr. Cüneyt Güzeliş

Supervisor

Asst.Prof.Dr.G.K.Demir

(Jury Member)

Examining Committee Member

Asst Prof. Dr. A. ALPKOCAK

(Jury Member)

Examining Committee Member

Prof.Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to my supervisor Prof. Dr. Cüneyt Güzeliş. The co-operation is much indeed appreciated. Without the enthusiasm and help from him, I would not be able to accomplish this thesis. I also would like to thank to my friends and my family for their patience with absence of me and encouraging me while writing this thesis.

Gökhan DEMİRKIRAN

REACHABILITY ANALYSIS OF LAC OPERON UNDER PARAMETER UNCERTAINTIES

ABSTRACT

In this thesis, reachability analysis simulation challenges are discussed. In order to understand simulation challenges in higher orders, first one dimensional case is studied. And after that, the information and experience obtained from one dimensional case is used to simulate higher dimension systems and interpret the results. For one dimensional case, first, a trivial piecewise linear system is discussed and after that a real model of lac-T is studied in one dimension for certain parameters. In this model, how reachability analysis can be used for uncertainty of parameters is explained. And after that, a 3 dimensional gene-protein relationship case is considered, for fixed partition and dynamic partition. Results are interpreted.

Keywords: Reachability analysis, lac-operon.

PARAMETRE BELİRSİZLİĞİ ALTINDA LAC OPERON İÇİN ERİŞİLEBİLİRLİK ANALİZİ

ÖZ

Bu tezde, erişilebilirik analizinin zorlukları üzerine durulmuştur. Yüksek mertebeli sistem modellerinde, erişilebilirlik analizi yapabilmek için, öncelikle tek boyutta erişilebilirlik analizi çalışmaları yapılıp, gerekli tecrübe ve bilgi edinilmiştir. Bu bilgiler daha sonra, yüksek mertebeli sistem modellerinin simulasyonlarını yapma ve sonuçları yorumlamada kullanılmıştır. Bir boyut için, öncelikle uydurulmuş parça-parça doğru bir sistem çalışılmış daha sonra , lac-T nin tek boyutta erişilebilirlik analizi yapılmıştır. Ayrıca, parametrelerin belirsizliği durumunda erişilebilirlik analizi kullanılmıştır. Daha sonra, 3 boyutlu bir protein-gen modelinin simulasyonu sabit ve dinamik bölme için yapılmıştır.

Anahtar sözcükler: Erişilebilirlik analizi, lac-operon

CONTENTS

CHAPTER ONE
INTRODUCTION

1.1 Systems Biology

Systems biology is the study of biological systems in terms of dynamical systems theory or control theory. As the name suggests, it tries to understand the biology in system level, because, biological systems cannot be understood by its constituents alone. Regulatory systems, molecular biology, enzyme kinetics are the areas of the systems biology. The motivation is that, if we can understand biology in system level, then we can make any biological system do what we want and control it. In biological systems, parameters cannot be determined accurately. So we need to find a way to study and model the biological systems. To model a biological system, rate equation must be understood first. After modeling a biological system, we will see that parameters play important role and they are usually uncertain and reachability analysis must be done.

1.2 Interpreting Differential Equations in the Sense of Rate of Change

A simple differential equation is in the form of:

$$\frac{dy}{dt} = k * y \tag{1-1}$$

And the solution to this equation is:

$$y = C * e^{k*t} \tag{1-2}$$

C is determined from the initial conditions. Differential equations give the rate of change of a variable. If t is time, then $\frac{dy}{dt}$ is the rate of change of y with time and it is proportional to the value of the variable "y" itself multiplied with a constant k. The bigger the k, the rapid is the rate of change.

1.2.1 Newton's Law of Cooling

I thought that it would be a good starting point to mention about Newton's law of cooling in order to understand linear differential equations in the sense of rate of change. Everyone who is familiar with the calculus heard about Newton's cooling law. It states that the heat loss rate of a body is proportional to difference between its own temperature and the temperature of its surroundings. When we translate this sentence into differential equation we obtain:

$$\frac{dT}{dt} = -k(T - Ts) \tag{1-3}$$

Where T is the temperature of the body and Ts is the temperature of the surroundings. And k is positive constant so that –k is negative and the rate of change of temperature is decreasing. We know the solution of a differential equation of "$\frac{\mathrm{dy}}{\mathrm{dt}} = \mathrm{k} * \mathrm{y}$" is in the form of $y = c * e^{k*t}$. With a few modifications we can obtain c=T0-Ts and adding final value to the equation:

$$\mathrm{T(t) = Ts + (T0 - Ts) * e^{-kt}} \tag{1-4}$$

Simulating this solution for k=0.054, T0=95 and Ts=5:

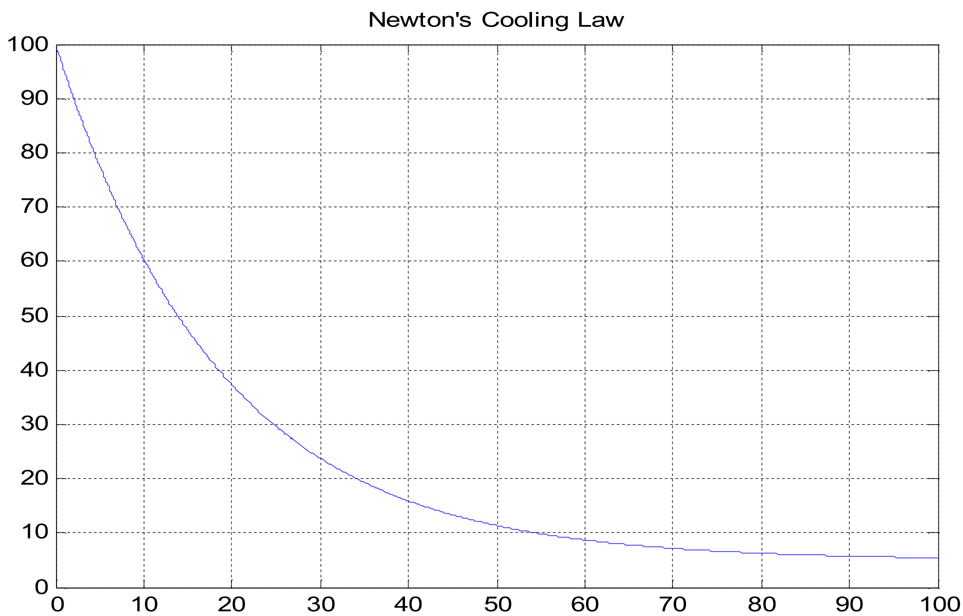


Figure 1.1 Newton's cooling law for t=0 to 100.

1.2.2 Radioactivity

Another similar problem is about radioactivity. The rate of the change of the mass (loss) of radioactive material is proportional to its mass. Mass is always positive so, turning this sentence into differential equation:

$$\frac{dm}{dt} = -km \tag{1-5}$$

As you see this equation has a well-known solution:

$$\mathrm{m(t)} = m0 * e^{-kt} \tag{1-6}$$

Think about a radioactive element with mass =10 gr and constant=1.2*10^-4 and half-life of 5600 years.

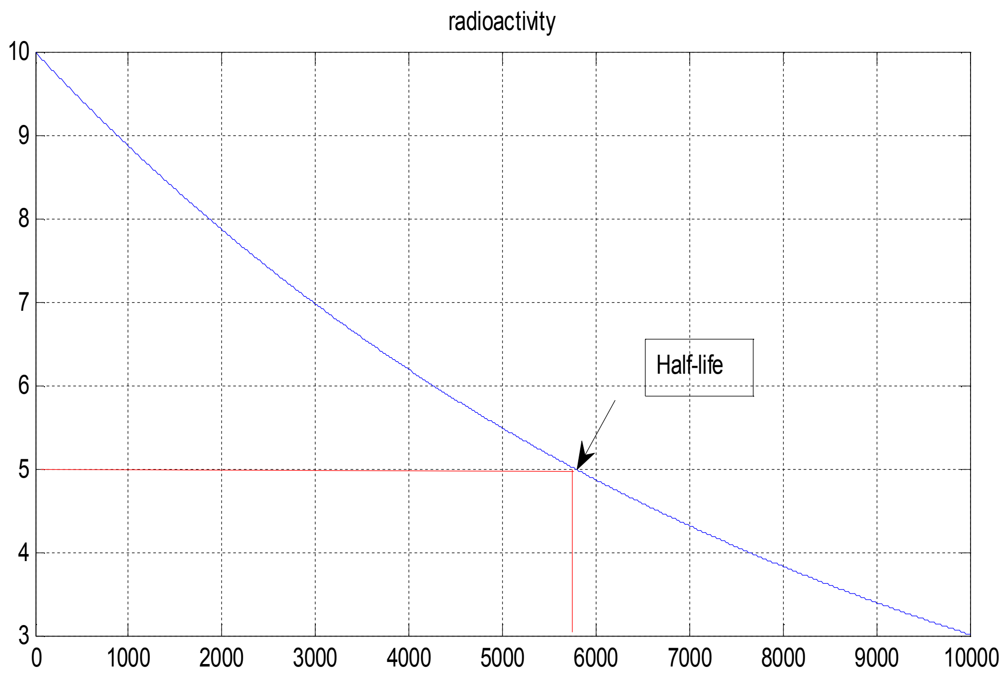


Figure 1.2 Radioactivity of a radioactive element.

Now let’s talk about a biological phenomenon.

1.2.3 Bacterial Growth

The rate of change of the population of bacteria is proportional to its present population. This means the more the population, the more the growth rate.So we can write the equation (1-1) again:

$$\frac{dy}{dt} = k * y \tag{1-7}$$

K is a positive constant, because the rate is positive.

If K, the grow rate, is 1.3 and initial population of 10000 then the solution becomes:

$$y = y0 * e^{kt} \tag{1-8}$$

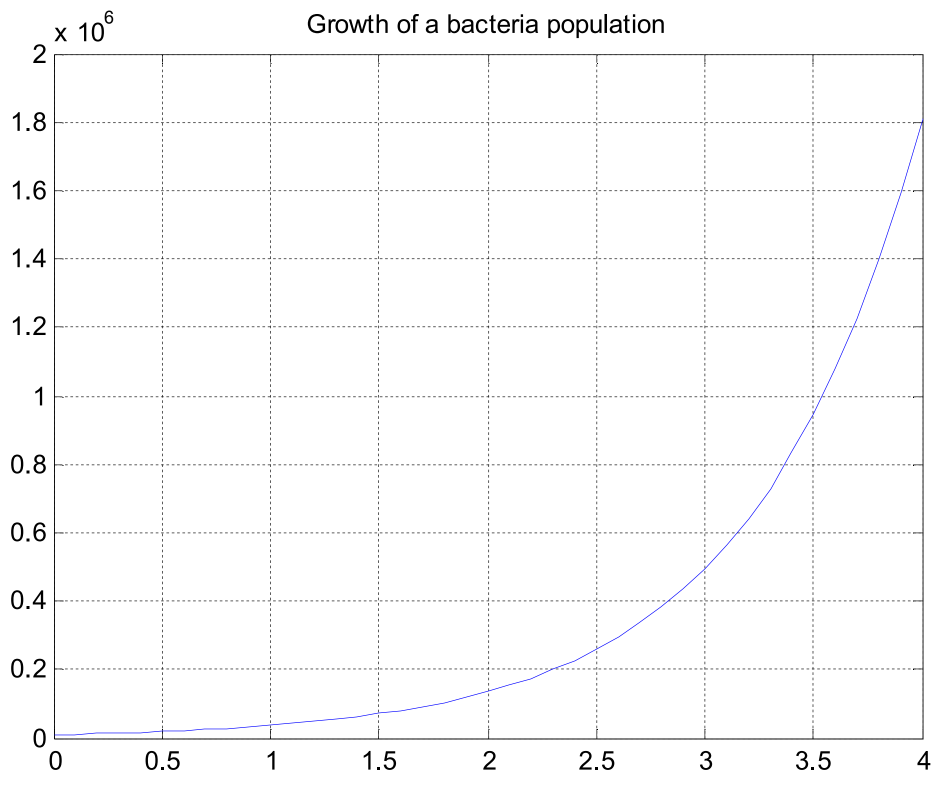


Figure 1.3 Growth of a bacteria population

As you see as the number increases growth increases exponentially. Even for t=10, it becomes $4.5*10^9$.

Something is not quite right in growth of bacteria population. Because the growth rate is proportional to its population and every bacteria divides into 2 bacteria. But the growth cannot go to infinity since the food in the environment is limited and will not support the population. So the population will decrease. This differential equation is missing some parameters. This example is trivial and it has nothing to do with real modeling. Bacterial growth curve is a nonlinear system and has been studied in detail. It is beyond our subject now. This simple idea is the core understanding of the modeling of biological systems.

In biological systems, nonlinear systems are a must. In order to understand nonlinear systems, first we must understand differential equations in the sense of linear system theory. So, now let's focus on stability of the systems in one dimension.

1.2 Stability in One Dimension

Differential equations occur whenever there is a rate of change, for example the rate of change of distance which is velocity or the rate of change of population or the rate of change of protein concentration in a cell etc.

In order to understand how the systems of differential equations will behave, let's talk about equilibria in Newton's cooling law. The system went to its surrounding temperature and in our radioactivity problem, it went to zero, which we call it stable, but in bacteria example, it went to infinity which is unstable. Let's explore this behavior in one dimension. Think about:

$$\frac{dx}{dt} = 4x \qquad (1\text{-}9)$$

The solution is:

$$x = x0 * e^{4t} \tag{1-10}$$

Figure 1.4 shows $\frac{dx}{dt}$ vs. x. The positive $\frac{dx}{dt}$ means positive rate of change which increases x and negative $\frac{dx}{dt}$ means negative rate of change which decreases x. So if x is any positive real number, so is $\frac{dx}{dt}$ and x will increase through infinity; if it is negative, so is $\frac{dx}{dt}$ and it will decrease through minus infinity. So as you see this system is not stable unless you start with zero at which the system will stay forever since the rate of change will be zero.

$$\frac{dx}{dt} = 4x = 4 * 0 = 0 \tag{1.11}$$

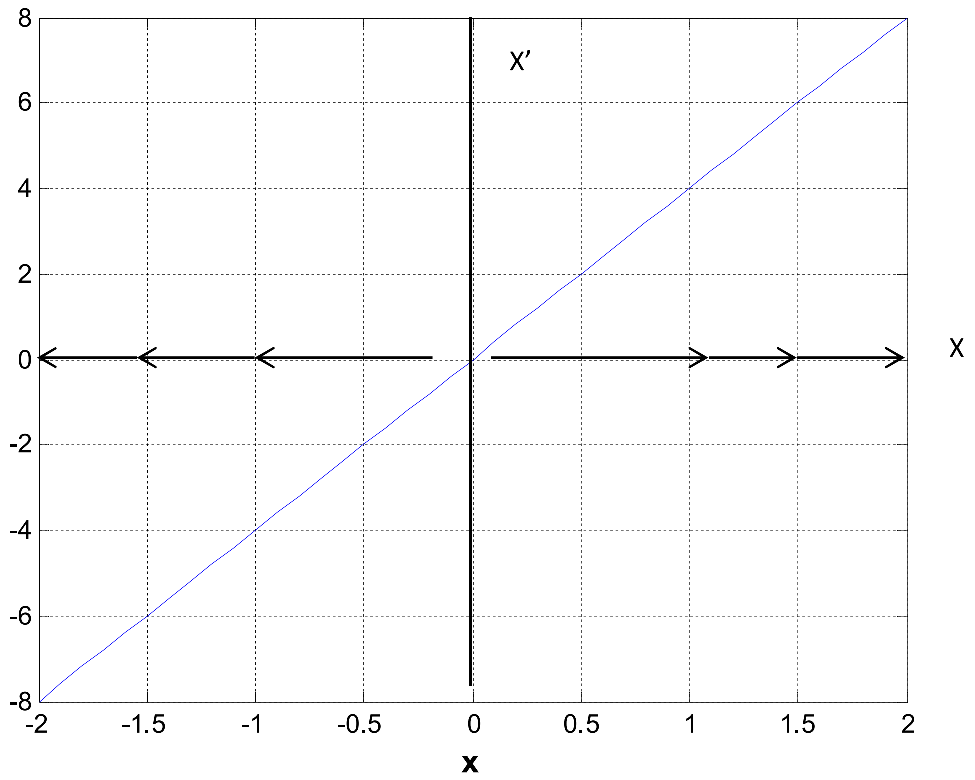


Figure 1.4 The equation (1-10), dx/dt vs. x.

Look at the arrows in Figure 1.4 and interpret. If we start at a point which is not exactly at zero, then the point will go to plus or minus infinity, which is not equilibrium.

Our starting points are -2, -1, 1 and 2. Wherever we begin, the starting point goes to plus or minus infinity. So we can't mention about equilibrium here, unless you start with zero. If you start at zero, it will always be zero. So, is zero an equilibrium point?

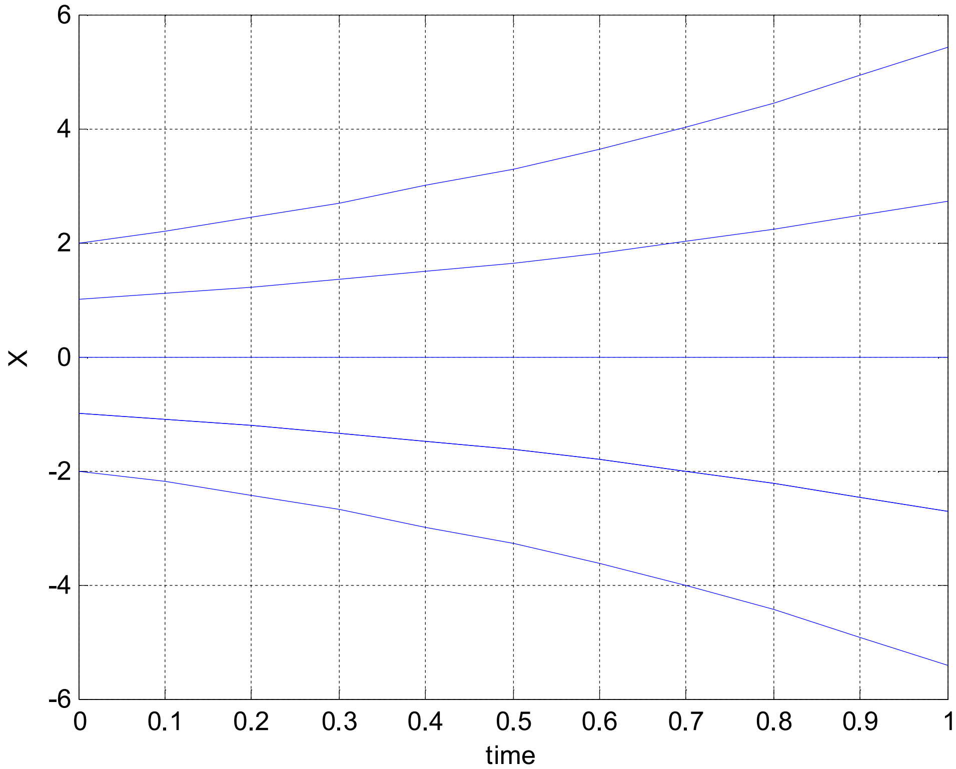


Figure 1.5 Simulation of the equation dx/dt=4x for different starting points

Suppose you started with an initial point very close to zero but not exactly zero: 10^{-9} and -10^{-9}. Then, these initial points will go to either minus or plus infinity. But if your initial point is zero, then it stays at zero. It is true that, it does not move away from zero if you start with zero. But, any noise around zero will take the point to infinity. Simulation is done below. You can see that the starting points go to infinity.

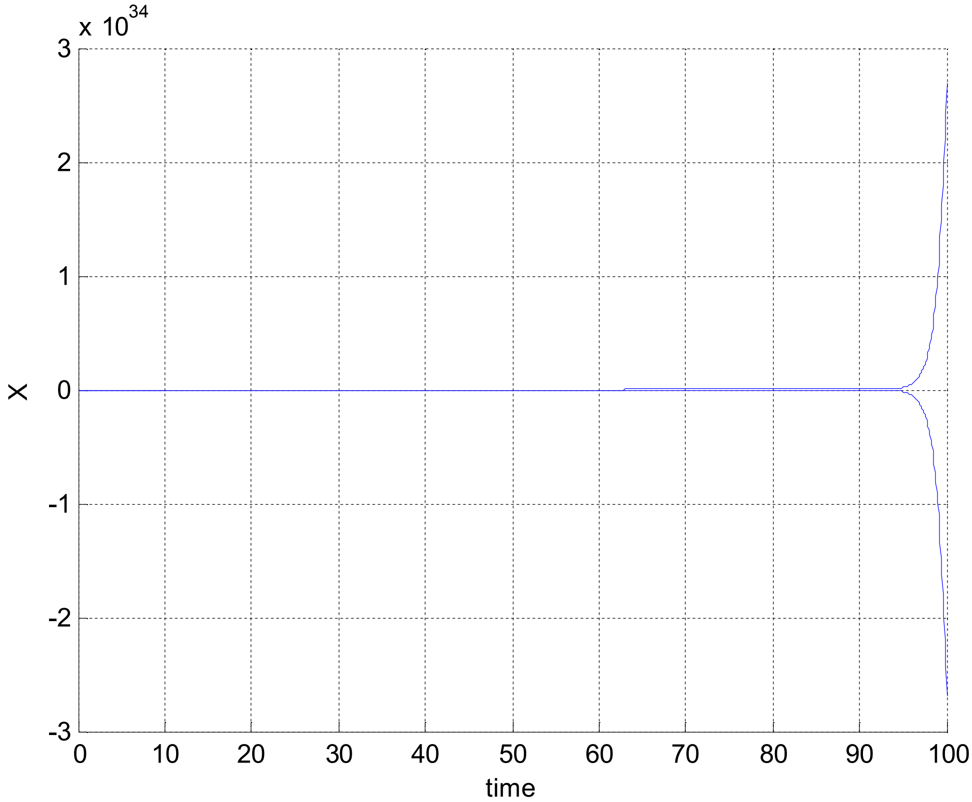


Figure 1.6 Starting with points that are very near to zero.

So we say that here zero is an equilibrium point, but not a stable equilibrium point. These points had to be taken into consideration in polytope calculations as you will see in following chapters. Now think about following differential equation:

$$\frac{dx}{dt} = 1 - x \tag{1-12}$$

Look at the arrows in Figure 1.70; wherever the starting points are, they go to point 1 which is the equilibrium point. You may not know the solution, how fast it goes to equilibrium point but, you can say that whatever the starting point is it will go to equilibrium point 1. The solution is:

$$x = 1 - e^{-t} \tag{1-13}$$

X' vs. x plot is below:

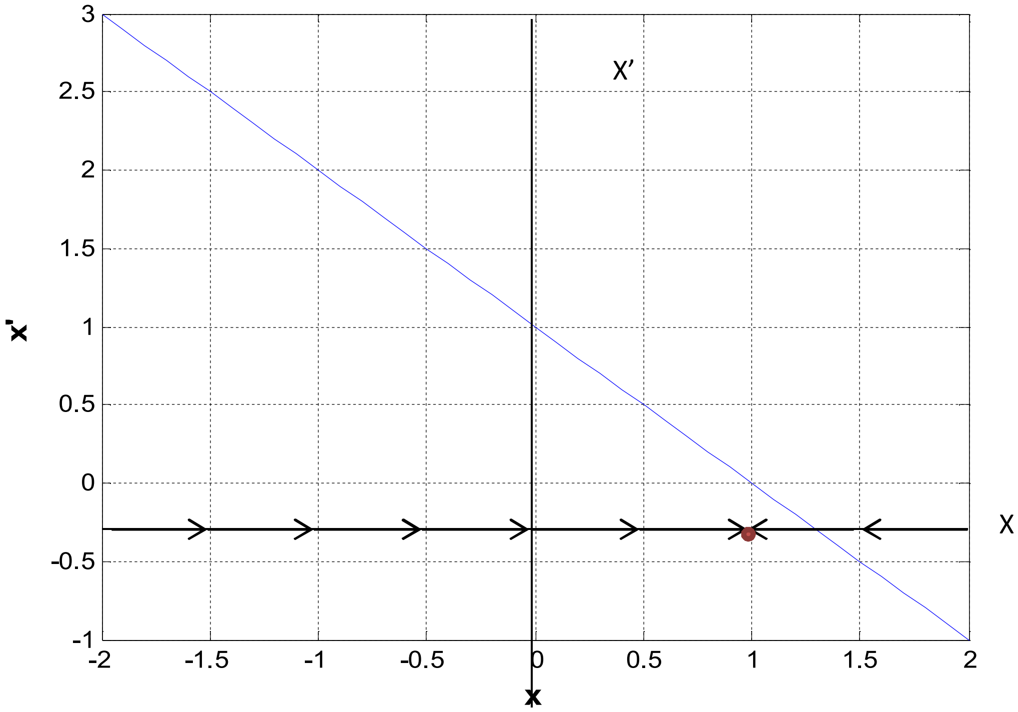


Figure 1.7 The equation (1-12), dx/dt vs. x.

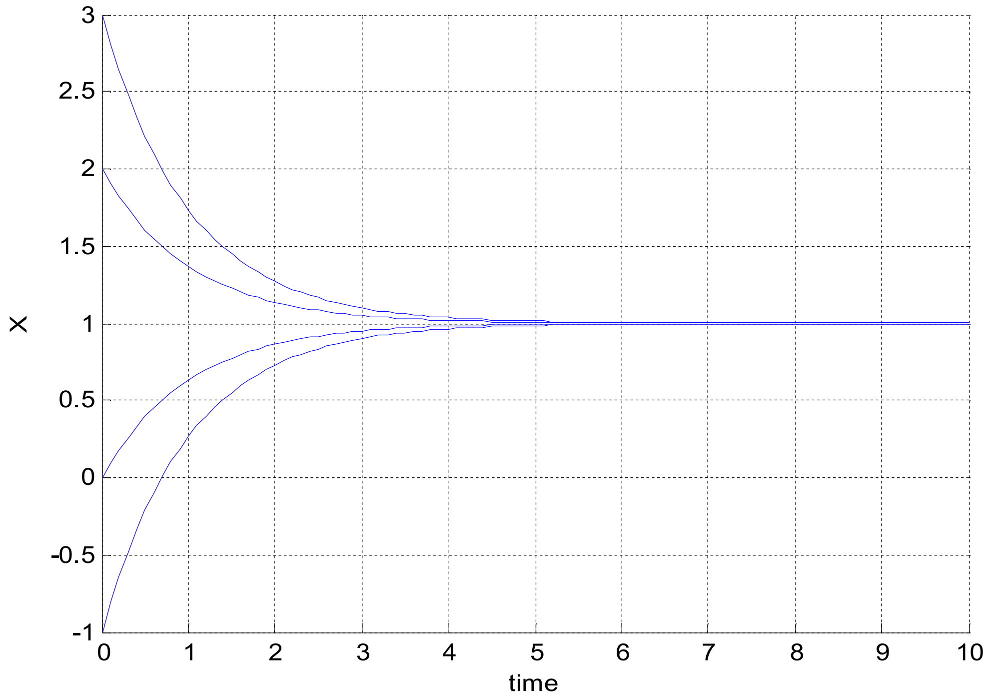


Figure 1.8 Simulation of points -1, 0, 2 and 3 for the equation (1-14)

It does not matter where you begin. The point goes to point 1. We call it an equilibrium point which is stable.

Figure1.6 is the unstable equilibrium; figure 1.8 is the stable equilibrium or an attractor. So we can say: Suppose that x* is an equilibrium point of differential equations. This equilibrium point is non-stable if dx/dt is greater than zero and stable if dx/dt is smaller than zero.

1.3.1 Stability of Logistic Population Growth

The equation (1-14) is the logistic equation of a population growth. But let's give values to r and k value from scratch, r=2 and k=3. Without knowing the solution function, let's guess where the critical points are and which are stable and which are nonstable as in figure 1.9.

$$\frac{dn}{dt} = r * n * (1 - n/k) \tag{1-14}$$

Let's discuss the equilibrium points of this differential equation by looking at the graph. Near zero, this equations is very similar to our first example $\frac{dx}{dt} = t$ and near 3, it is similar to our other example of $\frac{dx}{dt} = 1 - 1 * x$. So we say, if we start with a number close to zero and positive, it will increase, but this time not through infinity, but through 3. We call this point as repulsor. If we start with a number near zero and negative, then it will go to minus infinity. If we start with a number greater than 3, it will display the characteristics of second equation, it will decrease through 3. We call this point as attractor. Let's show this with a graph in figure 1.10:

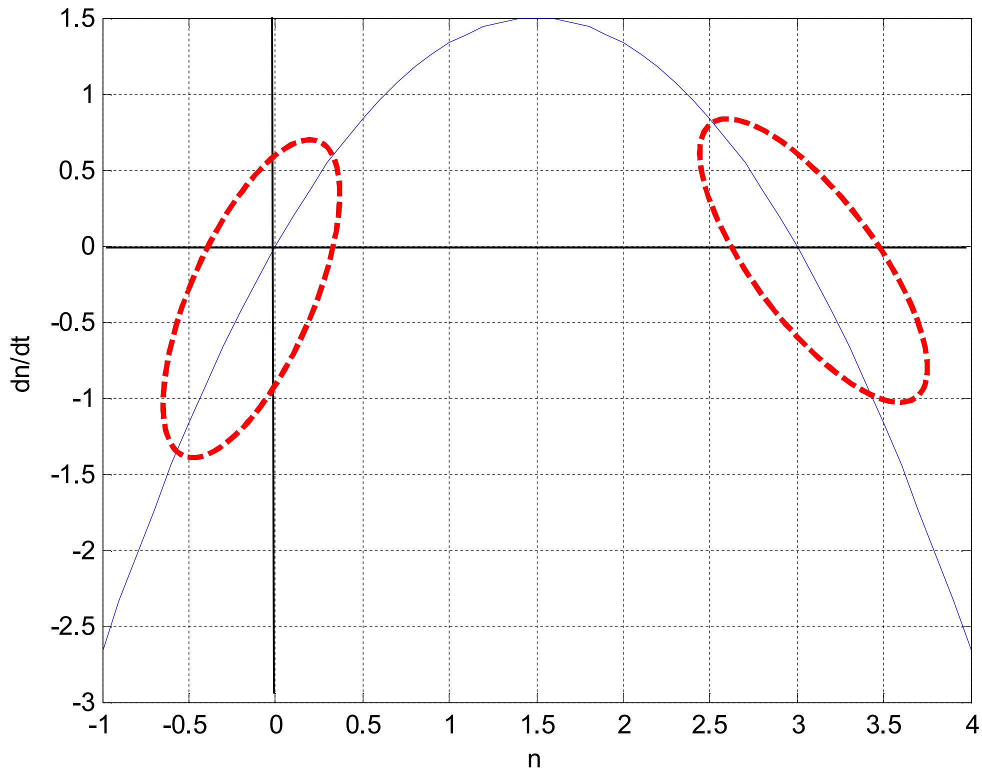


Figure 1.9 The equation (1-14), dn/dt vs. n

Arrows in figure 1.10 show us the way where the starting points will end. If you start with a number greater than 3 or between $0<x<3$ it will go to 3 and with a number smaller than zero, it will go to minus infinity. And we say the basin of attraction of equilibrium point 3 is (0, infinity).

The logical population model is a nonlinear model and I won't bother to explain the solution of it now. As you see, even the most simplistic model in biology is nonlinear. We have to understand linear systems which will help us to study nonlinear systems. This attractor and repulsor concept is critical. We have to understand it in one dimension. In higher dimensions, the results will be complex.

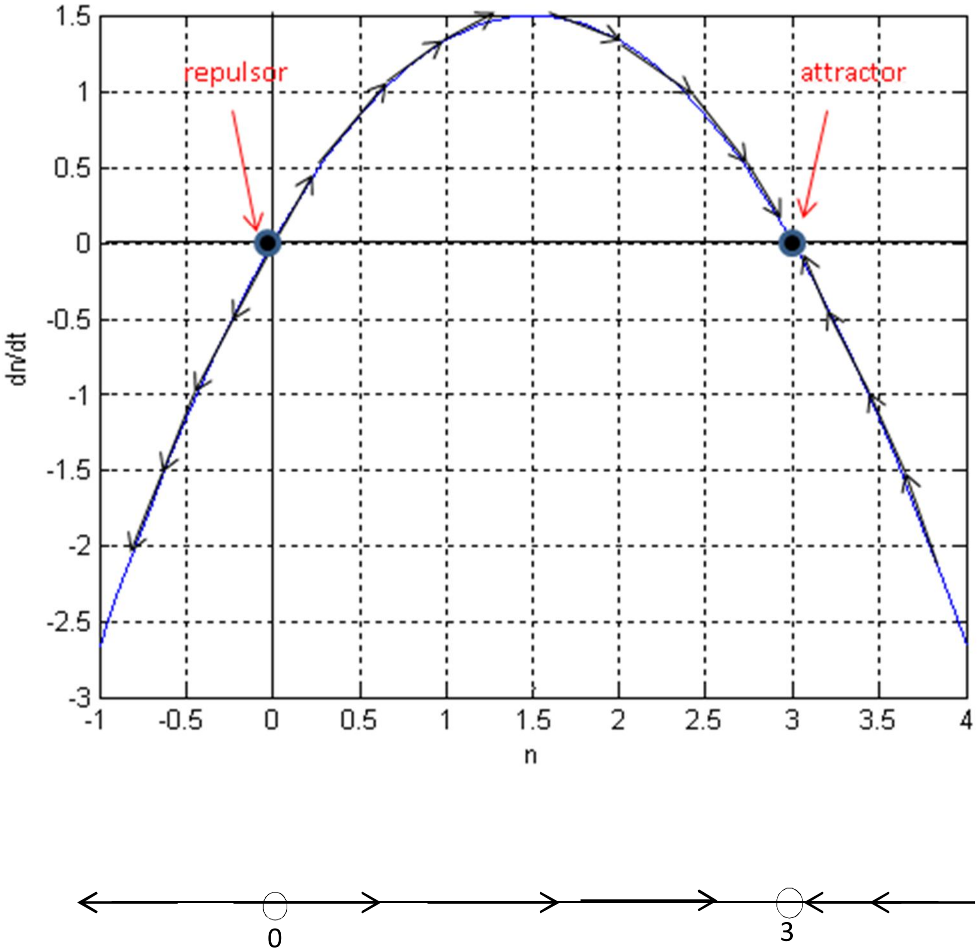


Figure 1.10 Direction of flows for equation (1-14)

1.4 Stability in Two Dimension

Critical points occur at the solution of the system where dx/dt=0; The differential equation below is linear. How do we know it? Because it does not have any product of X's components and can be written as follows:

$$X' = A * X \tag{1-15}$$

For 2 dimension A=

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{1-16}$$

Another form:

$$\frac{dx_1}{dt} = ax + by \tag{1-17.1}$$

$$\frac{dx_2}{dt} = cx + dy \tag{1-17.2}$$

Find the eigenvalues using:

$$Det(A - \lambda I) = 0 \tag{1-18.1}$$

$$Det\begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0 \tag{1-18.2}$$

Suppose eigenvalues of the equation below is:

$$X' = \begin{bmatrix} 1 & 4 \\ 1 & 1 \end{bmatrix} X \tag{1.19}$$

$\lambda_{1=}$-1, λ_2=3**.** The solution to the system is in the form of:

$$x = C_1 * e_1 * e^{\lambda_1 * t} + C_2 * e_2 * e^{\lambda_2 * t} \tag{1-20}$$

Where $e_1$and e_2 are eigenvectors and also the span of the solution space. But I don't bother to explain eigenvectors now. We will focus on eigenvalues. For this example, eigenvectors are:

$$e_1 = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \tag{1-21}$$

And

$$e_2 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \tag{1-22}$$

The examples I have given is Newton's law of cooling, bacteria growth and radioactivity is in one dimension and they are the same as $C_1 * e_1 * e^{\lambda_1 * t}$, the first part of the solution for two dimension. So you can guess how the system will behave. λ_1 is negative, so we say, as t increases, it will decrease in e_1 direction, but λ_2 is positive so it will increase in e_2 direction. What will be the flow arrows like in two dimension? If we plot them, we obtain:

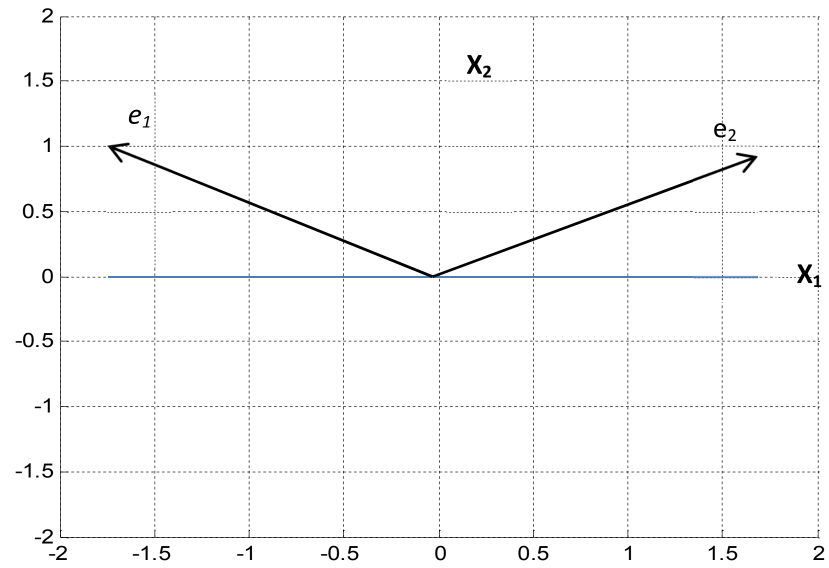


Figure 1.11 Eigenvectors of equation (1-19).

The direction flows would be:

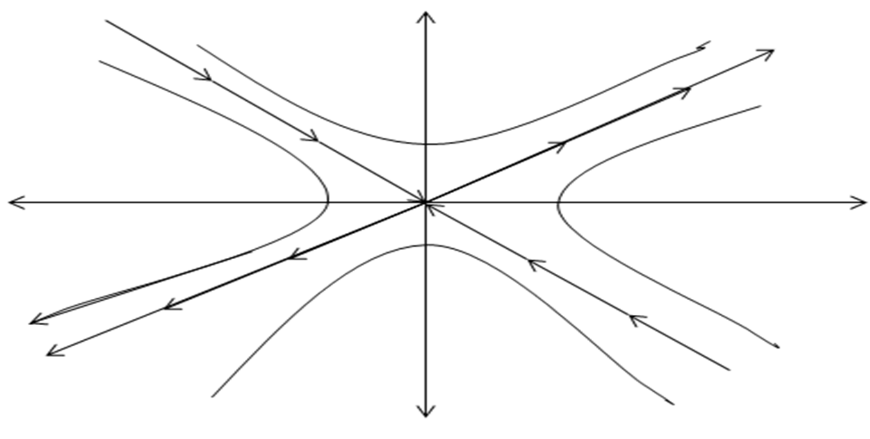


Figure 1.12 Direction flows of the equation (1-19)

What the arrows in figure 1.12 are telling is, if you start with any point, it will decrease in e1 direction and increase in e2 direction. This characteristic point is called saddle point. This was for the case, one of the eigenvalues is positive and one of the eigenvalues is negative. There are other cases where both of them negative, positive or complex. But this was just an example to show that in higher dimension, flows will be complex. And higher dimension set of equations are studied using Jacobean matrix.

CHAPTER TWO
HOW TO MODEL A BIOLOGICAL SYSTEM: A PROTEIN-GENE RELATIONSHIP

Every protein is synthesized from a gene. And some proteins activate genes so that it can start producing a protein. Let's say a protein A is produced from "gene a" and protein A activates "gene b", so that "gene b" can produce protein B. Concentration of proteins are our quantities.

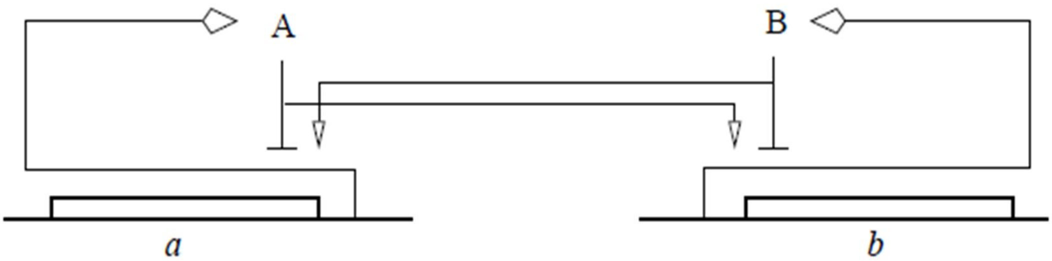


Figure 2.1 Example of a genetic regulatory network of two genes (a and b), each coding for a regulatory protein (A and B). K.W. Kohn. (2001).

Look at the figure 2.1 above. "a" stands for gene a and "b" stands for gene b. "A" stands for protein A, "B" stands for protein B. The arrow means activation, and $\perp$ means inhibition. Since every product of biological reaction is the inhibitor of its own reaction. Every protein is the inhibitor of its own production gene. So, "gene a", produces protein A; "gene b", produces protein B, protein A activates "gene b", protein B activates "gene a", protein A inhibits "gene a" and protein B inhibits "gene b". And also for a protein to activate a gene, its concentration must be above some threshold level. This relationship is an example of a gene regulatory network. Gene regulatory networks are perhaps the most important organization in cells. This network includes DNA, RNA, protein and small molecules in terms of activation and inhibition of genes. Proteins activate or inhibit the genes. And proteins are also the product of gene expression so there is a feedback. So protein-gene mechanisms can be modeled as differential equations. Turning this information into a set of differential equations will be examined. For threshold levels, we can use step functions:

$$s^+(x_j, \theta_j) = \begin{Bmatrix} 1, & x_j > \theta_j \\ 0, & x_j < \theta_j \end{Bmatrix} \quad (2\text{-}1)$$

$$s^-(x_j, \theta_j) = 1 - s^+(x_j, \theta_j) \quad (2\text{-}2)$$

Look at the function below, what this tells us is that: let x_a be concentration of protein A, if protein A is above some threshold level T_a, then, “gene b” is activated at k_a rate.

$$g_b(x_a) = k_a s^+(x_a, T_a) \quad (2\text{-}3)$$

The rate equations express a balance between the number of molecules appearing and disappearing per unit time.

$$\frac{dx_i}{dt} = g_i(x) - \gamma_i x_i \quad (2\text{-}4)$$

γ_i is degradation constant, since every product is the inhibitor of its production gene and g_i is the activation function preserving the characteristics of positive feedback. - γ_i can be thought of as negative feedback loop. And now for the figure 2.1, we can write the equations:

$$\frac{dx_a}{dt} = K_a * s^+(x_b, \theta_b^1) * s^-(x_a, \theta_a^2) - \gamma_a * x_a \quad (2\text{-}5)$$

$$\frac{dx_b}{dt} = K_b * s^+(x_a, \theta_a^1) * s^-(x_b, \theta_b^2) - \gamma_b * x_b \quad (2\text{-}6)$$

Since every protein is the inhibitor of its production, degradation factor is needed. For $x_a$if x_b is above threshold level θ_b^1 and $x_a$is below a threshold level θ_a^2, then "gene a" is active at k_a rate. And the same rule applies to B. If concentration of protein A is above threshold level θ_a^1 and concentration of B is below some threshold level θ_b^2, which means concentration of b will be produced till there is enough concentration, then "gene b" is active at K_b rate.

We can write more complex gene regulatory network:

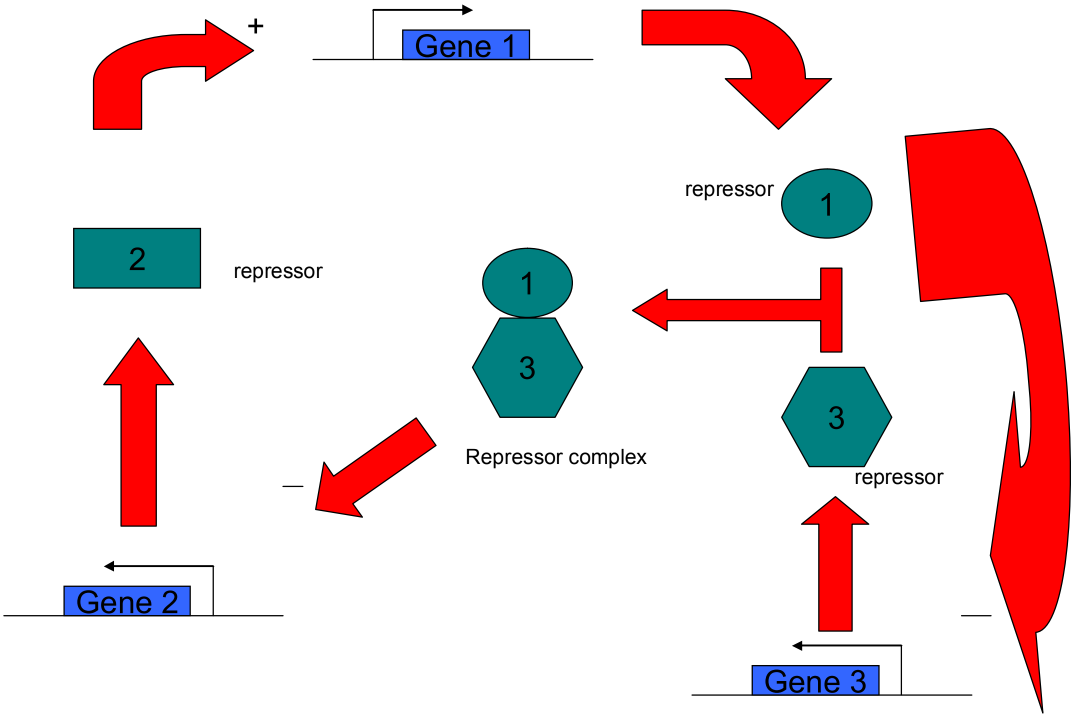


Figure 2.2 A protein and gene relationship.

The positive sign on the arrows means activating and negative sign means inhibition:

$$\dot{x}_1 = \kappa_1 s^+(x_2, \theta_{21}) - \gamma_1 x_1 \tag{2-7.1}$$

$$\dot{x}_2 = \kappa_2 (1 - s^+(x_1, \theta_{11}) s^+(x_3, \theta_{31})) - \gamma_2 x_2 \tag{2-7.2}$$

$$\dot{x}_3 = \kappa_3 s^-(x_1, \theta_{12}) + \kappa_4 s^-(x_3, \theta_{32}) - \gamma_3 x_3 \tag{2-7.3}$$

Now look at the figure 2.3 below:

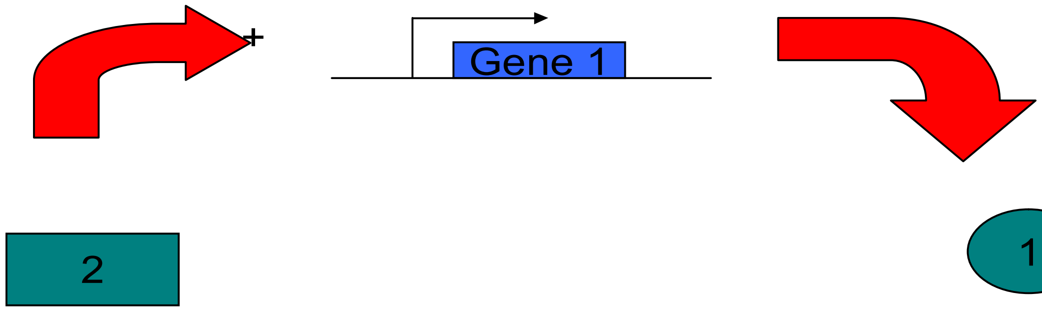


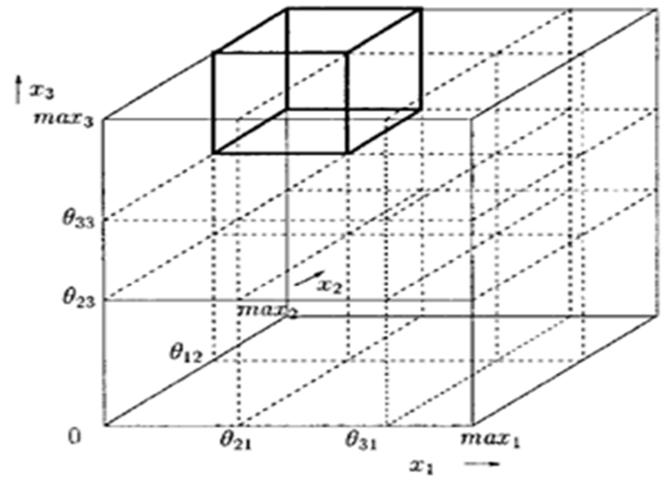
Figure 2.3 A gene-protein relationship, where protein 2 is activator and gene1 produces protein 1.

$$\dot{x}_1 = \kappa_2 - \gamma_1 x_1 \qquad (2\text{-}8.1)$$

$$\dot{x}_1 = \kappa_2 s^+(x_2, \theta_{21}) - \gamma_1 x_1 \qquad (2\text{-}8.2)$$

For x_2 greater than θ_{21}, this formula is just $K_2 - \gamma_2 * x_2$. You know, this formula has a well-known solution, $\frac{K}{\gamma_1} + e^{-\gamma_1 t}$. So if you divide 3 dimensional space into threshold states, which means into orthants, then for every box there is a different set of linear differential equations and no couple.

If you look at figure 2.4, you will see that for different threshold levels, there are different differential functions. For the bold box, the set of equations are given next to the figure 2.4. For different regions, this network works different with a different set of differential equations. In the box, the equations are linear anymore. Any nonlinear system state space can be divided into linear boxes. This is how we study nonlinear systems in terms of linear systems. This is called hybridization.



$$\dot{x}_1 = \kappa_{12} - \gamma_1 x_1$$

$$\dot{x}_2 = -\gamma_2 x_2$$

$$\dot{x}_3 = \kappa_{31} - \gamma_3 x_3$$

$$\dot{\boldsymbol{x}}_i = \boldsymbol{\mu}_i - \gamma_i \boldsymbol{x}_i, \qquad \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{n}$$

Figure 2.4 Phase space of the model divided into 2x3x3= 18 orthants by the threshold planes. The state equations for the orthant 0 · x1 < μ21, μ12 < x2 · *max*2, and μ33 < x3 · *max*3 is given. (the orthant demarcated by bold lines). Hidde de Jong. (2002)

CHAPTER THREE
REACHABILITY

In order to understand reachability, first let's examine discrete states:

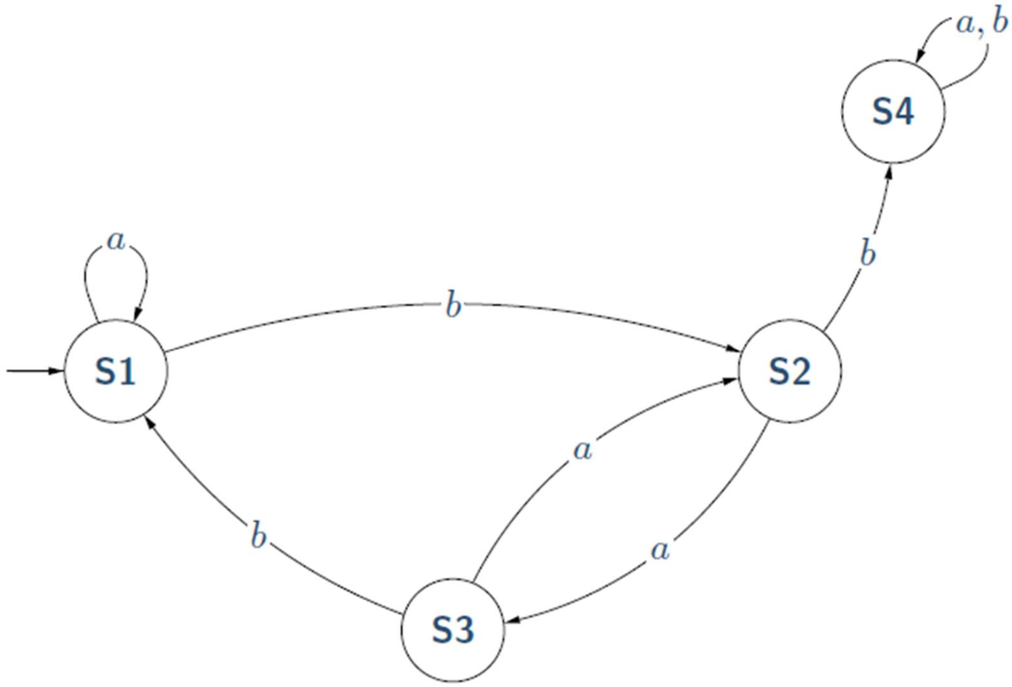


Figure 3.1 A system with discrete states

Suppose S1, S2, S3 and S4 are the states of a system and "a" and "b" are the inputs. Suppose the system is in S1 state now, and inputs are "abbab", then states will go through this trajectory: S1- S1- S2- S4- S4- S4. And the system will stay at S4 whatever the input is. Then we say that S4 is reachable from S1 through "abb".let's say S4 is a bad state, then if you don't want to go to S4 then your inputs must be aaaaaa or inputs must be abaaaaa or abab… etc.

And now think about a hybrid dynamic system:

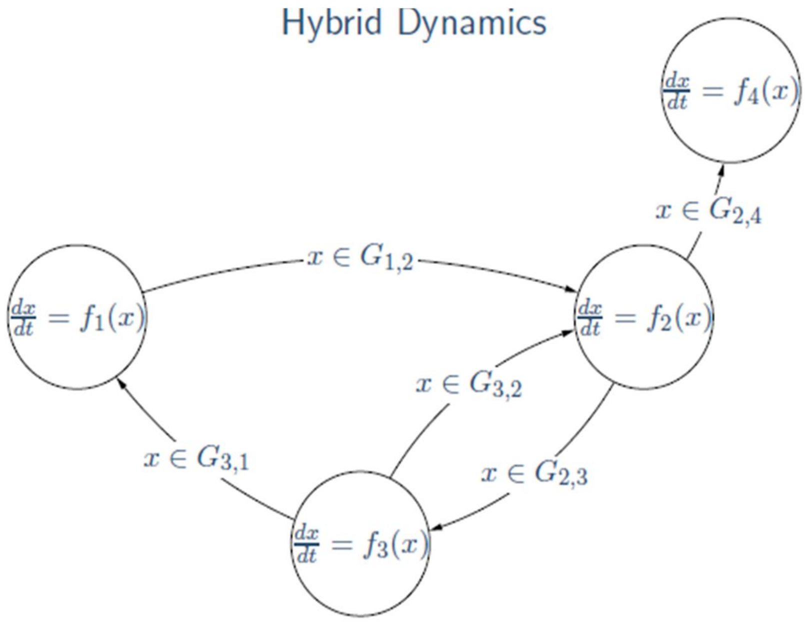


Figure 3.2 A system of Hybrid Dynamics

The figure 3.2 tells us that: suppose you are in state1, then you system acts according to the equation dx/dt = f_1(x) until x belongs to this state, when $x \in G_{1,2}$.Then a new state will appear with a different set of equations.

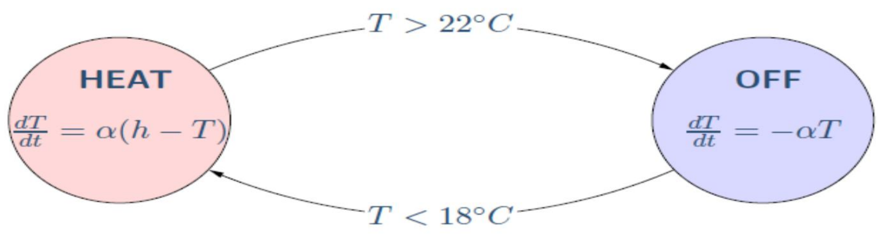


Figure 3.3 A climate system is an example of hybid dynamics system. Le Guernic,C. (October 28, 2009).

For any system, if we can name a state as bad state, then we can then we can try to simulate, if any state will go bad state under some conditions, this can be any system with states, a computer network, or a cell, or a nuclear plant etc. In biological networks, molecular concentrations, reaction rates and other parameters cannot be determined with %100 precision. So you need to take a space that contains the uncertainty of parameter and then make a reachability analysis.

If we start with a set of points, let's say P0, then we can compute another set of points P1 which contains all the points reachable from P0. This set of points must be chosen in a format of convex polytope.

3.1 Polytopes

A polytope is set of points indicated by linear inequalities:

$$m * x \leq b \tag{3.1}$$

M is a real matrix and b is a real vector. You can find the positions of the vertices given by the above equations using vertex enumeration method. A polytope example in 2 dimension: Vertices of the polytope is (1,2);(2,4);(1,3) and inequality equation is:

$$\begin{bmatrix} -0.70711 & 0.70711 \\ -1 & -0 \\ 0.89443 & -0.44721 \end{bmatrix} \mathbf{x} <= \begin{bmatrix} 1.4142 \\ -1 \\ 0 \end{bmatrix} \tag{3-2}$$

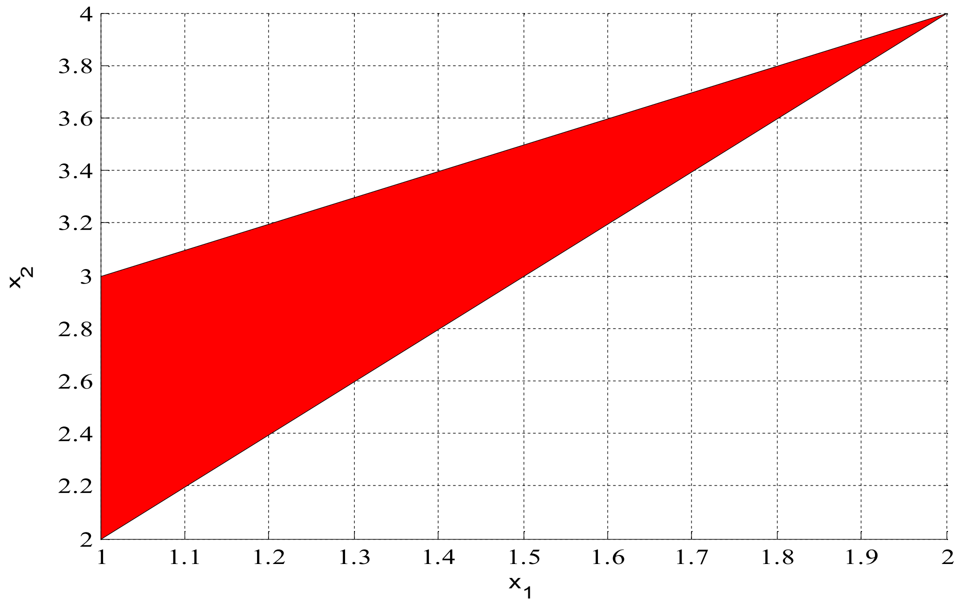


Figure 3.4 Polytope indicated by the equation (3-2).

Another polytope example in 2-dimension. The polytope has the vertices of (0.90, 1.1), (1.2, 1.3), (1.3, 1.5), (0.95, 4.5). The equation of inequality is:

$$\begin{bmatrix} 0.89443 & -0.44721 \\ -0.70711 & 0.70711 \\ 0.5547 & -0.83205 \end{bmatrix} x <= \begin{bmatrix} 0.49193 \\ 0.14142 \\ -0.41603 \end{bmatrix} \tag{3-3}$$

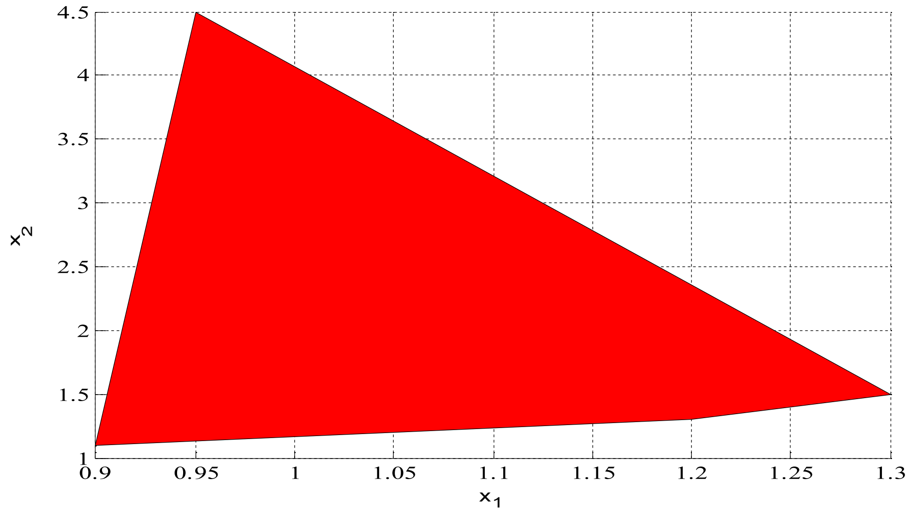


Figure 3.5 Polytope indicated by the equation (3-3).

A polytope in 3 dimension would be:

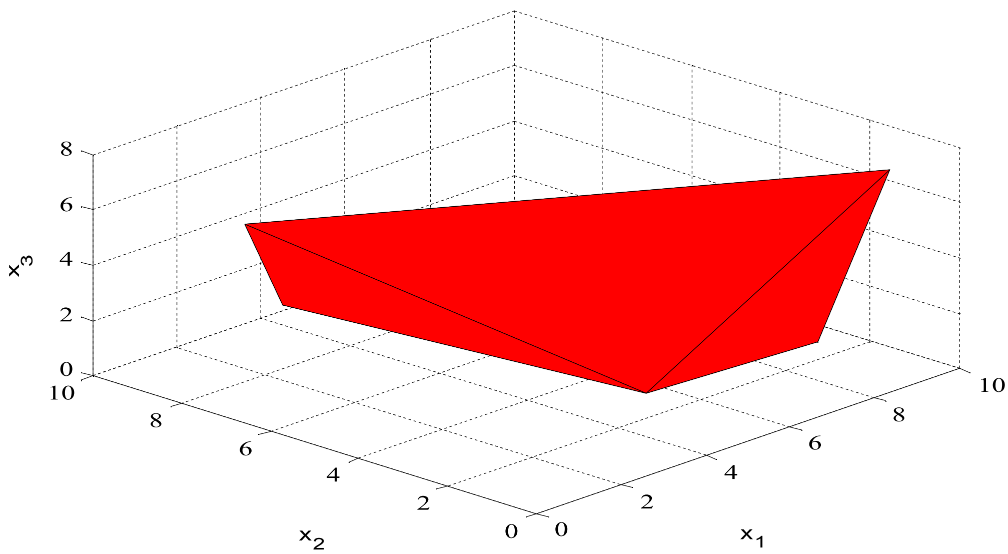


Figure 3.6 A polytope in 3 dimension.

If we choose X_1, X_2 and X_3 three states, number of vertices must be at least 3+1=4. As you see for 2-dimension the number of the vertices of the polytope is at least 3, and for a 3-dimension, number of vertices of the polytope is at least 4.

3.2 Polytopes Under Linear Transformation

A polytope's image is again a polytope under linear transformation. A polytope can be defined in terms of its vertices. Although we say, it contains all the points, by only transforming its vertices, all the points are mapped into the new polytope. For P1=A*P0:

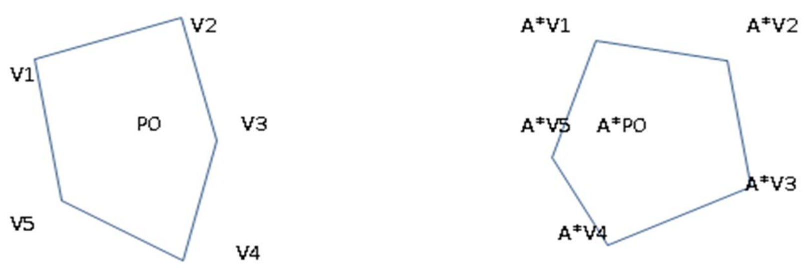


Figure 3.7 Transformation of a polytope for P1=A*P0

P contains all the points inside. By only applying Vnew=A*V, where V is the vertices of polytope, all the points inside P is mapped into P'. This information is valuable. Calculate the new vertices and take the convex hull of them. This is our new polytope.

This works for linear reachability. But real models are nonlinear, so you have to make small time steps and use local linear. A nonlinear system must be linearized if it is intended to be studied. Linearization is done by using the Taylor's formula

$$f(x) \approx f'(x_0) * (x - x_0) \tag{3-4}$$

CHAPTER FOUR

STUDYING REACHABILITY IN ONE DIMENSION

4.1 Finding Critical Points

In order to understand simulation challenges of reachability in higher dimensions, first we must come over them in one dimension. In one dimension also finding the intersection of the polytopes and dividing them into new polytopes are easier, so that we can focus on other issues of reachability by examining it on one dimension.

Lets say we have this set of equations: x' vs. x. As you see this is a fixed partitioned linear system. Between -0.5 and 0.5 it is y=ax, and between 0.5 and 2, it is y=-x+b and between -0.5 and -2 it is y=-x-1. Let's discuss the stability of this system.There are three equations. See Appendix 1 for m-file written in matlab.

$$\frac{dx}{dt} = \begin{cases} x & \text{for } -0.5 \le x \le 0.5 \\ -x+1 & \text{for } 0.5 \le x \le 2 \\ -x-1 & \text{for } -2 \le x \le -0.5 \end{cases} \tag{4-1}$$

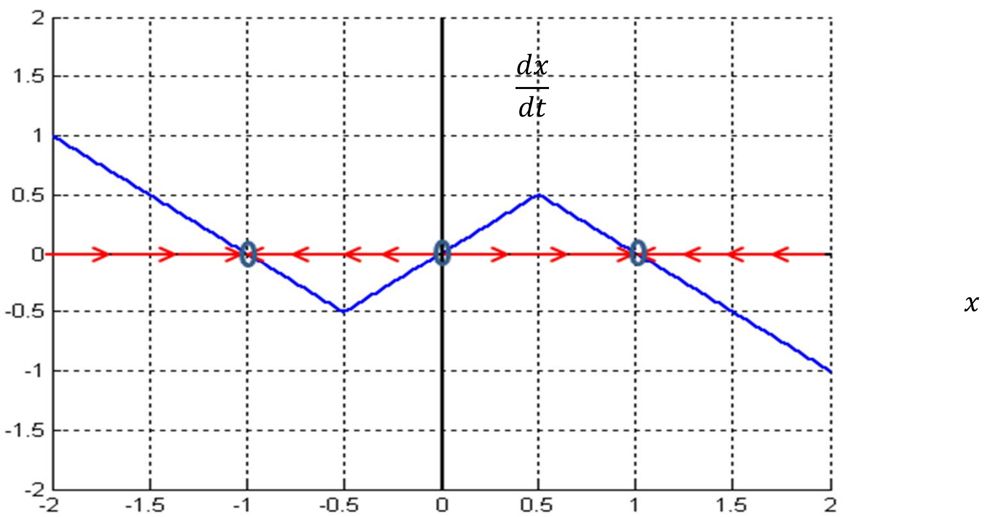


Figure 4.1 A fixed partition linear system modeled by the equation (4-1).

Let's define the critical points of this set of linear differential equations.

$$\frac{dx}{dt} = x = 0 \tag{4-2}$$

X=0 is a critical point and unstable. Since dx/dt=1*x and 1 is greater than zero. The solution is $x = e^{t}$. As t → ∞, x moves away from equilibrium point.

$$\frac{dx}{dt} = -x + 1 = 0 \tag{4-3}$$

X=1 is a critical point and stable. Since dx/dt=-1*x+1=-1 and -1 is smaller than zero. The solution is $x = 1 - e^{-t}$. As t → ∞, x moves through equilibrium point 1.

$$\frac{dx}{dt} = -x - 1 = 0 \tag{4-4}$$

X=-1 is a critical point and stable. Since dx/dt =-x-1 and -1 is smaller than zero. The solution is $x = -1 - e^{-t}$. As t → ∞, x moves through equilibrium point -1.

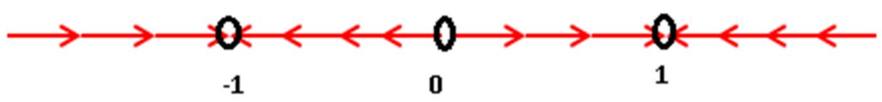


Figure 4.2 direction flows

To study this set of equations in Matlab, let's discretize them:

Region 1:

$$\frac{\mathrm{dx}}{\mathrm{dt}} = \frac{\mathrm{x(k+1) - x(k)}}{\mathrm{T}} = \mathrm{x(k)} \tag{4-5}$$

$$\mathrm{x(k+1) = (1+T) * x(k)} \tag{4-6}$$

Region 2:

$$\frac{\mathrm{dx}}{\mathrm{dt}} = \frac{\mathrm{x(k+1) - x(k)}}{\mathrm{T}} = \mathrm{-x(k) - 1 = 0} \tag{4-7}$$

$$x(k+1) = (1-T) * x(k) - T \tag{4-8}$$

Region 3:

$$\frac{dx}{dt} = \frac{x(k+1)-x(k)}{T} = -x(k) + 1 = 0 \tag{4-9}$$

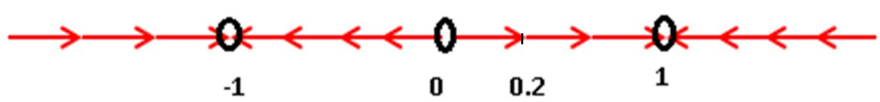
$$x(k+1) = (1-T) * x(k) + T \tag{4-10}$$

So the equation becomes:

$$x(k+1) = \left\{\begin{matrix} (1+T) * x(k) & \text{for} & -0.5 \le x \le 0.5 \\ (1-T) * x(k) + T & \text{for} & 0.5 \le x \le 2 \\ (1-T) * x(k) - T & \text{for} & -2 \le x \le -0.5 \end{matrix}\right\} \tag{4-11}$$

And -0.5 and 0.5 are the breakpoints of the system. These points are also critical for reachability analysis.

Now, let's choose some starting points and see the trajectories: let's choose x(1)=0.2.



The arrows show that, the point 0.2 will go to equilibrium point 1.

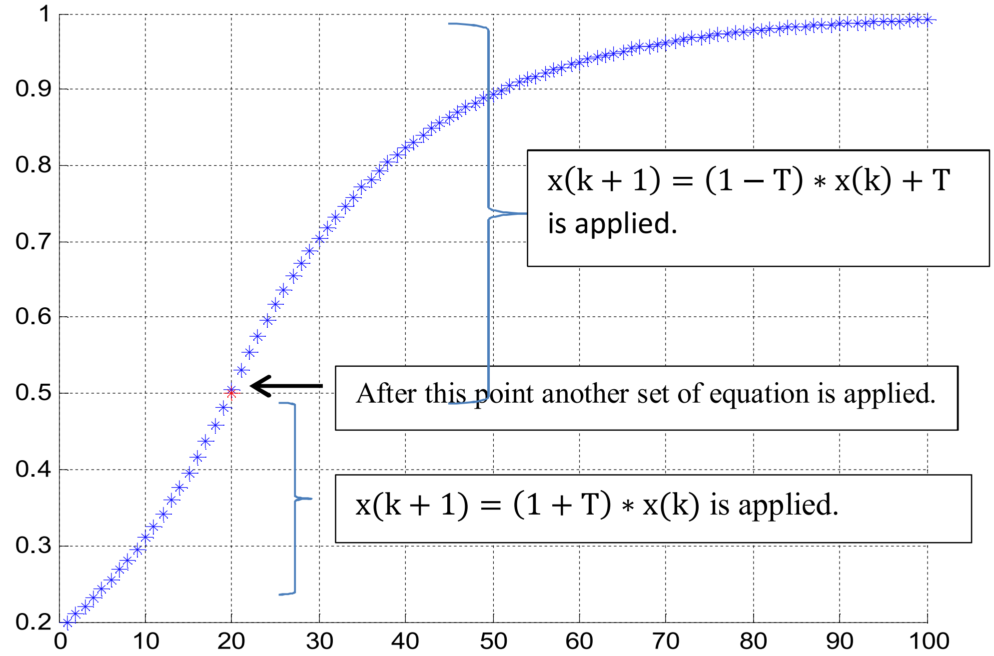


Figure 4.3 Trajectory of point 0.2.

If we start at equilibrium points, nothing will change, since dx/dt=0.

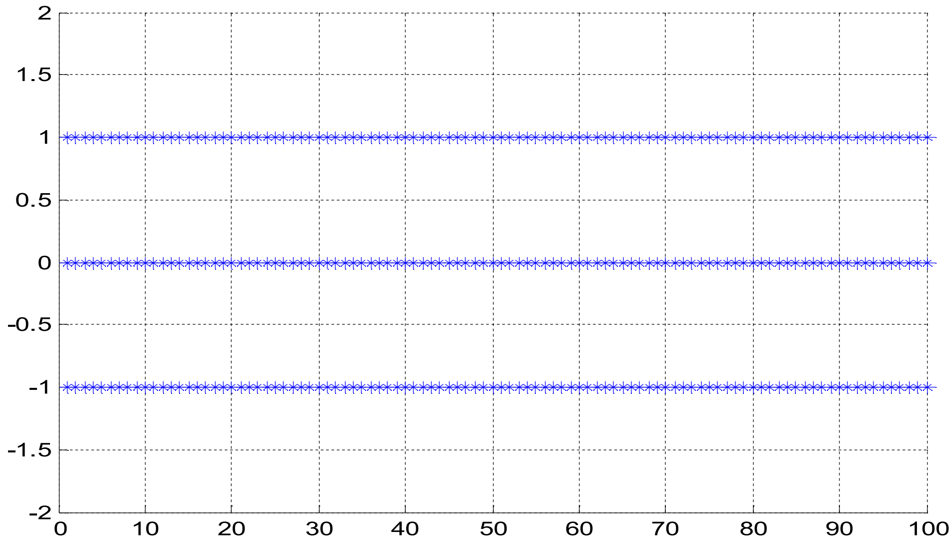


Figure 4.4 Trajectory of points starting exactly at equilibrium points.

Let's choose the points -0.1, 0.1, 1.2 -1.2. Simulating all these 4 points independently, we obtain:

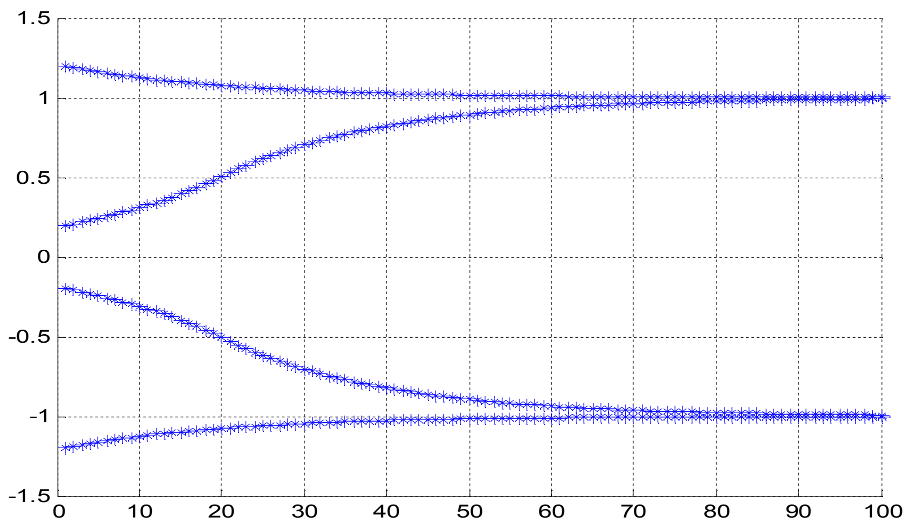
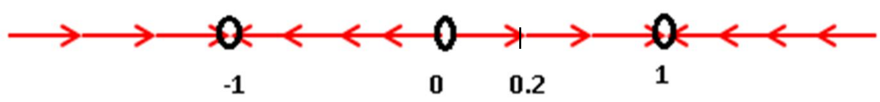


Figure 4.5 Trajectories for different starting points.

4.2 Studying with Polytopes

Because of uncertainty of parameters, we choose a set of states which is a convex polytope. In one dimension, we have to determine 2 vertices. We already saw where a single point will go:



But what about the region between the points 0.1 and 0.2? Without using polytopes, we have to map all the points between 0.1 and 0.2:

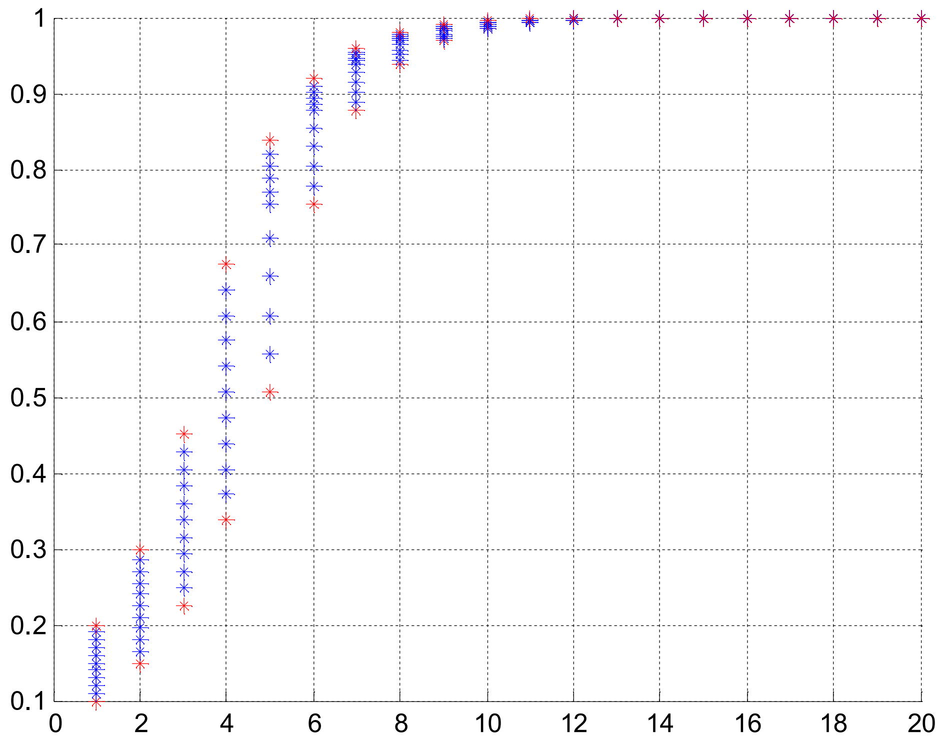


Figure 4.6 The points between 0.1 and 0.2.

There are infinitely many points between 0.1 and 0.2. For this simulation, 0.10, 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19 and 0.20 is taken. But intuitively, we can see that every point between 0.1 and 0.2 will go to the equilibrium point 1. Since the arrows also show that every point between 0^+ and 2 will go to 1.

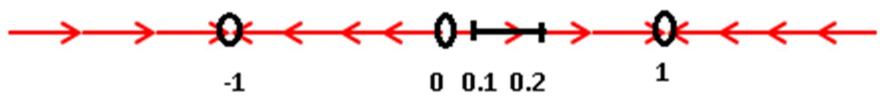


Figure 4.7 The polytope [0.1 0.2]. The region will flow through 1.

Now let’s use polytopes, we define a polytope by its vertices

$$P0 = \begin{bmatrix} 0.8 \\ 0.9 \end{bmatrix} \tag{4-12}$$

In this region, P0 is transformed into a new polytope $P0^1$. $P0^{k+1} = A * P0^k + f$ and A is:

$$P0^{k+1} = \begin{cases} (1+T) \text{ if } P0^k \text{is in region } (0^+, 0.5) \text{ and } (-0.5, 0^-) \\ (1-T) * P0^k \text{ if } P0^k \text{is in region } \qquad (0.5,2) \\ (1-T) * P0^k - T \text{ if } P0^k \text{ is in region } \quad (-2,-0.5) \end{cases} \qquad (4\text{-}13)$$

P0 is in region (0.5, 2). So we will apply: $P0^{k+1} = (1-T) * P0^k + T$. In the figure 4.8 below, the points constituting the above line indicate the first vertices of P0 and the points constituting below line are the second vertices of the P0. The region between these vertices is all mapped into new P0.

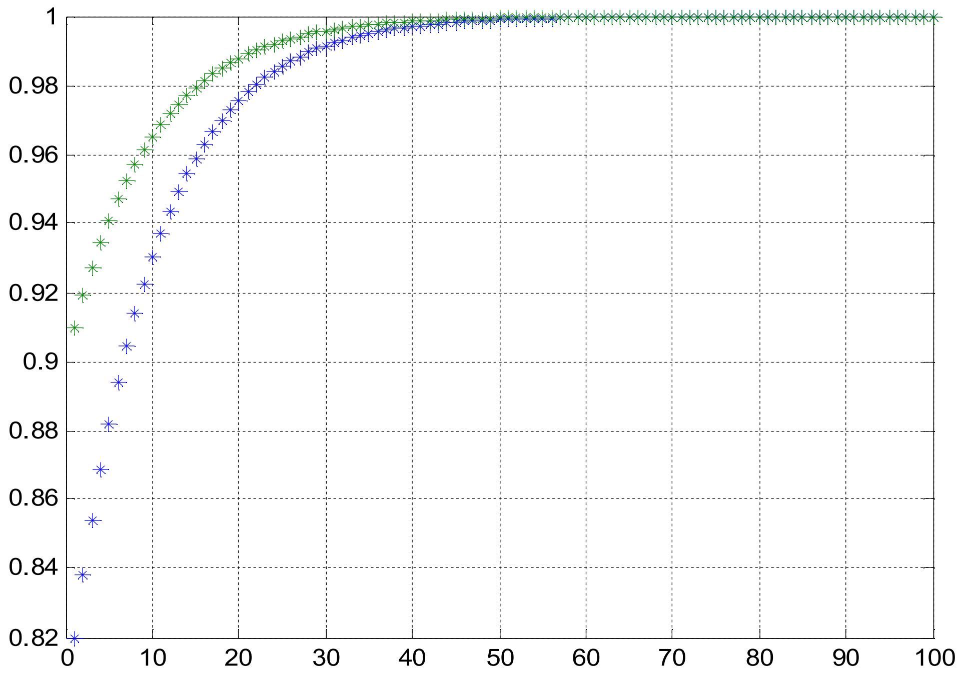


Figure 4.8 Trajectory of polytope [0.8 0.9].

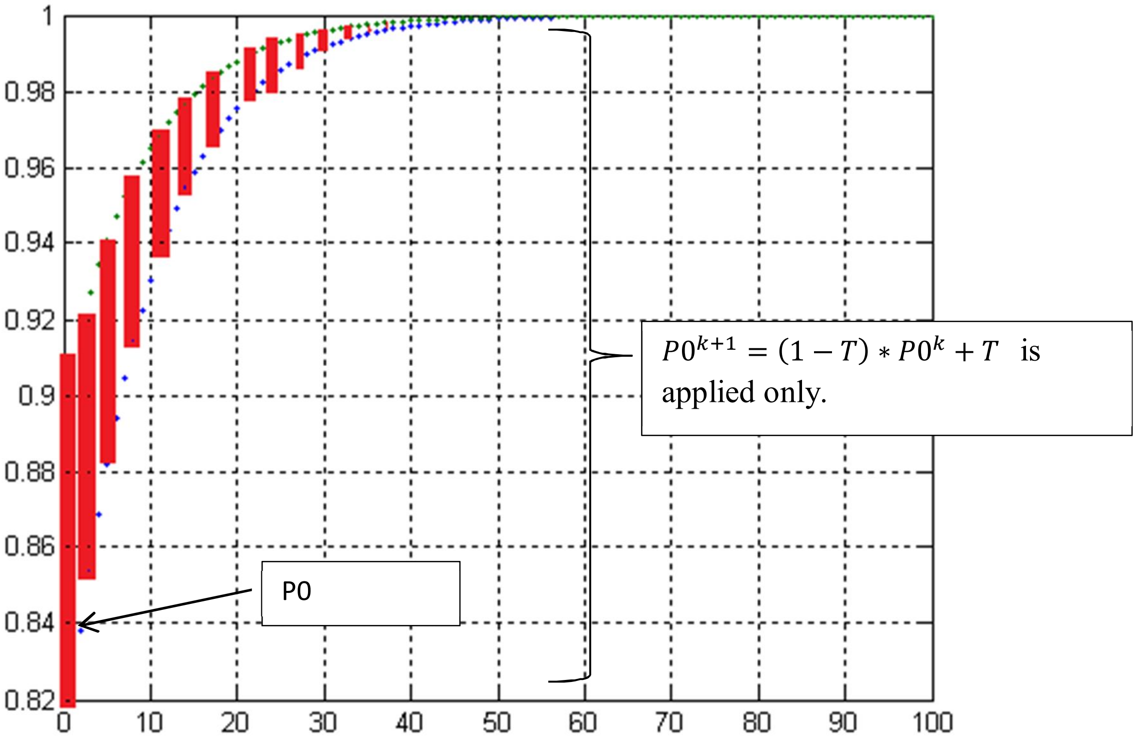


Figure 4.9 Trajectory of polytope region with the equation applied.

For y= ax. The equilibrium point is at ax=0; which x =0. Is this a stable or non-stable point? Since differential is greater than zero, it is nonstable. If you start with zero, you stay at zero. If you start with any smaller number than zero, you go away from the equilibrium point.

4.3 Splitting Polytopes around Breakpoints

In previous example, polytope always stood in the region (0.5, 2). What will happen if the polytope crosses a boundary?

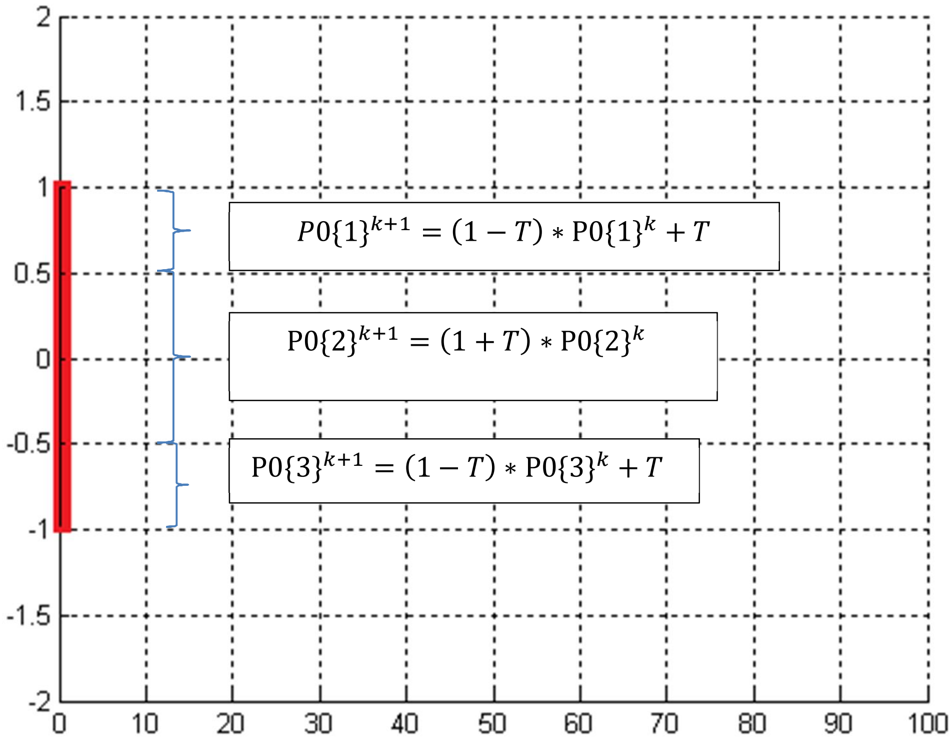


Figure 4.10 Polytope crossing breakpoints 0.5 and -0.5.

We have to divide the polytope into 3 new polytopes: P0{1},P0{2},P0{3}. Now 3 polytopes will be simulated independently. But it is still not enough. Because P0{2} contains the point 0 which is an unstable equilibrium point. What to do when a polytope contains an unstable equilibrium point.

Let's start with a polytope, P0=[-0.2 0.2]. In this region we have to apply $P0^{k+1} = (1 + T) * P0^{k}$. Take the vertices of the polytope and apply the A*V1 and A*V2 where V1 and V2 are the vertices of the polytope.

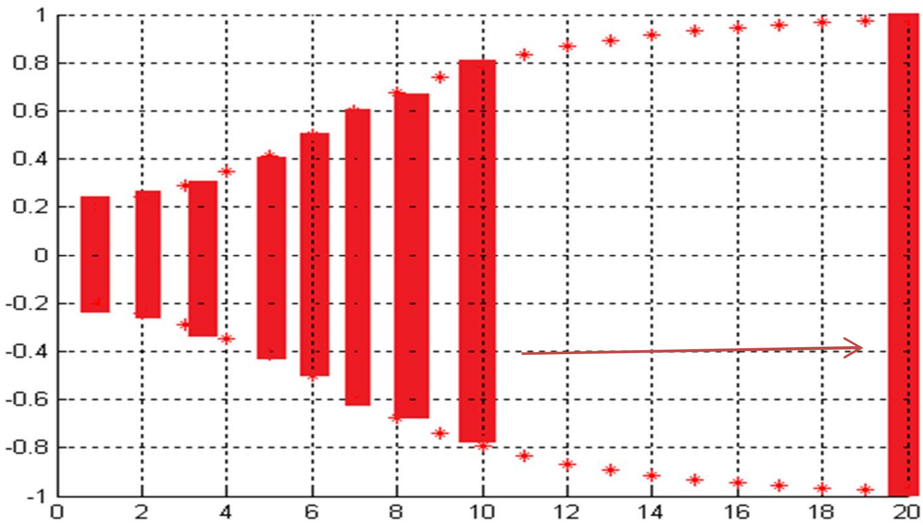


Figure 4.11 The polytope with the vertices -0.2 and 0.2, containing unstable equilibrium point

We started with the vertices of polytope, and take the convex hull of the vertices. Figure 4.11 tells us that, the region between -0.2 and 0.2 will go to -1 and 1 in 20 step. But it is wrong. The region between -0.2 and 0.2 will either go to -1 or 1. Not to the region between -1 and 1. The correct simulation would be.

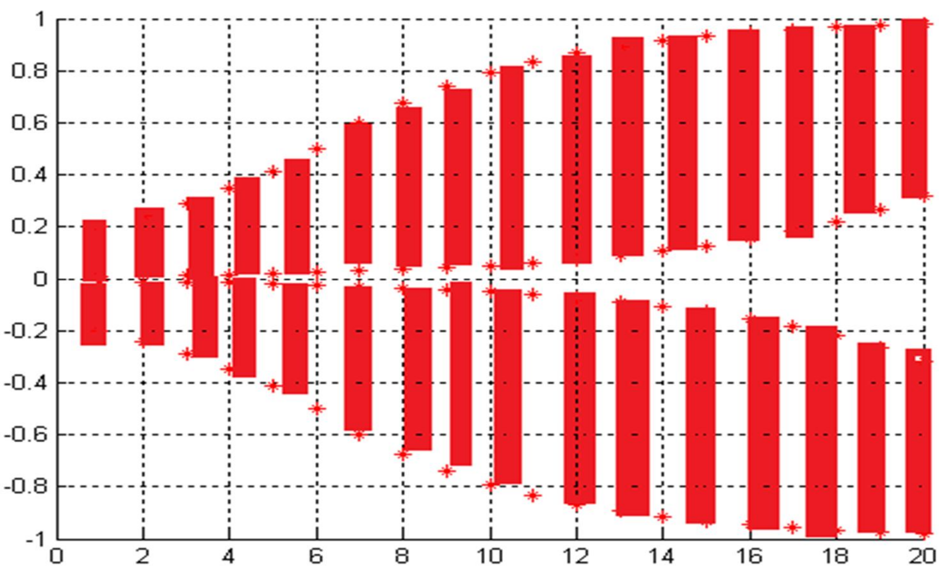


Figure 4.12 The polytope containing unstable equilibrium point is divided

So, splitting the polytope around unstable equilibrium point would give the correct trajectory. This was for the case of unstable equilibrium point. Now let's discuss the breakpoints.

4.4 Challenges of Splitting Around Breakpoints

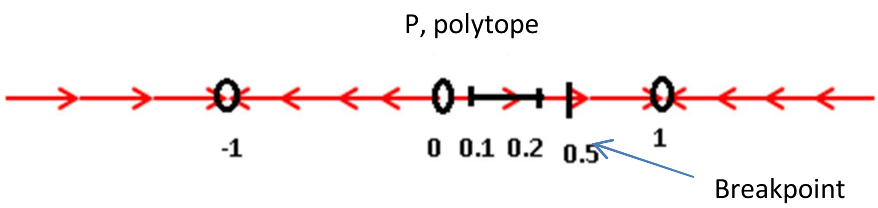


Figure 4.13 A polytope with the vertices 0.1 and 0.2.

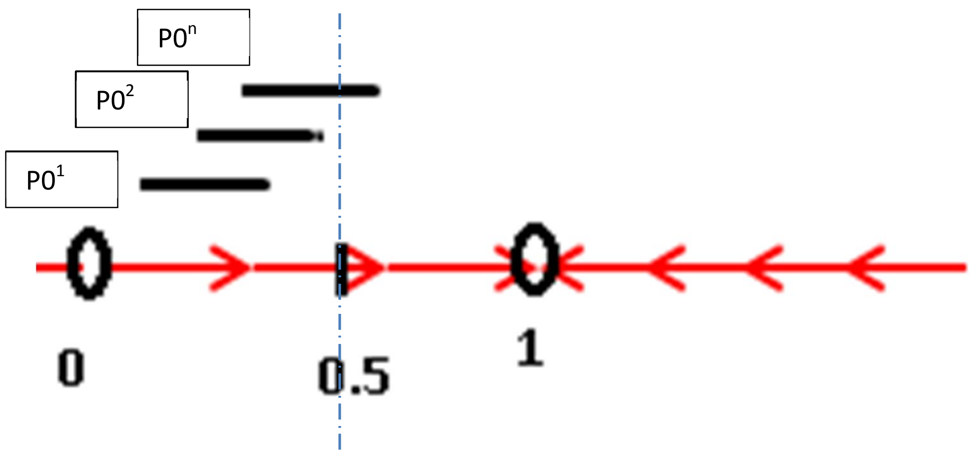


Figure 4.14 Polytope crossing breakpoint 0.5 after n steps.

For figure 4.15, the polytope crosses the breakpoint at 3rd step. Now, the region that falls into the (0.5,2) and (0,0.5) will have different set of equations. So we have to split the polytope around 0.5. P03 is splitted into 2 polytopes $P0^3_1$ and $P0^3_2$. Now, 2 polytopes will be simulated differently at each step.

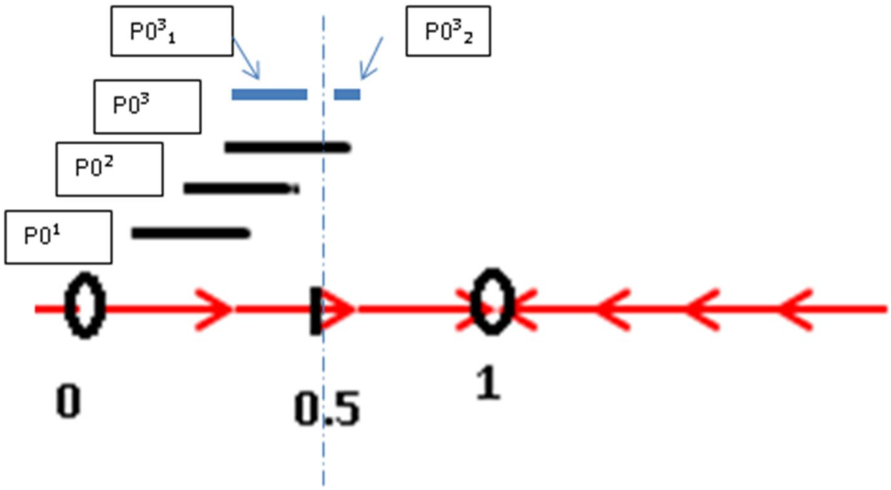


Figure 4.15 P0 is splitted into 2 new polytopes around 0.5 which is a breakpoint.

$P0^3_1$ polytope is again splitted around breakpoint after one more step. After this step there will be 3 polytopes. And one more step, there will be 4 polytopes as shown figure 4.16 below.

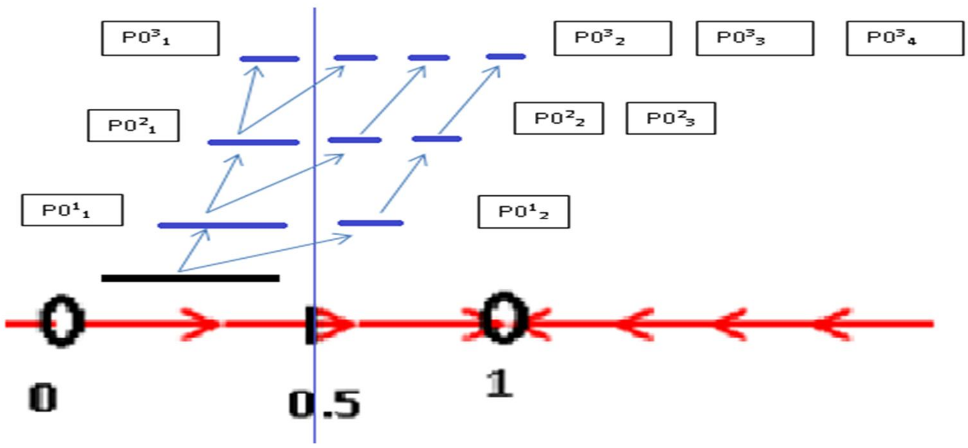


Figure 4.16 Polytope splitting around breakpoints.

After some time, the polytope will completely cross the breakpoint, after that the number of polytopes will stay constant.

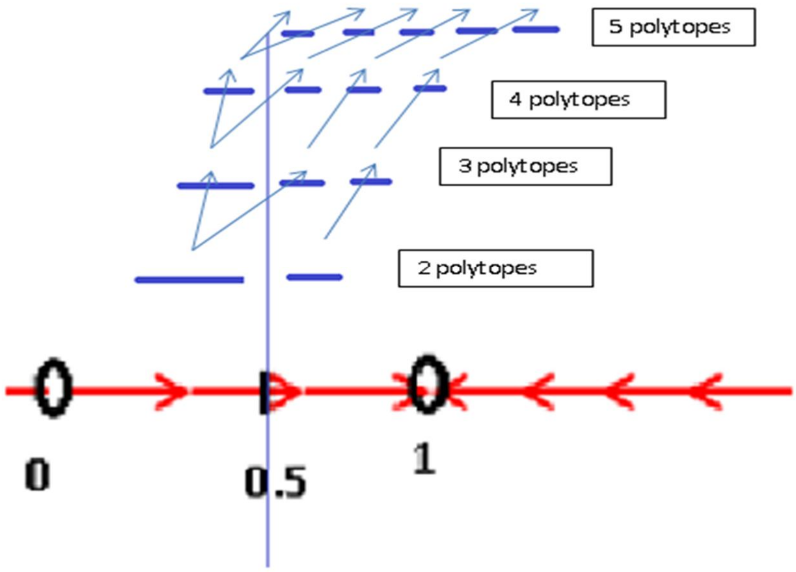


Figure 4.17 After 4 step, number of polytopes will stay constant.

4.5 Step Size

We have to determine the step size of linearization and resolution of breakpoints. The following figure 4.18 show what will happen if we don't do it accordingly. Let's say our vertices of polytope will go according to the equation x(k+1)=(1-a*T)*x(k)+b, where T is our step size. If T is relatively small:

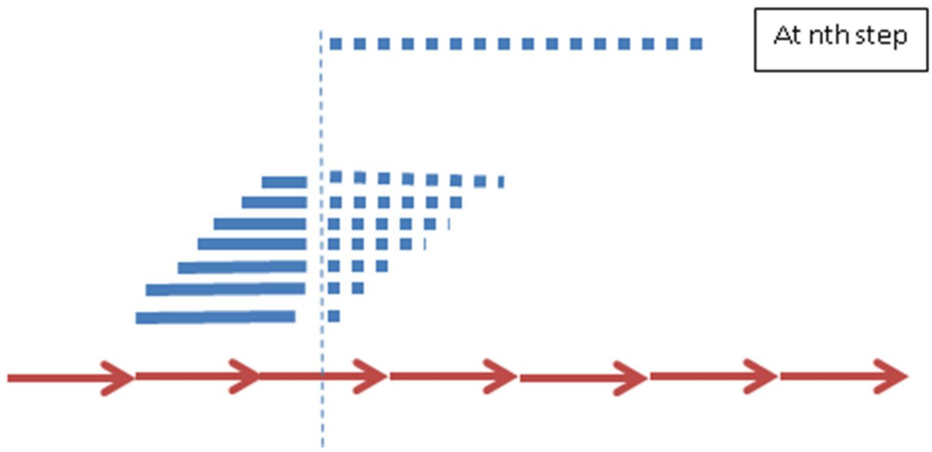


Figure 4.18 The polytope crossing breakpoint with relatively small T.

If T is relatively large:

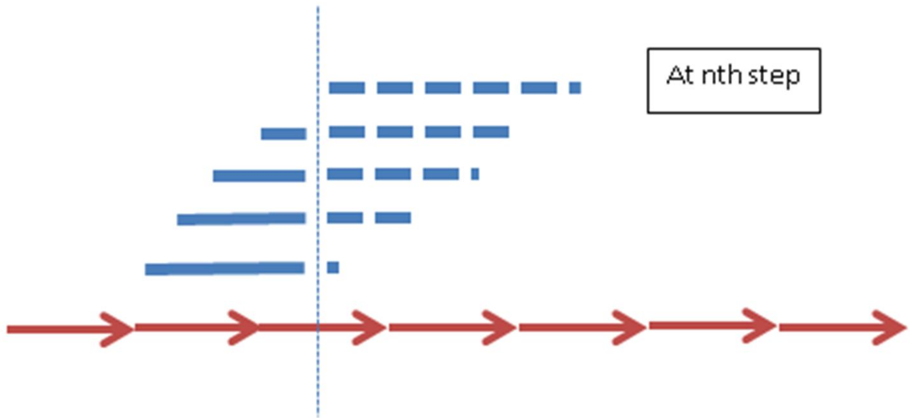


Figure 4.19 The number of polytopes decreased according to our T.

So as it can be seen in figure 4.19, if T is large, the polytopes splitted around breakpoints are less than the polytopes with small T. And also T can be chosen so large that in one step, it can cross the breakpoint without splitting as in figure 4.20.

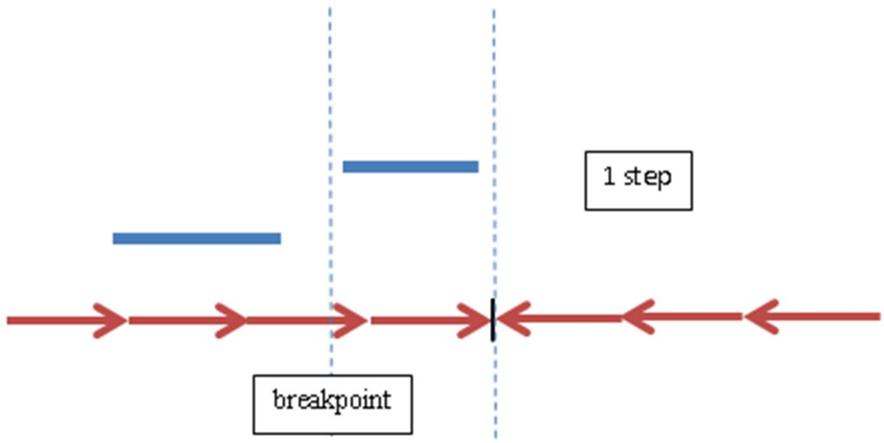


Figure 4.20 Polytope can jump over breakpoint in one step with appropriate T.

But you should be careful here, if T is very large, so that both of the vertices crosses the attractor. A scenario would be like in figure 4.21:

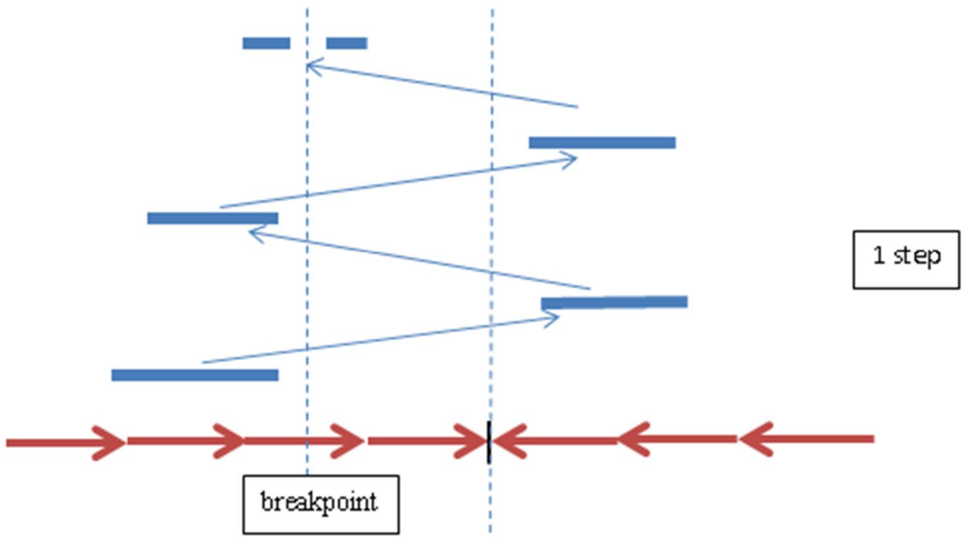


Figure 4.21 A scenario that would happen in case of T chosen too large inappropriately.

4.6 Resolution of Breakpoints

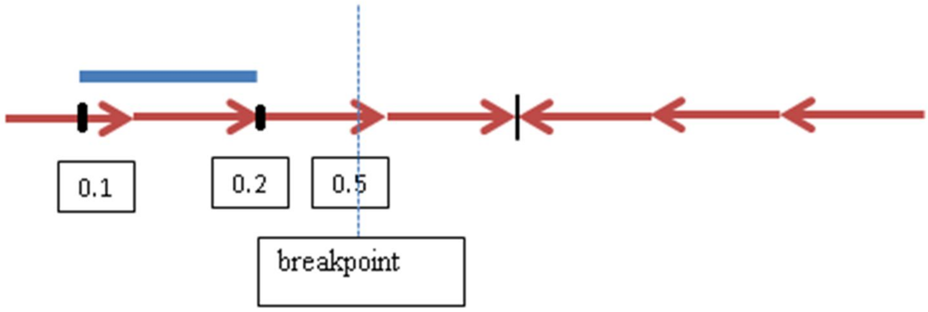


Figure 4.22 One dimensional polytope with vertices [0.1 0.2] and breakpoint at 0.5.

Look at the figure 4.22, P0=$\begin{bmatrix}0.1\\0.2\end{bmatrix}$; after n step P0^n has to be splitted around 0.5 and P0^n =$\begin{bmatrix}0.45\\0.52\end{bmatrix}$. The polytope has to be splitted around 0.5. P0$^n_1$=$\begin{bmatrix}0.45\\0.5^-\end{bmatrix}$ and P0$^n_2$=$\begin{bmatrix}0.5^+\\0.52\end{bmatrix}$; where 0.5^+ is the closest point to 0.5 from right. And 0.5^- is the closest point to 0.5 from left. We call it as the resolution of the breakpoint.

The resolution of the breakpoint has to be smaller than the one step displacement of the polytope. Otherwise the polytope will disappear completely as in figure 4.23.

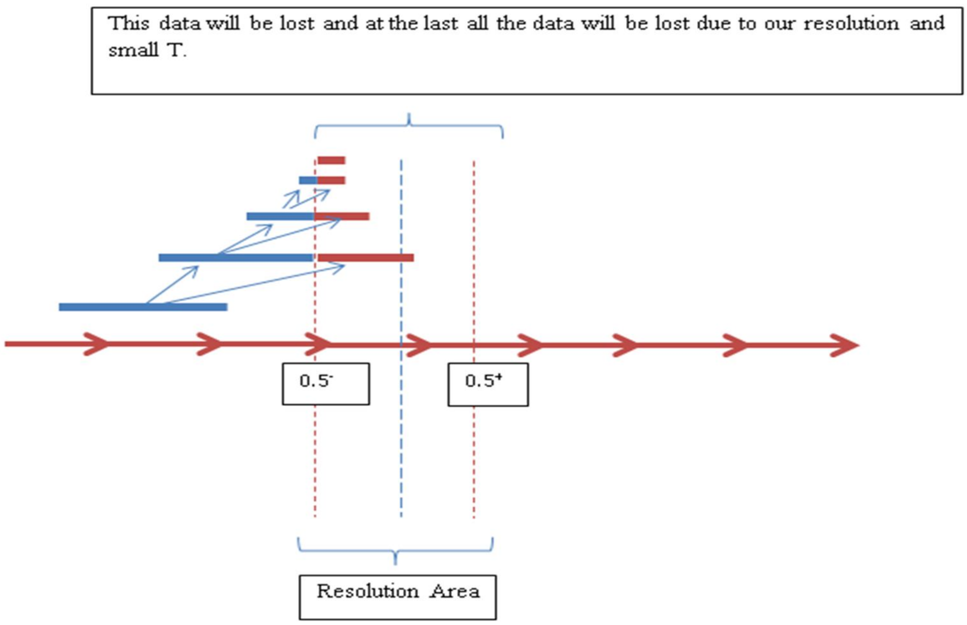


Figure 4.23 Breakpoint resolution area

Even if T is set accordingly, in resolution area data will be lost, but this is a small portion of data compared to whole. This is illustrated in figure 4.24.

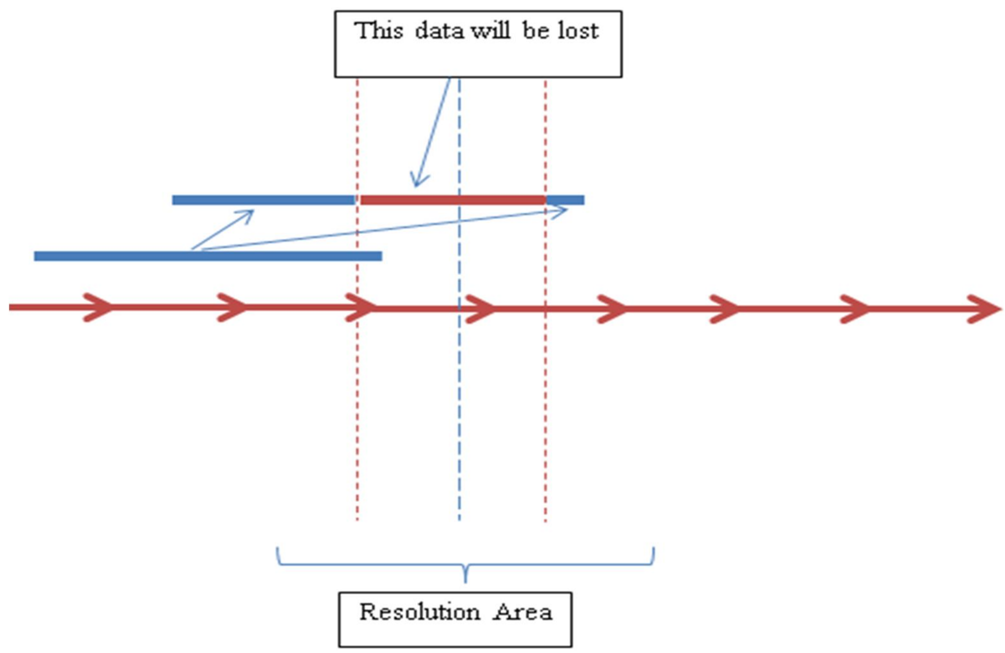


Figure 4.24 Data is lost in resolution area.

Another scenario would be like that if polytope is relatively small as in figure 4.25:

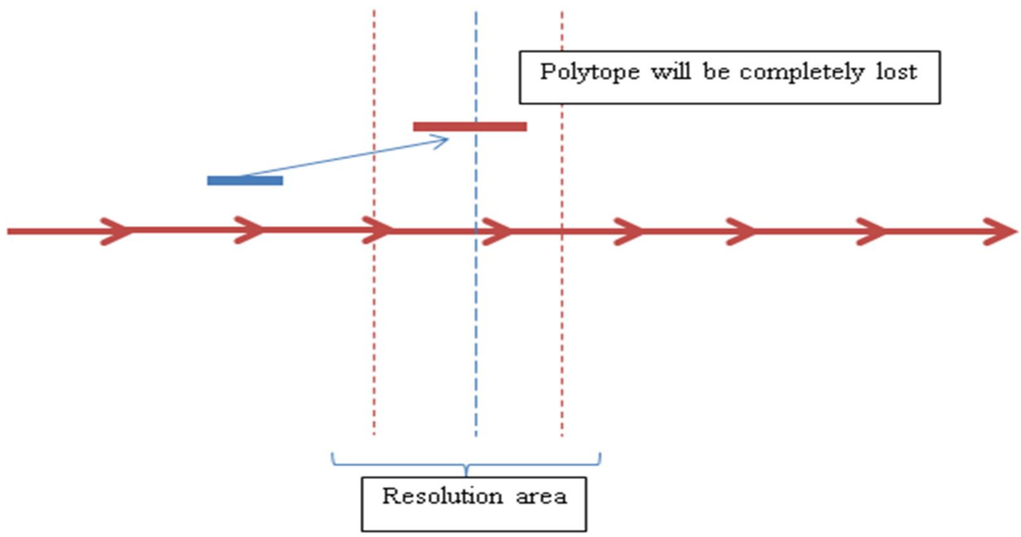


Figure 4.25 Polytope will be completely lost in resolution area.

The solution of resolution problem is that: T step size must be relatively large and polytopes that are smaller than resolution area must be carefully detected and resolution may be changed accordingly.

4.7 Discussion over Splitting Polytopes Around Breakpoints

If $P1 \subset P2$ you don't have to examine the P1 in one dimension just do it for P2. If $1 \cap P2 \neq \emptyset$, then take the convex hull of the polytopes and go on with the new polytope $P3 = P1 \cup P2$.

4.8 Studying a Piece-wise Linear Model

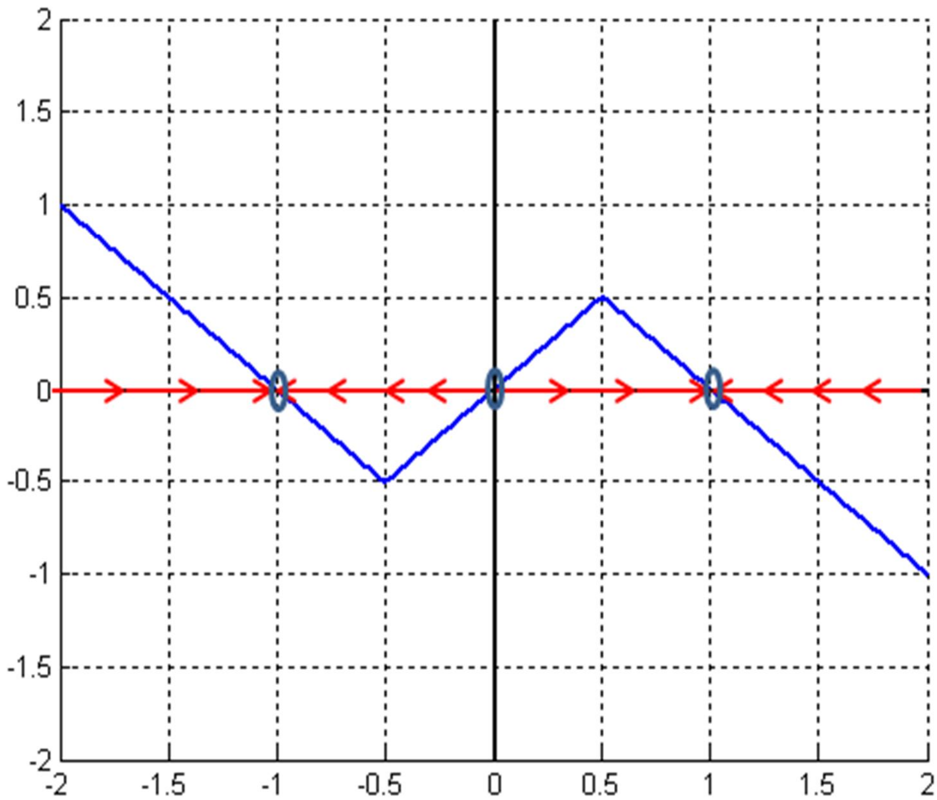


Figure 4.26 Piece-wise linear model to be studied.

Figure 4.26 shows the piece-wise linear model that we will study. Now we will apply the techniques we have developed for one dimensional reachability analysis after writing a suitable program in Matlab and improve it. These simulations are all done in Matlab.

Starting with $P0 = \begin{bmatrix} 0.8 \\ 0.9 \end{bmatrix}$ and plotting the trajectory we have:

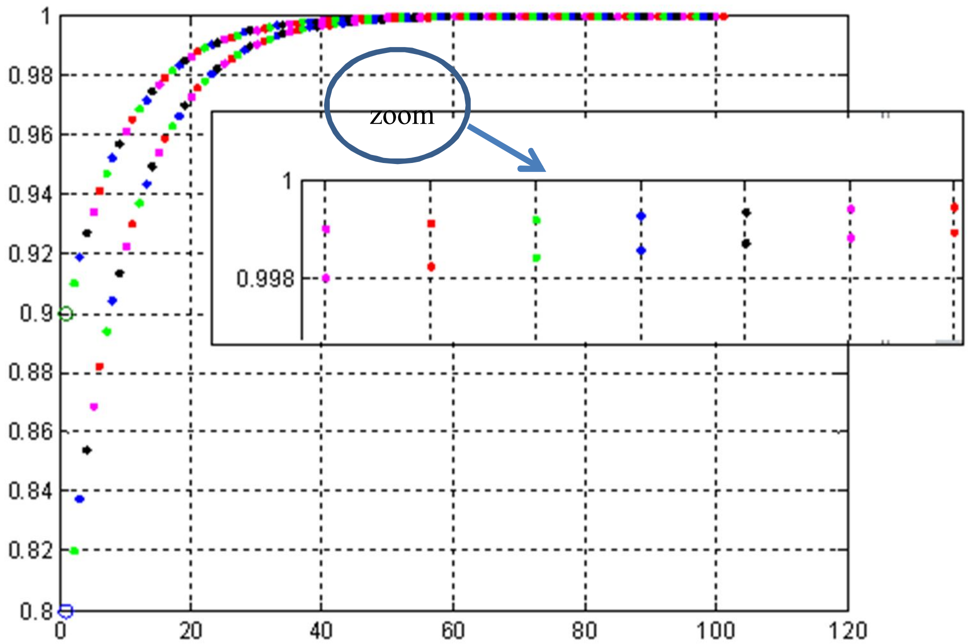


Figure 4.27 Polytope starting with vertices 0.8 and 0.9

In figure 4.28, another polytope $P0 = \begin{bmatrix} 0.1 \\ 0.2 \end{bmatrix}$ will move away from 0, unstable equilibrium point and after some time, it will cross the 0.5 breakpoint. Now the polytope will be splitted as shown in figure 4.29:

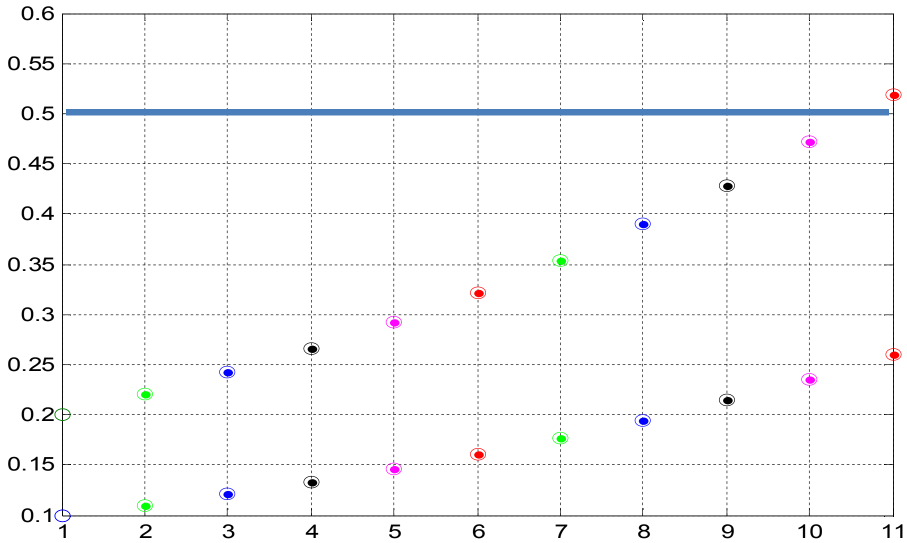


Figure 4.28 Polytope crossing 0.5 breakpoint.

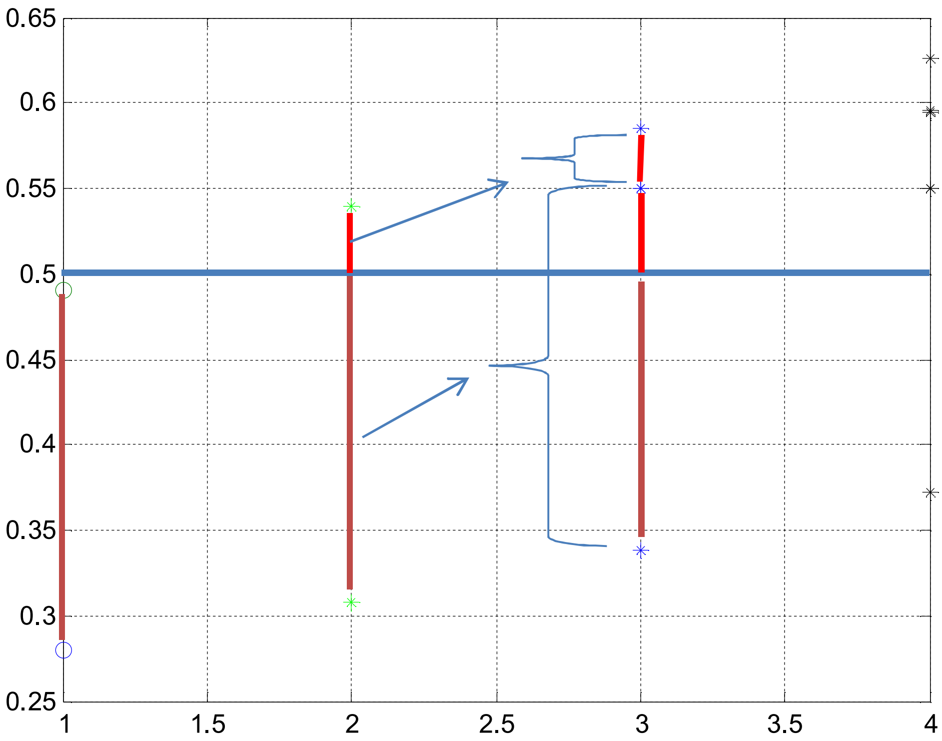


Figure 4.29 Simulating a polytope crossing 0.5 breakpoint

Look at the figure 4.30, If we start with the polytope $P0 = \begin{bmatrix}-0.1\\-0.2\end{bmatrix}$;

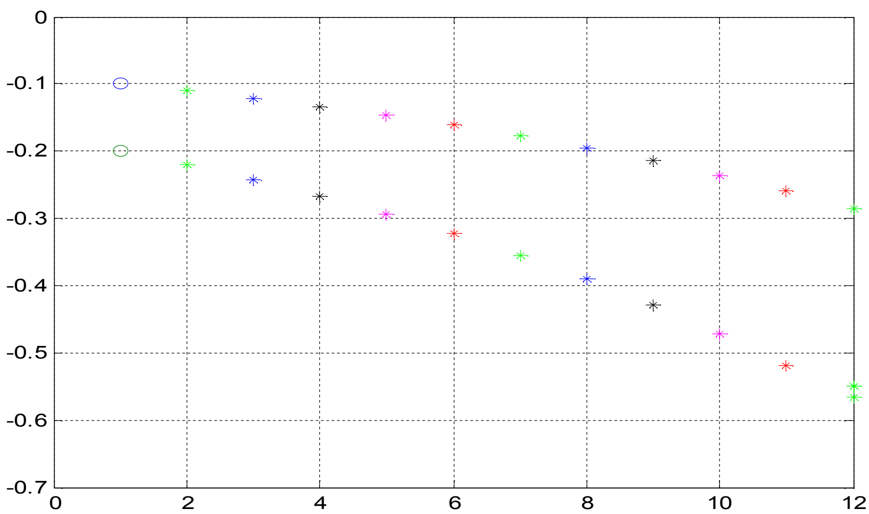


Figure 4.30 Polytope with the vertices of -0.1 and -0.2.

In case, $P0 = \begin{bmatrix} -0.8 \\ -0.9 \end{bmatrix}$, figure 4.31 shows the simulation.

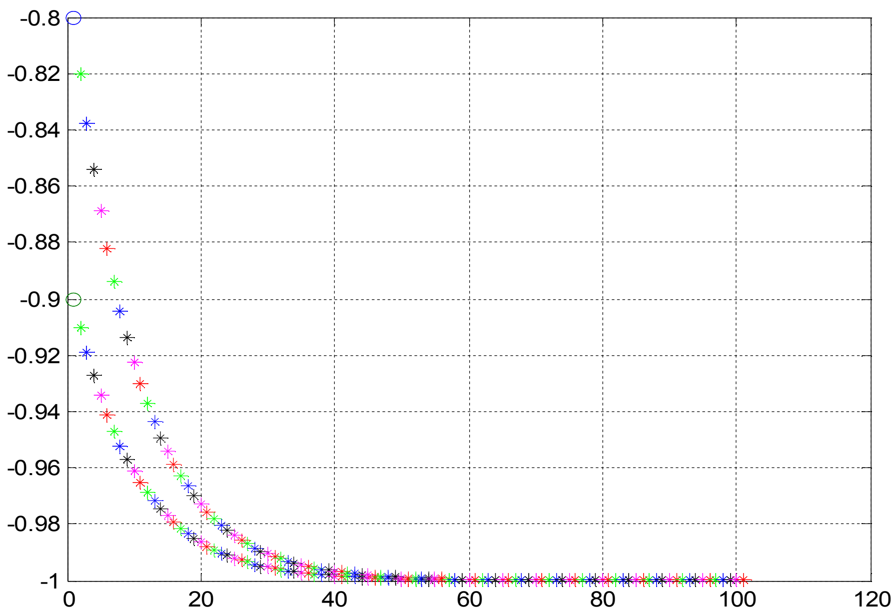


Figure 4.31 Simulation of polytope of [-0.8 -0.9]. Polytope converges to point -1.

In case, $P0 = \begin{bmatrix} -1.5 \\ -1.2 \end{bmatrix}$, figure 4.32 shows the simulation:

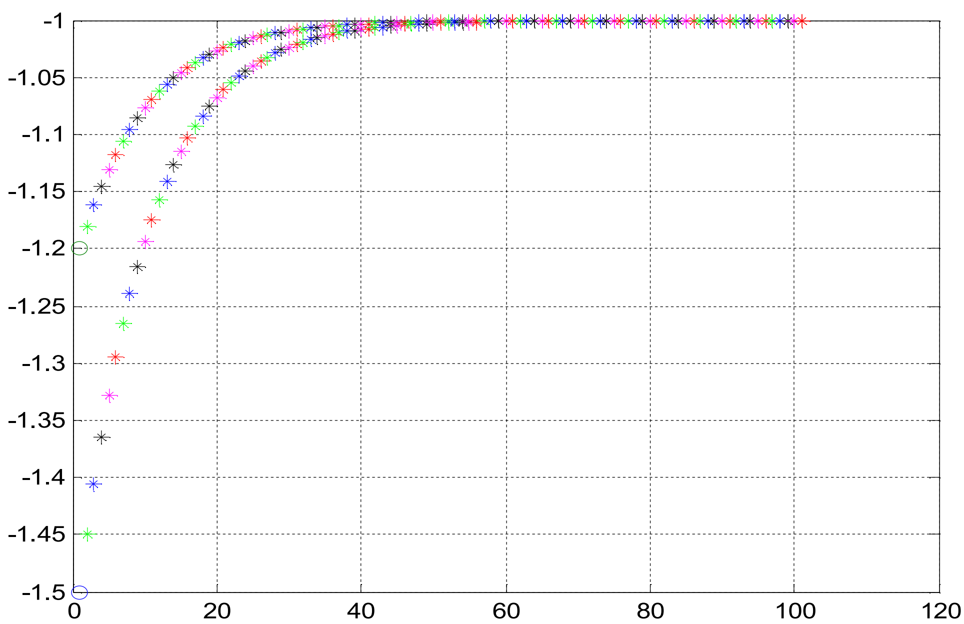


Figure 4.32 Simulation of polytope of [-1.5 -1.2]. Polytope converges to -1.

In case, $P0 = \begin{bmatrix} 1.5 \\ 1.2 \end{bmatrix}$, figure 4.33 shows the simulation:

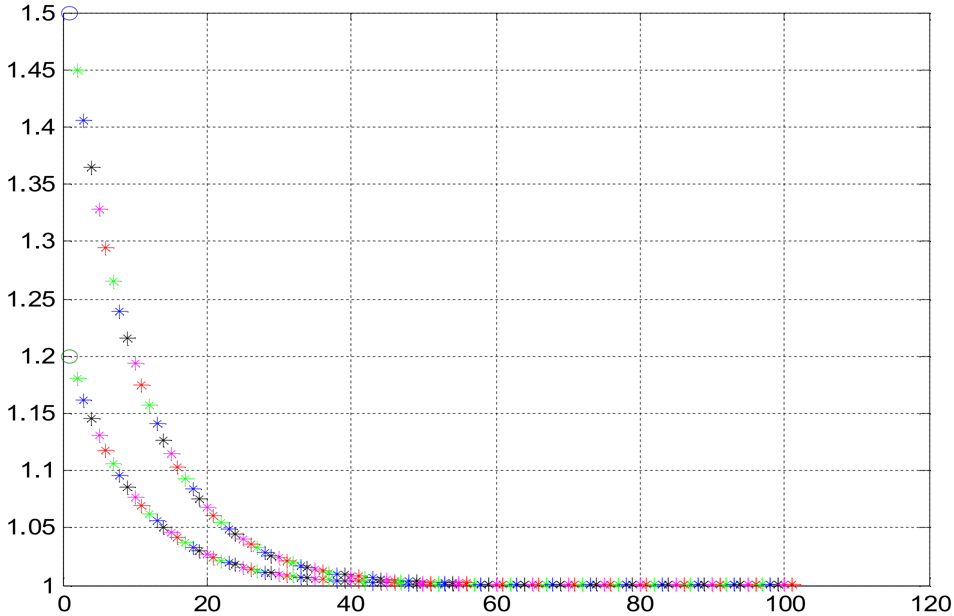


Figure 4.33 Simulation of polytope of [1.5 1.2]. Polytope converges to 1.

4.8.1 Breakpoint Resolution and T Step Size Study of the Model

We already saw that for different step sizes, number of polytopes splitted will be different. Even, if you choose resolution of breakpoint low, and step size relatively large polytope will disappear as in figure 4.34. And now, let's simulate it for different step sizes. Table 4.1 shows the number of polytopes after splitting for different step sizes.

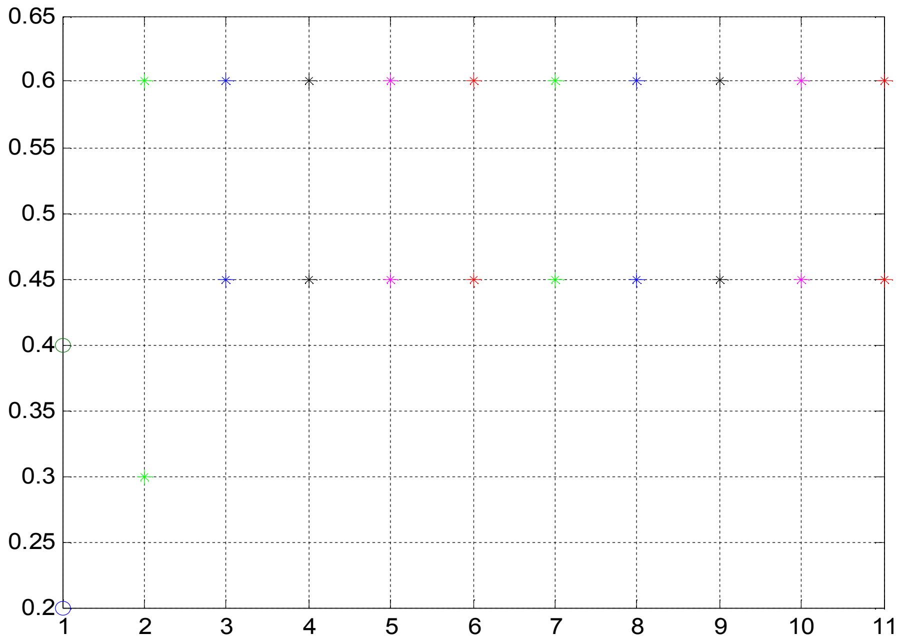


Figure 4.34 Choosing T relatively large and resolution low:The polytope data is lost and only the resolution area is left. Res = 0.1 and T =0.5

Table 4.1 Number of polytopes occuring for different step sizes

T, step size	Number of polytopes
0.01	21
0.05	5
0.1	3
0.5	1

For step size T=0.5, only one polytope completes the simulation. The system again went to its stable point -1, this time with only 20 steps, without splitting. In figure 4.36, you can see that polytope jumps over -0.5 breakpoint in one step. Looking at figure 4.35, you can see the polytopes splitted at breakpoint -0.5. they cover an area, in every step every splitted polytope needs to be run, so this makes the computation hard. Choosing a good step size is crucial.

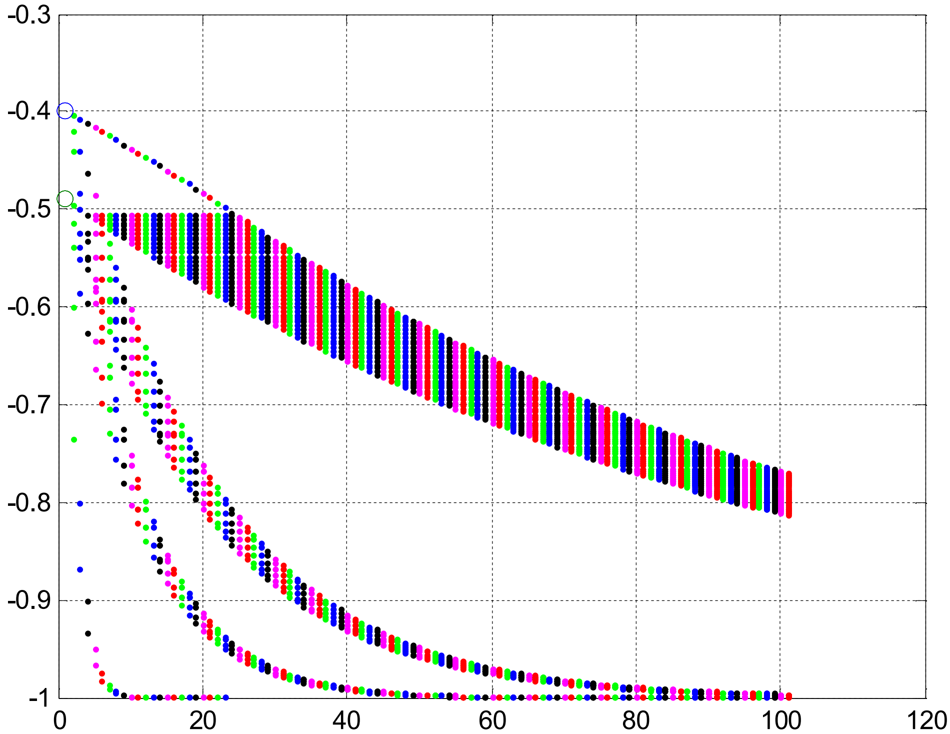


Figure 4.35 Splitting of a polytope around breakpoint for different step sizes, T=0.01, 0.05, 0.1 and 0.5.

The more the splitting, the harder the simulation is. Also simulation can be impossible if the splitting is so much. Resolution area does not affect the splitting; it may lose the data instead of splitting. See appendix 2 for m-file written in Matlab.

AS you see in figure 4.36 below, in one step, polytope completely crosses the breakpoint -0.5. For fixed partition linear systems, in one dimension, PWA systems and PWL systems, once a polytope crosses a breakpoint, you don't need to split the polytope just go on with the crossing one if polytope step size and breakpoint resolution is set accordingly and polytope crosses one breakpoint at most at one step. One dimension is easy and we can see that for every breakpoint combination in piece-wise affine systems, you can only simulate the polytope that crosses the breakpoint. Others will follow the leading polytope. This is simulated below:

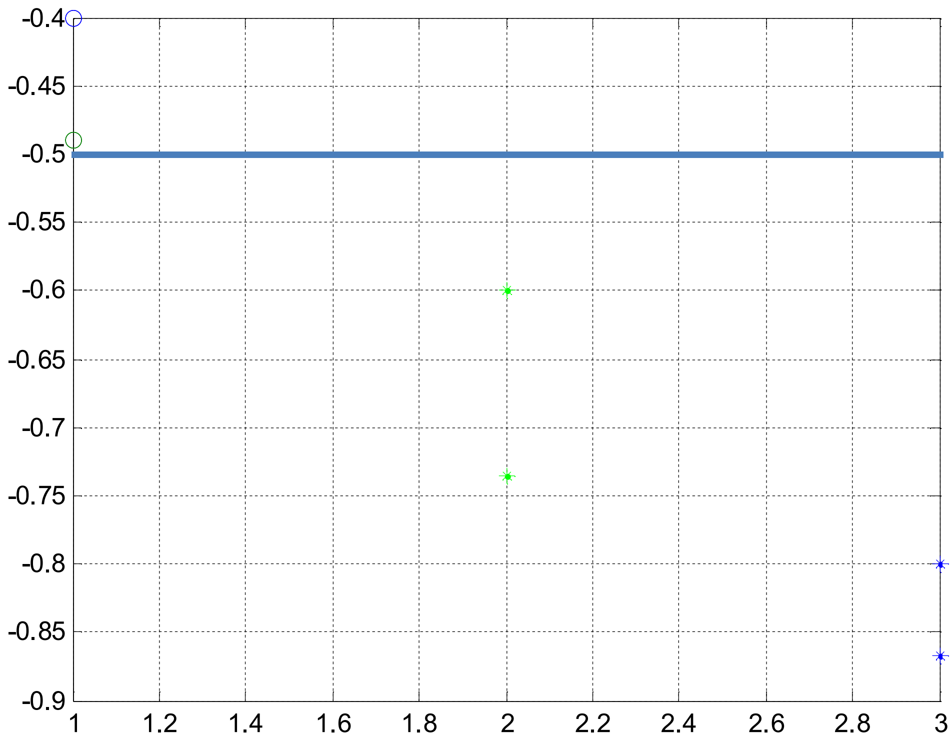


Figure 4.36 For T=0.5, polytope jumps over the breakpoint in one step.

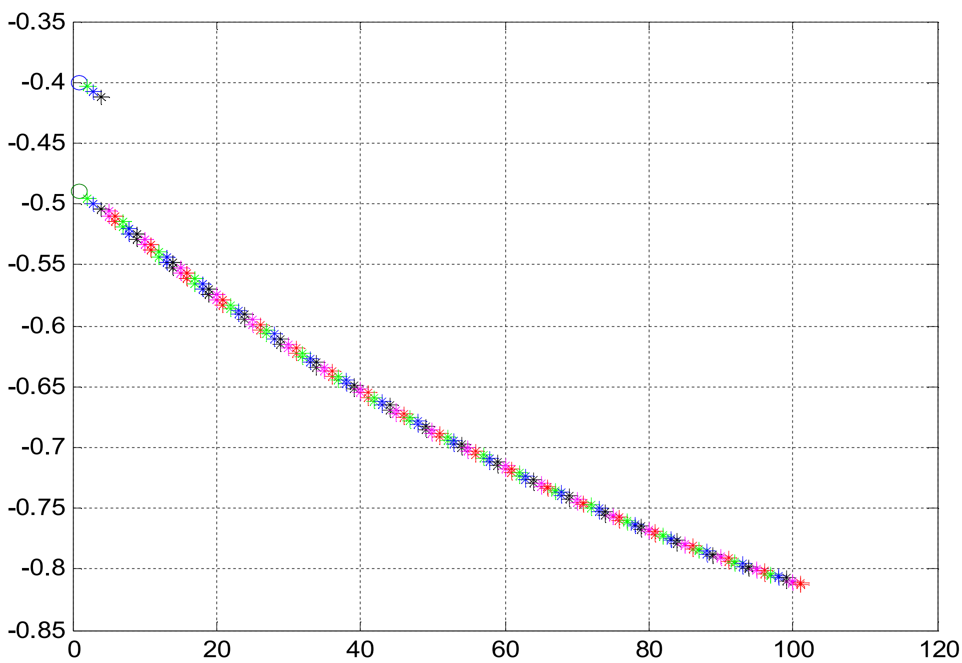


Figure 4.37 Polytope splitted once.

Zoom:

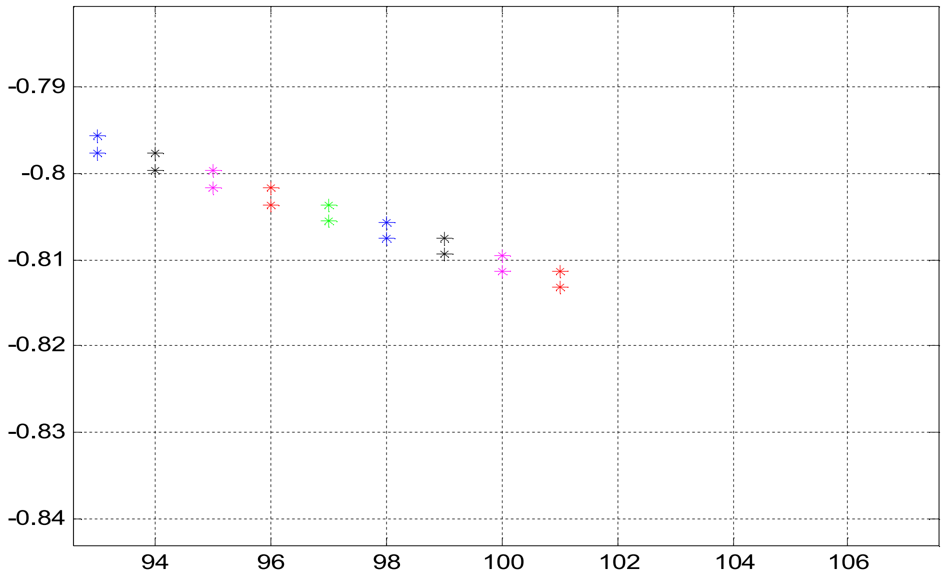


Figure 4.38 The splitted polytope in figure 4.37 zoomed.

4.8.2 Discussion of Breakpoints

A polytope in one dimension crossing a breakpoint can be shown as in figure 4.39 below. After splitting two polytopes must be simulated independently. The polytope on the left that has not been crossed the breakpoint will cross the breakpoint in next step. So, this splitting will occur repeatedly until the polytope is crosses other side completely. There are lots of polytopes now, this is hard to compute. But we can find some effective methods to reduce the number of polytopes to be simulated. Some of them can be ignored, because there are some equivalent polytopes as shown in figure 4.40.

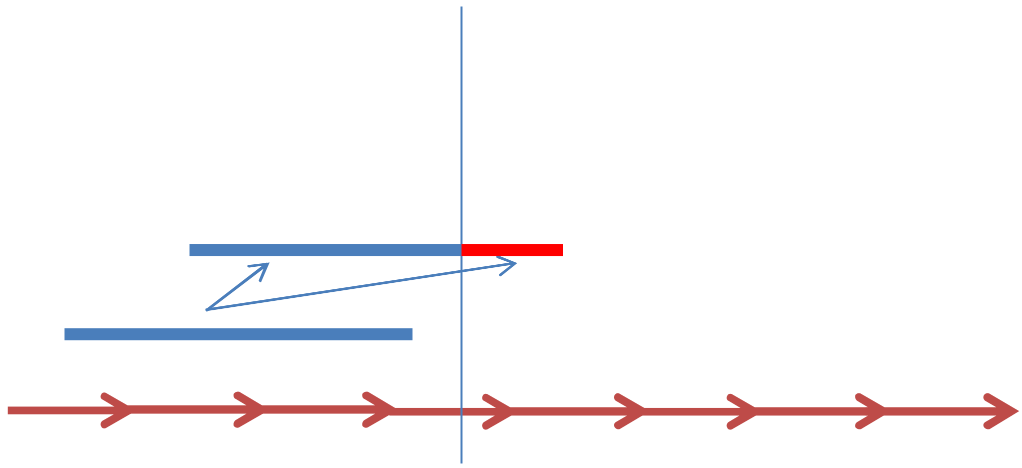


Figure 4.39 A polytope crossing a breakpoint.

The polytopes P1, P2, P3 and P4 shown in figure 4.40, all fall into the same region. We are already computing the trajectory of this area. So we don't need to calculate all of them, just one of them is enough; it is the one that is first splitted, that is P1.

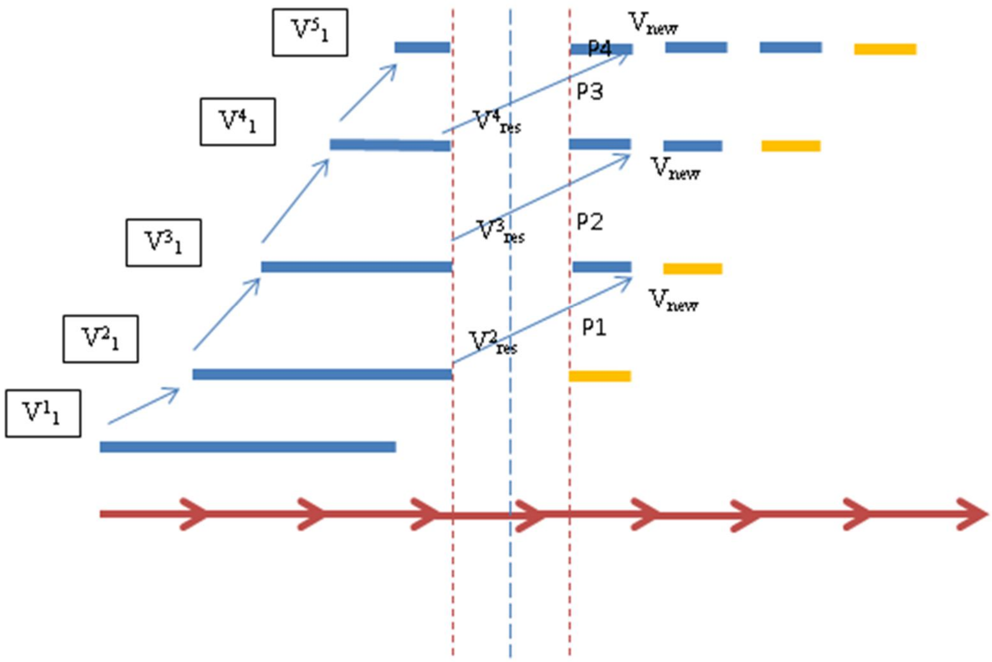


Figure 4.40 Polytopes splitted around breakpoints all fall into the same region.

So as you see, the vertex of the polytope, that is splitted at the resolution area produces the same polytope every time. You don't need to calculate it again and again. P1 =P2 = P3 =P4. They will already go on the same trajectory with P1. So you have to check the equality of these polytopes to reduce the number of polytopes to ease the computation.

CHAPTER FIVE

REACHABILITY OF TMG IN LAC OPERON ONE DIMENSIONAL CASE FOR DYNAMIC PARTITION

5.1 Model of TMG

Below is a nonlinear differential equation that belongs to lac operon.

$$\frac{dT}{dt} = P * \frac{1+K1*T^2}{K+K1*T^2} - T \qquad (5\text{-}1)$$

K=167.1, K1=0.02 $(\mu M)^{-2}$ and p is a constant whose value may change between 0 and 500. For P=250 you can see dT/dt vs. T in figure 5.1;

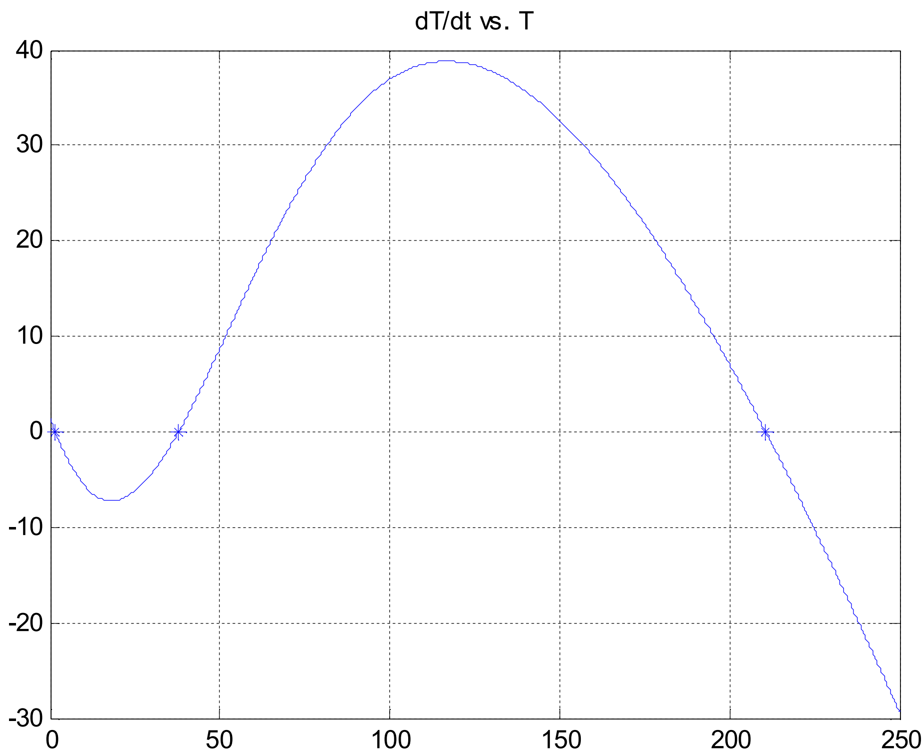


Figure 5.1 The equation (5-1), dT/dt vs T.

Critical points are marked with (*). Critical points occur at the points 1.5704, 37.7888, 210.6408. See appendix 3 for m-file written in Matlab.

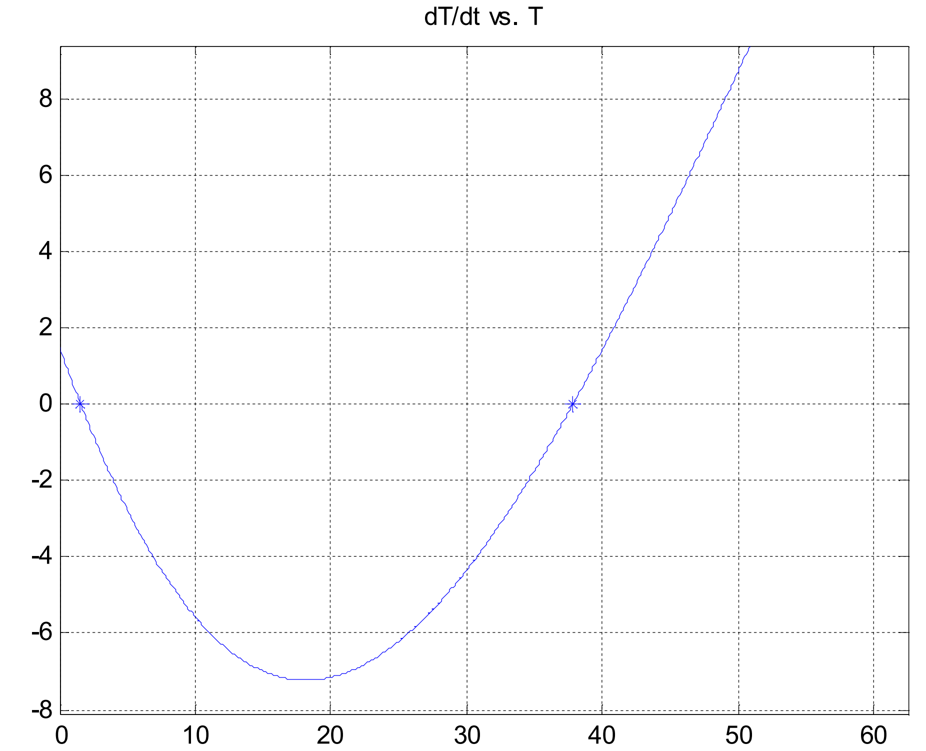


Figure 5.2 Zoom of figure 5.1 around 0.

P is crucial here, for different p, equation has different number of critical points. P is chosen so that there are 3 equilibrium points to model lac operon.

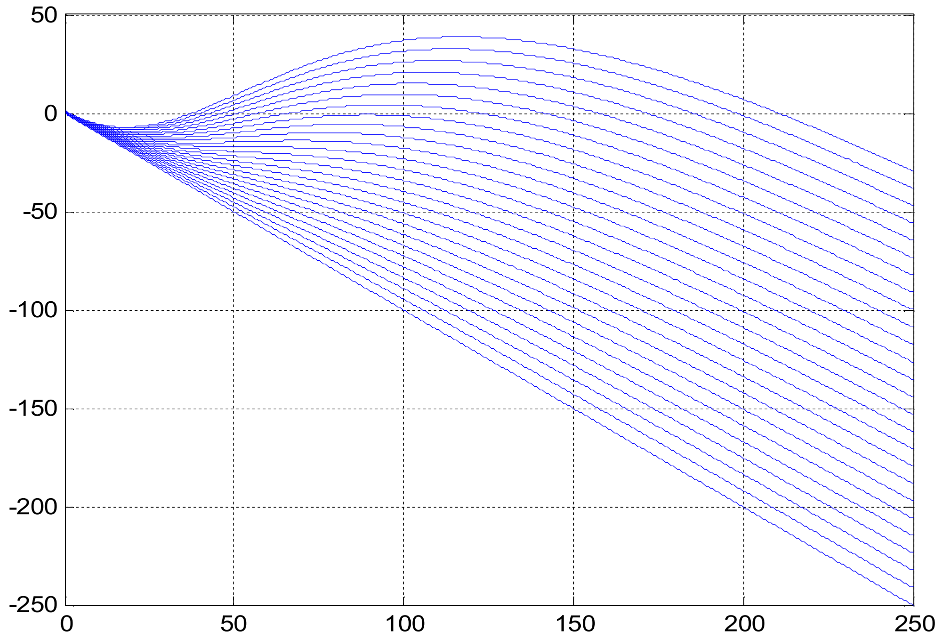


Figure 5.3 Simulation of the equation (5-1) for different P.

As you see, for smaller P, equation has only one critical point. But lac operon has 3 critical points. This equation is one dimension and differs from the case we did in chapter 4 which was piecewise linear as shown in figure 5.4.

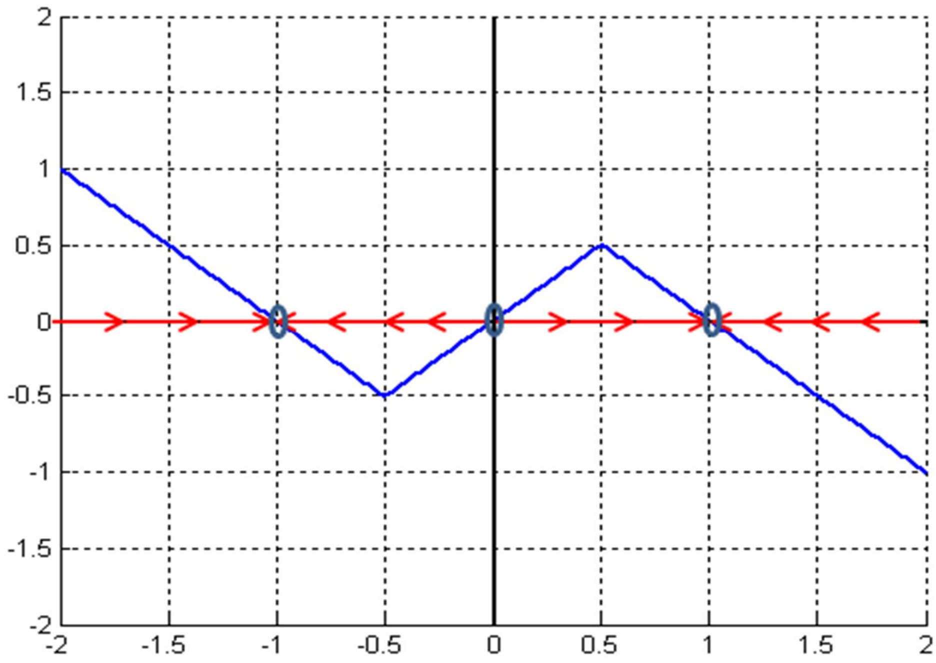


Figure 5.4 Piece-wise linear model studied in chapter 4.

Now, let's study our model for P=250, K=167.1 and K1=0.02, and T is between 0 and 250.

Critical points occurs at=1.5704, 37.7888, 210.6408. and tangents at these points are -0.9066, 0.6396, -0.6854.

Table 5.1 Critical points of equation(5-1) for P=250

Critical points	Tangent	Comment
1.5704	-0.9066	<0, stable
37.7888	0.6396	>0, unstable
210.6408	-0.6854	<0, stable

So the flow diagram would be as shown in figure 5.5:

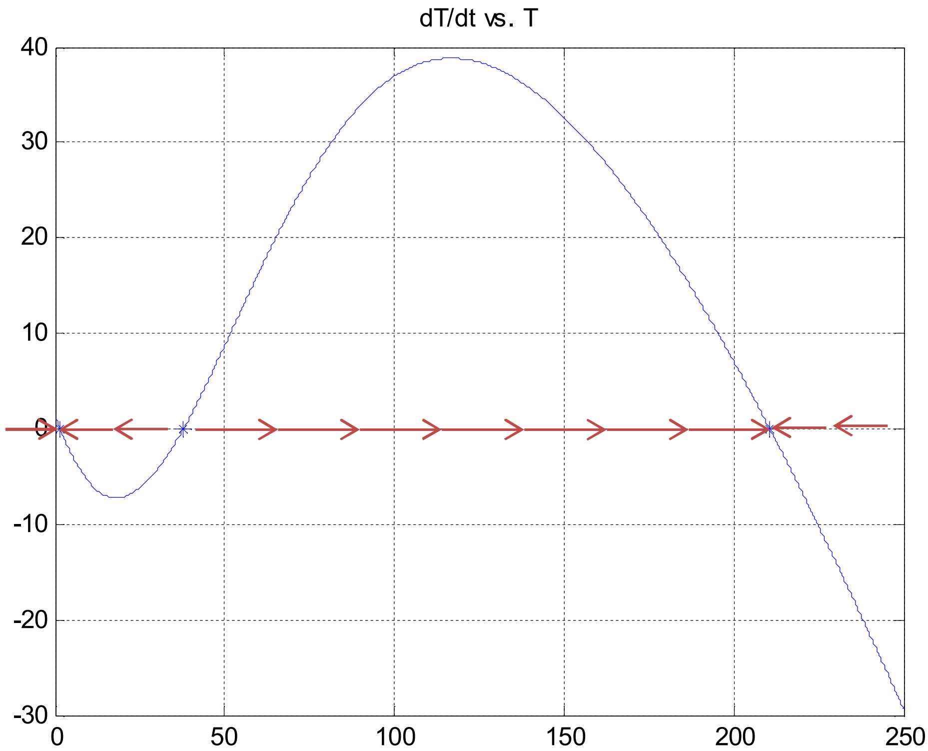


Figure 5.5 Direction of flows of T for equation (5-1).

5.2 How to Study Dynamic Partition

First, let's simulate one point, and then we will apply it to polytopes. First linearize then discretize it. Linearization is done with the help of following formula:

$$f(x) \approx f'(x_0) * (x - x_0) \tag{5-2}$$

Discretization:

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

$$\frac{T(k+1) - T(k)}{\Delta} = f\big(T(k)\big) + \frac{df(T)}{dT}\Big|_{T=T(k)} * \big(T(k-1) - T(k)\big)$$

$$T(k+1) = T(k) + \Delta * \left(f\big(T(k)\big) + \tfrac{df(T)}{dT}\Big|_{T=T(k)} * \big(T(k-1) - T(k)\big)\right) \tag{5-3}$$

Where T(k-1) is in fact, the point differentiated before. So remember the points of reachability in one dimension, if an unstable critical point is included in a polytope, split polytope around the point. Endpoints are crucial and simulate endpoints of the polytope. In every step, linearize the system and take one step more. P0=[0.1 3] which includes stable critical point 1.5704.

For dynamic partition we take the derivative of the middle point of the polytope, and linearize it at that point, and then use the polytope transformation as shown in figure5.6.

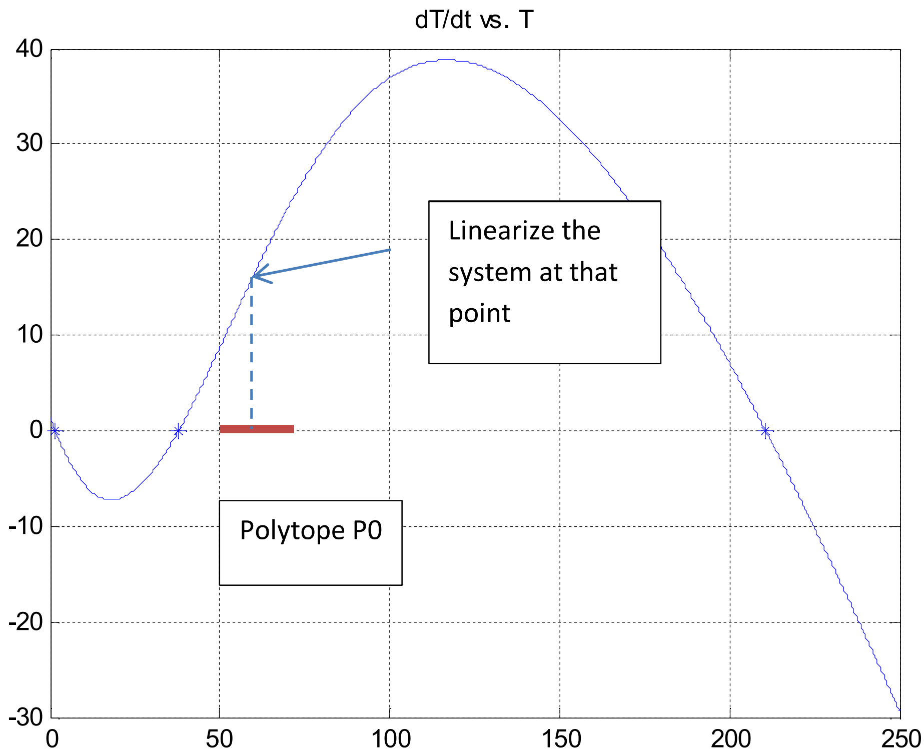


Figure 5.6 Dynamic partition.

This simulation uses forward Euler method. If we start at any point, the points will go to stable points either 1.5704 or 210.6408 as shown in figure 5.7.

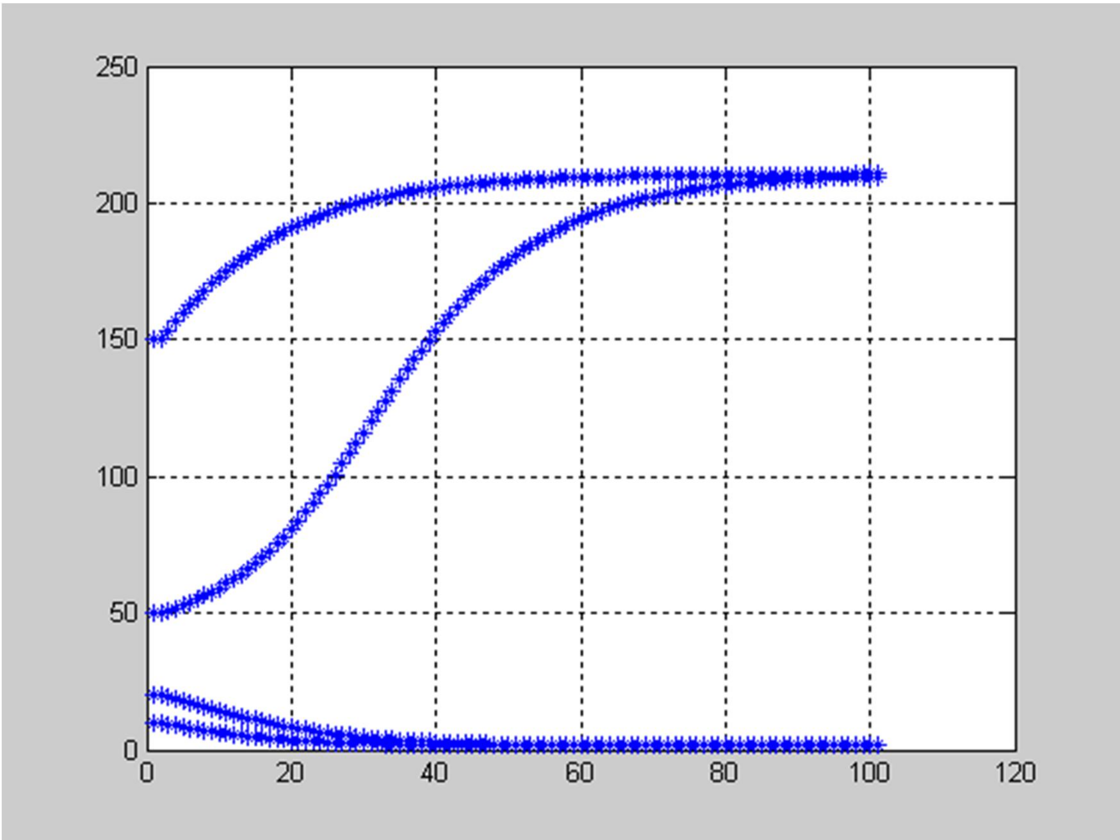


Figure 5.7 The simulation of 4 points: 10, 20, 50 and 150.

Now let's use polytope, to use polytopes we have to linearize the system. We don't have to linearize the system in every point, just the points that we need. This is called dynamic partition. The system is linearized at the middle point of the polytope and then transformation is applied to vertices and a new polytope is created.

The result is shown in figure 5.8 below, now you can say, every point between 50 and 150, goes to equilibrium point 210.6; and every point between 10 and 20 goes to another equilibrium point 1.5. Lines consisting of dense points indicate the vertices of the polytope. See appendix 4 for m-file written in Matlab.

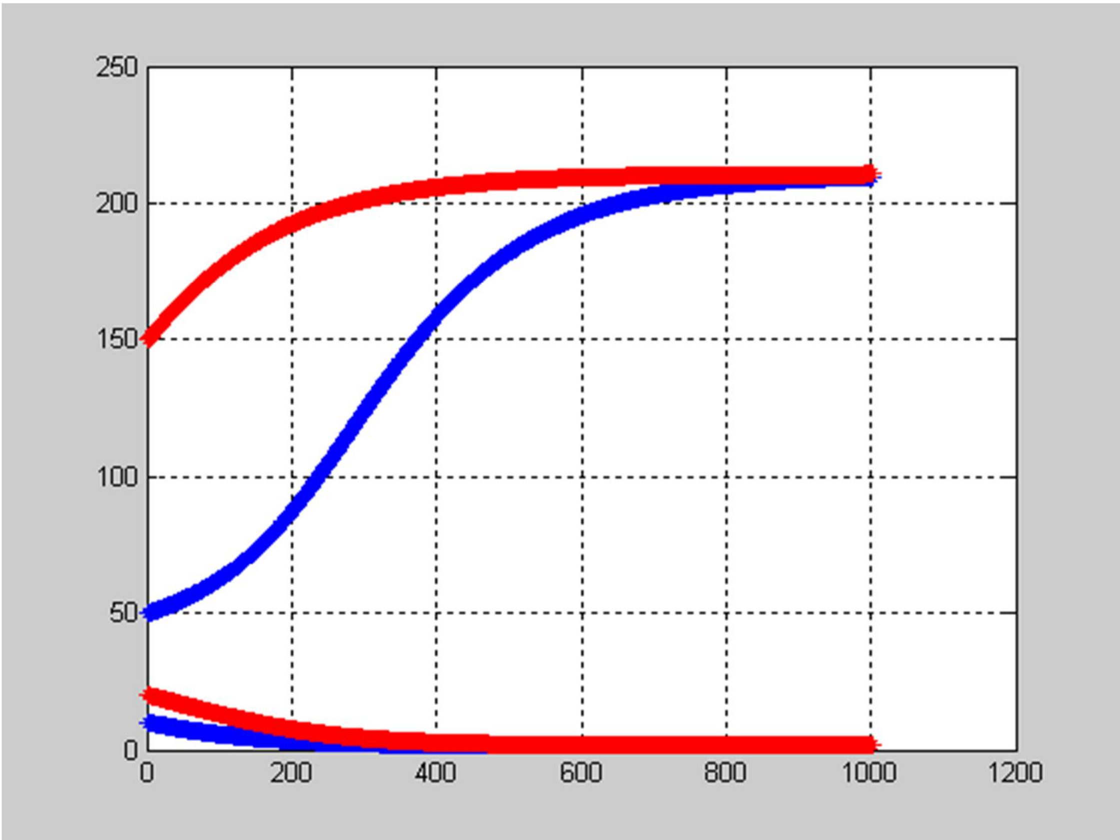


Figure 5.8 Simulation of two polytopes [50 150] and [10 20].

The simulation in figure 5.9 below belongs to the polytope of vertices [10 100]. This polytope includes the unstable equilibrium point [37.7888]. If we don't split, we get the wrong idea that polytope will go to region between 210.6 and 1.5. The points between 10 and 100 go to the region between 210.6 and 1.5. But this is wrong, the right simulation would be after splitting the polytopes as shown in figure 5.10. In our case we know there is an unstable equilibrium point, but normally we don't know it and we have to search for the unstable equilibrium points in every step of the simulation. In our case, we assume that there is an effective and correct way of finding an unstable equilibrium point.

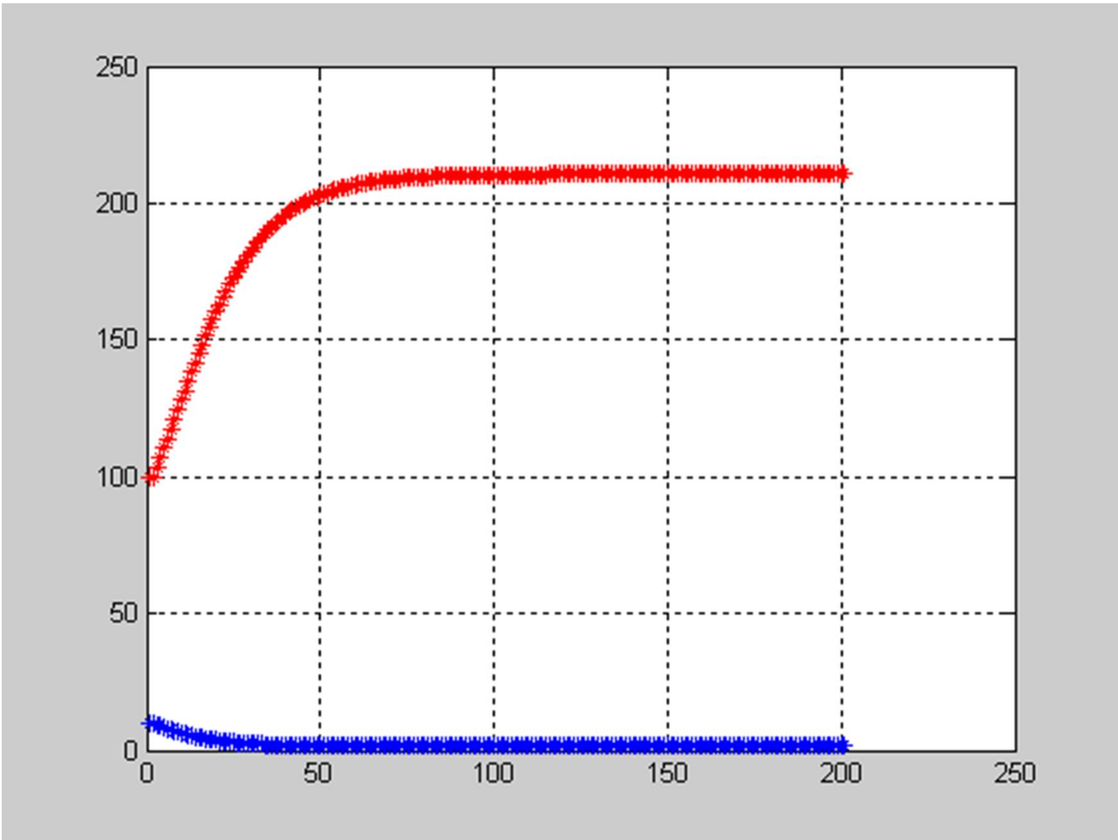


Figure 5.9 Simulation of polytope [10 100]. Polytope contains unstable equilibria and simulation gives wrong result.

The simulation after splitting polytope around unstable point will give the correct results as shown in figure 5.10. To split, the points close to unstable point is chosen, which is: point-0.01 and point + 0.01. Now, there are two polytopes that needs to be simulated. These two polytopes are free of unstable equilibrium point anymore. It is easy to see that these two polytopes will go to equilibrium points by looking the arrows in figure 5.5. So we say that the region between 10 and 100, goes to either points of 210.6 or 1.5, not into the region between 210.6 and 1.5.

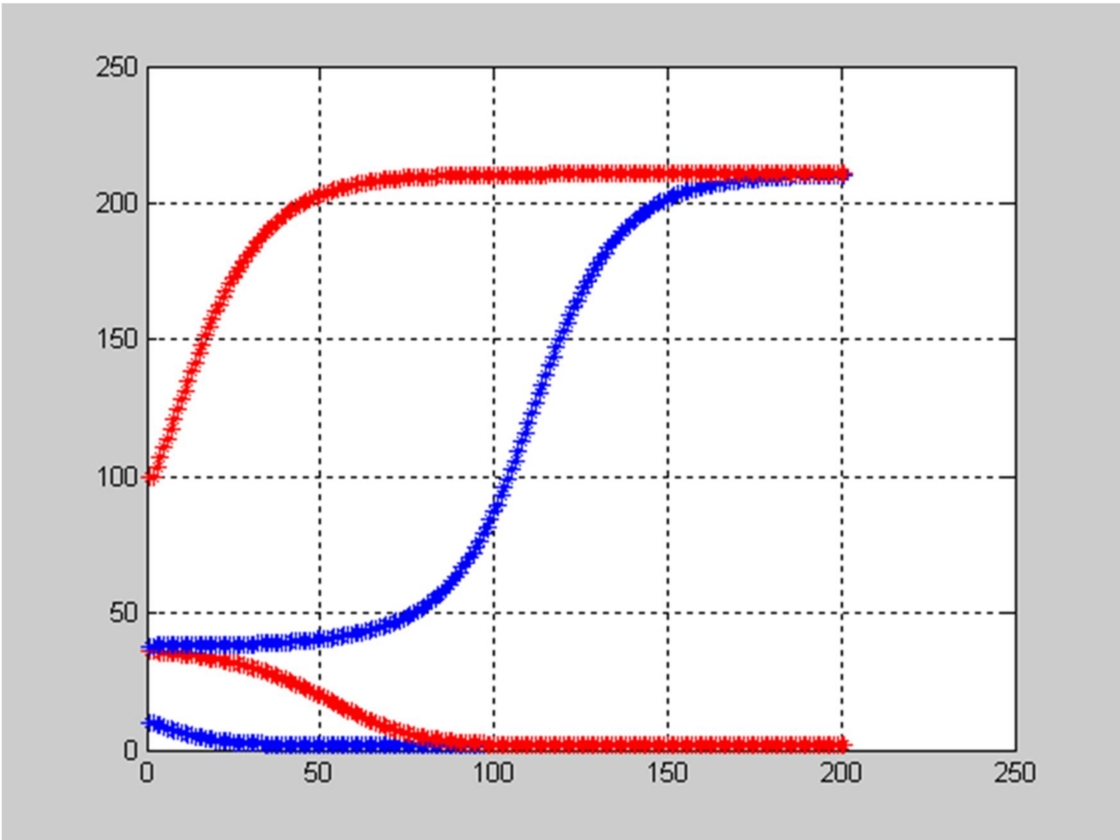


Figure 5.10 The correct simulation of Figure 5.9. Polytope has to be splitted around unstable equilibria point 37.79.

For polytopes, to compute a region we just calculated two points. In case of splitting, it is 4 points, instead of several points in case of simulating every point in region by brute-forcing. This system does not have any breakpoints, so there are no wrapping effects. In previous case, where our system was piecewise linear, and we had to consider the wrapping effects of polytope, where in every step; polytope had to be splitted around breakpoint.

5.3 Uncertainty of Parameter

One of the reachability goals is to simulate the system under uncertain parameters. The above function takes a polytope with vertices [100 200] and computes all possible region under uncertain parameter P, between 250-10 and 250+10. See appendix 5 for m-file written in Matlab.

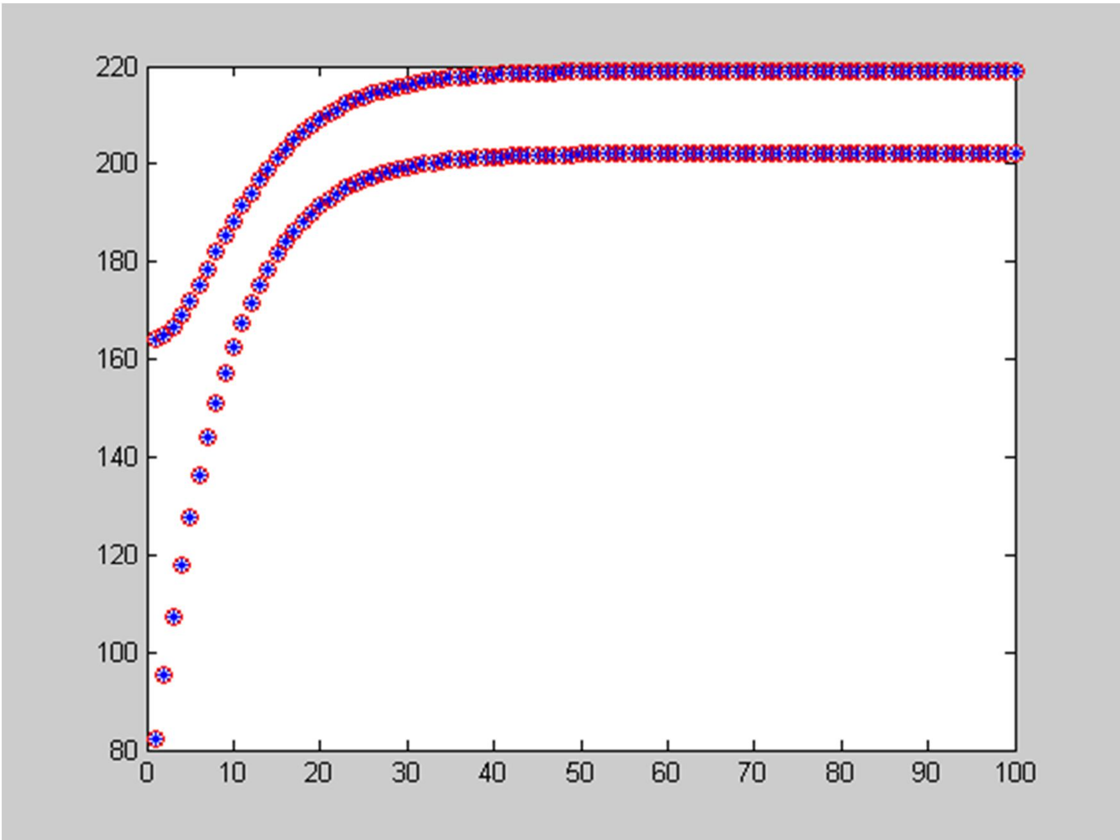


Figure 5.11 Simulation of T region between 100 and 200 under uncertain parameter P between 240 and 260.

The starting polytope region falls into this region as steps increases. The last polytope has the vertices:

```
>> extreme(X0)
ans =
  202.1869
  219.0358
```

This figure is okay. Because, if we simulate a region for P=240 and 260 we obtain the figure below. This means under uncertain parameter P, polytope will converge to a region, not a point.

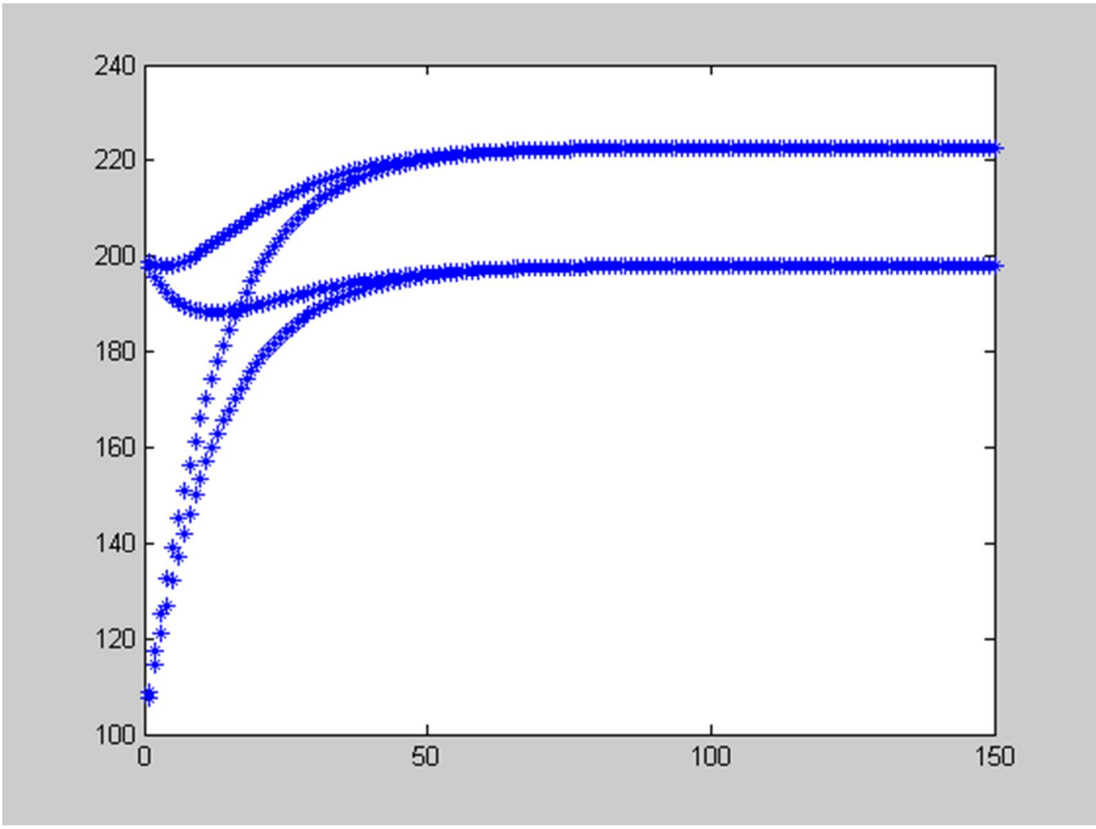


Figure 5.12 The simulation of polytope of [100 200] for P=240 and P=260.

CHAPTER SIX
REACHABILITY ANALYSIS IN THREE DIMENSIONAL GENE-PROTEIN RELATIONSHIP

6.1 Three Dimensional Gene-Protein Relationship

Below is the set of equations we derived in chapter 2. These equations are called piece-wise linear differential equations. These equations model the discrete nature of gene-protein relationship. A gene is active, only if the protein that activates the gene is above the threshold. This is the discrete nature of genes. So, they can be modeled by piece-wise linear differential equations.

$$\dot{x}_1 = \kappa_1 s^+(x_2, \theta_{21}) - \gamma_1 x_1 \tag{2-6.1}$$

$$\dot{x}_2 = \kappa_2 (1 - s^+(x_1, \theta_{11}) s^+(x_3, \theta_{31})) - \gamma_2 x_2 \tag{2-6.2}$$

$$\dot{x}_3 = \kappa_3 s^-(x_1, \theta_{12}) + \kappa_4 s^-(x_3, \theta_{32}) - \gamma_3 x_3 \tag{2-6.3}$$

This set of equations divide the state-space into orthants and in every orthant, there is a different set of differential equations. For example, if θ_{21}<x_2, θ_{12}<x_1< θ_{11}, θ_{32}<x_3< θ_{33}, then equations become:

$$\dot{x}_1 = \kappa_1 - \gamma_1 x_1 \tag{6-1}$$

$$\dot{x}_2 = \kappa_2 - \gamma_2 x_2 \tag{6-2}$$

$$\dot{x}_3 = \kappa_3 - \gamma_3 x_3 \tag{6-3}$$

This equation divides the state space into 2x3x3=18 orthants. To simulate this system, I chose the variables as: K_1=K_2=K_3=K_4=20; and γ_1= γ_2= γ_3=2; and θ_{21} =θ_{11}= θ_{31}=4, θ_{12}= θ_{32} =8. Below is the orthants that is divided by these thresholds. See appendix 6 for m-file written in Matlab.

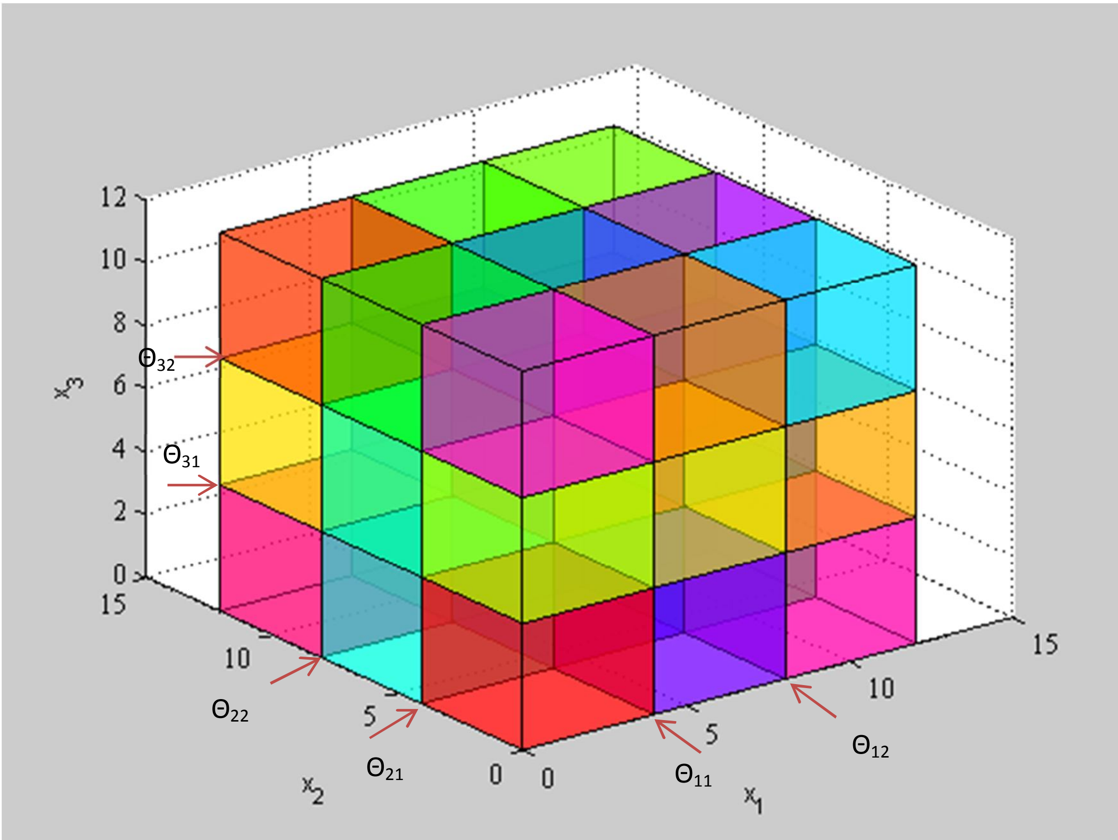


Figure 6.1 Phase space divided by thresholds.

Every box is an orthant, and in every orthant there are different set of equations. Now let's simulate one point using ode functions of Matlab.

6.2 Simulation of One Point by Using Signum Function

Simulation of one point is done using the ode45 function in Matlab. The simulation is shown in figure 6.2. See the appendix 7 for the m-file written in Matlab. In our case, the equilibrium points of the equations belonging to the specific orthant does not lie in the same orthant. These equilibrium points are called virtual equilibrium points. Every orthant just pushes the point into another orthant.

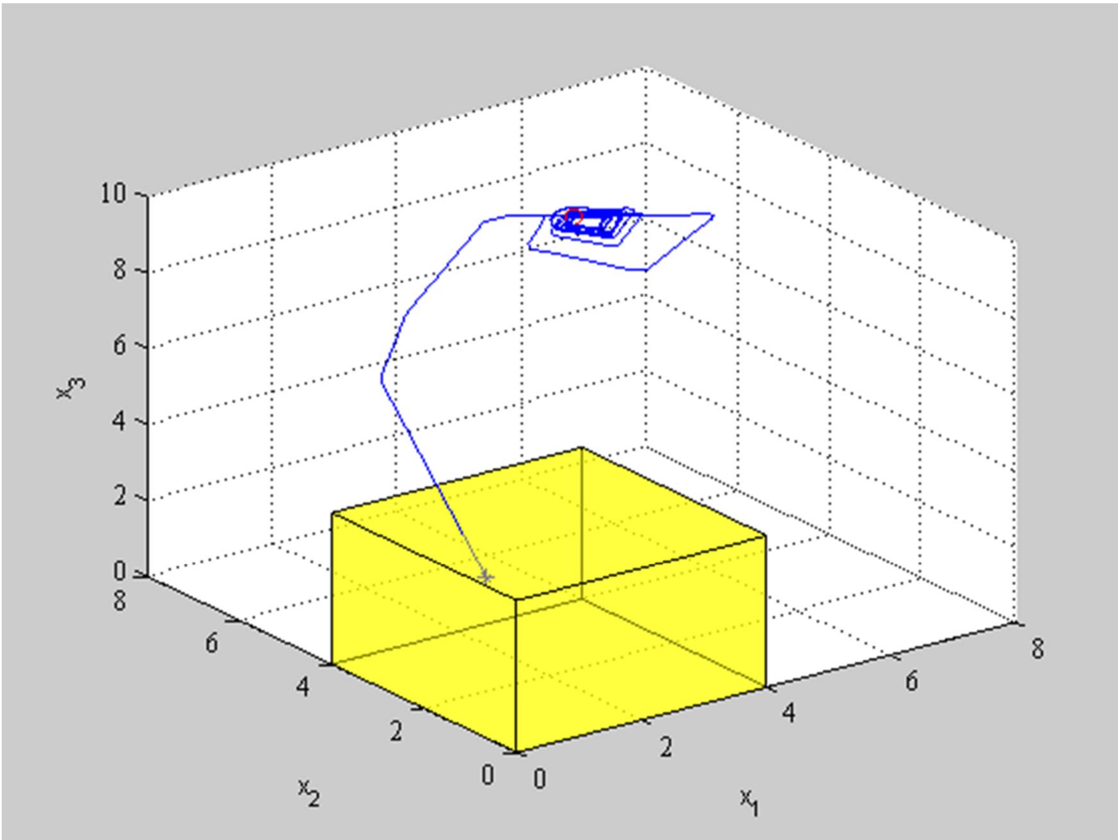


Figure 6.2 Simulation of point [1 2 3]. This point is inside orthant 1.

This was the simulation of one point in one orthant; now let's look at other orthants. By taking one point in each orthant and simulating them, the below simulation is obtained. These differential equations are all has stable point. But the focal point of orthant may not be inside that orthant. In this case, trajectories will tend towards to near orthant. The focal state of an orthant, in form of x'=k-gx is, x*=k/g; where in our case, k is 0, 20 or 40, and g is 2. So, the focal points of an orthant may be 10, 20 or 0. By choosing another g3 as 20, let's say, for x_3, then another simulation would be as in case in figure 6.4. In this case, the focal point of one orthant is [20/2, 20/2, 20/20]=[10 10 1] and by chance it is inside that orthant, as we can see from figure 6.4, because every point leaks into it.

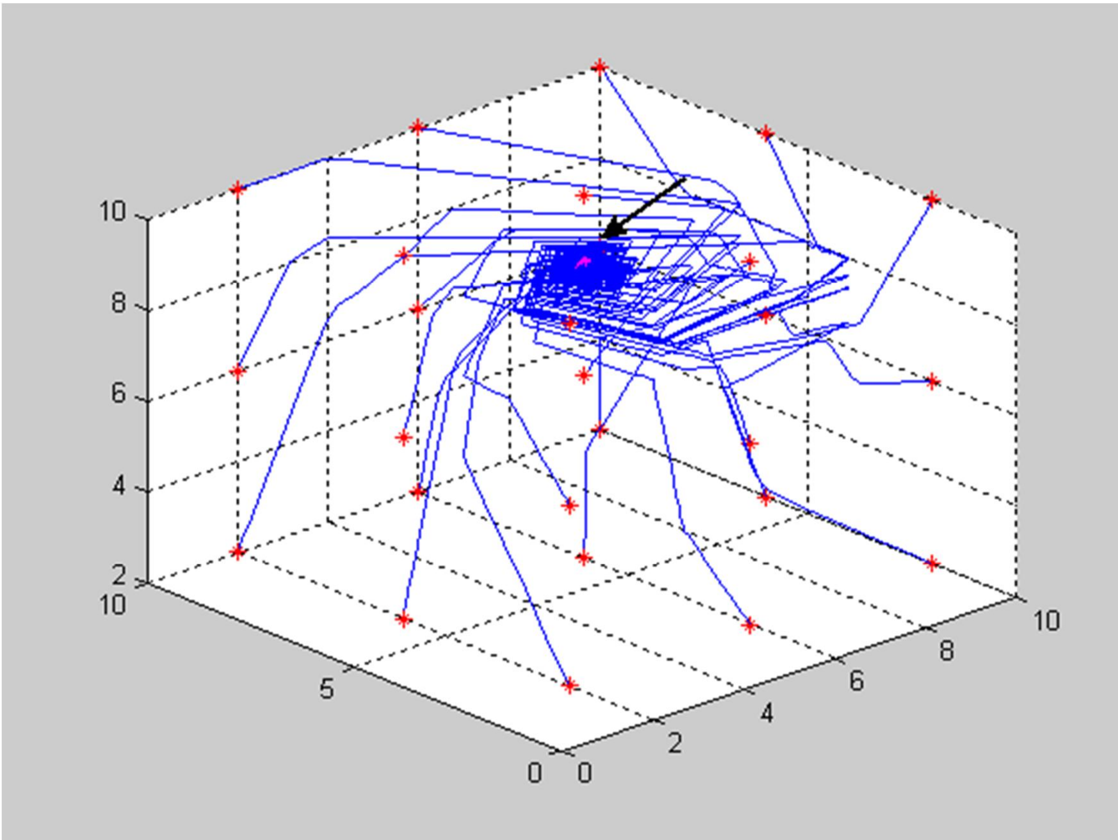


Figure 6.3 Simulation of middle points taken from every orthant.

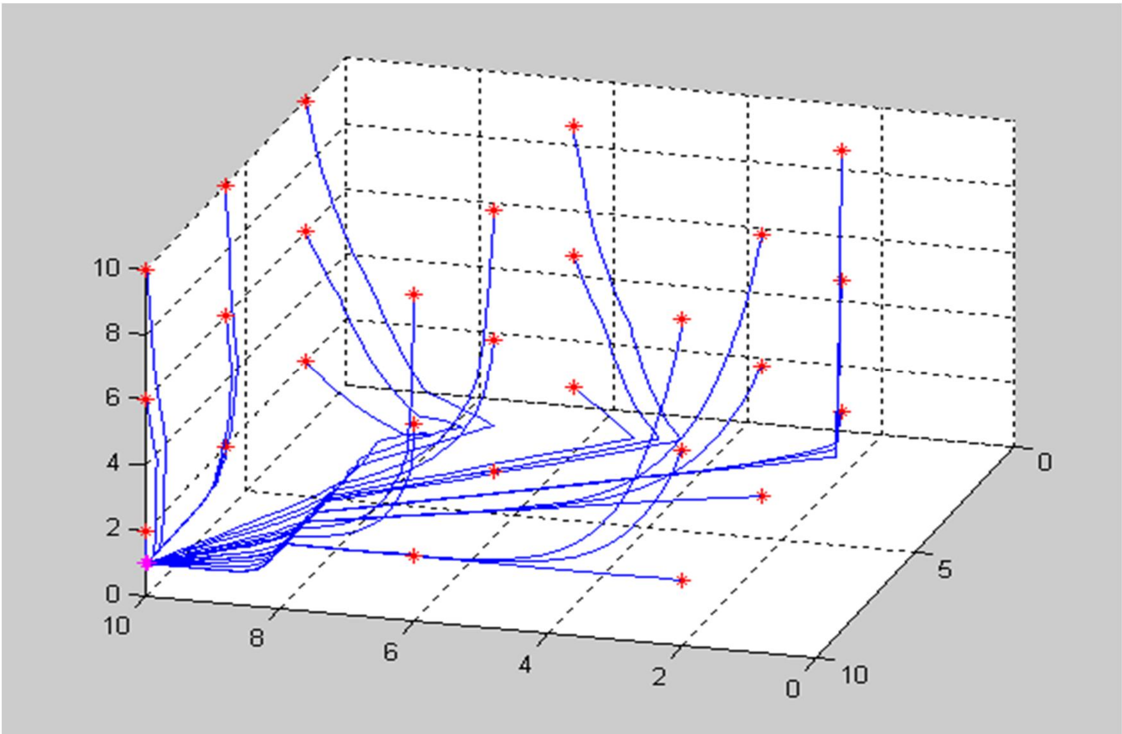


Figure 6.4 Simulation of every middle point of orthants for different k and degradation values.

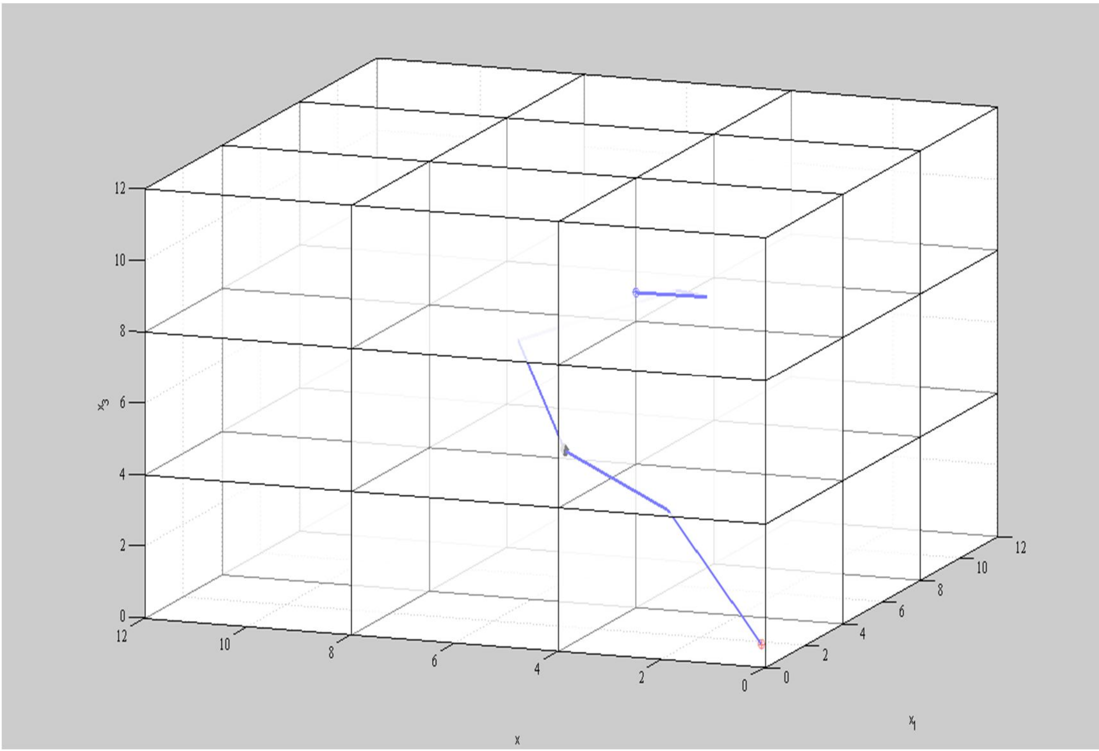


Figure 6.5 Simulation of a random point in orthant 1. This point crosses to other orthants.

6.3 Simulation of One Point Using Hill Function

An hill function is an approximation of step function. You can see the hill curve in figure 6.6 below with increasing m.

$$h^{+}(x_j, \theta_j, m) = \frac{x_j^m}{x_j^m + \theta_j^m} \tag{6-4}$$

Simulating the same points, using hill and step functions we obtain the figure 6.7 and 6.8 below. As you see, trajectories with hill function are smoother.

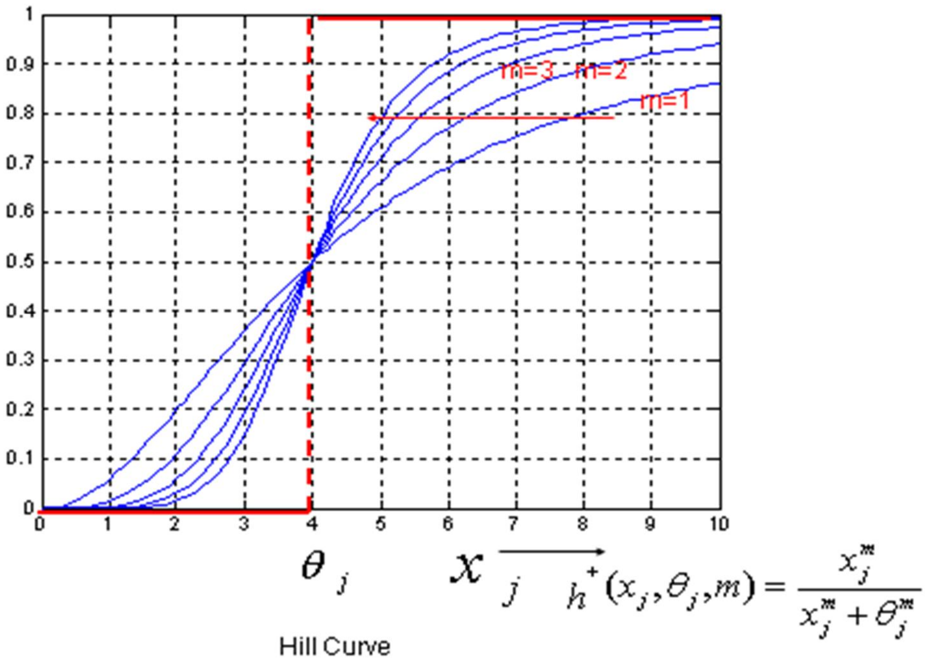


Figure 6.6 Approximation to signum function for threshold value of 4.

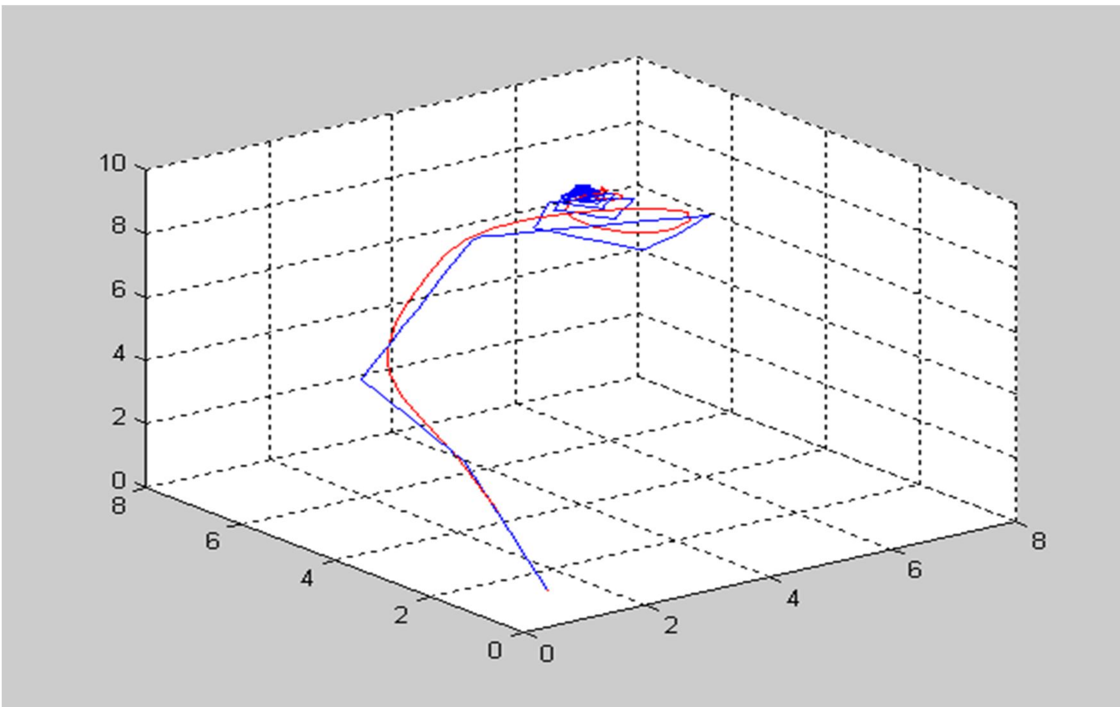


Figure 6.7 Simulation of same points with using hill functions (approximated signum) and signum function.

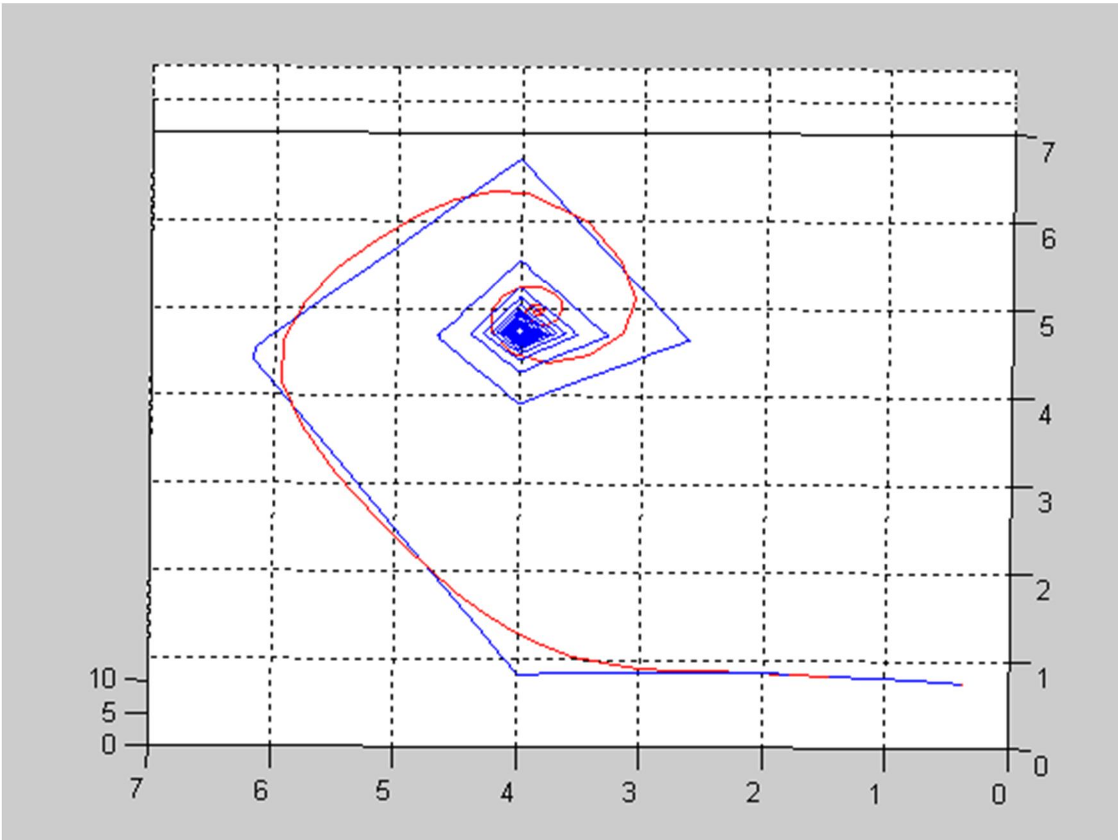


Figure 6.8 Above perspective of figure 6.7.

6.4 Defining Polytopes

Before studying reachability analysis, first let's look at how we can define polytopes. The "polytope( )" is a mpt toolbox function that defines a polytope for given vertices. Let's take 4 random points between 0 and 1, and define a polytope in region 1 as in figure 6.9. After running polytopegrid.m, which is a m-file written for this study and writing the below code to the command window.

```
>>Pgrid=polytopgrid;P=polytope(rand(4,3));plot(P),hold on, plot(Pgrid(1),'y')
```

Polytopgrid is a function that I wrote that divides the space into orthants determined by the equations and returns a polytope array.

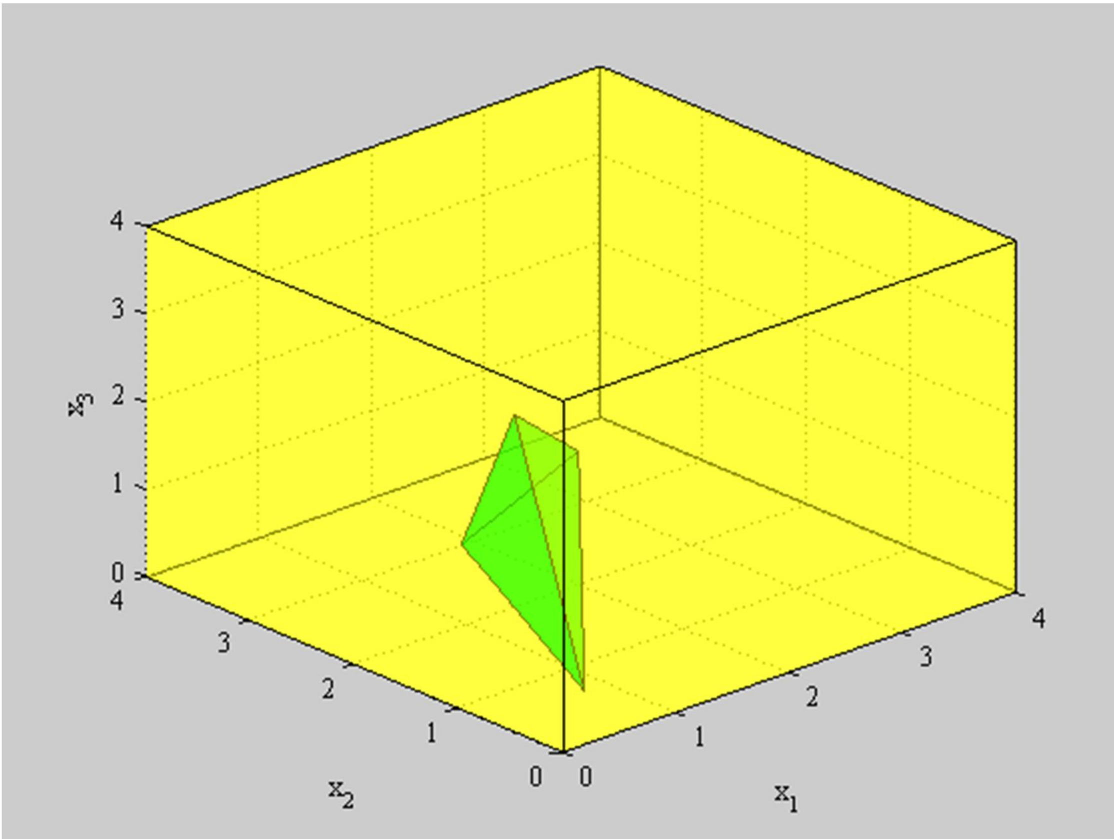


Figure 6.9 A polytope in orthant 1.

And another polytope example, for another polytope, that is divided by different regions. Polytope that has the vertices below, rows indicates the point location.

4.9605	3.6203	2.0137
4.7653	2.6332	4.9634
3.1833	2.9160	4.4354
3.9869	2.0402	4.3112

This polytope is divided by the 4 orthants as shown in figure 6.10. Polytope_bol is a function written by me that divides the polytope and returns the sub-polytopes as a polytope array.

```
>> [P1 KK]=polytope_bol(P,Pgrid)
```

P1=

Polytope array: 4 polytopes in 3D

KK =

1 2 4 8

Where KK is the array of id's of the orthants that divides the polytope. If we plot P1

>>plot(P1)

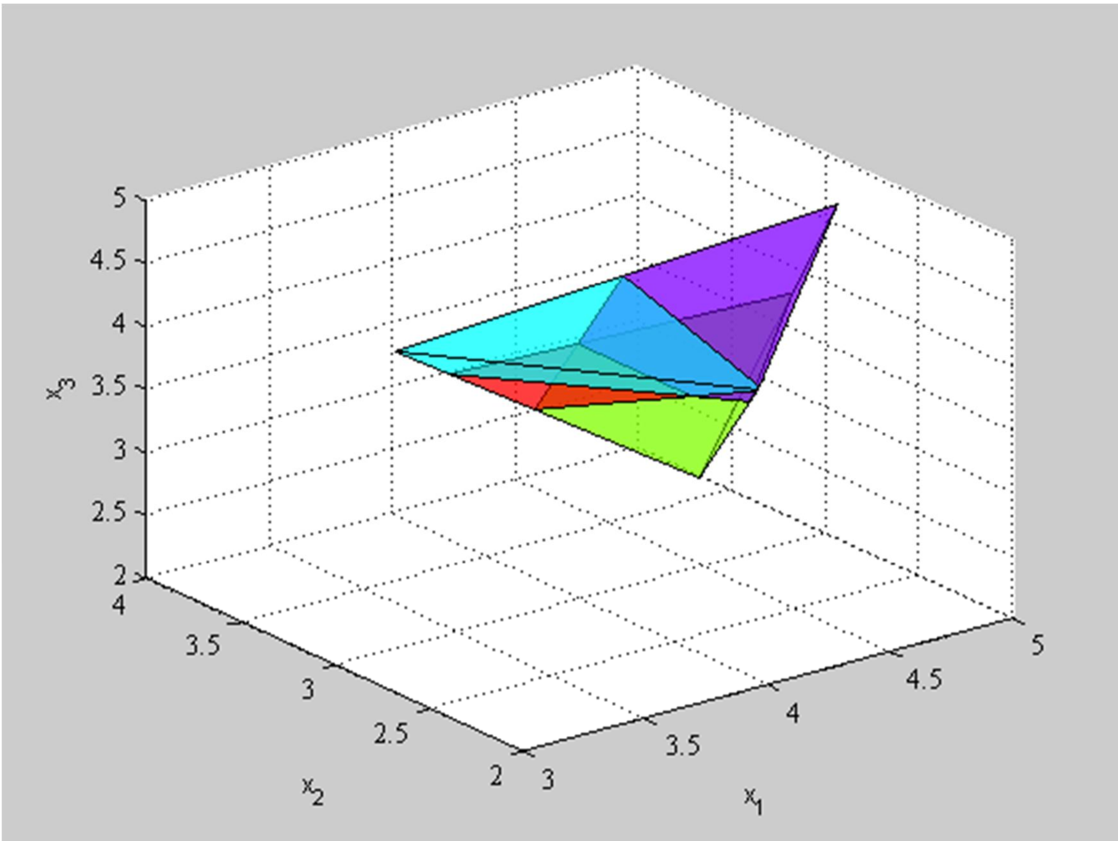


Figure 6.10 A polytope divided by the orthants. Each color indicates different orthant.

In figure 6.11, 6.12, 6.13 and 6.14 you can see the intersection of the polytope in figure 6.10 and the 4 orthants crossing that polytope.

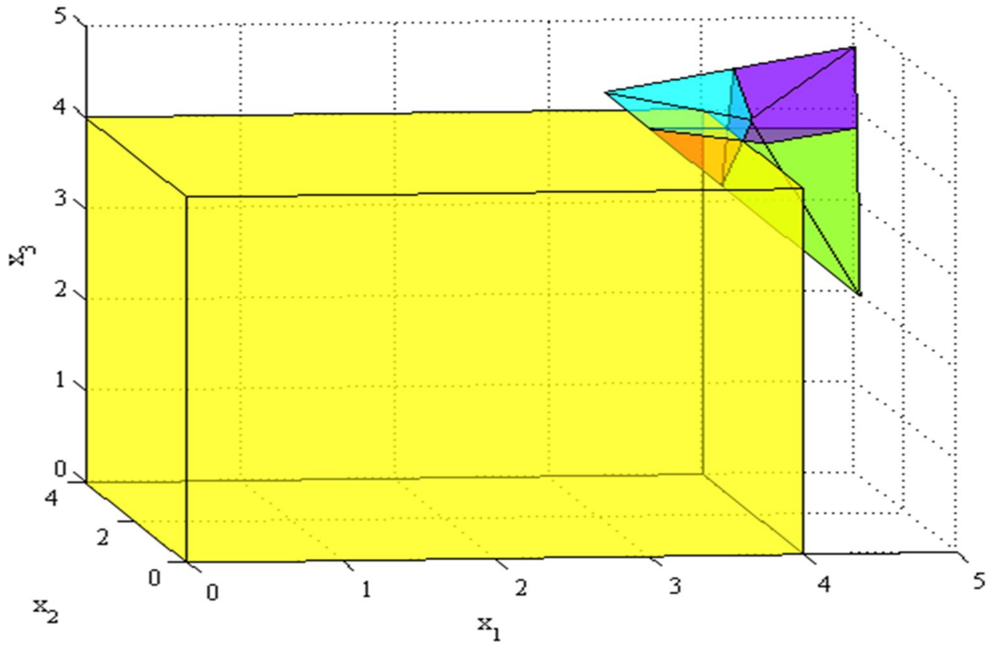


Figure 6.11 Polytope crossing orthant 1.

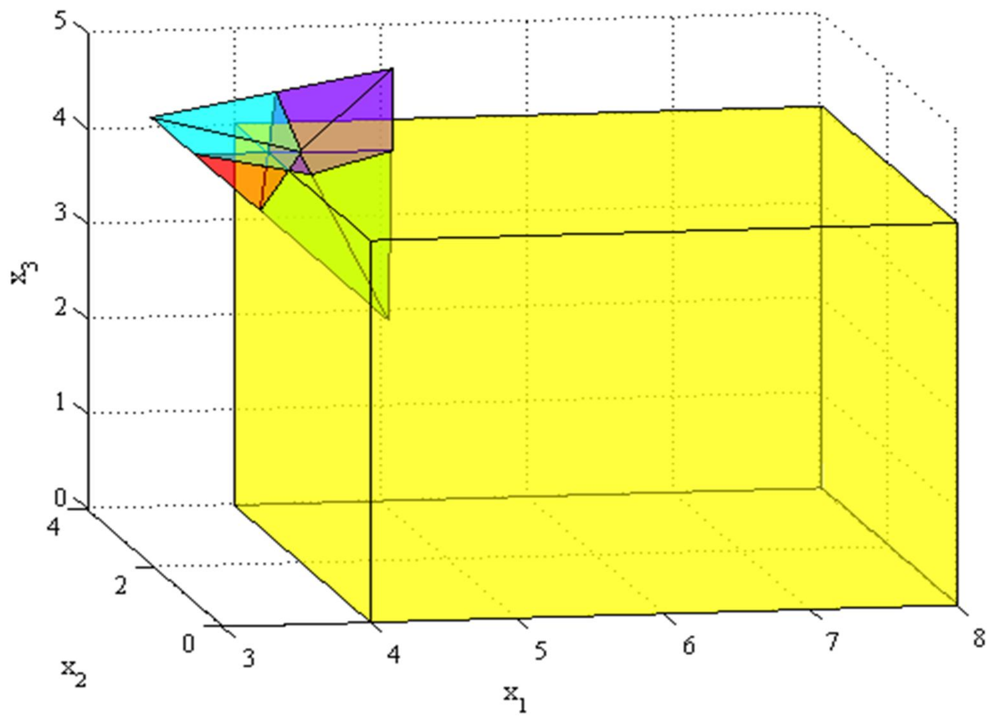


Figure 6.12 Polytope crossing orthant 2.

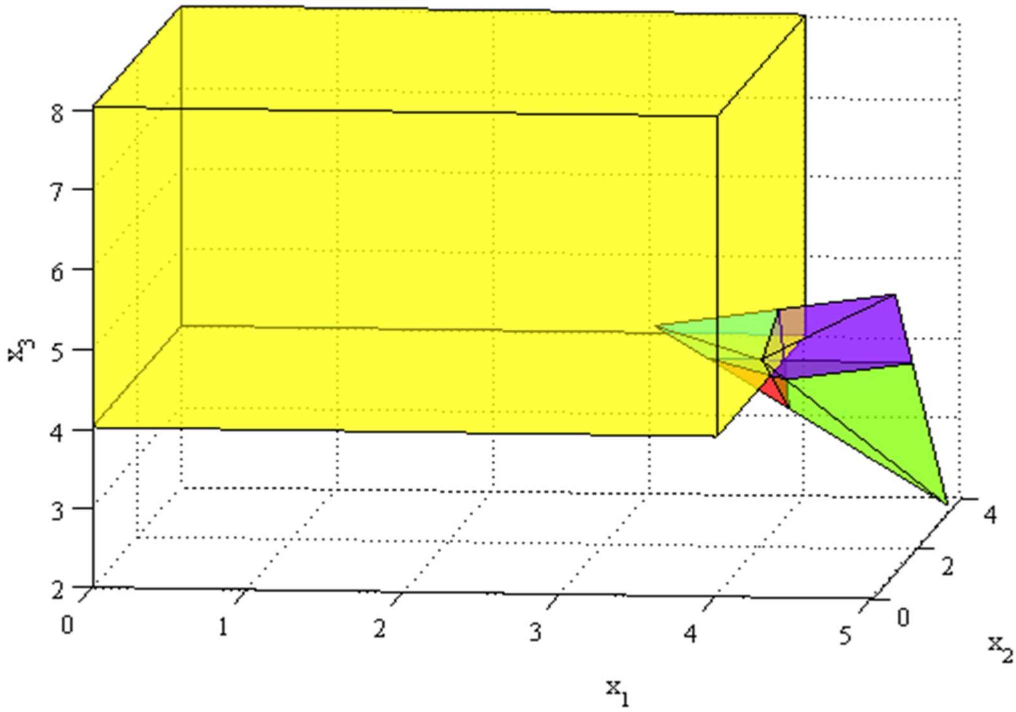


Figure 6.13 Polytope crossing Orthant 4.

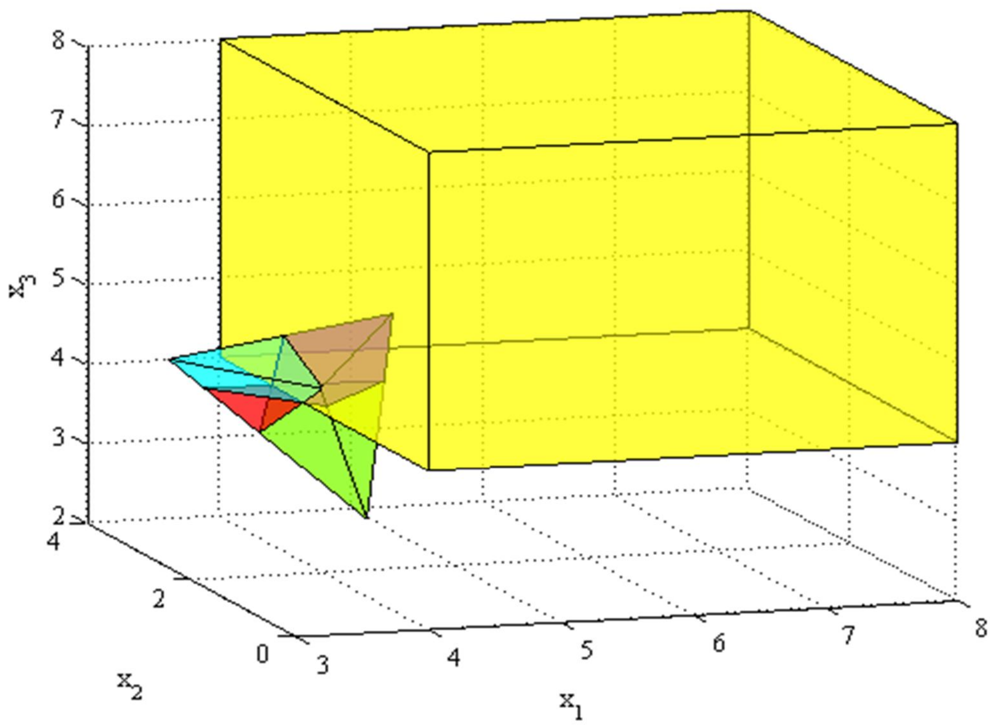


Figure 6.14 Polytope crossing orthant 8.

As we discussed before in order to continue reachability analysis after polytope falls into different regions, we have to divide it into sub-polytopes and simulate each sub-polytope. Below is the simulation of the sub-polytope that is divided by 8th orthant.

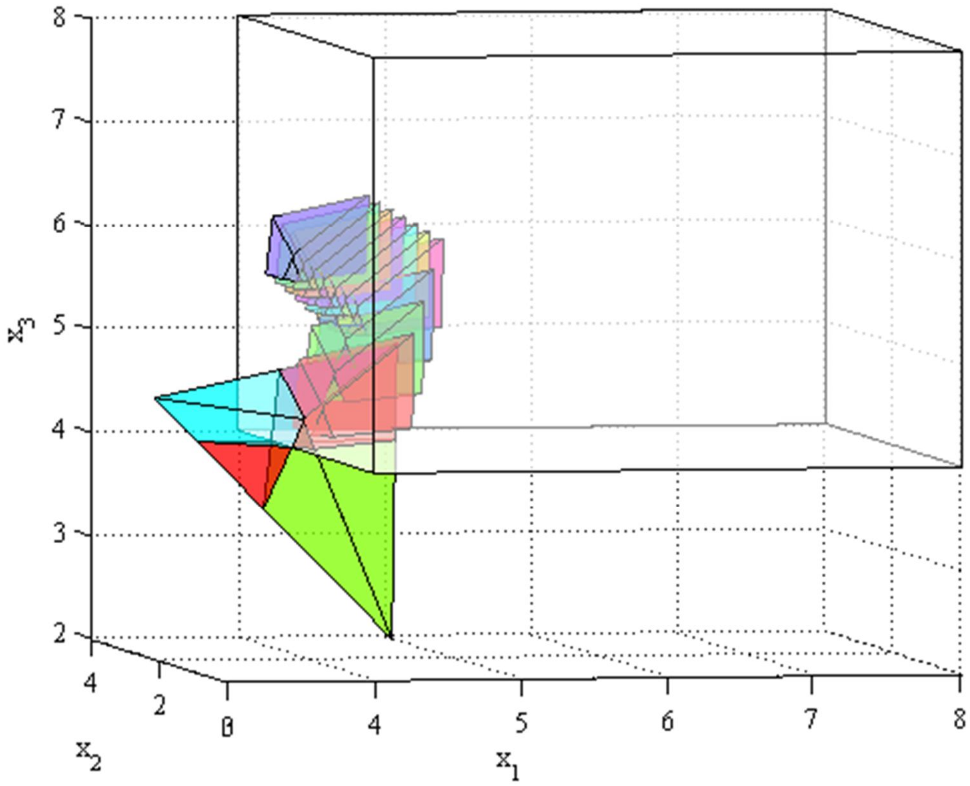


Figure 6.15 The sub-polytopes in different orthants are simulated independently.

Polytope simulation makes the computing easier, suppose you want to know where some region in orthant will go. Without using polytopes, you have to take several points and simulate each of them which is called brute-force technique. But using polytopes, you only have to compute the vertices, in our case 4 vertices, under linear transformation. Brute-force simulation would be as shown in figure 6.16.

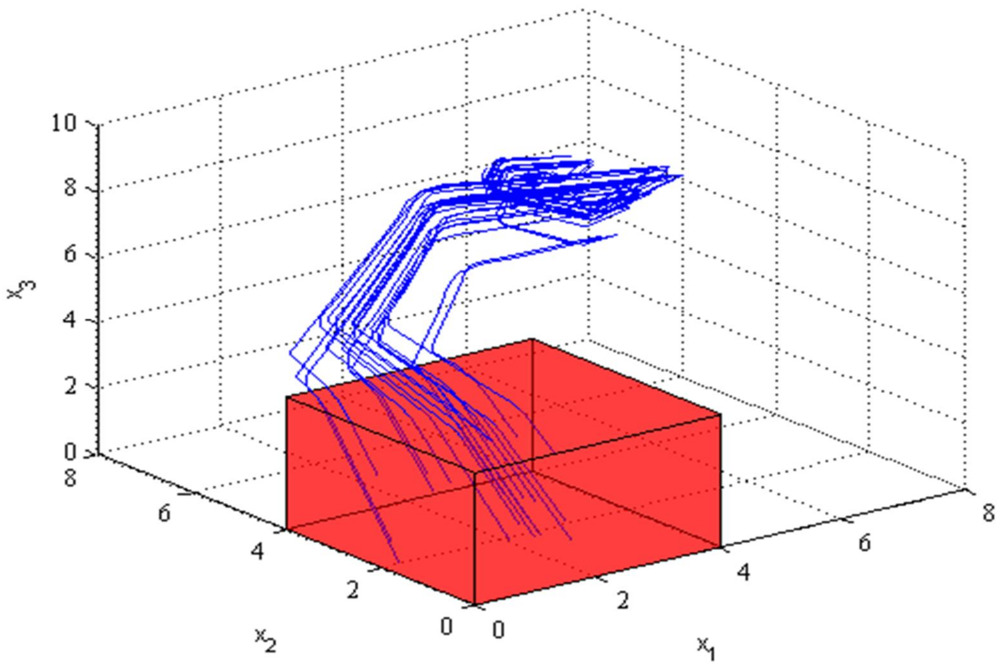


Figure 6.16 Brute-force of orthant 1. Random points are simulated.

Above perspective:

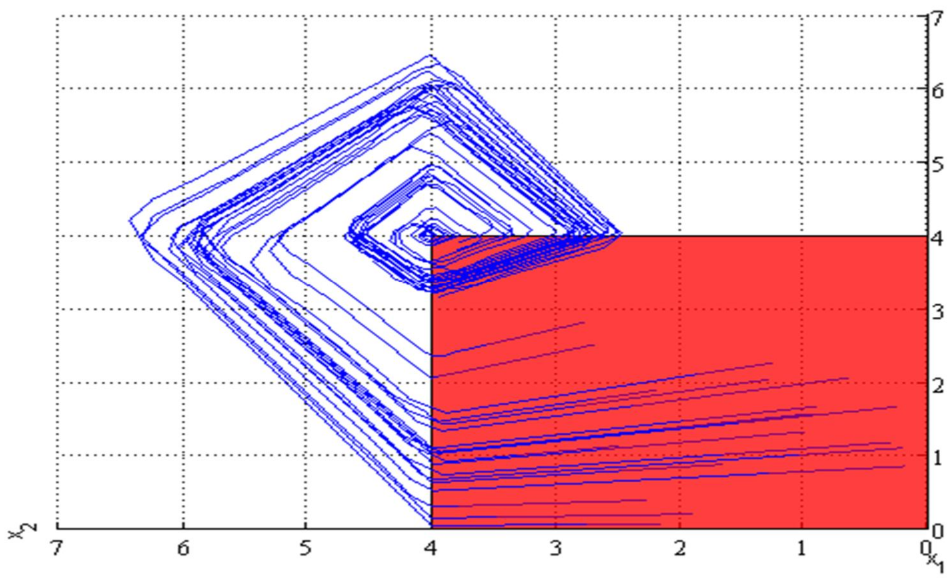


Figure 6.17 Above perspective of figure 6.16.

6.5 Fixed Grid

The system is piece-wise affine all the time, for a piecewise affine system, we can use “range” function that is included in mpt toolbox. Range function computes the affine transformation of a polytope. Its usage is: P=range(Q,A,f), where $P = \{Ax + f \in$ $R_n \mid x \in Q\}$.

In orthant 1, our system is:

$$\begin{aligned} \dot{x}_1 &= 0 - 2x_1 \\ \dot{x}_2 &= 20 - 2x_2 \\ \dot{x}_3 &= 40 - 2x_3 \end{aligned} \tag{6.5}$$

Where f is a column vector of [0 20 40] and A is `[-2 0 0; 0 -2 0; 0 0 -2]`.

$$A = \begin{bmatrix} -2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

We have to discretize the system so that we can use range function. In order to use range function, we have to discretize the system in the form of x(k+1)=A*x(k)+B, where B here is, in fact f. In mpt toolbox, B is used for the vector to use to multiply with U. In orthant 1,

$$\frac{x(k+1) - x(k)}{\Delta} = A * x(k) + f \tag{6-6}$$

And so the equation becomes:

$$x(k+1) = (I + \Delta * A) * x(k) + \Delta * f \tag{6-7}$$

And now our parameters are A’=(I+Δ*A) and f’=Δ*f; Writing a simple routine in Matlab and choosing Δ=0.01.

```
d=0.01;
A=[-2 0 0; 0 -2 0; 0 0 -2];
A_=(eye(3)+d*A);
B=[0 20 40]';
f=d*B;
V=rand(4,3);
P=polytope(V)
teknokta(mean(V),1,'signum')
for k=1:10
P(k+1)=range(P(k),A_,f);
end
plot(P)
```

Where teknokta computes the middle point of the vertices and simulate it as a point using an ode solver in Matlab. You can see the simulation in figure 6.18.

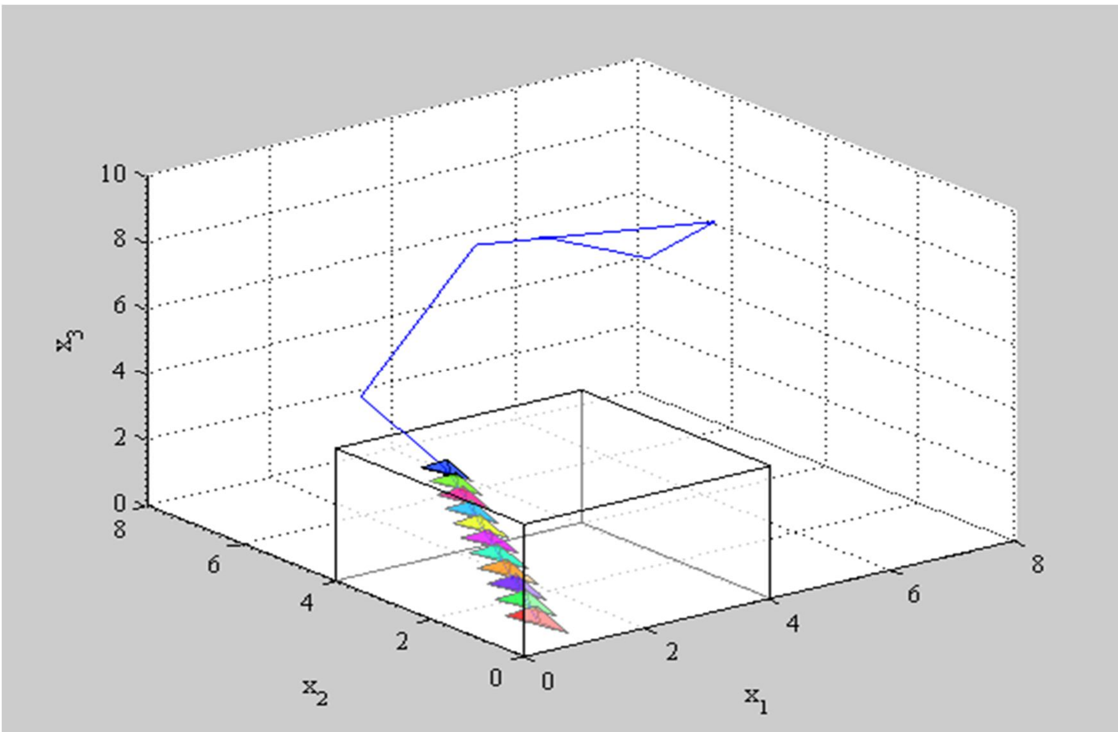


Figure 6.18 Simulation of a random polytope and the middle point of that polytope for orthant 1.

As you see, the point inside polytope always stays in the polytope as long as it is in orthant 1 as shown in figure 6.19. After orthant 1, polytopes will across another orthant and after that different set of equations must be applied and if polytope falls into two orthants then it has to be divided into sub-polytopes.

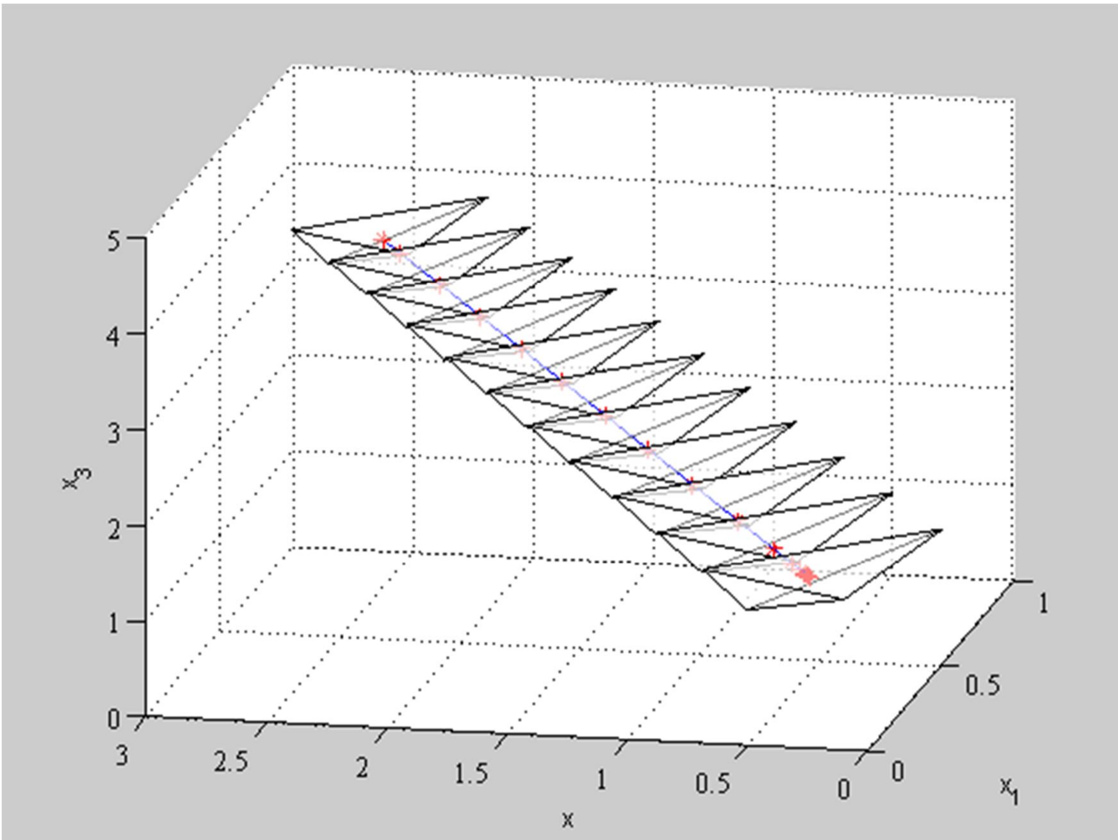


Figure 6.19 Simulation of random polytope in orthant 1 and middle point of that polytope. In orthant 1.

The points indicated by asterisks (*) are simulated with ode45. As you see, the point that starts with a polytope, always stay in that polytope. As we go on simulating, polytope will tend towards another orthant, so it will be divided by the orthants. The divided part has different color. You can see the simulation in figure 6.20 and 6.21.

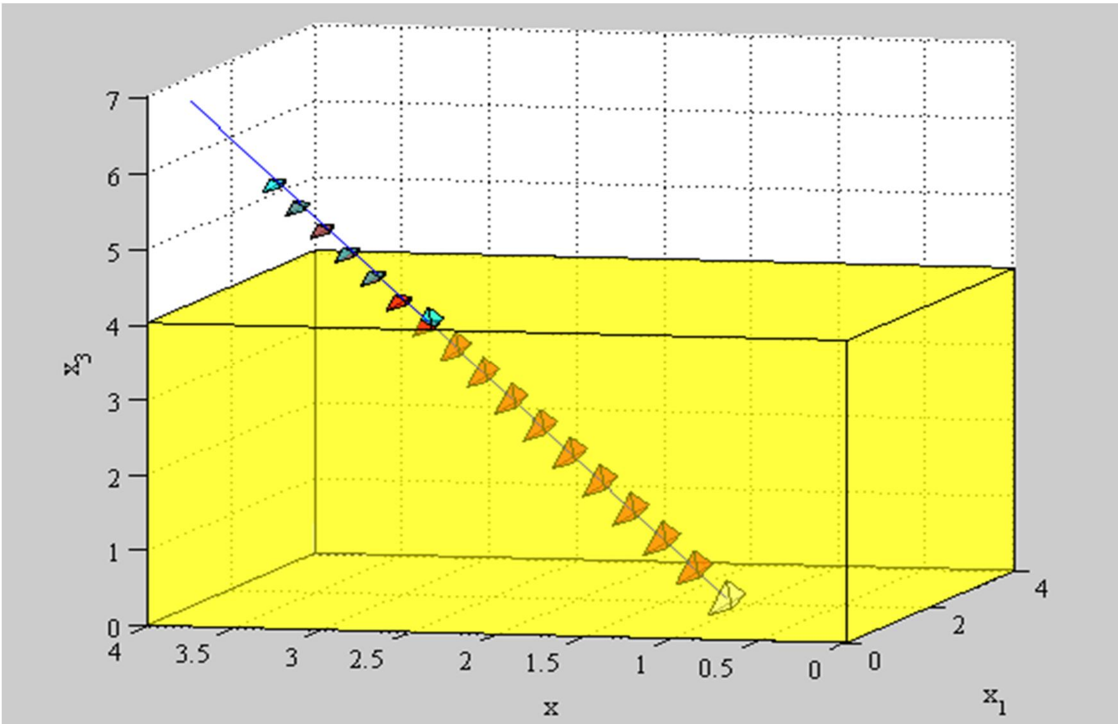


Figure 6.20 Simulation of random polytope and its middle point.

The 2 polytopes that is divided after crossing into another orhant is:

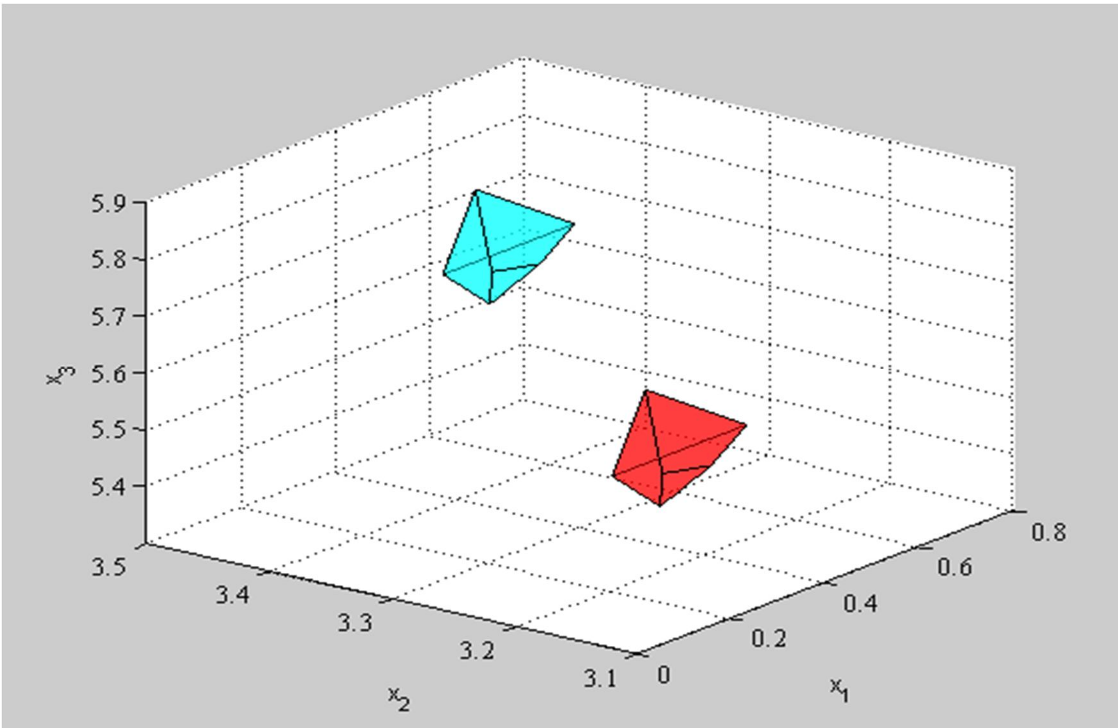


Figure 6.21 These are the polytopes that are the sub-polytopes of P0 and simulated independently in figure 6.20

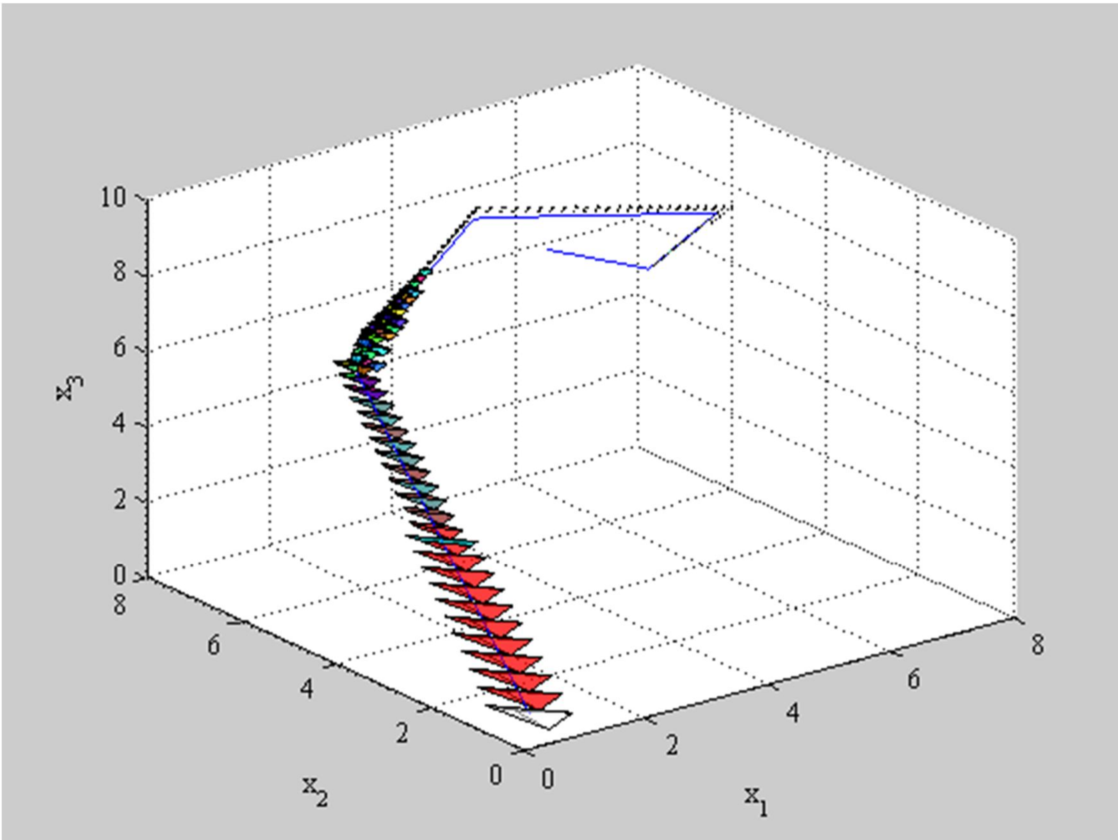


Figure 6.22 The simulation of random polytope for 60 step. It goes to equilibrium point and shrinks.

In figure 6.22, you can see the simulation of random polytope for 60 steps. In 60^{th} step, there are 33 polytopes. See appendix 8 for m-file written in Matlab, for fixed partition case using piece-wise linear differential equations.

```
>> Pset
Pset=
Polytope array: 33 polytopes in 3D.
```

These are the polytopes that belong to Polytope array Pset.

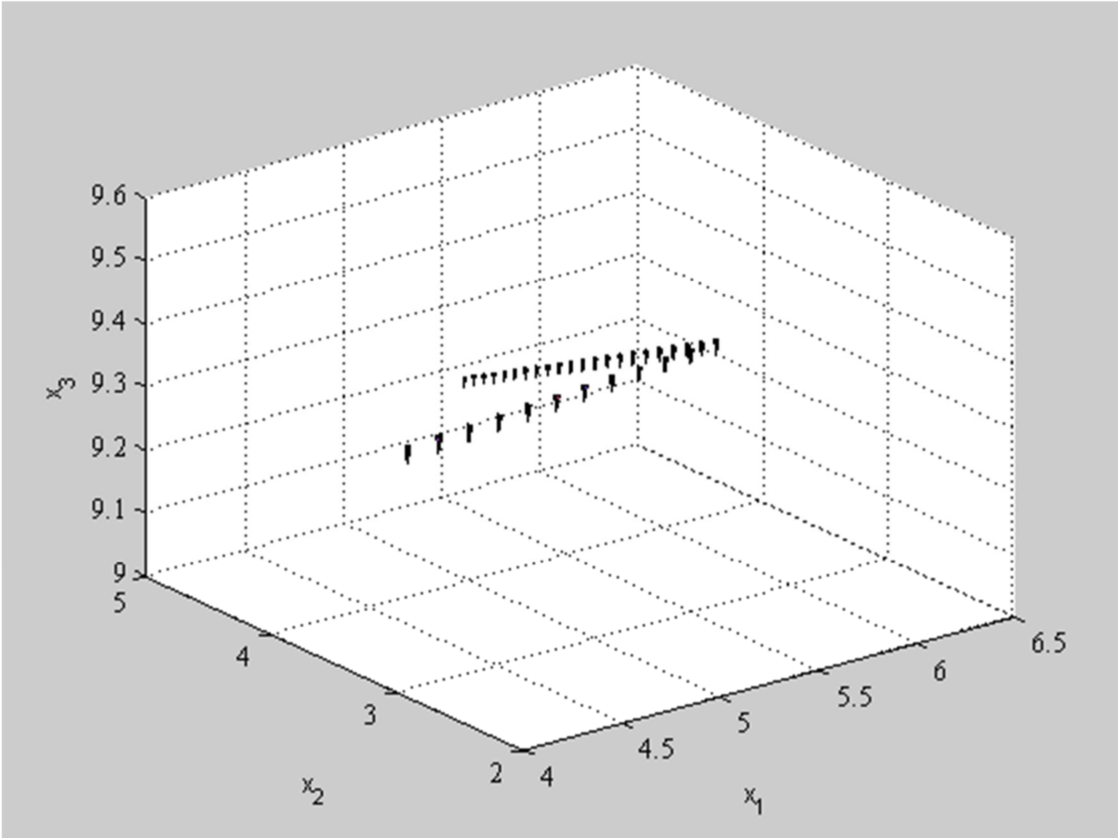


Figure 6.23 The polytope shrinks as it goes to equilibrium point.

This problem is about the division of polytopes. To overcome this problem, we can increase step size, so that polytope can jump over the boundary. Choosing d=0.05;

>> Pset

Pset=

Polytope array: 4 polytopes in 3D

But this time, the trajectory is not as accurate as before, due to the insufficient d.

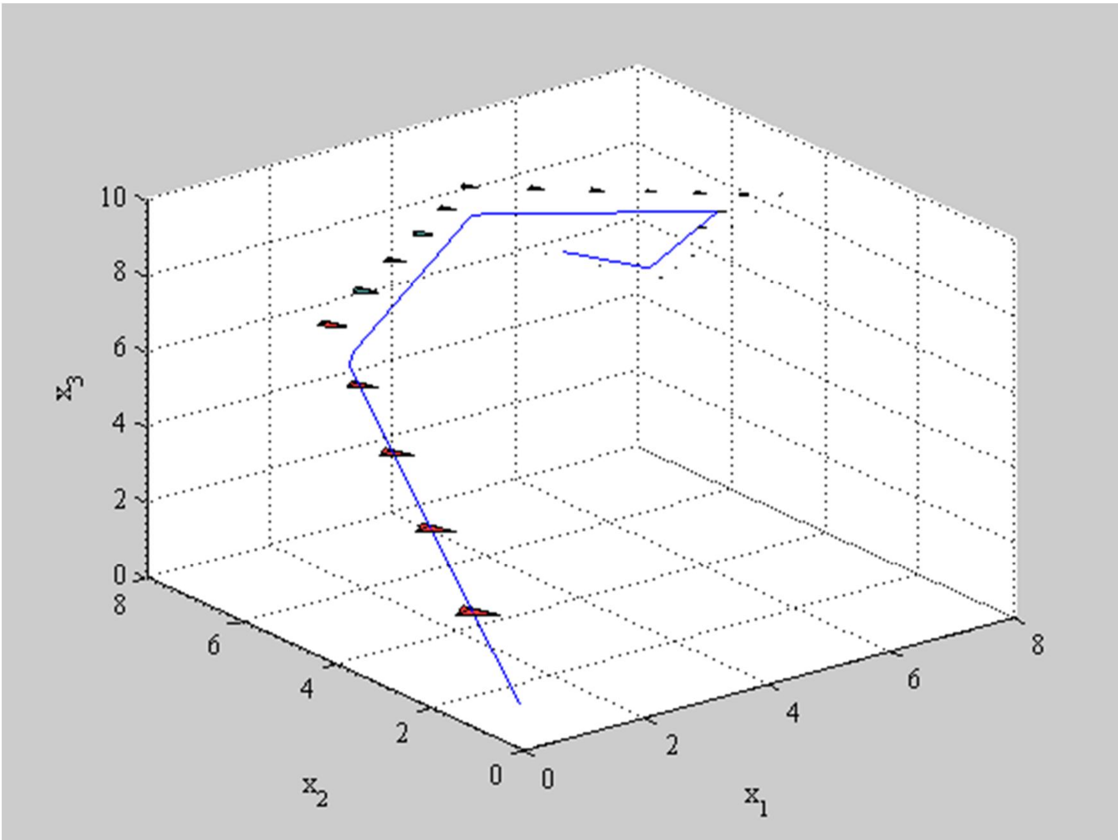


Figure 6.24 Choosing delta relatively large so that polytope can jump over the boundary. But this time, trajectory is not as accurate as before. If we don't want to divide the polytope then we can use other methods like dynamic partition. The necessity of division of polytope is discussed in chapter 6.7

6.6 Dynamic Partition

This is the case of dynamic partition. For dynamic partition, I used hill function instead of step functions and linearize the system at the middle point of polytope, as we did in chapter 5, for lac T. See appendix 9 for m-file written in Matlab.

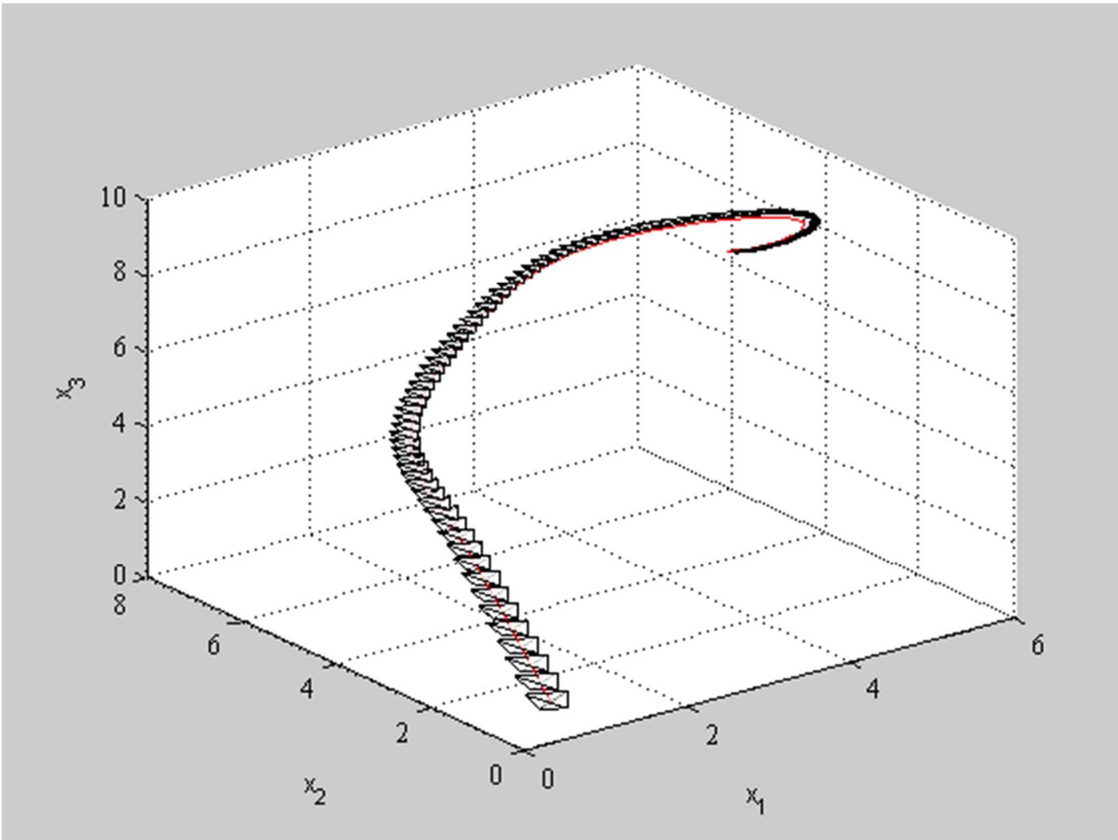


Figure 6.25 The line indicates the point trajectory by ode45 and polytopes trajectory.

6.7 Compare Fixed Grid and Dynamic Partition

In fixed grid we had to divide the polytopes, this makes the computing complex and hard to comment, but dynamic partition is much simpler. Do we have to really divide the polytope for fixed grid? Below simulation in figure 6.26 is alternative to figure 6.22. This time simulation is done without splitting the polytope. Simulation is more accurate and computing and programming challenges are easier. See appendix 10 for the m-file written in Matlab.

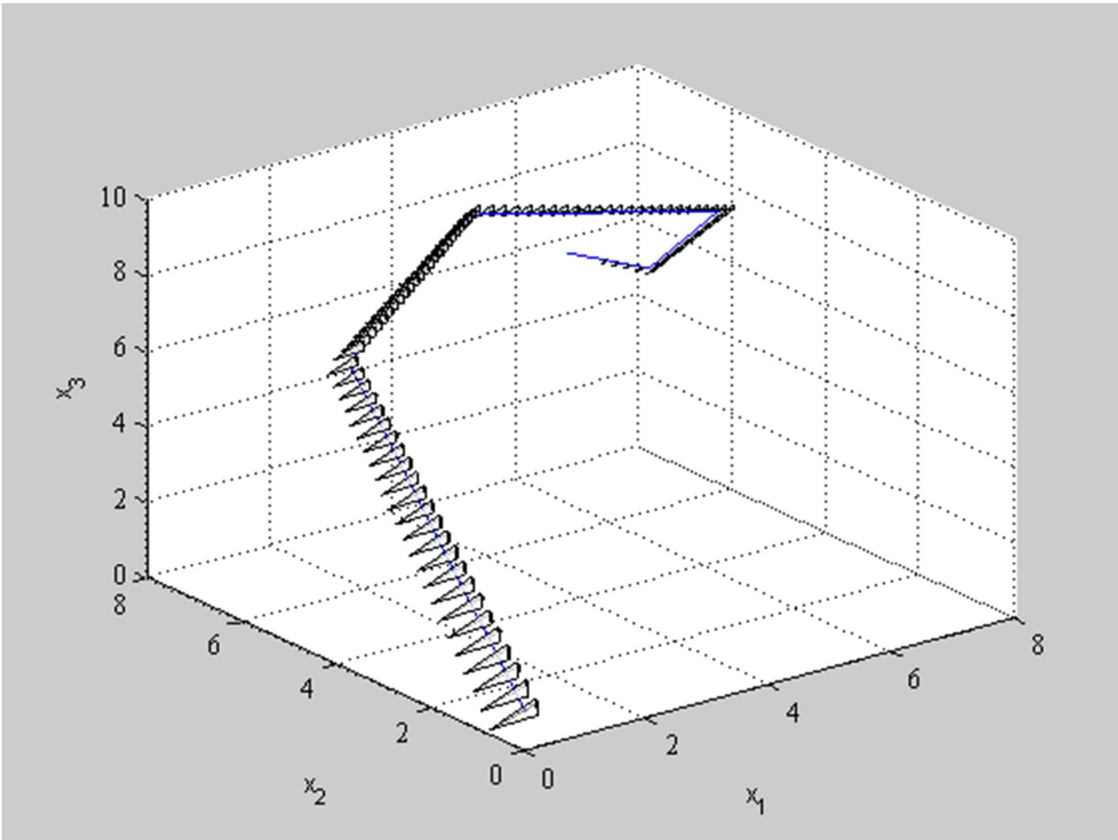


Figure 6.26 Simulation of a random polytope in orthant 1 without splitting at the boundaries

If we don’t divide the polytopes, the results are much better for fixed grid. If we don’t divide the polytope, in fact it is like dynamic partition with signum function case.

For fixed grid, vertices do not differ as step increases in particular orthant. Suppose, we run the vertices and the middle point of the polytope independently. This won’t make a difference because every vertices or points in that orthant are characterized by the same linear differential equations. But in dynamic case, the matrices A and B differs for every vertex of the polytope and middle point. So there comes the linearization error. See figure 6.27 and 6.28.

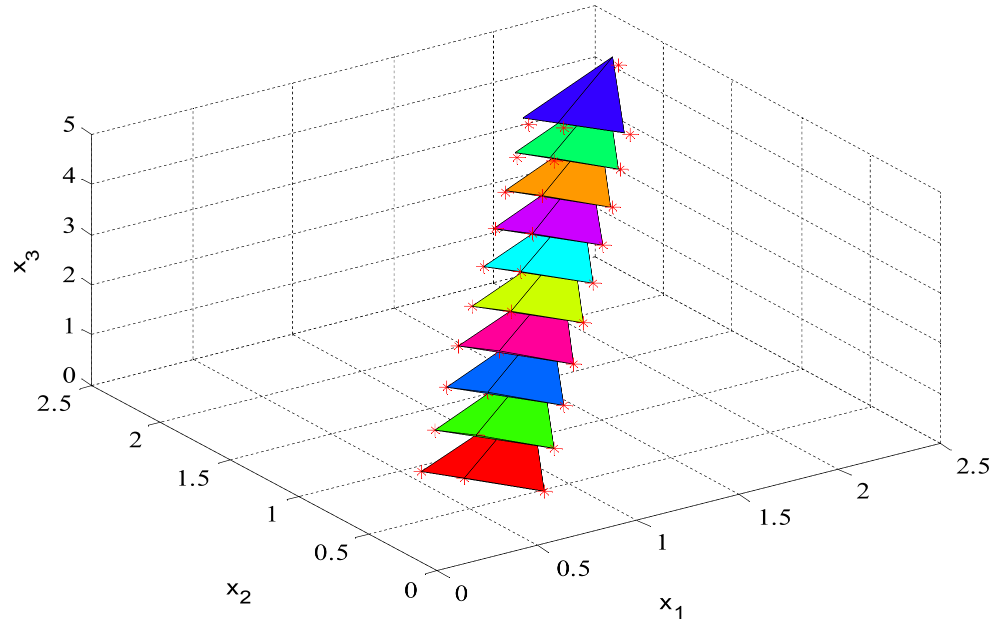


Figure 6.27 Vertices of the starting polytope is simulated using ode45. They differ from the vertices of the polytope as number of steps increases. This is for case dynamic partition.

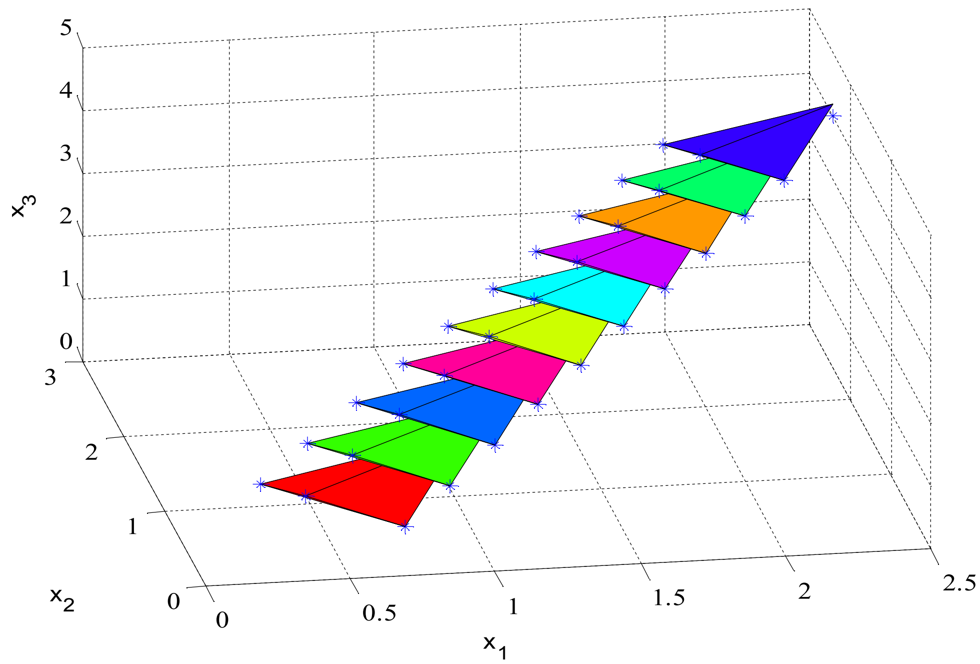


Figure 6.28 This is for case of fixed grid. System is characterized by piece-wise linear differential equations. Vertices do not differ. The difference at the top of the polytope and the point is due to crossing to new orthant.

6.8 A Method for Reachability Analysis Using Polytopes

A method that I used for reachability analysis is that, instead of linearizing and discretizing the middle point. Do these for every vertex, transform them independently, and take the convex hull of the vertices. This method does not work for every case theoretically and practically. But for the cases it works, it gives better results. See appendix for the discussion.

CHAPTER SEVEN
CONCLUSION

In this thesis, reachability analysis for nonlinear systems is examined. First, one dimensional case is handled, so that reachability analysis challenges are solved and these methods are applied to 3 dimensional case.

In one dimension, for a piece-wise linear system these were the challenges:

1- How to split polytopes around breakpoints?
   Polytopes at breakpoints need to be splitted into sub-polytopes. For this, the greatest value that is smaller than breakpoint value and the smallest value that is greater than breakpoint is determined according to our resolution area and well-chosen step size.

2- How to determine breakpoint resolution and step size?
   Breakpoint resolution is determined according to our polytopes size. This is rather an experimental and intuitive decision. It may change according to areas that reachability analysis is used. Resolution area cannot be greater than polytope size. Because polytopes that are smaller than breakpoint resolution will be lost or will lose relatively large amount of data. For these cases, step size may be set accordingly, so that polytopes can jump over the resolution area, but this time the accuracy of analysis will drop down.A trade-off decision need to be done here.

3- How to ease computing?
   Polytopes splitted around breakpoints may be huge. This makes the computing harder and slow. If any polytope is a sub-set of another polytope, than you don't need to run the smaller one. Or if any polytope is in the area of another polytope's past trajectory, then we already know that this polytope will follow the same trajectory. For this, a sub-program may be written. This sub program runs every time polytope set that will make one step, searching if any new

splitted polytopes fall into any past trajectory. This sub-program although may reduce the number of polytopes to be simulated, it slows the program. This sub-program may speed up, if the past trajectories of the polytopes to be searched are restricted to some number.

For a detailed analysis of system, these challenges need to be considered. In 3 dimensional case, if our system is fixed grid, than polytope splitting challenges need to be considered, for small step sizes, computation may be very hard due to increasing number of polytopes. And I discuss the subject that: For fixed grid, do we really have to split a polytope. In some cases no, as the case in chapter 6.

For dynamic partition and 3 dimensional cases, challenges are that: polytopes shouldn't be much greater relatively, because it will make the errors big. Dynamic partition is about linearizing the system at quiescent point. If polytope is relatively big, it will produce large errors at the vertices of the polytopes that are relatively away from the quiescent point.

For one dimension, the challenges of reachability are examined. Resolution area, the step size, these are all examined. These experiences will be used for 2 and higher dimension sets, after that we will be able to make reachability analysis for biological systems that has uncertain parameters or any nonlinear systems that needs to be checked for bad states robustness.

REFERENCES

Althoff, M. (2010). *Reachability Analysis of Nonlinear and Hybrid Systems using Zonotopes*, Retrieved February, 2012, from http:// cmacs .cs. cmu. edu/ seminars/slides/althoff.pdf

Aswani, A., & Tomlin, C.(2007). Reachability Algorithm for Biological Piecewise-affine Hybrid Systems. In A. Bomperad, A. Bicchi, G. Buttazzo (Eds). *HSCC* (633–636). Berlin: Springer.

Bemporad, A. (2010). *Automatic Control 1, Reachability Analysis,* Retrieved Februrary, 2012, from http:// cse.lab.imtlucca.it /~bemporad/ teaching/ ac/pdf/ 05a-reachability.pdf.

Dang, T. (2006). Approximate Reachability Computation for Polynomial Systems. In Hespanha, J. P., & Tiwari, A. (Eds.). *Hybrid Systems: Computation and Control*, (138-152). Berlin: Springer.

Dang,T., Le Guernic,C., & Maler, O. (2011). Computing Reachable States for Nonlinear Biological Models. *Theoretical Computer Science, 412*(21), 2095-2107

De Jong, H., Gouz'e , J., Hernandez, C., Page, M, Sari,T., Geiselmann, J. (2004) Qualitative Simulation of Genetic Regulatory Networks Using Piecewise-Linear Models, *Bulletin of Mathematical Biology* 66, 301–340.

Ghosh, R. and Tomlin, C. J. (2001). Lateral Inhibition Through Delta–Notch Signaling: A Piecewise Affine Hybrid Model. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli (Eds), *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science (232–246). Berlin: Springer.

Girard, A. (2005). Reachability of Uncertain Linear Systems using Zonotopes. In *Hybrid Systems: Control and Computation HSCC'05,* (291-305).Berlin: Springer.

Girard A., & Le Guernic, C. (2008). *Zonotope-hyperplane Intersection for Hybrid Systems Reachability Analysis*. In Egerstedt, M., Mishra, B. (Eds.). *HSCC,* (215–228) . Berlin: Springer.

Gouz´e, J. L. and Sari, T. (2002) A Class of Piecewise Linear Differential Equations Arising in Biological Models. *Dynamical Systems, 17*(4), 299–316.

De Jong, H. (2002). Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology,* 9, Number 1.

Kloetzer, M. and Belta, C.(2006). *Reachability Analysis of Multi-affine Systems*. In Hespanha, J. P., & Tiwari, A. (Eds.). *HSCC*, (348-362). Berlin: Springer.

K.W. Kohn. (2001) Molecular Interaction Maps as Information Organizers and Simulation Guides. *Chaos*, 11, 1–14 .

Le Guernic,C. (October 28, 2009).*Reachability Analysis of Hybrid Systems with Linear Continuous Dynamics*. Retrieved February 14, 2012, from http:// rillette.imag.fr/documents/ Le Guernic_present.pdf

Snoussi, E. H. (1989). Qualitative Dynamics of Piecewise-linear Differential Equations: A Discrete *Mapping Approach. Dynamics and Stability of Systems*. 4, 189–207.

APPENDICES

1- Piece-wise Linear System for Chapter 4

```
%A piece-wise linear system
hold on
t=-2:0.01:-0.5
y=-t-1;
plot(t,y,'lineWidth',2.2)
grid on


t=-0.5:0.01:0.5
y=t;
plot(t,y,'lineWidth',2.2)


t=0.5:0.01:2
y=-t+1;
plot(t,y,'lineWidth',2.2)

t=-2:0.01:2
y=0*t
plot(t,y,'k','lineWidth',2.2)
plot(y,t,'k','lineWidth',2.2)
```

2- Reachability Analysis of Piece-wise Linear Model for Chapter 4

```
 %% calisma 13 te iki sıra halinde tutacağız D leri. Dler de adımatacak
%% politoplar olacak, birinci sıra geçmişi tutacak, ikinci sıra ise, adım
%% atacak

% P1=[0.01 0.749];P2=[0.751 10];P3=[-0.01 -0.749];P4=[-0.751 -10];
% P1=polytope(P1');P2=polytope(P2');P3=polytope(P3');P4=polytope(P4');
% P_cell{1}=P1;P_cell{2}=P2;P_cell{3}=P3;P_cell{4}=P4;

clear
Res=0.0001
P3=[0+Res 0.5-Res ];P4=[0.500+Res 10];P2=[0-Res -0.5+Res];P1=[-0.500-Res -10];
P1=polytope(P1');P2=polytope(P2');P3=polytope(P3');P4=polytope(P4');
P_cell{1}=P1;P_cell{2}=P2;P_cell{3}=P3;P_cell{4}=P4;

D=[];T=0.5;U0=1;
DD=polytope([-0.4 -0.49]');a=1;
ext=extreme(DD);
```

```
plot(1,ext,'o');hold on
 D{1}=DD;
%  D{2,1}=DD;
%%
 S=parcala(D{1},P_cell);
   D=S;a=1;
   renk='rgbkm';
%%
for n=1:100
    leng=length(D);% D parçaladıktan sonra adım atanları tuttuğu için,
    %parçalanmış olanı değil, parçalandıktan sonra adım atanı tutar,
    %bu yüzden 2 yerine 3 polytope görebilirsin D nin içinde veya 4 yerine 6


    kopya_D=D;
       for k=1:leng


       ext= extreme(D{k});
       %if abs(ext(1)-ext(2))>10^-6 % end 67de.
            i=mod(a,500)+1;
%             history_D{i}=D{k};a=a+1;

    [A f]=ABCD_yeni(D{k},P1,P2,P3,P4);

    ext= A*ext+f;
    plot(n+1,ext,strcat( renk(1+mod(n,5)),'*'));
    %ykm=linspace(ext(1),ext(2),2);
    %plot(n+1,ykm)
    hold on
    R=polytope(ext);

    [S a2]=parcala2(R,P_cell);% R politop, Pcell politop cell i.
    %calisma15
    Rr{1}=R;%% adimatmış politoptur.adimatacak..fonksiyonunun çalışması için onu
celle çevirdim
   % if a2>=2
%%     index=adimatacak_history_D(kopya_D,S);
% index=adimatacak_history_D(Rr,S);
 %   S=S(index);
  %  end
      %
      %if isempty(S{:})==0% bunu sonra sil program hata veriyorsa
   if k==1 % burada k vardı, n olarak değişti.
          gecici_D=S;
   else
        gecici_D =[gecici_D S];
   end
```

```
  %end

 % end



end

D=gecici_D;
% burada kopya D ile D yi karşılaştırıp, yeni D yi tekrardan bulacağım.
%function adimatacakD

% index=adimatacakD(kopya_D,D); %% tek bir adım öncesinde yapılan kopyada
hiç içinde olan çıkmadı. fonksiyon gereksiz.
% %bunun için alternatif bir yol, bence round yapmak.
% D=D(index);

end
grid on
```

This m-file uses the functions below:

```
% this function checks for subset to ease computation.

function [A f]=ABCD_yeni(D_N,P1,P2,P3,P4)
e=extreme(D_N);T=0.01;
if isinside(P1,e(1))& isinside(P1,e(2))

A=1-T;%arasında ise
f=-T;


end


if isinside(P2,e(1))& isinside(P2,e(2))

A=1+T;
% B=-0.5625;
f=0;

end

if isinside(P3,e(1))& isinside(P3,e(2))
A=1+T;
f=0;
```

```
end


if isinside(P4,e(1))& isinside(P4,e(2))

A=1-T;
f=T;


end
```

```
 function  [ S a] = parcala2(D_N,P_cell)
a=1;
%S{1}=[];%% bunu sil, eğer program hata veriyorsa
%S parçalanmış politopu elinde tutar.

%  for j=1:ax
%    p_pol=polytope(p(j,:)');
   p_pol=D_N;
   for kk=1:4
        P_kes=p_pol&P_cell{kk};
        if isempty(extreme(P_kes))==0
%               P_gicik(a,:)=[ extreme(P_kes)'];a=a+1;
%               plot(k,extreme(P_kes)','b*')
              S{a}=P_kes;a=a+1;


%          plot(P_kes)
      end
    end
%    end
% a=a-1;
```

3- Graph of dT/dt vs. T in Lac-T for Chapter 5

```
close all
clear
critics(1)=fzero(@(x)250*(1+0.02*x^2)/(167+0.02*x^2)-x,0 );
critics(2)=fzero(@(x)250*(1+0.02*x^2)/(167+0.02*x^2)-x,50 );
critics(3)=fzero(@(x)250*(1+0.02*x^2)/(167+0.02*x^2)-x,150 );

K=167.1;
K1=0.02; %uM
X=0:0.1:250;
```

```
%X=0:0.01:0.02;
P=250;
f=P*(1+K1*X.^2)./(K+K1*X.^2)-X;
plot(X,f), hold on
grid on
plot(critics(1),0,'*');plot(critics(2),0,'*');plot(critics(3),0,'*');
```

4- Reachability Analysis for Lac-T Chapter 5

```
clear
P=260;K=167.1;K1=0.02; %uM
X=1;delta=0.1;
A = [1-delta];
X0=polytope([100;200]);R2=X0;U0 =250 +unitbox(1,10);

V=mean(extreme(X0));f_tek=extreme(X0);
for k=1:150
  X=mean(extreme(X0));
%  f=zeros(2,1);

  f(1) = delta*[P*(1+K1*X.^2)./(K+K1*X.^2)];
 %  f(2) = delta*[P*(1+K1*X.^2)./(K+K1*X.^2)];

%   X=mean(f_tek);
%    f_tek=(1-delta)*f_tek+delta*[P*(1+K1*X.^2)./(K+K1*X.^2)];
%  turev=2*K1*X*(K-1)/(K+K1*X^2)^2;f=delta*turev;
   X0=range(X0,A,f);
   plottekboyut(k,X0,'b*');
  %
%   struct=system_struct(X0);
%   R2=mpt_reachSets(struct,R2,U0,1);
%   plottekboyut(k,R2,'r*');
%
%  plot(k,f_tek,'g*')

end
```

5- Uncertain Parameter P for Chapter 5.

```
X0=polytope([50;140]);R2=X0;
U0 =250 +unitbox(1,10);
delta=0.1;
for k=1:150
```

```
        X=mean(extreme(X0));
        S=con_lineerF(X,delta);
        X0=mpt_reachSets(S,X0,U0,1);
        plottekboyut(k,X0,'r*')

    end
```

This m-file uses the functions below:

```
function [sysStruct ]=con_lineerF(point,delta)
P=250;K=167.1;K1=0.02;%K1=9;
A=[-1];Tk=point;
B=(1+K1*Tk.^2)./(K+K1*Tk.^2);
C=eye(1);
D=0;

syst=ss(A,B,C,D);
sysStruct=mpt_sys(syst,delta);
sysStruct.ymax = 500; sysStruct.ymin = 200;
sysStruct.umax = 500; sysStruct.umin =200;


function plottekboyut(k,X0,str)
 x=extreme(X0);
 plot(k,x,str),hold on
```

6- Dividing the Space into Orthants

```
function P=polytopgrid();
vertex=[0 0 0; 4 0 0; 4 4 0; 0 4 0; 0 0 4; 4 0 4; 4 4 4; 0 4 4];

% P(1)=polytope([0 0 0; 4 0 0; 4 4 0; 0 4 0; 0 0 4; 4 0 4; 4 4 4; 0 4 4]);
% P(2)=polytope([4 0 0; 8 0 0; 8 4 0 ; 4 4 0; 4 0 4; 8 0 4; 8 4 4; 4 4 4]);
% P(3)=polytope([0 0 4; 4 0 4; 4 4 4; 0 4 4; 0 0 8; 4 0 8; 4 4 8; 0 4 8]);
% P(4)=polytope([4 0 4; 8 0 4; 8 4 4; 4 4 4; 4 0 8; 8 0 8; 8 4 8; 4 4 8]);

matrix=[0  0 0; 4 0 0; 0 4 0;0 0 4; 4 4 4; 4 4 0 ;0 4 4 ; 4 0 4];
matrix2=[ 0 8 0; 4 8 4; 4 8 0 ;0 8 4 ;];
matrix3=[ 8 0 0; 8 4 4; 8 4 0 ; 8 0 4];
matrix4=[0 0 8; 4 4 8 ;0 4 8 ; 4 0 8];
matrix5=[ 8 8 4; 8 8 0 ];%x y
matrix6=[4  8 8;0 8 8 ];%y z
matrix7=[8 4 8;8 0 8];
matrix8=[8 8 8];
matrix=[matrix;matrix2;matrix3;matrix4;matrix5;matrix6;matrix7;matrix8]
kk=1;
```

```
for m=1:length(matrix)
vertex=[0 0 0; 4 0 0; 4 4 0; 0 4 0; 0 0 4; 4 0 4; 4 4 4; 0 4 4];

for k=1:length(vertex)

   vertex(k,:)=matrix(m,:)+ vertex(k,:);

end
P(kk)=polytope(vertex);
kk=kk+1;
end
% plot(P)
```

7- **Simulation of one point Using ode45 and Using either hill or signum function as threshold functions.**

```
function tr=tekNokta(nokta,t,str)
disp('str hill veya hill2 olabilir')
disp('satır vektörü olmalıdır, nokta.')


%for k=1:kk
if strcmp(str,'signum' )

    [t y]=ode113(@lacodesignum,[0 t],[nokta]);
   plot3(y(:,1),y(:,2),y(:,3))
end
if strcmp( str,'hill' )
   [t y]=ode113(@lacode,[0 t],[nokta]);
     plot3(y(:,1),y(:,2),y(:,3),'r')

end
if strcmp( str,'global' )
   [t y]=ode113(@lacodeGlobal,[0 t],[nokta]);
   plot3(y(:,1),y(:,2),y(:,3))

end



%end
tr=y;
hold on
```

this function uses the functions below:

```
 function dy=lacodesignum(t,y)
```

```
% mu=1;
% dy=zeros(2,1);
% dy(1)=y(2);
% dy(2)=-y(1)+mu*(1-y(1).^2).*y(2);
teta21=4;
teta31=4;teta32=4;
teta12=8;teta11=4;
g1=2;g2=2;g3=2;
k1=20;k2=20;k3=20;k4=20;
dy=zeros(3,1);


 dy(1)=k1*(hill2(y(2),teta21))-g1*y(1);
 dy(2)=k2*(1-hill2(y(1),teta11)*hill2(y(3),teta31))-g2*y(2);
 dy(3)=k3*(1-hill2(y(1),teta12))+k4*(1-hill2(y(3),teta32))-g3*y(3);



function dy=lacode(t,y)
% mu=1;
% dy=zeros(2,1);
% dy(1)=y(2);
% dy(2)=-y(1)+mu*(1-y(1).^2).*y(2);
teta21=4;
teta31=4;teta32=4;
teta12=8;teta11=4;
g1=2;g2=2;g3=2;
k1=20;k2=20;k3=20;k4=20;
dy=zeros(3,1);

 dy(1)=k1*(hill(y(2),teta21))-g1*y(1);
 dy(2)=k2*(1-hill(y(1),teta11)*hill(y(3),teta31))-g2*y(2);
 dy(3)=k3*(1-hill(y(1),teta12))+k4*(1-hill(y(3),teta32))-g3*y(3);
 %bu bölüm deneme amaçlı konulmuştur.
% x1=y;
% B=[  k1*(1-hill2(x1(2),teta21));k2*(1-hill2(x1(1),teta11)*hill2(x1(3),teta31));...
% k3*(1-hill2(x1(1),teta12))+k4*(1-hill2(x1(3),teta32))]

function dy=lacodeGlobal(t,y)
% mu=1;
% dy=zeros(2,1);
% dy(1)=y(2);
% dy(2)=-y(1)+mu*(1-y(1).^2).*y(2);
teta21=4;
teta31=4;teta32=8;
teta12=8;teta11=4;
g1=2;g2=2;g3=2;
k1=20;k2=20;k3=20;k4=20;
```

```
dy=zeros(3,1);
%
% dy(1)=k1*(hill(y(2),teta21))-g1*y(1);
%  dy(2)=k2*(1-hill(y(1),teta11)*hill(y(3),teta31))-g2*y(2);
%  dy(3)=k3*(1-hill(y(1),teta12))+k4*(1-hill(y(3),teta32))-g3*y(3);
global B;global A;
B
A
 dy=A*y+B;
```

8- Reachaiblity Analysis of Fixed grid partition for Chapter 6.

```
clear
close all
d=0.01;
A=[-2 0 0; 0 -2 0; 0 0 -2];
A_=(eye(3)+d*A);% A yı discretize et,
B=[0 20 40]';%
f=d*B;% B yi discretize et.
V=rand(4,3);
P=polytope(V);
teknokta(mean(V),1,'signum');% orta noktayı, 1 sn süresince simule et.
Pgrid=polytopgrid;
    P=polytope_bol(P,Pgrid);%polytope is divided if necessary.
    Pset=P;%P may be polytope array;polytopes that will be evaluated at each step.
    % Pset,:
    Pall=[];
for k=1:60
    Pset_kopya=Pset;
    Pset=[];
    Pall=[Pall Pset];
    %aşağıdaki for, Pset te adım atacak olan politoplara adım attırıp,
    % ardından, polytope_bol fonksiyonu ile bölünmesi gereken politop
    % varsa,böldürür.
   for kk=1:length(Pset_kopya)

         f=d*B_signum(mean(extreme(Pset_kopya(kk))));% A hiçbirzaman
değişmezken,
         %f her seferinde değişir. orta noktaya ait olan f yi bulur.

         Pset_kopya(kk+1)=range(Pset_kopya(kk),A_,f);
         [PP KK]=polytope_bol(Pset_kopya(kk+1),Pgrid);

         Pset=[ Pset PP ];

   end
```

```
    Pall=[Pall Pset];

    plot(Pset)

end
plot(P,'w')
figure,plot(Pall)
```

9- Dynamic Partition for Chapter 6.6

```
clear
close
d=0.01;
A=[-2 0 0; 0 -2 0; 0 0 -2];
A_=(eye(3)+d*A);
B=[0 20 40]';
f=d*B;
V=rand(4,3);
P=polytope(V)
teknokta(mean(V),1,'hill')
Pgrid=polytopgrid;
%     P=polytope_bol(P,Pgrid);%polytope is divided if necessary.
%     Pset=P;%P may be polytope array;polytopes that will be evaluated at each step.
    Pall=[];
for k=1:100
    Pk=P;
    Pall=[Pall Pk]
%    for kk=1:length(Pset_kopya)

         f=d*B_hill(mean(extreme(Pk)));

         P=range(Pk,A_,f);

%          Pset=[ Pset polytope_bol(Pset_kopya(kk+1),Pgrid)];

%        Pset=[ Pset Pset_kopya(kk+1)];

%    end
    Pall=[Pall P]

%     plot(Pset)

   end
% plot(P,'w')
plot(Pall,'w')
```

10- Fixed Grid without Splitting

```
clear
close all
d=0.01;
A=[-2 0 0; 0 -2 0; 0 0 -2];
A_=(eye(3)+d*A);% A yı discretize et,
B=[0 20 40]';%
f=d*B;% B yi discretize et.
V=rand(4,3);
P=polytope(V);
teknokta(mean(V),1,'signum');% orta noktayı, 1 sn süresince simule et.
Pgrid=polytopgrid;
   P=polytope_bol(P,Pgrid);%polytope is divided if necessary.
   Pset=P;%P may be polytope array;polytopes that will be evaluated at each step.
   % Pset,:
   Pall=[];
for k=1:70
   Pset_kopya=Pset;
   Pset=[];
   Pall=[Pall Pset];
   %aşağıdaki for, Pset te adım atacak olan politoplara adım attırıp,
   % ardından, polytope_bol fonksiyonu ile bölünmesi gereken politop
   % varsa,böldürür.
  for kk=1:length(Pset_kopya)

       f=d*B_signum(mean(extreme(Pset_kopya(kk))));% A hiçbirzaman
değişmezken,
       %f her seferinde değişir. orta noktaya ait olan f yi bulur.

       Pset_kopya(kk+1)=range(Pset_kopya(kk),A_,f);
    %    [PP KK]=polytope_bol(Pset_kopya(kk+1),Pgrid);

     %   Pset=[ Pset PP];

Pset=Pset_kopya(kk+1);

  end
   Pall=[Pall Pset];

  % plot(Pset)

 end
plot(P,'w')
figure,plot(Pall)
```

11- Dynamic Partition Case Linearization at the Vertices and Taking Convex Hull Method

I must say that this is rather a practical approach than theoretical approach. Suppose you want to transform a polytope for one step, you first choose the middle point of that polytope, linearize it there, and transform it into an other polytope. But, there is another case, for dynamic partition, if the polytope doesnot include any unstable equilibrium point, take the vertices of the polytope as points, and simulate the points for one step, then take the convex hull of the new polytope. This will produce better results. In first approach, vertices are transformed with the linearization equation that belongs to the middle point. This will produce errors. But second approach, vertices are taken as points and simulated. Every point is transformed into new point and these new points are the vertices of the resultant polytope. There is no theory saying that you can do this. In fact, this doesnot aggree with polytope computation. But it gives better results. Why? I don't know the exact answer theoretically but intuitively speaking: suppose that two points are mapped into new 2 points. And we know that every point between these 2 points willl map into the region between those new 2 points for affine systems or if you can prove this for another system, it will work for it too. And now you take a pentagon as polytope. Every point between any vertex will map into the region of the new vertices of the polytope. this works for every point in the polytope. So there is no drawback using this method. Or you can think the vertices as the middle points of another polytopes. P0 is the polytope to be studied that is inside the confusing shape. The other polytopes P1, P2, P3 and P4. Now, you linearize P1,P2,P3 and P4 and transform them then you are taking a subset of union of these polytopes from the points that are the range of the vertices of P0. And this method gives better results.