DYNAMIC SIMULATION TOOLS FOR MAIN METABOLIC PATHWAY OF *ESCHERICHIA COLI*

HAZWAN ARIF BIN MAZLAN

THESIS SUBMITTED IN FULFILMENT OF THE DEGREE IN COMPUTER SCIENCE (GRAPHICS AND MULTIMEDIA TECHNOLOGIES) WITH HONOURS

FACULTY OF COMPUTER SYSTEM & SOFTWARE ENGINEERING, UNIVERSITI MALAYSIA PAHANG

DECEMBER 2013

STUDENT'S DECLARATION

"I declare that this thesis entitled "**Dynamic Simulation Tools for Main Metabolic Pathway of *Escherichia coli***" is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree."

Signature : ……..

Student Name : HAZWAN ARIF BIN MAZLAN

Student Number : CD 10001

Date : 7th DECEMBER 2013

SUPERVISOR'S DECLARATION

"I hereby declare that I have read this thesis and in my opinion, this thesis is sufficient in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Graphics And Multimedia Technologies) With Honours"

Signature : ……..

Supervisor's Name : DR. TUTY ASMAWATY BINTI ABDUL KADIR

Date : ……..

DEDICATION

"In the name of Allah, the most Gracious, the most Merciful"

I dedicate my dissertation work to my family, lecturers and friends.

The most gratitude feeling to my lovely parents,
Mazlan Bin Mat Sahad & Siti Balkhis, whose words of encouragement for me to finish my thesis and always supported me in everything.

To my Supervisor,
Dr. Tuty Asmawaty Binti Abdul Kadir

To my Academic Advisor,
Mr. Aziman Bin Abdullah & Dr. Abdulrahman Ahmed Mohammed Al-Sewari

To my best friends,
Thanukphong, Sabri, Amirul, Azrulnizam, Nazar, Nor'atikah & Shin Yee

To all FSKKP's Staffs and Lecturers,

To all 3BCG2010,
Thank you for your invaluable supports. May God Bless You All and
THANK YOU FOR EVERYTHING!

ACKNOWLEDGEMENT

First and foremost, Alhamdulillah, I would like to express my gratitude to Allah SWT for giving me an opportunity to finish my undergraduate project in great condition. I would like to thank my parents for supporting me in my journey of studies. Without their support, I won't be able to finish this Undergraduate Project on time.

Besides, I would like to thank my supervisor, Dr. Tuty Asmawaty Binti Abdul Kadir, for helping me out as well as giving me the moral supports throughout the development of this project. I also would like to dedicate my appreciation to my best friends, friends and course mate whose are also helping me out by giving the moral supports, valuable opinions and sharing ideas during the progress of this project. You guys are awesome!

Last but not least, I would like to thank everyone that is also helping me out.

May Allah bless all of you.

THANK YOU SO MUCH

ABSTRACT

Metabolic engineering is the study of metabolic pathways modification in one organism by increasing the metabolite production through genetic engineering in order to bring the new and better product in food, feed, pharmaceuticals and many more. The studies of metabolic pathway give opportunity to scientists to discover a new phenomenon in cells and proposed many mathematical models for each of the cells behavior. Most of the cells used in experiments are *Escherichia coli* (*E. coli*). However, conducting experiments of *E. coli* are time and money consuming and scientists found it the existing dynamic simulators are complex in order to simulate the proposed mathematical model and some of them prefer to do hardcode simulation. These problems give opportunity to computer scientist to develop the dynamic simulation tools for main metabolic pathway of *E. coli* that are simpler than the existing tools. The development tools used for the project are Visual Studio 2010. The dynamic simulation tools contain of 2 main modules – Metabolites configurations and Graph configurations. User needs to undergo four (4) tab pages in Metabolites Configurations module and two (2) tab pages in Graph Configurations. The technique used in this project is Ordinary Differential Equations (ODE) of Runge-Kutta.

ABSTRAK

Kejuruteraan metabolik adalah kajian laluan pengubahsuaian metabolik dalam satu organisma bagi meningkatkan pengeluaran metabolit melalui kejuruteraan genetik dalam usaha menghasilkan produk baru dan lebih baik dalam makanan, makanan haiwan, farmaseutikal, nutraseutikal dan banyak lagi. Kajian laluan metabolik memberi peluang kepada saintis menemui fenomena baru dalam sel-sel dan banyak model matematik telah dihasilkan bagi setiap tingkah laku sel-sel. Kebanyakan sel-sel yang digunakan dalam eksperimen ialah *Escherichia coli* (*E. coli*). Walau bagaimanapun, menjalankan eksperimen *E. coli* memakan banyak masa dan wang dan bagi mensimulasikan model matematik pula, saintis mendapati simulator yang sedia ada terlalu kompleks dan sesetengah daripada mereka lebih suka untuk melakukan simulasi dari segi hardcode. Masalah-masalah ini memberi peluang untuk menghasilkan alat simulasi yang dinamik untuk laluan utama metabolik *E. coli* yang lebih mudah daripada alat-alat yang sedia ada. Pembangunan alat yang digunakan untuk projek ini adalah Visual Studio 2010. Aplikasi simulasi dinamik ini mengandungi dua (2) modul penting iaitu 'Metabolites Configurations' dan 'Graph Configurations'. Pengguna dikehendaki melalui empat (4) permukaan aplikasi di dalam 'Metabolite Configuration' dan dua (2) permukaan aplikasi di dalam 'Graph Configurations'. Teknik yang digunakan untuk penghasilan simulasi ialah teknik persamaan pembezaan Runge-Kutta.

CHAPTER 1

INTRODUCTION

1.1 Introduction

The technologies of bioprocess that are specialize in Biotechnology, Chemical Engineering and Agricultural Engineering have play the important role in expanding the knowledge of living systems and bring new or better products in food, feed, pharmaceuticals and many more. One of the examples of the bioprocess technology is metabolic engineering. Metabolic engineering is the modification of metabolic pathways in one organism for understanding the purpose and utilizes their pathways for chemical transformation and energy transduction (Lessard, 1996) in order to produce a lot of beneficial for mankind. It is also a distinct subfield of genetic engineering as the scientists are practicing the modification of genetic within the cell or microorganisms according to cell behavior which related to the series of biochemical reaction and enzyme within a cell.

As one organism consists of various pathways in metabolic network, it may give an opportunity to scientists to discover a new phenomenon of organisms that can improve human life. Many mathematical models have been proposed to study the metabolic pathway but yet there is still more undiscovered behavior of the cell. This gives an opportunity to mathematician

and computer scientist to propose and create a new model and simulate the behavior of engineered cells (Abdul Kadir, 2010).

Simulation gives many advantages and it is practical in engineering field as the experiments on the real systems are quite dangerous and consuming time and money. From simulations, engineers or scientists can save lots of time and money from doing the exhaustive experiments. Moreover, it is also a great chance to the computer scientists to develop the simulation tools that maximize the requirement needs. Therefore, this thesis is about the development of simulation tools, focusing on metabolic pathway of microorganisms – *Escherichia coli*

1.2 Problem Statement

Dynamic simulation in metabolic engineering gives advantages in study the behavior of cells respective to timeline. Since bacteria grow fast in one culture, it takes time to extract the behavior of the bacteria one by one and yet one must conduct the experiment again in order to get desired result. Besides, conducting a lot of experiments consume a lot of money.

Nowadays, there are a lot of simulation tools and applications are available in market. As the simulation application has been introduced, most of scientists found it quite difficult to use (Osman Balci, 2001) due to complexity of the modeling and simulation (M&S) applications in the technical part. It requires expertise of self-explore to use advance simulation tools. So, there are two approach that scientist used in simulate their model. The first technique is manual simulation (Abdul Kadir, 2010) and another one is the used the available simulation tools.

For manual simulation, scientists normally prefer to do hard code in well-known available tools like matrix laboratory (MATLAB). If they want to change the mathematical model, they need to change the programming code according to respective rules. It consumes time and risk during the hardcode-changing-process as the one may accidentally delete certain part of codes as mathematical models in biology computing is way too complex.

As for available simulation tools, scientists need to study the features available before using it. Some of the features are complex and scientists might found it difficult to use. Therefore, it is a great chance to develop simplifies simulation tools that simulate the model as well as verifying experimental data with simulation data.

1.3 Objectives

The objectives of the project are: -

1. To identify the general dynamic model process of the main metabolic pathway of *Escherichia coli*

2. To design the dynamic simulation tools for *Escherichia coli* main metabolic pathway.

3. To develop the prototype for dynamic simulation tools by implementing ordinary differential equation (ODE) of Runge-Kutta

4. To validate the functionality of the dynamic simulation tools.

1.4 Scopes

The scopes of this project are divided into three (3) categories – data scope, platform and functionality of the system tools. In data scope, the case study is based on Abdul Kadir, 2010 thesis. For the first prototype, user can input minimum two (2) metabolites and maximum (9) metabolites and follow by inputting kinetic equations which amounts are less than one metabolites. For example, if user wants to input two metabolites, the amount of kinetic equations will be one. For data simulation, this simulation tools are using Runge-Kutta method (ODE).

For platform scope, at the meantime, this simulation tools are developed in Windows based platform system using C# language and .NET framework of Visual Studio 2010 integrated with MATLAB R2013a.

Besides that, tools' functionalities are divided into two categories. The first part is on the metabolites configuration while the second part is on the graph configurations. In metabolites configurations, the user need to undergo four (4) tab pages – Metabolites, Dynamic Equations, Kinetic Equations and Kinetic Parameters in order to configure the metabolites. The last tab will be the summary of the user metabolite configurations. In graph configurations, the user need to undergo one (1) tab page which is about displaying the graphs. The details of design can be found in chapter 4 of this thesis.

1.5 Thesis Organization

This thesis consists of six (6) chapters – chapter 1, introduction; chapter 2, literature review; chapter 3, methodology; chapter 4, design; chapter 5, implementation; chapter 6, result and discussion and last but not least, chapter 7, the conclusion chapter. The introduction chapter is mainly about the dynamic simulation tools overview. It is brief explanation about the metabolic engineering and how it is related to mathematician with M&S and computer scientist. Chapter 2 is about the detail of metabolic pathway model used in tools development and technique that being used. This chapter is also about the study of existing system that available in market. Overall framework discussion on the development of dynamic simulation tools are discussed in chapter 3, the methodology chapter. Meanwhile, chapter 4 is mainly about designing the system and interface. Moreover, chapter is a discussion on a framework development where the main metabolic pathway model is implemented in algorithm and been developed in Microsoft Visual Studio 2010 integrated development environment (IDE). Moreover, chapter 6, the result and discussions chapter is the discussion chapter based on tools development. Lastly, chapter 7 (conclusion chapter) is the overall project discussion.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter is about the overview of metabolic engineering, modelling and simulation, dynamic equations, existing tools available in market, numerical method of ODE and any domain of studies that are related to this thesis. The main purpose of this chapter is to increase the knowledge and understanding about the background of this project.

2.2 Metabolic engineering and its advantages

Metabolic engineering is the systematic analysis and modification of subjects' metabolic pathway using molecular biological techniques in order to improve their cellular properties (Koffas M. et al., 1999). This field focuses in systems biology, biochemical engineering, genetic engineering, applied microbiology and cell physiology fields. Metabolic engineering promising a better improvement in many fields and it is practically importance as it improves mankind life. For example, we usually see the blue roses in cartoons and we are very keen to have it in real life. More than 200 years, scientists are doing research, trial by error method, or any related inheritance techniques to get the desire result. As times goes by, the technology of Deoxyribonucleic Acid (DNA) recombinant has brought the fairy tales come to live (Sang Yup Lee and Papoutsakis, 1999).

According to James E. Bailey, Ph.D., more chemically complexes are implemented as industries keep moving drastically for better production of mankind. One of the main thing need to be concern is environment and metabolic engineering can undoubtedly eliminate the several central limitations in order to reach the aim. This can be proven as there is the availability of microbial biodegradation (Jianlin Xu et al., 2003) which is for clean up contaminated environments.

Wurtzel and Erich Grotewold (2006) stated that there are a lot of benefits if we know the correct way to manipulate the metabolic pathway of the plant in order to increase the desired plants or even undesired products. By identified mutant gene of respective enzyme, one can eliminate the specific enzymes activity in metabolic pathway. In culture the plant cells, it is one of way to growing differentiated plant tissues if transform cell lines with genes encoding regulatory factors in order to modify the metabolic pathway.

Mankind genome is like blueprint (Pedro Romero et al., 2004) that consists of complex process. In order to study and understand the process, it is recommend defining biochemical transformation regulated sequences or in other word to assign enzyme production to metabolic pathway. It will then trigger missing enzymes within pathway after being validate and yet, they predicted that they can find the missing enzymes within the human body. Moreover, applying metabolomics in metabolic pathway can also reveals alteration of lipid and amino acid catabolism of HIV-infected person (Edana Cassol et al., 2011).

2.3 Modelling and Simulation in metabolic engineering

In general, modeling is representative of one model that similar to real system while simulation is the operation of one system. The practice used of modeling and simulation in engineering is quite well known. Meanwhile, dynamic simulation is better known as

experimentations with model based on interest timeline in order to study the behaviour of the one system.

Before simulate the system, scientists and engineers must define general-purpose models that abstract the important behavior of one constraint in real world and usually it is represented in mathematical models. In metabolic engineering, the mathematical models usually described in Ordinary Differential Equations (ODE) or Partial Differential Equation (PDE) and it required numerical methods in order to solve the equations. A lot of simulation runs can increase the possibility to get the desired result. Therefore, it is important to consider the parameters, dynamic equations and fast and user-friendly computer programs (Korn, 2011) as the requirements.

Simulation is the safer way to experiment with design and operation of projects compared to real-world experiments and it is cheaper. In addition, it allowed to manipulate the models and validated it later on by real-world tests. According to Abdul Kadir (2010), dynamic simulation helps determine the undiscovered phenomenon by checking the metabolism of specific gene knockout with helps of mathematical model. Besides, it helps in checking the desired characteristics in cells without conducting lots of trial by error experiments in order to find the potential cells candidates that will be verified with experiment in real world later on.

There are many existing tools available to simulate the metabolic pathway of one subject. The example of tools are CADLIVE dynamic simulator (Kurata et al, 2005), CellDesigner (Funahashi et al, 2008), E-Cell (Tomita et al., 1999), E-Cell 3D and MATLAB.

2.3.1 CADLIVE dynamic simulator

Computer-Aided Design of LIVing systEms (CADLIVE) is a dynamic simulator developed by Professor Hiroyuki Kurata with staffs at Mitsui Knowledge Industry Co., Ltd. in 2005 that used rule-based automatic way to convert biochemical network map into dynamic models using Graphic User Interface (GUI) in order to simplify the details of exact kinetic parameters. According to Kurata (2003), the tools used regulator-reaction equation to show the relationship between regulators and their regulated reactions. It is available for windows based Internet Explorer platform as what being mention in the simulator website.

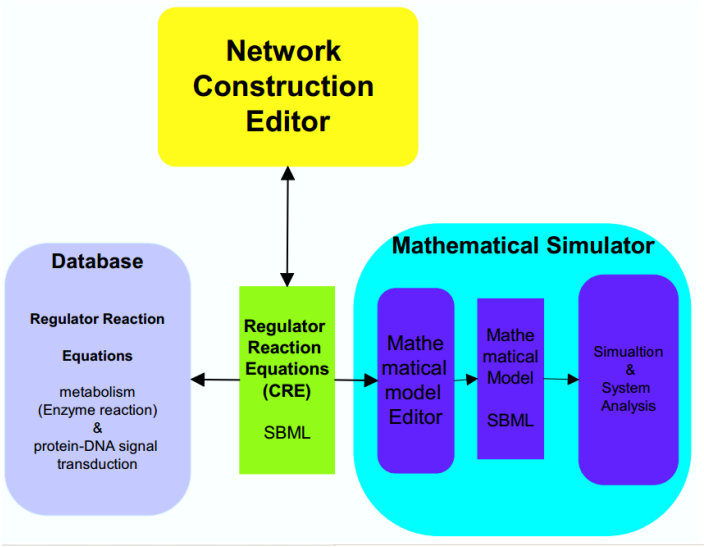


Figure 2.1.1: The process flow of the conversion of a biochemical network map to mathematical equations in CADLIVE Simulator.

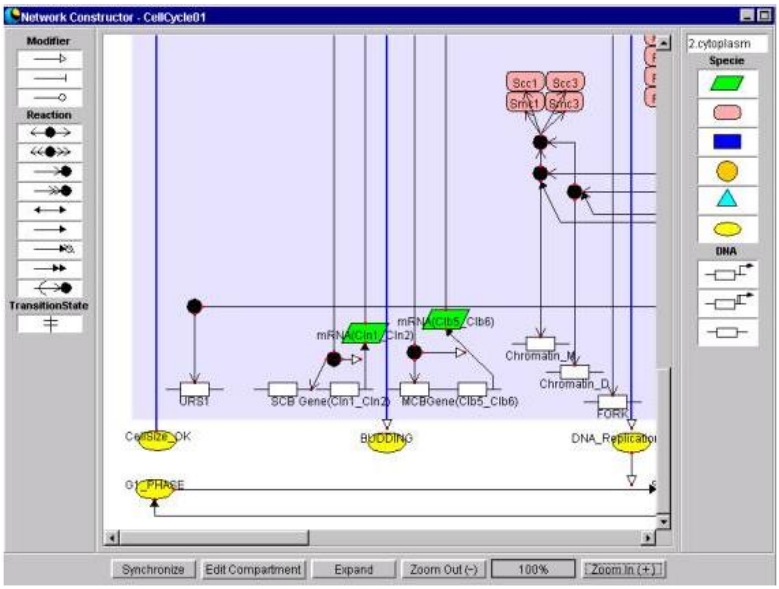


Figure 2.1.2: Screenshot of network construction of biochemical network using CADLIVE Simulator

Features that available in CADLIVE are as following:-

a) User do not have to go detail on exact biochemical reaction
b) It use a powerful control engineering method
c) The kinetic parameters is generated by genetic algorithm
d) Efficiency and practically used right rules to integrate
e) Convert mathematical model efficiency and accurately.
f) Analyse the mathematical model with simulation tools IDE

2.3.2 CellDesigner

CellDesigner is an advance graphical model representation with simple user-interface that describe biochemical and gene regulatory network. Kitano (2008) stated that one can convert the biochemical network drawing based on process diagram with graphical notation and stored it using System Biology Markup Language (SBML). One of the good thing about this tools is that it can be linked with simulation by integrate with SBML ODE Solver and Copasi and other analysis packages via Systems Biology Workbench (SBW). Moreover, this

system can modify the existing SBML model as well as use the existing model in database provided.

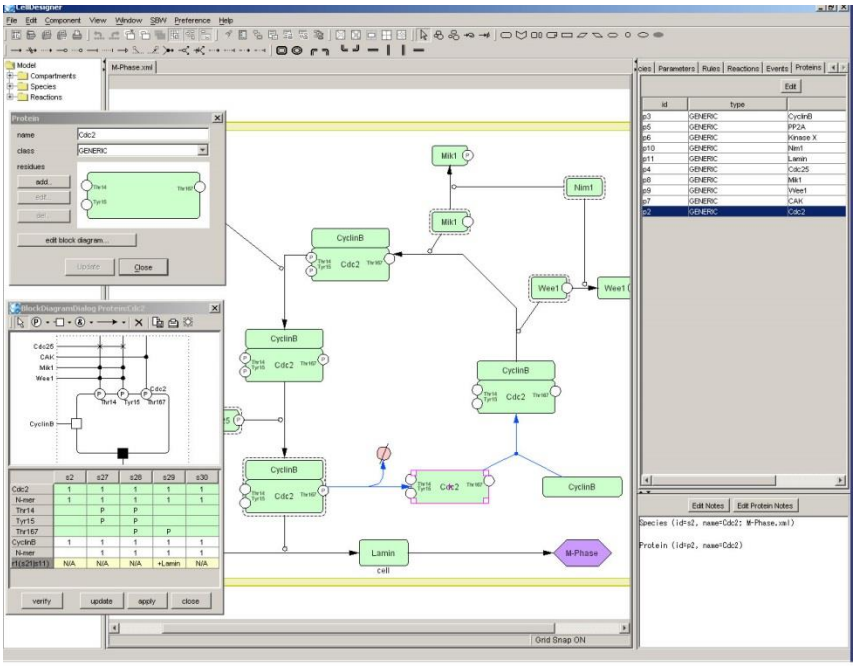


Figure 2.2: CellDesigner screenshot

Features available in CellDesigner:-

a) Used SBML as native file format
b) Graphical notation or mathematical model to represent biochemical network
c) It can integrate with simulation tools and existing resources

2.3.3 E-Cell

E-Cell is created by a team of Laboratory for Biochemical Simulation, Riken Quantitative Biology Center (QBiC) and Institute for Advanced Biosciences, Keio University of Japan. The purpose of the project is to make cell simulation at the molecular level precisely. Unique molecular mass information of wide range organisms' model from the genome sequencing projects and further systematic functional analyses of complete genes

gives advantages to team to build model for simulating the intracellular molecular process in order to predict the behaviour of living organisms dynamically (Tomita et al., 1999). User needs to define the parameter of extracellular as set rules of reaction, and E-Cell will simulate using numerical method of ODE. The graphical view then allowed the user to observe the dynamic changes in proteins concentration.

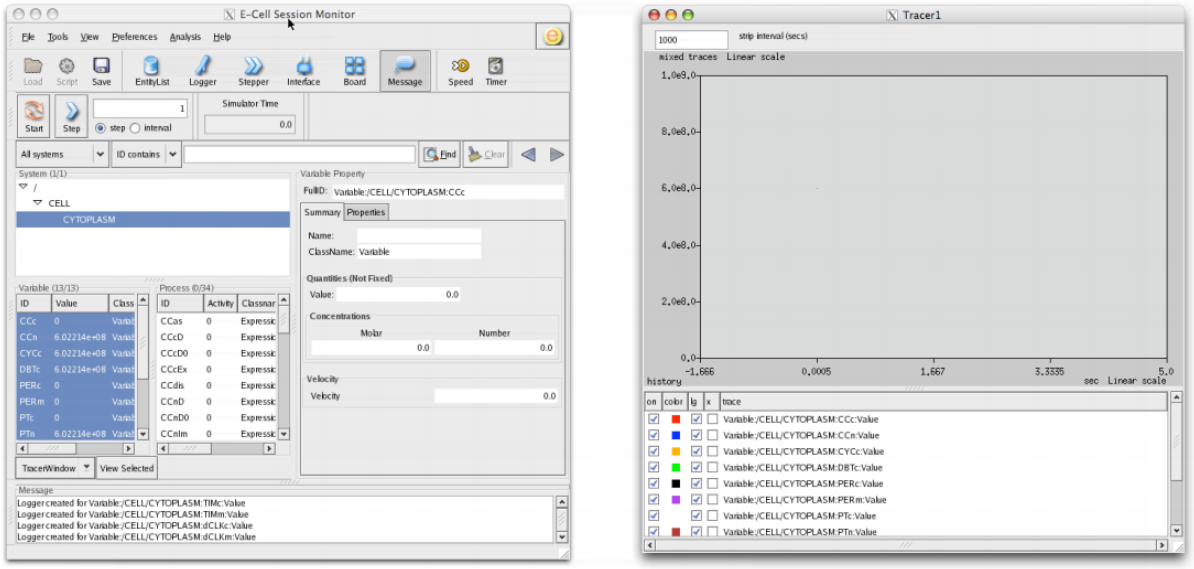


Figure 2.3: E-Cell Simulator Screenshot

2.3.4 E-Cell 3D

E-Cell 3D is a simulation tools that used a new way for better understanding of dynamic behaviour of complex cell system using cutting-edge 3D graphics. The used of quartz composer allows 3D graphic compositions prototyping rapidly via visual programming environment. E-Cell 3D visualization is converted from model stored in SBML and then it automatically laying out the 3D space. The simulation tools are still in demo and only available in Mac OS X.

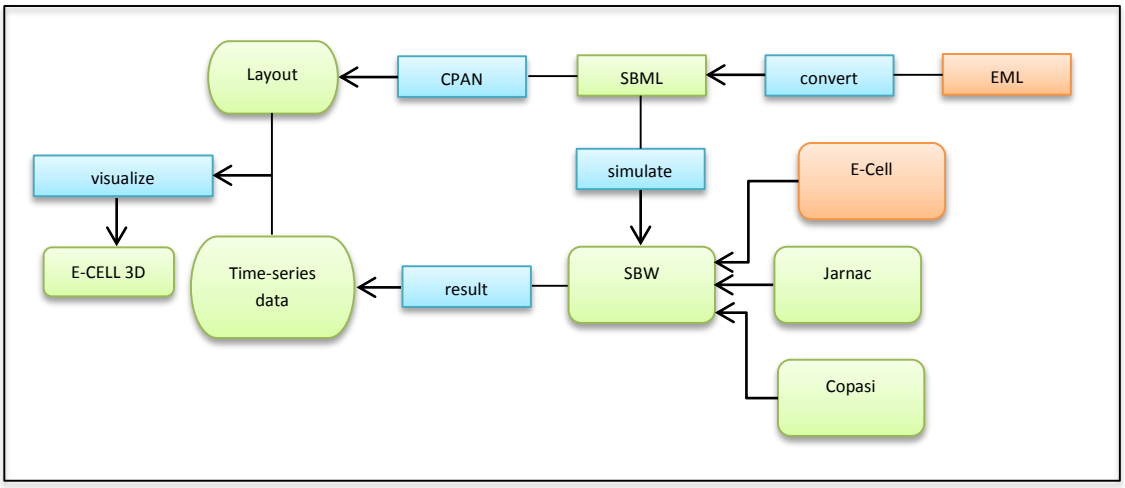


Figure 2.4.1: Process flow of E-Cell 3D

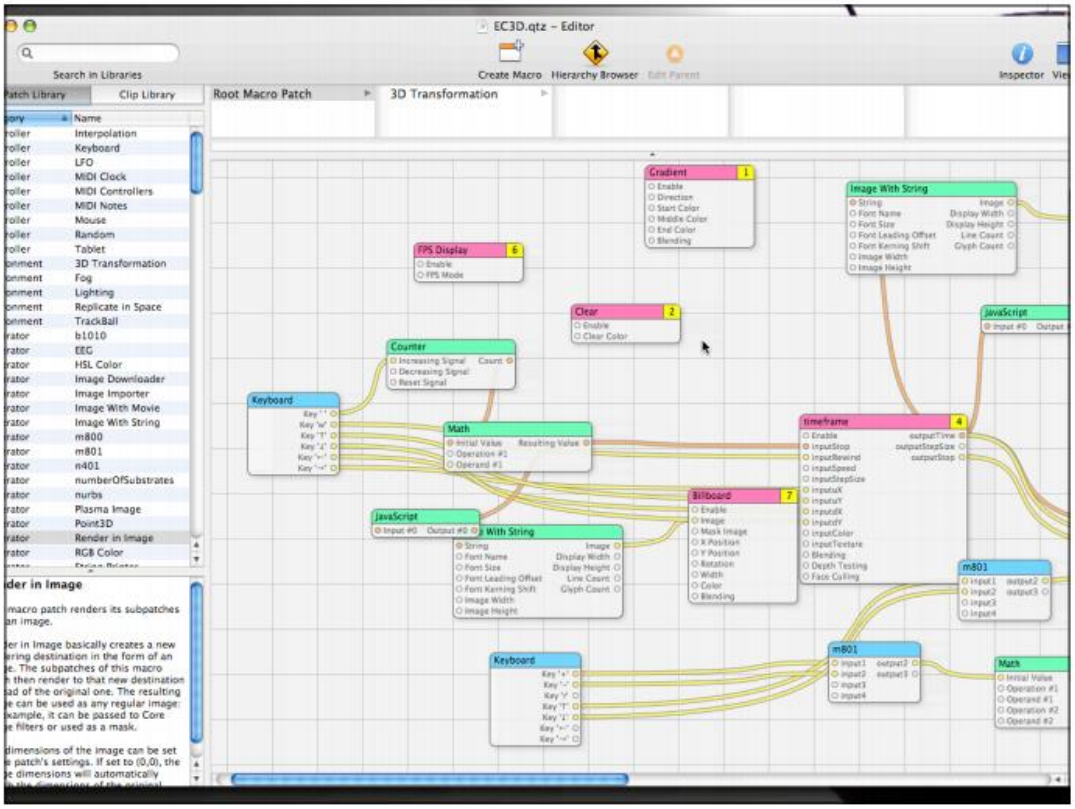


Figure 2.4.2: E-Cell 3D Screenshot

2.3.5 Summary comparison of modeling simulation tools

This section is about the summary of the existing tools that available in market (refer Fig. 2.1)

DIFFERENCES	CADLIVE	CELLDESIGNER	E-CELL	E-CELL3D
Platform available	Web-based for windows	Linux, Mac OS X and Windows	Windows	Mac OS X
Status	Well known	Well known	Well known	Demo
Complexity	Yes (refer section 2.3.1)	Yes (refer section 2.3.2)	Yes (refer section 2.3.3)	Yes (refer section 2.3.4)
Available of tutorial	Yes	Yes	Yes	Yes
Database	SBML	SBML	Not mention	SBML
Techniques	Integration rule based	Graphical notation	Numerical method ODE	Not mention
Visualization	Line graph	Line graph	Line graph	3D visualization
Comparison with experimental data	No	No	No	No

Table 2.1: Summary of the existing tools

2.4 Fundamental Modeling of Metabolic Pathway: Case study of *E. coli*

Within a cell, there is a series of chemical reactions that play an important role in metabolic engineering. This series of chemical reactions is known as metabolic pathway that consists of enzyme, substrate and product. In biochemistry, enzyme is the catalyst of chemical reaction (Grisham and Garrett, 1999) which catalyst the reaction rate by lowering

down the activation energy and often requires dietary mineral, vitamins and other cofactors in order to smooth the process.

The process that involved in the metabolic pathway are catabolic, the break-down process, and anabolic, the synthesis process that that independently work by producing new biomolecules in the final product and the process can be either reversible or irreversible way depending on cultural environment that affect the enzyme's activities such as temperature, pH, oxygen level and many more that related to the rate of reaction.

In metabolic pathway, the conversional process of substrate to product by the help of enzyme can be illustrated as in Fig 2.5

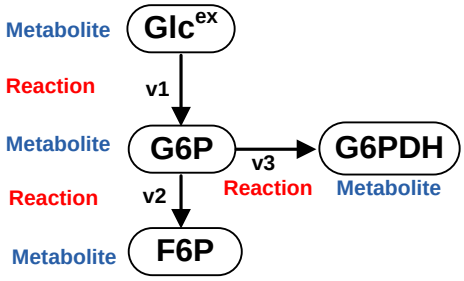


Figure 2.5: Conversion process in designing metabolic network

Metabolic network play an important role in determining the behavior and biochemical properties of on cell. In this study, *E. coli* is taken as subject because it is extremely well known bacterium in this world and yet most of scientists take it as model in their research (Madigan et al., 2000). These Gram-negative, rod-shaped bacteria can be easily grew under laboratory condition and thus it save a lot of time in cultural the *E. coli.*

To understand more on background, one must study the growth rate of bacteria. There are several of phases in bacteria need to be considered in order to study their growth rate such as lag, log, stationary and death phase.

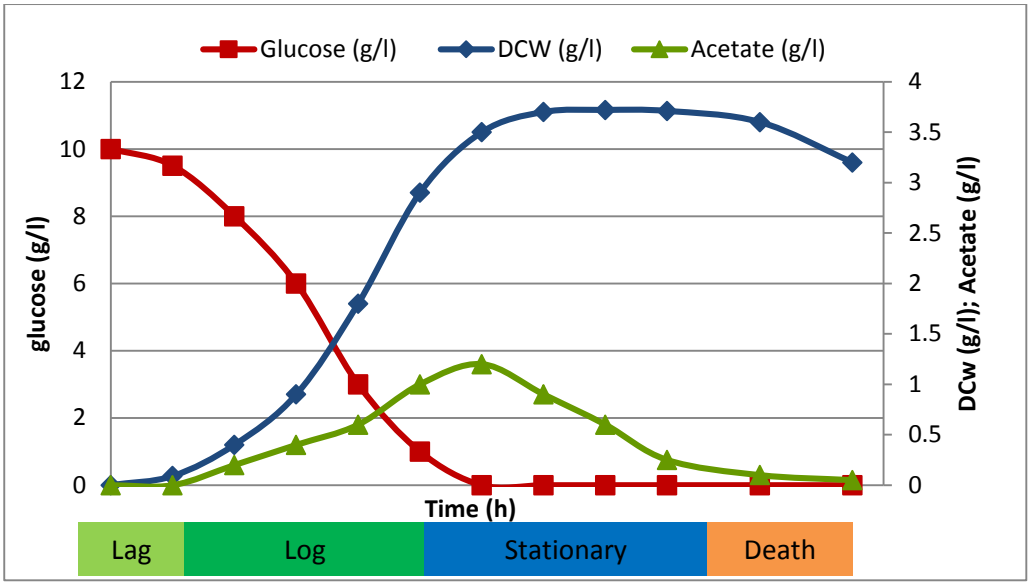


Figure 2.6: Typical batch culture of a cell

Based on Fig. 2.6, DNA replication and enzyme synthesis of cells are occurred in lag phase and this affect the growth rate of a cell in lag phase as it takes time. As DNA and enzyme synthesis complete, cells started to grow drastically by doubling the number of cells along the time period and this has increased the dry cell weight (DCW) exponentially while level of glucose or nutrient started to drop drastically and increased the production of acetate in log phase. In stationary, growth rate of cells started to maintain. As level of glucose decreasing slowly, the production of cells are equal to cells in death. Lacking of nutrient and accumulation of toxic metabolites makes the cells begin to reduce the cells production in death phase. This phase where cells death rate is higher than cells formed.

Based on the explanations above, it is show that most of attributes or behavior inside a batch culture of a cell affected along a time and the changes in cultural environment and/or the specific genetic modification. Therefore, in order to do the process of the simulation, one must take several steps involving in simulation of main metabolic pathway such are:-

a) Identify the model – metabolite, cofactors
b) Identify the mass balance
c) Identify the enzyme kinetic equation
d) Identify kinetic parameters
e) Simulate and compare with experimental data

Metabolites are the intermediates and products of metabolism and they are usually restricted to small molecules. They are needed to be stated in order to simulate the metabolic pathway of *E. coli*. Meanwhile, cofactor such as NADH, NAD, ATD, ATP and many are also being identified. Co-factors or co-metabolites (Abdul Kadir, 2010) are the non-protein chemical compound that are bound to a protein and are required for the protein's biological activity. They are also known as "helper molecules" that assist in biochemical transformation. (Refer Appendix C) for the metabolites and co-factors of the case study

Then, mass balance is identified. Conservation of mass that needs to take into account is biomass, intracellular metabolite, and extracellular glucose and acetate concentrations. Stephanopoulus et al (1991) stated that the mass balance for biomass concentration as below:-

$$\frac{dX}{dt} = (\mu - D)X \qquad (2.1)$$

where X is the concentration of the biomass and μ is the specific growth rate (h^{-1}). According to Abdul Kadir (2010) that the specific growth rate is equal to the dilution rate at steady state condition based on the equation. Thus, as the dilution rate is changed, the specific growth rate will be different. (Refer Appendix C) for the whole dynamic equations of the model used in case study.

Next, enzyme kinetic equations are identified. (Refer Appendix C) for whole enzyme kinetic equations of the case study. Enzyme kinetics is the chemical reaction study

that catalysed by enzymes. The main purposed of enzyme kinetics is to reveal the catalytic mechanism of the enzyme in how the activity is controlled.

Inside the kinetic equations, there are enzyme kinetic parameters which are identified separately in the next step. The parameters will change when the simulation is occurred. Then, the simulation is run and compared with experimental data. The purpose of study of the manual process of modelling for metabolic pathway is for development of the dynamic simulation tools and it will be discussed in chapter 3.

2.5 Dynamic model

Dynamic model is the behaviour of one parameter that can change to another product. There are several of methods to describe the dynamic model. In this project, ODE is chose as technique of dynamic model. Involving one or more dependent variables with respect to single independent variable is known as ODE.

ODE in numerical method approach play an important role in predicting and describing one quantity precisely (Severance, 2001) response to changes in some other quantity in one systems that undergoing time-dependent changes or transients which can helps engineers and scientists to solve problems by using many well-known analytical techniques. There are many methods are available that can obtain a series of expansion solution from lowest order methods such as Euler method and Bessel's equations to the highest order such as Runge-Kutta methods. In this project, Runge-Kutta is used as main technique. Severance stated that dynamical equations are characterized by their system state and often described by a set of differential equations and if they are combined with their initial conditions that uniquely specify the system; the variables specified by initial conditions create the system state variables.

Generally, consider the *N*th Order Ordinary Differential Equations of the form

$$F\left(x, y, \frac{dy}{dx}, \frac{d^2y}{dx^2}, \frac{d^3y}{dx^3}, \dots, \frac{d^ny}{dx^n}\right) = 0 \tag{2.3.1}$$

Based on Eq. of type (2.3.1) is in termed *nth-order* because the highest derivative is of order *n.* It is an ordinary because it is independent variable as it is only total derivatives appear. A unique solution can be obtained if there is a supply of additional important information of its specific values of *y(x).*

It can also be written in n first-order equation by introducing new variables of *n* – 1. By using second order equation, Bessel's equation,

$$x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - p^2)y = 0 \tag{2.3.2}$$

where *p* is constant. It can also be written in form of first-order equations if one new variable $z = {}^{dy}/_{dx}$ is introduced and substitute into Eq. 2.3.2

$$\frac{dy}{dx} - z = 0 \tag{2.3.3}$$

$$x^2 \frac{dz}{dx} + xz + (x^2 - p^2)y = 0 \tag{2.3.4}$$

The illustration of the equations in (2.3.1) until (2.3.4) are for familiarize of highest-order equations as they are written in the similar way in order to describe first-order equations. In

other word, the declaration of new variable is to solve complex equation easily and usually it useful to incorporate into their definition some factors in equation or some powers of independent variables (Press et al., 1992).

2.5.1 Runge-Kutta method

Brice Carnahan et al. (1969) stated that it is possible to develop step by step procedure that used only first-order derivative evaluation and it produced results that equivalent in accuracy to Taylor expansions up series and that method propagate over an interval which is known as Runge-Kutta methods. It used the combination information from several Euler-style steps that involve the right-hand $f's$ evaluations in order to match Taylor series expansions up to higher derivatives.

Consider the Euler's method of Eq. 2.4

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad (2.4.1)$$

The Eq. 2.4.1 is advances a solution from x_n to $x_{n+1} \equiv x_n + h$ and the formula is not symmetrical. Based on the Eq. 2.4.1, we take it to "trial" step to midpoint of the interval and use the value of x and y at point to compute the "real" across the whole interval.

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + \frac{h}{2}, y_n + k_1 \frac{1}{2})$$

$$k_3 = hf(x_n + \frac{h}{2}, y_n + k_2 \frac{1}{2})$$

$$k_4 = hf(x_n + h, y_n + k_3)$$

$$y_{n+1} = y_n + k_1\frac{1}{6} + k_2\frac{1}{3} + k_3\frac{1}{3} + k_4\frac{1}{6} + O(h^5) \quad (2.4.2)$$

Runge-Kutta method treats every step identically make it easy to relatively simple "driver" scheme. The simple "driver" goes from initial x_s and to final x_f

2.6 Statistical method

The used of statistic is widely used in science in order to understand the data better. Statistic helps to describe and understand the numerical relationship between variable. It also makes better decision in the face of uncertainty. The basic steps that involve using the right statistical techniques are described as following:-

a) Identify the problem
b) Decide the data
c) Collecting the data
d) Classify and summarize the data
e) Present and analyse the data
f) Make decision

Collection, organization of sample data is known as descriptive statistics. It is one of the traditional statistical method and it is used for summarize the data by using measures of central tendency, measures of dispersion and measures of position. Measures of central tendency consist of mean, median, mode and midrange while measures of position are divided into quartiles, deciles and percentiles. In this project, the main focus of the descriptive statistic that will be used is measurement of variation.

Measurement of dispersion includes range, variance and standard deviation focus on the wellness of average value and yet it can measure the variability of data sets. In this project, the main focus is the variance. Variance is the average square distance of each value is from mean. It suitable to determine the spread of the data and if the variance is larger, the data are more distributed. If $s_1 < s_2$, the data from s_1 is more consistent and if data more consistent, the data are more precise.

2.7 Development tools

"Rosario", the codename for Microsoft Visual Studio 2010 is IDE created by Microsoft that used to develop console and GUI application. Although, it comes with different version such as Standard, Professional and Enterprise, it actually does not affect at all. The central component of Visual Studio is .NET Framework. It has three important parts – .NET languages compiler, Common Language Runtime (CLR) and Framework Class Library (FCL). The frameworks are Basic, C++, C#, ASP.

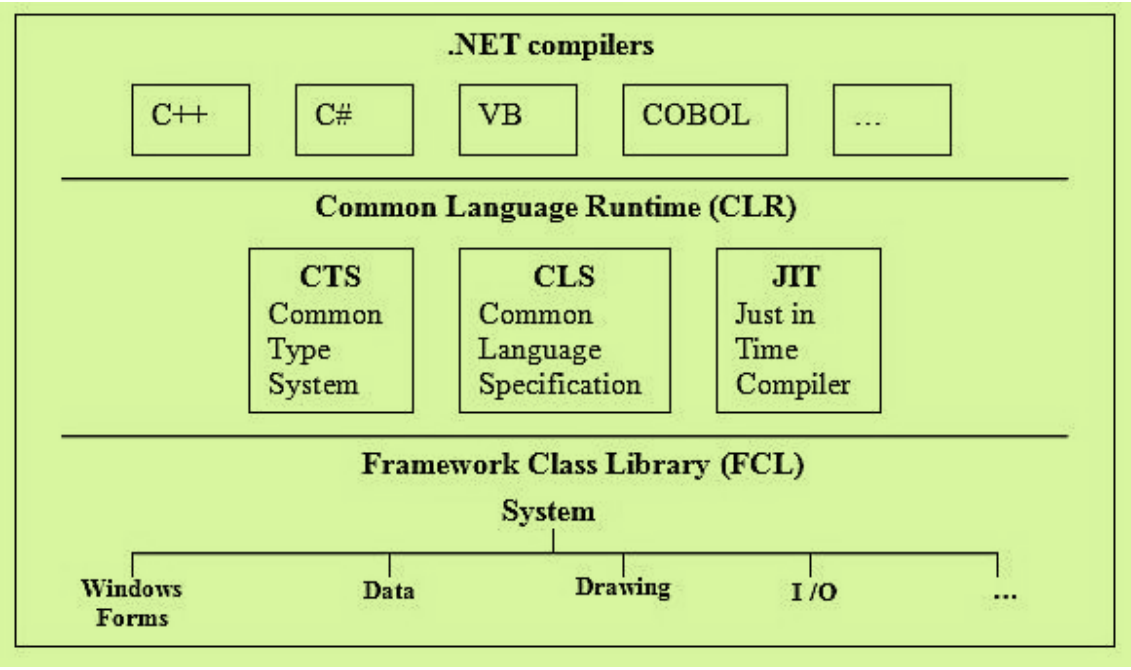


Figure 2.7.1: .NET framework (Mohd Azhar, 2005)

In the .NET framework, there is a compiler that checks syntax as well as translate it into executable form (Mohd Azhar, 2005). In other word, the process is then translating the source code into Intermediate language (IL) that used by CLR and translates in into executable code. This gives benefits to us to write one program in languages that supported in Orcas and interact with each other. (Mohd Azhar, 2005).

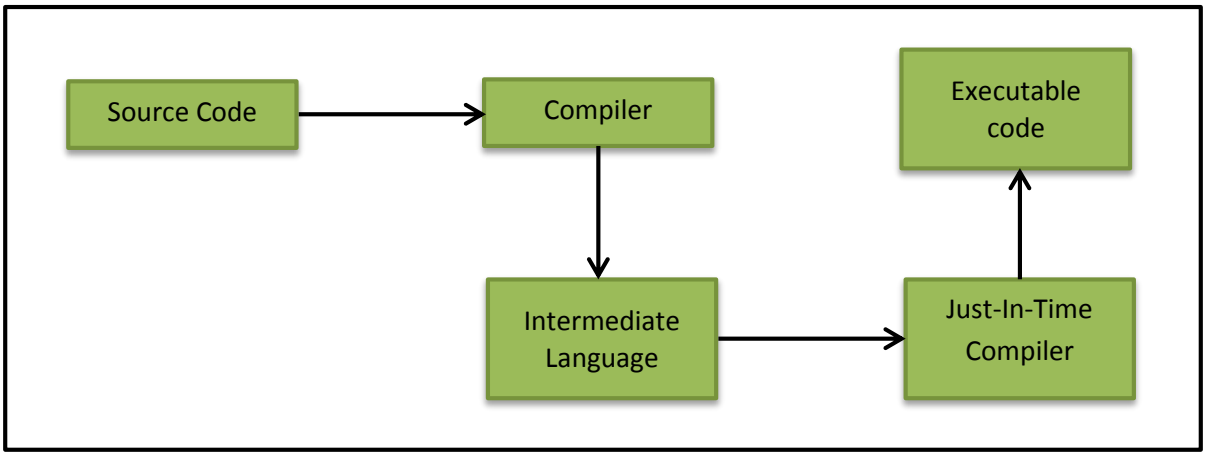


Figure 2.7.2: Compiling and Executing

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter is about the whole dynamic simulation tools development process that covers overall view of dynamic simulation tools that being created. This chapter consists of six (6) phases:-

a) Project Planning
b) Project Analysis
c) Project Design
d) Project Development
e) Project Completion and thesis submission

3.2 Framework of project

The development framework used is based on Software Development Life-Cycle (SDLC) waterfall approach with some modification in order to match the development process of modeling and simulation. (Refer Fig. 3.1) The process flow started with planning, system analysis, system design, system development and testing and project completion.

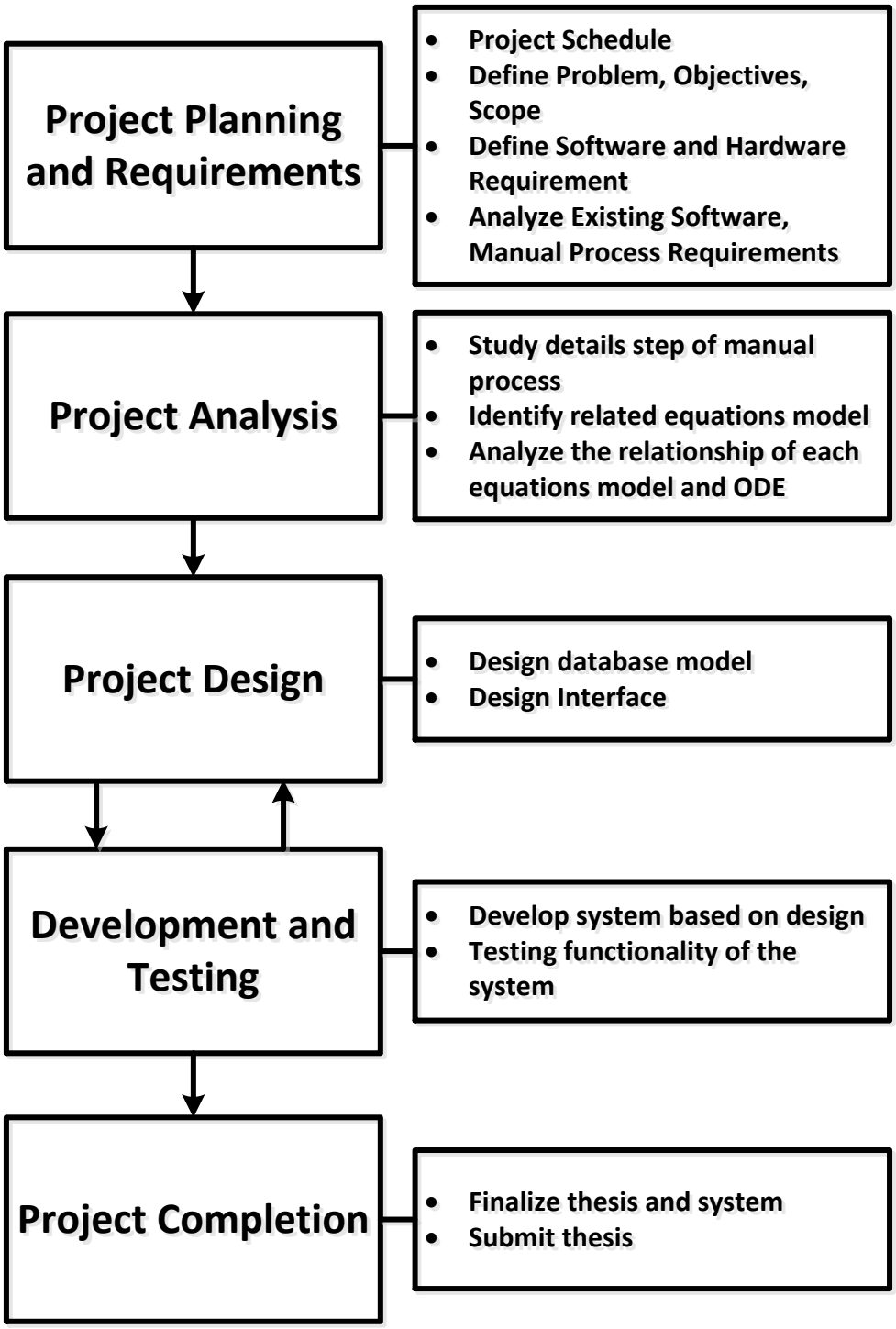


Figure 3.1: Framework of the project

3.2.1 Planning of Project Work and its Requirements

The purpose of this phase is to ensure the project is well organized and well started. Firstly, the research on market values is conducted. It is found that modeling and simulation (M&S) is well known in engineering field. The details of advantages of M&S can be found in chapter two (2). Then, the problem statement, project objectives and the project scope are well defined. These are the most important steps in this project and they can be found in chapter one (1) for more details. Next, the duration of the project development is constructed in a Gantt chart (refer Appendix A for the details of activities) by using Microsoft Project 2010. The duration of activities are stated as well.

Project requirements analyses are made next. There are several analysis are made to determine the requirement need to meet for development purpose. The analyses are:-

a) Analysis on software tools and techniques used
b) Existing dynamic simulation tools
c) The manual process of simulation based on existing system

As for the analysis of software tools, the general requirements need to have are:-

a) Laptop or Personal Computer (PC) with operating system (OS) of Windows or UNIX based
    i) The OS recommended is Windows XP and above for Windows based.

    ii) For UNIX based, user can either use Mac OS X Tiger and above or Linux OS and must have virtual machine that can run Windows OS.

b) Laptop or PC that support the technologies that used in the project
    i) The user must have Microsoft Excel 2010, Visual Studio 2010 technology, Microsoft Access 2010 and notepad pre-installed in his laptop or PC.

There are 2 types of tools that are used in the development which are:-

a) Software tools
b) Hardware tools

For the project development, the software tools used are Windows based platform. The details of software are described in table 3.1 while hardware tools are described in table 3.2

SOFTWARE TOOLS	PURPOSE
Windows 7 Professional 64 bits	Platform of whole development of project
Microsoft Visual Studio 2010	Development purpose of the project such as design interface, coding and testing
Microsoft Office Word 2010	Platform of documentation of the project
Microsoft Office Power Point 2010	Platform for creating presentation slide
Microsoft Office Excel 2010	Platform for calculation of variance
Microsoft Office Visio 2010	Platform for drawing flow chart
Microsoft Office Project 2010	Platform for drawing Gantt Chart and planning the project outline
Google chrome	The searching platform for research purpose
Bitdefender Internet Security 2013	Protection of viruses and malware
Foxit Reader	Viewing platform of PDF files

Table 3.1.1: Software tools

HARDWARE TOOLS	SPECIFICATION	PURPOSE
Laptop	Name : Sony VAIO Processor : Intel ® Core 2 Duo RAM : 4.00 GB Clock Speed : 2.53 GHz	- Research purpose - Platform to run the application, software and project - Overall project development platform

Table 3.1.2: Hardware Tools

Moreover, the existing tools are analyzed and some of the features are take used based on section 2.3 in chapter two (2). The features that taken into account are:-

a) Easy to used
b) Consistency of interface
c) User control the system
d) Used database

Meanwhile, the technique used in the development is Runge-Kutta, one of the highest derivatives in the numerical method of ODE. The algorithm details are discussed in chapter two (2).

Modeling and simulation consists of three (3) phases. The phases are:-

a) Develop the Simulation model
b) Design and conduct simulation experiment
c) Perform simulation analysis.

In this project, the main focus is the steps (b) and (c) as the models are used based on the user defined. In order to design the simulation experiment, the manual simulation techniques need to be analyzed first and it is found that there are six (6) steps need to be highlighted. The six steps are:-

a) Identify the model - input the metabolites
b) Identify the mass balance / dynamic equation
c) Identify the kinetic equations
d) Identify the kinetic parameters
e) Simulate the model
f) Compare with experimental data.

Another purpose of analysis of manual simulation process is for designing the flow of the process and it will be discussed detail later on in chapter four (4).

3.2.2 Project Development Analysis

In this phase, the detail steps in M&S of metabolic pathway are studied in order to understand the flow process clearly. From the studies, the related equations and parameters (refer Appendix C) involved in dynamic simulation of metabolic network are well defined. The amount of metabolic pathway is also can be found based on these studies (refer Appendix B).

In this project, the amount of metabolites are limited to maximum of nine (9) and minimum two (2) to be input. Based on analysis previously, the metabolites are identified first in order to setup the model. Below is the example of reactions:-

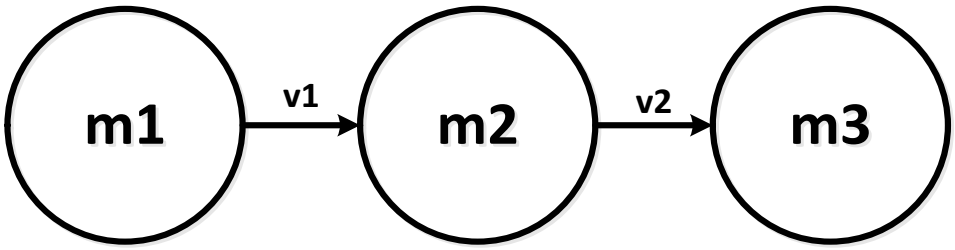


Figure 3.2: Metabolites reaction

Where *m1, m2, m3* are metabolites and *v1,v2* are kinetic reactions. Next, the mass balance / dynamic equations are identified. The basic concept for determined the equations are illustrated in equations below:-

$$INPUT + GENERATION = OUTPUT \quad \text{(Eq. 1.0)}$$

Inside the equations, the *INPUT* is the preliminary or initial state of the metabolite and OUTPUT is the altered metabolite. If considered in figure 3.2 example, *m1* is *INPUT* while *m2* is *OUTPUT*. For reactions *m2*, *m2* is *INPUT* and *m3* is an *OUTPUT*. The dynamic equations are based on intracellular metabolite concentration proposed by (Chassagnole et al., 2002) in (eq. 2.0) and mass balance given by (Stephanopoulus et. al., 1998) in (eq. 3.0). The intracellular metabolite concentration equations are illustrated as below:-

$$\frac{dC_i}{dt} = \sum_j v_{ij}\, r_j - \mu C_i \qquad \text{(eq. 2.0)}$$

where C_i is the concentration of metabolite *i*, v_{ij} is the stoichiometric coefficient in the reaction *j*, r_i represents the rate of formation of component *i*, $-\mu C_j$ represent the reduction of concentration by dilution due to increase in cell volume.

Meanwhile, the mass balance equations are illustrated as:-

$$\frac{dX}{dt} = (\mu - D)X \qquad \text{(eq. 3.0)}$$

where X is the concentration of the biomass and μ is the specific growth rate (h^{-1}). From this equation, we can see that the specific growth rate is equal to the dilution rate at steady state condition. Therefore, the specific growth rate will be different as the dilution rate is changed.

Enzyme kinetics is the chemical reaction study that catalyzed by enzymes. The main purposed of enzyme kinetics is to reveal the catalytic mechanism of the enzyme in how the activity is controlled. Enzyme kinetics is identified based on number of metabolites. For example, a pair of reaction (two (2) metabolites) required one (1) kinetic equations. Yet, inside the kinetic equations, there are enzyme kinetic parameters which are identified separately in the next step. The parameters will change when the simulation is occurred.

Lastly, the relationship between mass balance and kinetic parameter in order to simulate are also been studied and analyzed. The reactions are simulated using Runge-Kutta technique. The details studies can be found in literature review of chapter two (2)

3.2.3 Design the Flow and Interface of Project

In this section, the project is developed based on the analysis of the previous phase. There are two design are made in this section. The first design is the process design of the project and the second design is the interface design of the project. The flow process of the project is represented in data flow diagram (DFD). DFD is a graphical representation that can be used to visualize the structured design of the project. Fig 3.2 is the overall of the flow process of the project. The details will be discussed in chapter four (4).

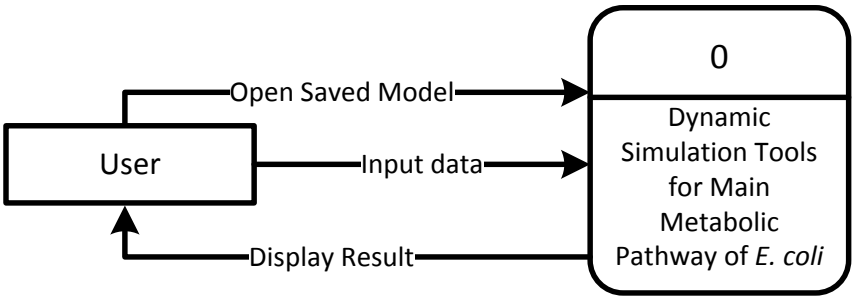


Figure 3.2: A context diagram of Dynamic Simulation tools for main metabolic pathway of *E. coli*

Designing the database is also the most important step in this phase. The purpose of database in this project is to store the equations and their parameter in the organized way so that they can be retrieved easily when needed. Below is the entity-relationship diagram of the database model (refer Fig. 3.3)

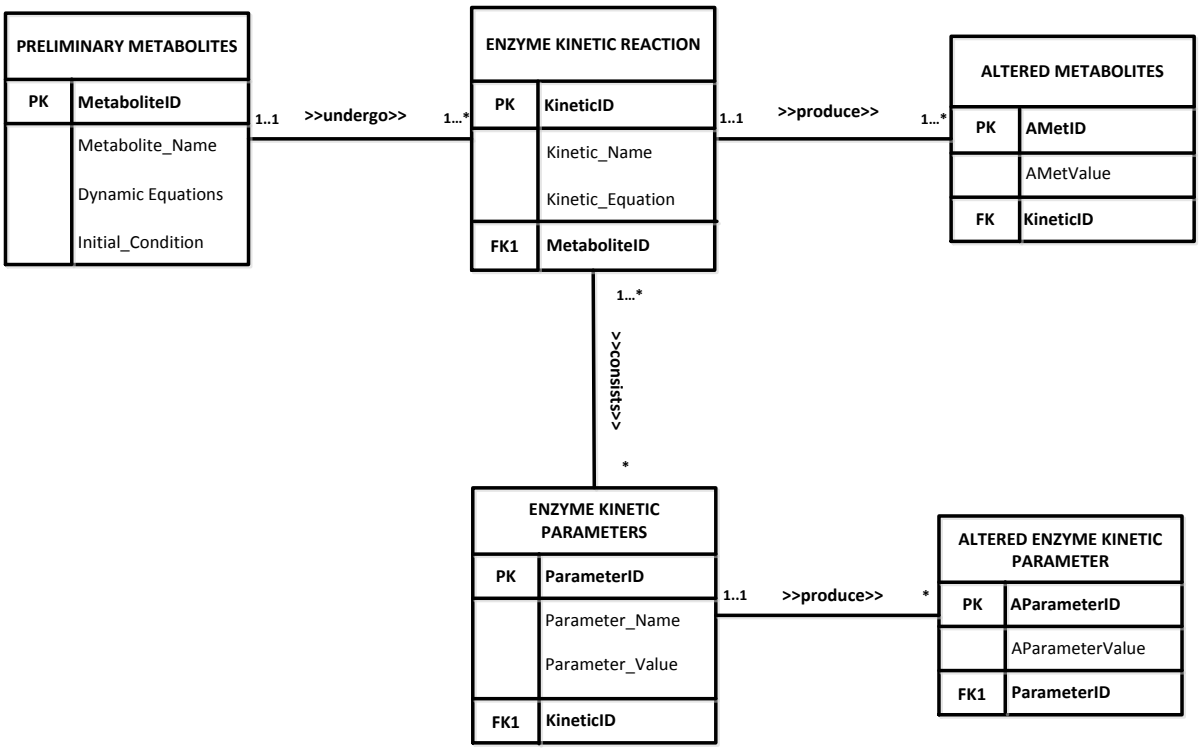


Figure 3.3: Database design of the dynamic simulation tools

Meanwhile, there are several steps need to take into account for the good design of the interface. The stages of the design process are:

a) Pre-production design phase
b) Design during production phase
c) Post-production design phase
d) Redesign phase

The detail of interface design will be discussed in Chapter 4.

3.2.4 Project Development and Testing

This is the most important phase in development of project. There are two sub-phases that are development and testing. In development phase, the project is developed based on the

planning design. In this step, the algorithm is implemented and the language will be used is C# in Visual Studio platform while the technique that focused on is Runge-Kutta method of ODE numerical method. There are also conditions need to take into account for the project to runs. The conditions are:-

a) The dynamic simulation can input 9 metabolites for the first trial
b) The simulation is non-stationary as it changes overtimes
c) User needs to input the mathematical equation of his/her model as condition the simulation to be run.

Meanwhile for project testing, as the project coding is completed, it will proceed to testing phase where each module will be tested on its function. There are two things that been tested which are the non-functionality and functionality modules. Non-functionality modules are focus on interface functionality and the module that being tested are:-

a) Create Project Module
b) Open Project Module
c) Input metabolites / kinetic equations / kinetic parameters modules
d) Modifying metabolites / kinetic equations / kinetic parameters modules
e) Displaying graph module

For functionality test, the module is focused in using Runge-kutta method correctly in the simulations. If there is an error, the error will be fixed immediately. As there is no error, the project can be declared as successful for the first version. The detail of the development phase will be discussed detail in Chapter five (5).

3.2.5 Project Completion and Thesis Submission

This is the last process of development is where the project is done and the thesis is being finalized and sends to supervisor for approval. The whole thesis needs to consist seven (7) chapters.

CHAPTER 4

DESIGN

4.1 Introduction

The purpose of this chapter is to design the needs of the program and how the overall program is interacted with each other. There are two designs that will be discussed. The first one is the requirement structuring design. Requirement structuring design or logical design is the analysis studies of the previous chapter which help in structuring and representing the program in the form of DFD diagram and Use Case Diagram (UCD) as well as designing the database structure. Besides that, another design that will discuss is the physical design or interface design of the program. The interfaces are design based on the requirement needs that are stated by client and the requirements are transforming into flow process of storyboard

4.2 Overall project process flow design

In the project design, the overall project is developed based on the criteria and requirements analysis made previously. It is represented in flow chart as shown in Figure 4.1.

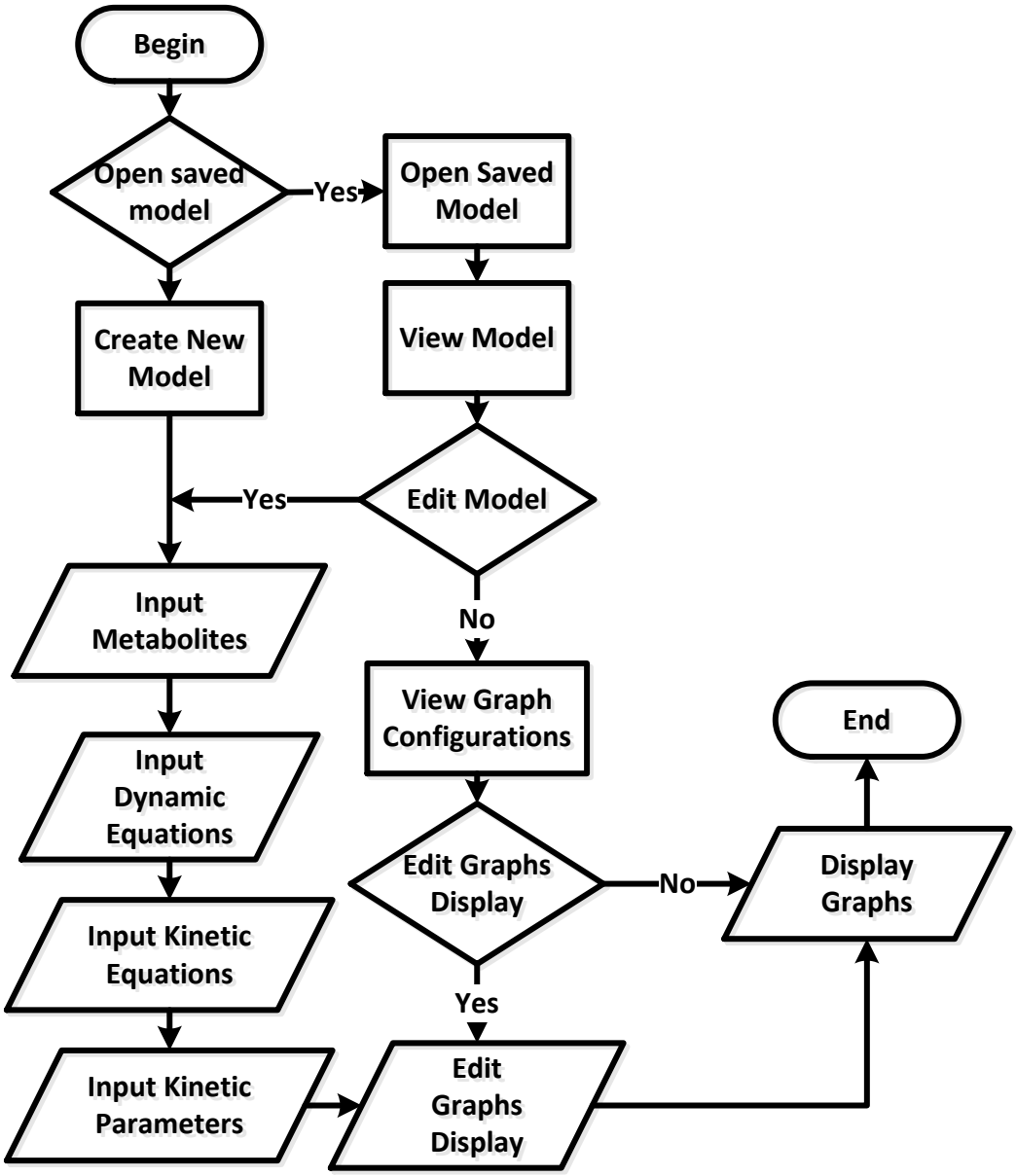


Figure 4.1: Flowchart of the project

There are assumptions being made in designing structure of the project. The assumptions are:-

a) It is assumed that the experimental data uploaded is correct.
b) It is assumed that the mathematical model is correct. The dynamic simulation tools do not tend to optimize the mathematical model of user.

Types of simulation results are divided into:-

a) Model simulation result
b) Experimental result

c) Model and experimental result
d) Result analysis verification

4.3 Systems Design

Systems design is focuses on high level design like, what programs are needed and how they are going to interact with each other. This phase is how the individual programs are going to work in Low-level design, what are the interfaces going to look in interface design phase and what data will be required in data design. There are two type of design that will be discussed. The first design is the logical design which is an abstract representation of the data flow, input and outputs of the system. Meanwhile, another design that will be discussed is the physical design which be more discussed on user interface design.

4.3.1 Data flow diagram (DFD) of the system

A DFD is graphic representation of the flow of the data through the system. It is used for the visualization of the structure design and it is separated into 3 levels – Context Diagram (refer Fig. 4.2.1), Decomposed in lower level DFD Level 0 (refer Fig. 4.2.2) and Decomposed in lower level DFD Level 1 (refer Fig. 4.2.3). In order to produce a practical DFD, one needs to list down all the activities that are involved in the system. In this system, the activities involved are:-

a) Create New Model
b) Open Saved Model
c) View Input
d) Display Result

Once the list is made, the context diagram of the project is created. Fig. 4.2.1 shows the overall context of the project. This level shows the overall context of the system and its operating environment and shows the whole system as one process. The system allowed user to input data or open saved model and display the result back to user.

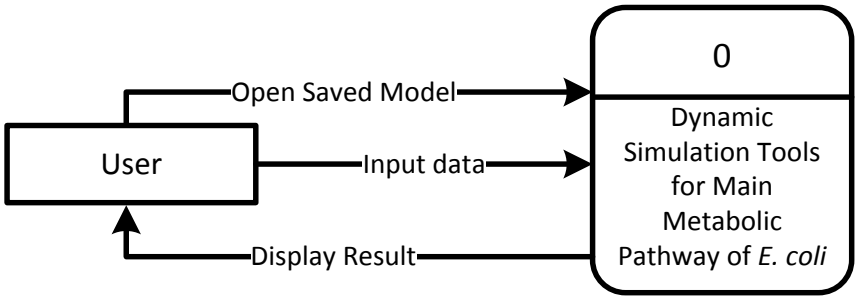


Figure 4.2.1: Context Diagram

Then, context diagram is break down into surface detail or known as Level 0 DFD based on the previous list down activities. This level shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major high level processes of the system and their interrelation. Figure 4.2.2 shows the Level 0 of DFD.

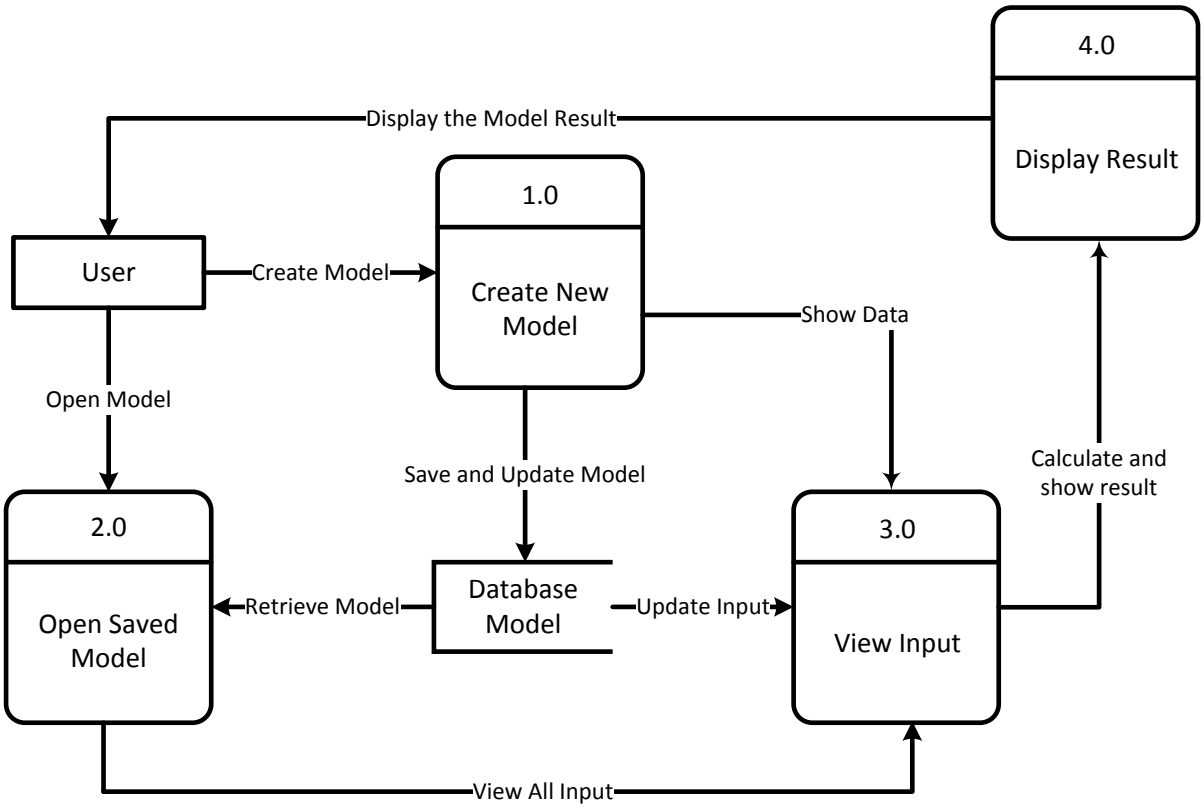


Figure 4.2.2: Level 0 DFD

Level 1 DFD is created after the creation of Level 0 DFD, as such there should be a level 1 diagram for each and every process shown in a level 0 diagram. The main reason of the level 1 is to identifying the data store of the system. In this project, the data source used is the database model. The design of database model will be discussed in subtopic 4.3.3.

In this example processes 1.1 & 1.2 are all children of process 1, processes 2.1 & 2.2 are all children of process 2, processes 3.1 & 3.2 are all children of process 3, processes 4.1 & 4.2 are all children of process 4, together they wholly and completely describe process 1,2,3,4 and combined must perform the full capacity of this parent process. A level 1 diagram must be balanced with its parent level 0 diagram. Figure 4.2.3 shows the level 1 DFD model.

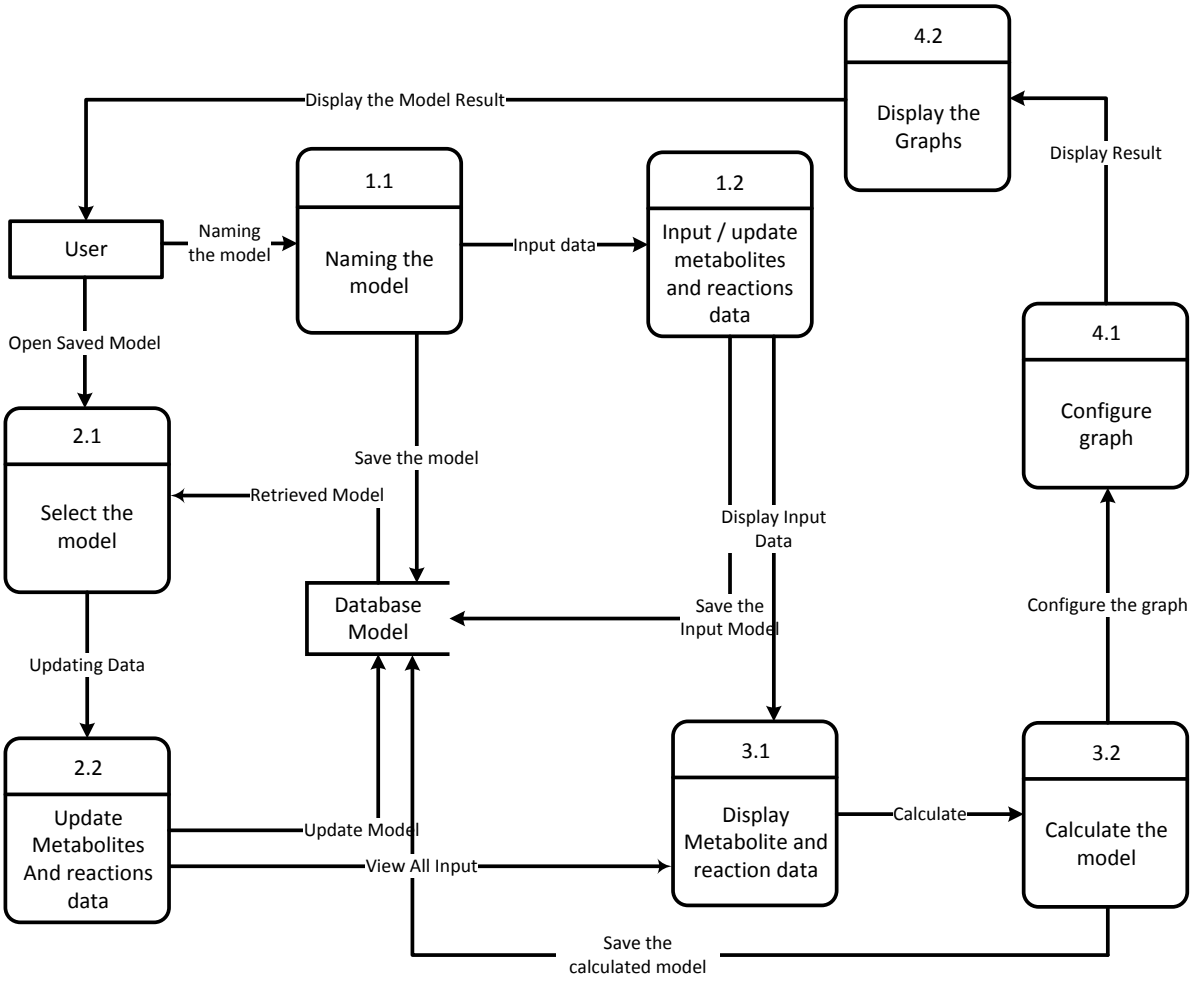


Figure 4.2.3: Level 1 DFD

4.3.2 Use Case Diagram (UCD)

A use case diagram (UCD) is a graphical representative of system behavior along with actors that interact with the system. UCD is used to show all of its available functionality. In this project, the user (actor) is interacting with the respective function of dynamic simulator as shown in the figure 4.3

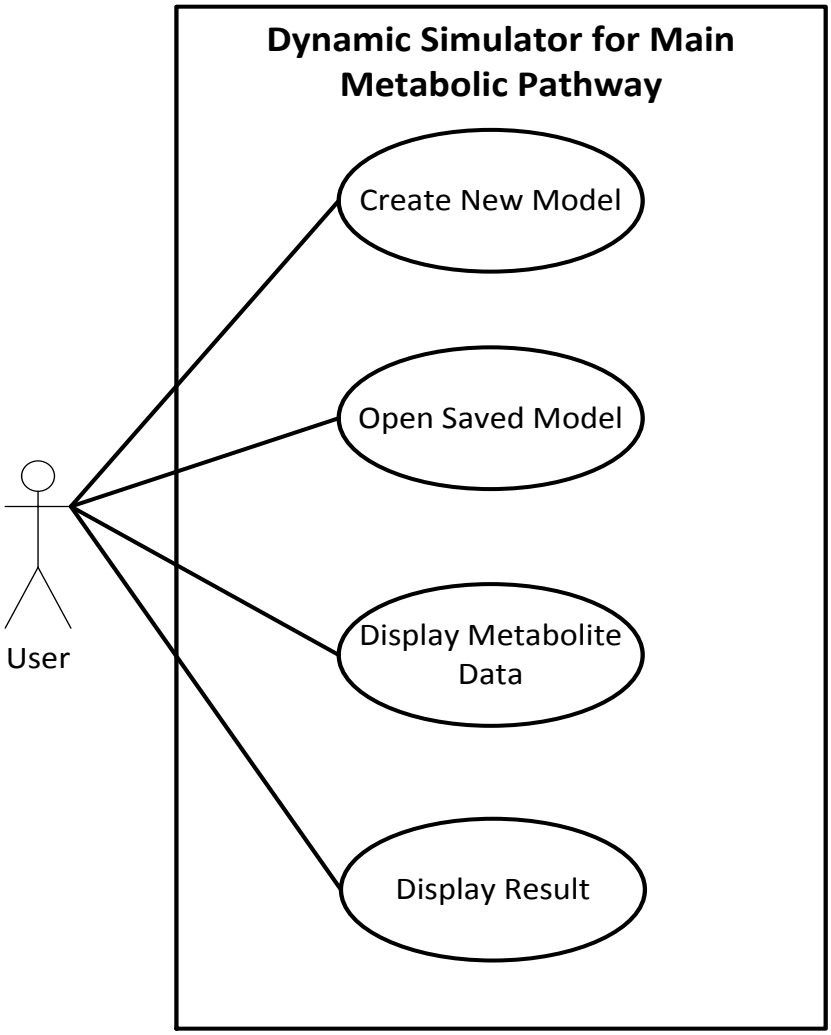


Figure 4.2.4: Use Case Diagram of Dynamic Simulation Tools

4.3.3 Database Design

Detailing the data model of database is one of the important phases in producing the good system. A lot of variables need to be considered and analyze in this project. The variables are stated as below:-

ENTITY	ENTITY SETS
PRELIMINARY METABOLITES	MetaboliteID, Metabolite_Name, DynamicEquations, Initial_condition
ENZYME KINETICS REACTION	KineticID, Kinetic_Name, Equations
ALTERED METABOLITES	AMetaboliteID, AMetaboliteValue
ENZYME KINETIC PARAMETERS	ParameterID, Parameter_Name, Parameter_Value
ALTERED ENZYME KINETIC PARAMETER	AParameterID, AParameterValue

Table 4.1: Entities involved in simulations

The attributes involved in the simulations are analyzed based on business rules. Below are the business rules of the project:-

- PRELIMINARY METABOLITES undergo one or more ENZYME KINETICS REACTION and each ENZYME KINETICS REACTION belongs to one PRELIMINARY METABOLITES
- Each ENZYME KINETICS REACTION produce one or many ALTERED METABOLITES and each ALTERED METABOLITES belongs to one ENZYME KINETICS REACTION
- Many ENZYME KINETICS REACTION consists many ENZYME KINETIC PARAMETERS
- Many ENZYME KINETIC PARAMETERS produce many ALTERED ENZYME PARAMETERS

The entity-relationship diagram (ERD) is drawn in order to represent the relationship between each of the entity. Roles are also stated at each relationship between entities' link.

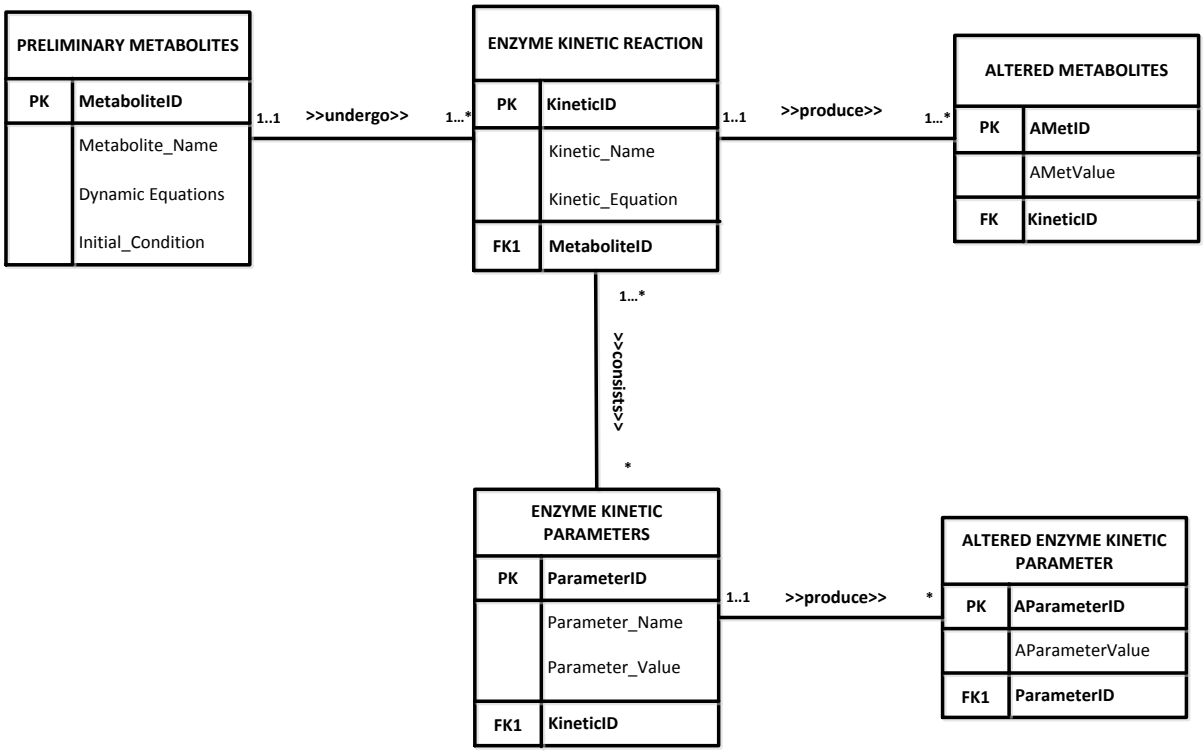


Figure 4.3: ERD of Dynamic Simulation Tools

As ERD has been established, relation model is then being created. The relation model used the basic concept of a relation or table and every tuplets must have a unique identification. Below is the Relation Model of the project:-

PRELIMINARY METABOLITES

PreliminaryMetabolites (MetaboliteID, Metabolite_Name, DynamicEquations, Initial_Condition)

Primary Key: MetaboliteID

Foreign Key: None

ENZYME KINETICS REACTION

KineticEquation (KineticID, MetaboliteID, Kinetic_Name, Equations)

Primary Key: KineticID

Foreign Key: MetaboliteID

ALTERED METABOLITES

AlteredMetabolite (AMetaboliteID, KineticID, AMetaboliteValue)

Primary Key: AMetID

Foreign Key: KineticID

ENZYME KINETIC PARAMETERS

EnzymeKineticParameters (ParameterID, KineticID, Parameter_Name, Parameter_Value)

Primary Key: ParameterID

Foreign Key: KineticID

ALTERED ENZYME KINETIC PARAMETER

AlteredEnzymeKineticParameter (AParameterID, ParameterID, AParameterValue)

Primary Key: AParameterID

Foreign Key: ParameterID

There are three (3) tables involved which are Metabolite, KineticEquation and KineticParameter tables that involved in this project. Therefore, three data dictionaries are created for each table. Table 4.2.1 – table 4.2.3 shows the data dictionaries of this three tables.

FIELD	TYPE	SIZE	DESCRIPTION	CONSTRAINT
MetaboliteID	VARCHAR	10	Unique ID for metabolite	Primary Key
Metabolite_Name	VARCHAR	45	Name of Metabolite	
DynamicEq	VARCHAR	70	Dynamic Equation	
InitialCondition	DOUBLE	-	Initial Condition of Mass Balance	

Table 4.2.1: Data Dictionary for table ‘Preliminary Metabolites’

FIELD	TYPE	SIZE	DESCRIPTION	CONSTRAINT
KineticID	VARCHAR	10	Unique ID for Enzyme Kinetic	Primary Key
PMetID	VARCHAR	10	Unique ID for metabolite	Foreign Key
KineticName	VARCHAR	45	Name of Kinetic Equation	
KineticEquation	VARCHAR	70	Kinetic Equations	

Table 4.2.2: Data Dictionary for table 'EnzymeKineticsReaction'

FIELD	TYPE	SIZE	DESCRIPTION	CONSTRAINT
AMetaboliteID	DOUBLE	-	Unique ID for Altered Metabolite	Primary Key
KineticID	VARCHAR	10	Unique ID for Enzyme Kinetic	Foreign Key
AMetaboliteValue	DOUBLE	-	Value for Altered Metabolite	

Table 4.2.3: Data Dictionary for table 'AlteredMetabolites'

FIELD	TYPE	SIZE	DESCRIPTION	CONSTRAINT
ParameterID	VARCHAR	10	Unique ID for Kinetic Parameter	Primary Key
KineticID	VARCHAR	10	Unique ID for Enzyme Kinetic	Foreign Key
ParameterName	VARCHAR	45	Name of Kinetic Parameter	
ParameterValue	DOUBLE	-	Value of Kinetic Parameter	

Table 4.2.4: Data Dictionary for table 'EnzymeKineticParameter'

FIELD	TYPE	SIZE	DESCRIPTION	CONSTRAINT
AParameterID	DOUBLE	-	Unique ID for Altered Kinetic Parameter	Primary Key
ParameterID	VARCHAR	10	Unique ID for Kinetic Parameter	Foreign Key
AParameterValue	DOUBLE	-	Value of Kinetic Parameter	

Table 4.2.5: Data Dictionary for table 'AlteredEnzymeKineticParameter'

4.4 Interface Design

In this section, dynamic simulation tools are developed based on the analysis of the previous phase. Design the interface is categorize as pre-production design. The benefit of pre-production design is for illustrate the concept of the project. The interface design is discussed in next sub-section. Below is the flow process pre-design interface of the projects.

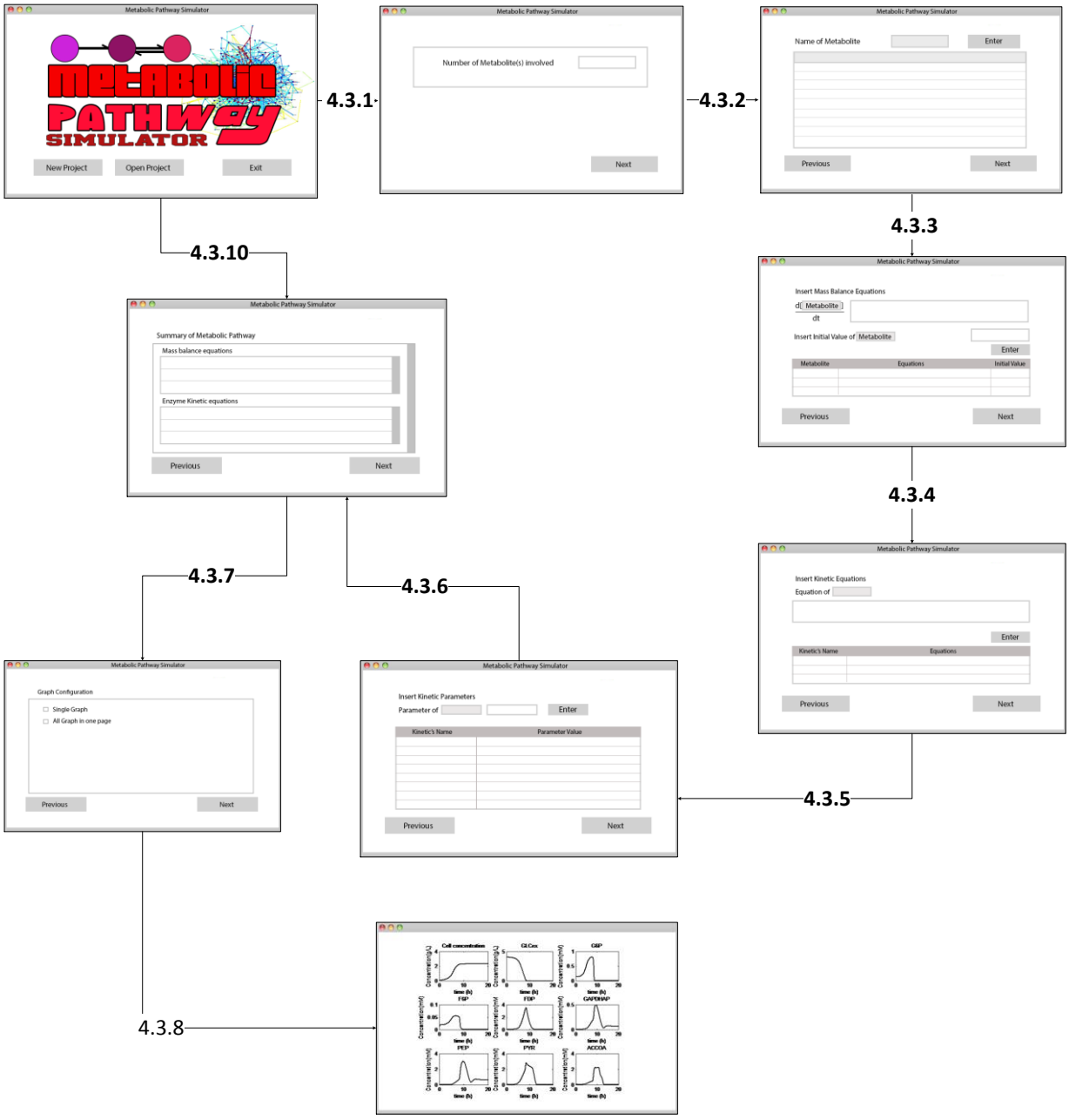


Figure 4.4: Flow process of the pre-design interface

There are eight (8) fixed interfaces and one or several graphs interfaces based on user configuration. Basically user can choose either create a new model or open the previous saved model on the first interface (Refer Figure 4.2.1). If user wanted to create a new model, he needed to click 'New Project' button at the first interface (refer Figure 4.2.1) and the second interface will be appeared next.

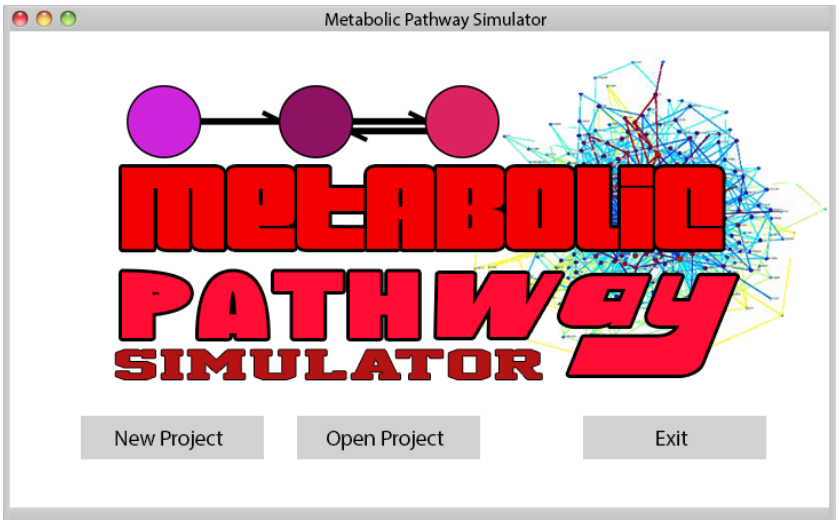


Figure 4.4.1: Main interface of metabolic pathway simulator

On the second interface (refer figure 4.2.2), user is needed to insert amount of metabolites. The amount of metabolites is required at least two in order the reaction to take place.

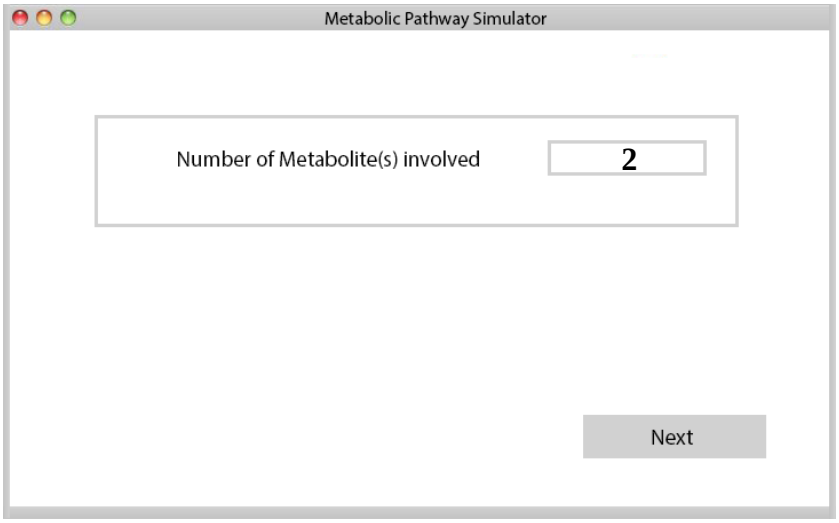


Figure 4.4.2: Interface of amount of metabolites and co-metabolite(s) involved

Next, user is required to insert the name of each metabolites involved (Refer Figure 4.2.3). The names needed to be different to each other and it must not have space in the string of the names. Each of the names is being mapped on the database which will be discussed on the next sub-topic.

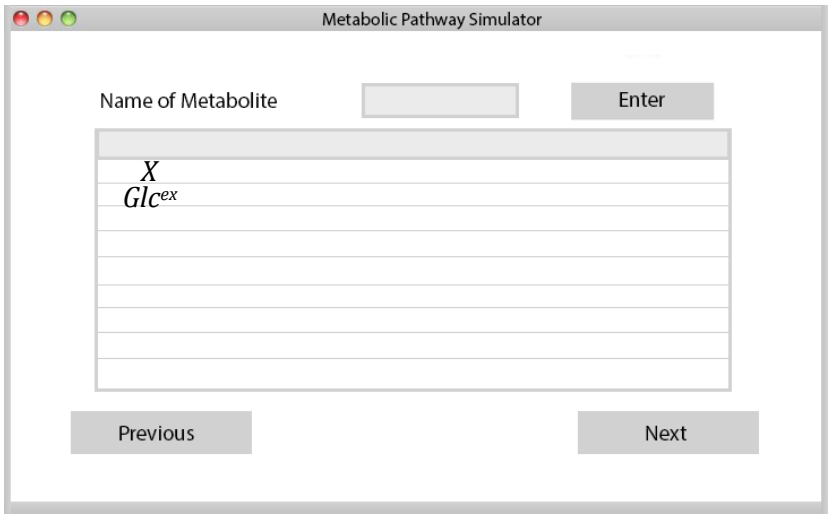


Figure 4.4.3: Interface of metabolite(s) and co-metabolite(s) name

Then, user is required to insert mass balance / dynamic equation in next interface (refer figure 4.2.4). Each of the mass balance / dynamic equations needs to have the initial value. Therefore, it is compulsory to insert the initial value. The equations will be displayed below once the user clicked 'Enter' button.

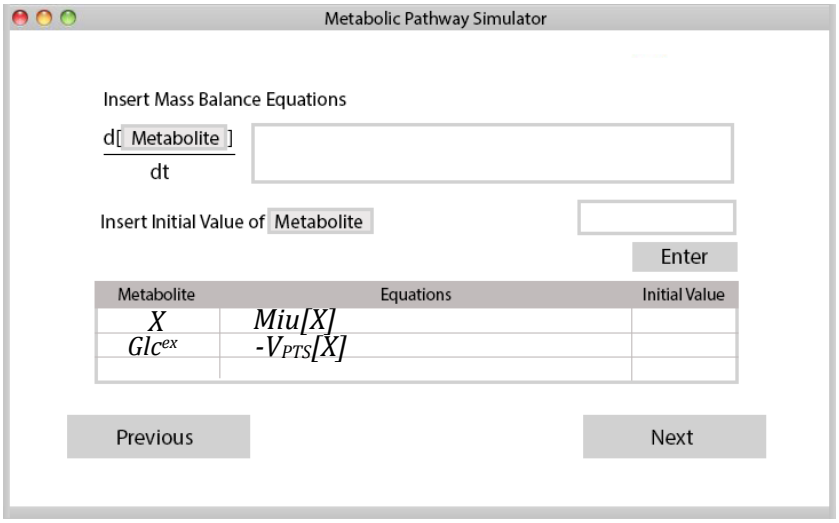


Figure 4.4.4: Interface of mass balance equations

Each of the mass balance /dynamic equations will be then parser into chunk of strings. For example, in metabolite *x* (refer figure 4.2.5), *Miu* is considered as enzyme kinetic reaction meanwhile *x* is metabolite concentration. When the string is parser into chunk, *Miu* will be placed in kinetic equations interface as user need to insert another equations of reaction while *x* will be placed in kinetic parameter interface (refer figure 4.2.6).

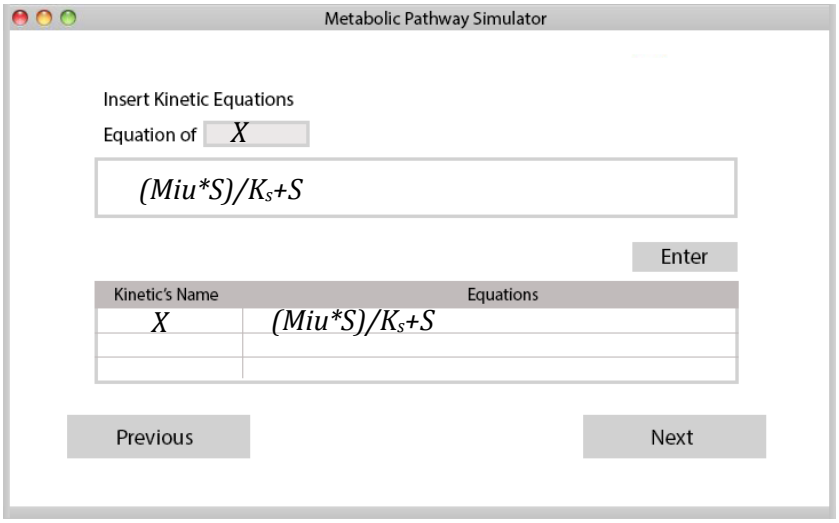


Figure 4.4.5: Interface of kinetic equations

Once user had inserted the kinetic equations, the next interface (refer figure 4.2.6) is appeared when user click 'Next' button. User is required to insert the parameter value in order the simulation to take place.

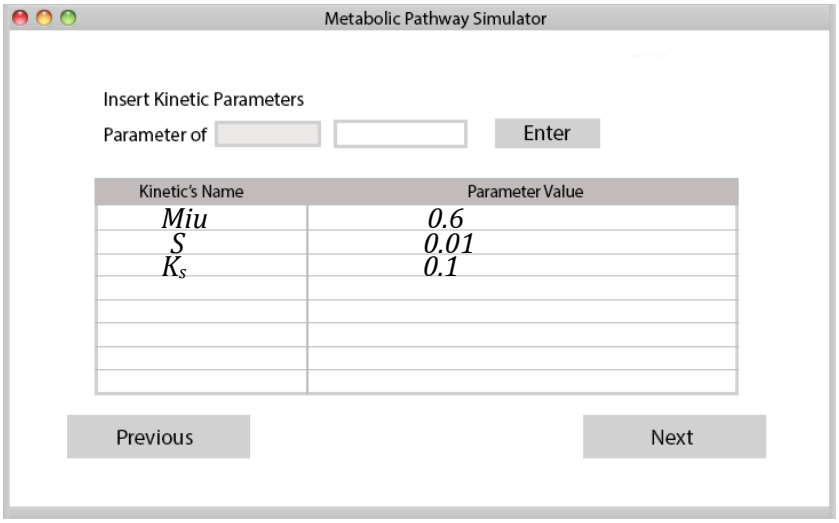


Figure 4.4.6: Interface of kinetic parameter

The summary interface (figure 4.2.6) will show once user click ‘Next’. It summarized all the mass balance equations, kinetic equations and kinetic parameter in one interface. User may make an immediate change in this interface

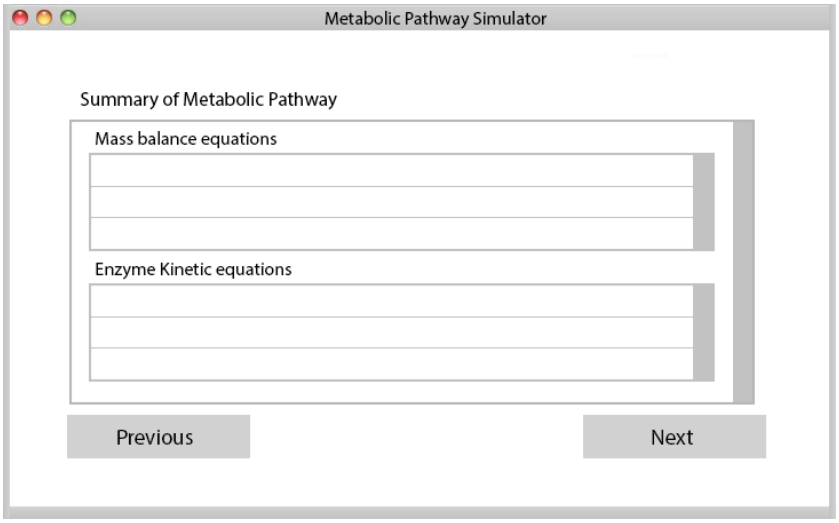


Figure 4.4.7: Summary of Metabolic Pathway equation

Besides, user need to configure the graph in the graph configuration interface (refer 4.2.7). The options available are all graphs in one page, one graph per page and user customize graph

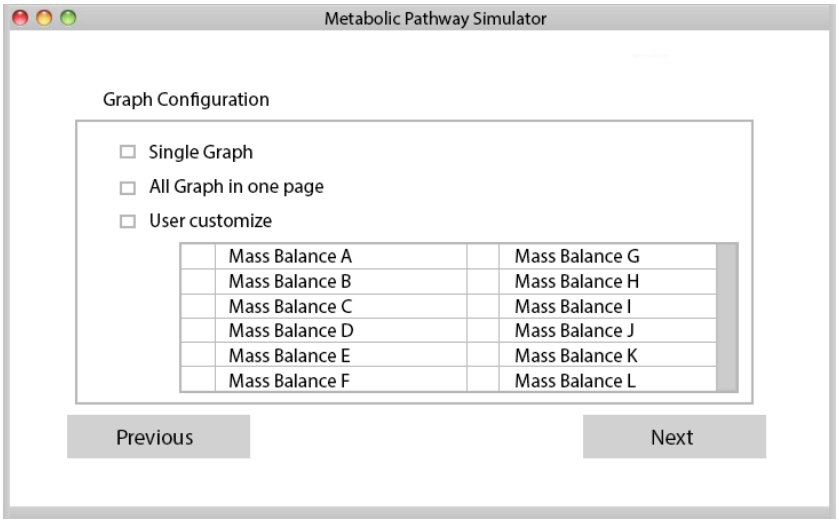


Figure 4.4.8: Interface of graph configuration

Once configurations of graphs are done, user will see the graphs he/she wanted. The example of graph interface is shown as in figure 4.4.9.

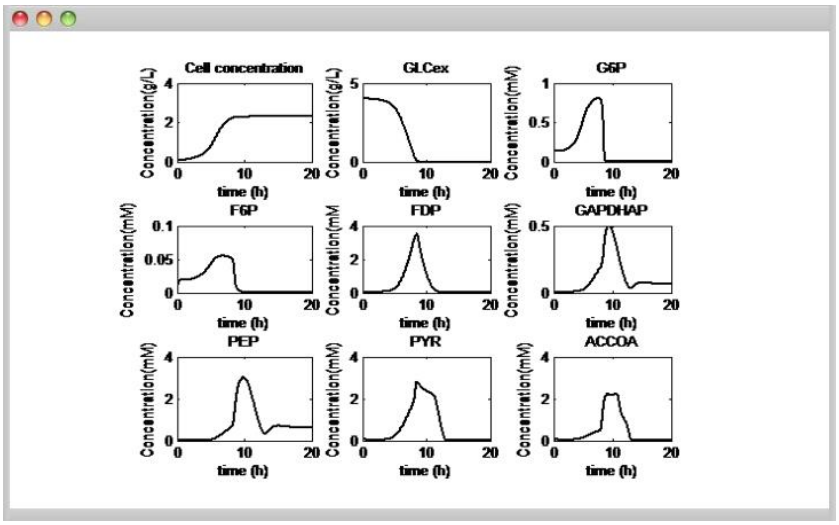


Figure 4.4.9: Example of graph interface

CHAPTER 5

IMPLEMENTATION

5.1 Introduction

The main purpose of this chapter is to document all process that involved in the development of the system. Basically, this chapter is explaining about project development that has been designed. The content that contains in this chapter is depending on the project that has been developed. This chapter will explain details on how to implement the entire subject that has been discussed earlier.

This system is developed in C# language and .NET Framework by using Visual Studio 2010 and it is created as a Windows Form Applications. Moreover, the database is design programmatically and it is been implemented in MySQL software.

5.2 Development of Graphical User Interface (GUI)

GUI plays an important role in order to allow user to interact with the system. In other words, it allowed user to provide input and it will process and produce the output. The development of GUI is divided into three (3) main interfaces – Main interface, Metabolites configurations and Graphs configurations. These developments are done in Visual Studio 2010. Figure 5.1 shows the start page of Visual Studio 2010.

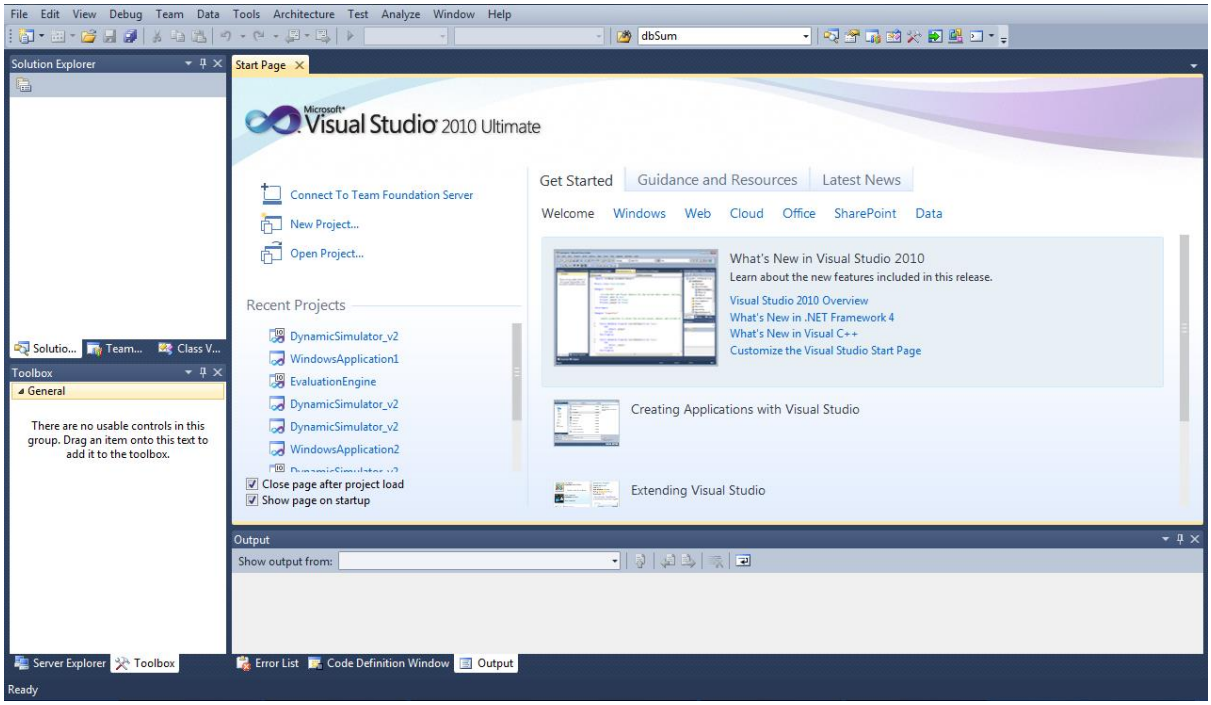


Figure 5.1: Start page for Visual Studio 2010

5.2.1 Main Interface Module

Figure 5.2: Main Interface

Figure 5.2 shows the main interface of the system. It consists of main logo, 3 buttons – 'New Project' button, 'Open Project' button and 'Exit' button. Each button has its own function. The function for 'New Project' is to create new project. Meanwhile, 'Open Project' is for open the previous project. Lastly, the 'Exit Button' is for terminated the program from running.

```
private void btnNew_Click(object sender, EventArgs e)
      {
          if (schemaForm.ShowDialog() == System.Windows.Forms.DialogResult.OK)
          {
              CreateDatabase();
          }
      }
```

Figure 5.3.1: Code for 'New Project' button

Figure 5.3.1 shows the code implementation for 'New Project' button. When user click the button, the schemaForm module form will appear and demand the user to key-in schema or database name for the new project. Note that user can only input character without space and error will prompt out if user tends to input symbol in the database name. Figure 5.3.2 – figure 5.3.4 illustrate on wrong entering database name.

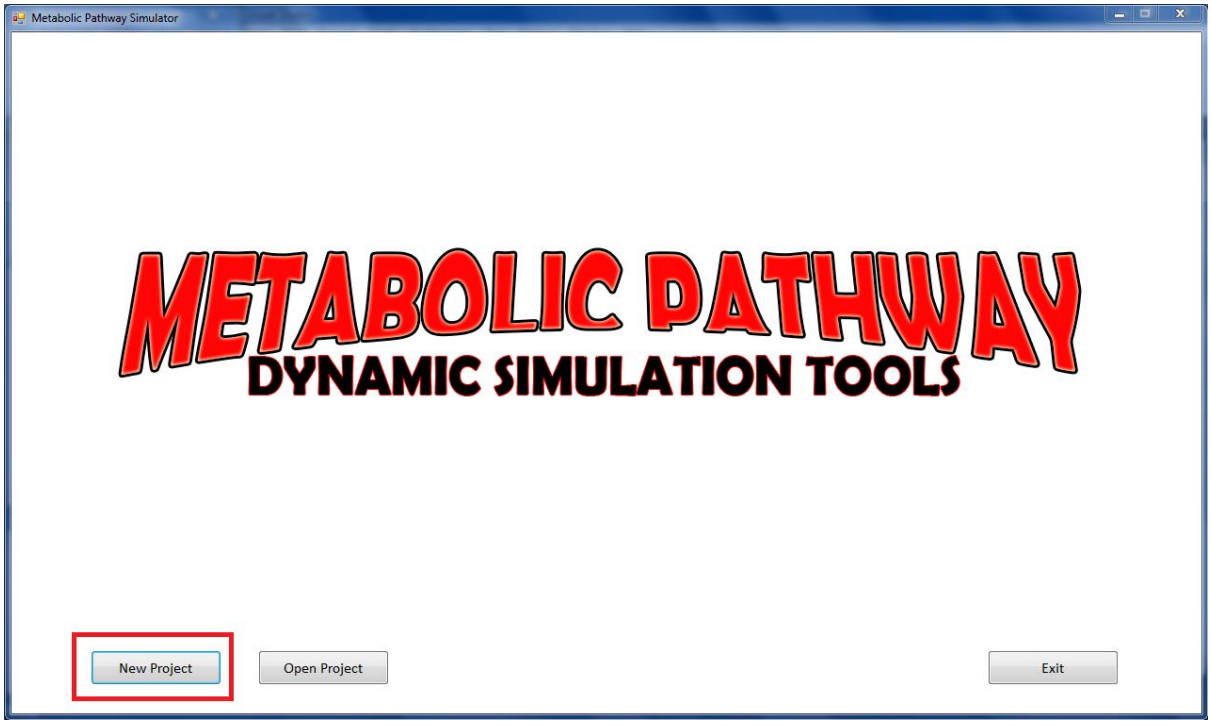


Figure 5.3.2: Illustrate on new project button – Click 'New Project' Button

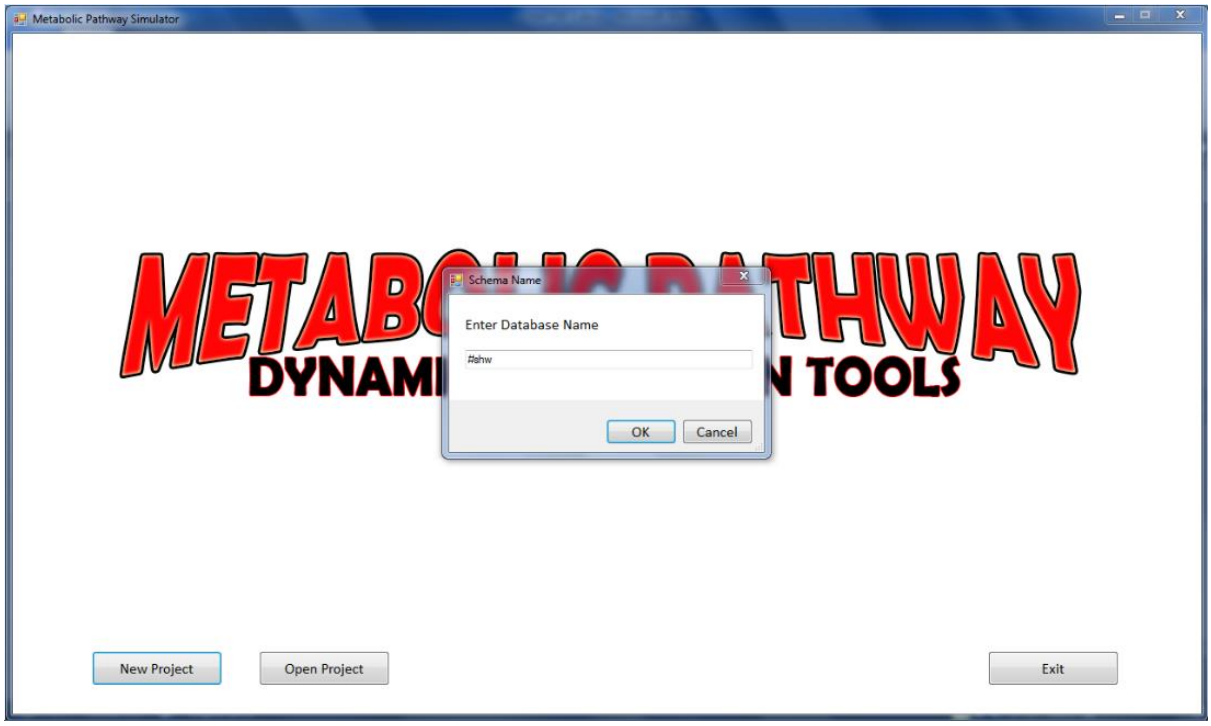


Figure 5.3.3: Illustrate on new project button – Enter database name with symbol

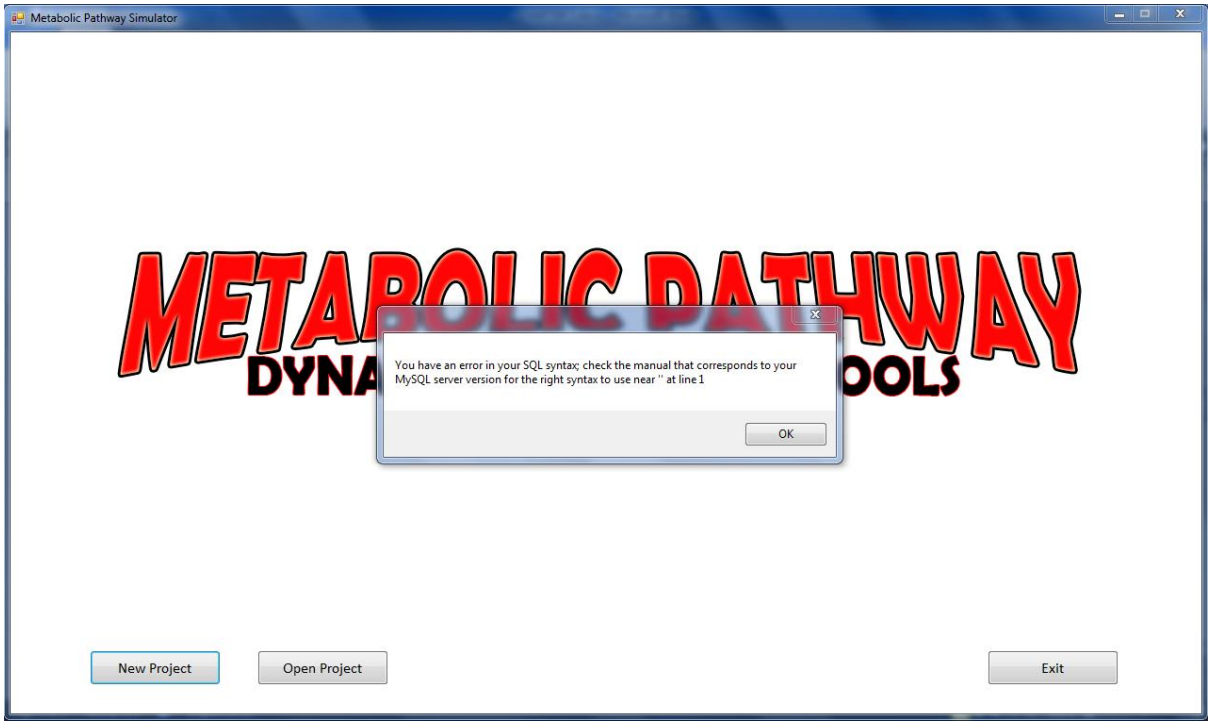


Figure 5.3.4: Illustrate on new project button – Error prompt on error

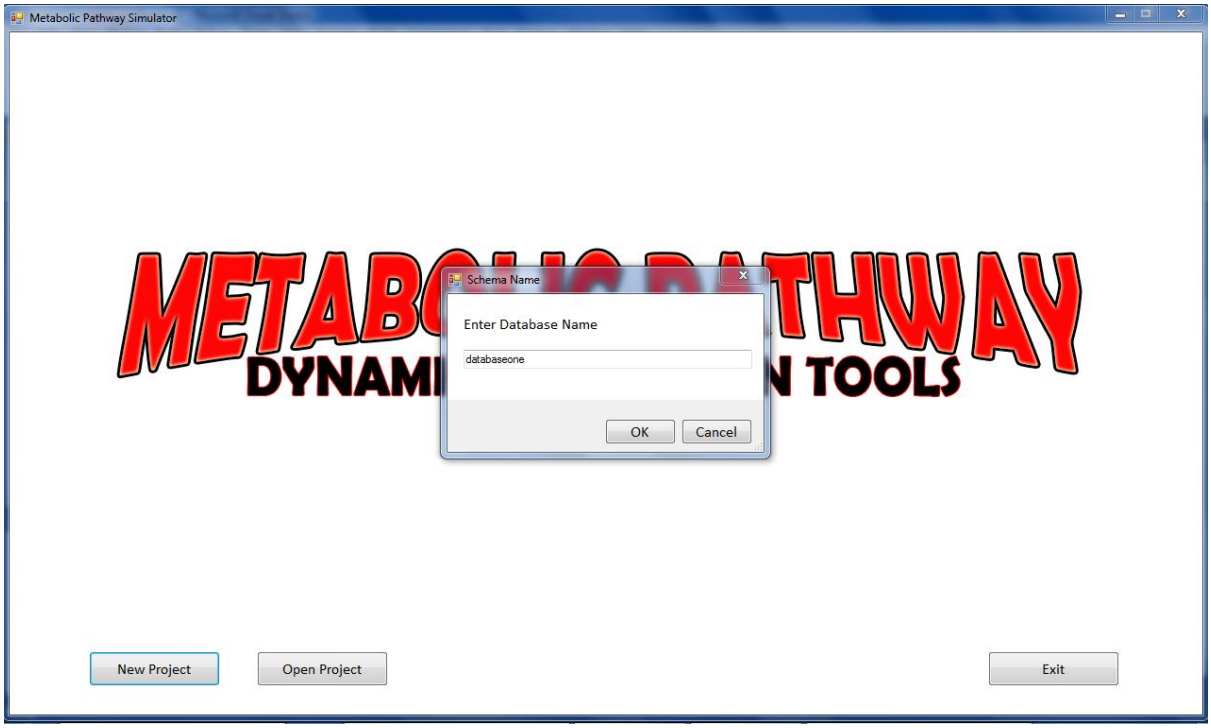


Figure 5.3.5: Illustrate on new project button – Entering the database name correctly

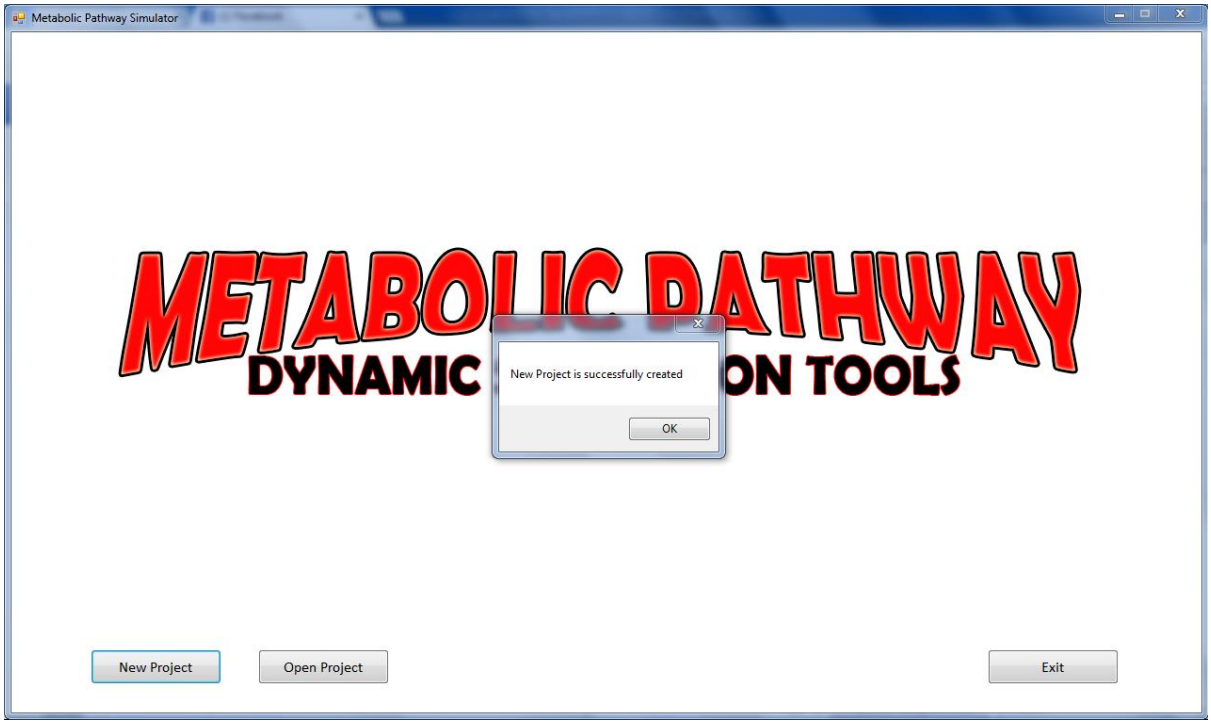


Figure 5.3.6: Illustrate on new project button – Prompt successful message

If the user enter database name according to rules, the database will be created in MySQL software and the successful message will be prompted as well as the next form will be showed (refer figure 5.3.5 – figure 5.3.6). The rule of creation database is limited to

alphabet character only and user cannot use spacing bar in naming the database as name with symbol or numbers for schema or database are case sensitive. Figure 5.3.7 shows the code for calling creation of database.

```
private void CreateDatabase()
      {
          try
          {
              db.Database();
              MessageBox.Show("New Project is successfully created");
              this.Hide();
              form2.Show();
          }
          catch (Exception ex)
          {
              MessageBox.Show(ex.Message);
          }
      }
```

Figure 5.3.7: Calling database creation

Figure 5.3.8.1: Illustrate on Open Project button – Clicking Open Project Button

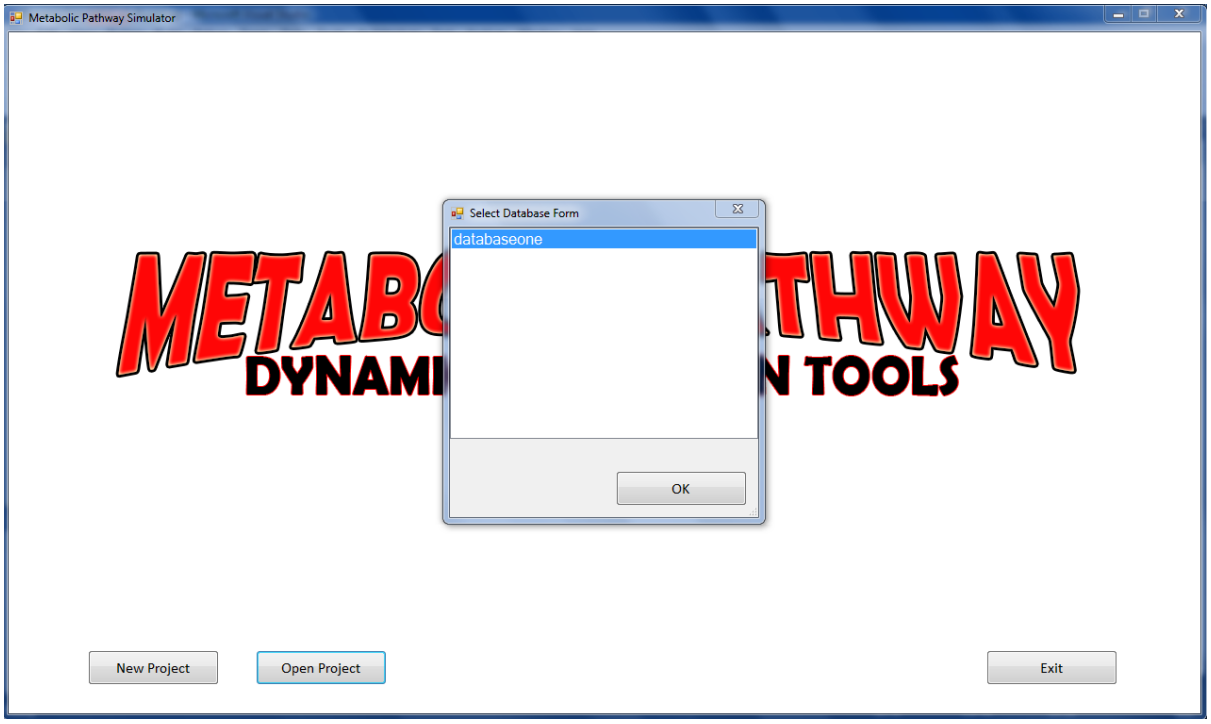


Figure 5.3.8.2: Illustrate on Open Project button – Selecting the interest database

Figure 5.3.8.1 – figure 5.3.8.2 show the illustration of 'Open Project' module. Once user press the 'Open Project' button, the 'Select Database Module' will be open. User has to select his or her interest database project and the Review form will be open. In review form, user can re-edit back the project. The code for Open project is implemented in figure 5.3.9.1 – figure 5.3.9.2

```
private void btnOpen_Click(object sender, EventArgs e)
      {
          if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
          {
              OpenDatabase();
          }
      }
```

Figure 5.3.9.1: Code for Clicking Open Button

```
private void OpenDatabase()
      {
          try
          {
              this.Hide();
              form4.Show();
          }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Figure 5.3.9.2: Code for Open Database

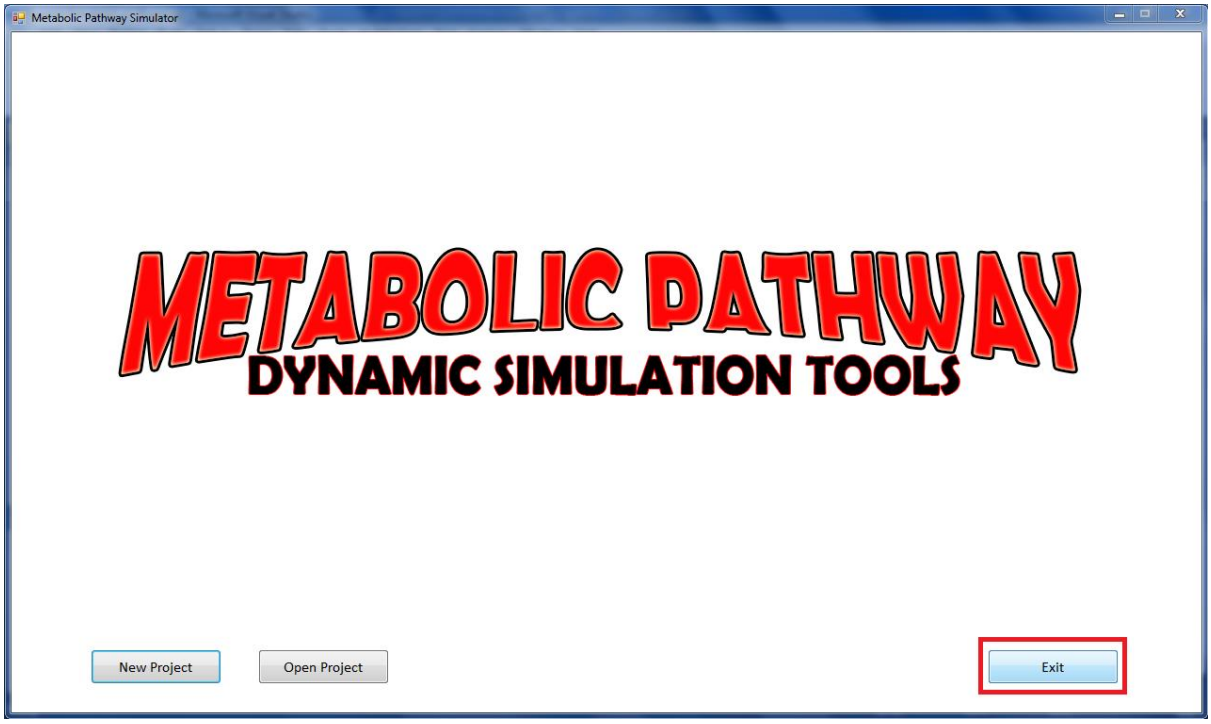


Figure 5.3.10: Illustration on Exit Button

Figure 5.3.10.1 shows the illustration of exit button. For 'Exit' button, the code is implemented in figure 5.3.10.2. This code is used to terminate the application from running or debugging.

```
private void btnExit_Click(object sender, EventArgs e)
      {
          Application.Exit();
      }
```

Figure 5.3.10.2: Code for 'Exit' Button

5.2.2 Metabolites configurations Module

Metabolites configurations Module is the second form of the system. It will show once user inserts the database name correctly. It consists five (5) tab pages – Metabolites' page, Dynamic Equations' page, Kinetic Equations' page, Kinetic Parameters' page and summary page.

5.2.2.1 Metabolites Page

Figure 5.4.1 shows the windows form of metabolites configurations in Metabolites' page. Meanwhile figure 5.4.2 shows the appearing of groupbox for renaming the metabolites. The main purposes of naming the metabolites are for declaration names for variables that are later have been used in MATLAB.

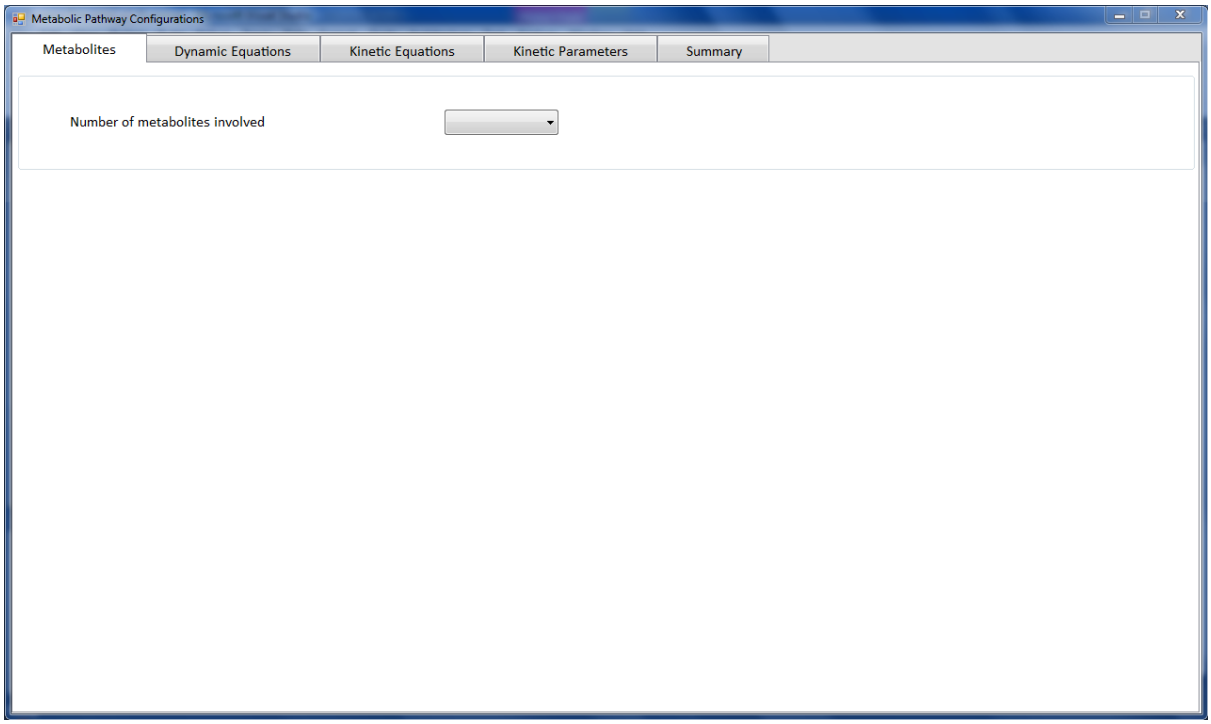


Figure 5.4.1: Metabolites' page in Metabolite Configuration form

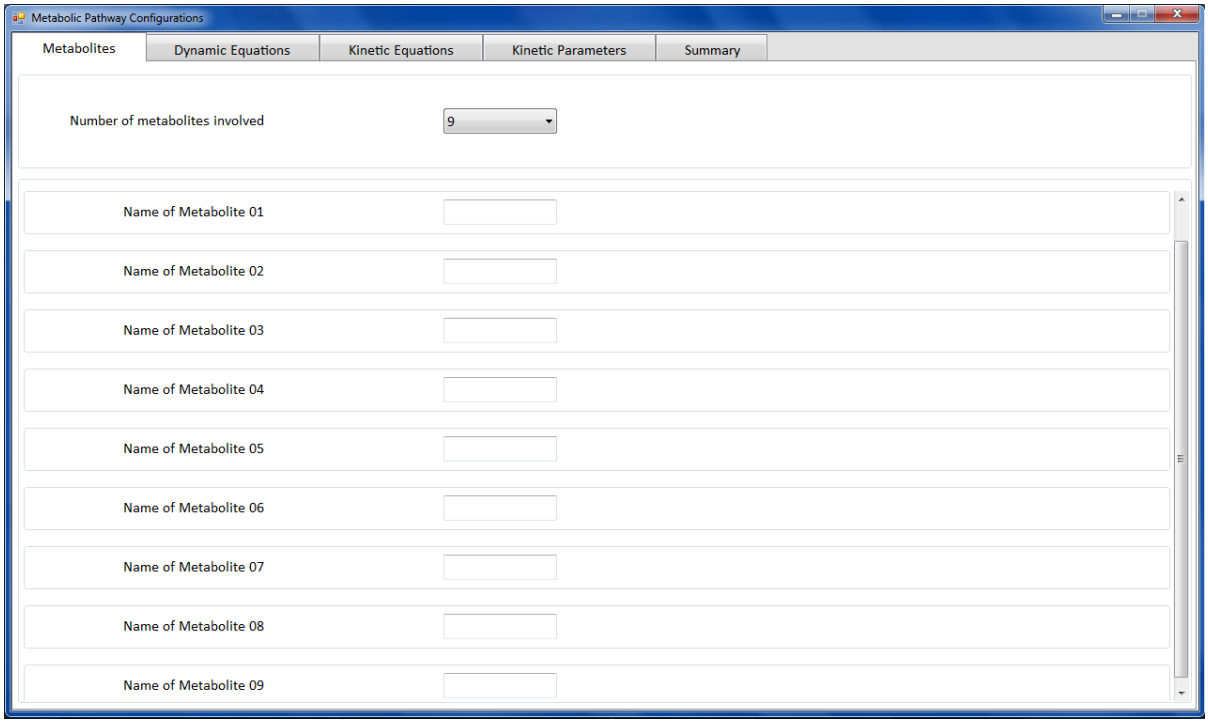


Figure 5.4.2: Screenshot on appearing of renaming the metabolites name

Inside metabolite's page, user needs to choose the number of metabolites involved. The minimum number of metabolites are two (2) while the maximum number of metabolites are nine (9). User has to select number of metabolites and the rename group box will appear. Then, he/she needs to rename the metabolites. There is one limit for renaming the metabolites which is user cannot include space bar in name of metabolites. This is because it is more practical to name the variables (metabolite name) for MATLAB without a spacebar. If there is a spacebar, MATLAB will detect it as two variables. Figure 5.4.3 – figure 5.4.4 show the implemented of code for selecting from combobox and appearing of groupbox and error handling for user key in.

```
private void cmbMet_SelectedIndexChanged(object sender, EventArgs e)
      {
          grpMetName.Visible = true;
          switch (cmbMet.SelectedIndex)
          {
              case 0:
                  grpMetName01.Visible = true;
                  grpMetName02.Visible = true;
                  grpMetName03.Visible = false;
                  grpMetName04.Visible = false;
                  grpMetName05.Visible = false;
                  grpMetName06.Visible = false;
                  grpMetName07.Visible = false;
                  grpMetName08.Visible = false;
```

```
                    grpMetName09.Visible = false;
                    break;

                case 1:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = false;
                    grpMetName05.Visible = false;
                    grpMetName06.Visible = false;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;

                case 2:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = true;
                    grpMetName05.Visible = false;
                    grpMetName06.Visible = false;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;

                case 3:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = true;
                    grpMetName05.Visible = true;
                    grpMetName06.Visible = false;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;

                case 4:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = true;
                    grpMetName05.Visible = true;
                    grpMetName06.Visible = true;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;

                case 5:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = true;
                    grpMetName05.Visible = true;
                    grpMetName06.Visible = true;
                    grpMetName07.Visible = true;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;
```

```
            case 6:
                grpMetName01.Visible = true;
                grpMetName02.Visible = true;
                grpMetName03.Visible = true;
                grpMetName04.Visible = true;
                grpMetName05.Visible = true;
                grpMetName06.Visible = true;
                grpMetName07.Visible = true;
                grpMetName08.Visible = true;
                grpMetName09.Visible = false;
                break;

            case 7:
                grpMetName01.Visible = true;
                grpMetName02.Visible = true;
                grpMetName03.Visible = true;
                grpMetName04.Visible = true;
                grpMetName05.Visible = true;
                grpMetName06.Visible = true;
                grpMetName07.Visible = true;
                grpMetName08.Visible = true;
                grpMetName09.Visible = true;
                break;
        }
    }
```

Figure 5.4.3: Code implemented for Groupbox Selected Changed

The code for groupbox selected changed is for visibility of groupbox which consist of metabolites' textbox. This textbox is part that allowed user to key-in his/her metabolites' name. On the other hand, error handlers are for user to alert on which part that he / she missed out. Once the user saw the error message, he / she fixed the problem. Figure 5.4.4 shows the implementation of error handler

```
private void txtMetName1_Validated(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtMetName1.Text))
        {
            this.errorProvider1.SetError(txtMetName1, "This field must contain
                                                        text");
        }

        else
        {
            this.errorProvider1.SetError(txtMetName1, "");
        }
    }
```

```
private void txtMetName2_Validated(object sender, EventArgs e)
      {
          if (string.IsNullOrWhiteSpace(txtMetName2.Text))
          {
              this.errorProvider1.SetError(txtMetName2, "This field must contain
                                                          text");
          }

          else
          {
              this.errorProvider1.SetError(txtMetName2, "");
          }
      }
private void txtMetName3_Validated(object sender, EventArgs e)
      {
          if (string.IsNullOrWhiteSpace(txtMetName3.Text))
          {
              this.errorProvider1.SetError(txtMetName3, "This field must contain
                                                          text");
          }

          else
          {
              this.errorProvider1.SetError(txtMetName3, "");
          }
      }
private void txtMetName4_Validated(object sender, EventArgs e)
      {
          if (string.IsNullOrWhiteSpace(txtMetName4.Text))
          {
              this.errorProvider1.SetError(txtMetName4, "This field must contain
                                                          text");
          }

          else
          {
              this.errorProvider1.SetError(txtMetName4, "");
          }
      }
private void txtMetName5_Validated(object sender, EventArgs e)
      {
          if (string.IsNullOrWhiteSpace(txtMetName5.Text))
          {
              this.errorProvider1.SetError(txtMetName5, "This field must contain
                                                          text");
          }

          else
          {
              this.errorProvider1.SetError(txtMetName5, "");
          }
      }
private void txtMetName6_Validated(object sender, EventArgs e)
      {
          if (string.IsNullOrWhiteSpace(txtMetName6.Text))
          {
              this.errorProvider1.SetError(txtMetName6, "This field must contain
                                                          text");
          }
```

```
            else
            {
                this.errorProvider1.SetError(txtMetName6, "");
            }
        }

    private void txtMetName7_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMetName7.Text))
            {
                this.errorProvider1.SetError(txtMetName7, "This field must contain
                                                           text");
            }

            else
            {
                this.errorProvider1.SetError(txtMetName7, "");
            }
        }

    private void txtMetName8_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMetName8.Text))
            {
                this.errorProvider1.SetError(txtMetName8, "This field must contain
                                                           text");
            }

            else
            {
                this.errorProvider1.SetError(txtMetName8, "");
            }
        }

    private void txtMetName9_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMetName9.Text))
            {
                this.errorProvider1.SetError(txtMetName9, "This field must contain
                                                           text");
            }

            else
            {
                this.errorProvider1.SetError(txtMetName9, "");
            }
        }
```

Figure 5.4.4: Code implemented for Error Handler capturing null or white space

Based on figure 5.4.4, the textbox of each respective metabolite are undergoing validated process. If user left the textbox with empty or null space, the system will produce red symbol beside of empty textbox.

5.2.2.2 Mass Balance Page

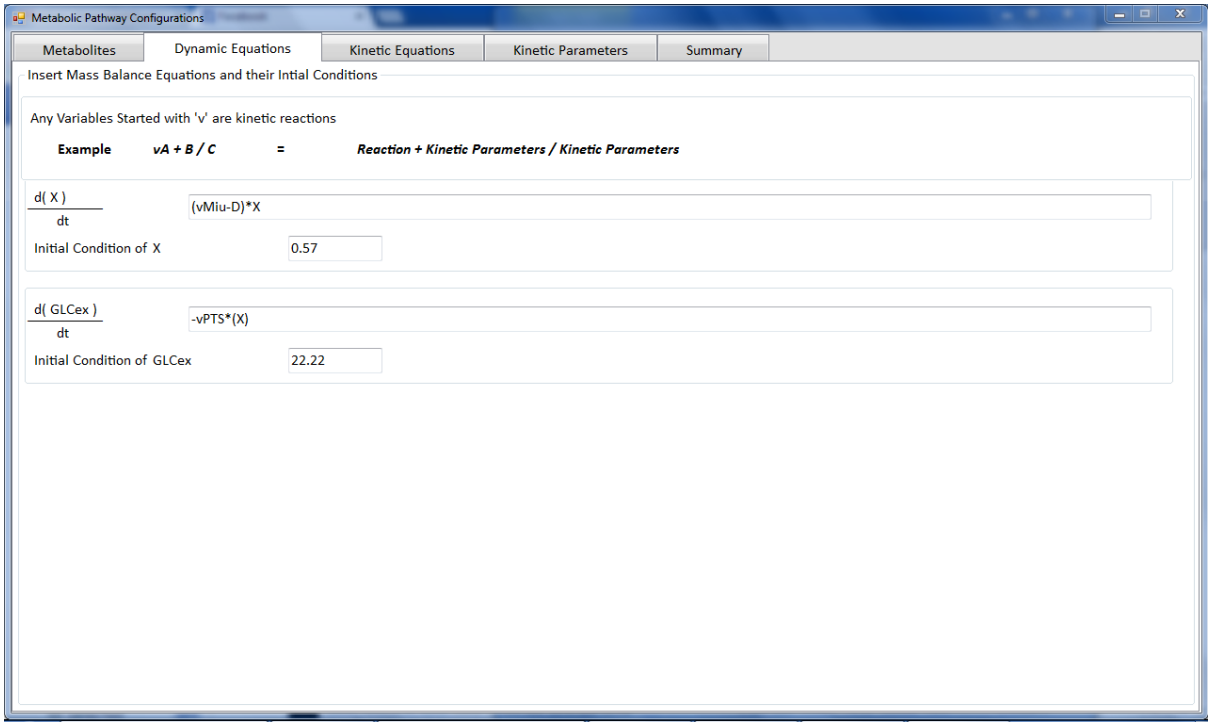


Figure 5.5.1: Screenshot on inputting dynamic equations.

Figure 5.5.1 shows the screenshot on inputting dynamic equations. The formats of dynamic equations are:-

$$\frac{dC_i}{dt} = \sum_j v_{ij}\, r_j - \mu C_i \qquad \text{(eq.01)}$$

where C_i is the concentration of metabolite i, v_{ij} is the stoichiometric coefficient in the reaction j, r_i represents the rate of formation of component i, $-\mu C_j$ represent the reduction of concentration by dilution due to increase in cell volume.

$$\frac{dX}{dt} = (\mu - D)X \qquad \text{(eq. 1.1)}$$

where X is the concentration of the biomass and μ is the specific growth rate (h^{-1}). From this equation, we can see that the specific growth rate is equal to the dilution rate at steady state condition. Therefore, the specific growth rate will be different as the dilution rate is changed.

User needs to bear in mind that the textbox of each dynamic equations are case sensitive. Each variable in string of equations are parser to kinetic equations and kinetic parameters. Any variable started with 'v' is categorized as kinetic equations while any variable is not started with 'v' are kinetic parameters. Figure 5.5.2 shows the implemented code for dynamic equations page.

```
public void TextTokenizer(string equation)
      {
          Equation = equation.Trim();
          char[] delimiters = new char[] { '+', '-', '/', '*', '(', ')', '[', ']',
                                           '^' };

          if (String.IsNullOrEmpty(Equation) == true) return;
          equations = Equation.Split(delimiters)
                              .Distinct()
                              .Where(x => x != string.Empty).ToArray();

          analyzeEquations(equations);
      }

private void analyzeEquations(string[] eq)
      {
          int charIndex = 0;
          for (int i = 0; i < eq.Length; i++)
          {
              char c = eq[i][charIndex];
              if (c != 'v') // ensure that the chunk string is for parameter
              {
                  Tokencombine += "+" + eq[i].ToString(); // combine after rearrange
                                                             for kinetic parameters
              }
              else
              {
                  combines += "+" + eq[i].ToString(); // combine after rearrange for
                                                         kinetic equations
              }
          }
          parentToken = new KineticEquation(combines);  // Token the kinetic
                                                           equations
          parentToken = new KineticParameter(Tokencombine); // Token the kinetic
                                                               parameters
      }
```

Figure 5.5.2: Tokenize equations according to respective categories

Based on figure 5.5.2, the equations are split and undergo looping process in order to categories the string chunk according to right categories. Once done, the equations are combined together according to respective categories and been undergo same process to ensure the string chunks are in right categories.

5.2.2.3 Kinetic Equations Page

Inside kinetic equations, user needs to insert the respective kinetic equations based on the part of string that been chunked on dynamic equations. Figure 5.5.3 shows the sample illustration of inserting kinetic equations.

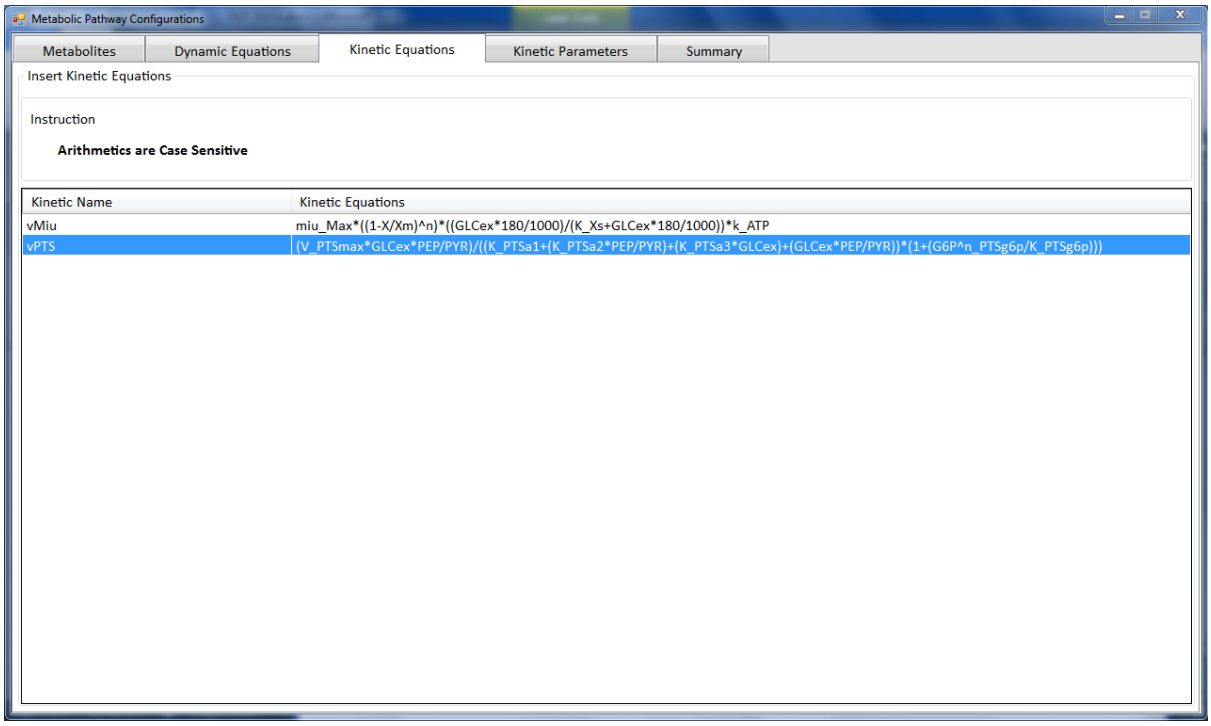


Figure 5.5.3: Illustration of inserting kinetic equations

Based on figure 5.5.3, user needs to carefully insert the kinetic equations as the arithmetic is case sensitive. For prototype level, the handling error is not included. Therefore, user must carefully insert the equations. Figure 5.5.4 shows the implemented code for kinetic equations page

```
public ParentToken(string equation)
      {
          schemaForm = new SchemaName();
          connStr = "datasource=localhost;port=3306;username=root;password=root;";
          conn = new MySqlConnection(connStr);
          command = conn.CreateCommand();
          database = schemaForm.getData;

          Equation = equation.Trim();
          char[] delimiters = new char[] { '+', '-', '/', '*', '(', ')', '[', ']',
                                           '^'};
```

```
            if (String.IsNullOrEmpty(Equation) == true) return;
            equations = Equation.Split(delimiters)
                                .Distinct()
                                .Where(x => x != string.Empty).ToArray();
        }
```

Figure 5.5.4: Code implementation for kinetic equations based on tokenize code

Note that the code for tokenize is same as figure 5.5.4. This is to ensure that there are only kinetic equations that been placed in kinetic equation page. Once the kinetic equations are inserted, they are then been parser in order to place it in kinetic parameters page.

5.2.2.4 Kinetic Parameters Page

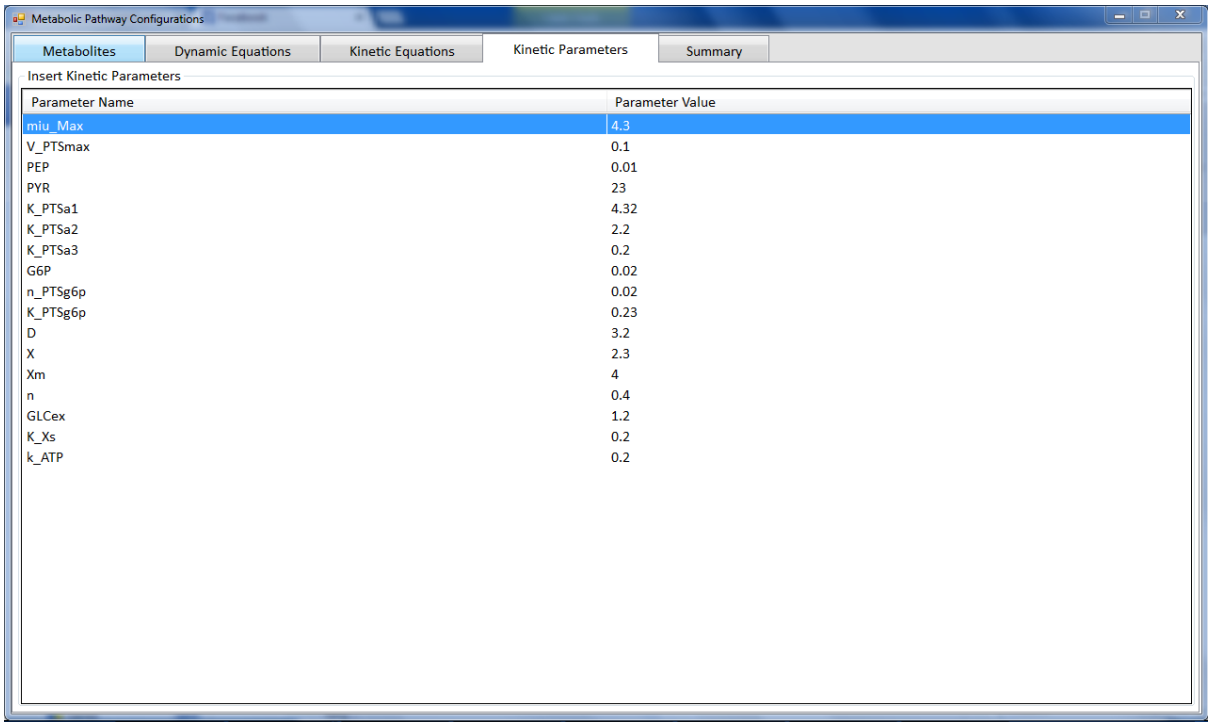


Figure 5.5.5: Illustration of Kinetic Parameter's page

Figure 5.5.5 shows the illustration of kinetic parameter's page. The parameter value format is case sensitive. User can only insert double or integer value for each parameter. If user inserted wrongly, the alert message box will pop out. The parameter values are bind and update with database as user inserted the values. The database parts will be discussed in section 5.4.

5.2.3 Graphs configurations Module

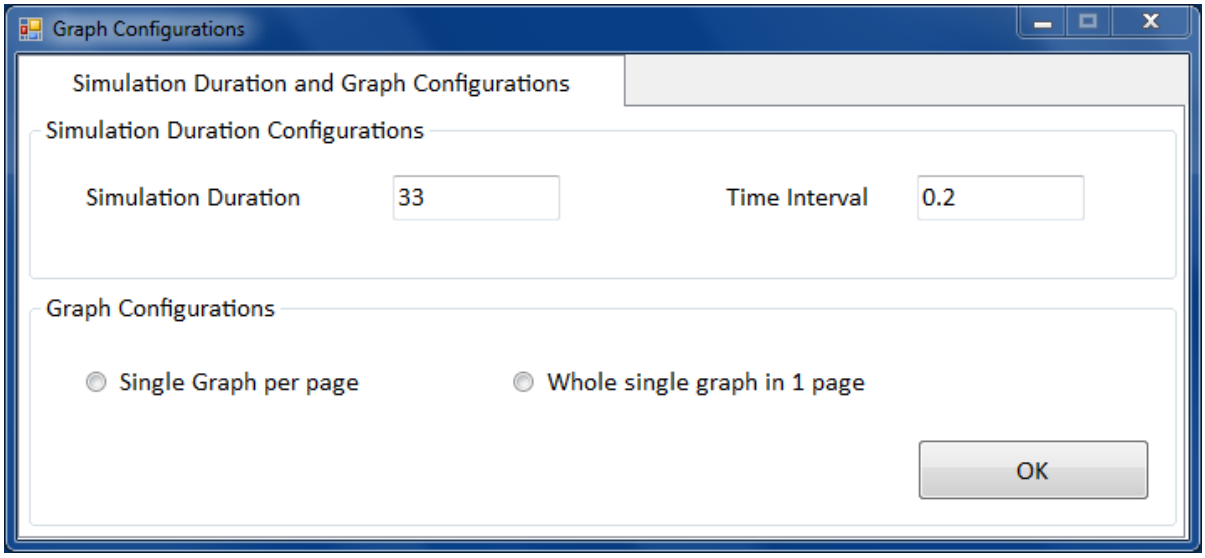


Figure 5.6.1: Graph configuration module

Figure 5.6.1 shows the graph configuration module for dynamic simulation tools. There are two panel involved in this module. The first panel is about simulation duration configurations while the second panel is about graph configurations.

For graph configurations, the simulation duration must be bigger than time interval as the time interval is 'time-step' for simulation from started and end by simulation duration. The simulation duration and time interval is case sensitive where user needs to input double or positive integer values. Meanwhile, the error message will popup if user enters time interval. Figure 5.6.2 shows the example of error handling if user insert time interval value bigger than simulation durations.

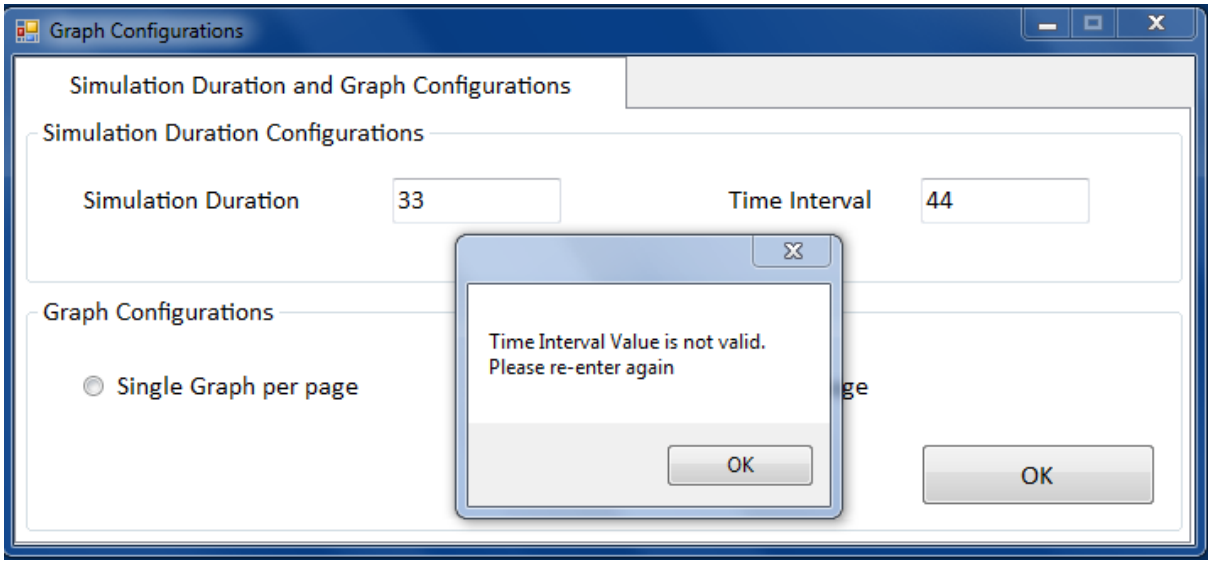


Figure 5.6.2: Error Handling in Graph Configurations module

Figure 5.6.3 shows the implementation code for simulation configurations. The first part of code is about the keypress of the textbox.

```
private void txtSimDuration_KeyPress(object sender, KeyPressEventArgs e)
      {
          // allows 0-9, backspace, and decimal
          if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8 && e.KeyChar !=
                46))
          {
              e.Handled = true;
              return;
          }

          // checks to make sure only 1 decimal is allowed
          if (e.KeyChar == 46)
          {
              if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                  e.Handled = true;
          }
      }

      private void txtTimeInterval_KeyPress(object sender, KeyPressEventArgs e)
      {
          // allows 0-9, backspace, and decimal
          if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8 && e.KeyChar !=
                46))
          {
              e.Handled = true;
              return;
          }
```

```
            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }

        private void txtTimeInterval_Leave(object sender, EventArgs e)
        {
            double interval = double.Parse(txtTimeInterval.Text);
            double max = double.Parse(txtSimDuration.Text);
            if (interval > max)
            {
                MessageBox.Show("Time Interval Value is not valid.\nPlease re-enter
                                 again");
                txtTimeInterval.Text = "0";
            }
        }
```

Figure 5.6.3: Implementation code for simulation configurations

```
private void btnOKSim_Click(object sender, EventArgs e)
        {
            if (rdNineGraphs.Checked)
            {
                file.ConfigureMain(this);       //  for 9 graph
                file.ConfigureMainSingle(this); //  for single graph
                matlab.Execute("ODE.m;");
                matlab.Execute("Main.m;");

                frmLog log = new frmLog();
                log.Show();
                log.txtMatlabLog.AppendText(matlab.Execute("ODE"));
                log.txtMatlabLog.AppendText(matlab.Execute("Main"));
                log.txtMatlabLog.AppendText(matlab.Execute("pwd"));

            }

            if (rdSingleGraph.Checked)
            {
                file.ConfigureMain(this);       //  for 9 graph
                file.ConfigureMainSingle(this); //  for single graph
                matlab.Execute("ODE.m;");
                matlab.Execute("MainSingle.m;");
                frmLog log = new frmLog();
                log.Show();
                log.txtMatlabLog.AppendText(matlab.Execute("ODE"));
                log.txtMatlabLog.AppendText(matlab.Execute("MainSingle"));
                log.txtMatlabLog.AppendText(matlab.Execute("pwd"));
            }
        }
```

Figure 5.6.4: Implementation code for graph configurations

Based on figure 5.6.4, the graphs appearance are based on user chosen of graph types. It can be either single or all in one graph.

5.3 Development of Database using MySQL

Database is a collection of organized data that play the most important role in the development of the dynamic simulation tools. In this project, the database is used to store the dynamic equations, kinetic equations and parameters. If user want to use his / her last time project, he/she can easily retrieved it based calling the respective database name. The database is developed based on user defined and it is created in MySQL database software. Figure 5.7 shows the database software used for this project.

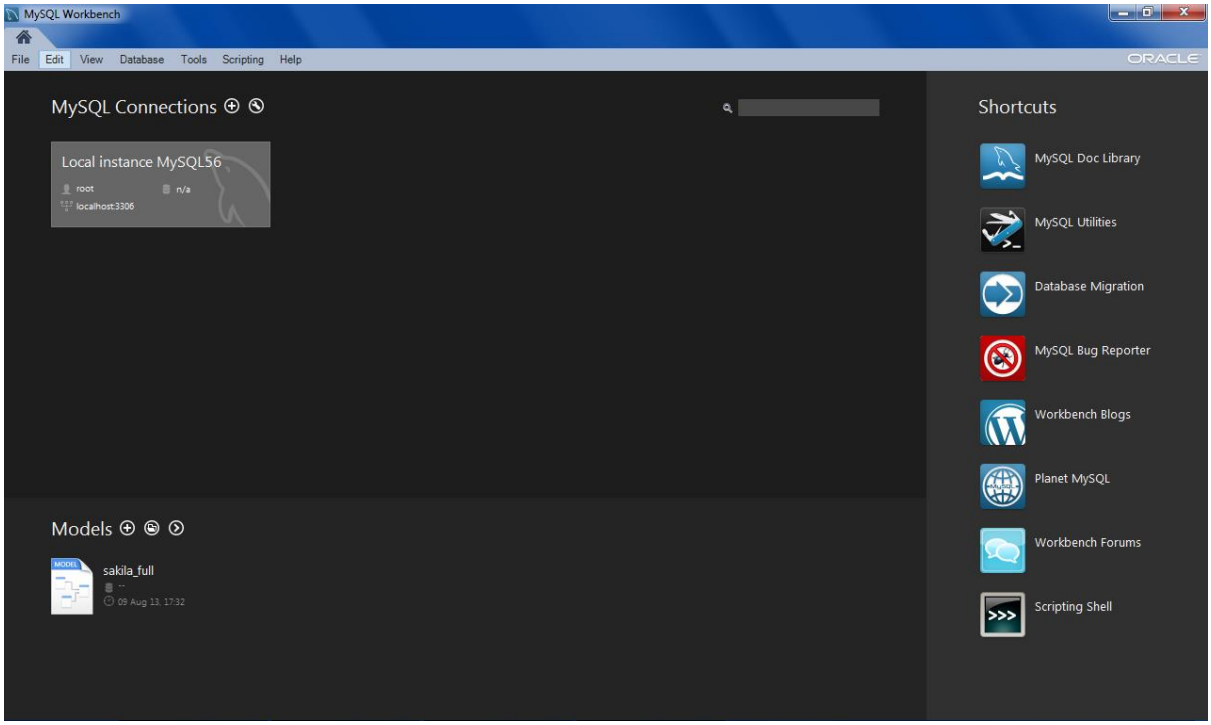


Figure 5.7: MySQL interface

In this project, the database is created when user defined entire configuration in metabolites configurations. Figure 5.8.1 shows the code implementation for database.

```
public void Database()
      {
          string connStr = "datasource=localhost;port=3306;username=root;
                            password=root;";
          string database = schemaForm.getData;
          conn = new MySqlConnection(connStr);

          MySqlCommand command = conn.CreateCommand();
          conn.Open();
          command.CommandText = "DROP DATABASE IF EXISTS " + database;
          command.ExecuteNonQuery();
          command.CommandText = "CREATE DATABASE " + database;
          command.ExecuteNonQuery();

          //Coding for PRELIMINARY METABOLITES
          command.CommandText = "CREATE TABLE " + database +
                                ".Metabolites(" +
                                "MetaboliteID VARCHAR(30) NOT NULL," +
                                "Metabolite_Name VARCHAR(600) NULL," +
                                "Mass_Balance VARCHAR(600) NULL," +
                                "Initial_Condition DOUBLE NULL," +
                                "PRIMARY KEY (MetaboliteID));";
          command.ExecuteNonQuery();

          command.CommandText = "INSERT INTO " + database +
                                ".Metabolites " +
                                "(MetaboliteID)" +
                                "VALUES " +
                                "('Metabolite01');";
          command.ExecuteNonQuery();

          command.CommandText = "INSERT INTO " + database +
                                ".Metabolites " +
                                "(MetaboliteID)" +
                                "VALUES " +
                                "('Metabolite02');";
          command.ExecuteNonQuery();

          command.CommandText = "INSERT INTO " + database +
                                ".Metabolites " +
                                "(MetaboliteID)" +
                                "VALUES " +
                                "('Metabolite03');";
          command.ExecuteNonQuery();

          command.CommandText = "INSERT INTO " + database +
                                ".Metabolites " +
                                "(MetaboliteID)" +
                                "VALUES " +
                                "('Metabolite04');";
          command.ExecuteNonQuery();

          command.CommandText = "INSERT INTO " + database +
                                ".Metabolites " +
                                "(MetaboliteID)" +
                                "VALUES " +
                                "('Metabolite05');";
          command.ExecuteNonQuery();
```

```
            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite06');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite07');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite08');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite09');";
            command.ExecuteNonQuery();

            command.CommandText = "CREATE TABLE " + database +
                                  ".AlteredMetabolites(" +
                                  "AMetaboliteID DOUBLE NOT NULL," +
                                  "AMetaboliteValue DOUBLE NULL," +
                                  "PRIMARY KEY (AMetaboliteID));";
            command.ExecuteNonQuery();

            conn.Close();

            TableEnzymeKinetic();
            TableKineticParameters();
        }

    public void TableEnzymeKinetic()
        {
            string connStr = "datasource=localhost;port=3306;username=root;
                              password=root;";
            string database = schemaForm.getData;
            conn = new MySqlConnection(connStr);

            MySqlCommand command = conn.CreateCommand();
            conn.Open();
            command.CommandText = "CREATE TABLE " + database +
                                  ".EnzymeKinetics(" +
                                  "KineticID VARCHAR(30) NOT NULL," +
                                  "Kinetic_Name VARCHAR(800) NULL," +
                                  "Equations VARCHAR(800) NULL," +
                                  "PRIMARY KEY (KineticID));";
            command.ExecuteNonQuery();

            conn.Close();
        }
```

```
public void TableKineticParameters()
       {
           string connStr = "datasource=localhost;port=3306;username=root;
                             password=root;";
           string database = schemaForm.getData;
           conn = new MySqlConnection(connStr);

           MySqlCommand command = conn.CreateCommand();
           conn.Open();
           command.CommandText = "CREATE TABLE " + database +
                                 ".KineticParameters(" +
                                 "ParameterID VARCHAR(30) NOT NULL," +
                                 "Parameter_Name VARCHAR(800) NULL," +
                                 "Parameter_Value DOUBLE NULL," +
                                 "PRIMARY KEY (ParameterID));";
           command.ExecuteNonQuery();

           command.CommandText = "CREATE TABLE " + database +
                                 ".AlteredKineticParameters(" +
                                 "AParameterID DOUBLE NOT NULL," +
                                 "AParameterValue DOUBLE NULL," +
                                 "PRIMARY KEY (AParameterID));";
           command.ExecuteNonQuery();
           conn.Close();
       }
       #endregion
```

Figure 5.8.1: Implementation in Creation of Database

Once the database is created, user can insert the data and edit it by help of coding in figure 5.8.2. The database is update whenever user edited it. Note that inside UpdateMassBalance class, there is one variable that combine and hold a string that been tokenize which is explained in section in 5.2.2.2

```
 public Database(frmMetCon frm2) // parent of database
       {
           schemaForm = new SchemaName();
           connStr = "datasource=localhost;port=3306;username=root;password=root;";
           database = schemaForm.getData;
           conn = new MySqlConnection(connStr);
           command = conn.CreateCommand();
       }


class UpdateMetaboliteName : Database
    {
       public UpdateMetaboliteName(frmMetCon frm2) // Update Metabolite Name in
                                                      database
           : base(frm2)
       {
            conn.Open();
```

```
            try
            {
                if (!string.IsNullOrWhiteSpace(frm2.txtMetName1.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "'" + frm2.txtMetName1.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite01'";
                    command.ExecuteNonQuery();
                }
                if (!string.IsNullOrWhiteSpace(frm2.txtMetName2.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                        ".Metabolites " +
                                        "SET Metabolite_Name = " +
                                        "'" + frm2.txtMetName2.Text + "'" +
                                        "WHERE MetaboliteID = 'Metabolite02'";
                    command.ExecuteNonQuery();
                }

                if (!string.IsNullOrWhiteSpace(frm2.txtMetName3.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "'" + frm2.txtMetName3.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite03'";
                    command.ExecuteNonQuery();
                }

                if (!string.IsNullOrWhiteSpace(frm2.txtMetName4.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "'" + frm2.txtMetName4.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite04'";
                    command.ExecuteNonQuery();
                }

                if (!string.IsNullOrWhiteSpace(frm2.txtMetName5.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "'" + frm2.txtMetName5.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite05'";
                    command.ExecuteNonQuery();
                }

                if (!string.IsNullOrWhiteSpace(frm2.txtMetName6.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "'" + frm2.txtMetName6.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite06'";
                    command.ExecuteNonQuery();
                }
```

```
                    if (!string.IsNullOrWhiteSpace(frm2.txtMetName7.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "'" + frm2.txtMetName7.Text + "'" +
                                              "WHERE MetaboliteID = 'Metabolite07'";
                        command.ExecuteNonQuery();
                    }

                    if (!string.IsNullOrWhiteSpace(frm2.txtMetName8.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "'" + frm2.txtMetName8.Text + "'" +
                                              "WHERE MetaboliteID = 'Metabolite08'";
                        command.ExecuteNonQuery();
                    }

                    if (!string.IsNullOrWhiteSpace(frm2.txtMetName9.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "'" + frm2.txtMetName9.Text + "'" +
                                              "WHERE MetaboliteID = 'Metabolite09'";
                        command.ExecuteNonQuery();
                    }

                    if (string.IsNullOrWhiteSpace(frm2.txtMetName1.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "' '" +
                                              "WHERE MetaboliteID = 'Metabolite01'";
                        command.ExecuteNonQuery();
                    }

                    if (string.IsNullOrWhiteSpace(frm2.txtMetName2.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "' '" +
                                              "WHERE MetaboliteID = 'Metabolite02'";
                        command.ExecuteNonQuery();
                    }

                    if (string.IsNullOrWhiteSpace(frm2.txtMetName3.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "' '" +
                                              "WHERE MetaboliteID = 'Metabolite03'";
                        command.ExecuteNonQuery();
                    }
```

```
                if (string.IsNullOrWhiteSpace(frm2.txtMetName4.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "' '" +
                                          "WHERE MetaboliteID = 'Metabolite04'";
                    command.ExecuteNonQuery();
                }

                if (string.IsNullOrWhiteSpace(frm2.txtMetName5.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "' '" +
                                          "WHERE MetaboliteID = 'Metabolite05'";
                    command.ExecuteNonQuery();
                }

                if (string.IsNullOrWhiteSpace(frm2.txtMetName6.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "' '" +
                                          "WHERE MetaboliteID = 'Metabolite06'";
                    command.ExecuteNonQuery();
                }

                if (string.IsNullOrWhiteSpace(frm2.txtMetName7.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "' '" +
                                          "WHERE MetaboliteID = 'Metabolite07'";
                    command.ExecuteNonQuery();
                }

                if (string.IsNullOrWhiteSpace(frm2.txtMetName8.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "' '" +
                                          "WHERE MetaboliteID = 'Metabolite08'";
                    command.ExecuteNonQuery();
                }

                if (string.IsNullOrWhiteSpace(frm2.txtMetName9.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Metabolite_Name = " +
                                          "' '" +
                                          "WHERE MetaboliteID = 'Metabolite09'";
                    command.ExecuteNonQuery();
                }
```

```
                conn.Close();
                bind.BindingMetaboliteName(frm2);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }

    class UpdateMassBalance : Database
    {
        public UpdateMassBalance(frmMetCon frm2)  //Update Mass Balance in database
                  : base(frm2)                      and parse the equations
        {
            try
            {
                conn.Open();
                if (!string.IsNullOrWhiteSpace(frm2.txtMet01.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Mass_Balance = " +
                                          "'" + frm2.txtMet01.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite01'";
                    command.ExecuteNonQuery();
                    combineText += "+" + frm2.txtMet01.Text;
                }
                if (!string.IsNullOrWhiteSpace(frm2.txtMet02.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Mass_Balance = " +
                                          "'" + frm2.txtMet02.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite02'";
                    command.ExecuteNonQuery();
                    combineText += "+" + frm2.txtMet02.Text;
                }

                if (!string.IsNullOrWhiteSpace(frm2.txtMet03.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Mass_Balance = " +
                                          "'" + frm2.txtMet03.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite03'";
                    command.ExecuteNonQuery();
                    combineText += "+" + frm2.txtMet03.Text;
                }

                if (!string.IsNullOrWhiteSpace(frm2.txtMet04.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                          ".Metabolites " +
                                          "SET Mass_Balance = " +
                                          "'" + frm2.txtMet04.Text + "'" +
                                          "WHERE MetaboliteID = 'Metabolite04'";
                    command.ExecuteNonQuery();
                    combineText += "+" + frm2.txtMet04.Text;
                }
```

```
            if (!string.IsNullOrWhiteSpace(frm2.txtMet05.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Mass_Balance = " +
                                      "'" + frm2.txtMet05.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite05'";
                command.ExecuteNonQuery();
                combineText += "+" + frm2.txtMet05.Text;
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMet06.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Mass_Balance = " +
                                      "'" + frm2.txtMet06.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite06'";
                command.ExecuteNonQuery();
                combineText += "+" + frm2.txtMet06.Text;
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMet07.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Mass_Balance = " +
                                      "'" + frm2.txtMet07.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite07'";
                command.ExecuteNonQuery();
                combineText += "+" + frm2.txtMet07.Text;
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMet08.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Mass_Balance = " +
                                      "'" + frm2.txtMet08.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite08'";
                command.ExecuteNonQuery();
                combineText += "+" + frm2.txtMet08.Text;
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMet09.Text))
            {
                command.CommandText = "UPDATE " + database +
                                     ".Metabolites " +
                                     "SET Mass_Balance = " +
                                     "'" + frm2.txtMet09.Text + "'" +
                                     "WHERE MetaboliteID = 'Metabolite09'";
                command.ExecuteNonQuery();
                combineText += "+" + frm2.txtMet09.Text;
            }


            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet1.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                double.Parse(frm2.txtInitialConMet1.Text) + "'" +
                                      "WHERE MetaboliteID = 'Metabolite01'";
```

```
                command.ExecuteNonQuery();
            }

            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet2.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                double.Parse(frm2.txtInitialConMet2.Text) + "'" +
                                      "WHERE MetaboliteID = 'Metabolite02'";
                command.ExecuteNonQuery();
            }

            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet3.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                double.Parse(frm2.txtInitialConMet3.Text) + "'" +
                                      "WHERE MetaboliteID = 'Metabolite03'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet4.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                double.Parse(frm2.txtInitialConMet4.Text) + "'" +
                                      "WHERE MetaboliteID = 'Metabolite04'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet5.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                double.Parse(frm2.txtInitialConMet5.Text) + "'" +
                                      "WHERE MetaboliteID = 'Metabolite05'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet6.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                double.Parse(frm2.txtInitialConMet6.Text) + "'" +
                                      "WHERE MetaboliteID = 'Metabolite06'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet7.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
```

```
                    double.Parse(frm2.txtInitialConMet7.Text) + "'" +
                                        "WHERE MetaboliteID = 'Metabolite07'";
                    command.ExecuteNonQuery();
                }
                if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet8.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                        ".Metabolites " +
                                        "SET Initial_Condition = " +
                                        "'" +
                    double.Parse(frm2.txtInitialConMet8.Text) + "'" +
                                        "WHERE MetaboliteID = 'Metabolite08'";
                    command.ExecuteNonQuery();
                }
                if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet9.Text))
                {
                    command.CommandText = "UPDATE " + database +
                                        ".Metabolites " +
                                        "SET Initial_Condition = " +
                                        "'" +
                    double.Parse(frm2.txtInitialConMet9.Text) + "'" +
                                        "WHERE MetaboliteID = 'Metabolite09'";
                    command.ExecuteNonQuery();
                }
                conn.Close();
                tokenNew = new Tokenizer();
                tokenNew.TextTokenizer(combineText);
                bind.BindingKineticEquations(frm2); // Binding for summary of Kinetic
                                                       Equations and
                                                    // at tab page of Kinetic
                                                       Equations
                bind.BindingMassBalance(frm2); // Binding for summary of Mass Balance
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }

    class UpdateKineticEquations : Database
    {
        public UpdateKineticEquations(frmMetCon frm2)  // Updating Kinetic Equation in
            : base(frm2) datagridview
        {
            int currentRow = frm2.dbKineticName.CurrentCell.RowIndex;
            string data, label;
            data = frm2.dbKineticName.Rows[currentRow].Cells[1].ToString();
            label = frm2.dbKineticName.Rows[currentRow].Cells[0].ToString();
            conn.Open();
            try
            {
                command.CommandText = "UPDATE " + database +
                                    ".EnzymeKinetics " +
                                    "SET Equations = " +
                                    "'" + data + "'" +
                                    "WHERE Kinetic_Name = " +
                                    "'" + label + "'";
                command.ExecuteNonQuery();
```

```
                conn.Close();
                bind.BindingKineticEquations(frm2);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
```

Figure 5.8.2: Coding for Updating the Database

In order to show the database to user, the implementation codes for binding are used. The database is send to 'dtable' first then it is bound to datagridview. Figure 5.8.3 shows the code implemented for binding the database

```
public void BindingMetaboliteName(frmMetCon form2)
      {
          schemaForm = new SchemaName();
          connStr = "datasource=localhost;port=3306;username=root;password=root;";
          conn = new MySqlConnection(connStr);
          command = conn.CreateCommand();
          database = schemaForm.getData;

          try
          {
              dtable = new DataTable();
              bindingSource = new BindingSource();

              conn.Open();
              form2.dbSumMetName.DataSource = null;
              form2.dbSumMetName.ResetBindings();
              form2.dbSumMetName.Rows.Clear();
              form2.dbSumMetName.AutoGenerateColumns = false;
              command.CommandText = "SELECT MetaboliteID, Metabolite_Name FROM " +
                                                                    database +
                                    ".Metabolites " +
                                    "WHERE Metabolite_Name IS NOT NULL " +
                                    "AND Metabolite_Name <> '';"; //Select not empty
                                                                    metabolite name
              MySqlDataReader dr = command.ExecuteReader +
                                   (CommandBehavior.CloseConnection);
              dtable.Load(dr);
              for (int i = 0; i < dtable.Rows.Count; i++)
              {
                  form2.dbSumMetName.Rows.Add();
                  form2.dbSumMetName.Rows[form2.dbSumMetName.Rows.Count - 1].Cells
                  + ["MetaboliteID"].Value = dtable.Rows[i]
                  + ["MetaboliteID"].ToString();
                  form2.dbSumMetName.Rows[form2.dbSumMetName.Rows.Count - 1].Cells
                  + ["Metabolite_Name"].Value = dtable.Rows[i]
                  + ["Metabolite_Name"].ToString();
              }
```

```
            conn.Close();
        }



        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    public void BindingMassBalance(frmMetCon frm)
    {
        schemaForm = new SchemaName();
        connStr = "datasource=localhost;port=3306;username=root;password=root;";
        conn = new MySqlConnection(connStr);
        command = conn.CreateCommand();
        database = schemaForm.getData;

        try
        {
            dtable = new DataTable();
            bindingSource = new BindingSource();
            conn.Open();
            frm.dbSumMassBal.DataSource = null;
            frm.dbSumMassBal.ResetBindings();
            frm.dbSumMassBal.Rows.Clear();
            frm.dbSumMassBal.AutoGenerateColumns = false;
            command.CommandText = "SELECT Metabolite_Name, Mass_Balance,
                                   Initial_Condition FROM " + database +
                                  ".Metabolites " +
                                  "WHERE Metabolite_Name IS NOT NULL " +
                                  "AND Metabolite_Name <> '';";
            MySqlDataReader dr = command.ExecuteReader
                                + (CommandBehavior.CloseConnection);
            dtable.Load(dr);
            for (int i = 0; i < dtable.Rows.Count; i++)
            {
                frm.dbSumMassBal.Rows.Add();
                frm.dbSumMassBal.Rows[frm.dbSumMassBal.Rows.Count - 1].Cells
                + ["MassBalances"].Value = "d( " + dtable.Rows[i]
                + ["Metabolite_Name"].ToString()
                + " )/dt";
                frm.dbSumMassBal.Rows[frm.dbSumMassBal.Rows.Count - 1].Cells
                + ["MassBalanceEquations"].Value = dtable.Rows[i]
                + ["Mass_Balance"].ToString();
                frm.dbSumMassBal.Rows[frm.dbSumMassBal.Rows.Count - 1].Cells
                + ["InitialConditions"].Value = dtable.Rows[i]
                + ["Initial_Condition"].ToString();
            }
            conn.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
```

```
public void BindingKineticEquations(frmMetCon frm2)
{
    schemaForm = new SchemaName();
    connStr = "datasource=localhost;port=3306;username=root;password=root;";
    conn = new MySqlConnection(connStr);
    command = conn.CreateCommand();
    database = schemaForm.getData;


    try
    {
        dtable = new DataTable();
        bindingSource = new BindingSource();
        conn.Open();
        frm2.dbKineticName.DataSource = null;
        frm2.dbKineticName.ResetBindings();
        frm2.dbKineticName.Rows.Clear();
        frm2.dbKineticName.AutoGenerateColumns = false;
        command.CommandText = "SELECT Kinetic_Name, Equations FROM " +
                                                          database +
                               ".EnzymeKinetics " +
                               "WHERE CONCAT ('',Kinetic_Name*1)<>
                                Kinetic_Name";
        MySqlDataReader dr = command.ExecuteReader
                             + (CommandBehavior.CloseConnection);
        dtable.Load(dr);
        for (int i = 0; i < dtable.Rows.Count; i++)
        {
            frm2.dbKineticName.Rows.Add();
            frm2.dbKineticName.Rows[frm2.dbKineticName.Rows.Count - 1].Cells
            + ["Kinetic_Name"].Value = dtable.Rows[i]
            + ["Kinetic_Name"].ToString();
            frm2.dbKineticName.Rows[frm2.dbKineticName.Rows.Count - 1].Cells
            + ["Equations"].Value = dtable.Rows[i]["Equations"].ToString();
        }

        frm2.dbSumKinEq.DataSource = null;
        frm2.dbSumKinEq.ResetBindings();
        frm2.dbSumKinEq.Rows.Clear();
        frm2.dbSumKinEq.AutoGenerateColumns = false;
        for (int i = 0; i < dtable.Rows.Count; i++)
        {
            frm2.dbSumKinEq.Rows.Add();
            frm2.dbSumKinEq.Rows[frm2.dbSumKinEq.Rows.Count - 1].Cells
            + ["Kinetic_Names"].Value = dtable.Rows[i]
            + ["Kinetic_Name"].ToString();
            frm2.dbSumKinEq.Rows[frm2.dbSumKinEq.Rows.Count - 1].Cells
            + ["KEquations"].Value = dtable.Rows[i]["Equations"].ToString();
        }
        conn.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }

}
```

```
public void BindingKineticParameter(frmMetCon frm2)
{
    schemaForm = new SchemaName();
    connStr = "datasource=localhost;port=3306;username=root;password=root;";
    conn = new MySqlConnection(connStr);
    command = conn.CreateCommand();
    database = schemaForm.getData;

    try
    {
        #region Bind Parameter

        dtable = new DataTable();
        bindingSource = new BindingSource();
        conn.Open();
        frm2.dbParameter.DataSource = null;
        frm2.dbParameter.ResetBindings();
        frm2.dbParameter.Rows.Clear();
        frm2.dbParameter.AutoGenerateColumns = false;
        command.CommandText = "SELECT Parameter_Name, Parameter_Value FROM " +
                                                                    database +
                              ".KineticParameters " +
                              "WHERE CONCAT ('',Parameter_Name*1)<>
                               Parameter_Name";
        MySqlDataReader dr = command.ExecuteReader
                            + (CommandBehavior.CloseConnection);
        dtable.Load(dr);
        for (int i = 0; i < dtable.Rows.Count; i++)
        {
            frm2.dbParameter.Rows.Add();
            frm2.dbParameter.Rows[frm2.dbParameter.Rows.Count - 1].Cells
            + ["ParameterName"].Value = dtable.Rows[i]
            + ["Parameter_Name"].ToString();
            frm2.dbParameter.Rows[frm2.dbParameter.Rows.Count - 1].Cells
            + ["ParameterValue"].Value = dtable.Rows[i]
            + ["Parameter_Value"].ToString();
        }

        frm2.dbSumKinPar.DataSource = null;
        frm2.dbSumKinPar.ResetBindings();
        frm2.dbSumKinPar.Rows.Clear();
        frm2.dbSumKinPar.AutoGenerateColumns = false;
        for (int i = 0; i < dtable.Rows.Count; i++)
        {
            frm2.dbSumKinPar.Rows.Add();
            frm2.dbSumKinPar.Rows[frm2.dbSumKinPar.Rows.Count - 1].Cells
            + ["ParameterNames"].Value = dtable.Rows[i]
            + ["Parameter_Name"].ToString();
            frm2.dbSumKinPar.Rows[frm2.dbSumKinPar.Rows.Count - 1].Cells
            + ["KineticParameterValues"].Value = dtable.Rows[i]
            + ["Parameter_Value"].ToString();
        }
        int counter = dtable.Rows.Count;
        conn.Close();
        #endregion

    }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Figure 5.8.3: Code Implemented For Binding the Database

In addition, figure 5.8.4 shows the update code in datagridview. This code is important in updating the database as well as viewing in datagridview for user to see what he/she has inserted.

```
        #region Constructor
         public UpdateDataGrid(frmMetCon frm2)
         {
             schemaForm = new SchemaName();
             connStr = "datasource=localhost;port=3306;username=root;password=root;";
             database = schemaForm.getData;
             conn = new MySqlConnection(connStr);
             command = conn.CreateCommand();
         }
         #endregion

    }
    class UpdateBindingKineticEquations : UpdateDataGrid
    {
        public UpdateBindingKineticEquations(frmMetCon frm2)
            : base(frm2)
        {
            int rowIndex = frm2.dbKineticName.CurrentCell.RowIndex;
            int columnIndex = frm2.dbKineticName.CurrentCell.ColumnIndex;

            try
            {
                string kineticName = frm2.dbKineticName.Rows[rowIndex]
                                     .Cells[0].Value.ToString();
                string kineticEq = frm2.dbKineticName.Rows[rowIndex]
                                   .Cells[columnIndex].Value.ToString();
                conn.Open();
                command.CommandText = "UPDATE " + database +
                                      ".EnzymeKinetics " +
                                      "SET Equations = " +
                                      "'" + kineticEq + "'" +
                                      "WHERE Kinetic_Name = " +
                                      "'" + kineticName + "'";
                command.ExecuteNonQuery();
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
```

```
class UpdateBindingKineticParameter : UpdateDataGrid
{
    public UpdateBindingKineticParameter(frmMetCon frm2)
        : base(frm2)
    {
        int rowIndex = frm2.dbParameter.CurrentCell.RowIndex;
        int columnIndex = frm2.dbParameter.CurrentCell.ColumnIndex;

        try
        {
            string ParameterName = frm2.dbParameter.Rows[rowIndex]
                                   .Cells[0].Value.ToString();
            double ParameterValue = double.Parse(frm2.dbParameter.Rows[rowIndex]
                                    .Cells[columnIndex].Value.ToString());
            conn.Open();


            command.CommandText = "UPDATE " + database +
                                  ".KineticParameters " +
                                  "SET Parameter_Value = " +
                                  "'" + ParameterValue + "'" +
                                  "WHERE Parameter_Name = " +
                                  "'" + ParameterName + "'";
            command.ExecuteNonQuery();
            conn.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
```

Figure 5.8.4: Code for updating and binding database in datagridview

5.4 Development of MATLAB .m file

MATLAB is a numerical computing environment and it is suitable to create graph and calculate mathematical equations correctly. In this project, the equations and parameters are sent to .m file (MATLAB file format) and it used MATLAB engine to calculate the result. Figure 5.9 shows the code of creating .m file

```
    public void ConfigureMain(frmGraphConfig frm3)
    {
        try
        {
            int i = 0;
            int counter = 0;

            connStr = "datasource=localhost;port=3306;username=root;
                       password=root;";
```

```
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
            database = schemaForm.getData;

            FileLocation = path + "\\Main.m";

            dtable = new DataTable();

            file = new StreamWriter(FileLocation);

            file.Write("clc\nclear all;\nparameter;\ninitialCons;\n");
            file.Write("duration = " + frm3.txtSimDuration.Text + ";\n");
            file.Write("interval = " + frm3.txtTimeInterval.Text + ";\n");
            file.Write("time = [0:interval:duration];\n");

            conn.Open();



            command.CommandText = "SELECT Metabolite_Name FROM " + database +
                                  ".Metabolites " +
                                  "WHERE Metabolite_Name IS NOT NULL " +
                                  "AND Metabolite_Name <> '';";
            MySqlDataReader dr = command.ExecuteReader
                                 (CommandBehavior.CloseConnection);
            dtable.Load(dr);

            file.Write("init_C1 = [ ");

            for (i = 0; i < dtable.Rows.Count; i++)
            {
                file.Write("c" + dtable.Rows[i]
                + ["Metabolite_Name"].ToString() + " ");
            }
            file.Write("];\n[t, C1] = ode15s(@ODE, time, init_C1);\n");

            for (i = 0; i < dtable.Rows.Count; i++)
            {
                counter = i + 1;
                file.Write(dtable.Rows[i]["Metabolite_Name"].ToString()
                + "\t = C1(:," + counter + ");\n");
            }
            file.Write("figure(1), clf;\n");

            for (i = 0; i < dtable.Rows.Count; i++)
            {
                counter = i + 1;
                file.Write("subplot(33" + counter + "), plot(t, " +
                dtable.Rows[i]["Metabolite_Name"].ToString()
                + ",'k-','LineWidth',1.7), xlabel('time (h)'),
                ylabel('Concentration');\n");
            }
            file.Write("set(gcf, 'color', 'white');\n");
            file.Write("set(gcf, 'InvertHardCopy', 'off');\n");
            conn.Close();
            file.Close();
        }
```

```
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    public void ConfigureInitialCondition()
    {
        try
        {
            connStr = "datasource=localhost;port=3306;username=root;
                       password=root;";
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
            database = schemaForm.getData;
            FileLocation = path + "\\initialCons.m";

            file = new StreamWriter(FileLocation);

            dtable = new DataTable();
            conn.Open();



            command.CommandText = "SELECT Metabolite_Name,
                                   Initial_Condition FROM " + database +
                                  ".Metabolites " +
                                  "WHERE Metabolite_Name IS NOT NULL " +
                                  "AND Metabolite_Name <> '';";
            MySqlDataReader dr = command.ExecuteReader
                                 (CommandBehavior.CloseConnection);
            dtable.Load(dr);

            for (int i = 0; i < dtable.Rows.Count; i++)
            {
                file.Write("c" + dtable.Rows[i]
                ["Metabolite_Name"].ToString() + "\t= " +
                dtable.Rows[i]["Initial_Condition"].ToString() + "\n");
            }
            conn.Close();
            file.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    public void ConfigureParameter()
    {
        try
        {
            connStr = "datasource=localhost;port=3306;username=root;
                       password=root;";
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
            database = schemaForm.getData;
            FileLocation = path + "\\parameter.m";

            file = new StreamWriter(FileLocation);
```

```
                dtable = new DataTable();
                conn.Open();
                command.CommandText = "SELECT Parameter_Name, Parameter_Value FROM " +
                                                                              database +
                                      ".KineticParameters " +
                                      "WHERE CONCAT ('',Parameter_Name*1)<>
                                       Parameter_Name";
                MySqlDataReader dr = command.ExecuteReader
                                     (CommandBehavior.CloseConnection);
                dtable.Load(dr);

                for (int i = 0; i < dtable.Rows.Count; i++)
                {
                    file.Write(dtable.Rows[i]["Parameter_Name"].ToString() + "\t= " +
                    dtable.Rows[i]["Parameter_Value"].ToString() + ";\n");
                }
                conn.Close();
                file.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

    public void ConfigureODE()
        {
            try
            {
                int i = 0;
                connStr = "datasource=localhost;port=3306;username=root;
                           password=root;";
                conn = new MySqlConnection(connStr);
                command = conn.CreateCommand();
                command2 = conn.CreateCommand();
                database = schemaForm.getData;
                FileLocation = path + "\\ODE.m";

                file = new StreamWriter(FileLocation);
                file.Write("function y = ODE(t,C)\n");
                file.Write("parameter;\n");
                conn.Open();
                dtable2 = new DataTable();

                //Calling Kinetic Equations
                command.CommandText = "SELECT Kinetic_Name, Equations FROM " +
                                                                    database +
                                      ".EnzymeKinetics " +
                                      "WHERE CONCAT ('',Kinetic_Name*1)<>
                                       Kinetic_Name";
                MySqlDataReader dr2 = command.ExecuteReader
                                      (CommandBehavior.CloseConnection);
                dtable2.Load(dr2);

                for (i = 0; i < dtable2.Rows.Count; i++)
                {
                    file.Write(dtable2.Rows[i]["Kinetic_Name"].ToString() + "\t= " +
                    dtable2.Rows[i]["Equations"].ToString() + ";\n");
                }
                conn.Close();
```

```
                conn.Open();
                dtable3 = new DataTable();

                //Calling Kinetic Equations
                command2.CommandText = "SELECT Mass_Balance FROM " + database +
                                       ".Metabolites " +
                                       "WHERE Metabolite_Name IS NOT NULL " +
                                       "AND Metabolite_Name <> '';";
                MySqlDataReader dr3 = command2.ExecuteReader
                                       (CommandBehavior.CloseConnection);
                dtable3.Load(dr3);

                file.Write("\n\ny = [   ");
                for (i = 0; i < dtable3.Rows.Count; i++)
                {
                    file.Write(dtable3.Rows[i]["Mass_Balance"].ToString() + "\n\t\t");
                }
                file.Write("];");

                conn.Close();
                file.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
```

Figure 5.9: Code implementation for creating .m file

5.5 Runge-Kutta Method

Runge-Kutta technique is implemented using MATLAB engine. The code is created based on formula of Runge-Kutta 2 steps technique. Figure 5.10 shows the code implementation of Runge-Kutta

```
function [t, y]  = ode12(func,t0,tfinal,y0,dmax)

% [t, y]  = ODE12(func,t0,tfinal,y0,dmax)
%
% A second-order accurate Runge-Kutta subroutine, with
% automatic time-step size control. The trunction error
% tolerance DMAX is a optional argument.

if nargin == 4, dmax = 1.0e-4; end

x     = y0;
dt    = (tfinal-t0)/100; % Initial guess for time step size
time  = t0;
```

```
i     = 1;
t(i)   = time;  % Initial point
y(i,:) = x';

while abs(time) < abs(tfinal)

   u1 = dt*feval(func,time,x);
   u2 = dt*feval(func,time+dt/2,x+u1/2);
   xnp1   = x + u2;
   dnp1   = u2 - u1;
   C      = norm(dnp1,inf)/dt^2;
   dt_old = dt;
   dt     = sign(dt_old)*sqrt(dmax/C);

   if abs(time + dt) > abs(tfinal), dt = tfinal - time; end   % Exact
endpoint
   u1 = dt*u1/dt_old;
   u2 = dt*feval(func,time+dt/2,x+u1/2);
   x    = x + u2;
   time = time + dt;

   i = i + 1;
   t(i)   = time;
   y(i,:) = x';

   if i > 1000, disp('Too many steps'), break, end

   end
```

Figure 5.10: Runge-Kutta Technique

CHAPTER 6

RESULT AND DISCUSSION

6.1 Introduction

This chapter is mainly discussed on the result and discussion of the dynamic simulation tools for main metabolic pathway of *Escherichia coli*. This chapter covers the functionality of the tools, result of simulations, comparison between the result of this project with hardcode and future enhancements for this project.

Thus, from this discussion, it can be concluded that this project has met the objectives which are:-

a) To identify the general dynamic model process of the main metabolic pathway of *Escherichia coli*
b) To design the dynamic simulation tools for *Escherichia coli* main metabolic pathway
c) To develop the prototype for dynamic simulation tools by implementing ordinary differential equations (ODE) of Runge-Kutta
d) To validate the functionality of the dynamic simulation tools

6.2 Functionality of the dynamic simulation tools

Software functionality is the quality of system that being suited to serve the purpose well in getting the input and displaying the output. In other word, it expressed in the form ‘system must do’ to accomplish something. Thus, the requirements of the project must meet in order to declare the system is fully functioning.

In this project, the purpose of the tools is to simplify the complexity of the available dynamic simulations of system biology in market. Since this project is focusing on micro-organism, *E. coli,* the specific requirement to develop the tools is more toward the metabolic pathway of *E. coli*. The requirements that involved in functionality are:-

- Able to create new model
- Able to open previous model
- Able to define the number of metabolites
- Able to input the dynamic equations of metabolites
- Able to input kinetic equations and parameters
- Able to re-edit all equations
- Able to configure the durations and time interval of simulations
- Able to show the graphs of simulations

Based on the requirements stated above, the testing on this project is made to verify the functionality of the tools. Figure 6.1 shows the result of the verified functionality of the dynamic simulation tools for main metabolic pathway of *Escherichia coli.*

Requirements	Qualified?	
	Yes	No
Able to create new model	X	
Able to open previous model	X	
Able to define the number of metabolites	X	
Able to input the dynamic equations of metabolites	X	
Able to input kinetic equations and parameters	X	
Able to re-edit all equations	X	
Able to configure the durations and time interval of simulations	X	
Able to show the graphs of simulations	X	

Table 6.1: Summary of functionality of the dynamic simulation tools

Meanwhile, non-functionality is required criteria that can be used to judge the operation of the system. It defines how system is supposed to be and often called qualities of a system or constraints for one system. In other words, the present of non-functionality can stabilize the system more. In this project, there are several factors that need to pay attention in order to produce more stabilize dynamic simulation tools. Figure 6.2 shows the listed non-functionality summary that been tested for this project.

Requirements	Qualified?	
	Yes	No
Prompt error message if user input database name wrongly	X	
No spacing is allowed in naming the metabolites as well as naming the database	X	
Name for each metabolites are distinct		X
User can only enter name and dynamic equations based on amount of his defined.	X	
The tools runs smoothly		X
Database is well managed	X	

Table 6.2: Summary for non-functionality of the dynamic simulation tools

6.3 Result of simulation

This section is about the simulation result based on data in Appendix C. The nine metabolites that involved in this testing is shown in figure 6.1

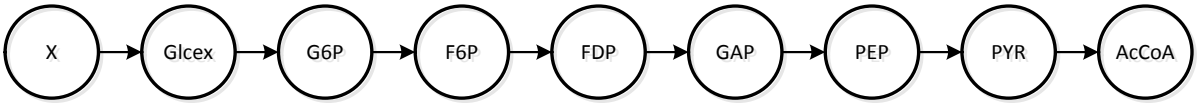


Figure 6.1: Nine metabolites involved in testing

Figure 6.2 shows the simulation result for nine metabolites involved.

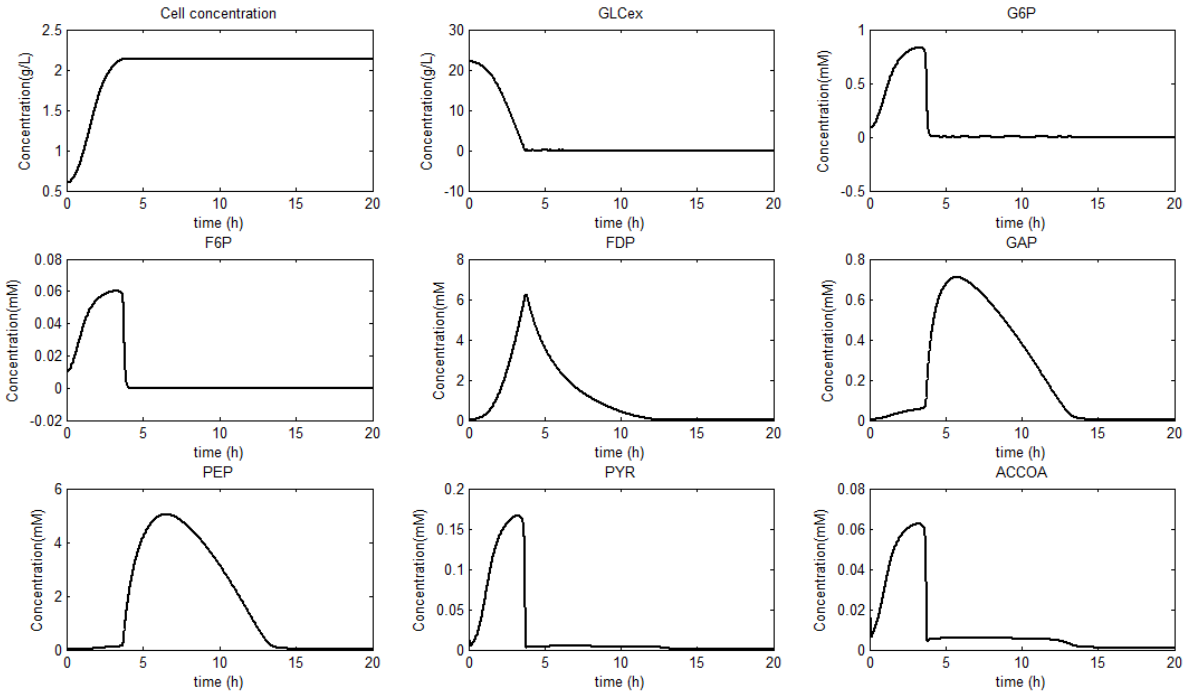


Figure 6.2: Simulation Graphs for 9 Metabolites using Dynamic Simulation Tools for Main Metabolic Pathway of *Escherichia coli*

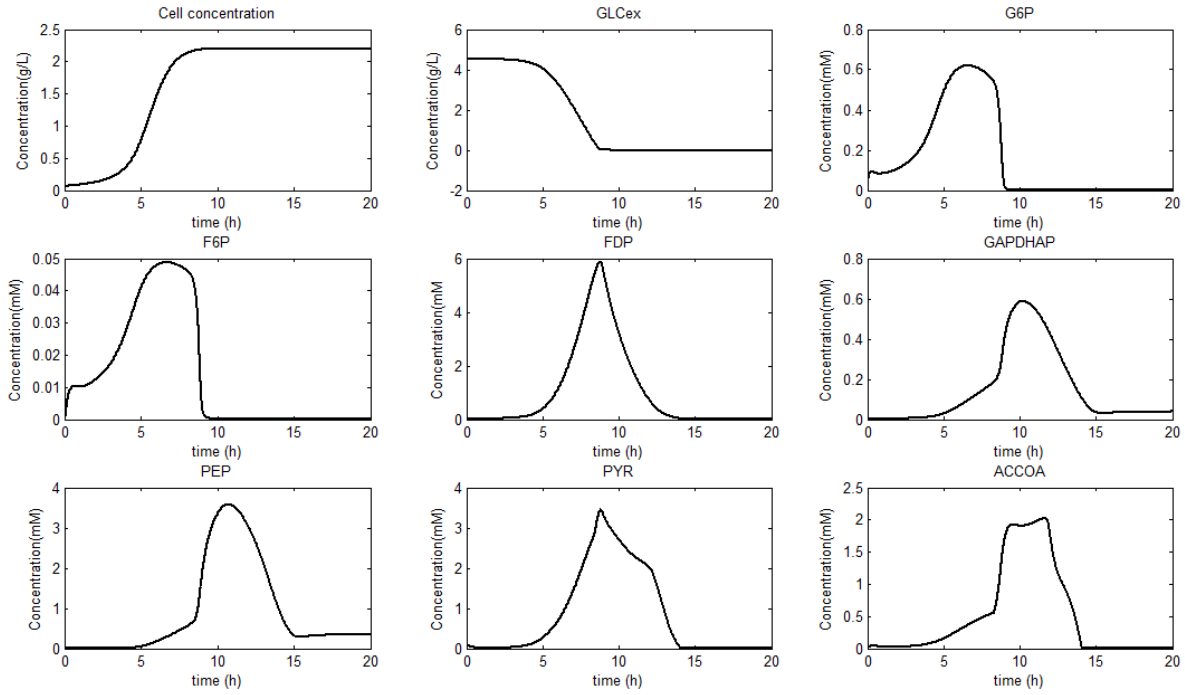


Figure 6.3: Simulation Result of Wild Type Batch Culture using hardcoded with some rule-based

Meanwhile, figure 6.3 shows the Wild Type Batch Culture. Based on figure 6.2 and figure 6.3, the simulation result using dynamic simulation tools is not the same as the one with the hard-coded (refer figure 6.3) due to some rules based and some other co-metabolites added to the hard-coded simulation result in figure 6.3. However, if user sees the results closely, it seems that the shapes of the graphs are almost the same. Thus, these results have ensured that the prototype of project is succeeded.

6.4 Future Enhancement

Prototype systems usually are not stable compared to beta version of one system. Due to that, it needs some enhancement in order to sell it to market. Same goes to Dynamic Simulation Tools for Main Metabolic Pathway of *Escherichia coli.*

Firstly, this system gives advantage to user that do not know any programming language especially target user in biochemistry field to use it easily by key in the mathematical model step by step. However, key-in lots of equations are not very suitable and time consuming. Moreover, the risk of user mistake in key in the equations is high. Thus, for future enhancement due to this problem, the project will be modify to capture and parser the equations that user has saved in text file.

Another enhancement to this project is that user can later on simulate more than nine (9) metabolites in future as well as inputting co-metabolites in their model.

CHAPTER 7

CONCLUSION

As conclusion, hopefully the dynamic simulation tools for metabolic pathway of *E. coli* have simplified enough to use it. Besides that, hopefully this project can give many beneficial to scientist in order to simulate the general models which do not involved any co-metabolites or any rule-bases. Although these tools' scopes are limited as the tools are only supported several modules, hopefully these tools can contributed something to user in finding something beneficial to mankind via metabolic pathway.

Moreover, hopefully the project can be enhancement more in processing the data equations and visualize the graph effectively. For example, these tools will be more interactive if the graph simulated can be presented in animation style. The animation of graphs help user to understand more on how the simulation of metabolic pathway takes place.

Last but not least, it is good if these tools can optimize the project model that been inserted by user / scientist. Optimization of the mathematical model allowed the simulation of the project model to be more accurate.

REFERENCES

Abdul Kadir, T.A. (2010) *Modeling and simulation of main metabolism in Escherichia coli and its several single-gene knockout mutants with experimental verification.* A Thesis Submitted in partial fulfillment of Requirements of Kyushu Institute of Technology for the Degree of Doctor of Philosophy. Iizuka, Fukuoka: Kyushu Institute of Technology.

Bailey, J. E. (1991). *Toward a science of metabolic engineering*. Science 252:1668–1675.

Brice Carnahan; Luther, H.A. and Wilkes, J.O. (1969) *Applied Numerical Method.* John Wiley & Sons, Inc.: USA.

Edana Cassol, Vikas Misra, Alexander Holman, Anupa Kamat and Dana Gabuzda. (2011). *Metabolomic Analysis of Plasma from HIV-infected Individuals on Suppressive Antiretroviral Therapy Reveals Alteration in Lipid and Amino Acid Catabolism with Immune Dysfunction*.(Online) http://www.retroconference.org/2011/PDFs/287.pdf (Retrieved date: 24th November 2012, 3.48 a.m.)

Funahashi, A.; Matsuoka, Y.; Jouraku, A.; Morohashi, M.; Kikuchi, N.; Kitano, H. (2008). *CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks.* Proceeding of the IEEE 96(8):1254–1265

Grisham, Charles M.; Reginald H. Garrett (1999). *Biochemistry*. Philadelphia: Saunders College Pub. 426–7.

Hiroyuki Kurata; Kouichi Masaki; Yoshiyuki Sumida and Rei Iwasaki (2005) *CADLIVE dynamic simulator: Direct link of biochemical networks to dynamic models* Genome Research 15(4):590–600

Jianlin Xu, Yanguang Song, Booki Min, Lisa Steinberg, and Bruce E. Logan. (2003) *Microbial Degradation of Perchlorate: Principles and Applications* Environmental Engineering Science. 20(5): 405-422

.

Koffas M.; Roberge C.; Lee K. and Stephanopoulos G. (1999) *Metabolic Engineering* Annual Review Biomedical Engineering 535-57

Korn, G.A. (2011). Interactive Dynamic-System Simulation 2nd ed. CRC Press: Florida, USA

Lessard, P. (1996). *Metabolic engineering, the concept coalesces*. Nature Biotechnology 14:1654–1655.

Madigan, Michael T.; John M. Martika and Jack Parker. (2000). *Brook Biology of Micro-organisms, 9th ed,* Pearson Prentice Hall: Upper Saddle River, NJ.

Mohammad Azhar, M.A. (2005). Fuel Consumption Cost Estimator. A Thesis Submitted in partial fulfillment of Requirements of Universiti Malaysia Pahang for Bachelor Degree. Kuantan, Gambang: Universiti Malaysia Pahang

Osman Balci. (2001). *A Methodology for Certification of Modeling and Simulation Applications.* ACM Transactions on Modeling and Computer Simulation Vol. 11 352–377.

Pedro Romero; Jonathan Wagg; Green, M.L.; Dale Kaiser; Markus Krummenacker and Karp, P.D. (2004). *Computational prediction of human metabolic pathways from complete human genome.* Open Access, Genome Biology 6:R2

Sang Yup Lee and Papoutsakis, E. T. 1999. *Metabolic engineering.* Marcel Dekker: NY.
Severance, F.L. (2001). *System Modeling and Simulation: An Introduction.* John Wiley & Sons Ltd.: England,UK.

Stephanopoulos, G. and Vallino, J.J. (1991). *Network rigidity and metabolic engineering in metabolite overproduction.* Science 252:1675–1681.

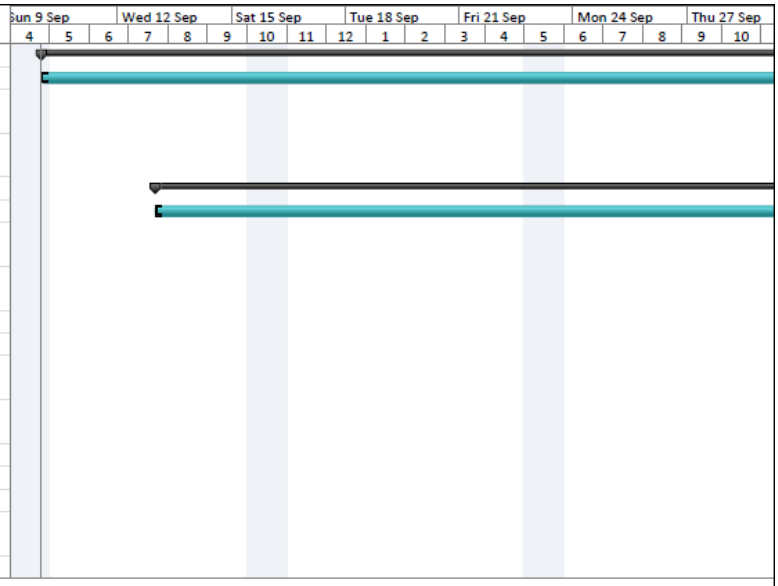
Tomita, M.; Hashimoto, K.; Takahashi, K.; Shimizu, T.S.; Matsuzaki, Y.; Miyoshi, F.; Saito, K.; Tanida, S.; Yugi, K.; Venter, J.C.; Hutchison, C.A. 3rd (1999). *E-CELL: software environments for whole-cell simulation.* Bioinformatics 15(1):72–84.

Wurtzel, E.T. and Erich Grotewold (2006). *Plant Metabolic Engineering.* (Online) http://a32.lehman.cuny.edu/webwurtzel/LabPapers/revised_wurtzel_grotewold_2006_preprint.pdf (Retrieved date: 24th November 2012, 2.37 a.m.)

APPENDIX A

GANTT CHART

ID	Task Mode	Task Name	Duration
1		**CHAPTER 01**	**20 days**
2		Introduction	20 days
3		Presentation on Chapter 01	1 day
4		Submission Chapter 01	1 day
5		**CHAPTER 02**	**57 days**
6		Literature Review	55 days
7		Presentation on Chapter 02	1 day
8		Submission on Chapter 02	1 day
9		**CHAPTER 03**	**11 days**
10		Methodology	9 days
11		Presentation on Chapter 03	1 day
12		Submission on Chapter 03	1 day
13		**CHAPTER 04**	**2 days**
14		Design	2 days
15		Present on design	1 day
16		Present proposal finalize to SV	1 day
17		**Presentation PSM1**	**1 day**



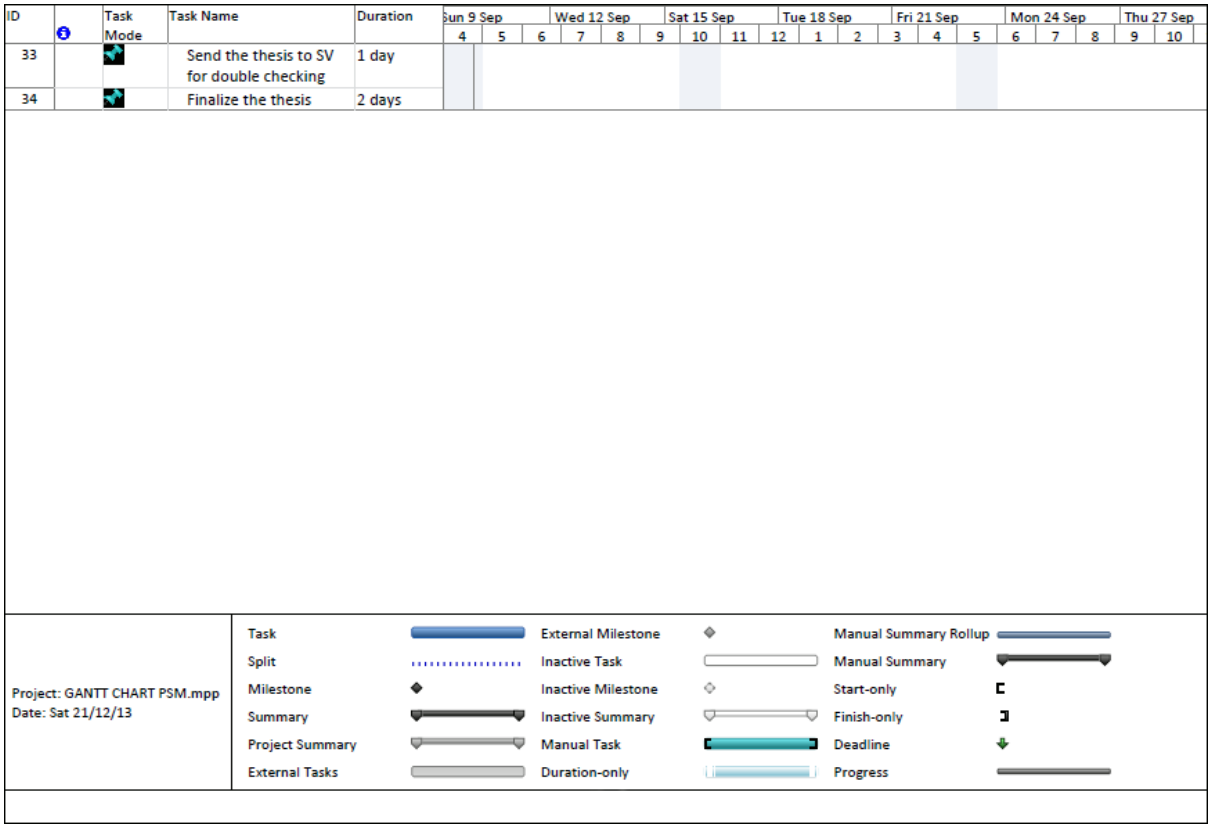
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13

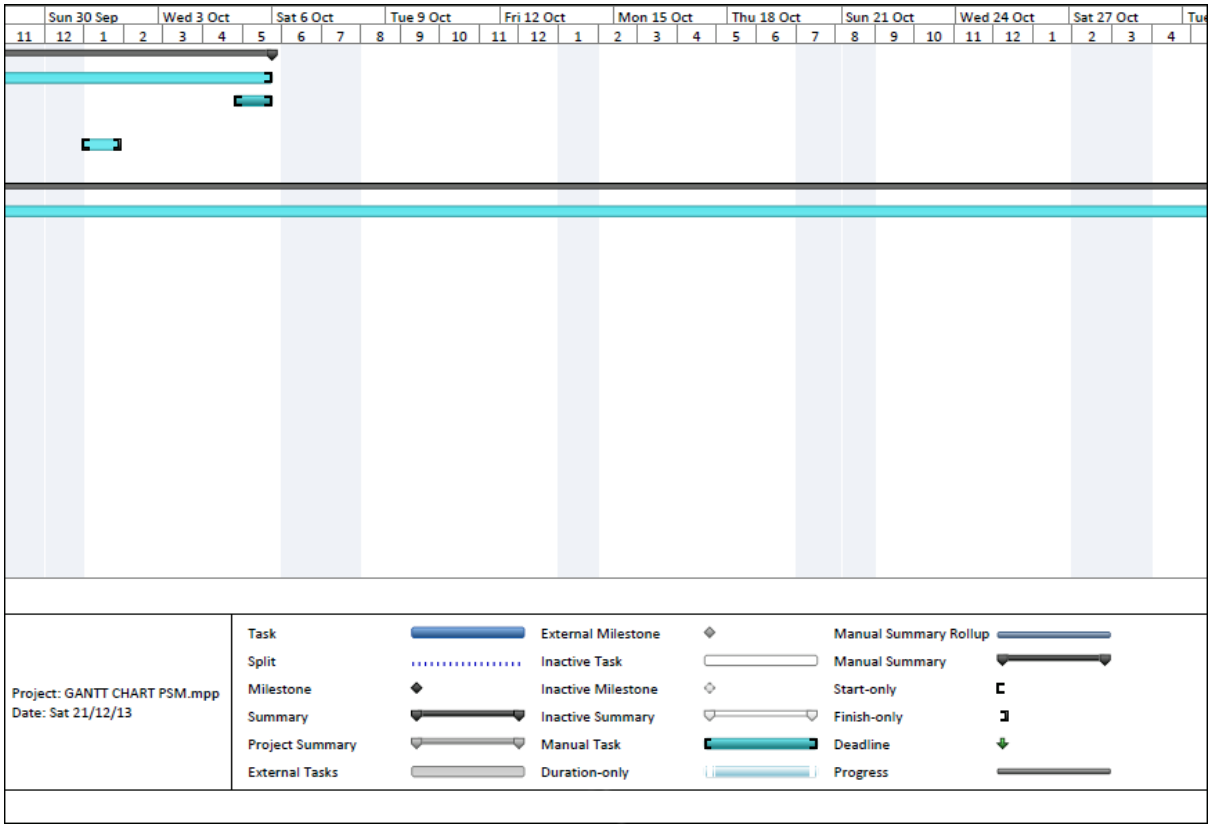
Task	External Milestone	Manual Summary Rollup	
Split	Inactive Task	Manual Summary	
Milestone	Inactive Milestone	Start-only	
Summary	Inactive Summary	Finish-only	
Project Summary	Manual Task	Deadline	
External Tasks	Duration-only	Progress	

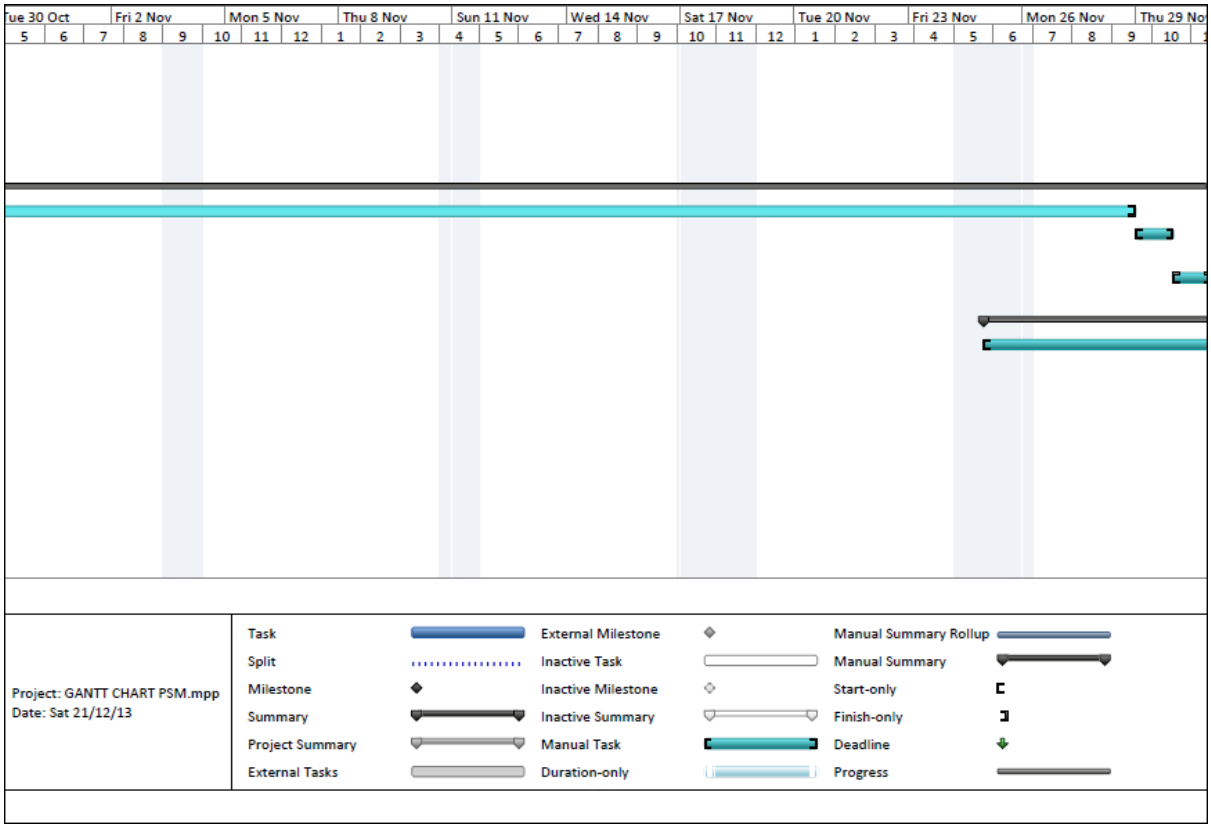
ID	Task Mode	Task Name	Duration
18		Present proposal to panel	1 day
19		**CHAPTER 04 (cont.)**	**51 days**
20		Design the system	8 days
21		Present System Design to SV	1 day
22		Implement the system design	26 days
23		Present the system prototype to SV	1 day
24		Optimize and finalize the system	12 days
25		Present to SV the final product	1 day
26		Make some repairation of the system	3 days
27		**CHAPTER 05**	**14 days**
28		Discuss the result	12 days
29		Present to SV the discussion	1 day
30		Finalize the discussion	1 day
31		**CHAPTER 06**	**5 days**
32		Make conclusion	2 days

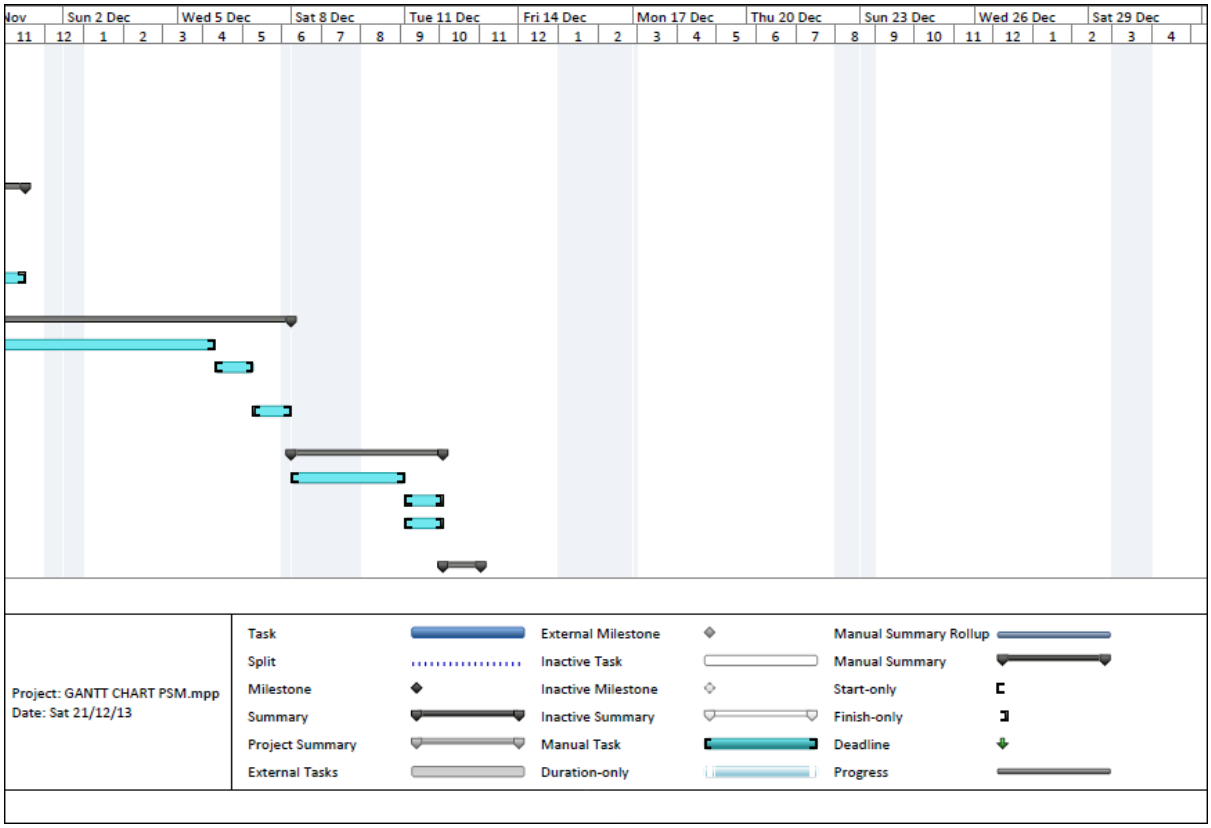
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13

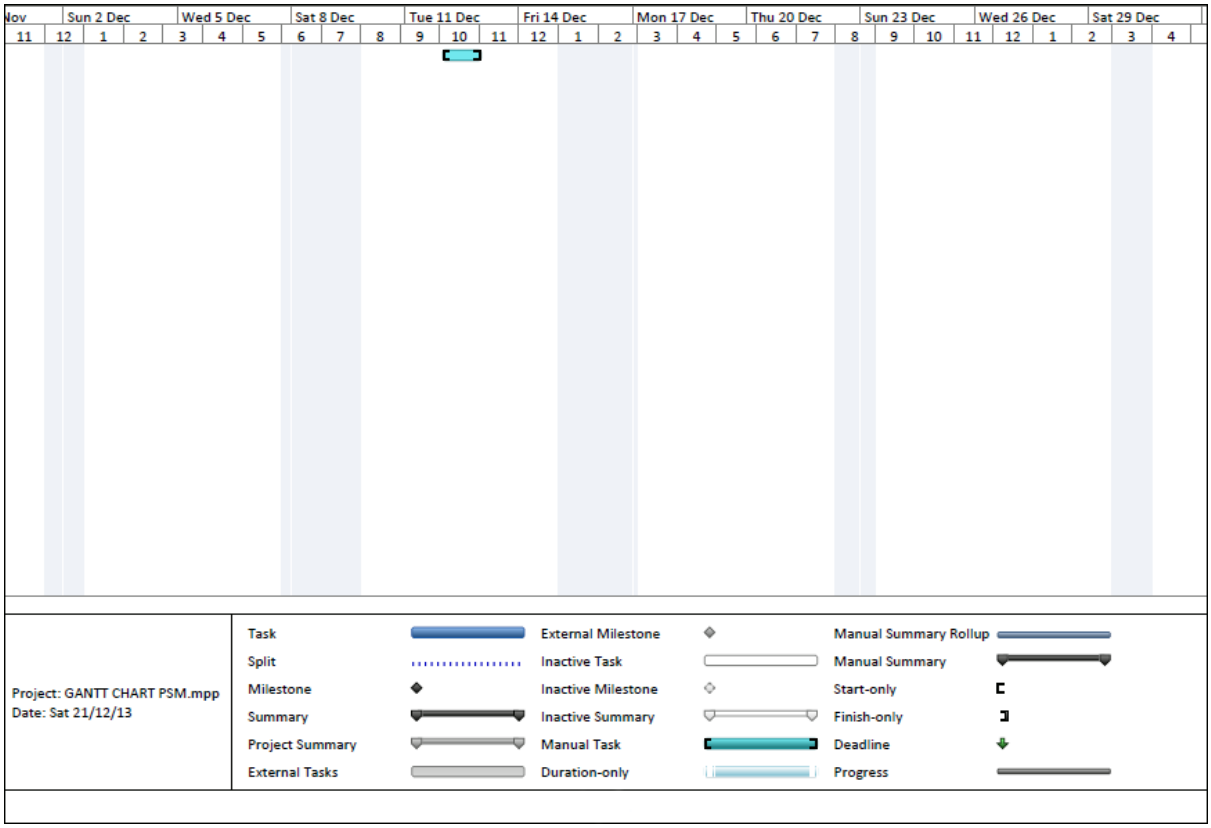
Task	External Milestone	Manual Summary Rollup	
Split	Inactive Task	Manual Summary	
Milestone	Inactive Milestone	Start-only	
Summary	Inactive Summary	Finish-only	
Project Summary	Manual Task	Deadline	
External Tasks	Duration-only	Progress	

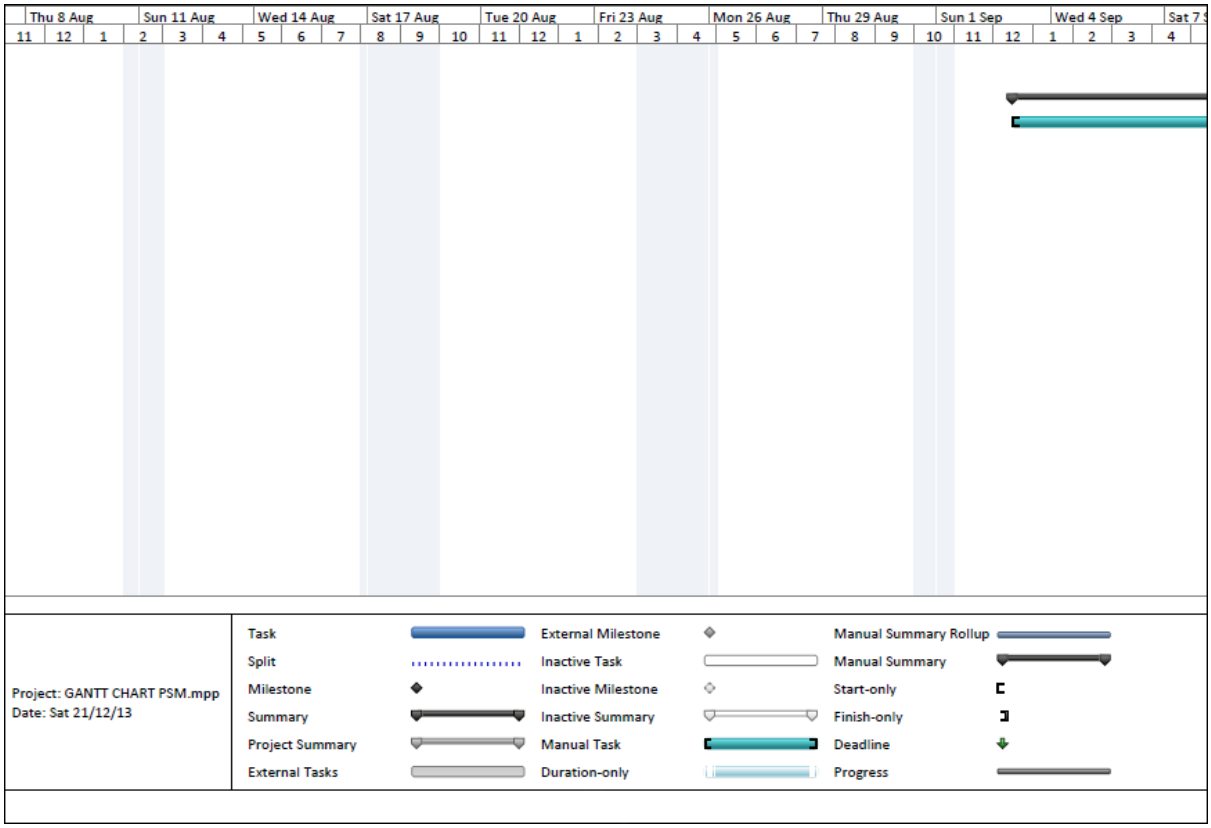

ID
Task Mode
Task Name
Duration
Sun 9 Sep
Wed 12 Sep
Sat 15 Sep
Tue 18 Sep
Fri 21 Sep
Mon 24 Sep
Thu 27 Sep
33
Send the thesis to SV for double checking
1 day
34
Finalize the thesis
2 days
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress

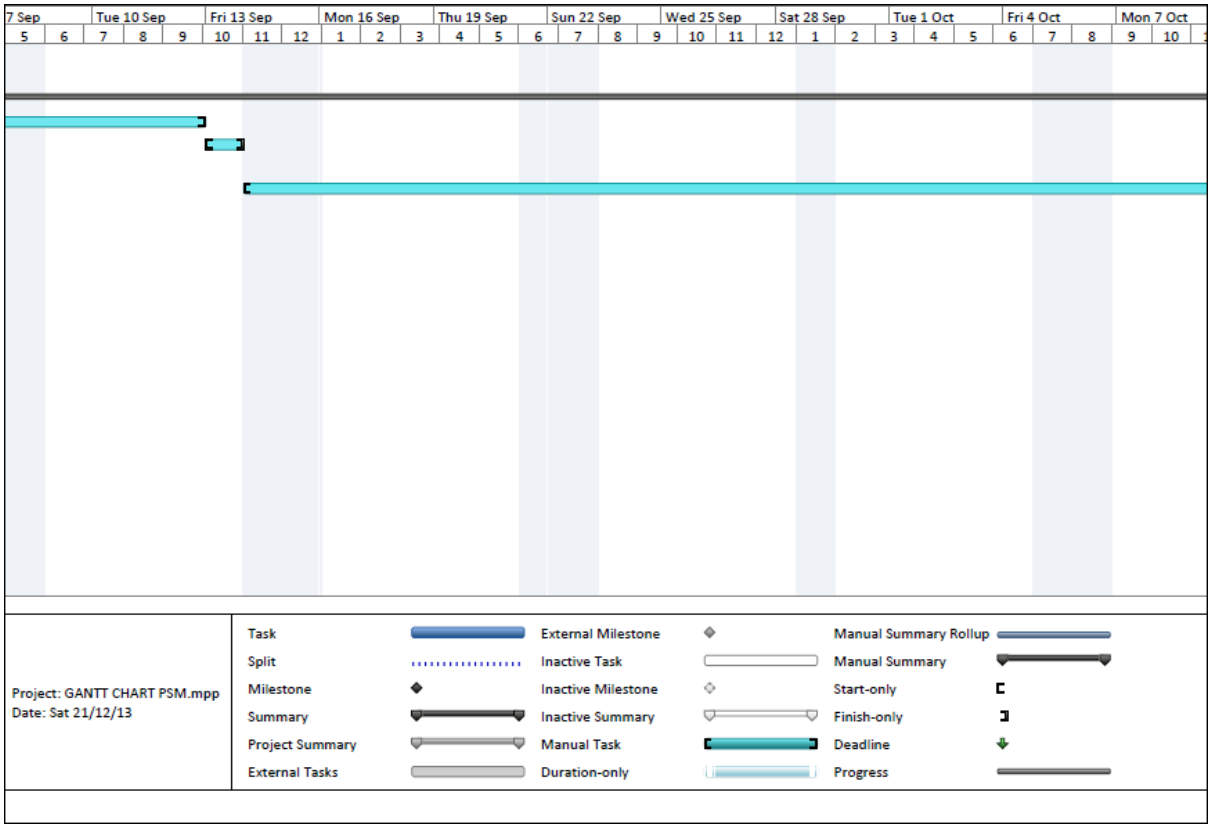

Sun 30 Sep
Wed 3 Oct
Sat 6 Oct
Tue 9 Oct
Fri 12 Oct
Mon 15 Oct
Thu 18 Oct
Sun 21 Oct
Wed 24 Oct
Sat 27 Oct
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress

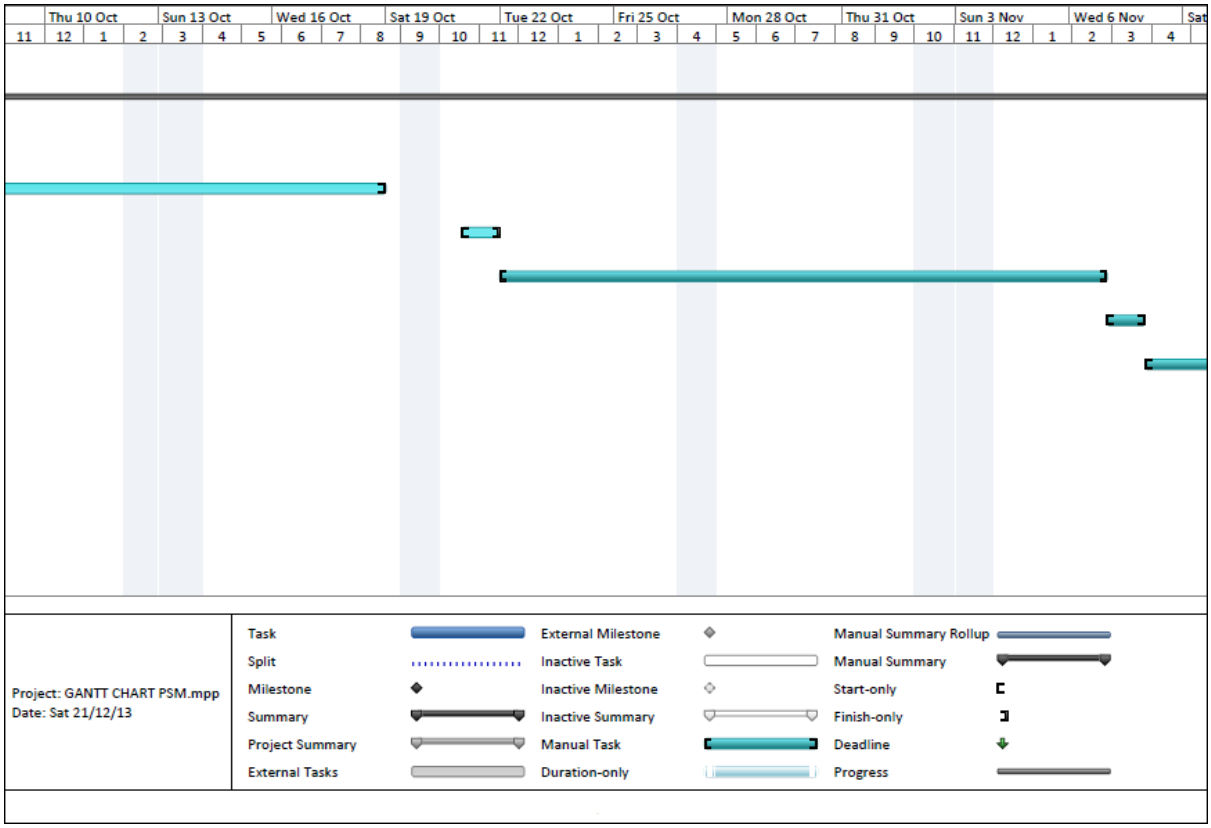

Tue 30 Oct
Fri 2 Nov
Mon 5 Nov
Thu 8 Nov
Sun 11 Nov
Wed 14 Nov
Sat 17 Nov
Tue 20 Nov
Fri 23 Nov
Mon 26 Nov
Thu 29 Nov
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress

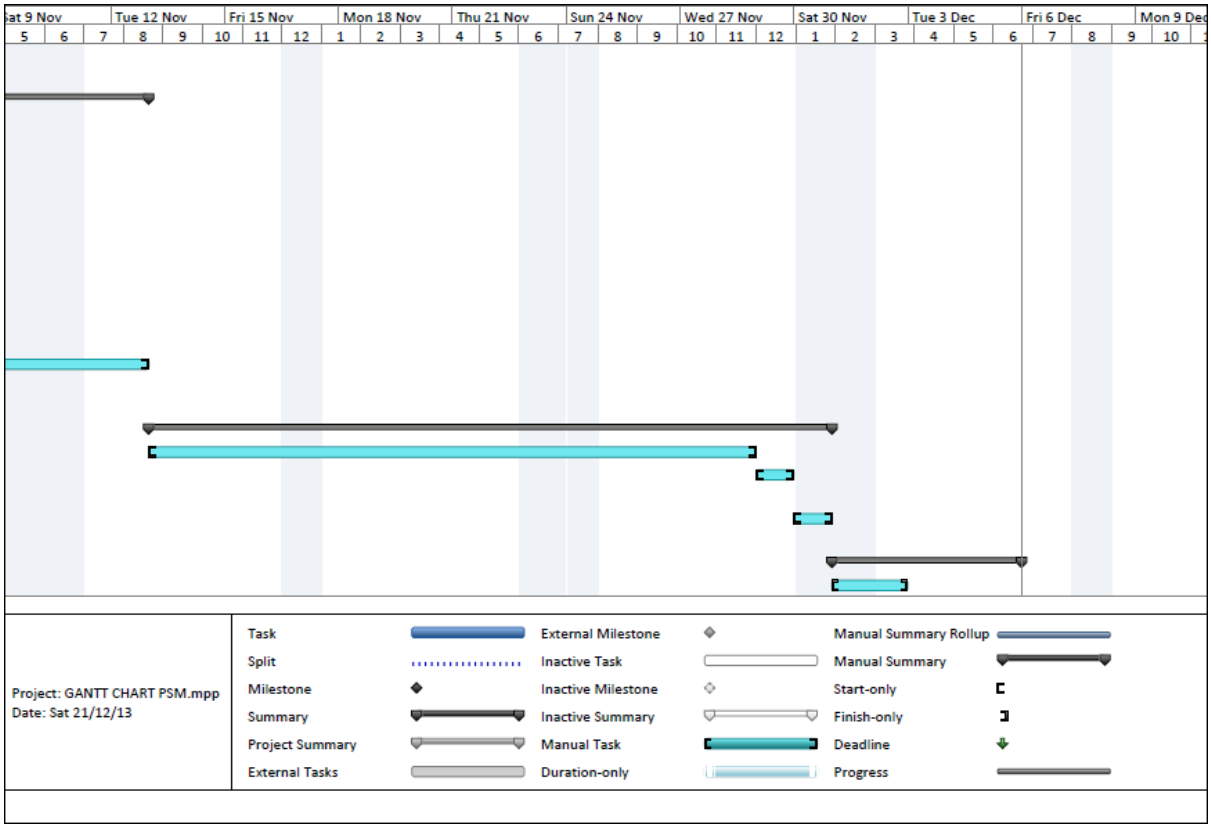

Sun 2 Dec
Wed 5 Dec
Sat 8 Dec
Tue 11 Dec
Fri 14 Dec
Mon 17 Dec
Thu 20 Dec
Sun 23 Dec
Wed 26 Dec
Sat 29 Dec
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress

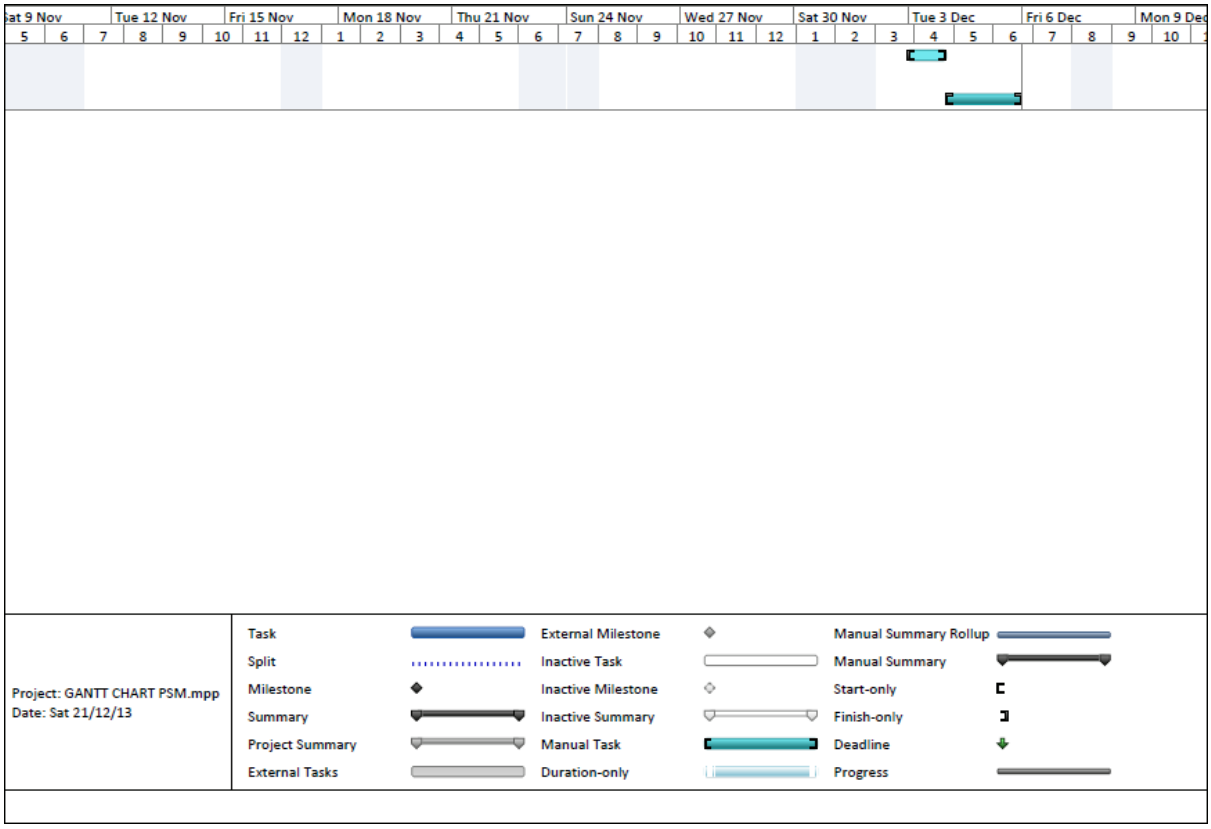

Nov
Sun 2 Dec
Wed 5 Dec
Sat 8 Dec
Tue 11 Dec
Fri 14 Dec
Mon 17 Dec
Thu 20 Dec
Sun 23 Dec
Wed 26 Dec
Sat 29 Dec
11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress


Thu 8 Aug
Sun 11 Aug
Wed 14 Aug
Sat 17 Aug
Tue 20 Aug
Fri 23 Aug
Mon 26 Aug
Thu 29 Aug
Sun 1 Sep
Wed 4 Sep
11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress


7 Sep
Tue 10 Sep
Fri 13 Sep
Mon 16 Sep
Thu 19 Sep
Sun 22 Sep
Wed 25 Sep
Sat 28 Sep
Tue 1 Oct
Fri 4 Oct
Mon 7 Oct
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress

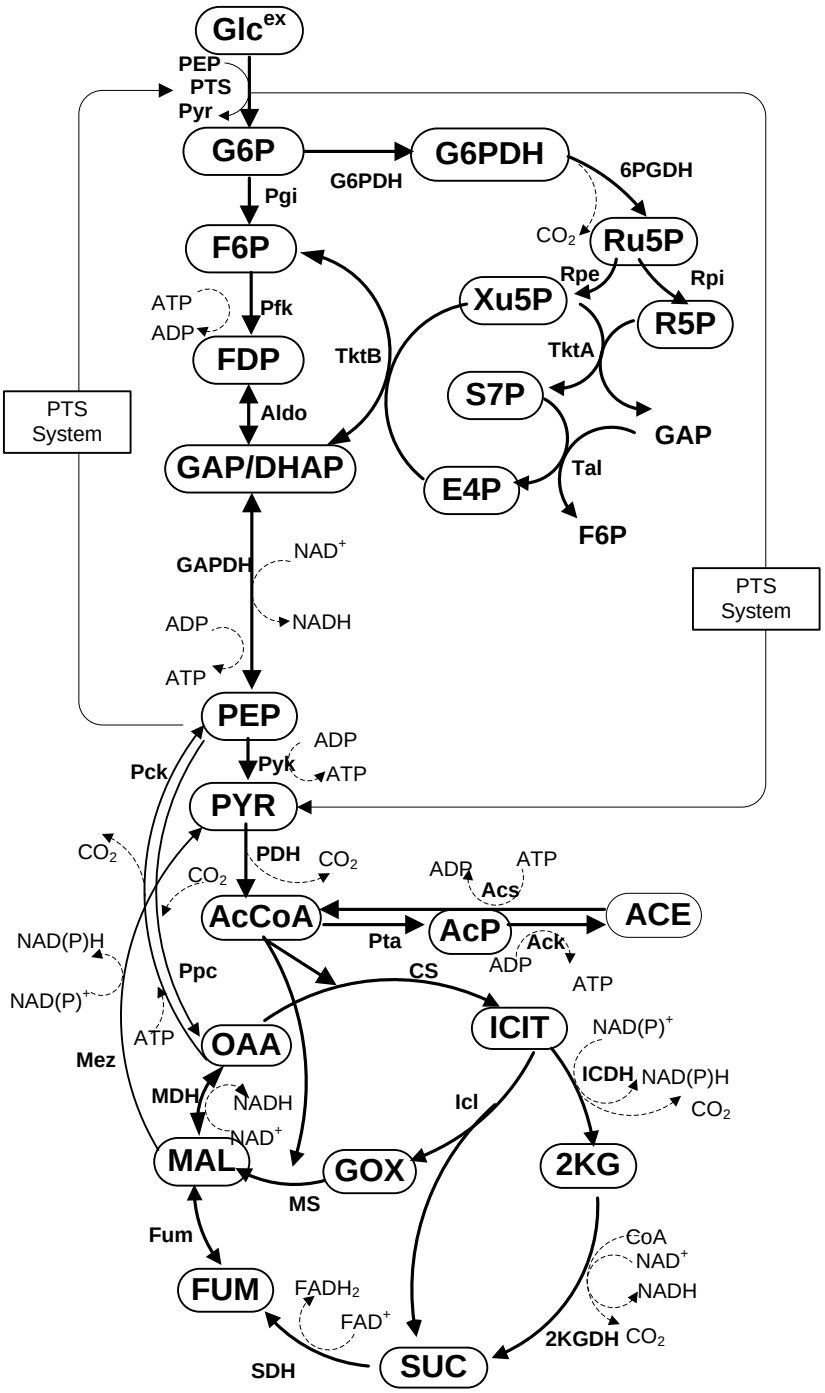

Thu 10 Oct
Sun 13 Oct
Wed 16 Oct
Sat 19 Oct
Tue 22 Oct
Fri 25 Oct
Mon 28 Oct
Thu 31 Oct
Sun 3 Nov
Wed 6 Nov
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress


at 9 Nov
Tue 12 Nov
Fri 15 Nov
Mon 18 Nov
Thu 21 Nov
Sun 24 Nov
Wed 27 Nov
Sat 30 Nov
Tue 3 Dec
Fri 6 Dec
Mon 9 Dec
5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress


at 9 Nov
Tue 12 Nov
Fri 15 Nov
Mon 18 Nov
Thu 21 Nov
Sun 24 Nov
Wed 27 Nov
Sat 30 Nov
Tue 3 Dec
Fri 6 Dec
Mon 9 Dec
5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10 11 12 1 2 3 4 5 6 7 8 9 10
Project: GANTT CHART PSM.mpp
Date: Sat 21/12/13
Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Deadline
Progress

APPENDIX B

DIAGRAM FOR MAIN METABOLIC PATHWAY OF *ESCHERICHIA COLI*


Glc^ex
PEP
PTS
Pyr
G6P
G6PDH
6PGDH
Pgi
F6P
CO2
Ru5P
Rpe
Rpi
Xu5P
R5P
ATP
ADP
Pfk
FDP
TktB
TktA
S7P
PTS System
Aldo
GAP
GAP/DHAP
Tal
E4P
F6P
GAPDH
NAD+
NADH
ADP
ATP
PTS System
PEP
ADP
Pyk
ATP
Pck
PYR
CO2
PDH
CO2
CO2
ADP
ATP
Acs
AcCoA
ACE
Pta
AcP
Ack
ADP
ATP
NAD(P)H
NAD(P)+
Ppc
CS
ATP
ICIT
NAD(P)+
Mez
OAA
ICDH
NAD(P)H
CO2
MDH
NADH
NAD+
Icl
MAL
GOX
2KG
MS
Fum
CoA
NAD+
NADH
FUM
FADH2
FAD+
2KGDH
CO2
SDH
SUC

APPENDIX C

LIST OF DATA

Mass Balance / Dynamic Equations

Mass Balance Name	Equations
Cell growth rate	$\frac{d[X]}{dt} = \mu[X]$
Extracellular Glucose Concentration	$\frac{d[GLC^{ex}]}{dt} = -v_{PTS}[X]$
Glucose-6-phosphate	$\frac{d[G6P]}{dt} = v_{PTS} - v_{PGI} - v_{G6PDH} - \mu[G6P]$
Fructose 6-phosphate	$\frac{d[F6P]}{dt} = v_{PGI} - v_{PFK} + v_{TKTB} + v_{TAL} - \mu[F6P]$
Fructose 1,6-bisphosphate	$\frac{d[FDP]}{dt} = v_{PFK} - v_{ALDO} - \mu[FDP]$
Glyceraldehyde 3-phosphate	$\frac{d[GAP]}{dt} = 2v_{ALDO} - v_{GAPDH} + v_{TKTA} + v_{TKTB} - v_{TAL} - \mu[GAP]$
Phosphoenolpyruvate	$\frac{d[PEP]}{dt} = v_{GAPDH} + v_{PCK} - v_{PTS} - v_{PYK} - v_{PPC} - \mu[PEP]$
Pyruvate	$\frac{d[PYR]}{dt} = v_{PYK} + v_{PTS} + v_{MEZ} - v_{PDH} - \mu[PYR]$
Acetyl-CoA	$\frac{d[AcCoA]}{dt} = v_{PDH} + v_{ACS} - v_{CS} - v_{PTA} - \mu[AcCoA]$
Isocitrate	$\frac{d[ICIT]}{dt} = v_{CS} - v_{ICDH} - v_{ICL} - \mu[ICIT]$
2-Keto-D-gluconate	$\frac{d[2KG]}{dt} = v_{ICDH} - v_{2KGDH} - \mu[2KG]$
Succinate	$\frac{d[SUC]}{dt} = v_{2KGDH} + v_{ICL} - v_{SDH} - \mu[SUC]$
Fumarate	$\frac{d[FUM]}{dt} = v_{SDH} - v_{FUM} - \mu[FUM]$
Malate	$\frac{d[MAL]}{dt} = v_{FUM} + v_{MS} - v_{MDH} - v_{MEZ} - \mu[MAL]$
Oxaloacetate	$\frac{d[OAA]}{dt} = v_{MDH} + v_{PPC} - v_{CS} - v_{PCK} - \mu[OAA]$
Glyphosate Oxidoreductase	$\frac{d[GOX]}{dt} = v_{ICL} - v_{MS} - \mu[GOX]$
Acetyl phosphate	$\frac{d[ACP]}{dt} = v_{PTA} - v_{ACK} - \mu[ACP]$
Extracellular Acetate	$\frac{d[ACE^{ex}]}{dt} = (v_{ACK} - v_{ACS})[X]$
6-Phosphogluconolactone	$\frac{d[6PG]}{dt} = v_{G6PDH} - v_{6PGDH} - \mu[6PG]$
Ribose 5-phosphate	$\frac{d[Ru5P]}{dt} = v_{6PGDH} - v_{RPE} - v_{RPI} - \mu[Ru5P]$

Ribulose 5-phosphate	$\frac{d[R5P]}{dt} = v_{RPI} - v_{TKTA} - \mu[R5P]$
Xylulose 5-phosphate	$\frac{d[X5P]}{dt} = v_{RPE} - v_{TKTA} - \mu[X5P]$
Sedoheptulose 7-phosphate	$\frac{d[S7P]}{dt} = v_{TKTA} - v_{TAL} - \mu[S7P]$
Erythrose 4-phosphate	$\frac{d[E4P]}{dt} = v_{TAL} - v_{TKTB} - \mu[E4P]$

Enzyme Kinetic Equations

Enzyme Kinetic Equations Name	Equations
Cell growth rate	$$\mu(S)=\frac{\mu_m S}{K_s+S}$$ $$\mu(S,X)=\mu_m\left(1-\frac{X}{X_m}\right)\frac{S}{K_s+S}$$ $$\mu(S,X,v_{ATP})=\mu_m\left(1-\frac{X}{X_m}\right)^n\frac{S}{K_s+S}k_{ATP}v_{ATP}(\bullet)$$ $$\mu=\begin{cases}\mu_m\left(1-\frac{[X]}{X_m}\right)\left(\frac{[Glc^{ex}]}{K_s+[Glc^{ex}]}\right)k_{ATP}v_{ATP}(.),([Glc^{ex}]>0)\\ \frac{\mu_{mA}[Ace^{ex}]}{K_{sA}+[Ace^{ex}]}k_{ATP}v_{ATP}(.),([Glc^{ex}]\le 1 \ and\ [Ace^{ex}]>0)\end{cases}$$
Phosphotransferase system	$$v_{PTS}=\frac{v_{PTS}^{\max}[Glc^{ex}]\frac{[PEP]}{[PYR]}}{\left(K_{a1}+K_{a2}\frac{[PEP]}{[PYR]}+K_{a3}[Glc^{ex}]+[Glc^{ex}]\frac{[PEP]}{[PYR]}\right)\left(1+\frac{[G6P]^{n_{G6P}}}{K_{G6P}}\right)}$$

Phophoglucose isomerase / Glucosephosphate isomerase	$$v_{Pgi} = \frac{v_{Pgi}^{\max}\left([G6P] - \frac{[F6P]}{K_{eq}}\right)}{K_{G6P}\left(1 + \frac{F6P}{K_{F6P}\left(1 + \frac{[F6P]}{K_{6pginh}^{F6P}}\right)} + \frac{[6PG]}{K_{6pginh}^{G6P}}\right) + G6P}$$
Glucose-6-phosphate dehydrogenase	$$v_{G6PDH} = \frac{v_{G6PDH}^{\max}[G6P]}{([G6P] + K_{G6P})\left(1 + \frac{[NADPH]}{K_{NADPH}^{G6P}}\right)\left(\frac{K_{NADP}}{[NADP]}\left(1 + \frac{[NADPH}{K_{NADPH}^{NADP}}\right) + 1\right)}$$
Phosphofructokinase-1	$$v_{Pfk} = \frac{v_{Pfk}^{\max} K_{ATP}[F6P]}{K_{(ATP,ADP)}\left([F6P] + K_s^{F6P}\frac{K_{b(ADP,AMP)} + \frac{[PEP]}{K_{PEP}}}{K_{a(ADP,AMP)}}\right) \times \left(1 + \frac{L_{Pfk}}{\left(1 + [F6P]\frac{K_{a(ADP,AMP)}}{K_s^{F6P}\left(K_{b(ADP,AMP)} + \frac{[PEP]}{K_{PEP}}\right)}\right)^{n_{Pfk}}}\right)}$$
TransketolaseII	$$v_{TktB} = v_{TktB}^{\max}\left([Xu5P][E4P] - \frac{[F6P][GAP]}{K_{eq}^{TktB}}\right)$$
Transaldolase	$$v_{Tal} = v_{Tal}^{\max}\left([GAP][S7P] - \frac{[E4P][F6P]}{K_{eq}^{TktB}}\right)$$
Aldolase	$$v_{Aldo} = \frac{v_{Aldo}^{\max}\left([FDP] - \frac{[DHAP][GAP]}{K_{eq}}\right)}{\left(K_{FDP} + [FDP] + \frac{K_{GAP}[DHAP]}{[K_{eq}V_{blf}]} + \frac{K_{DHAP}[GAP]}{[K_{eq}V_{blf}]} + \frac{[FDP][GAP]}{K_{inh}^{PEP}} + \frac{[DHAP][GAP]}{K_{eq}V_{blf}}\right)}$$
Glyceraldehyde 3-phosphate dehydrogenase	$$v_{GAPDH} = \frac{v_{GAPDH}^{\max}\left([GAP] - \frac{[PEP][NADH]}{K_{eq}[NAD]}\right)}{\left(K_{GAP}\left(1 + \frac{[PEP]}{K_{PGP}}\right) + [GAP]\right)\left(\frac{K_{NAD}}{NAD}\left(1 + \frac{[NADH]}{K_{NADH}}\right) + 1\right)}$$
TransketolaseI	$$v_{TktA} = v_{TktA}^{\max}\left([R5P][Xu5P] - \frac{[S7P][GAP]}{K_{eq}^{TktA}}\right)$$

Phosphoenolpyruvate carboxykinase	$$v_{Pck} = v_{Pck}^{\max}\left(\frac{[OAA]\frac{[ATP]}{[ADP]}}{K_m^{OAA}\frac{[ATP]}{[ADP]} + [OAA]\frac{[ATP]}{[ADP]} + \frac{K_i^{ATP}K_m^{OAA}}{K_i^{ADP}} + \frac{K_i^{ATP}K_m^{OAA}}{K_m^{PEP}K_i^{ADP}}[PEP] + \frac{K_i^{ATP}K_m^{OAA}}{K_i^{PEP}K_I^{ATP}}\frac{[ATP][PEP]}{[ADP]} + \frac{K_i^{ATP}K_m^{OAA}}{K_i^{ADP}K_I^{OAA}}[OAA]}\right)$$
Pyruvate kinase	$$v_{Pyk} = \frac{v_{Pyk}^{\max}[PEP]\left(\frac{[PEP]}{K_{PEP}}+1\right)^{n_{Pyk}-1}[ADP]}{K_{PEP}\left(L_{Pyk}\left(\frac{1+\frac{[ATP]}{K_{ATP}}}{\frac{[FDP]}{K_{FDP}}+\frac{[AMP]}{K_{AMP}}+1}\right)^{n_{Pyk}} + \left(\frac{[PEP]}{K_{PEP}}+1\right)^{n_{Pyk}}\right)([ADP]+K_{ADP})}$$
PEP carboxylase	$$v_{Ppc} = \frac{K_1 + K_2[AcCoA] + K_3[FDP] + K_4[AcCoA][FDP]}{1 + K_5[AcCoA] + K_6[FDP]}\left(\frac{[PEP]}{K_m + [PEP]}\right)$$
Malic enzyme	$$v_{Mez} = \frac{v_{Mez}^{\max}[MAL][NADP]}{(K_{MAL}+[MAL])(K_{eq}+[NADP])}$$
Malate dehydrogenase	$$v_{PDH} = \frac{\frac{v_{PDH}^{\max}}{[NAD]}\left(\frac{1}{1+K_i\frac{[NADH]}{[NAD]}}\right)\left(\frac{[PYR]}{K_m^{PYR}}\right)\left(\frac{1}{K_m^{NAD}}\right)\left(\frac{[COA]}{K_m^{COA}}\right)}{\left(1+\frac{[PYR]}{K_m^{PYR}}\right)\left(\frac{1}{NAD}+\frac{1}{K_m^{NAD}}+\frac{[NADH]}{K_m^{NADH}[NAD]}\right)\left(1+\frac{[COA]}{K_m^{COA}}+\frac{[AcCoA]}{K_m^{AcCoA}}\right)}$$
Acetyl coenzyme Asynthetase	$$v_{Acs} = \frac{v_{Acs}^{\max}[ACE][NADP]}{(K_m+[ACE])(K_{eq}+[NADP])}$$
Citrate synthase	$$v_{CS} = \frac{v_{CS}^{\max}[AcCoA][OAA]}{\left(K_d^{AcCoA}K_m^{OAA}+K_m^{AcCoA}[OAA]\right)+\left([AcCoA]K_m^{OAA}\left(1+\frac{[NADH]}{K_{i1}^{NADH}}\right)\right)+\left([AcCoA][OAA]\left(1+\frac{[NADH]}{K_{i2}^{NADH}}\right)\right)}$$
Phosphotransacetylase	$$v_{Pta} = \frac{v_{Pta}^{\max}\left(\frac{1}{K_i^{AcCoA}K_m^{P}}\right)\left([AcCoA][P]-\frac{[AcP][CoA]}{K_{eq}}\right)}{\left(1+\frac{[AcCoA]}{K_i^{AcCoA}}+\frac{[P]}{K_i^{P}}+\frac{ACP}{K_i^{ACP}}+\frac{[CoA]}{K_i^{CoA}}+\left(\frac{[AcCoA][P]}{K_i^{AcCoA}K_m^{P}}\right)+\left(\frac{[AcP][CoA]}{K_m^{ACP}K_i^{CoA}}\right)\right)}$$
Isocitrate dehydrogenase	$$v_{ICDH} = \frac{[ICDH]\frac{K_f}{K_m^{iCiT}}K_d^{NADP}\left([ICIT]-\frac{[NADPH][2KG]}{K_{eq}^{ICDH}[NADP]}\right)}{\left(\frac{1}{[NADP]}+\frac{[ICIT]K_m^{NADP}}{K_m^{iCiT}K_d^{NADP}[NADP]}+\frac{1}{K_d^{NADP}}+\frac{[ICIT]}{K_m^{iCiT}K_d^{NADP}}+\frac{[ICIT]}{K_d^{iCiT}[NADP]}\frac{[NADPH]K_m^{NADP}}{K_m^{iCiT}K_d^{NADP}K_{einh}^{NADPH}}+\frac{[NADPH]K_{eknh}^{2KG}}{K_m^{2KG}K_{enhe}^{NADPH}[NADP]}+\frac{[2KG]K_m^{NADPH}}{K_m^{2KG}K_{enhe}^{NADPH}[NADP]}+\frac{[2KG]}{K_m^{2KG}}\frac{[NADPH]}{K_{enhe}^{NADPH}[NADP]}+\frac{[2KG]K_m^{NADPH}}{K_m^{2KG}K_{enhe}^{NADPH}}\frac{[NADPH]}{K_{ekn}^{NADP}[NADP]}\right)}$$

Isocitrate lyase	$$v_{Icl} = \frac{v_{Icl\_f}^{\max}\frac{[ICIT]}{K_m^{iCiT}}}{\left(1+\frac{[ICIT]}{K_m^{iCiT}}+\frac{[SUC]}{K_m^{SUC}}+\frac{[GOX]}{K_m^{GOX}}+\frac{[ICIT]}{K_m^{iCiT}}\frac{[SUC]}{K_m^{SUC}}+\frac{[SUC]}{K_m^{SUC}}\frac{[GOX]}{K_m^{GOX}}+\frac{I}{K_I}\right)}$$
2-Keto-D-gluconate Dehydrogenase	$$v_{\alpha KGDH} = \frac{v_{2KGDH}^{\max}[\alpha KG][CoA]}{\left(\begin{array}{l}\frac{K_m^{NAD}[\alpha KG][CoA]}{[NAD]}+K_m^{CoA}[\alpha KG]+K_m^{2KG}[CoA]+[\alpha KG][CoA]+\\ \frac{K_m^{2KG}K_z[SUC][NADH]}{K_I^{SUC}[NAD]}+\frac{K_m^{NAD}[\alpha KG][CoA][NADH]}{K_I^{NADH}[NAD]}+\\ \frac{K_m^{CoA}[\alpha KG][SUC]}{K_I^{SUC}}+\frac{K_m^{2KG}K_z[\alpha KG][SUC][NADH]}{K_I^{2KG}K_I^{SUC}[NAD]}\end{array}\right)}$$
Succinate dehydrogenase	$$v_{SDH} = \frac{v_{SDH1}v_{SDH2}\left([SUC]-\frac{[FUM]}{K_{eq}}\right)}{K_m^{SUC}v_{SDH2}+v_{SDH2}[SUC]+\frac{V_{SDH1}[FUM]}{K_{eq}}}$$
Fumurate	$$v_{Fum} = \frac{v_{Fum1}v_{Fum2}\left([FUM]-\frac{[MAL]}{K_{Fumeq}}\right)}{K_m^{Fum}v_{Fum1}+v_{Fum2}[FUM]+\frac{V_{Fum1}[MAL]}{K_{eq}}}$$
Malate synthase	$$v_{MS} = \frac{v_{MS\_f}^{\max}\frac{[GOX]}{K_m^{GOX}}\frac{[AcCoA]}{K_m^{AcCoA}}-v_{MS\_r}^{\max}\frac{[MAL]}{K_m^{MAL}}}{\left(1+\frac{[GOX]}{K_m^{GOX}}+\frac{[MAL]}{K_m^{MAL}}+\left(1+\frac{[AcCoA]}{K_m^{AcCoA}}\right)\right)}$$
Malate dehydrogenase	$$v_{MDH} = \frac{v_{MDH1}v_{MDH2}\left([MAL]-\frac{[OAA]}{K_{eq}}\right)}{\left(\begin{array}{l}\frac{K_I^{NAD}K_m^{MAL}v_{MDH2}}{[NAD]}+K_m^{MAL}v_{MDH2}+\frac{K_m^{NAD}V_{MDH2}[MAL]}{[NAD]}+v_{MDH2}[MAL]+\\ \frac{K_m^{OAA}v_{MDH1}[NADH]}{K_{eq}[NAD]}+\frac{K_m^{NADH}v_{MDH1}[OAA]}{K_{eq}[NAD]}+\frac{v_{MDH1}[NADH][OAA]}{K_{eq}[NAD]}+\\ \frac{v_{MDH1}K_m^{OAA}[NADH]}{K_{eq}K_I^{NAD}}+\frac{v_{MDH2}K_m^{NAD}[MAL][OAA]}{K_I^{OAA}[NAD]}+\frac{v_{MDH2}[MAL][NADH]}{K_I^{NADH}}+\\ \frac{v_{MDH1}[MAL][NADH][OAA]}{K_{eq}K_I^{MAL}[NAD]}+\frac{v_{MDH2}[MAL][OAA]}{K_{II}^{OAA}}+\frac{v_{MDH1}[NADH][OAA]}{K_{II}^{NAD}K_{eq}}+\\ \frac{K_I^{NAD}v_{MDH2}[MAL][NADH][OAA]}{K_{II}^{NAD}K_m^{OAA}K_I^{NADH}}\end{array}\right)}$$
Acetate kinase	$$v_{Ack} = \frac{v_{Ack}^{\max}\left(\frac{1}{K_m^{ADP}K_m^{ACP}}\right)\left([AcP][ADP]-\frac{[ACE][ATP]}{K_{eq}}\right)}{\left(1+\frac{[AcP]}{K_m^{ACP}}+\frac{ACE}{K_m^{ACE}}\right)\left(1+\frac{[ADP]}{K_m^{ADP}}+\frac{[ATP]}{K_m^{ATP}}\right)}$$

Acetyl coenzyme Asynthetase	$v_{Acs} = \frac{v_{Acs}^{\max}[ACE][NADP]}{(K_m + [ACE])(K_{eq} + [NADP])}$
6-Phosphogluconolactone dehydrogenase	$v_{6GPDH} = \frac{v_{G6PDH}^{\max}[6PG]}{([6PG] + K_{6PG})\left(1 + \frac{K_{NADP}}{NADP}\left(1 + \frac{[NADPH]}{K_{NADPH}^{inh}}\right)\left(1 + \frac{[ATP]}{K_{ATP}^{inh}}\right)\right)}$
Ribulose phosphate 3-epimerase	$v_{Rpe} = v_{Rpe}^{\max}\left([Ru5P] - \frac{[Xu5P]}{K_{eq}^{Rpe}}\right)$
Ribulose 5-phosphate 3-isomerase	$v_{Rpi} = v_{Rpi}^{\max}\left([Ru5P] - \frac{[R5P]}{K_{eq}^{Rpi}}\right)$

Enzyme Kinetic Parameter / Modal Parameter

Enzyme	Kinetic parameter values	Original source
Cell growth	$\mu_m = 0.6$, $K_s = 0.1$, $\mu_{mA} = 0.9$, $K_{sA} = 0.01$, $X_m = 2.3$	estimated
Glk	$v_{Glk}^{\max} = 4.503 mmol/gDCW.h$, $K_m^{GLC} = 0.12mM$, $K_m^{AATP} = 0.5mM$	Ishii et al., 2006
PTS	$v_{PTS}^{\max} = 25.739 mmol/gDCW.h$, $K_{a1} = 1.0mM$, $K_{a2} = 0.01mM$, $K_{a3} = 1.0$, $n_{G6P} = 4$, $K_{G6p} = 0.5mM$	Chassagnole et al., 2002; Liao et al., 1996
Pgi	$v_{PGI}^{\max} = 26.3711 mmol/gDCW.h$, $K_{G6P} = 2.46mM$, $K_{F6P} = 0.2mM$, $K_{eq} = 0.43mM$, $K_{6pginh}^{G6P} = 0.2mM$, $K_{6pginh}^{F6P} = 0.2mM$	Chassagnole et al., 2002
Pfk	$v_{Pfk}^{\max} = 24.613 mmol/gDCW.h$, $K_{(ATP)} = 4.27mM$, $K_{(ATP,ADP)} = 4.6944mM$, $K_{a(ADP,AMP)} = 1.1118mM$, $K_{b(ADP,AMP)} = 98.88mM$, $n_{Pfk} = 4$, $L_{Pfk} = 1000$, $K_s^{F6P} = 0.14mM$, $K_{PEP} = 3.26mM$	Chassagnole et al., 2002
Aldo	$v_{Aldo}^{\max} = 2.8337 mmol/gDCW.h$, $K_{FDP} = 0.133mM$, $K_{GAP} = 0.088mM$, $K_{DHAP} = 0.088mM$, $K_{inh}^{GAP} = 0.6mM$, $K_{eq} = 0.14mM$, $V_{blf} = 2$	Chassagnole et al., 2002
GAPDH	$v_{GAPDH}^{\max} = 121.29 mmol/gDCW.h$, $K_{GAP} = 0.15mM$, $K_{PGP} = 0.1mM$, $K_{NAD} = 0.45mM$, $K_{NADH} = 0.02mM$, $K_{eq} = 0.63$	Chassagnole et al., 2002
Pyk	$v_{Pyk}^{\max} = 1.0849 mmol/gDCW.h$, $K_{PEP} = 0.31mM$, $K_{FDP} = 0.19mM$, $K_{AMP} = 0.2mM$, $n_{Pyk} = 4$ $K_{ATP} = 22.5mM$, $K_{ADP} = 0.26mM$, $L_{Pyk} = 1000$,	Chassagnole et al., 2002

PDH	$v_{PDH}^{\max} = 27171 mmol/gDCW.h$, $K_m^{PYR} = 1mM$, $K_m^{NAD} = 0.4mM$, $K_m^{AcCoA} = 0.008mM$, $K_m^{NADH} = 0.1mM$, $K_m^{COA} = 0.014mM$, $K_i^{PDH} = 46.4mM$	Hoefnagel et al., 2002
CS	$v_{CS}^{\max} = 17.36 mmol/gDCW.h$, $K_{d1}^{H} = 0.00001mM$, $K_{d2}^{H} = 0.0002mM$, $K_m^{AcCoA} = 0.18mM$, $K_m^{OAA} = 0.04mM$, $K_d^{AcCoA} = 0.1mM$, $K_i^{ATP} = 0.58mM$, $K_{i1}^{2KG} = 0.015mM$, $K_{i2}^{2KG} = 0.256mM$, $K_{i1}^{NADH} 0.00033mM$, $K_{i2}^{NADH} = 0.0084mM$, $K_{cat0} = 1$	Mogilevskaya et al., 2006
ICDH	$v_{ICDH}^{\max} = 24.42 mmol/gDCW.h$, $K_{eq} = 1000mM$, $k_f = 4830 - 1/\min$, $K_m^{2KG} = 0.038mM$, $K_m^{iCiT} = 0.0059mM$, $K_d^{NADP} = 0.0013mM$, $K_m^{NADP} = 0.0227mM$, $K_d^{NADPH} = 0.12mM$, $K_d^{iCiT} = 0.03mM$, $K_{einh}^{NADPH} = 0.007mM$, $K_{eknh}^{2KG} = 5.5mM$, $K_d^{CO_2} = 1.6mM$, $K_{eke}^{CO_2} = 1.6mM$, $K_{ekn}^{NADP} = 0.00016mM$, $K_m^{NADPH} = 0.0036mM$, $K_{enhe}^{NADPH} = 0.028mM$	Mogilevskaya et al., 2007
Icl	$V_{Icl\_f}^{\max} = 3.8315 mmol/gDCW.h$, $v_{Icl\_r}^{\max} = 2.585/100 mmol/gDCW.h$, $K_m^{iCiT} = 0.604mM$, $K_m^{SUC} = 0.59mM$, $K_m^{GOX} = 0.13mM$, $K_I^{ICL} = 0.003mM$	Singh and Ghosh, 2006
MS	$K_{MS\_f}^{\max} = 3.6968 mmol/gDCW.h$, $K_{MS\_r}^{\max} = 13.742/100 mmol/gDCW.h$, $K_m^{GOX} = 2mM$, $K_m^{AcCoA} = 0.01mM$, $K_m^{MAL} = 1mM$, $K_m^{CoA} = 0.1mM$	Singh and Ghosh, 2006
2KGDH	$v_{2KGDH}^{\max} = 137.435 mmol/gDCW.h$, $K_m^{2KG} = 1.0mM$, $K_I^{2KG} = 0.75mM$, $K_m^{CoA} = 0.002mM$, $K_m^{NAD} = 0.07mM$, $K_m^{SUC} = 1.0mM$, $K_m^{NADH} = 0.018mM$, $K_z = 1.5$	Wright et al., 1992
SDH	$K_m^{SUC} = 0.1$, $v_{SDH1} = 1.1334 mmol/gDCW.h$, $v_{SDH2} = 1.1334 mmol/gDCW.h$, $K_{eq} = 10$	Wright et al., 1992
MDH	$v_{MDH1} = 25.874 mmol/gDCW.h$, $v_{MDH2} = 25.874 mmol/gDCW.h$, $K_{eq} = 1.0$, $K_I^{NAD} = 0.31mM$, $K_I^{NADH} = 0.04mM$, $K_I^{MAL} = 3.30mM$, $K_I^{OAA} = 0.27mM$, $K_m^{NAD} = 0.1mM$, $K_m^{NADH} = 0.04mM$, $K_m^{MAL} = 1.33mM$, $K_m^{OAA} = 0.27mM$, $K_{II}^{NAD} = 0.31mM$, $K_{II}^{OAA} = 0.17mM$	Wright et al., 1992
FUM	$v_{FUM1} = 1.1334 mmol/gDCW.h$, $V_{FUM2} = 1.1334 mmol/gDCW.h$, $K_{eq} = 10$, $K_m^{FUM} = 0.1mM$	Wright et al., 1992
Ppc	$v_{Ppc}^{\max} = 0.1885 mmol/gDCW.h$, $K_m^{PEP} = 0.3231mM$, $k_1 = 0.03176$, $k_2 = 1.2878mM$, $k_3 = 0.05425mM$, $k_4 = 0.8139mM$, $k_5 = 0.0939mM$, $k_6 = 0.2693mM$	Lee et al., 1999
Pck	$v_{Pck}^{\max} = 4.5116 mmol/gDCW.h$, $K_m^{ATP} = 0.06mM$, $K_I^{ATP} = 0.04mM$, $K_i^{ATP} = 0.04mM$, $K_m^{OAA} = 0.67mM$, $K_i^{PEP} = 0.06mM$, $K_I^{OAA} = 0.67mM$, $K_m^{PEP} = 0.07mM$, $K_i^{ADP} = 0.04mM$	Yang et al., 2003
Mez	$v_{Mez}^{\max} = 0.069945 mmol/gDCW.h$, $K_m^{MAL} = 0.37mM$, $K_{eq} = 0.10$	Wright et al., 1992

Pta	$v_{Pta}^{max} = 12.585 mmol/gDCW.h$, $K_i^{AcCoA} = 0.2mM$, $K_m^P = 2.6mM$, $K_i^P = 2.6mM$, $K_m^{ACP} = 0.7mM$, $K_i^{ACP} = 0.2mM$, $K_i^{CoA} = 0.029mM$, $K_{eq} = 0.0281mM$	Hoefnagel et al., 2002
Ack	$v_{Ack}^{max} = 2865.3 mmol/gDCW.h$, $K_m^{ACP} = 0.16mM$, $K_m^{ADP} = 0.5mM$, $K_m^{ACE} = 7mM$, $K_m^{ATP} = 0.07mM$, $K_{eq} = 174.217mM$	Hoefnagel et al., 2002
Acs	$v_{Acs}^{max} = 17.9068 mmol/gDCW.h$ $K = 0.089971 mmol/gDCW.h$, $K_m = 0.07mM$	Fung et al., 2005
G6PDH	$v_{G6PDH}^{max} = 0.97922 mmol/gDCW.h$, $K_{G6P} = 14.4mM$, $K_{NADP} = 0.015mM$, $K_{NADPH}^{NADP} = 0.01mM$, $K_{NADPH}^{G6P} = 0.18mM$	Chassagnole et al., 2002
6PGDH	$v_{6GPDH}^{max} = 1.81 mmol/gDCW.h$, $K_{G6P} = 0.1mM$, $K_{NADP} = 0.028mM$, $K_{NADPinh} = 0.01mM$, $K_{ATPinh} = 3.0mM$	Chassagnole et al., 2002
Rpe	$v_{Rpe}^{max} = 18.485 mmol/gDCW.h$, $K_{eq}^{Rpe} = 1.4mM$	Chassagnole et al., 2002
Rpi	$v_{Rpi}^{max} = 13.318 mmol/gDCW.h$, $K_{eq}^{Rpi} = 4.0mM$	Chassagnole et al., 2002
TktA	$v_{TktA}^{max} = 29.348 mmol/gDCW.h$, $K_{eq}^{TktA} = 1.2mM$	Chassagnole et al., 2002
TktB	$v_{TktB}^{max} = 316.22 mmol/gDCW.h$, $K_{eq}^{TktB} = 10.0mM$	Chassagnole et al., 2002
Tal	$v_{Tal}^{max} = 24.499 mmol/gDCW.h$, $K_{eq}^{Tal} = 1.05mM$	Chassagnole et al., 2002

Cofactor concentrations used for the simulation

Cofactor	Concentrations	Original source
ADP	0.595 mM	Chassagnole et al., 2002
AMP	0.955 mM	Chassagnole et al., 2002
ATP	4.27 mM	Chassagnole et al., 2002
CoA	0.001mM	arbitrary
NAD	1.47 mM	Chassagnole et al., 2002
NADH	0.1 mM	Chassagnole et al., 2002
NADP	0.195 mM	Chassagnole et al., 2002
NADPH	0.062 mM	Chassagnole et al., 2002
P	10 mM	Hoefnagel et al., 2002

APPENDIX E

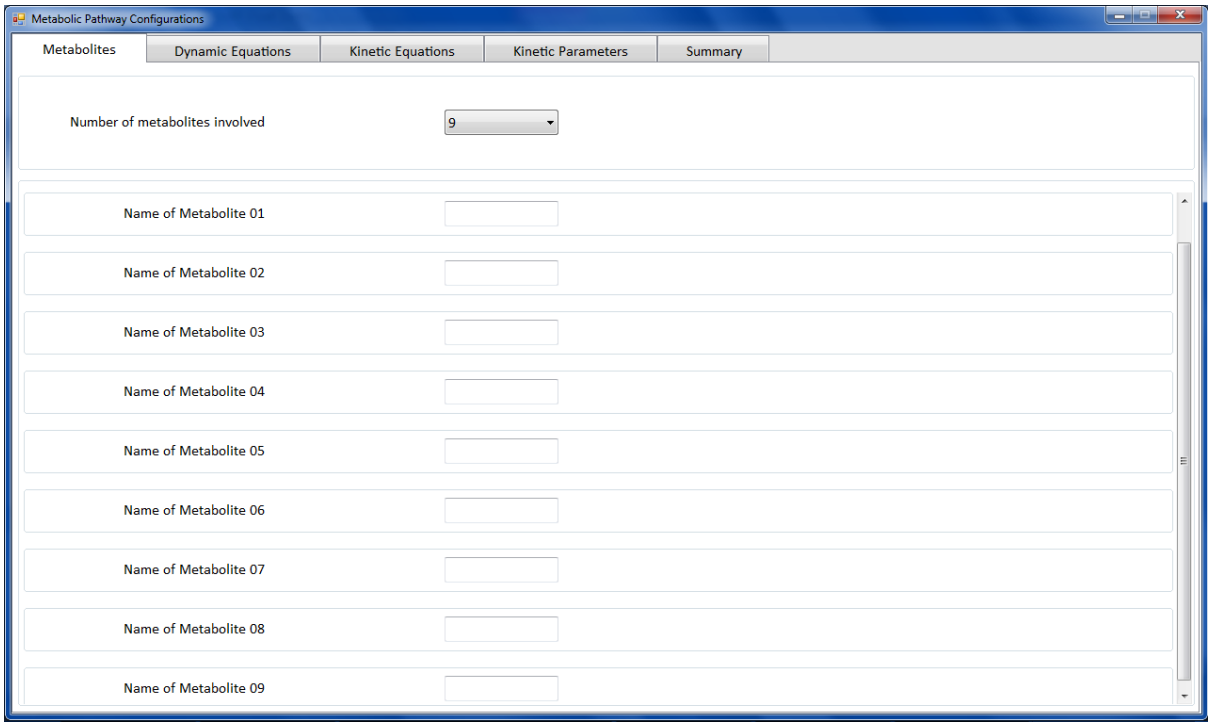
SOURCE CODE

Main Form


```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using MySql.Data;
using MySql.Data.MySqlClient;

namespace DynamicSimulator_v2
{
    public partial class frmMain : Form
    {
        public frmMain()
        {
            InitializeComponent();
        }

        #region Object Declarations
        SchemaName schemaForm = new SchemaName();
        frmMetCon form2 = new frmMetCon();
        CreateDatabase db = new CreateDatabase();
        selectFile open = new selectFile();
        frmReview form4 = new frmReview();
        #endregion
```

```
        #region Private Method
        private void btnExit_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void btnNew_Click(object sender, EventArgs e)
        {
            if (schemaForm.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                CreateDatabase();
            }
        }

        private void CreateDatabase()
        {
            try
            {
                db.Database();
                MessageBox.Show("New Project is successfully created");
                this.Hide();
                form2.Show();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void OpenDatabase()
        {
            try
            {
                this.Hide();
                form4.Show();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
        private void btnOpen_Click(object sender, EventArgs e)
        {
            if (open.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                OpenDatabase();
            }
        }

        #endregion
    }
}
```

Metabolites Configuration Form




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Text.RegularExpressions;

using MySql.Data;
using MySql.Data.MySqlClient;

namespace DynamicSimulator_v2
{
    public partial class frmMetCon : Form
    {
        public frmMetCon()
        {
            InitializeComponent();
        }
        #region Object Declaration
        private Database db;              // calling database
        private Binding bindData = new Binding();
        private UpdateDataGrid update;
        private ParserKineticParameter parse = new ParserKineticParameter();
        private FileMetaboliteConfig file = new FileMetaboliteConfig();
        private frmConfirmation sidefrm2 = new frmConfirmation();
        private frmGraphConfig graph = new frmGraphConfig();
        #endregion
```

```
        #region Local Variables
        private string metName01 = "", metName02 = " ", metName03 = "",
                       metName04 = "";
        private string metName05 = "", metName06 = "", metName07 = "", metName08 = "",
                       metName09 = "";
        #endregion

        private void UpdateMetaboliteName()
        {
            metName01 = txtMetName1.Text;
            metName02 = txtMetName2.Text;
            metName03 = txtMetName3.Text;
            metName04 = txtMetName4.Text;
            metName05 = txtMetName5.Text;
            metName06 = txtMetName6.Text;
            metName07 = txtMetName7.Text;
            metName08 = txtMetName8.Text;
            metName09 = txtMetName9.Text;

            lblMet1_0.Text = "d( " + metName01 + " )";
            lblMet2_0.Text = "d( " + metName02 + " )";
            lblMet3_0.Text = "d( " + metName03 + " )";
            lblMet4_0.Text = "d( " + metName04 + " )";
            lblMet5_0.Text = "d( " + metName05 + " )";
            lblMet6_0.Text = "d( " + metName06 + " )";
            lblMet7_0.Text = "d( " + metName07 + " )";
            lblMet8_0.Text = "d( " + metName08 + " )";
            lblMet9_0.Text = "d( " + metName09 + " )";

            lblInitialConMet1_2.Text = metName01;
            lblInitialConMet2_2.Text = metName02;
            lblInitialConMet3_2.Text = metName03;
            lblInitialConMet4_2.Text = metName04;
            lblInitialConMet5_2.Text = metName05;
            lblInitialConMet6_2.Text = metName06;
            lblInitialConMet7_2.Text = metName07;
            lblInitialConMet8_2.Text = metName08;
            lblInitialConMet9_2.Text = metName09;

            //Visibility of Groupbox of Metabolite Name
            if (!string.IsNullOrWhiteSpace(metName01))
            {
                grpMet01.Visible = true;
            }
            if (!string.IsNullOrWhiteSpace(metName02))
            {
                grpMet02.Visible = true;
            }
            if (!string.IsNullOrWhiteSpace(metName03))
            {
                grpMet03.Visible = true;
            }
            if (!string.IsNullOrWhiteSpace(metName04))
            {
                grpMet04.Visible = true;
            }
            if (!string.IsNullOrWhiteSpace(metName05))
            {
                grpMet05.Visible = true;
            }
```

```
            if (!string.IsNullOrWhiteSpace(metName06))
            {
                grpMet06.Visible = true;
            }
            if (!string.IsNullOrWhiteSpace(metName07))
            {
                grpMet07.Visible = true;
            }
            if (!string.IsNullOrWhiteSpace(metName08))
            {
                grpMet08.Visible = true;
            }
            if (!string.IsNullOrWhiteSpace(metName09))
            {
                grpMet09.Visible = true;
            }

            if (string.IsNullOrWhiteSpace(metName01))
            {
                grpMet01.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName02))
            {
                grpMet02.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName03))
            {
                grpMet03.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName04))
            {
                grpMet04.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName05))
            {
                grpMet05.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName06))
            {
                grpMet06.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName07))
            {
                grpMet07.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName08))
            {
                grpMet08.Visible = false;
            }
            if (string.IsNullOrWhiteSpace(metName09))
            {
                grpMet09.Visible = false;
            }

        }

        #region Button Controller
        private void btnTabOk_Click(object sender, EventArgs e)
        {
```

```
            if (sidefrm2.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            {
                graph.CustomName(this); // passing name for metabolites
                file.ConfigureInitialCondition();
                file.ConfigureParameter();
                file.ConfigureODE();
                this.Hide();
                graph.Show();
            }
        }
        #endregion

        #endregion

        #region Event Handling
        #region Input Keypress
        private void txtMetName1_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName2_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName3_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName4_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName5_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName6_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName7_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName8_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMetName9_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMet01_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }

        private void txtMet02_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
```

```
        private void txtMet03_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMet04_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMet05_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMet06_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMet07_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMet08_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtMet09_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtInitialConMet1_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }

            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }
        private void txtInitialConMet2_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }
            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }
```

```
        private void txtInitialConMet3_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }

            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }
        private void txtInitialConMet4_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }

            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }

        private void txtInitialConMet5_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }

            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }
        private void txtInitialConMet6_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }
```

```
        // checks to make sure only 1 decimal is allowed
        if (e.KeyChar == 46)
        {
            if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                e.Handled = true;
        }
    }

    private void txtInitialConMet7_KeyPress(object sender, KeyPressEventArgs e)
    {
        // allows 0-9, backspace, and decimal
        if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
           && e.KeyChar != 46))
        {
            e.Handled = true;
            return;
        }

        // checks to make sure only 1 decimal is allowed
        if (e.KeyChar == 46)
        {
            if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                e.Handled = true;
        }
    }
    private void txtInitialConMet8_KeyPress(object sender, KeyPressEventArgs e)
    {
        // allows 0-9, backspace, and decimal
        if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
           && e.KeyChar != 46))
        {
            e.Handled = true;
            return;
        }

        // checks to make sure only 1 decimal is allowed
        if (e.KeyChar == 46)
        {
            if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                e.Handled = true;
        }
    }
    private void txtInitialConMet9_KeyPress(object sender, KeyPressEventArgs e)
    {
        // allows 0-9, backspace, and decimal
        if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
           && e.KeyChar != 46))
        {
            e.Handled = true;
            return;
        }

        // checks to make sure only 1 decimal is allowed
        if (e.KeyChar == 46)
        {
            if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                e.Handled = true;
        }
    }
```

```
        private void txtKineticEq_KeyPress(object sender, KeyPressEventArgs e)
        {
            e.Handled = (e.KeyChar == (char)Keys.Space);
        }
        private void txtParameter_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }

            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }
        #endregion

        #region Leave Tabpage
        private void tbMet_Leave(object sender, EventArgs e)
        {
            txtMetName1_Validated(this, e);
            txtMetName2_Validated(this, e);
            txtMetName3_Validated(this, e);
            txtMetName4_Validated(this, e);
            txtMetName5_Validated(this, e);
            txtMetName6_Validated(this, e);
            txtMetName7_Validated(this, e);
            txtMetName8_Validated(this, e);
            UpdateMetaboliteName(); // Passing user key in to label

            if (!string.IsNullOrWhiteSpace(txtMetName1.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName2.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName3.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName4.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName5.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName6.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName7.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName8.Text) ||
                !string.IsNullOrWhiteSpace(txtMetName9.Text))
            {
                grpMass.Visible = true;
                db = new UpdateMetaboliteName(this);
            }
            else
            {
                grpMass.Visible = false;
            }
        }

        private void tbMassBal_Leave(object sender, EventArgs e)
        {
            txtMet01_Validated(this, e);
            txtMet02_Validated(this, e);
            txtMet03_Validated(this, e);
```

```
            txtMet04_Validated(this, e);
            txtMet05_Validated(this, e);
            txtMet06_Validated(this, e);
            txtMet07_Validated(this, e);
            txtMet08_Validated(this, e);
            txtMet09_Validated(this, e);
            txtInitialConMet1_Validated(this, e);
            txtInitialConMet2_Validated(this, e);
            txtInitialConMet3_Validated(this, e);
            txtInitialConMet4_Validated(this, e);
            txtInitialConMet5_Validated(this, e);
            txtInitialConMet6_Validated(this, e);
            txtInitialConMet7_Validated(this, e);
            txtInitialConMet8_Validated(this, e);

            //Condition for binding
            if ((!string.IsNullOrWhiteSpace(txtMet01.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet1.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet02.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet2.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet03.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet3.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet04.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet4.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet05.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet5.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet06.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet6.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet07.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet7.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet08.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet8.Text)) ||
                (!string.IsNullOrWhiteSpace(txtMet09.Text) &&
                 !string.IsNullOrWhiteSpace(txtInitialConMet9.Text)))
            {
                db = new UpdateMassBalance(this); // Update Mass Balance
            }
        }

        private void tbKinEq_Leave(object sender, EventArgs e)
        {
            bindData.BindingKineticEquations(this);
            parse.ParserParameter();
            bindData.BindingKineticParameter(this);
        }

        private void dbParameter_Leave(object sender, EventArgs e)
        {
            bindData.BindingKineticParameter(this);
        }

        #endregion

        #region Metabolite Name Error Provider



        private void txtMetName1_Validated(object sender, EventArgs e)
        {
```

```
        if (string.IsNullOrWhiteSpace(txtMetName1.Text))
        {
            this.errorProvider1.SetError(txtMetName1, "This field must contain
                                                       text");
        }
        else
        {
            this.errorProvider1.SetError(txtMetName1, "");
        }
    }
    private void txtMetName2_Validated(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtMetName2.Text))
        {
            this.errorProvider1.SetError(txtMetName2, "This field must contain
                                                       text");
        }
        else
        {
            this.errorProvider1.SetError(txtMetName2, "");
        }
    }
    private void txtMetName3_Validated(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtMetName3.Text))
        {
            this.errorProvider1.SetError(txtMetName3, "This field must contain
                                                       text");
        }
        else
        {
            this.errorProvider1.SetError(txtMetName3, "");
        }
    }
    private void txtMetName4_Validated(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtMetName4.Text))
        {
            this.errorProvider1.SetError(txtMetName4, "This field must contain
                                                       text");
        }
        else
        {
            this.errorProvider1.SetError(txtMetName4, "");
        }
    }
    private void txtMetName5_Validated(object sender, EventArgs e)
    {
        if (string.IsNullOrWhiteSpace(txtMetName5.Text))
        {
            this.errorProvider1.SetError(txtMetName5, "This field must contain
                                                       text");
        }
        else
        {
            this.errorProvider1.SetError(txtMetName5, "");
        }
    }
```

```
        private void txtMetName6_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMetName6.Text))
            {
                this.errorProvider1.SetError(txtMetName6, "This field must contain
                                                           text");
            }
            else
            {
                this.errorProvider1.SetError(txtMetName6, "");
            }
        }
        private void txtMetName7_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMetName7.Text))
            {
                this.errorProvider1.SetError(txtMetName7, "This field must contain
                                                           text");
            }
            else
            {
                this.errorProvider1.SetError(txtMetName7, "");
            }
        }
        private void txtMetName8_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMetName8.Text))
            {
                this.errorProvider1.SetError(txtMetName8, "This field must contain
                                                           text");
            }
            else
            {
                this.errorProvider1.SetError(txtMetName8, "");
            }
        }
        private void txtMetName9_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMetName9.Text))
            {
                this.errorProvider1.SetError(txtMetName9, "This field must contain
                                                           text");
            }
            else
            {
                this.errorProvider1.SetError(txtMetName9, "");
            }
        }
        #endregion

        #region Mass Balance Error Provider
        private void txtMet01_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet01.Text))
            {
                this.errorProvider1.SetError(txtMet01, "This field must contain
                                                        text");
            }
```

```
            else
            {
                this.errorProvider1.SetError(txtMet01, "");
            }
        }
        private void txtMet02_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet02.Text))
            {
                this.errorProvider1.SetError(txtMet02, "This field must contain
                                                        text");
            }
            else
            {
                this.errorProvider1.SetError(txtMet02, "");
            }
        }
        private void txtMet03_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet03.Text))
            {
                this.errorProvider1.SetError(txtMet03, "This field must contain
                                                        text");
            }
            else
            {
                this.errorProvider1.SetError(txtMet03, "");
            }
        }
        private void txtMet04_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet04.Text))
            {
                this.errorProvider1.SetError(txtMet04, "This field must contain
                                                        text");
            }
            else
            {
                this.errorProvider1.SetError(txtMet04, "");
            }
        }
        private void txtMet05_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet05.Text))
            {
                this.errorProvider1.SetError(txtMet05, "This field must contain
                                                        text");
            }
            else
            {
                this.errorProvider1.SetError(txtMet05, "");
            }
        }
        private void txtMet06_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet06.Text))
            {
                this.errorProvider1.SetError(txtMet06, "This field must contain
                                                        text");
            }
```

```
            else
            {
                this.errorProvider1.SetError(txtMet06, "");
            }
        }
        private void txtMet07_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet07.Text))
            {
                this.errorProvider1.SetError(txtMet07, "This field must contain
                                                        text");
            }
            else
            {
                this.errorProvider1.SetError(txtMet07, "");
            }
        }

        private void txtMet08_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet08.Text))
            {
                this.errorProvider1.SetError(txtMet08, "This field must contain
                                                        text");
            }
            else
            {
                this.errorProvider1.SetError(txtMet08, "");
            }
        }

        private void txtMet09_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtMet09.Text))
            {
                this.errorProvider1.SetError(txtMet09, "This field must contain
                                                        text");
            }
            else
            {
                this.errorProvider1.SetError(txtMet09, "");
            }
        }
        #endregion

        #region Initial Condition Error Provider

        private void txtInitialConMet1_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet1.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet1, "This field must
                                            contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet1, "");
            }
        }
```

```
        private void txtInitialConMet2_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet2.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet2, "This field must
                                            contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet2, "");
            }
        }
        private void txtInitialConMet3_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet3.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet3, "This field must
                                            contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet3, "");
            }
        }
        private void txtInitialConMet4_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet4.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet4, "This field must
                                            contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet4, "");
            }
        }
        private void txtInitialConMet5_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet5.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet5, "This field must
                                            contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet5, "");
            }
        }
        private void txtInitialConMet6_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet6.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet6, "This field must
                                            contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet6, "");
            }
        }
```

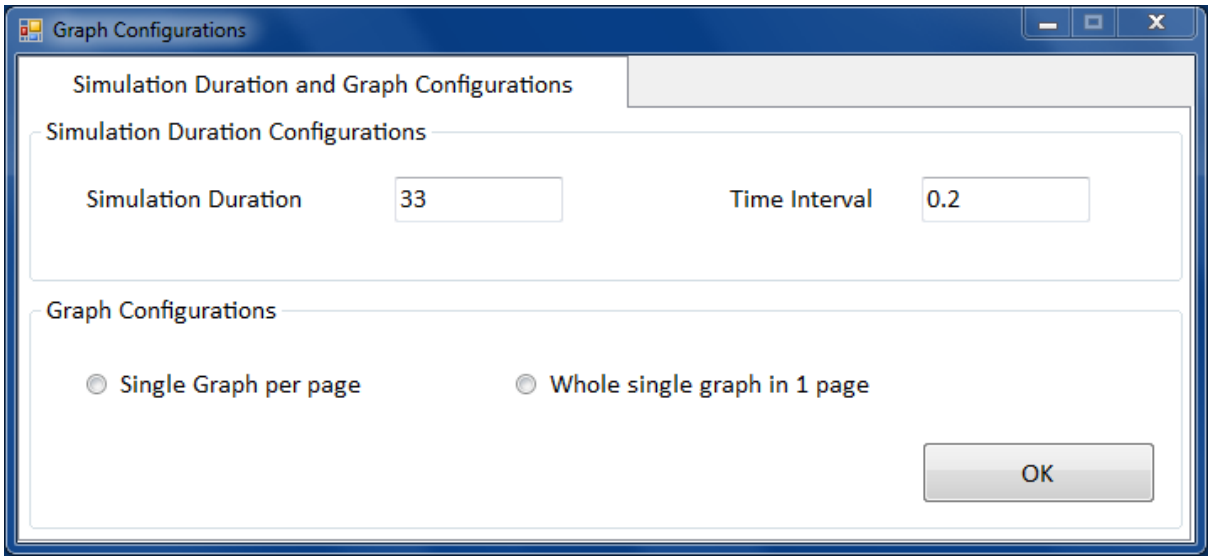
```
        private void txtInitialConMet7_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet7.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet7, "This field must
                                             contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet7, "");
            }
        }
        private void txtInitialConMet8_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet8.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet8, "This field must
                                             contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet8, "");
            }
        }
        private void txtInitialConMet9_Validated(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(txtInitialConMet9.Text))
            {
                this.errorProvider2.SetError(txtInitialConMet9, "This field must
                                             contain integer value");
            }
            else
            {
                this.errorProvider2.SetError(txtInitialConMet9, "");
            }
        }
        #endregion

        #region Mouse Hovering
        private void pnlRename_MouseHover(object sender, EventArgs e)
        {
            pnlRename.TabIndex = 0;
            if (!pnlRename.Focused)
            {
                pnlRename.Focus();
            }
        }

        private void pnlMassBal_MouseHover(object sender, EventArgs e)
        {
            pnlMassBal.TabIndex = 0;
            if (!pnlMassBal.Focused)
            {
                pnlMassBal.Focus();
            }
        }

        private void pnlSum_MouseHover(object sender, EventArgs e)
        {
            pnlSum.TabIndex = 0;
```

```
            if (!pnlSum.Focused)
            {
                pnlSum.Focus();
            }
        }
        #endregion

        #region Selected Index Change for combobox in tabpage Metabolite
        private void cmbMet_SelectedIndexChanged(object sender, EventArgs e)
        {
            grpMetName.Visible = true;
            switch (cmbMet.SelectedIndex)
            {
                case 0:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = false;
                    grpMetName04.Visible = false;
                    grpMetName05.Visible = false;
                    grpMetName06.Visible = false;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;
                case 1:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = false;
                    grpMetName05.Visible = false;
                    grpMetName06.Visible = false;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;
                case 2:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = true;
                    grpMetName05.Visible = false;
                    grpMetName06.Visible = false;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;
                case 3:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
                    grpMetName04.Visible = true;
                    grpMetName05.Visible = true;
                    grpMetName06.Visible = false;
                    grpMetName07.Visible = false;
                    grpMetName08.Visible = false;
                    grpMetName09.Visible = false;
                    break;
                case 4:
                    grpMetName01.Visible = true;
                    grpMetName02.Visible = true;
                    grpMetName03.Visible = true;
```

```
                grpMetName04.Visible = true;
                grpMetName05.Visible = true;
                grpMetName06.Visible = true;
                grpMetName07.Visible = false;
                grpMetName08.Visible = false;
                grpMetName09.Visible = false;
                break;
            case 5:
                grpMetName01.Visible = true;
                grpMetName02.Visible = true;
                grpMetName03.Visible = true;
                grpMetName04.Visible = true;
                grpMetName05.Visible = true;
                grpMetName06.Visible = true;
                grpMetName07.Visible = true;
                grpMetName08.Visible = false;
                grpMetName09.Visible = false;
                break;
            case 6:
                grpMetName01.Visible = true;
                grpMetName02.Visible = true;
                grpMetName03.Visible = true;
                grpMetName04.Visible = true;
                grpMetName05.Visible = true;
                grpMetName06.Visible = true;
                grpMetName07.Visible = true;
                grpMetName08.Visible = true;
                grpMetName09.Visible = false;
                break;
            case 7:
                grpMetName01.Visible = true;
                grpMetName02.Visible = true;
                grpMetName03.Visible = true;
                grpMetName04.Visible = true;
                grpMetName05.Visible = true;
                grpMetName06.Visible = true;
                grpMetName07.Visible = true;
                grpMetName08.Visible = true;
                grpMetName09.Visible = true;
                break;
        }
    }
    #endregion

    #region Edit table
    private void dbKineticName_CellClick(object sender,
                                         DataGridViewCellEventArgs e)
    {
        dbKineticName.Columns[0].ReadOnly = true;
    }

    private void dbKineticName_CellEndEdit(object sender,
                                           DataGridViewCellEventArgs e)
    {
        update = new UpdateBindingKineticEquations(this);
        bindData.BindingKineticParameter(this);
    }

    private void dbParameter_CellClick(object sender,
                                       DataGridViewCellEventArgs e)
    {
```

```
            dbParameter.Columns[0].ReadOnly = true;
        }

        private void dbParameter_CellEndEdit(object sender,
                                             DataGridViewCellEventArgs e)
        {
            update = new UpdateBindingKineticParameter(this);
            bindData.BindingKineticParameter(this);
        }
        #endregion

        #endregion
    }
}
```

Graph Configuration Form



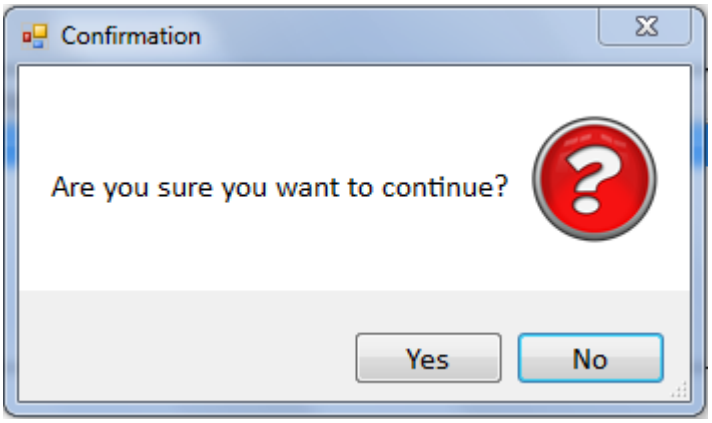

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using MLApp;

namespace DynamicSimulator_v2
{
    public partial class frmGraphConfig : Form
    {
        #region Object Declarations
        private MLApp.MLApp matlab;
        private FileMetaboliteConfig file = new FileMetaboliteConfig();
```

```
        #endregion

        #region Variables
        private string path = AppDomain.CurrentDomain.BaseDirectory; // enter debug
                                                                     folder
        #endregion

        public frmGraphConfig()
        {
            InitializeComponent();
            matlab = new MLApp.MLApp();
            matlab.Execute("cd ('" + path + "')");
            matlab.Execute("addpath('" + path + "')");
            matlab.Execute("initialCons;");
            matlab.Execute("parameter;");
        }

        #region Event Handler
        private void txtSimDuration_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }

            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }

        private void txtTimeInterval_KeyPress(object sender, KeyPressEventArgs e)
        {
            // allows 0-9, backspace, and decimal
            if (((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8
               && e.KeyChar != 46))
            {
                e.Handled = true;
                return;
            }

            // checks to make sure only 1 decimal is allowed
            if (e.KeyChar == 46)
            {
                if ((sender as TextBox).Text.IndexOf(e.KeyChar) != -1)
                    e.Handled = true;
            }
        }

        private void txtSimDuration_Leave(object sender, EventArgs e)
        {
            file.ConfigureMain(this);       // full graphs
            file.ConfigureMainSingle(this); // Single graphs
        }
```

```
        private void txtTimeInterval_Leave(object sender, EventArgs e)
        {
            double interval = double.Parse(txtTimeInterval.Text);
            double max = double.Parse(txtSimDuration.Text);
            if (interval > max)
            {
                MessageBox.Show("Time Interval Value is not valid.
                                 \nPlease re-enter again");
                txtTimeInterval.Text = "0";
            }
        }

        #endregion

        public void CustomName(frmMetCon frm2)
        {
            // Metabolite 01
            grpGraphMet01.Text = frm2.txtMetName1.Text;
            // Metabolite 02
            grpGraphMet02.Text = frm2.txtMetName2.Text;
            // Metabolite 03
            grpGraphMet03.Text = frm2.txtMetName3.Text;
            // Metabolite 04
            grpGraphMet04.Text = frm2.txtMetName4.Text;
            // Metabolite 05
            grpGraphMet05.Text = frm2.txtMetName5.Text;
            // Metabolite 06
            grpGraphMet06.Text = frm2.txtMetName6.Text;
            // Metabolite 07
            grpGraphMet07.Text = frm2.txtMetName7.Text;
            // Metabolite 08
            grpGraphMet08.Text = frm2.txtMetName8.Text;
            // Metabolite 09
            grpGraphMet09.Text = frm2.txtMetName9.Text;

        }

        private void btnOKSim_Click(object sender, EventArgs e)
        {
            if (rdNineGraphs.Checked)
            {
                file.ConfigureMain(this);       //  for 9 graph
                file.ConfigureMainSingle(this); //  for single graph
                matlab.Execute("ODE.m;");
                matlab.Execute("Main.m;");

                frmLog log = new frmLog();
                log.Show();
                log.txtMatlabLog.AppendText(matlab.Execute("ODE"));
                log.txtMatlabLog.AppendText(matlab.Execute("Main"));
                log.txtMatlabLog.AppendText(matlab.Execute("pwd"));

            }

            if (rdSingleGraph.Checked)
            {
                file.ConfigureMain(this);       //  for 9 graph
                file.ConfigureMainSingle(this); //  for single graph
                matlab.Execute("ODE.m;");
                matlab.Execute("MainSingle.m;");
```

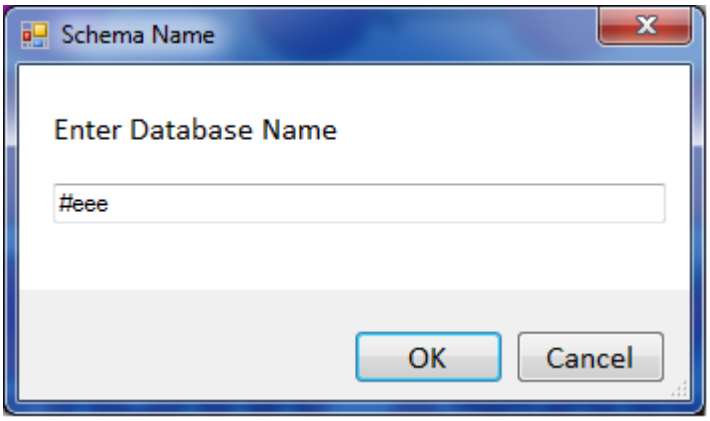
```
                frmLog log = new frmLog();
                log.Show();
                log.txtMatlabLog.AppendText(matlab.Execute("ODE"));
                log.txtMatlabLog.AppendText(matlab.Execute("Main"));
                log.txtMatlabLog.AppendText(matlab.Execute("pwd"));
            }
        }
    }
}
```

Confirmation Form




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace DynamicSimulator_v2
{
    public partial class frmConfirmation : Form
    {
        public frmConfirmation()
        {
            InitializeComponent();
        }

        private void btnYes_Click(object sender, EventArgs e)
        {
            this.Hide();
        }

        private void btnNo_Click(object sender, EventArgs e)
        {
            this.Hide();
        }
    }
}
```

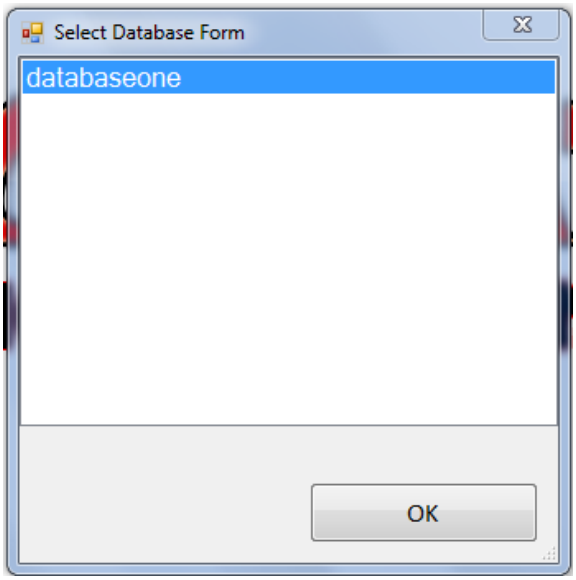
Schema Form




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Text.RegularExpressions;

namespace DynamicSimulator_v2
{
    public partial class SchemaName : Form
    {
        private static string data;

        private selectFile createFile;

        public SchemaName()
        {
            InitializeComponent();
        }

        private void btnCancel_Click(object sender, EventArgs e)
        {
            this.Hide();
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            createFile = new selectFile();
            data = txtDB.Text;
            createFile.CreateListFile(data);
            this.Hide();
        }

        public string getData
        {
            set
            {
                data = txtDB.Text;
            }
```

```
            get
            {
                return data;
            }
        }

        private void txtDB_KeyPress(object sender, KeyPressEventArgs e)
        {
           e.Handled = (e.KeyChar == (char)Keys.Space);
           if (Char.IsControl(e.KeyChar) != true && Char.IsNumber(e.KeyChar) == true)
           {
              e.Handled = true;
           }
        }

    }
}
```

Select Database




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using System.IO;

namespace DynamicSimulator_v2
{
```

```
    public partial class selectFile : Form
    {
        #region Variable
        private string path = AppDomain.CurrentDomain.BaseDirectory; // enter debug
                                                                     folder
        private string FileLocation = "";
        #endregion

        #region Object Declaration
        private StreamWriter file;
        private OpenDatabase open = new OpenDatabase();
        #endregion
        public selectFile()
        {
            InitializeComponent();
            readFile();
        }

        private void btnOK_Click(object sender, EventArgs e)
        {
            string getDatabase;
            if (lstDatabase.Items.Count > 0) // items exist
            {
                getDatabase = lstDatabase.GetItemText(lstDatabase.SelectedItem);
                open.setupOpenDatabase(getDatabase);
                this.Hide();
            }
            else
            {
                this.DialogResult = DialogResult.None;
                MessageBox.Show("List Box is empty!!");
                this.Hide();
            }

        }

        public void CreateListFile(string database)
        {
            FileLocation = path + "\\listdatabase.txt";
            file = new StreamWriter(FileLocation, true);

            file.WriteLine(database+"\n");
            file.Close();
        }

        private void readFile()
        {

            FileLocation = path + "\\listdatabase.txt";

            StreamReader readLine = new StreamReader(FileLocation);

            string[] lines= File.ReadAllLines(FileLocation);
            var result = lines.Distinct();
            lstDatabase.DataSource = result.ToList();
            readLine.Close();
        }
    }
}
```

Binding Database

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using MySql.Data;
using MySql.Data.MySqlClient;
using System.Data;


namespace DynamicSimulator_v2
{
    class Binding
    {
        #region Object Declaration
        SchemaName schemaForm;
        MySqlConnection conn;
        MySqlCommand command;
        BindingSource bindingSource;
        DataTable dtable;
        #endregion

        #region Local Variables
        private string connStr = "";
        private string database = "";
        #endregion

        #region Public Method

        public void BindingMetaboliteName(frmMetCon form2)
        {
            schemaForm = new SchemaName();
            connStr = "datasource=localhost;port=3306;username=root;password=root;";
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
            database = schemaForm.getData;

            try
            {
                dtable = new DataTable();
                bindingSource = new BindingSource();

                conn.Open();
                form2.dbSumMetName.DataSource = null;
                form2.dbSumMetName.ResetBindings();
                form2.dbSumMetName.Rows.Clear();
                form2.dbSumMetName.AutoGenerateColumns = false;
                command.CommandText = "SELECT MetaboliteID, Metabolite_Name FROM "
                                        +database +
                                      ".Metabolites " +
                                      "WHERE Metabolite_Name IS NOT NULL " +
                                      "AND Metabolite_Name <> '';"; //Select not empty
                                                                    metabolite name
                MySqlDataReader dr = command.ExecuteReader
                                     (CommandBehavior.CloseConnection);
                dtable.Load(dr);
```

```
            for (int i = 0; i < dtable.Rows.Count; i++)
            {
                form2.dbSumMetName.Rows.Add();
                form2.dbSumMetName.Rows[form2.dbSumMetName.Rows.Count - 1].Cells
               ["MetaboliteID"].Value = dtable.Rows[i]["MetaboliteID"].ToString();
                form2.dbSumMetName.Rows[form2.dbSumMetName.Rows.Count - 1].Cells
               ["Metabolite_Name"].Value = dtable.Rows[i]
                                           ["Metabolite_Name"].ToString();
            }

            conn.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }

    public void BindingMassBalance(frmMetCon frm)
    {
        schemaForm = new SchemaName();
        connStr = "datasource=localhost;port=3306;username=root;password=root;";
        conn = new MySqlConnection(connStr);
        command = conn.CreateCommand();
        database = schemaForm.getData;

        try
        {
            dtable = new DataTable();
            bindingSource = new BindingSource();
            conn.Open();
            frm.dbSumMassBal.DataSource = null;
            frm.dbSumMassBal.ResetBindings();
            frm.dbSumMassBal.Rows.Clear();
            frm.dbSumMassBal.AutoGenerateColumns = false;
            command.CommandText = "SELECT Metabolite_Name, Mass_Balance,
                                   Initial_Condition FROM " + database +
                                  ".Metabolites " +
                                  "WHERE Metabolite_Name IS NOT NULL " +
                                  "AND Metabolite_Name <> '';";
            MySqlDataReader dr = command.ExecuteReader
                                 (CommandBehavior.CloseConnection);
            dtable.Load(dr);
            for (int i = 0; i < dtable.Rows.Count; i++)
            {
                frm.dbSumMassBal.Rows.Add();
                frm.dbSumMassBal.Rows[frm.dbSumMassBal.Rows.Count - 1].Cells
                ["MassBalances"].Value = "d( " + dtable.Rows[i]
                ["Metabolite_Name"].ToString()+ " )/dt";

                frm.dbSumMassBal.Rows[frm.dbSumMassBal.Rows.Count - 1].Cells
                ["MassBalanceEquations"].Value = dtable.Rows[i]
                ["Mass_Balance"].ToString();
                frm.dbSumMassBal.Rows[frm.dbSumMassBal.Rows.Count - 1].Cells
                ["InitialConditions"].Value = dtable.Rows[i]
                ["Initial_Condition"].ToString();
            }
            conn.Close();
        }
```

```
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        public void BindingKineticEquations(frmMetCon frm2)
        {
            schemaForm = new SchemaName();
            connStr = "datasource=localhost;port=3306;username=root;password=root;";
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
            database = schemaForm.getData;

            try
            {
                dtable = new DataTable();
                bindingSource = new BindingSource();
                conn.Open();
                frm2.dbKineticName.DataSource = null;
                frm2.dbKineticName.ResetBindings();
                frm2.dbKineticName.Rows.Clear();
                frm2.dbKineticName.AutoGenerateColumns = false;
                command.CommandText = "SELECT Kinetic_Name, Equations FROM "
                                                        +database +
                                      ".EnzymeKinetics " +
                                      "WHERE CONCAT ('',Kinetic_Name*1)<>
                                       Kinetic_Name";
                MySqlDataReader dr = command.ExecuteReader
                                     (CommandBehavior.CloseConnection);
                dtable.Load(dr);
                for (int i = 0; i < dtable.Rows.Count; i++)
                {
                   frm2.dbKineticName.Rows.Add();
                   frm2.dbKineticName.Rows[frm2.dbKineticName.Rows.Count - 1].Cells
                   ["Kinetic_Name"].Value = dtable.Rows[i]["Kinetic_Name"].ToString();
                   frm2.dbKineticName.Rows[frm2.dbKineticName.Rows.Count - 1].Cells
                   ["Equations"].Value = dtable.Rows[i]["Equations"].ToString();
                }

                frm2.dbSumKinEq.DataSource = null;
                frm2.dbSumKinEq.ResetBindings();
                frm2.dbSumKinEq.Rows.Clear();
                frm2.dbSumKinEq.AutoGenerateColumns = false;
                for (int i = 0; i < dtable.Rows.Count; i++)
                {
                  frm2.dbSumKinEq.Rows.Add();
                  frm2.dbSumKinEq.Rows[frm2.dbSumKinEq.Rows.Count - 1].Cells
                  ["Kinetic_Names"].Value = dtable.Rows[i]["Kinetic_Name"].ToString();
                  frm2.dbSumKinEq.Rows[frm2.dbSumKinEq.Rows.Count - 1].Cells
                  ["KEquations"].Value = dtable.Rows[i]["Equations"].ToString();
                }
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
```

```
public void BindingKineticParameter(frmMetCon frm2)
{
    schemaForm = new SchemaName();
    connStr = "datasource=localhost;port=3306;username=root;password=root;";
    conn = new MySqlConnection(connStr);
    command = conn.CreateCommand();
    database = schemaForm.getData;

    try
    {
        #region Bind Parameter
        dtable = new DataTable();
        bindingSource = new BindingSource();
        conn.Open();
        frm2.dbParameter.DataSource = null;
        frm2.dbParameter.ResetBindings();
        frm2.dbParameter.Rows.Clear();
        frm2.dbParameter.AutoGenerateColumns = false;
        command.CommandText = "SELECT Parameter_Name, Parameter_Value FROM "
                                                    + database +
                              ".KineticParameters " +
                              "WHERE CONCAT ('',Parameter_Name*1)<>
                               Parameter_Name";
        MySqlDataReader dr = command.ExecuteReader
                             (CommandBehavior.CloseConnection);
        dtable.Load(dr);
        for (int i = 0; i < dtable.Rows.Count; i++)
        {
            frm2.dbParameter.Rows.Add();
            frm2.dbParameter.Rows[frm2.dbParameter.Rows.Count - 1].Cells
            ["ParameterName"].Value = dtable.Rows[i]
            ["Parameter_Name"].ToString();
            frm2.dbParameter.Rows[frm2.dbParameter.Rows.Count - 1].Cells
            ["ParameterValue"].Value = dtable.Rows[i]
            ["Parameter_Value"].ToString();
        }

        frm2.dbSumKinPar.DataSource = null;
        frm2.dbSumKinPar.ResetBindings();
        frm2.dbSumKinPar.Rows.Clear();
        frm2.dbSumKinPar.AutoGenerateColumns = false;
        for (int i = 0; i < dtable.Rows.Count; i++)
        {
            frm2.dbSumKinPar.Rows.Add();
            frm2.dbSumKinPar.Rows[frm2.dbSumKinPar.Rows.Count - 1].Cells
            ["ParameterNames"].Value = dtable.Rows[i]
            ["Parameter_Name"].ToString();
            frm2.dbSumKinPar.Rows[frm2.dbSumKinPar.Rows.Count - 1].Cells
            ["KineticParameterValues"].Value = dtable.Rows[i]
            ["Parameter_Value"].ToString();
        }
        int counter = dtable.Rows.Count;
        conn.Close();
        #endregion

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
```

```
        }
        #endregion
    }
}
```

Create Database

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using MySql.Data;
using MySql.Data.MySqlClient;

namespace DynamicSimulator_v2
{
    class CreateDatabase
    {
        #region Object Declaration
        MySqlConnection conn;
        SchemaName schemaForm = new SchemaName();
        #endregion

        #region Public Properties

        public void Database()
        {
            string connStr = "datasource=localhost;port=3306;
                              username=root;password=root;";
            string database = schemaForm.getData;
            conn = new MySqlConnection(connStr);

            MySqlCommand command = conn.CreateCommand();
            conn.Open();
            command.CommandText = "DROP DATABASE IF EXISTS " + database;
            command.ExecuteNonQuery();
            command.CommandText = "CREATE DATABASE " + database;
            command.ExecuteNonQuery();

            //Coding for PRELIMINARY METABOLITES
            command.CommandText = "CREATE TABLE " + database +
                                  ".Metabolites(" +
                                  "MetaboliteID VARCHAR(30) NOT NULL," +
                                  "Metabolite_Name VARCHAR(600) NULL," +
                                  "Mass_Balance VARCHAR(600) NULL," +
                                  "Initial_Condition DOUBLE NULL," +
                                  "PRIMARY KEY (MetaboliteID));";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite01');";
            command.ExecuteNonQuery();
```

```
            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite02');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite03');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite04');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite05');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite06');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite07');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite08');";
            command.ExecuteNonQuery();

            command.CommandText = "INSERT INTO " + database +
                                  ".Metabolites " +
                                  "(MetaboliteID)" +
                                  "VALUES " +
                                  "('Metabolite09');";
            command.ExecuteNonQuery();

            command.CommandText = "CREATE TABLE " + database +
                                  ".AlteredMetabolites(" +
                                  "AMetaboliteID DOUBLE NOT NULL," +
                                  "AMetaboliteValue DOUBLE NULL," +
                                  "PRIMARY KEY (AMetaboliteID));";
```

```
            command.ExecuteNonQuery();

            conn.Close();

            TableEnzymeKinetic();
            TableKineticParameters();
        }

        public void TableEnzymeKinetic()
        {
            string connStr = "datasource=localhost;port=3306;
                              username=root;password=root;";
            string database = schemaForm.getData;
            conn = new MySqlConnection(connStr);

            MySqlCommand command = conn.CreateCommand();
            conn.Open();
            command.CommandText = "CREATE TABLE " + database +
                                  ".EnzymeKinetics(" +
                                  "KineticID VARCHAR(30) NOT NULL," +
                                  "Kinetic_Name VARCHAR(800) NULL," +
                                  "Equations VARCHAR(800) NULL," +
                                  "PRIMARY KEY (KineticID));";
            command.ExecuteNonQuery();

            conn.Close();
        }

        public void TableKineticParameters()
        {
            string connStr = "datasource=localhost;port=3306;
                              username=root;password=root;";
            string database = schemaForm.getData;
            conn = new MySqlConnection(connStr);

            MySqlCommand command = conn.CreateCommand();
            conn.Open();
            command.CommandText = "CREATE TABLE " + database +
                                  ".KineticParameters(" +
                                  "ParameterID VARCHAR(30) NOT NULL," +
                                  "Parameter_Name VARCHAR(800) NULL," +
                                  "Parameter_Value DOUBLE NULL," +
                                  "PRIMARY KEY (ParameterID));";
            command.ExecuteNonQuery();

            command.CommandText = "CREATE TABLE " + database +
                                  ".AlteredKineticParameters(" +
                                  "AParameterID DOUBLE NOT NULL," +
                                  "AParameterValue DOUBLE NULL," +
                                  "PRIMARY KEY (AParameterID));";
            command.ExecuteNonQuery();
            conn.Close();
        }
        #endregion
    }
}
```

Update Database

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using MySql.Data;
using MySql.Data.MySqlClient;
using System.Windows.Forms;
using System.Data;
using System.Drawing;

namespace DynamicSimulator_v2
{
    class Database
    {
        #region Local Variable
        protected string connStr = "";
        protected string database = "";
        protected string combineText = "";
        #endregion

        #region Object Declaration
        protected MySqlConnection conn;           // connection for database
        protected SchemaName schemaForm;          // calling database name
        protected Tokenizer tokenNew;
        protected Binding bind = new Binding();   // bind database
        protected MySqlCommand command;           // Query SQL
        #endregion

        #region Constructor
        public Database(frmMetCon frm2) // parent of database
        {
            schemaForm = new SchemaName();
            connStr = "datasource=localhost;port=3306;username=root;password=root;";
            database = schemaForm.getData;
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
        }
        #endregion

    }

    class UpdateMetaboliteName : Database
    {
        public UpdateMetaboliteName(frmMetCon frm2) // Update Metabolite Name in
                                                    database
            : base(frm2)
        {
             conn.Open();
                try
                {
                    if (!string.IsNullOrWhiteSpace(frm2.txtMetName1.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "'" + frm2.txtMetName1.Text + "'" +
                                              "WHERE MetaboliteID = 'Metabolite01'";
```

```
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMetName2.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Metabolite_Name = " +
                                      "'" + frm2.txtMetName2.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite02'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMetName3.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Metabolite_Name = " +
                                      "'" + frm2.txtMetName3.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite03'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMetName4.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Metabolite_Name = " +
                                      "'" + frm2.txtMetName4.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite04'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMetName5.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Metabolite_Name = " +
                                      "'" + frm2.txtMetName5.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite05'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMetName6.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Metabolite_Name = " +
                                      "'" + frm2.txtMetName6.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite06'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMetName7.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Metabolite_Name = " +
                                      "'" + frm2.txtMetName7.Text + "'" +
                                      "WHERE MetaboliteID = 'Metabolite07'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtMetName8.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
```

```
                                "SET Metabolite_Name = " +
                                "'" + frm2.txtMetName8.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite08'";
            command.ExecuteNonQuery();
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMetName9.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Metabolite_Name = " +
                                "'" + frm2.txtMetName9.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite09'";
            command.ExecuteNonQuery();
        }

        if (string.IsNullOrWhiteSpace(frm2.txtMetName1.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Metabolite_Name = " +
                                "' '" +
                                "WHERE MetaboliteID = 'Metabolite01'";
            command.ExecuteNonQuery();
        }
        if (string.IsNullOrWhiteSpace(frm2.txtMetName2.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Metabolite_Name = " +
                                "' '" +
                                "WHERE MetaboliteID = 'Metabolite02'";
            command.ExecuteNonQuery();
        }
        if (string.IsNullOrWhiteSpace(frm2.txtMetName3.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Metabolite_Name = " +
                                "' '" +
                                "WHERE MetaboliteID = 'Metabolite03'";
            command.ExecuteNonQuery();
        }
        if (string.IsNullOrWhiteSpace(frm2.txtMetName4.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Metabolite_Name = " +
                                "' '" +
                                "WHERE MetaboliteID = 'Metabolite04'";
            command.ExecuteNonQuery();
        }
        if (string.IsNullOrWhiteSpace(frm2.txtMetName5.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Metabolite_Name = " +
                                "' '" +
                                "WHERE MetaboliteID = 'Metabolite05'";
            command.ExecuteNonQuery();
        }
```

```
                    if (string.IsNullOrWhiteSpace(frm2.txtMetName6.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "' '" +
                                              "WHERE MetaboliteID = 'Metabolite06'";
                        command.ExecuteNonQuery();
                    }
                    if (string.IsNullOrWhiteSpace(frm2.txtMetName7.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "' '" +
                                              "WHERE MetaboliteID = 'Metabolite07'";
                        command.ExecuteNonQuery();
                    }
                    if (string.IsNullOrWhiteSpace(frm2.txtMetName8.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "' '" +
                                              "WHERE MetaboliteID = 'Metabolite08'";
                        command.ExecuteNonQuery();
                    }
                    if (string.IsNullOrWhiteSpace(frm2.txtMetName9.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Metabolite_Name = " +
                                              "' '" +
                                              "WHERE MetaboliteID = 'Metabolite09'";
                        command.ExecuteNonQuery();
                    }
                    conn.Close();
                    bind.BindingMetaboliteName(frm2);
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
            }
        }

        class UpdateMassBalance : Database
        {
            public UpdateMassBalance(frmMetCon frm2)  //Update Mass Balance in database
and parse the equations
                : base(frm2)
            {
                try
                {
                    conn.Open();
                    if (!string.IsNullOrWhiteSpace(frm2.txtMet01.Text))
                    {
                        command.CommandText = "UPDATE " + database +
                                              ".Metabolites " +
                                              "SET Mass_Balance = " +
```

```
                                "'" + frm2.txtMet01.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite01'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet01.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet02.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
                                "'" + frm2.txtMet02.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite02'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet02.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet03.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
                                "'" + frm2.txtMet03.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite03'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet03.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet04.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
                                "'" + frm2.txtMet04.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite04'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet04.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet05.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
                                "'" + frm2.txtMet05.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite05'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet05.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet06.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
                                "'" + frm2.txtMet06.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite06'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet06.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet07.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
```

```
                                "'" + frm2.txtMet07.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite07'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet07.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet08.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
                                "'" + frm2.txtMet08.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite08'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet08.Text;
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtMet09.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Mass_Balance = " +
                                "'" + frm2.txtMet09.Text + "'" +
                                "WHERE MetaboliteID = 'Metabolite09'";
            command.ExecuteNonQuery();
            combineText += "+" + frm2.txtMet09.Text;
        }


        if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet1.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Initial_Condition = " +
                                "'"
                                double.Parse(frm2.txtInitialConMet1.Text)
                                + "'" +
                                "WHERE MetaboliteID = 'Metabolite01'";
            command.ExecuteNonQuery();
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet2.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Initial_Condition = " +
                                "'" +
                                double.Parse(frm2.txtInitialConMet2.Text)
                                + "'" +
                                "WHERE MetaboliteID = 'Metabolite02'";
            command.ExecuteNonQuery();
        }
        if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet3.Text))
        {
            command.CommandText = "UPDATE " + database +
                                ".Metabolites " +
                                "SET Initial_Condition = " +
                                "'" +
                                double.Parse(frm2.txtInitialConMet3.Text)
                                + "'" +
                                "WHERE MetaboliteID = 'Metabolite03'";
            command.ExecuteNonQuery();
        }
```

```
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet4.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                                      double.Parse(frm2.txtInitialConMet4.Text) ]
                                      + "'" +
                                      "WHERE MetaboliteID = 'Metabolite04'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet5.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                                      double.Parse(frm2.txtInitialConMet5.Text)
                                      + "'" +
                                      "WHERE MetaboliteID = 'Metabolite05'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet6.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                                      double.Parse(frm2.txtInitialConMet6.Text)
                                      + "'" +
                                      "WHERE MetaboliteID = 'Metabolite06'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet7.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                                      double.Parse(frm2.txtInitialConMet7.Text)
                                      + "'" +
                                      "WHERE MetaboliteID = 'Metabolite07'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet8.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
                                      "'" +
                                      double.Parse(frm2.txtInitialConMet8.Text)
                                      + "'" +
                                      "WHERE MetaboliteID = 'Metabolite08'";
                command.ExecuteNonQuery();
            }
            if (!string.IsNullOrWhiteSpace(frm2.txtInitialConMet9.Text))
            {
                command.CommandText = "UPDATE " + database +
                                      ".Metabolites " +
                                      "SET Initial_Condition = " +
```

```
                                        "'" +
                                        double.Parse(frm2.txtInitialConMet9.Text)
                                        + "'" +
                                        "WHERE MetaboliteID = 'Metabolite09'";
                    command.ExecuteNonQuery();
                }
                conn.Close();
                tokenNew = new Tokenizer();
                tokenNew.TextTokenizer(combineText);
                bind.BindingKineticEquations(frm2); // Binding for summary of Kinetic
                                                    // Equations and
                                                    // at tab page of Kinetic
                                                    // Equations

                bind.BindingMassBalance(frm2); // Binding for summary of Mass Balance
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }

    class UpdateKineticEquations : Database
    {
        public UpdateKineticEquations(frmMetCon frm2)  // Updating Kinetic Equation
                                                       // in datagridview
            : base(frm2)
        {
            int currentRow = frm2.dbKineticName.CurrentCell.RowIndex;
            string data, label;
            data = frm2.dbKineticName.Rows[currentRow].Cells[1].ToString();
            label = frm2.dbKineticName.Rows[currentRow].Cells[0].ToString();
            conn.Open();
            try
            {
                command.CommandText = "UPDATE " + database +
                                      ".EnzymeKinetics " +
                                      "SET Equations = " +
                                      "'" + data + "'" +
                                      "WHERE Kinetic_Name = " +
                                      "'" + label + "'";
                command.ExecuteNonQuery();

                conn.Close();
                bind.BindingKineticEquations(frm2);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}
```

Open Database

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace DynamicSimulator_v2
{
    class OpenDatabase
    {
        private static string Database;

        public void setupOpenDatabase(string database)
        {
            Database = database;
            Database = getName();
        }

        public string getName()
        {
            return Database;
        }
    }
}
```

Update Binding

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using MySql.Data;
using MySql.Data.MySqlClient;
using System.Windows.Forms;

namespace DynamicSimulator_v2
{
    class UpdateDataGrid
    {
        #region Local Variable
        protected string connStr = "";
        protected string database = "";
        #endregion

        #region Object Declaration
        protected MySqlConnection conn;           // connection for database
        protected SchemaName schemaForm;          // calling database name
        protected MySqlCommand command;           // Query SQL
        #endregion

        #region Constructor
```

```
        public UpdateDataGrid(frmMetCon frm2)
        {
            schemaForm = new SchemaName();
            connStr = "datasource=localhost;port=3306;username=root;password=root;";
            database = schemaForm.getData;
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
        }
        #endregion

    }
    class UpdateBindingKineticEquations : UpdateDataGrid
    {
        public UpdateBindingKineticEquations(frmMetCon frm2)
            : base(frm2)
        {
            int rowIndex = frm2.dbKineticName.CurrentCell.RowIndex;
            int columnIndex = frm2.dbKineticName.CurrentCell.ColumnIndex;

            try
            {
                string kineticName =
frm2.dbKineticName.Rows[rowIndex].Cells[0].Value.ToString();
                string kineticEq =
frm2.dbKineticName.Rows[rowIndex].Cells[columnIndex].Value.ToString();
                conn.Open();
                command.CommandText = "UPDATE " + database +
                                      ".EnzymeKinetics " +
                                      "SET Equations = " +
                                      "'" + kineticEq + "'" +
                                      "WHERE Kinetic_Name = " +
                                      "'" + kineticName + "'";
                command.ExecuteNonQuery();
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
    class UpdateBindingKineticParameter : UpdateDataGrid
    {
        public UpdateBindingKineticParameter(frmMetCon frm2)
            : base(frm2)
        {
            int rowIndex = frm2.dbParameter.CurrentCell.RowIndex;
            int columnIndex = frm2.dbParameter.CurrentCell.ColumnIndex;

            try
            {
                string ParameterName =
frm2.dbParameter.Rows[rowIndex].Cells[0].Value.ToString();
                double ParameterValue =
double.Parse(frm2.dbParameter.Rows[rowIndex].Cells[columnIndex].Value.ToString());
                conn.Open();
                command.CommandText = "UPDATE " + database +
                                      ".KineticParameters " +
                                      "SET Parameter_Value = " +
                                      "'" + ParameterValue + "'" +
```

```
                                    "WHERE Parameter_Name = " +
                                    "'" + ParameterName + "'";
                command.ExecuteNonQuery();
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}
```

Tokenize

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using MySql.Data;
using MySql.Data.MySqlClient;
using System.Data;

namespace DynamicSimulator_v2
{
    class ParentToken
    {
        #region Local Variables
        protected string Equation = "";
        protected string database = "";
        protected string connStr = "";
        protected string[] equations;
        #endregion

        #region Object Declaration
        protected MySqlConnection conn;
        protected SchemaName schemaForm;
        protected MySqlCommand command;
        #endregion

        public ParentToken(string equation)
        {
            schemaForm = new SchemaName();
            connStr = "datasource=localhost;port=3306;username=root;password=root;";
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
            database = schemaForm.getData;

            Equation = equation.Trim();
            char[] delimiters = new char[] { '+', '-', '/', '*', '(', ')', '[', ']',
                                            '^'};

            if (String.IsNullOrEmpty(Equation) == true) return;
```

```
            equations = Equation.Split(delimiters)
                                .Distinct()
                                .Where(x => x != string.Empty).ToArray();
        }
    }

    class KineticEquation : ParentToken
    {
        public KineticEquation(string eq)
            : base(eq)
        {
            try
            {
                conn.Open();

                command.CommandText = "TRUNCATE TABLE " + database +
                                      ".EnzymeKinetics";
                command.ExecuteNonQuery();

                for (int i = 0; i < equations.Length; i++)
                {
                    command.CommandText = "INSERT INTO " + database +
                                          ".EnzymeKinetics " +
                                          "(KineticID)" +
                                          "VALUES " +
                                          "('kinetic" + i + "')";
                    command.ExecuteNonQuery();

                    command.CommandText = "UPDATE " + database +
                                          ".EnzymeKinetics " +
                                          "SET Kinetic_Name = " +
                                          "'" + equations[i] + "'" +
                                          "WHERE KineticID = " +
                                          "'kinetic" + i + "'";
                    command.ExecuteNonQuery();
                }
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }

    class KineticParameter : ParentToken
    {
        public KineticParameter(string eq)
            : base(eq)
        {
            try
            {
                conn.Open();
                command.CommandText = "TRUNCATE TABLE " + database +
                                      ".KineticParameters";
                command.ExecuteNonQuery();
                for (int i = 0; i <equations.Length; i++)
                {
                    command.CommandText = "INSERT INTO " + database +
                                          ".KineticParameters " +
```

```
                                        "(ParameterID)" +
                                        "VALUES " +
                                        "('parameter" + i + "')";
                    command.ExecuteNonQuery();

                    command.CommandText = "UPDATE " + database +
                                        ".KineticParameters " +
                                        "SET Parameter_Name = " +
                                        "'" + equations[i] + "'" +
                                        "WHERE ParameterID = " +
                                        "'parameter" + i + "'";
                    command.ExecuteNonQuery();
                }
                conn.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }
    }
}
```

Tokenize Parameters

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
using System.Data;

namespace DynamicSimulator_v2
{
    class ParserKineticParameter
    {
        #region Object Declaration
        SchemaName schemaForm;
        MySqlConnection conn;
        MySqlCommand command;
        DataTable dtable;
        DataTable dtable2;
        ParentToken parser;
        #endregion

        #region Local Variables
        private string connStr = "";
        private string database = "";
        private string combine = "";
        //private string combine2 = "";
        #endregion

        public void ParserParameter()
        {
            schemaForm = new SchemaName();
```

```
            connStr = "datasource=localhost;port=3306;username=root;password=root;";
            conn = new MySqlConnection(connStr);
            command = conn.CreateCommand();
            database = schemaForm.getData;
            dtable2 = new DataTable();
            dtable = new DataTable();

            conn.Open();
            command.CommandText = "SELECT Equations FROM " + database +
                                  ".EnzymeKinetics " +
                                  "WHERE CONCAT ('',Kinetic_Name*1)<> Kinetic_Name";
            MySqlDataReader dr = command.ExecuteReader
                                 (CommandBehavior.CloseConnection);
            dtable.Load(dr);
            for (int i = 0; i < dtable.Rows.Count; i++)
            {
                combine += "+" + dtable.Rows[i]["Equations"].ToString();
            }
            conn.Close();

            conn.Open();
            command.CommandText = "SELECT Parameter_Name FROM " + database +
                                     ".KineticParameters " +
                                     "WHERE CONCAT ('',Parameter_Name*1)<>
                                      Parameter_Name";
            MySqlDataReader dr2 = command.ExecuteReader
                                  (CommandBehavior.CloseConnection);
            dtable2.Load(dr2);
            for (int i = 0; i < dtable2.Rows.Count; i++)
            {
                combine += combine + "+" + dtable2.Rows[i]
                           ["Parameter_Name"].ToString();
            }
            conn.Close();
                parser = new KineticParameter(combine.Trim());
        }
    }
}
```

Tokenize and Rearrange

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using MySql.Data;
using MySql.Data.MySqlClient;
using System.Data;
using System.Windows.Forms;

namespace DynamicSimulator_v2
{
    class Tokenizer
    {
        #region Variables
        private string Equation = "";
```

```
        private string[] equations;
        private string Tokencombine = ""; // for kinetic parameters
        private string combines = ""; // for kinetic equations
        #endregion

        #region Object Declaration
        private ParentToken parentToken;

        #endregion

        #region Public Method
        public void TextTokenizer(string equation)
        {
            Equation = equation.Trim();
            char[] delimiters = new char[] { '+', '-', '/', '*', '(', ')', '[', ']',
                                             '^' };

            if (String.IsNullOrEmpty(Equation) == true) return;
            equations = Equation.Split(delimiters)
                                .Distinct()
                                .Where(x => x != string.Empty).ToArray();

            analyzeEquations(equations);
        }

        #endregion

        #region Private Method
        private void analyzeEquations(string[] eq)
        {
            int charIndex = 0;
            for (int i = 0; i < eq.Length; i++)
            {
                char c = eq[i][charIndex];
                if (c != 'v') // ensure that the chunk string is for parameter
                {
                    Tokencombine += "+" + eq[i].ToString(); // combine after rearrange
                                                               for kinetic parameters
                }
                else
                {
                    combines += "+" + eq[i].ToString(); // combine after rearrange for
                                                           kinetic equations
                }
            }
            parentToken = new KineticEquation(combines);  // Token the kinetic
                                                             equations
            parentToken = new KineticParameter(Tokencombine); // Token the kinetic
                                                                 parameters
        }
        #endregion
    }
}
```