DEVELOPMENT OF A COMPUTATIONAL MODEL FOR BIOFILM BASED MICROBIAL FUEL CELL

N.M.N.C. Marapana

(119262 J)

Degree of Master of Science

Department of Chemical and Process Engineering

University of Moratuwa

Sri Lanka

December 2015

DEVELOPMENT OF A COMPUTATIONAL MODEL FOR BIOFILM BASED MICROBIAL FUEL CELL

Narangahawatta Mudiyanselage Nilanthi Chandani Marapana

(119262 J)

Thesis submitted in partial fulfillment of the requirements for the degree Master of Science

Department of Chemical and Process Engineering

University of Moratuwa

Sri Lanka

December 2015

"I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books)."

Signature: Date:

The above candidate has carried out research for the Masters/MPhil/PhD thesis/Dissertation under my supervision.

Signature of the supervisor: Date:

Abstract

Studies on Microbial Fuel Cells (MFC) as Power production units are of increasing interest, because it can convert a variety of bio-degradable organic compounds into electricity. In a MFC; biological, chemical and electro-chemical reactions take place resulting in a change of concentration of substrate, suspended solids and growth of a biofilm leading to a production of an electrical current. In this study a dynamic mathematical model is developed that represents the behavior of microbial fuel cell using set of derived equations those describe the consumption of substrate by microorganisms, production of oxidized mediators using reduced mediators, growth of microorganisms in the bulk liquid and the biofilm attached to the anode and production of current at the electrode surface. The system consists of a bulk liquid with suspended cells and the anode with an attached biofilm.

Performance of a MFC is evaluated by analyzing the variation of production of current with time, variation of concentration of components (microorganisms, substrate, oxidized mediator and reduced mediator) in the bulk liquid with time and variation of concentration of mediators at the electrode surface with time in various combinations of selected operating parameters (reaction rate, exchange current density and total cell resistance). It was found that, higher the reaction rate the production of current by the fuel cell is high. At the same time, reaching of maximum current production is rapid in the systems simulated with high reaction rates compared to that of the others. On the other hand, high exchange current density values give relatively low current production from the cell where the low exchange current densities give somewhat high current production. Variation of total cell resistance affects in a similar manner on current production. That is, when the cell is simulated with high cell resistance values, the production of current is low. But, the current production sustain for a rather long period.

Key words: Microbial Fuel Cell, biofilm, dynamic modeling

I like to dedicate this work to all who were with me during past time and gave me their helping hands and kind words always.

Acknowledgement

I would like to express my deepest gratitude to my advisor, Dr. P.G. Ratnasiri, head of the Department of Chemical & Process Engineering, Faculty of Engineering, University of Mratuwa for his excellent guidance, caring, patience on my research work, in spite of his tight schedule.

I must express my gratitude to Prof. R Shanthinie, Senior Lecturer, Department of Chemical and Process Engineering, Faculty of Engineering, University of Peradeniya, who let me experience this area of research and allowing me to go through this interested field, where I have not been before.

I would like to thank Mr. Anushka Perera, who gave me necessary guidance in solving the equations also Mr. Chathura Wijerathne who assisted me in solving the model.

I would like to thank my family members, my beloved parents and two sisters in helping me various ways to finish the task.

Finally, I want to thank my friends who supported me in different aspects to complete this task.

Contents

List of Figures

List of Tables

1. INTRODUCTION

1.1. Introduction to fuel cells

A fuel cell consists of a negatively charged electrode (anode), a positively charged electrode (cathode) and an electrolyte. Hydrogen is oxidized on the anode and oxygen is reduced on the cathode. The produced ions from the half reaction at the anode are transported to the cathode through the electrolyte and the resultant electrons are carried out to the cathode via an external circuit.

1.1.1. Classification of fuel cells

Fuel cells are classified based on electrolyte used and this is the widely used categorization method. Under this classification, fuel cells can be identified under six primary classes, and they are shown in Figure 1.1.

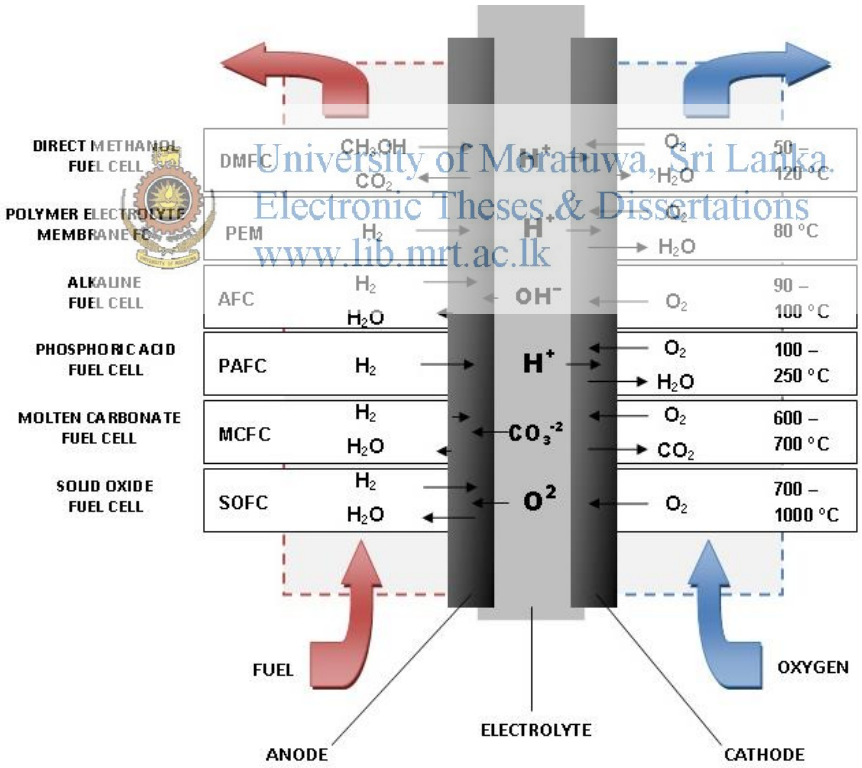


Figure 1.1 Different types of fuel cells
(http://www.fuelcells.org/base.cgim?template=types_of_fuel_cells)

The primary categories are,

- Polymer Electrolyte Membrane Fuel Cell (PEMFC)
- Direct Methanol Fuel Cell (DMFC)
- Alkaline Fuel Cell (AFC)
- Phosphoric Acid Fuel Cell (PAFC)
- Molten Carbonate Fuel Cell (MCFC)
- Solid Oxide Fuel Cell (SOFC)

In both polymer electrolyte membrane fuel cell and direct methanol fuel cell, a membrane is used where the ions can transfer through the membrane. In other fuel cell types, alkaline, phosphoric acid, molten carbonate, the used electrolytes are the solutions indicated by the names of fuel cells accordingly. In solid oxide fuel cell stabilized zirconia is used as the electrolyte. The transferring ion through the electrolyte is differ in some cases and they are also shown in Figure 1.1.

According to the above classification, microbial fuel cells can be placed under category of polymer electrolyte membrane. Because microbial fuel cell consists of a proton exchange membrane that separates anolyte and catholyte within the cell in to two compartments. Further hydrogen is oxidized on the anode and oxygen is reduced on the cathode and the membrane allows passing only proton through it. Hence the half reactions occurring at either electrode are similar to that of proton exchange membrane.

Investigation of the first fuel cell was carried in 1839 by William Grove (Spiegel, 2008). Although less work was carried out over the fuel cells during 1800s, from 1960s onwards the attention on this sector had increased.

1.1.2. Applications of fuel cells

Novel application of fuel cell is the treatment of waste water using microorganisms and current generated is used to partially fulfil the energy requirement of waste water treatment plant (WWTP). This is currently being practiced in lab scale using different types substrates and microorganisms using different fuel cell configurations.

Future market of fuel cells is its capability of being used as portable batteries for portable devices such as laptops, video recorders, cell phones and iPod, etc. Not only in portable devices but also in transportation sector and as stationary power sources. Some stationary fuel cell units generate power that is enough for application in household or small scale business.

1.2. Introduction to MFC

Microbial Fuel Cells (MFCs) are identical to fuel cells, besides they generate electricity from biodegradable organic material using metabolic activities of micro-organisms.

Chemical fuel cells has following limitations,

- The limited viability and high cost of catalysts
- The highly corrosive electrolytes
- The elevated operating temperatures

These problems can be overcome by application of microbial fuel cells, which use microorganisms as active catalytic components (Oh & Tack, 2010).

In a microbial fuel cell, organic matter (substrate) is oxidized by microorganisms while producing electrons, and the electrons travel through a series of enzymes inside the cell and make energy in the form of adenosine triphosphate (ATP) for the cell. In order to transform these electrons to a terminal electron acceptor (TEA), several mechanisms have been identified assuming that they are being used in the systems (Direct Electron Transfer (DET) and Mediated Electron Transfer (MET)). (Schroder & Uwe, 2007)

1.2.1. Direct Electron Transfer (DET)

There are no diffusional redox species involving in this method.

Physical contact of the bacterial cell with the fuel cell anode should be present (Figure 1.2).

The method requires that the microorganisms possess membrane bound electron transport protein relays that transfer electrons from the inside of the bacterial cell to its outside, terminating in an outer membrane redox protein that allows the electron transfer to an external solid electron acceptor (fuel cell anode) (Schroder & Uwe, 2007).

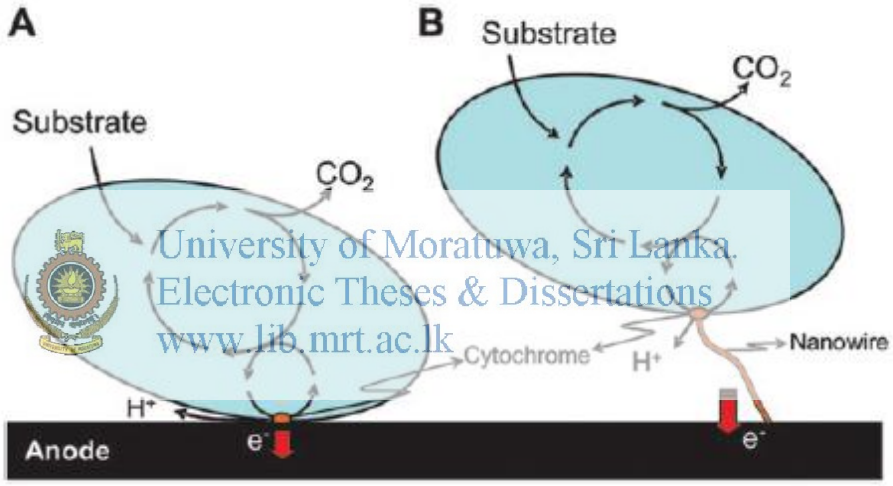


Figure 1.2 Illustration of the DET via (A) membrane bound cytochromes, (B) electronically conducting nanowires (Schroder & Uwe, 2007)

1.2.2. Mediated Electron Transfer (MET)

In case of thick biofilms where DET is impossible, microorganisms may use externally available (exogenous) electron shuttling compounds like humic acids or metal chelates or produce low molecular electron shuttling compounds via secondary metabolic pathways by microorganisms themselves.

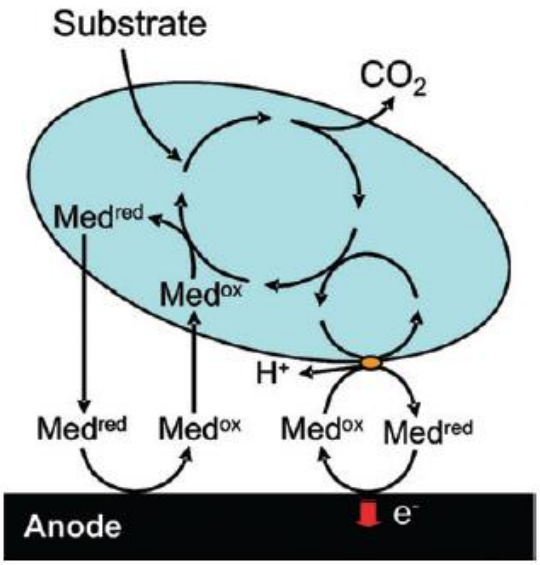


Figure 1.3 MET via microbial secondary metabolites. Two possible redox mechanisms have been proposed: shuttling via outer cell membrane cytochromes and via periplasmatic or cytoplasmatic redox couples (Schroder & Uwe, 2007)

In this method, the electron transfer independent of the presence of exogenous redox shuttles (Figure 1.3). The mediator serves as a reversible terminal electron acceptor, transferring electrons from the bacterial cell either to a solid oxidant (MFC anode) or into aerobic layers of the biofilm, where it becomes reoxidized and is again available for subsequent redox processes (Schroder & Uwe, 2007).

Unlike in fuel cells, microbial fuel cells require electron carriers from inside of the cell to the outside and to the terminal electron acceptor that is electrode – anode. Electron shuttles or artificial mediators are used to accomplish this task.

The first observation of microbial fuel cell was made in 1911 by Potter (Logan & Bruce, 2008). But until 1990s, the growth of this field remained in a somewhat stagnant condition and later it became an interesting topic among scientists and engineers who did studies and researches in relation to the sustainable energy sector.

Although MFC is a promising option for sustainable energy generation in the future, it is good to have other sustainable energy alternatives as well. Further, this technology has many challenges to overcome. The extractable power of an MFC is

affected by the difference in the potentials of the oxidizer and fuel compounds, irreversible losses due to kinetic limitations of the electron transfer processes at the electrode interfaces, ohmic resistances and concentration gradients, the electrode sizes and transport rates across the membrane separating the MFC compartments. (EG&G Technical Services, Inc.2004)

Rate limiting steps are the most important factors to be modified so that performance of MFC could be improved.

1.3. Applications of MFC

Microbial fuel cell technology is used to power several applications, the first such application was reported in 2008 (Frank & Ashley, 2010). The devices capable of measuring air temperature, pressure, relative humidity, water temperature and transferring data were powered by microbial fuel cells.

The use of anode as a terminal electron acceptor by microorganisms opens varieties of applications. Even though many applications have been identified, they are not currently feasible and need more improvements. The applications are still in laboratory scale and researches are conducted to improve these and to make them viable technologies.

The limited applications of MFC technology is due to low power output of the cells, therefore an understanding of microbiology of the current producing process is required to improve and increase the power output. The main problems to be considered are accumulation of proton within the biofilm and over potential at the cathode (Frank & Ashley, 2010).

In certain applications of MFC, current production is not the major objective. Treating of wastewater or in bioremediation usage of cathode and anode in a cell is much more promising than the electrical production by MFC (Frank & Ashley, 2010)

One of the most active area of microbial fuel cell (MFC) research is the production of power from wastewater while it is being treated. Microbial fuel cells can be used to produce power from range of substrates. Since these substrates completely brakes

down into carbon dioxide in MFC systems, Wide range of compounds such as acetate, glucose, starch, cellulose, wheat straw, pyridine, phenol, p-nitrophenol, complex solution like domestic wastewater, brewery waste, leachate, etc. can be used as substrates (Frank & Ashley, 2010)

Power implanted medical devices using glucose and oxygen from blood is another exceptional application. An implanted MFC could provide power indefinitely and negate the need for surgery to replace batteries (Frank & Ashley, 2010).

Sediment MFC are already powering low-energy devices in the marine environments which are difficult to access. This is the most promising version of this technology to be applied for power generation from electrons stored in the sediments. It is feasible in marine environments such as fish farms, natural reserves, harbors and isolated communities. Sediment MFC could also provide small amounts of power for lighting or monitoring devices. To meet higher demand, the sediment MFC could be replenished through the addition of chitin or other organics into the sediments, but such a process has to be balanced to possible environmental consequences .However, such technology could easily provide remote monitoring devices in a wide range of salt and freshwater systems. If the sediment MFC were to be enclosed and adapted, it is conceivable that electrons could be harvested as part of the composting waste organic or vegetable matter to power electronics in remote locations or third world countries. In fact, the World Bank has provided funding to start trials of MFCs that run on waste and provide electricity for lights and to charge batteries in rural areas of Tanzania and Namibia (Frank & Ashley, 2010).

MFCs can also be modified to produce hydrogen gas by removing oxygen at the cathode and adding in a small voltage via the bioelectrochemically assisted microbial reactor (BEAMR) process or the biocatalyzed electrolysis process (Logan, et al., 2006).

1.4. Modelling of MFC

Mathematical modelling of systems and validation of the simulated results with experimental data also are conducted in order to maximize the performances of systems and to optimize their operating conditions.

1.4.1. Parameters for MFC modelling

It is difficult to define a single parameter in order to analyze the efficiency of microbial fuel cells.

Coulombic Efficiency has been identified as one parameter to quantify the efficiencies of microbial fuel cells. For Microbial Fuel Cells, Coulombic Efficiency is defined as the fraction of electrons extracted for conversion in to electricity to that in the starting organic material (Devasahayam, 2012).

There are some other electrical, chemical and biological parameters used in analyzing the performances of MFCs. Current produced or variation of current density over the electrode and electrical charge are some commonly used electrical parameters. The electrical parameters are listed in Table 1.1.

Table 1.1 List of Electrical Parameters related to Microbial Fuel Cells

Symbol	Definition	Unit
η	Coulombic Efficiency	-
i	Current Density	A/m^2
I	Current Production	A
$\boldsymbol{V_{cell}}$	Cell Voltage	V
V	Voltage generated via external load	V
$\boldsymbol{\eta_{A,act}}$	Activation over potential at the anode	V
$\boldsymbol{\eta_{C,act}}$	Activation over potential at the cathode	V
P	Power	W

Depending on the objectives of the model to be developed, the variation of concentration of fuel/substrate, microorganisms and mediators (oxidized and reduced) in the bulk liquid with time and in the biofilm with time and spatial distribution are considered. Further concentrations of mediators at the electrode surface are also considered. Two different scenarios of reactor feeding i.e. batch reactor feeding and continuous feeding condition are also considered in developing models.

1.5. Validation of a model using experimental results

Experimental results are used to validate the developed mathematical models by changing used and assumed constants, and still if the model doesn't match with the observed data, the made assumption have to be reduced or modified and afterwards equations have to be modified accordingly. These steps should be continuing until the simulated results give same trends with the observed readings. Parameter estimation can be performed by changing model parameters until the simulated results give a successful best fit with the experimentally observed results. After achieving the required accuracy it is said that the model is calibrated and can be used to simulate other cases under given conditions with the assumptions made.

2. LITERATURE REVIEW

When modelling of a microbial fuel cell, two compartments are considered i.e. biofilm and fuel cell. Therefore, in this chapter, the attention is paid on past and recent research works that, have been conducted based on mathematical modelling of biofilms, fuel cells and finally microbial fuel cells.

2.1. Mathematical Models for Biofilms

Mathematical models available to describe behavior of biofilms are two folds ie simple models based on basic concepts and complex models.

Rittmann et. al. (1980) developed basic model for biofilm; assuming steady state biofilm conditions. The steady state biofilm is defined as one that has neither net growth nor decay over time. The model has been developed for steady state biofilm kinetics with a single substrate. The flux of substrate into the biofilm was coupled with the mass or thickness of the biofilm, that would exist at a steady state conditions for a given bulk substrate concentration.

This model considers kinetic and energy constrains and predicts the minimum bulk substrate concentration in order to have a steady state biofilm. Under two conditions occours simultaneously, there is no biofilm thickness. The two conditions are, concentration of substrate in the bulk liquid is below the required minimum and adsorption of bacteria from the bulk liquid does not exist.

For substrate concentrations greater than the minimum required value in the bulk liquid, the model can be used to calculate steady state substrate flux and biofilm thickness. (Rittmann & McCarty, 1980)

Wanner et al (2004) studied on mathematical models for biofilms and their capabilities. They implemented a model using. AQUASIM dynamic simulator for the identification and simulation of aquatic systems. This program includes a one dimensional multispecies and multisubstrate biofilm modelling. Using the model, one dimensional spatial profile of substrate and microbial species can be predicted. Development of the biofilm thickness, substrate and microbial species within the

biofilm and in the bulk liquid over time can be determined using this computer program. Detachment and attachment of microbial cells and sloughing events also can be simulated with this. (Wanner & Morgenroth, 2004)

Xavier et al (2005) developed a mathematical model for biofilms using an improved version of individual based modelling (IbM) that allows structured biofilm. In this approach biomass composition may be separated into any number of particulate species, including extracellular polymeric substances (EPS). For this polymeric substance, specific functionality has been included. Detachment of biofilm was also considered, and it is described as occurring at the biofilm surface with variable local rates derived from functions of state variables.

A two species biofilm has been studied with the model where a competition exists between an organism capable of accumulating polyhydroxybutyrate (PHB, an internal storage compound) and an EPS producing organism. The considered two microbial species are heterotopic and the competition between them was analyzed in a hypothetical system under two feeding conditions; continuous feeding conditions and feast/famine feeding condition. The total amount of substrate fed to the reactor per day was kept equal for both feeding conditions.

Variations of concentration of components (heterotrophic PHB, heterotrophic EPS, PHB, EPS and inert biomass) within the biofilm with time were simulated for above mentioned two feeding conditions. Growth of the biofilm with time was observed using the three dimensional model.

The model results illustrate that biofilm enriched in PHB producing organisms can be obtained by supplying substrate intermittently in feast/famine cycles. (Xavier, et al., 2005) .

2.2. Mathematical Models for Microbial Fuel Cells

Mathematical models have been developed based on various aspects and recently developed models are discussed here.

Mathematical models described below had been developed focusing on bio-catalytic (based on microorganisms) activities, which are substrate oxidation and its product that its electron transport from catalytic microorganism to electrode.

A model for microbial fuel cells with two-chambers was developed by Zeng et. al. (2010). In developing the model the similarities between a typical microbial fuel cell with chemical fuel cell were considered, such as direct ascorbic acid fuel cells and direct methanol fuel cells. By integrating biochemical reactions, Butler-Volmer expressions and mass/charge balances, the model was developed in order to simulate both steady state and dynamic behaviour of a microbial fuel cell, including voltage, power density, fuel concentration and the influence of various parameters on power generation.

In the model it was assumed that both the anode and cathode compartments are continues stirred tank reactors (CSTR), all mass transport processes were assumed to be so fast compared with the biochemical reactions and hence the concentrations of all reactants in the bulk solution can be considered to be equal to those in the electrode surface.

The model gave equal priority for anode and cathode chambers, while most of the other biofilm based models considers the variations inside the anodic chamber only.

The simulated results were compared with experimental data, where the experiments were conducted using acetate as fuel and also with artificial wastewater (solution of glucose and glutamic acid).

Six model parameters: forward rete constant of anode reaction at standard condition (maximum specific growth rate) and forward rate constant of cathode reaction at standard condition, half velocity rate constant for acetate and dissolved oxygen, charge transfer coefficient of anode and cathode) were further studied and estimated

using the experimental data for fuel cells fed with acetate and artificial wastewater separately (Zeng, et al., 2010).

Pinto et. al. (2010) developed a bio-electrochemical model of a microbial fuel cell which consists of two-population of microorganisms. The model describes the competition between two microbial populations; anodophilic and methanogenic for a common substrate in a microbial fuel cell.

A system of ordinary differential equations was used to describe biomass growth and retention in the anodic compartment. Because of these ordinary differential equations, the model was capable to give fast numerical solutions.

In developing the model, the entire system was considered as two main parts; anodic compartment and intracellular sections of cells. Further, kinetics equations and electrochemical equations were introduced to the model.

Several assumptions were also made to avoid extra complexity of the model, they are described below; carbon source is well distributed in the anodic compartment and substrate gradient in the biofilm is neglected, uniform distribution of microbial populations in the anodic compartment biofilm, constant pool of intracellular electron transfer mediator in a microorganism and temperature and pH are considered fully controlled and kept constant.

The parameters of the model were estimated and validated using experimental results that were obtained using four continuous-flow air-cathode microbial fuel cells operated at various external resistances and organic loads. The model analysis demonstrated the influence of organic load and external load resistance on power output of microbial fuel cell and its long term performance (Pinto, et al., 2010).

2.3. Mathematical Models for biofilm-based Microbial Fuel Cells

Picioreanu et. al. (2007) developed a computational model for biofilm-based microbial fuel cells (MFCs) based on redox mediators with several populations of suspended and attached biofilm microorganisms and multiple dissolved chemical species. With this model, important MFC parameters; current, charge, voltage and power production, consumption of substrates, suspended and attached biomass growth can be simulated under several operational conditions. The above mentioned profiles could also be obtained as variations with time.

Effect of different substrate utilization yields, standard potential of the redox mediator, ratio of suspended to biofilm cells, initial substrate and mediator concentrations, mediator diffusivity, mass transfer boundary layer thickness, external load resistance, endogenous metabolism, repeated substrate additions and completion between different microbial groups in the biofilm can be analyzed with the model.

Seventeen major simulation cases have been done where case number one was the standard case and the other cases were different from the standard case with variations of concentration of initial components, diffusivity coefficients, yield factors, reaction rate constants and biofilm thickness.

The model has been developed in order to carry out two-dimensional and three-dimensional heterogeneous current distribution over the planner anode surface for younger and patchy biofilms.

With the model, it has been realized that one-dimensional model gives sufficiently accurate description of produced current for uniformly flat biofilms.

Voltage-current and power-current characteristics also can be calculated at different moments in time to evaluate a limiting regime in which the microbial fuel cell operates with the developed model.

The predicted results with the model were evaluated using experimental data obtained in a batch microbial fuel cell with a geobacter biofilm fed with acetate (Picioreanu, et al., 2007).

Picioreanu et. al. (2008) developed a mathematical model for microbial fuel cells with anodic biofilms under anaerobic digestion. The considered microbial population consists of methanogenic and electroactive microorganisms and they coexist suspended in the anolyte and in the biofilm attached to the anode. The model was developed based on previously introduced model by Cristian Picioreanu et. al. in 2007. Further, the new model aimed at representing an anaerobic anode system involving a microbial community taken from an anaerobic wastewater treatment process and the biochemical model is based on the IWA's ADM1 (Anaerobic Digestion Model No.1 (Batstone, et al., 2002)).

The model outputs are evolution in time of current production, consumption of substrates, suspended and attached biomass growth. In order to find out the limiting regimes in which microbial fuel cells operates current, voltage and power characteristics can be calculated with the model. The simulated results of the model were compared with the experimental data of a batch fed microbial fuel cell.

The batch fed microbial fuel cell operated with smaller electrical resistance of the circuit, hence with electroactive bacteria in the reaction chamber. Therefore, electrons from substrate directly transfer to the anode rather than following a methanogeneses path, leading to higher coulombic yield. Further this result, higher current and faster COD (Chemical Oxygen Demand) consumption rates.

The effect of external resistance on electroactive bacteria (EAB) was also studied with the developed model.

The model can be used for further comprehensive studies of microbial fuel cells fed with wastewater. (Picioreanu, et al., 2008)

Picioreanu et. al. (2009) developed a mathematical model for modelling microbial fuel cells with suspended cells and added electron transfer mediator. The model is based on mass balances for several chemical species; substrate oxidized mediator and reduced mediator.

The model has been developed assuming that no biofilm on the anode, the biofilm was replaced with a boundary layer. The rates of biological reactions were expressed as double Monod limitations on substrate and oxidized mediator.

Model outputs are production of current and electrical charge with time and current-voltage and current-power curves. The variation of concentrations of components with time was evaluated as well the model behaviour was illustrated using a test case based on detailed experimental observations done for a microbial fuel cell operated in batch mode and repeatedly fed with a single substrate. The model was simulated using glucose as the anode substrate and diffusible electron transfer mediators. With the simulation results current-time and voltage-current curves were observed.

Effect of different parameters; (electrical resistance, mass transfer resistance, exchange current, coulombic yields and biomass, substrate and mediator concentrations on the performance of microbial fuel cell were analyzed with the developed model. (Picioreanu, et al., 2009)

Mathematical models for biofilm are available as 1-dimensional models, 2 and 3 dimensional models. Still 1dimensional models are capable of give precision predictions on experimental observations at the same time the models are not much complex as those of multi-dimensional models. Therefore it is sufficient to consider 1 dimensional variation of components in case of biofilm modelling with a fuel cell.

Models of microbial fuel cells are also exist, they are useful in studying the behaviour of microbial fuel cells in generation of current, growth of microorganisms and consumption of substrate.

Biofilm based microbial fuel cells are the ones focused in this study particularly. The considered models have a growing biofilm or a concentration boundary layer on the anode electrode. In developing the models with biofilm on the anode the growth of the biofilm was considered in multi-dimensional aspect and hence the models have become unnecessarily complex. One simple model is available where 1 dimensional variation of concentrations of components are considered, but still this model does not consider a biofilm instead a concentration boundary layer.

Therefore it was identified that developing a 1 dimensional simple model for microbial fuel cells which has a growing biofilm on the anode for further studies is essential. Where the simulated results can be obtained without using long time on high power computers.

In addition to above facts, none of the considered models have provided variation of concentration of mediators at the anode surface, even though it is a significant factor in current production of the cell. Thus new model should have this ability also. The model needs to have capability to analyze different model parameters, such as reaction rates, total cell resistances as the already existing models.

In order to achieve above objectives, this study focuses on developing a computational model for biofilm based microbial fuel cells.

3. METHODOLOGY

3.1. Model Description

When dissolved chemical species are oxidized on the anode and reduced at the cathode electrical current generates, mathematical model of a microbial fuel cell should be able to analyze the electrical current and voltage generated. Therefore the model needs to describe the electrical and chemical reactions on the electrode surfaces. This approach will be described under section 3.2.3.

The considered system is a microbial fuel cell with attached biofilm on the anode and with suspended microbial cells in the bulk liquid, operating under the batch mode. Further an artificial electron transfer mediator has been added to the system.

Two main compartments are assumed to exist in the system; they are bulk liquid and biofilm. Even though in nature, biofilm has a boundary layer, in this model it is assumed that bulk liquid and biofilm have direct contact so that no intermediate layers exist. Modelling of four components are focused, they are fuel/substrate, microorganism and reduced and oxidized mediators.

The mathematical model that is going to present here is a modified version of an existing model. The original model was developed by Cristian Picioreanu and team in 2009.

The original model describes the behaviour of a fuel cell, the cell – anode has a concentration boundary layer on it. With the new model the boundary layer has been replaced with a biofilm and a thin boundary layer, where the thickness of the layer is negligible.

3.2. Model Development

- Initial current density value was assumed as 3×10^{-8} A/m^2.

- The thickness of the concentration boundary layer in between the biofilm and bulk liquid was assumed as very thin and further inside this layer all the considered components behave as they are in the biofilm.

- It was assumed initial thickness of the biofilm as 1×10^{-6} m.

- Thionine was considered as the mediator and single substrate was taken as the carbon source for microorganisms and one heterotrophic type of microorganism exist in the system.

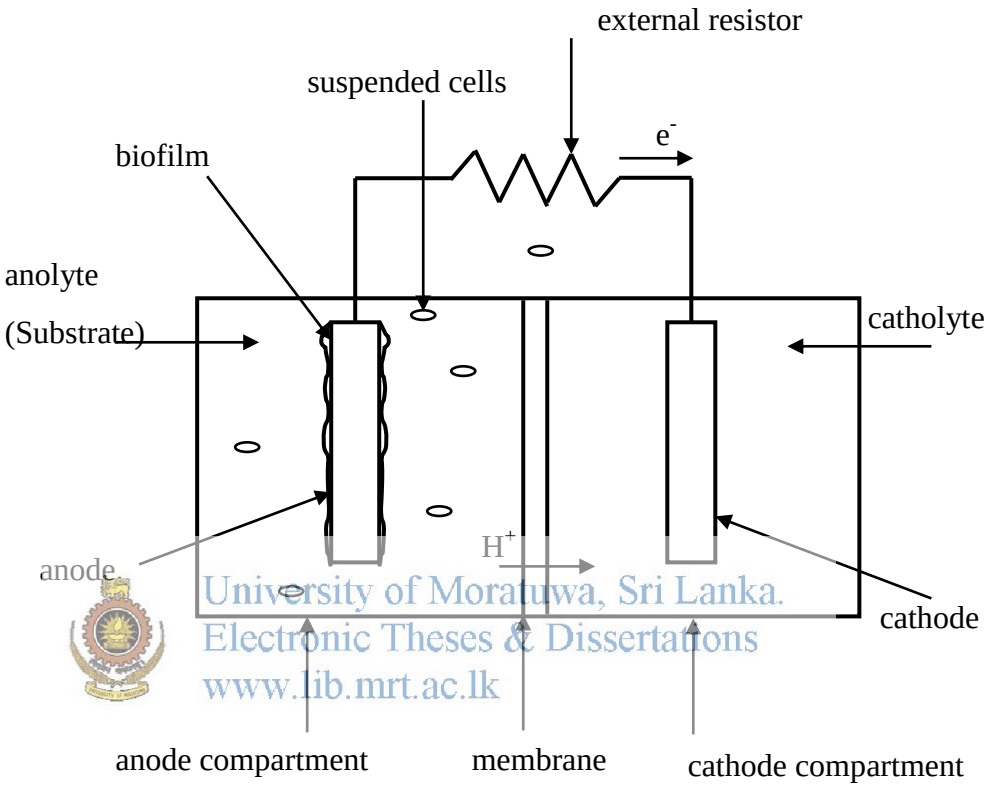


Figure 3.1 Schematic diagram of microbial fuel cell

- The variation of concentration of components other than the vertical direction inside the biofilm was assumed as not significant.

- The bulk liquid was assumed as a well mixed solution with suspended cells hence the solution is homogeneous.

- It was assumed that, 1 mol of substrate is equivalent to 100 g COD and 1 mol of microorganism is equivalent to 200 g COD.

- The container initially filled with substrate with the given initial concentration and the reactor behaves as a batch reactor, no more substrate or any other component are added to the system.
- It was assumed that the connection of external circuit is completed when the components in the biofilm are achieved the given initial concentrations in the Table 3.1.
- The attachment of microorganism to the biofilm and detachment from the biofilm was assumed as approximately same.

3.2.1. Bulk Liquid

In bulk liquid, it was assumed to be well mixed and hence homogeneous and no spatial distribution of concentrations but varies with time only. Therefore with the above mentioned assumptions, a set of ordinary differential equations can be used to define the system.

The bulk liquid has interactions with the biofilm and the liquid affects due to the electrical reactions at the electrodes as well. A set of derived equations extracted from a previous model with necessary simplifications were used to model the bulk liquid.

3.2.1.1. Biomass Component

A common mass balance for all biomass types can be written as in Eq. (01) (Picioreanu, et al., 2007) , this model considers a single type of microorganism.

$$\frac{dC_{X,L}}{dt} = r_{X,L} + r_{det}\frac{A_E}{V_L} - r_{ata}\frac{A_E}{V_L} \qquad \text{Eq. (01)}$$

$C_{X,L} = C_{X,0}$ (Initial Condition)

r_{det} = Detachment rate

r_{ata} = Attachment rate

It was assumed that the biomass exchange between the biofilm and bulk liquid can be ignored in order to make the model simple. It was considered that the rates of

attachment and detachment are approximately same and the situation is similar to the natural system.

3.2.1.2. Soluble Components

A system of ordinary differential equations was used to represent the behavior of concentrations of soluble components in the bulk liquid. The mass balances take into account the rates of reactions in the bulk liquid, in the biofilm and on the electrode surface. The corresponding reaction rates for the bulk liquid were calculated using equations (02) – (06). Calculation of reaction rates in the biofilm and on the electrode surfaces are described in later sections. (Sections 3.2.2.3 and 3.2.3.1).

The rate of exchange between bulk liquid and biofilm can be expressed in one of two ways,

1) As the product between an average mass flux of component to or from the biofilm and the surface area of the biofilm
2) An average rate of reaction in the biofilm times the biofilm volume

The second option was adopted, because the change in the volumetric rates of solute components can be easily calculated.

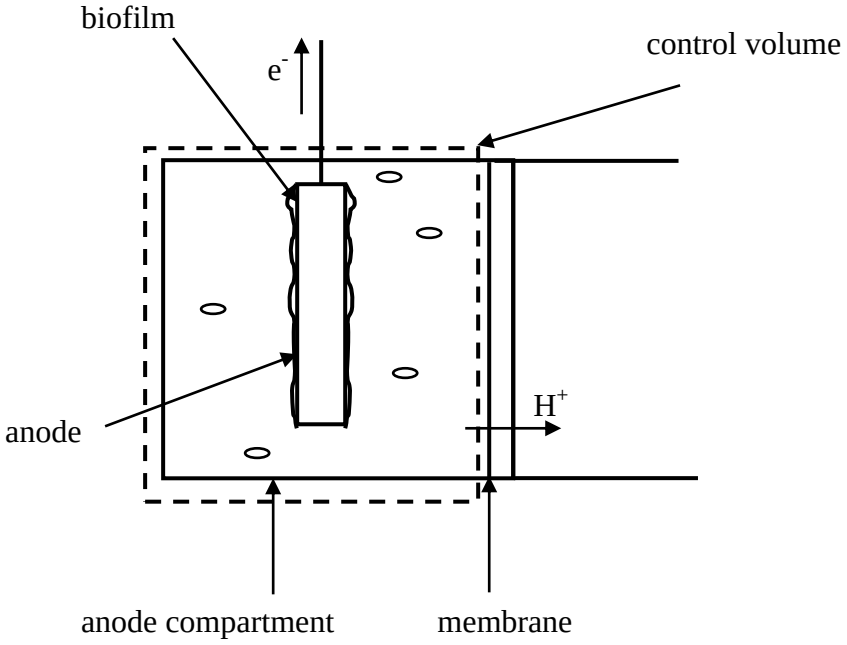


Figure 3.2 Control volume

The Eqs (02), (04) and (06) were extracted from a developed computational model for biofilm based microbial fuel cells by Cristian Picioreanu et al 2009.

The boundaries of the selected control volume are indicated in Figure 3.2. It includes the anode compartment with the electrode. Moreover a part of the external resistor to indicate the transfer of electrons from the anode was included.

For the selected control volume,

Substrate

$$\frac{dC_{S,L}}{dt} = r_{S,L} + \frac{1}{V_L}\int_{V_B} r_{S,B}\,dV \qquad \text{Eq. (02)}$$

Simplified form,

$$\frac{dC_{S,L}}{dt} = r_{S,L} + r_{S,B}\frac{V_B}{V_L} \qquad \text{Eq. (03)}$$

$C_{S,L} = C_{S,0}$ (Initial Condition)

Oxidized mediator

$$\frac{dC_{Mox,L}}{dt} = r_{Mox,L} + \frac{1}{V_L}\int_{V_B} r_{Mox,B}\,dV + \frac{1}{V_L}\int_{A_E} r_{Mox,E}\,dA \qquad \text{Eq. (04)}$$

Simplified form,

$$\frac{dC_{Mox,L}}{dt} = r_{Mox,L} + r_{Mox,B}\frac{V_B}{V_L} + r_{Mox,E}\frac{A_E}{V_L} \qquad \text{Eq. (05)}$$

$C_{Mox,L} = C_{Mox,0}$ (Initial Condition)

Reduced mediator

$$\frac{dC_{Mred,L}}{dt} = r_{Mred,L} + \frac{1}{V_L}\int_{V_B} r_{Mred,B}\,dV + \frac{1}{V_L}\int_{A_E} r_{Mred,E}\,dA \qquad \text{Eq. (06)}$$

Simplified form,

$$\frac{dC_{Mred,L}}{dt} = r_{Mred,L} + r_{Mred,B}\frac{V_B}{V_L} + r_{Mred,E}\frac{A_E}{V_L} \qquad \text{Eq. (07)}$$

$C_{Mred,L} = C_{Mred,0}$ (Initial Condition)

The original mass balance equations given by Eq. (02), (04) and (06) were further simplified and given by Eq. (03), (05) and (07).

In simplifying the equations, it was assumed that the reaction rates in the biofilm and on the electrode surface of a selected component are constants for a considered time step, which means the concentrations of components in the biofilm and on the electrode surface have different values but are steady in each domain. The validity of this assumption depends on the selection of time step, for a very small time step, the assumption makes small errors due to several reasons. One reason is during the considered time step, it was assumed that current density is constant even though it is not so. The other is, in solving the equations for the biofilm it was considered that the concentrations of components within the bulk liquid do not vary with time over one time step. The averaged concentrations (average of initial concentration and final concentration for the considered time step) of components in the bulk liquid were used to minimize the error of above mentioned consideration.

3.2.1.3. Microbial Reaction Kinetics in the Bulk Liquid

In this model, a double Monad limitation kinetic was assumed and the limiting components are substrate and oxidized mediator.

The expression for the rate,

$$\rho_L = \mu_{max,L} C_{X,L} \frac{C_{S,L}}{K_{S,L}+C_{S,L}} \frac{C_{Mox,L}}{K_{Mox,L}+C_{Mox,L}} \quad \text{Eq. (08)}$$

$\mu_{max,L}$ = Maximum rate coefficient in the bulk liquid

$C_{X,L}$ = Concentration of microorganisms in the bulk liquid

$C_{S,L}$ = Concentration of substrate in the bulk liquid

$C_{Mox,L}$ = Concentration of oxidized mediator in the bulk liquid

$K_{S,L}$ = Monod half saturation coefficient for substrate in the bulk liquid

$K_{Mox,L}$ = Monod half saturation coefficient for oxidized mediator in the bulk liquid

With the above expression, reaction rates for each component in the bulk liquid can be given as follows,

Reaction rate of microorganism in the bulk liquid,

$$r_{X,L} = Y_{X,L}\rho_L \quad \text{Eq. (09)}$$

$Y_{X,L}$= Yield of microorganisms in the bulk liquid

Reaction rate of substrate in the bulk liquid,

$$r_{S,L} = -Y_{S,L}\rho_L \quad \text{Eq. (10)}$$

$Y_{X,L}$= Yield of substrate in the bulk liquid

Reaction rate of oxidized mediator in the bulk liquid,

$$r_{Mox,L} = -Y_{Mox,L}\rho_L \quad \text{Eq. (11)}$$

$Y_{Mox,L}$= Yield of oxidized mediator in the bulk liquid

Reaction rate of reduced mediator in the bulk liquid,

$$r_{Mred,L} = Y_{Mox,L}\rho_L \quad \text{Eq. (12)}$$

The reaction rates are used in Equations (01), (03), (05) and (07) appropriately.

3.2.2. Modelling Biofilm

Biofilm is a growing phase, where its boundaries vary continuously with time due to the growth, attachment and detachment of microorganisms. The concentrations of components indicate both time and spatial distribution. Even though the concentrations vary in every direction, it was assumed that they have considerable

variations in the direction perpendicular to the electrode surface whereas the variations in the other directions are not significant.

Derived set of differential equations for a system that has no biofilm but suspended cells of microorganisms and a concentration boundary layer was extracted from a previous model (Picioreanu, et al., 2009). Then, the equations were modified so that they represent the behaviour of variation of concentration of components within the biofilm.

Microorganisms do not diffuse through a porous medium but it has a similar kind of behaviour as that of soluble components with an introducing term called the displacement rate. This displacement rate depends on concentration of microorganisms at the considered point and porosity of the system. Even though the diffusion coefficients of substrate and mediators are assumed as constant values, the displacement rate is not a constant and has spatial variation with the structure and the growth of the biofilm.

3.2.2.1. Biomass Component

A mass balance derived for the growth of microorganisms while consuming substrate is given in Eq. (13). (Modeling - IWA Task Group on Biofilm, 2004)

$$\frac{\partial C_{X,B}}{\partial t} = r_{X,B} \qquad \text{Eq. (13)}$$

$\frac{\partial C_{X,B}}{\partial z} = 0$ (Boundary Condition at the electrode surface)

$C_{X,B} = C_{X,L}$ (Initial Condition)

3.2.2.2. Soluble Components

For soluble components in the biofilm, it can be assumed that the mass transport is only by molecular diffusion and the dissolved components can be produced or consumed in several biotic or abiotic transformation processes, which are indicated by reaction rates in the model.

With the assumption that high medium conductivity will make the potential gradient is insignificant in the biofilm, migration of ions in an electrical potential field was neglected. The equations given below are considered to be in domains, where the domain has its bottom layer (z=0) on the electrode surface and the top layer ($z=z_L$) touching the bulk liquid.

Therefore it is clear that the upper boundary conditions are to be similar with those of bulk liquid conditions for the considered time.

In case of the bottom boundary conditions, it should be noted that the electrode surface is electrochemically active only for certain soluble components. In this case, only mediators (oxidized and reduced) are active for the electrode. For other components (substrate and microorganism), the electrode surface is inert and impermeable, hence have zero flux and zero reaction rate at the surface.

For mediators, the boundary condition at the electrode surface expresses the fact that the rate of superficial production of a species on the electrode surface must equal the flux out by diffusion.

For Substrate

$$\frac{\partial C_{S,B}}{\partial t} = D_S \frac{\partial^2 C_{S,B}}{\partial^2 z} + r_{S,B} \qquad \text{Eq. (14)}$$

$\frac{\partial C_{S,B}}{\partial z} = 0$ (Boundary Condition at the electrode surface)

$C_{S,B} = C_{S,L}$ (Initial Condition)

For Oxidized mediator

$$\frac{\partial C_{Mox,B}}{\partial t} = D_S \frac{\partial^2 C_{Mox,B}}{\partial^2 z} + r_{Mox,B} \qquad \text{Eq. (15)}$$

$D_{Mox} \frac{\partial C_{Mox,B}}{\partial z} + r_{Mox,E} = 0$ (Boundary Condition at the electrode surface)

$C_{Mox,B} = C_{Mox,L}$ (Initial Condition)

For Reduced mediator

$$\frac{\partial C_{Mred,B}}{\partial t} = D_S \frac{\partial^2 C_{Mred,B}}{\partial^2 z} + r_{Mred,B} \quad \text{Eq. (16)}$$

$D_{Mred} \frac{\partial C_{Mred,B}}{\partial z} + r_{Mred,E} = 0$ (Boundary Condition at the electrode surface)

$C_{Mred,B} = C_{Mred,L}$ (Initial Condition)

3.2.2.3. Microbial Reaction Kinetics in the Biofilm

In the biofilm as described in the bulk liquid above, a double Monad limitation kinetic was assumed and the limiting components are substrate and oxidized mediator.

The expression for the rate,

$$\rho_B = \mu_{max,B} C_{X,B} \frac{C_{S,B}}{K_{S,B}+C_{S,B}} \frac{C_{Mox,B}}{K_{Mox,B}+C_{Mox,B}} \quad \text{Eq. (17)}$$

$\mu_{max,B}$ = Maximum rate coefficient in the biofilm

$C_{X,B}$ = Concentration of microorganisms in the biofilm

$C_{S,B}$ = Concentration of substrate in the biofilm

$C_{Mox,B}$ = Concentration of oxidized mediator in the biofilm

$K_{S,B}$ = Monod half saturation coefficient for substrate in the biofilm

$K_{Mox,B}$ = Monod half saturation coefficient for oxidized mediator in the biofilm

With the above expression, reaction rates for each component in the biofilm can be given as follows,

Reaction rate of microorganism in the biofilm,

$$r_{X,B} = Y_{X,B} \rho_B \quad \text{Eq. (18)}$$

$Y_{X,B}$= Yield of microorganisms in the biofilm

Reaction rate of substrate in the biofilm,

$$r_{S,B} = -Y_{S,B}\rho_B \quad \text{Eq. (19)}$$

$Y_{S,B}$= Yield of substrate in the biofilm

Reaction rate of oxidized mediator in the biofilm,

$$r_{Mox,B} = -Y_{Mox,B}\rho_B \quad \text{Eq. (20)}$$

$Y_{Mox,B}$= Yield of oxidized mediator in the biofilm

Reaction rate of reduced mediator in the biofilm,

$$r_{Mred,B} = Y_{Mox,B}\rho_B \quad \text{Eq. (21)}$$

Reaction rates given by Equations (18), (19), (20) and (21) were used in Equations (13), (14), (15) and (16) appropriately.

3.2.3. Electrode Surface

Variation of concentrations at the electrode surface is almost similar to that in the biofilm and the concentration values can be extracted from the biofilm model. The values are similar to those in the biofilm at the electrode surface where the height value is zero (z=0).

Within the bulk liquid, a simple reaction was assumed that is an organic substrate S is oxidized by microorganisms using the oxidized mediator (mediator in the oxidized state has capability to oxidize other reactive components).

$$xS + M_{ox} \rightarrow M_{red} + Products \quad \text{Eq. (22)}$$

The reduced mediator produced biochemically is oxidized at the anode electrochemically,

$$M_{red} \Leftrightarrow M_{ox} + nH^+ + ne^- \quad \text{Eq. (23)}$$

The stoichiometric coefficient (x) depends on the electron content of the substrate. In case of complete oxidation of glucose to carbon dioxide, the reaction require 24 electrons to be accepted by the mediator, if thionine is the mediator, it accepts 2 electrons and require 12 mediators, gives x = 1/12 = 0.0833 mol mol^{-1} .

But not all available electrons contained in the substrate will be transferred to the anode, the real yield is lower than that of theoretical yield. Therefore the coulombic yield; $Y_Q<1$.

So, it is clear that in order to define the value of x, the actually produced current should be considered.

The stoichiometry of the electrochemical reaction reflects the number of electrons transferred, n. Here the considered mediator is thionine, and it is considered as n=2.

3.2.3.1. Electrochemical Reaction Rates

The rates of the electrochemical reactions, which occur at the electrode surface are necessary to be given to complete the model. When certain dissolved chemical species are oxidized on the anode and others are reduced on the cathode, an electrical current is generated.

The surface – based rates of electrochemical reactions occurring on the anode are expressed as a function of current density (i).

With the assumption of uniform current density all over the electrode surface, the current density can be expressed as,

$$i = \frac{I}{A_E} \quad \text{Eq. (24)}$$

Where,

i = current density (A/m^2)

I = current (A)

A_E = surface area of the electrode (m^2)

The net rates for oxidation of mediators at the electrode surface are then a function of current density, stoichiometry (n) and the concentration values of the oxidized and reduced mediators at the electrode surface (Faraday's Laws - see Appendix A).

Reaction rate of oxidized mediator on the electrode surface,

$$r_{Mox,E} = \frac{i(C_{Mox,E}, C_{Mred,E})}{nF} \quad \text{Eq. (25)}$$

Reaction rate of reduced mediator on the electrode surface,

$$r_{Mred,E} = -\frac{i(C_{Mox,E}, C_{Mred,E})}{nF} \quad \text{Eq. (26)}$$

Where,

n = stoichiometry coefficient (n = 2 for thionine)

F = Faraday constant (96485.33 C mol^{-1})

The current density produced in the electrochemical mediator oxidation can be expressed with the use of Butler – Volmer equation.

$$i = i_{0,ref}\left(\frac{C_{Mred,E}}{C_{Mox,E}}\right)\left(\frac{C_{Mred,E}}{C_{Mox,E}}\right)^{-1}\left[exp\left(\frac{2.303}{b}\eta_{A,act}\right) - exp\left(-\frac{2.303}{b}\eta_{A,act}\right)\right] \quad \text{Eq. (27)}$$

Where,

$i_{0,ref}$ = exchange current density for mediator oxidation in reference conditions

b = Tafel slope (V/decade of current)

$\eta_{A,act}$ = activation over potential (for the anodic electrochemical reaction) (V)

The unknown $\eta_{A,act}$ can be calculated with the Equation (28),

With constant pH value and temperature as 30 ^{0}C,

$$\eta_{A,act} = V_C - I(R_{int} + R_{ext}) - \left(E^0_{Mox/Mred} - 0.06pH + \frac{0.06}{2} lg \frac{C_{Mox,E}}{C_{Mred,E}}\right) \quad \text{Eq. (28)}$$

Where,

R_{int} = summation of electronic, ionic and contact resistance (Ω)

R_{ext} = external cell resistance (Ω)

V_C = cathode potential (V)

$E^0_{Mox/Mred}$ = standard redox potential

pH = pH value

Using equations (27) and (28), the current density can be calculated for known concentrations of oxidized and reduced mediators at the electrode surface (anode).

The calculated current density can be further used to find out reaction rates at the electrode surface using Equation (25) and (26). The resultant reactions rates were used in solving equations for bulk liquid and for biofilm.

Table 3.1 Initial conditions of components

Region	Component	Initial Concentration (mol/ m^3)
Bulk liquid	Microorganisms	0.02
	Substrate	0.0005
	Oxidized mediator	0.01
	Reduced mediator	0.00001
Biofilm	Microorganisms	100
	Substrate	$1x10^{-8}$
	Oxidized mediator	$1x10^{-8}$
	Reduced mediator	$1x10^{-12}$

3.3. Solving the Equations

Parameters used in the new biofilm model is given in Table 3.2

Table 3.2 Model Parameters

Parameter	Description	Value	Unit
D_S	Diffusion coefficient of substrate in the biofilm[a]	$1.1574x10^{-9}$	m^2/s
D_{Mox}	Diffusion coefficient of oxidized mediator in the biofilm[a]	$1.0417x10^{-9}$	m^2/s
D_{Mred}	Diffusion coefficient of reduced mediator in the biofilm[a]	$1.0417x10^{-9}$	m^2/s
pH	[a]	7	-
V_B	Bulk liquid volume[b]	0.000035	m^3
A_E	Anode surface area[b]	0.001	m^2
V_C	Cathode Potential[a]	0.68	V
$R_{int}+R_{ext}$	Total cell resistance[a]	100	Ω
$i_{0,ref}$	Exchange current density for mediator oxidation in reference conditions[d]	0.0001	A/m^2
$E^0_{Mox/Mred}$	Standard reduction potential for the mediator couple[c]	0.477	V
b	Tafel coefficient for mediator oxidation[d]	0.18	V
Y_X	Yield factor of microorganisms[d]	0.12	mol microorganism /mol substrate
Y_S	Yield factor of substrate[d]	1	mol substrate /mol substrate
Y_{Mox}	Yield factor of mediators[d]	3	mol mediator /mol substrate
K_S	Monod half-saturation coefficient for substrate[c]	$1x10^{-4}$	mol
K_{Mox}	Monod half-saturation coefficient for oxidized mediator[c]	$2x10^{-4}$	mol
μ_{max}	Maximum specific rate constant[b]	$5x10^{-8}$	Mol mediator $(mol\ biomass)^{-1}\ s^{-1}$

[a] From Picioreanu, et al., 2007

[b] Assumed values

[c] From Picioreanu, et al., 2009

[d] Assumed considering the value available in Picioreanu, et al., 2007

Execution of the developed model

MATLAB programming language was used to solve the model (Mathworks, 2010).

Execution of the model developed is as follows.

Step 1: Assign initial concentrations of components

Four components are considered and they have different concentration values in the bulk liquid and in the biofilm. In this step, concentration values are assigned for the components in the both regions. For each and every time steps, these concentration values are the selected solutions from previous time step.

Step 2: Assign a value for current density

A value for current density is needed to be assigned in order to solve the equations, therefore a small current density value is assigned at the beginning (t=0). Afterwards, for next iterations calculated current density value from previous iteration is used.

Step 3: Solve equations for bulk liquid

For the bulk liquid, set of ordinary differential equations are available and needed to be solved for each time step. Assigned initial concentration values and given parameters (reaction rate constants, yield factors, etc…) are used in solving these equations.

Step 4: Solve equations for biofilm

For the biofilm, set of partial differential equations are available and needed to be solved for each time step. Assigned initial concentration values and given parameters (reaction rate constants, yield factors, diffusivity coefficients, etc…) are used in solving these equations. For the boundary conditions at the biofilm and bulk liquid interface, averaged concentrations of each component within the bulk liquid over the simulated time step are used.

Step 5: Calculate current density

According to Butler-Volmer expression, for a microbial fuel cell, current density varies with the concentrations of mediators at the electrode surface. The concentration varies with time as well. In this method, averaged concentration of each mediator is calculated using the solutions of biofilm for the considered time step and using these averaged values current density is calculated.

Solving steps of the model are given in Figure 3.3.

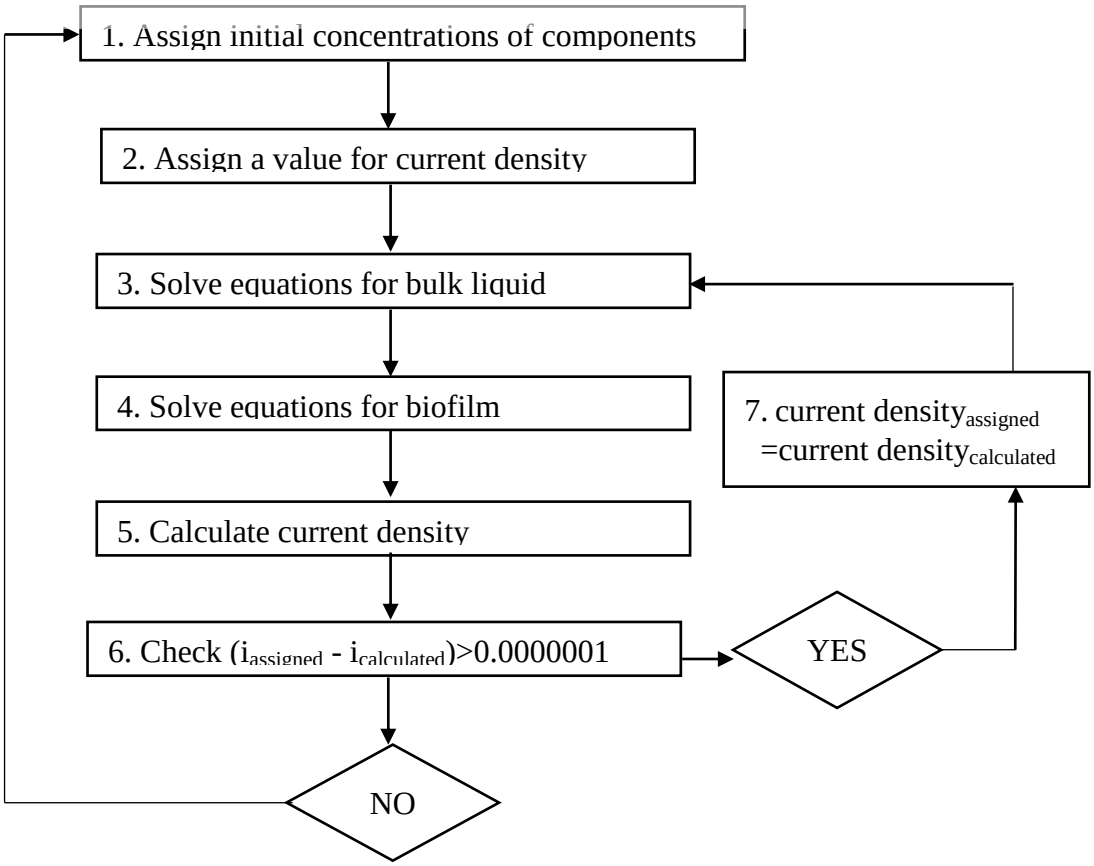


Figure 3.3 Model Solving Procedure

Step 6: Check the difference

It is checked whether there is a difference between calculated and assumed current density values. If the difference is considerably high the process is repeated from step 2 onwards taking the calculated current density as the assigned current density value (Step 7). The loop runs for number of times until the difference is negligible. When the required condition is reached; the process is moved to step 1.

Step 7: current density$_{assigned}$ =current density$_{calculated}$

For the next iteration within the same time step, it is taken that;

assigned current density = calculated current density.

3.4. Model Verification

The developed model was verified using simulated results of another computational model developed by Cristian Picioreanu and team in 2007. It should be noted that the selected model is a 3 dimensional model. But, with previous research works it has been proven that the 1 dimensional models give approximately similar results as those of 3 dimensional models.

Details of this model can be found in,

http://biofilms.bt.tudelft.nl/pdf/2007_WaterResearch_3_Picioreanu-et-al.pdf

In verification of the model, the parameters used in solving that model were applied in the developed model and the obtained simulated results were compared with the available results.

Table 3.3 indicates the initial conditions and used model parameters. The model parameters are exactly same those were given for the selected model.

The newly developed model was simulated using the model parameters and initial conditions given in Table 3.3. The comparisons of simulated results with the results of the previously developed model are illustrated in section 4.5.

Table 3.3 Model Parameters used in Verification

Parameter	Description	Value	Unit
$C_{Mox,L}$	Initial concentration of oxidized mediator (in the bulk liquid)	1	mM
$C_{Mred,L}$	Initial concentration of reduced mediator (in the bulk liquid)	$1x10^{-3}$	mM
D_S	Diffusion coefficient of substrate in the biofilm	$1.1574x10^{-9}$	m^2/s
D_{Mox}	Diffusion coefficient of oxidized mediator in the biofilm	$1.0417x10^{-9}$	m^2/s
D_{Mred}	Diffusion coefficient of reduced mediator in the biofilm	$1.0417x10^{-9}$	m^2/s
pH		7	-
V_C	Cathode Potential	0.68	V
$R_{int}+R_{ext}$	Total cell resistance	100	Ω
$i_{0,ref}$	Exchange current density for mediator oxidation in reference conditions	0.0002	A/m^2
$E^0_{Mox/Mred}$	Standard reduction potential for the mediator couple	0.477	V
b	Tafel coefficient for mediator oxidation	0.12	V
Y_X	Yield factor of microorganisms	0.243	g COD biomass / g COD acetate
Y_{Mox}	Yield factor of mediators	0.0473	mol mediator /g COD acetate
K_S	Monod half-saturation coefficient for substrate	100	$g\ COD/m^{-3}$
K_{Mox}	Monod half-saturation coefficient for oxidized mediator	$1x10^{-4}$	mol
μ_{max}	Maximum specific rate constant	10	g COD acetate $(g\ COD\ biomass)^{-1}$ day^{-1}

(Picioreanu, et al., 2007)

4. RESULTS AND DISCUSSION

This mathematical model describes the behaviour of biofilm based microbial fuel cells. The main objective of the model is the prediction of variation of current with time. Other than the prediction of current, this gives variation of concentration of components (substrate, microorganism, oxidized mediator and reduced mediator) in the bulk liquid, biofilm and at the electrode surface.

Several simulations were performed in order to analyze the effect of selected model parameters on the current production and further on the variations of concentrations. Selected model parameters and the used values for the parameters are listed in Table 4.1.

Other parameters and initial conditions of the components in the bulk liquid and in the biofilm are similar for all cases and as indicated with Table 3.1 and Table 3.2.

Table 4.1 Values of Selected Parameters

Simulation - Name	Reaction Rate Coefficient	Reference Current Density /(A/m^2)	Total Resistance /(Ω)
Case 1	5.0×10^{-8}	0.0001	100
Case 2	1.0×10^{-7}	0.0001	100
Case 3	2.0×10^{-7}	0.0001	100
Case 4	5.0×10^{-8}	0.0002	100
Case 5	5.0×10^{-8}	0.0001	150
Case 6	5.0×10^{-8}	0.0001	250
Case 7	5.0×10^{-8}	0.0001	500

4.1. Simulation for reference case (case 1)

Simulated results of considered case in graphical form are shown in Figures 4.1 to 4.7. Simulated current values of case 1 are indicated in Figure 4.1.

The concentration of microorganisms (Figure 4.2) and reduced mediator (Figure 4.5) in the bulk liquid initially increase with time. This increasing of microorganism and reduced mediator is due to availability of enough substrate to be used by

microorganism for their own growth and hence to produce reduced mediators. Concentration of oxidized mediator in the bulk liquid decreases initially (Figure 4.4) due to two reasons. Reason one; oxidized mediator is used and reduced mediator is produced. At the same time, the concentration of oxidized mediator at the electrode surface is not much high to give an increasing of concentration of oxidized mediator in the bulk liquid. Because of occurring of these two simultaneous incidents the concentration of oxidized mediator decreases in the beginning of the process.

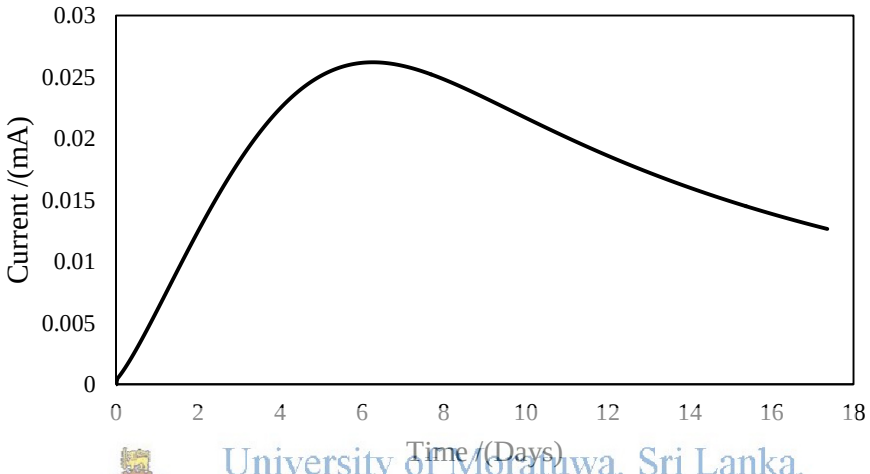


Figure 4.1 Variation of current with time (Case 1)

After depletion of substrate the concentration of microorganisms does not vary. Decay of microorganism has been neglected in this model (Figure 4.2). Concentration of reduced mediator starts to decrease, because microorganisms are not capable of producing any more reduced mediators under no substrate condition. Concentration of oxidized mediator starts to increase while that of reduced mediator is decreasing.

Initially the concentration of reduced mediator was a very low value, but with time it increases due to the activity of microorganisms in the biofilm. When the simulation reaches day 6 the current production rate is also high and this result in high reaction rate of consuming reduced mediators and producing oxidized mediators at the electrode surface (see Figures 4.8 and 4.9).

The current initially increases over several days and reached a peak value (0.027 mA). After that point it starts to decrease as indicated in Figure 4.1. This can be explained with variations of concentrations of oxidized mediator and reduced mediator at the electrode surface.

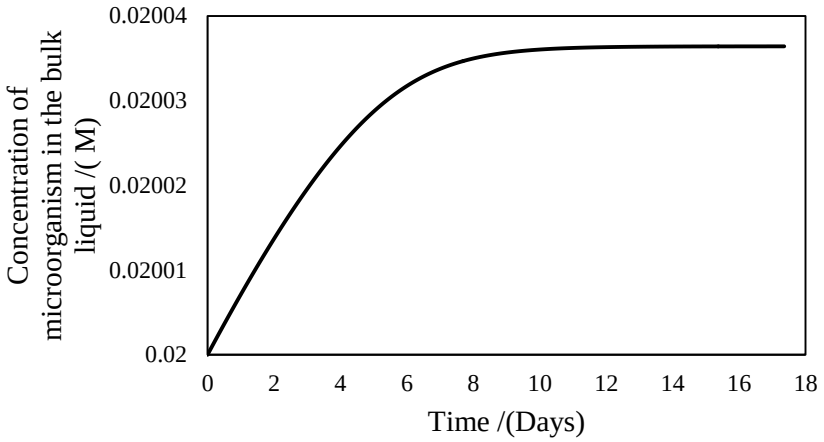


Figure 4.2 Variation of concentration of microorganisms in the bulk liquid with time (case 1)

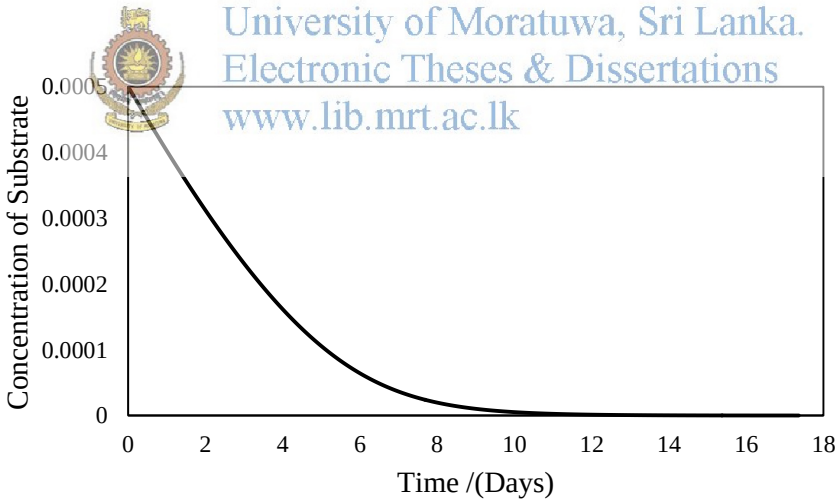


Figure 4.3 Variation of concentration of substrate in the bulk liquid with time (case 1)

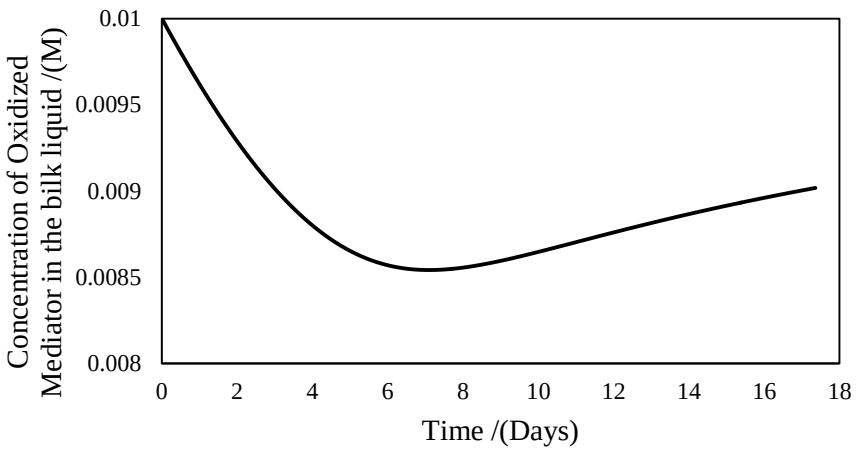


Figure 4.4 Variation of concentration of oxidized mediator in the bulk liquid with time (case 1)

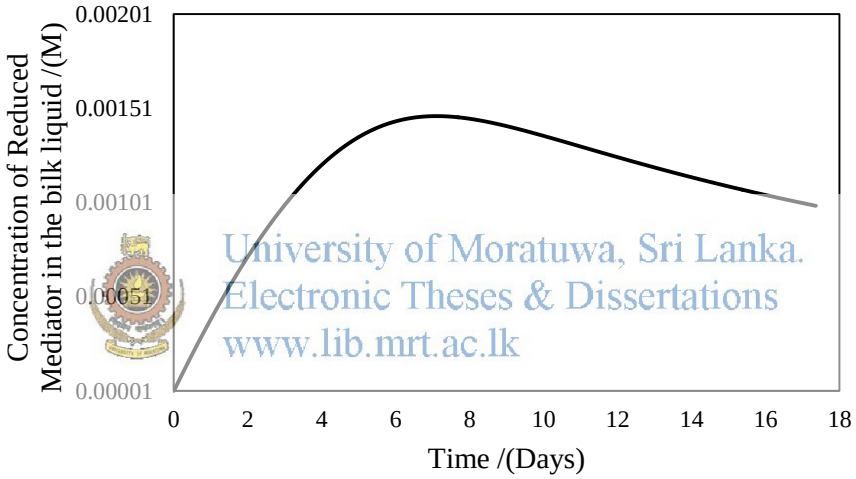


Figure 4.5 Variation of concentration of reduced mediator in the bulk liquid with time (case 1)

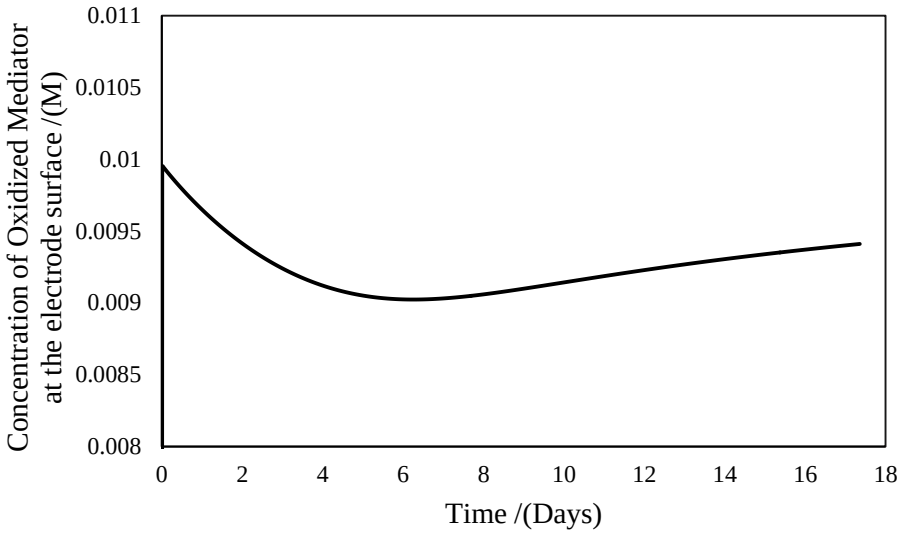


Figure 4.6 Variation of concentration of oxidized mediator at the electrode surface with time (case 1)

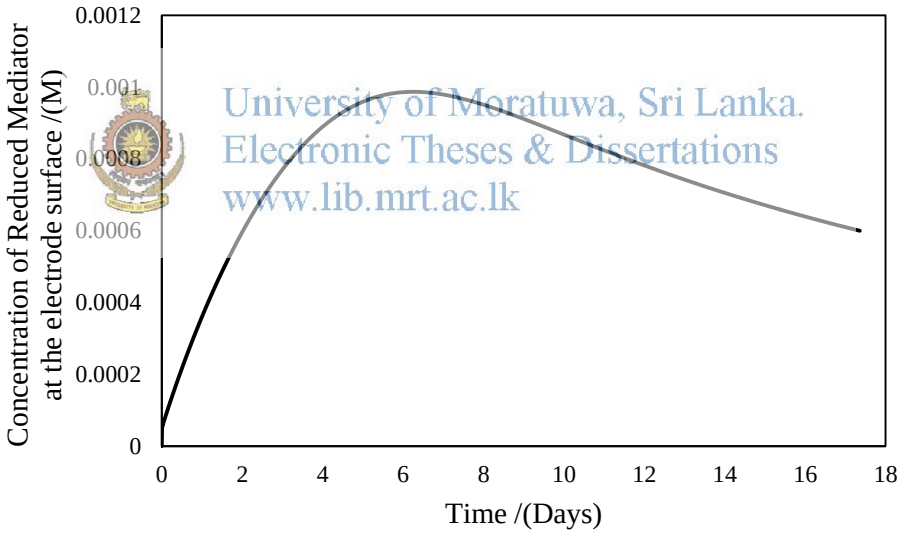


Figure 4.7 Variation of concentration of reduced mediator at the electrode surface with time (case 1)

4.2. Simulations for different reaction rate values

Reaction rate constant was assigned with different values under 3 cases (case 1, case 2 and case 3) and simulations were performed. The considered reaction rate values are indicated in Table 4.1.

Production of current and variation of concentrations of components in the bulk liquid and at the electrode surface were compared for the considered cases using simulated results.

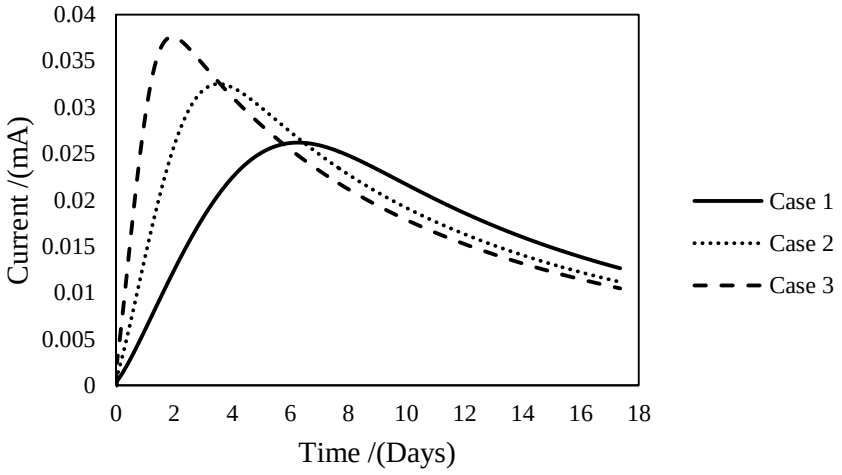


Figure 4.8 Variation of current for different reaction rates with time (case 1, case 2 and case 3)

Reaction rate constant values increase as shown in the below order,

case 1<case 2<case 3

The simulated results for variation of current with time indicate clearly the effect of different reaction rates. A system with lower reaction rate constants take longer time to give the peak value of current on the other hand a system with higher reaction rate constants gives its peak value rapidly.

Microorganisms reach their maximum concentration value in the bulk liquid with a comparatively short time when the used reaction rate constant is high (see Figure 4.10). Systems with low reaction rate constants take longer time to reach the maximum concentration. In all cases the achieved maximum concentrations are equal, cause the used initial concentration of substrate and microorganisms were kept similar for all cases.

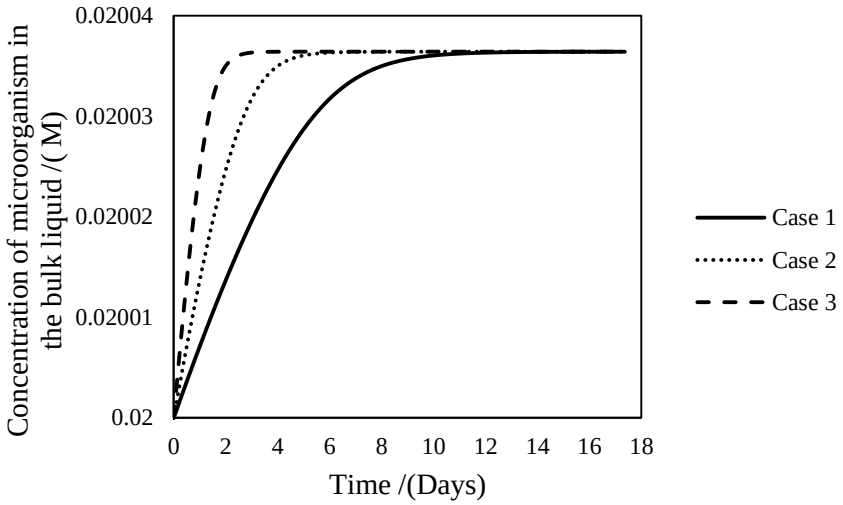


Figure 4.9 Variation of concentration of microorganisms in the bulk liquid for different reaction rates with time (case 1, case 2 and case 3)

The growth of microorganisms ceases when there is no substrate in the media; they achieve a maximum plateau phase by that time. The decay of microorganisms is not considered in this model.

The depletion of substrate is quite high in systems with high reaction rate constants and vice versa. The situations are shown in Figure 4.11.

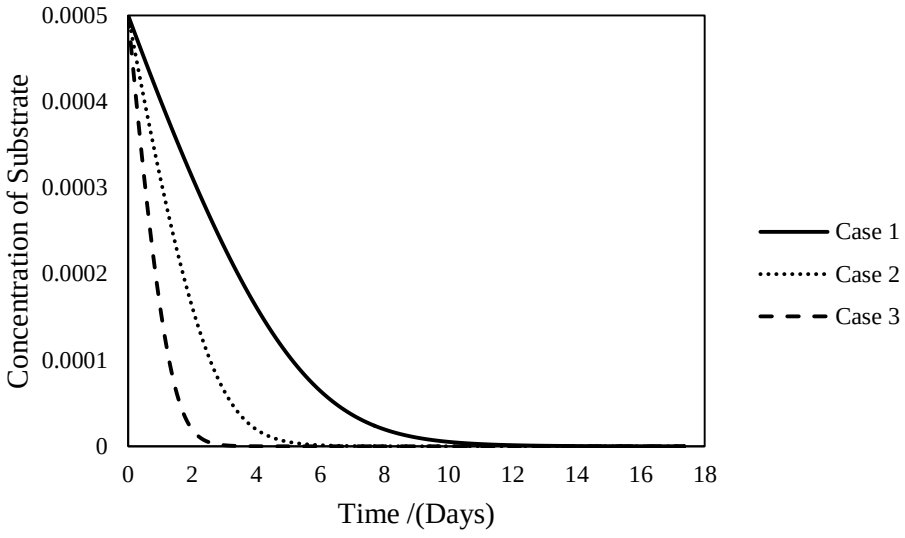


Figure 4.10 Variation of concentration of substrate in the bulk liquid with time for different reaction rates (case 1, case 2 and case 3)

Effects of reaction rates on the variation of concentrations of oxidized mediator and reduced mediator in the bulk liquid are shown in Figure 4.12 and Figure 4.13 respectively. Concentration of oxidized mediator reaches its top and concentration of reduced mediator reaches its minimum in short time when the reaction rate constant of the system is high (case 3). In case 1, where the system was operated with comparatively low reaction rate constants, reaching of turning points of concentrations of mediators in the bulk liquid take long time.

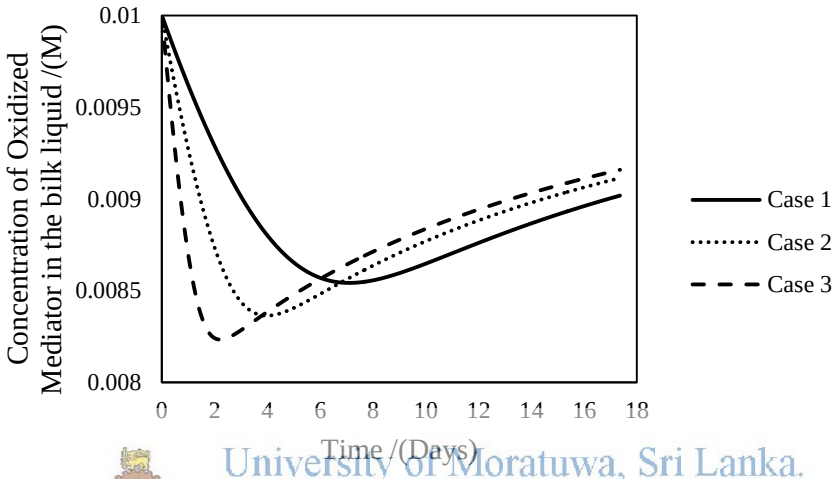


Figure 4.11 Variation of concentration of oxidized mediator in the bulk liquid with time for different reaction rates (case 1, case 2 and case 3)

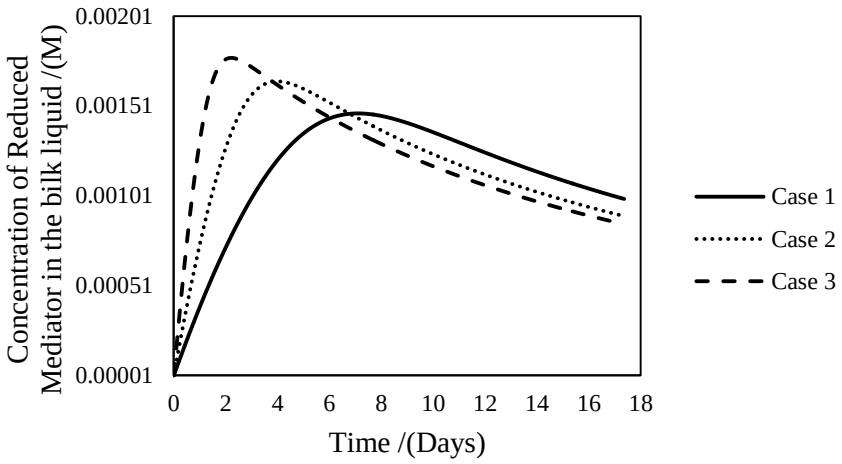


Figure 4.12 Variation of concentration of reduced mediator in the bulk liquid with time for different reaction rates (case 1, case 2 and case 3)

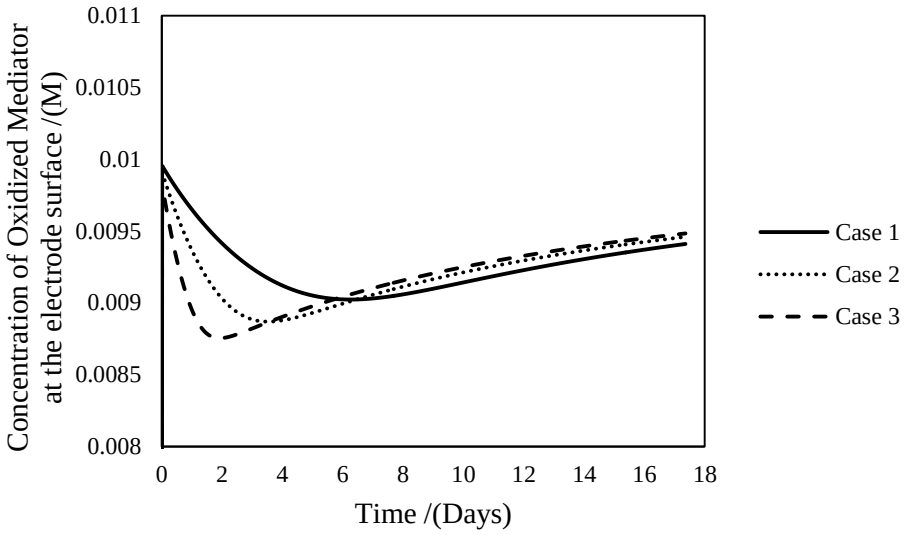


Figure 4.13 Variation of concentration of oxidized mediator at the electrode surface with time for different reaction rates (case 1, case 2 and case 3)

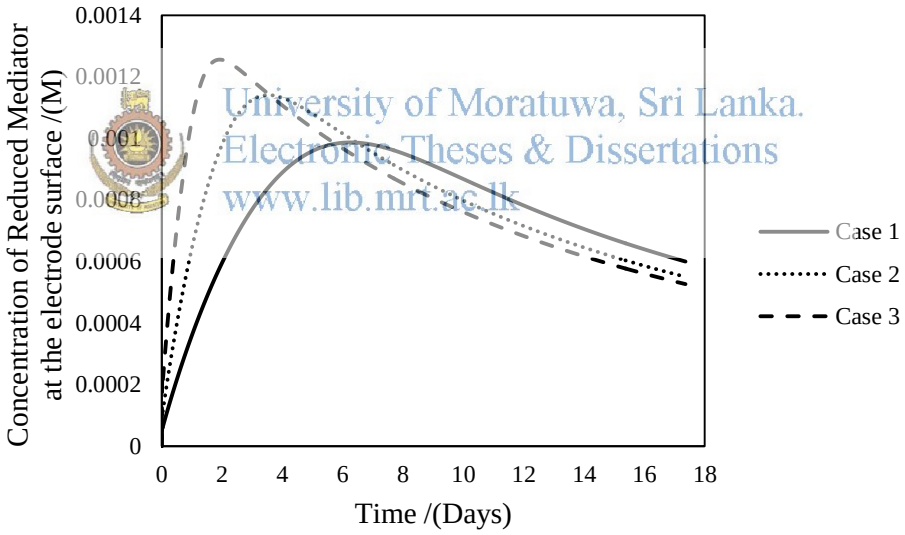


Figure 4.14 Variation of concentration of reduced mediator at the electrode surface with time for different reaction rates (case 1, case 2 and case 3)

Variations of concentration of mediators at the electrode surface also indicate similar behaviour as that at the bulk liquid (see Figure 4.14 and Figure 4.15). But the concentration of oxidized mediator is high than that of in the bulk liquid and contrarily the concentration of reduced mediator is low at the electrode surface than

that of in the bulk liquid. Production of current at the electrode surface, on the other hand resulting in a production of oxidized mediator using reduced mediator at the electrode surface. The microorganisms produce reduced mediator from oxidized mediator within the biofilm. The variation of concentration of mediators at the electrode surface is a balance of the mentioned reactions. Reason 1 is intensive than reason 2, specially when the production of current is high. Therefore the concentration of oxidized mediator is high and concentration of reduced mediator is low at the electrode surface than those at the bulk liquid.

4.3. Simulations for different exchange current density values

Exchange current density for mediator oxidation in reference conditions ($i_{o,ref}$) was assigned different values and the production of current and variation of concentration of components were analyzed using simulated results in each case.

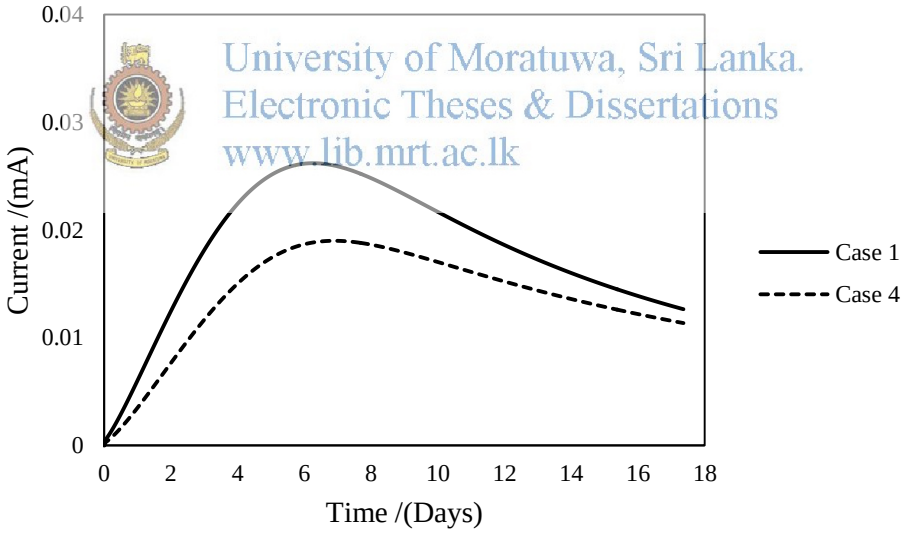


Figure 4.15 Variation of current with time for different exchange current densities (case 1 and case 4)

Case 1 and case 4 were compared in this section, see Table 4.1 for used parameters; the exchange current density values assigned for the system in case 1 is 0.0001 A/m^2 and in case 4 it is 0.0002 A/m^2. According to Butler-Volmer expression (Eq (27)) it is obvious that systems with less exchange current densities gives comparatively high

current production than those of systems with high exchange current density values. The simulated results in Figure 4.16 clearly indicate the mentioned concept.

Variation of concentration of microorganism and substrate in bulk liquid with time is negligible in the considered cases (case 1 and case 4). The exchange current density value directly effects the production of current hence concentration of mediators, and does not produce a considerable alteration on concentrations of microorganisms and substrate.

As mentioned above the effect on variation of concentration of mediators in the bulk liquid and at the electrode surface is substantial, the simulated results are shown in Figure 4.17, Figure 4.18, Figure 4.19 and Figure 4.20.

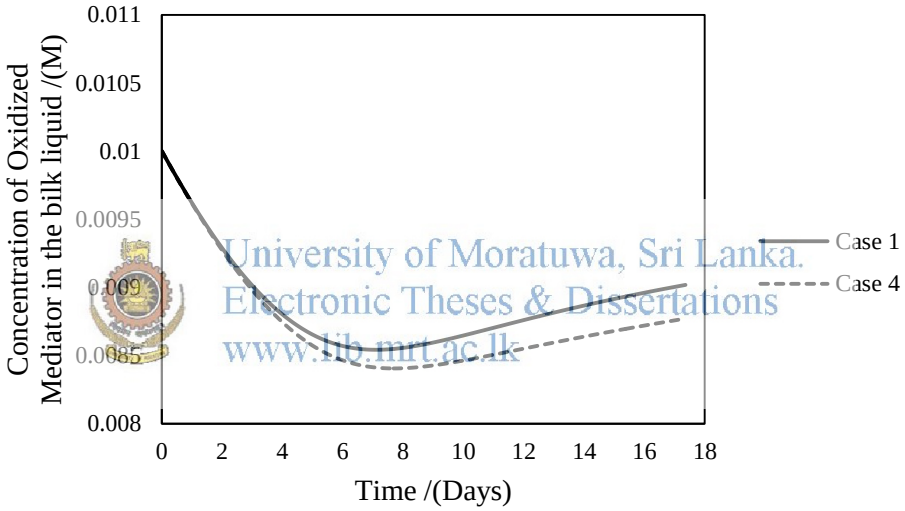


Figure 4.16 Variation of concentration of oxidized mediator in the bulk liquid with time for different exchange current densities (case 1 and case 4)

In case 4, the production of current is lesser than that in case 1. These results give a lower concentration of oxidized mediator and higher concentration of reduced mediator in the bulk liquid for case 4 when comparing with case 1 (see Figure 4.17 and Figure 4.18).

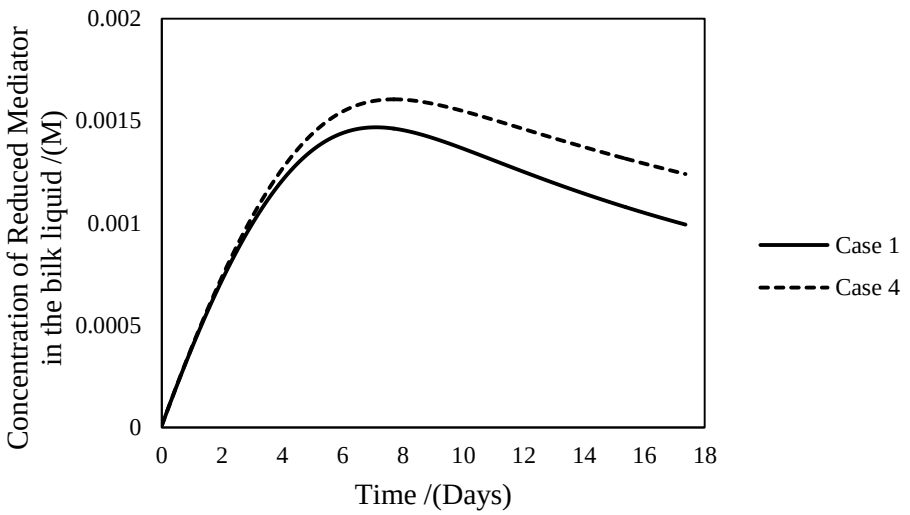


Figure 4.17 Variation of concentration of reduced mediator in the bulk liquid with time for different exchange current densities (case 1 and case 4)

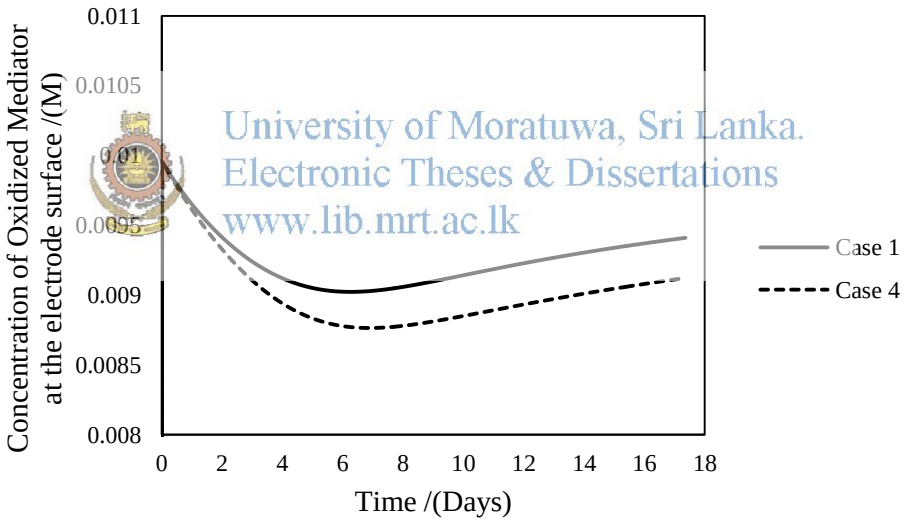


Figure 4.18 Variation of concentration of oxidized mediator at the electrode surface with time for different exchange current densities (case 1 and case 4)

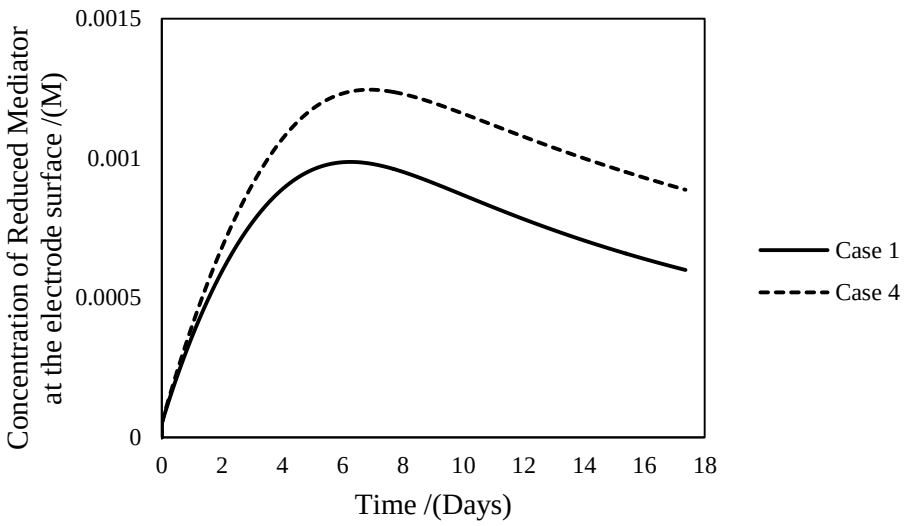


Figure 4.19 Variation of concentration of reduced mediator at the electrode surface with time for different exchange current densities (case 1 and case 4)

Variation of concentration of mediators at the electrode surface indicates similar pattern as those of in the bulk liquid. It is shown in Figure 4.19 and Figure 4.20 in sequence.

4.4. Simulations for different total cell resistance values

Total cell resistance (R_{tot} = R_{int} + R_{ext}) was assigned different values and the production of current and variation of concentration of components were analyzed using simulated results in each case. Here 3 different cases were considered as case 5, case 6 and case 7; the values are indicated in Table 4.1.

The simulated results of production of current for the three considered cases are shown in Figure 4.21. When the used total cell resistance values is less the production of current is low, but there is a tendency for being sustain the current for a little higher time in this case.

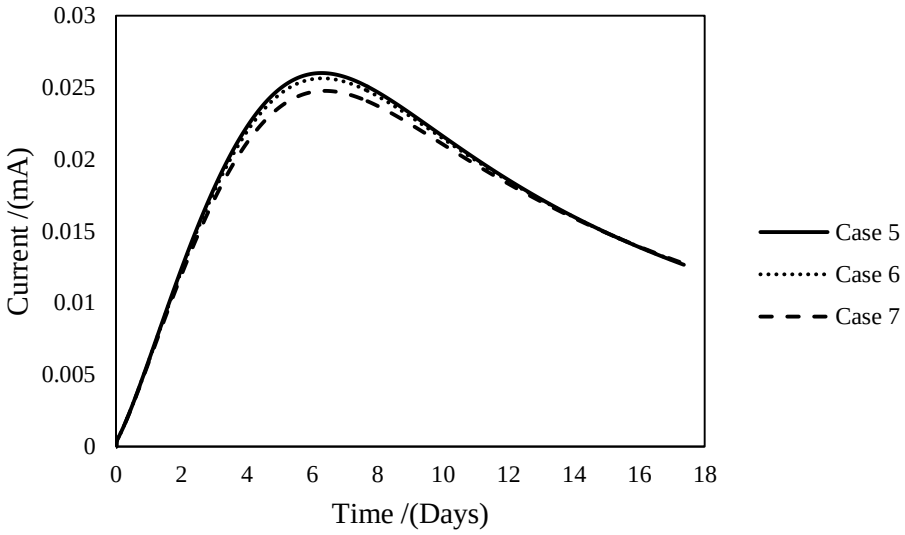


Figure 4.20 Variation of current with time for different total cell resistances (case 5, case 6 and case 7)

The variation of concentration of other components in the bulk liquid and mediators at the electrode surface is negligible in the considered cases.

4.5. Results on Model Verification

The simulated result of the developed model was compared with a previous model (developed by Cristian Picioreanu and team in 2007. In verification, same model parameters and initial conditions (see Table 3.3) were used as in the considered model.

Variation of current production with time, variation of concentrations of oxidized mediator and reduced mediator in the bulk liquid with time were considered.

Figure 4.22 indicates a comparison of current values in the two models. At the beginning both models shows a fairly good agreement in current production. But later, old model reaches a higher current production than the new model. The new model reaches a peak value little earlier than the old model that is by day 4. The difference between the peak values of the current in models is about 0.2 mA.

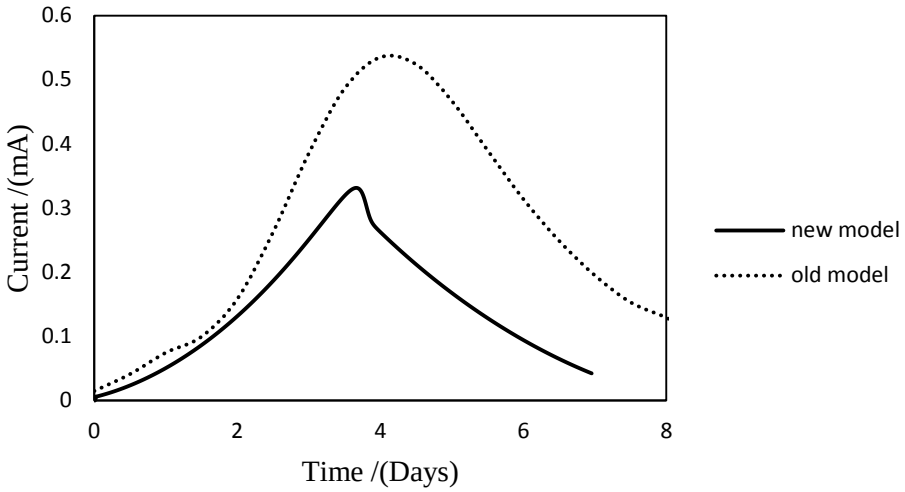


Figure 4.21 Variation of current with time in two models

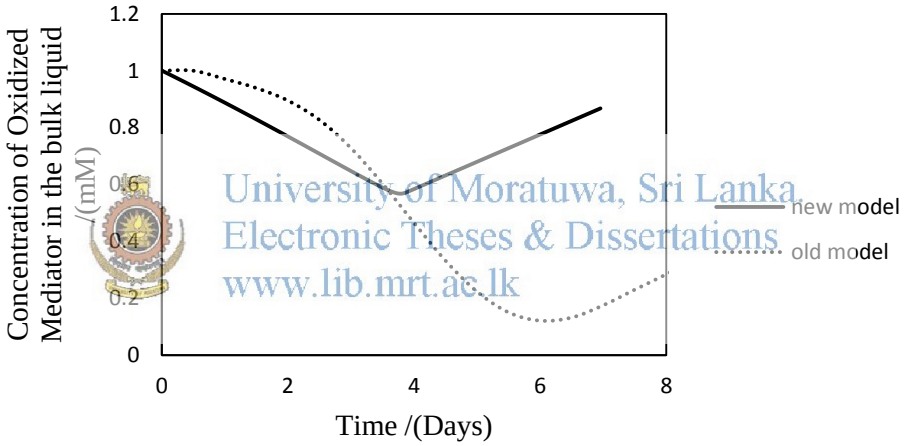


Figure 4.22 Variation of concentration of oxidized mediator in the bulk liquid with time in two models

Variations of concentrations of mediators in the bulk liquid also shows similar pattern in both models. Until day 4, the concentrations of mediators follow similar pattern in case of oxidized mediator and reduced mediator (see Figure 4.23 and Figure 4.24). But after day 4, the new model shows the starting of drop of current production, accordingly the concentrations indicate a turning point by that time.

According to Figure 4.23, old model indicates a further decreasing of concentration of oxidized mediator up to about 0.2 mM by day 6. Afterwards the concentration rises.

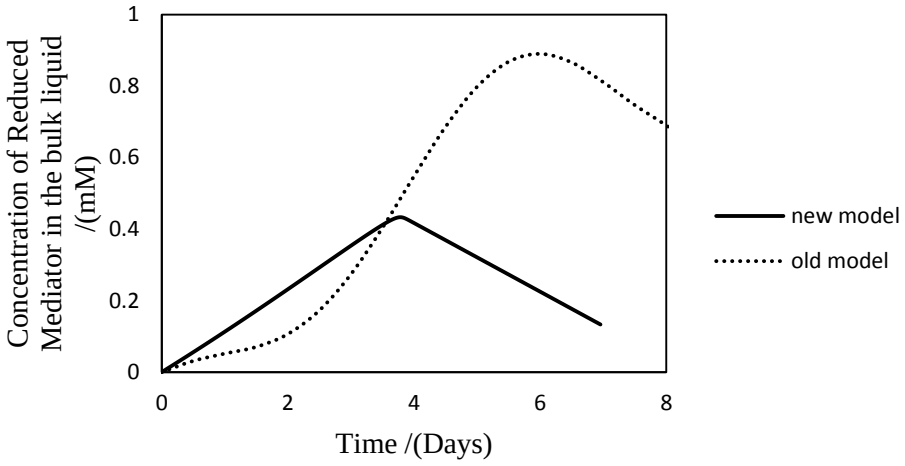


Figure 4.23 Variation of concentration of reduced mediator in the bulk liquid with time in two models

According to Figure 4.24, old model indicates an increasing of concentration of reduced mediator up to about 0.85 mM by day 6. Afterwards the concentration drops down.

5. CONCLUSIONS

A one dimensional model that was developed with a concentration boundary layer by Cristian and team in 2009 was modified so that the model has a biofilm on the anode instead of the concentration boundary layer.

The computational model presents in this study considers biological, chemical and physical processes occurring in microbial fuel cell with suspended cells, a biofilm attached on the anode and diffusible electron transfer mediators. The model can be used to predict experimental data with limited conditions.

The developed model is a simple one dimensional model; therefore it takes a comparatively small time in simulations (approximately 10 minutes for a 1 day simulation) whereas most of the other models take days for simulations.

Other than the simplification and reducing the simulation time, the model is able to give variations of current and concentrations of component in the bulk liquid and in the biofilm. The variations of concentration of mediators at the electrode surface with time, was identified as an important thing because the ratio of concentration of mediator plays a major role in production of current. The newly developed computational model is capable in observing this variation.

The simulated results show the effect of different operational parameters (reaction rate constant, exchange current density in reference conditions and total cell resistance) on the microbial fuel cell characteristics; production of current, variation of concentration of components in the bulk liquid and variation of concentration of mediators at the electrode surface with time.

The model verification was performed by comparing the simulated results of the model with simulated results of another validated computational model, due to the scarcity of experimental data in this section with well-defined experimental conditions.

The developed model does not indicate a good fitting with the observed data but gives a similar pattern of variations in current and concentrations of components in the bulk liquid. The two curves initially follow a same path but after day 4 they

indicate a deviation. The reasons may be the made assumptions in developing this model. It was assumed that, there is an initial thickness for the biofilm (1×10^{-6} m) in this model to avoid the complexity in the formation of biofilm at the beginning whereas the considered model (developed by Cristian and team in 2007) has no initial biofilm thickness. Moreover, in the developed model it was assumed that there is a thin intermediated layer exists in between the biofilm and bulk liquid, called as 'mass transfer boundary layer' which is not significant. But, the model used in verification has the mass transfer boundary layer with a considerable thickness.

This model assumes a simple mechanism for electron transfer to the electrode, but the actual mechanism may be not exactly the assumed one. Current research works on microbial fuel cells, reveals there can be different mechanism for electron transfer to the electrode and future works on this are needed to be considered in developing the model, these mechanisms also.

Further in this model single substrate is considered with only one type of microorganism in the system. The microbial fuel cell technology is needed to be extended to treat wastewater and produce energy in industrial scale.

In usual practice a mixture of different types of microorganisms are used to treat complex substrates like wastewater. Therefore it is necessary to modify the model to treat mixture of substrates when mixed microbial communities exist in the system (as suspended cells and on the biofilm attached to the anode).

Experimental studies are necessary to be conducted with different wastewater samples for microbial fuel cells in lab scale for validation of the model with experimental readings.

The amount of production of energy by microbial fuel cells while treating wastewater strongly depends on quality of wastewater and characteristics of microorganisms.

6. USER INETRFACE (GUI)

In order to make the model more user friendly and to view simulated results in graphical form (the most convenient way for easy understanding) a package that consists of several user interfaces was created using MATLAB.

The package contains three sub user interfaces and one pdf document, linked to a main graphical user interface.

- Main GUI – This is the main gui and has links to the other guis and the attached document (.pdf).
- Parameters GUI – This user interface indicates a list of used model parameters, descriptions and the values with appropriate units.
- Initialization GUI – This is a kind of active gui, where user can change certain parameters.

  Maximum specific rate constant, exchange current density for mediator oxidation in reference conditions and total cell resistance can be specified using the corresponding popupmenus. Further simulation time step also can be changed with this graphical user interface.

  See appendix B for a user manual of the package.

- Solving GUI – Even though the name is used as 'Solving GUI' this user interface can be used for two main purposes.
  i. To solve the model equations with given initial conditions over the defined time period
  ii. To view the simulated results – profiles of current, concentration of components in the bulk liquid, biofilm and at the electrode surface.
- PDF document – This document gives a brief description of the used model by indicating the used model equations.

For a detailed explanation about usage of the prepared package see the given user manual in Appendix B.

7. REFERENCES

Available at: http://en.wikipedia.org/wiki/Faraday%27s_laws_of_electrolysis [Accessed 8 5 2013]

Batstone, D. et al., 2002. The IWA Anaerobic Digestion Model No 1 (ADM1). *Water Science and Technology,* pp. 65-73.

Cristian Picioreanu, J.-L. K. M. C. v. L., 2004. Particle Based Multidimensional Multispecies Biofilm Model. *Applied and Environmental Microbiology,* pp. 3024-3040.

Devasahayam, M., 2012. Microbial fuel cells demonstrate high coulombic efficiency applicable for water remediation. *Indian Journal of Experimental Biology,* pp. 430-438.

EG&G Technical Services, Inc., 2004. *Fuel cell handbook.* 7th ed, West Virginia.

Frank & Ashley, E., 2010. Bacterial Biofilms: the powerhouse of a microbial fuel cell. *Biofuels,* pp. 589-604.

Frank & Ashley, E., 2010. Microbial Fue Cells, A Current Review. *energies,* pp. 900 - 903.

Logan, B. E., 2008. *Microbial FuelCells.* The United States of America: John Willey & Sons.

Logan & Bruce, E., 2008. *Microbial Fuel Cells.* New Jersy: John Wiley & Sons, Inc.,.

Logan, Bruce, E., Regan & John, M., 2006. MICROBIA Challenges and..... *Environmental Science and Technology,* pp. 5172-5180.

Modeling, I. T. G. o. B., 2004. *Mathematical Modeling of Biofilms.* s.l.:s.n.

Mathworks, 2010, *Matlab Computer Program*.

Oh & Tack, S., 2010. *Trend of Mathematical Models in Microbial fuel Cell for Environmental Refinery from Waste/Water.* s.l., Springler, pp. 25-30.

Picioreanu, C., Head, I. M., Katuri, K. P. & vanLoosdrecht, M. C., 2007. A Computational Model for Biofilm-Based Microbial Fuel Cells. *Water Research,* pp. 2921-2940.

Picioreanu, C. et al., 2008. Mathematical Model for Microbial Fuel Cells with Anodic Biofilms and Anaerobic Digestion. *WAter Science and Technology,* pp. 965-971.

Picioreanu, C. et al., 2009. Modelling Microbial Fuel Cells with Suspended Cells and added Electron Transfer Mediator.

Pinto, R., Srinivasan, B., Manuel, M. & Tartakovsk, B., 2010. A Two-Population Bio-Electrochemical Model of a Microbial Fuel Cell. *Bioresource Technology,* pp. 5256-5265.

Rittmann, B. E. & McCarty, P. L., 1980. Model of Steady-State-Biofilm Kinetics. *Biotechnology and Bioenginering,* pp. 2343-2357.

Rose, R., 2006. *Fuel Cells.* [Online]
Available at: http://www.fuelcells.org/base.cgim?template=types_of_fuel_cells
[Accessed 2 2 2014].

Schroder & Uwe, 2007. Anodic electron transfer mechanisms in microbial fuel cells and their energy efficiency. *Alternaive Fuel Tehnologies,* pp. 3-9.

Spiegel, C., 2008. *PEM Fuel Cell Modeling and Simulation using MATLAB.* Burlington: Elseveir.

Wanner, O. & Morgenroth, E., 2004. Biofilm modeling with AQUASIM. *Water Science and Technology,* 49(11–12), p. 137–144.

Xavier, J. B., Picioreanu, C. & Loosdrecht, v. C., 2005. A framework for multidimensional modelling of activity and structure of multispecies biofilms. *Environmental Microbiology,* pp. 1085-1103.

Zeng, Y., Choo, Y. F., Kim, B.-H. & Wu, P., 2010. Modelling and Simulation of Two-Chamber Microbial Fuel Cell. *Journal of Power Sources,* pp. 79-89.

Appendix A

Faraday's laws

- Faraday's 1st Law of Electrolysis - The mass of a substance altered at an electrode during electrolysis is directly proportional to the quantity of electricity transferred at that electrode. Quantity of electricity refers to the quantity of electrical charge, typically measured in coulomb.
- Faraday's 2nd Law of Electrolysis - For a given quantity of D.C electricity (electric charge), the mass of an elemental material altered at an electrode is directly proportional to the element's equivalent weight. The equivalent weight of a substance is equal to its molar mass divided by the change in oxidation state it undergoes upon electrolysis (often equal to its charge or valence).

Mathematical form,

$$m = \left(\frac{Q}{F}\right)\left(\frac{M}{z}\right)$$

Where;

m = mass of the substance liberated at an electrode (g)

Q = total electric charge passed through the substance

F = Faraday constant (96485 C mol^{-1})

M = molar mass of the substance

z = valency number of ions of the substance (electrons transferred per ion)

Note that M/z is the same as the equivalent weight of the substance altered.

For Faraday's first law; M, F, and z are constants, so that the larger the value of Q the larger m will be.

For Faraday's second law; Q, F, and z are constants, so that the larger the value of M/z (equivalent weight) the larger m will be.

In the simple case of constant-current electrolysis, $Q = It$ leading to,

$$m = \left(\frac{It}{F}\right)\left(\frac{M}{z}\right)$$

and then to;

$n = \left(\frac{It}{F}\right)\left(\frac{1}{z}\right)$

Where;

n is the amount of substance ("number of moles") liberated: $n = m/M$

t is the total time the constant current was applied.

(http://en.wikipedia.org/wiki/Faraday%27s_laws_of_electrolysis)

Appendix B

Manual

Table of Contents

List of Figures

List of Tables

1. Introduction

This package is provided with several graphical user interfaces and one .pdf document, in order to make the model more user friendly. With the given package it is easy to visualize model equations and model parameters which are given in nice visual forms.

Initialization of the model before solving the model equations can be done by the user, with the appropriate initialization user interface. A simulation time also can be given by the user according to the requirement.

Solving of the model equations and visualization of results in the more convenient way that is in the graphical form is available with another graphical user interface.

The package was constructed using MATLAB but it is not required to be familiar with MATLAB, to solve the model with the provided package.

2. Main GUI (Graphical User Interface)

After installing the given package, the graphical user interface – Main GUI (Figure 1) can be opened. It contains four push buttons, linked to other graphical user interfaces (gui s).

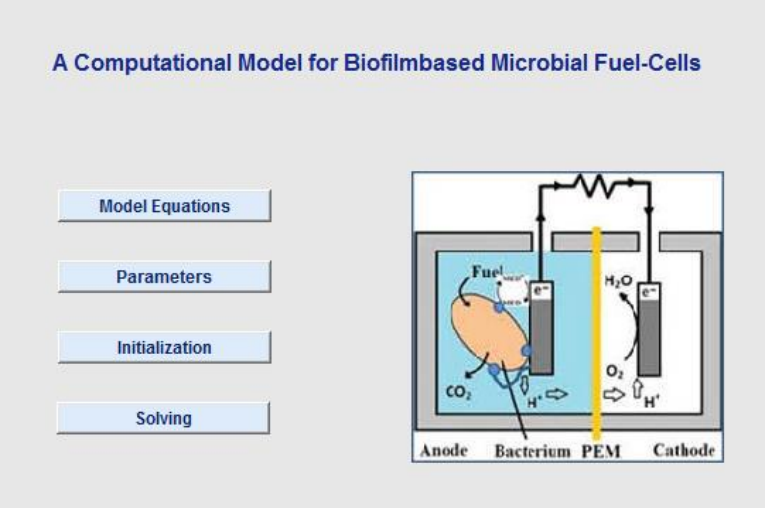
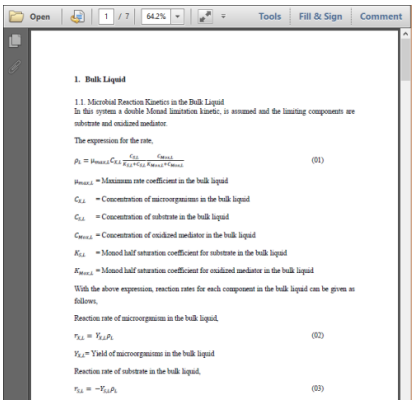


Figure 1: Main Graphical User Interface

2.1. Model Equations (.pdf Document)

Press the push uppermost which displays a .pdf document that contains all the model equations. A part of the document is displayed in Figure 2.



1. Bulk Liquid

1.1. Microbial Reaction Kinetics in the Bulk Liquid
In this system a double Monad limitation kinetic, is assumed and the limiting components are substrate and oxidized mediator.

The expression for the rate,

$$\rho_L = \mu_{max,L} C_{X,L} \frac{C_{S,L}}{K_{S,L}+C_{S,L}} \frac{C_{Mox,L}}{K_{Mox,L}+C_{Mox,L}} \quad (01)$$

$\mu_{max,L}$ = Maximum rate coefficient in the bulk liquid

$C_{X,L}$ = Concentration of microorganisms in the bulk liquid

$C_{S,L}$ = Concentration of substrate in the bulk liquid

$C_{Mox,L}$ = Concentration of oxidized mediator in the bulk liquid

$K_{S,L}$ = Monod half saturation coefficient for substrate in the bulk liquid

$K_{Mox,L}$ = Monod half saturation coefficient for oxidized mediator in the bulk liquid

With the above expression, reaction rates for each component in the bulk liquid can be given as follows,

Reaction rate of microorganism in the bulk liquid,

$$r_{X,L} = Y_{X,L}\rho_L \quad (02)$$

$Y_{X,L}$= Yield of microorganisms in the bulk liquid

Reaction rate of substrate in the bulk liquid,

$$r_{S,L} = -Y_{S,L}\rho_L \quad (03)$$

Figure 2: Part of the .pdf Document

2.2. Parameters GUI

Press the second push button displays a graphical user interface (gui) that is named as 'Parameters'. This gui displays the model parameters (Figure 3); reaction rate constants, diffusivity coefficients, surface area of the electrode surface and volume of the reactor.

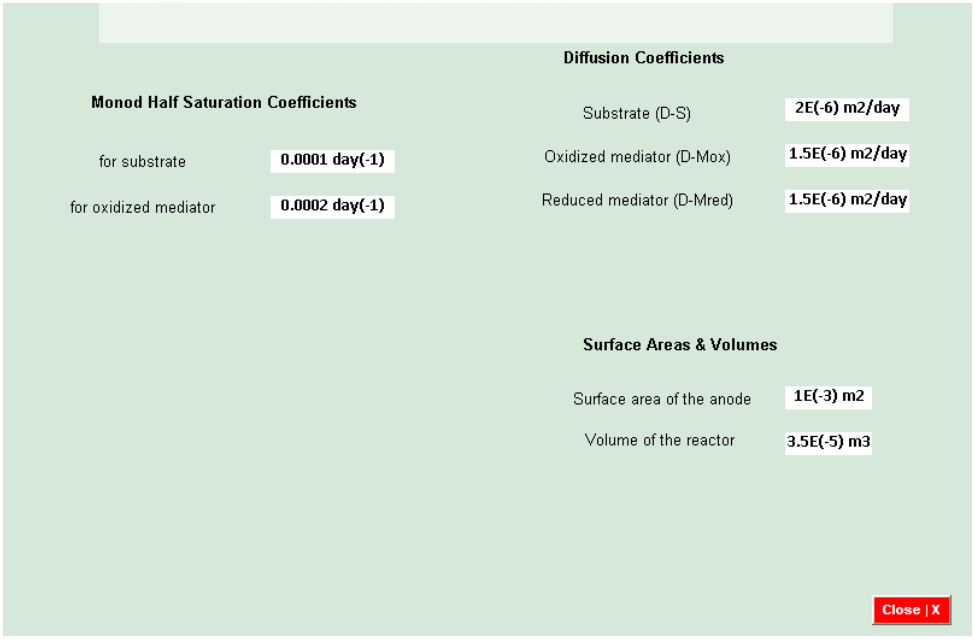


Figure 3: Parameters GUI

2.3. Initialization GUI

Press the third push button displays a gui that is named as 'Initialization' (Figure 4). With this gui a user can initialize the model by changing some of the initial conditions and model parameters.

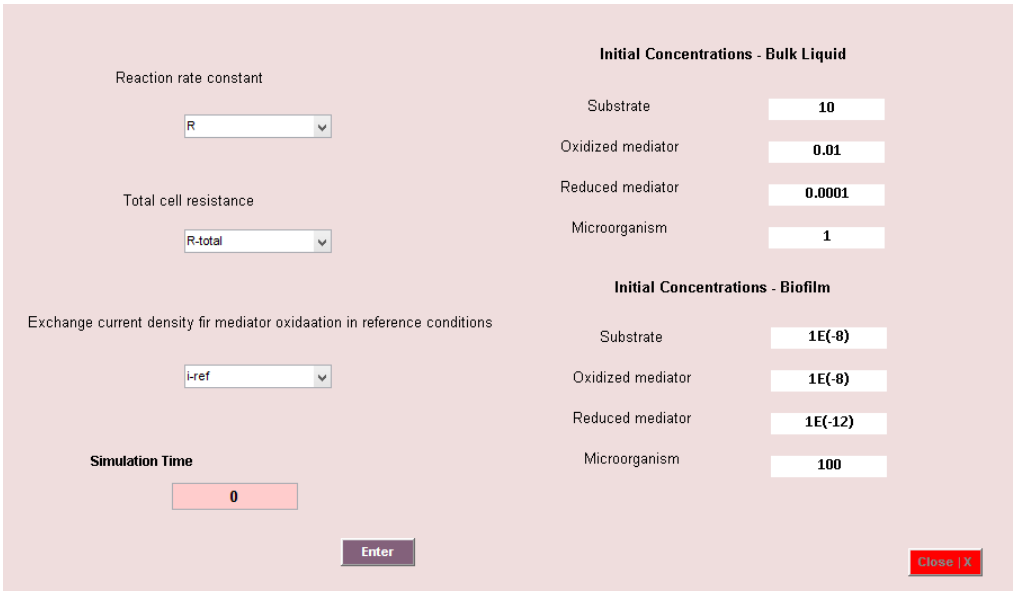


Figure 4: Initialization GUI

2.4. Solving GUI

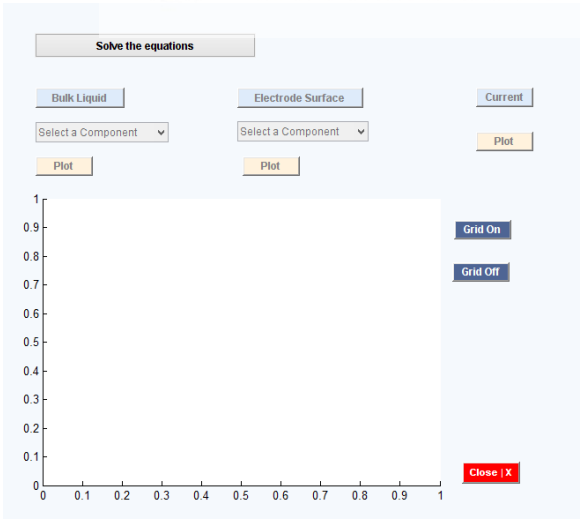


Figure 5: Solving GUI

Press the last push button of the main gui, displays a new gui, is named 'Solving' (Figure 5). In this gui, two tasks can be accomplished one after the other. They are;

solving of the model equations and visualizing of the simulated results in the graphical form.

3. Initialization of the Model

This gui facilitates for users to change some important model parameters.

3.1. Reaction Rate Coefficient

Reaction rates play a major role in the simulation hence reaction rate constants also. Therefore with this GUI a selected reaction rate constant is allowed to be changed by selecting provided values via a popupmenu. See Figure 6.

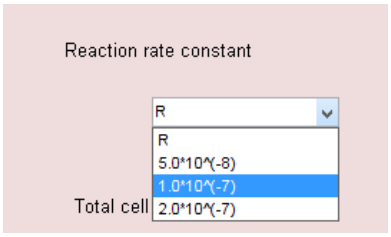


Figure 6: Selection of reaction rate coefficient

3.2. Total Cell Resistance

In order to analyze the current under different cell resistant values, the value of total cell resistance should be changed and the facility is provided with this gui. One of the values from the popup menu can be selected based on interest of the user (Figure 7). The popup menu provides four different values.

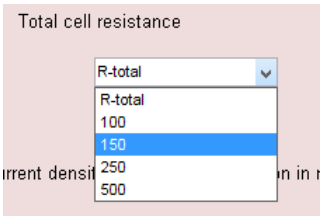


Figure 7: Selection of total cell resistance

3.3. Exchange Current Density at Reference Conditions

The exchange current density can be assigned with two different values according to the requirement of the user via the provided popup menu (Figure 8).

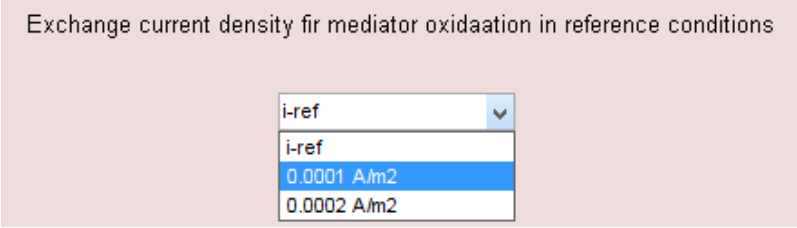


Figure 8: Selection of exchange current density at reference conditions

3.4. Simulation Time

Simulation time can be given by typing a value in the indicated text box and it should be given in seconds (no need to put the units, only the value). See the indicated example in Figure 9.

Eg: simulation time is 7200 s

Figure 9: Enter a simulation time

Note: If the user is not interested in initializing the model, still the system can be solved and it uses default values as given below. But to initialize the model with these default values, after giving a simulation type the 'Enter' button (Figure 9) is needed to be pressed.

Then the 'Close' button is active for closing the initialization GUI.

Table 2: List of Default Values

Parameter	Default Value
reaction rate constant	5.0 *10^(-8)
total cell resistance	100 Ω
exchange current density at reference conditions	0.0001 A/m^2
simulation time	3600 s

4. Solving the Equations

Open the ‘solving’ gui by pressing forth push button of the main gui. Then press the push button named ‘Solving Equations’ (Figure 10) in order to start the solving process. If the given simulation time is considerably high, solving takes a countable time to give the simulated results. By the time the solution is complete, the push buttons named ‘Bulk Liquid’, ‘Electrode Surface’ and ‘Current’ are active (Figure 11).

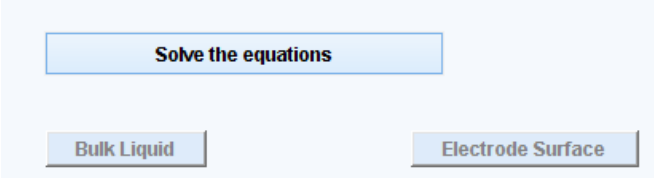


Figure 10: Press the push button 'Solve the equations'

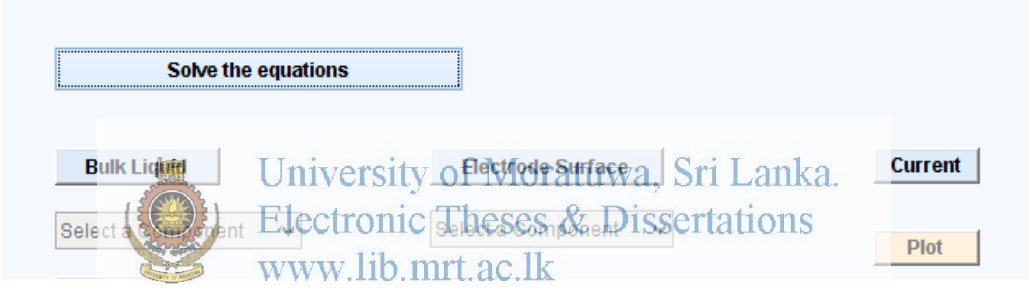


Figure 11: Solving of Equations is complete - Push buttons are active

5. View Results

With the same gui, simulated results also can be seen in graphical forms. This user interface provides the capability of plotting concentrations of different components in the bulk liquid and the concentrations of mediators at the electrode surface. The variation of current with time can also be obtained.

5.1. Variations of Concentrations of Components in the Bulk-Liquid

Press the push button named ‘Bulk Liquid’ activates the relevant popupmenu (Figure 12). Then select a component from the list (Figure 13) in order to see its variation with time in the bulk liquid.

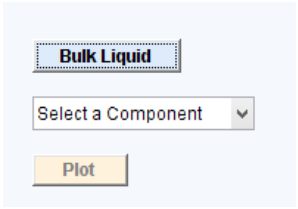


Figure 12: Press 'Bulk Liquid' push button - activate the below popup menu

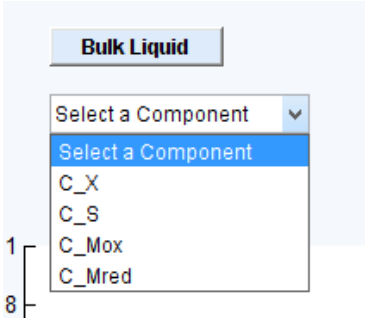


Figure 13: Selection of a component form the menu - bulk liquid

The components are given in a short form and the detailed description is given below as;

C-X - concentration of microorganism

C-S - concentration of substrate

C-Mox - concentration of oxidized mediator

C-Mred - concentration of reduced mediator

Correct selections will activate the ‘Plot’ button (Figure 14) and pressing it will give the requested graph.

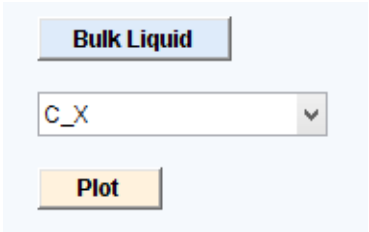


Figure 14: 'Plot' button is active - bulk liquid

5.2. Variations of Concentrations of Mediators at the Electrode Surface

Press the push button named ‘Electrode Surface’ activates the relevant popupmenu (Figure 15). Then select a component from the list (Figure 16) in order to see its variation with time in the bulk liquid.

Only variation of concentrations of oxidized mediator and reduced mediator with time are available.

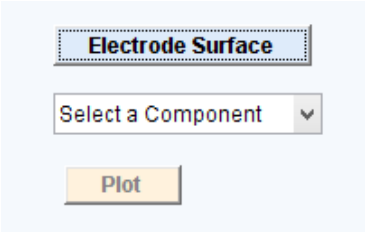


Figure 15: Press the 'Electrode Surface' push button - activate the below popupmenu

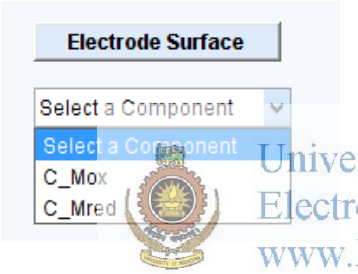


Figure 16: Selection of a component form the menu - electrode surface

Proper selection of the component activates the ‘Plot’ button. Press the button and get the graph of the selected component (Figure 17).

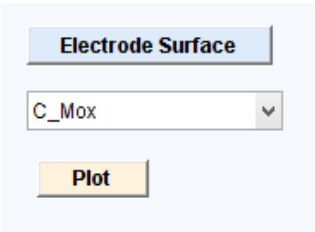


Figure 17: Plot' button is active - electrode surface

5.3. Variation of Current

Press the push button named ‘Current’ activating the ‘Plot’ button (Figure 18). Then press the activated ‘Plot’ button and it gives the graph of variation of simulated current with time (Figure 19).

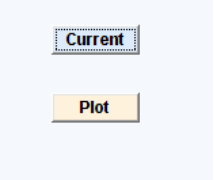


Figure 18: Press 'Current' push button - activate 'Plot' push button

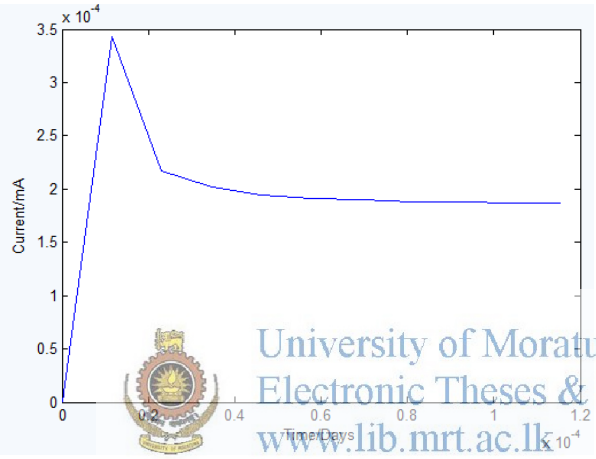


Figure 19: Variation of Current with time

Note:

- In order to activate the ‘Plot’ button relevant to ‘Bulk Liquid’ and ‘Electrode Surface’, a component from the appropriate popup menu should be selected. The selection of the first option will not activate the ‘Plot’ button. Instead it displays a message with the necessary instructions (Figure 20).

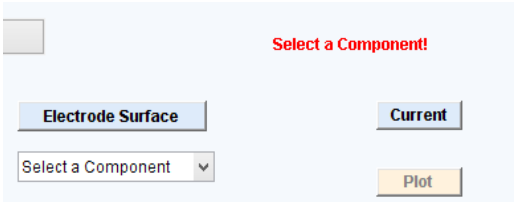


Figure 20: A message with the necessary instructions

- The graphs and variations of selected variables can be further examined using grid lines on the graph. Using the indicated push buttons named ‘Grid On’ and ‘Grid Off’, this can be done (Figure 21). The ‘Grid On’ push button adds grid lines to the graph (Figure 22) and the ‘Grid Off’ push button removes grid lines from the graph.

Figure 21: Push buttons - 'Grid On', 'Grid Off'

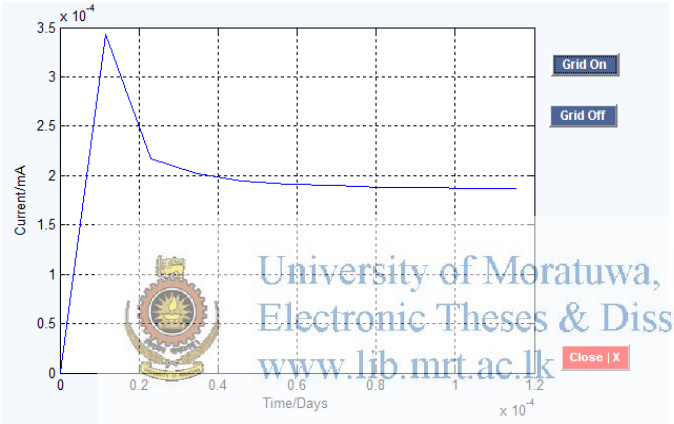


Figure 22: Plot with grid lines on it

Appendix C

Program Files

Current_m

```
%Recall the required globalized variables
global C_L_initial;
global C_B_initial;
global current_density;
%
%Introduce matrices to be used to store data
global M_current;
M_current = [10,1,current_density(1,1)];
global M_current_final;
M_current_final = [10,1,current_density(1,1)];
global M_ode;
M_ode =
[10,1,C_L_initial(1,1),C_L_initial(1,2),C_L_initial(1,3),C_L_initial
(1,4)];
global M_X_pde;
M_X_pde =
[10,1,C_B_initial(1,1),C_B_initial(1,1),C_B_initial(1,1),C_B_initial
(1,1),C_B_initial(1,1),C_B_initial(1,1),C_B_initial(1,1),C_B_initial
(1,1),C_B_initial(1,1),C_B_initial(1,1),C_L_initial(1,1)];
global M_S_pde;
M_S_pde =
[10,1,C_B_initial(1,2),C_B_initial(1,2),C_B_initial(1,2),C_B_initial
(1,2),C_B_initial(1,2),C_B_initial(1,2),C_B_initial(1,2),C_B_initial
(1,2),C_B_initial(1,2),C_B_initial(1,2),C_L_initial(1,2)];
global M_Mox_pde;
M_Mox_pde =
[10,1,C_B_initial(1,3),C_B_initial(1,3),C_B_initial(1,3),C_B_initial
(1,3),C_B_initial(1,3),C_B_initial(1,3),C_B_initial(1,3),C_B_initial
(1,3),C_B_initial(1,3),C_B_initial(1,3),C_L_initial(1,3)];
global M_Mred_pde;
M_Mred_pde =
[10,1,C_B_initial(1,4),C_B_initial(1,4),C_B_initial(1,4),C_B_initial
(1,4),C_B_initial(1,4),C_B_initial(1,4),C_B_initial(1,4),C_B_initial
(1,4),C_B_initial(1,4),C_B_initial(1,4),C_L_initial(1,4)];
global M_Melec_pde;
M_Melec_pde = [10,1,C_B_initial(1,3),C_B_initial(1,4)];


global loop_conditions;

for tt=[0:loop_conditions(1,2):loop_conditions(1,1)]

dt = loop_conditions(1,2);

tspan = [tt tt+dt];
t=linspace(tt,tt+dt,3);
%Introduce a global matrix to get the middle time of the existing
evaluation
global existing_time
existing_time =t(1,1);
```

```
%Define a final value for x;
b = 1*10^(-6)+(tt*10^(-12));
x = linspace(0,b,11);

%Introduce a tolarance limit for the while loop
tol = 0.0000001;

%Read the current value from the matrix
%solution of current
i_a = M_current_final(end,3);
%Define an initial value for current density

%Concentrations at the electrode surface of mediators
%Readout the required elements from the last row of the appropriate
matrix
%solution of pdepemfc
C_E = M_Melec_pde(end,3:4);

%Calculate the concentration ratio using read out data
C_C_E = (((C_E(1)/C_E(2))^2)^0.5);

current_density_i = (0.0002/C_C_E)*(exp((2.303/0.18)*(0.623-i_a*0.1-
0.03*log(C_C_E)))-exp((-2.303/0.18)*(0.623-i_a*0.1-
0.03*log(C_C_E))));
%
%Use an if loop, if the 2 current densities are approximately same
%(difference<tol) then to avoid the usage of while loop (to bypass
the loop)
i=0;

if abs(i_a-current_density_i)< tol

 options=odeset('RelTol',1e-8,'AbsTol',1e-7);

[C_L_X_new,C_L_S_new,C_L_Mox_new,C_L_Mred_new]
=ode15smfc_n0(i,tspan,tt,options);
[C_B_X_new,C_B_S_new,C_B_Mox_new,C_B_Mred_new,C_E_Mox_new,C_E_Mred_n
ew]=pdepemfc_n0(i,t,x);

%Getting the results at the end of each step of the while loop
[tt,i,C_L_X_new,C_L_S_new,C_L_Mox_new,C_L_Mred_new,C_B_X_new,C_B_S_n
ew,C_B_Mox_new,C_B_Mred_new,C_E_Mox_new,C_E_Mred_new,current_density
_i];
else

while abs(i_a-current_density_i) > tol

i_a=current_density_i;
current_density_i = (0.0002/C_C_E)*(exp((2.303/0.18)*(0.623-i_a*0.1-
0.03*log(C_C_E)))-exp((-2.303/0.18)*(0.623-i_a*0.1-
0.03*log(C_C_E))));
i=i+1;

%Generate the row with new results,
row_new = [tt,i,current_density_i];
```

```
%Add the new row in to the existing matrix,
M_current = cat(1,M_current,row_new);

options=odeset('RelTol',1e-8,'AbsTol',1e-7);
[C_L_X_new,C_L_S_new,C_L_Mox_new,C_L_Mred_new]
=ode15smfc_n0(i,tspan,tt,options);
[C_B_X_new,C_B_S_new,C_B_Mox_new,C_B_Mred_new,C_E_Mox_new,C_E_Mred_n
ew]=pdepemfc_n0(i,t,x);

%Getting the results at the end of each step of the while ioop
[tt,i,C_L_X_new,C_L_S_new,C_L_Mox_new,C_L_Mred_new,C_B_X_new,C_B_S_n
ew,C_B_Mox_new,C_B_Mred_new,C_E_Mox_new,C_E_Mred_new,current_density
_i];

end
end


%Writing the finalized results to the appropriate matrices
    %Generate the row with new results,
    row_final_ode =
[tt,i,C_L_X_new,C_L_S_new,C_L_Mox_new,C_L_Mred_new];
    %Add the new row in to the existing matrix,
    M_ode = cat(1,M_ode,row_final_ode);
    %
    %Generate the row with new results,
    row_final_pde_X = [tt,i,C_B_X_new];
    %Add the new row in to the existing matrix,
    M_X_pde = cat(1,M_X_pde,row_final_pde_X);
    %
    %Generate the row with new results,
    row_final_pde_S = [tt,i,C_B_S_new];
    %Add the new row in to the existing matrix,
    M_S_pde = cat(1,M_S_pde,row_final_pde_S);
    %
    %Generate the row with new results,
    row_final_pde_Mox = [tt,i,C_B_Mox_new];
    %Add the new row in to the existing matrix,
    M_Mox_pde = cat(1,M_Mox_pde,row_final_pde_Mox);
    %
    %Generate the row with new results,
    row_final_pde_Mred = [tt,i,C_B_Mred_new];
    %Add the new row in to the existing matrix,
    M_Mred_pde = cat(1,M_Mred_pde,row_final_pde_Mred);
    %
    %Generate the row with new results,
    row_final_pde_Melec = [tt,i,C_E_Mox_new,C_E_Mred_new];
    %Add the new row in to the existing matrix,
    M_Melec_pde= cat(1,M_Melec_pde,row_final_pde_Melec);
    %
    %Generate the row with new results,
    row_final_current = [tt,i,current_density_i];
    %Add the new row in to the existing matrix,
    M_current_final = cat(1,M_current_final,row_final_current);
end
```

Ode_m

```
function [C_L_X_new,C_L_S_new,C_L_Mox_new,C_L_Mred_new]
=ode15smfc_n0(i,tspan,tt,options)

tspan;
tt;


global M_ode
%solution of ode15s
C_L_0 = M_ode(end,3:6);

[T,C_L] = ode15s(@mfc_n0,tspan,C_L_0,options);

%Concentrations of components in the bulk liquid at the end of
simulated
%time step
C_L_X_new=C_L(end,1)     ;
C_L_S_new=C_L(end,2)     ;
C_L_Mox_new=C_L(end,3)   ;
C_L_Mred_new=C_L(end,4)     ;


function dC_L = mfc_n0(t,C_L)
dC_L = zeros(4,1);     % a column vector

%Concentrations in the biofilm
%Read the appropriate matrices
global M_X_pde;
global M_S_pde;
global M_Mox_pde;
global M_Mred_pde;



%Get the average concentration values from the matrices
C_B(1) =mean(M_X_pde(end,3:13));
C_B(2) =mean(M_S_pde(end,3:13));
C_B(3) =mean(M_Mox_pde(end,3:13));
C_B(4) =mean(M_Mred_pde(end,3:13));

%Monad constants in bulk liquid
K_S_L      = 2*10^-4    ;    %substrate
K_Mox_L    = 1*10^-4    ;    %oxidized mediator

%Monad constants in biofilm
K_S_B      = 2*10^-4    ;    %substrate
K_Mox_B    = 1*10^-4    ;    %oxidized mediator

%Maximum specific rate constant (for microbial reduction of
mediator)
global rate_constants;
K_1 = rate_constants(1,1);
```

```
%Yield coefficients in bulk liquid
Y_X_L = 0.12 ;     %biomass
Y_S_L = 1;         %substrate

%Yield coefficients in biofilm
Y_S_B = 1;         %substrate

%Required constants
n = 2                ;    %for thionine
F = 96485.34         ;    %Faraday constant

%Area and Volume values
A_E = 0.001 ;            %surface area of electrode
V_B = 0.000000005          ;    %volume of biofilm
V_L = 0.000035              ;    %volume of bulk liquid

%Required ratios
A_V = A_E/V_L          ; %electrode area to bulk volume ratio
V_V = V_B/V_L          ; %biofilm volume to bulk volume ratio

%Reaction rates in bulk liquid
rho = K_1*C_L(1)*(C_L(2)/(K_S_L+C_L(2)))*(C_L(3)/(K_Mox_L+C_L(3)));
r1_L = Y_X_L*rho       ; %biomass rate
r2_L = -Y_S_L*rho      ; %substrate rate
r3_L = -0.032*rho            ; %Mox rate
r4_L = 0.032*rho             ; %Mred rate
%Reaction rates in biofilm
beta = K_1*C_B(1)*(C_B(2)/(K_S_B+C_B(2)))*(C_B(3)/(K_Mox_B+C_B(3)));
r2_B = -Y_S_B*beta       ; %substrate rate
r3_B = -0.032*beta       ; %Mox rate
r4_B = 0.032*beta        ; %Mred rate

%Reaction rates at the electrode surface
%Read the last row of the appropriate matrix
%solution of current
global M_current
i_rate = M_current(end,3);
gama = i_rate /(n*F) ;
r3_E = gama            ; %Mox rate
r4_E = -gama             ; %Mred rate

dC_L(1) = r1_L;
dC_L(2) = r2_L+V_V*r2_B;
dC_L(3) = r3_L+V_V*r3_B+2500*A_V*r3_E;
dC_L(4) = r4_L+V_V*r4_B+2500*A_V*r4_E;
```

Pde_m

```
function
[C_B_X_new,C_B_S_new,C_B_Mox_new,C_B_Mred_new,C_E_Mox_new,C_E_Mred_n
ew]=pdepemfc_n0(i,t,x)

m = 0;
%x = linspace(0,b,11);
%t = linspace(0,5,10);
x;
t;


sol = pdepe(m,@pdepemfc_n0pde,@pdepemfc_n0ic,@pdepemfc_n0bc,x,t);
% Extract the solution components as u1,u2,u3 and u4.
u1 = sol(:,:,1);
u2 = sol(:,:,2);
u3 = sol(:,:,3);
u4 = sol(:,:,4);

%Resultant concentrations
%Concentrations of components over the biofilm at the end of
computed time
C_B_X_new = u1(end,:) ;
C_B_S_new = u2(end,:) ;
C_B_Mox_new = u3(end,:)    ;
C_B_Mred_new = u4(end,:)   ;

%Concentrations at the electode surface of mediators
C_E_Mox_new = u3(end,1);
C_E_Mred_new = u4(end,1);

 % ---------------------------------------------------------
function [c,f,s] = pdepemfc_n0pde(x,t,u,DuDx)

%Rate constants
K_S = 2*10^-4;
K_MOX = 1*10^-4;

%Yield coefficients in bulk liquid
Y_X_B          = 0.12;        % regarding to biomass
Y_S_B          = 1;           % regarding to substrate

% Diffusion coefficients
global diffusivity
Ds    = diffusivity(1,1);      % For substrate
DMox    = diffusivity(1,2);    % For Mox
DMred    = diffusivity(1,3);   % For Mred

%Maximum specific rate constant (for microbial reduction of
mediator)
global rate_constants;
K_1 = rate_constants(1,2);
K_2 = rate_constants(1,3);
```

```
%Displacement rate of microorganisms in the biofilm
UF    =0;
%rate coefficients
beta_1 = K_1*u(1)*(u(2)/(K_S+u(2)))*(u(3)/(K_MOX+u(3)));
beta_2 = K_2*u(1)*(u(2)/(K_S+u(2)))*(u(3)/(K_MOX+u(3)));

r1 = Y_X_B*beta_1;
r2 = -Y_S_B*beta_2;
r3 = -0.032*beta_2;
r4 = 0.032*beta_2;

c = [1; 1; 1; 1];
f = [UF; Ds.*DuDx(2); DMox.*DuDx(3); DMred.*DuDx(4)];
s = [r1; r2; r3; r4];
% --------------------------------------------------------------
function u0 = pdepemfc_n0ic(x)

%Writing an if loop to give initial values for components according
to x values
%
%Get the previous final values from the appropriate matrices
global M_X_pde
%Then averaged the values based on x-coordinates in order to feed
into the if loop from the last row of the matrix
X_mat=M_X_pde (end,:);
X_Matrix=[X_mat(3),X_mat(4),X_mat(5),X_mat(6),X_mat(7),X_mat(8),X_ma
t(9),X_mat(10),X_mat(11),X_mat(12),X_mat(13)];
global M_S_pde
S_mat=M_S_pde (end,:);
%tt=[M_S_pde(end,1),M_S_pde(end,2)];
S_Matrix=[S_mat(3),S_mat(4),S_mat(5),S_mat(6),S_mat(7),S_mat(8),S_ma
t(9),S_mat(10),S_mat(11),S_mat(12),S_mat(13)];
global M_Mox_pde
Mox_mat=M_Mox_pde (end,:);
Mox_Matrix=[Mox_mat(3),Mox_mat(4),Mox_mat(5),Mox_mat(6),Mox_mat(7),M
ox_mat(8),Mox_mat(9),Mox_mat(10),Mox_mat(11),Mox_mat(12),Mox_mat(13)
];
global M_Mred_pde
Mred_mat=M_Mred_pde (end,:);
Mred_Matrix=[Mred_mat(3),Mred_mat(4),Mred_mat(5),Mred_mat(6),Mred_ma
t(7),Mred_mat(8),Mred_mat(9),Mred_mat(10),Mred_mat(11),Mred_mat(12),
Mred_mat(13)];

%Recall the evaluation time and the bulk liquid concentration to get
the required film conditions
global existing_time; global M_ode;
b = 1*10^(-6)+(existing_time*10^(-12));
film_thick = linspace(0,b,11);

if x == 0
 CoX = X_Matrix(1,1);
 CoS = S_Matrix(1,1);
 CoMox = Mox_Matrix(1,1);
 CoMred = Mred_Matrix(1,1);
elseif x == film_thick(1,2)
 CoX = X_Matrix(1,2);
 CoS = S_Matrix(1,2);
```

```
 CoMox = Mox_Matrix(1,2);
 CoMred = Mred_Matrix(1,2);
elseif x == film_thick(1,3)
 CoX = X_Matrix(1,3);
 CoS = S_Matrix(1,3);
 CoMox = Mox_Matrix(1,3);
 CoMred = Mred_Matrix(1,3);
elseif x == film_thick(1,4)
 CoX = X_Matrix(1,4);
 CoS = S_Matrix(1,4);
 CoMox = Mox_Matrix(1,4);
 CoMred = Mred_Matrix(1,4);

 elseif x == film_thick(1,5)
 CoX = X_Matrix(1,5);
 CoS = S_Matrix(1,5);
 CoMox = Mox_Matrix(1,5);
 CoMred = Mred_Matrix(1,5);

 elseif x == film_thick(1,6)
 CoX = X_Matrix(1,6);
 CoS = S_Matrix(1,6);
 CoMox = Mox_Matrix(1,6);
 CoMred = Mred_Matrix(1,6);

 elseif x == film_thick(1,7)
 CoX = X_Matrix(1,7);
 CoS = S_Matrix(1,7);
 CoMox = Mox_Matrix(1,7);
 CoMred = Mred_Matrix(1,7);

elseif x == film_thick(1,8)
 CoX = X_Matrix(1,8);
 CoS = S_Matrix(1,8);
 CoMox = Mox_Matrix(1,8);
 CoMred = Mred_Matrix(1,8);

 elseif x == film_thick(1,9)
 CoX = X_Matrix(1,9);
 CoS = S_Matrix(1,9);
 CoMox = Mox_Matrix(1,9);
 CoMred = Mred_Matrix(1,9);

 elseif x == film_thick(1,10)
 CoX = X_Matrix(1,10);
 CoS = S_Matrix(1,10);
 CoMox = Mox_Matrix(1,10);
 CoMred = Mred_Matrix(1,10);

 elseif x == film_thick(1,11)
 CoX = X_Matrix(1,11);
 CoS = S_Matrix(1,11);
 CoMox = Mox_Matrix(1,11);
 CoMred = Mred_Matrix(1,11);
end
u0 = [CoX; CoS; CoMox; CoMred];
```

```
% --------------------------------------------------------------------
function [pl,ql,pr,qr] = pdepemfc_n0bc(xl,ul,xr,ur,t)

%Concentrations of the components in the bulk liquid
%Read the last row of the appropriate matrix
global M_ode   %solution of ode15s
C_L = M_ode(end,3:6)  ;

% Required constants
n = 2               ;   %for thionine
F = 96485.34         ;   %Faraday constant

% Reaction rates at the electrode surface
% Read the last row of the appropriate matrix
global M_current   %solution of current
i = M_current(end,3);
r_E_Mox = i/(n*F);
r_E_Mred =-i/(n*F);

pl = [0; 0; r_E_Mox; r_E_Mred];
ql = [1; 1; 1; 1];
pr = [ur(1)-C_L(1); ur(2)-C_L(2); ur(3)-C_L(3); ur(4)-C_L(4)];
qr = [0; 0; 0; 0];
```

GUI_1

```
function varargout = GUI_1(varargin)
% GUI_1 M-file for GUI_1.fig
%      GUI_1, by itself, creates a new GUI_1 or raises the existing
%      singleton*.
%
%      H = GUI_1 returns the handle to a new GUI_1 or the handle to
%      the existing singleton*.
%
%      GUI_1('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in GUI_1.M with the given input
arguments.
%
%      GUI_1('Property','Value',...) creates a new GUI_1 or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GUI_1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_1_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_1

% Last Modified by GUIDE v2.5 26-Oct-2014 14:55:13

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_1_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before GUI_1 is made visible.
function GUI_1_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_1 (see VARARGIN)

% Choose default command line output for GUI_1
handles.output = hObject;


% Update handles structure
guidata(hObject, handles);


% UIWAIT makes GUI_1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = GUI_1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Read the pdf file that contains the equations
open('Equations.pdf')


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Open the appropriate GUI
GUI_2('GUI_1')


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Open the appropriate GUI
```

```
GUI_3('GUI_1')

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Open the appropriate GUI
GUI_4('GUI_1')
```

GUI_2

```
function varargout = GUI_2(varargin)
% GUI_2 M-file for GUI_2.fig
%      GUI_2, by itself, creates a new GUI_2 or raises the existing
%      singleton*.
%
%      H = GUI_2 returns the handle to a new GUI_2 or the handle to
%      the existing singleton*.
%
%      GUI_2('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in GUI_2.M with the given input
arguments.
%
%      GUI_2('Property','Value',...) creates a new GUI_2 or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GUI_2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_2_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_2

% Last Modified by GUIDE v2.5 08-Nov-2014 08:28:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_2_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI_2 is made visible.
function GUI_2_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_2 (see VARARGIN)

% Choose default command line output for GUI_2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI_2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Make necessary arrangements on the initial display of the GUI
%Monod half saturation constants
set(handles.text27,'string','0.0001 day(-1)' );
set(handles.text28,'string','0.0002 day(-1)' );
%Diffusivity values
set(handles.text13,'string','2E(-6) m2/day' );
set(handles.text14,'string','1.5E(-6) m2/day' );
set(handles.text15,'string','1.5E(-6) m2/day' );
%Area & Volume values
set(handles.text31,'string','1E(-3) m2' );
set(handles.text32,'string','3.5E(-5) m3' );
% --- Outputs from this function are returned to the command line.
function varargout = GUI_2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%
%Close the existing GUI
delete(handles.figure1)

function varargout = GUI_2(varargin)
% GUI_2 M-file for GUI_2.fig
%      GUI_2, by itself, creates a new GUI_2 or raises the existing
%      singleton*.
%
%      H = GUI_2 returns the handle to a new GUI_2 or the handle to
%      the existing singleton*.
%
%      GUI_2('CALLBACK',hObject,eventData,handles,...) calls the
local
```

```
%      function named CALLBACK in GUI_2.M with the given input
arguments.
%
%      GUI_2('Property','Value',...) creates a new GUI_2 or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GUI_2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_2_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_2

% Last Modified by GUIDE v2.5 08-Nov-2014 08:28:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_2_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI_2 is made visible.
function GUI_2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_2 (see VARARGIN)

% Choose default command line output for GUI_2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes GUI_2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Make necessary arrangements on the initial display of the GUI
%Monod half saturation constants
set(handles.text27,'string','0.0001 day(-1)' );
set(handles.text28,'string','0.0002 day(-1)' );
%Diffusivity values
set(handles.text13,'string','2E(-6) m2/day' );
set(handles.text14,'string','1.5E(-6) m2/day' );
set(handles.text15,'string','1.5E(-6) m2/day' );
%Area & Volume values
set(handles.text31,'string','1E(-3) m2' );
set(handles.text32,'string','3.5E(-5) m3' );

% --- Outputs from this function are returned to the command line.
function varargout = GUI_2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Close the existing GUI
delete(handles.figure1)
```

GUI_3

```
function varargout = GUI_33(varargin)
% GUI_33 M-file for GUI_33.fig
%      GUI_33, by itself, creates a new GUI_33 or raises the
existing
%      singleton*.
%
%      H = GUI_33 returns the handle to a new GUI_33 or the handle
to
%      the existing singleton*.
%
%      GUI_33('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in GUI_33.M with the given input
arguments.
%
%      GUI_33('Property','Value',...) creates a new GUI_33 or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GUI_33_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_33_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_33

% Last Modified by GUIDE v2.5 30-Dec-2014 10:53:06

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_33_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_33_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before GUI_33 is made visible.
function GUI_33_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_33 (see VARARGIN)

% Choose default command line output for GUI_33
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI_33 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
%Make necessary arrangements on the initial display of the GUI
%Initial concentration of components in the bulk liquid
set(handles.text1,'string','10' );
set(handles.text4,'string','0.01' );
set(handles.text3,'string','0.0001' );
set(handles.text2,'string','1' );
%Initial concentration of components in the biofilm
set(handles.text5,'string','1E(-8)' );
set(handles.text8,'string','1E(-8)' );
set(handles.text7,'string','1E(-12)' );
set(handles.text6,'string','100' );
%
set(handles.popupmenu1,'value',1);
set(handles.popupmenu2,'value',1);
set(handles.popupmenu3,'value',1);
%
set(handles.edit1,'string',num2str(0));
%
set(handles.pushbutton1,'Enable', 'off');


% --- Outputs from this function are returned to the command line.
function varargout = GUI_33_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: contents = get(hObject,'String') returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1
%
%Get the selected component from the popupmenu
value1 = get(handles.popupmenu1,'value');
global reaction_rate_common;
global menu_1;
switch value1
    case 1
     reaction_rate_common=5*10^(-8);
     menu_1=1;
    case 2
     reaction_rate_common=5*10^(-8);
     menu_1=1;
    case 3
     reaction_rate_common=1*10^(-7);
     menu_1=1;
    case 4
     reaction_rate_common=2*10^(-7);
     menu_1=1;
    otherwise
end


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called



% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu2
%
%Get the selected component from the popupmenu
value1 = get(handles.popupmenu2,'value');
global cell_resistance;
```

```
global menu_2;
switch value1
    case 1
    cell_resistance=100;
    menu_2 = 1;
    case 2
    cell_resistance=100;
    menu_2 = 1;
    case 3
    cell_resistance=150;
    menu_2 = 1;
    case 4
    cell_resistance=250;
    menu_2 = 1;
    case 5
    cell_resistance=500;
    menu_2 = 1;
    otherwise
end


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu3
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu3
%
%Get the selected component from the popupmenu
value1 = get(handles.popupmenu3,'value');
global reference_current;
global menu_3;
switch value1
    case 1
    reference_current=0.0001;
    menu_3=1;
```

```
    case 2
    reference_current=0.0001;
    menu_3=1;
    case 3
    reference_current=0.0002;
    menu_3=1;
    otherwise
end
% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Close the existing GUI
delete(handles.figure1)


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Recall required globalized variables
global reaction_rate_common;
global reaction_rate_common_new;
global menu_1;
%
global cell_resistance;
global cell_resistance_new;
global menu_2;
%
global reference_current;
global reference_current_new;
global menu_3;
%
if (menu_1==1)
   reaction_rate_common_new=reaction_rate_common;
else
   reaction_rate_common_new=5*10^(-8);
```

```
end
%
if (menu_2==1)
   cell_resistance_new=cell_resistance;
else
   cell_resistance_new=100;
end
%
if (menu_3==1)
   reference_current_new=reference_current;
else
   reference_current_new=0.0001;
end
set(handles.pushbutton1,'Enable', 'on');


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double
%
%Globalization the value
global simulation_time1
% Validate that the text in the field converts to a real number
simulation_time1 = str2double(get(hObject,'String'));
if isnan(simulation_time1) || ~isreal(simulation_time1)
% isdouble returns NaN for non-numbers and it cannot be complex
% Give the edit text box focus so user can correct the error
uicontrol(hObject)
else
end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

GUI_4

```
function varargout = GUI_4(varargin)
% GUI_4 M-file for GUI_4.fig
%      GUI_4, by itself, creates a new GUI_4 or raises the existing
%      singleton*.
%
%      H = GUI_4 returns the handle to a new GUI_4 or the handle to
%      the existing singleton*.
%
%      GUI_4('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in GUI_4.M with the given input
arguments.
%
%      GUI_4('Property','Value',...) creates a new GUI_4 or raises
the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before GUI_4_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to GUI_4_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUI_4

% Last Modified by GUIDE v2.5 26-Oct-2014 15:23:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUI_4_OpeningFcn, ...
                   'gui_OutputFcn',  @GUI_4_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before GUI_4 is made visible.
function GUI_4_OpeningFcn(hObject, eventdata, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUI_4 (see VARARGIN)

% Choose default command line output for GUI_4
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes GUI_4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

%Make necessary arrangements on the GUI
set(handles.popupmenu1,'value',1);
set(handles.popupmenu1,'Enable', 'off');
set(handles.popupmenu2,'value',1);
set(handles.popupmenu2,'Enable', 'off');
%
set(handles.pushbutton5,'Enable', 'off');
set(handles.pushbutton6,'Enable', 'off');
set(handles.pushbutton7,'Enable', 'off');
set(handles.pushbutton8,'Enable', 'off');
set(handles.pushbutton9,'Enable', 'off');
set(handles.pushbutton10,'Enable', 'off');
% --- Outputs from this function are returned to the command line.
function varargout = GUI_4_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%
%Close the existing GUI
delete(handles.figure1)

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
grid on;

% --- Executes on button press in pushbutton3.
```

```
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
grid off;
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Solving the m-files
current_1
%
set(handles.pushbutton5,'Enable', 'on');
set(handles.pushbutton6,'Enable', 'on');
set(handles.pushbutton7,'Enable', 'on');

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Read out appropriate matrix
global M_ode
%
global M_bulkliquid
M_bulkliquid = M_ode(2:end,:);
%
set(handles.popupmenu1,'value',1);
set(handles.popupmenu1,'Enable', 'on');
set(handles.pushbutton10,'Enable', 'off');
set(handles.popupmenu2,'value',1);
set(handles.popupmenu2,'Enable', 'off');
set(handles.pushbutton9,'Enable', 'off');
set(handles.pushbutton10,'Enable', 'off');

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Read out appropriate matrix
global M_Melec_pde
%
global M_electrode
M_electrode = M_Melec_pde(2:end,:);
%
set(handles.popupmenu2,'value',1);
set(handles.popupmenu2,'Enable', 'on');
set(handles.pushbutton10,'Enable', 'off');
set(handles.popupmenu1,'value',1);
set(handles.popupmenu1,'Enable', 'off');
set(handles.pushbutton8,'Enable', 'off');
```

```
set(handles.pushbutton10,'Enable', 'off');


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Read out appropriate matrix
global M_current_final
%
global M_current
M_current = M_current_final(2:end,:);
%
set(handles.pushbutton10,'Enable', 'on');
set(handles.popupmenu1,'value',1);
set(handles.popupmenu1,'Enable', 'off');
set(handles.popupmenu2,'value',1);
set(handles.popupmenu2,'Enable', 'off');
set(handles.pushbutton8,'Enable', 'off');
set(handles.pushbutton9,'Enable', 'off');

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu1
%
%Get the selected component from the popupmenu
value1 = get(handles.popupmenu1,'value');
global column_bulkliquid;
global bulkliquid_axislabel
switch value1
    case 1
    set(handles.text1,'string','Select a Component!');
    set(handles.pushbutton8,'Enable', 'off');
    case 2
    column_bulkliquid=3;
    bulkliquid_axislabel ='Concentration of Microorganisms/(mol/m3)';
    set(handles.pushbutton8,'Enable', 'on');
    case 3
    column_bulkliquid=4;
    bulkliquid_axislabel ='Concentration of Substrate/(mol/m3)';
    set(handles.pushbutton8,'Enable', 'on');
    case 4
    column_bulkliquid=5;
    bulkliquid_axislabel ='Concentration of Oxidized Mediator/(mol/m3)';
    set(handles.pushbutton8,'Enable', 'on');
    case 5
    column_bulkliquid=6;
    bulkliquid_axislabel ='Concentration of Reduced Mediator/(mol/m3)';
```

```
    set(handles.pushbutton8,'Enable', 'on');
    otherwise
end

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
popupmenu2
%
%Get the selected component from the popupmenu
value1 = get(handles.popupmenu2,'value');
global column_electrode;
global electrode_axislabel
switch value1
    case 1
    set(handles.text1,'string','Select a Component!');
    set(handles.pushbutton9,'Enable', 'off');
    case 2
    column_electrode=3;
    electrode_axislabel ='Concentration of Oxidized Mediator/(mol/m3)';
    set(handles.pushbutton9,'Enable', 'on');
    case 3
    column_electrode=4;
    electrode_axislabel ='Concentration of Reduced Mediator/(mol/m3)';
    set(handles.pushbutton9,'Enable', 'on');
    otherwise
end

% --- Executes during object creation, after setting all properties.
```

```
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on
Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Plot the corressponding graph
global column_bulkliquid;
global M_bulkliquid;
global bulkliquid_axislabel
%Readout the simulated time
time = M_bulkliquid(:,1);
time_new = time/(24*3600);
%Read out the selected component's concentration values from the
matrix
%using the obtained column number
concentrations = M_bulkliquid(:,column_bulkliquid);
%Plot the graph
clear figure
hold off
plot(time_new,concentrations)
%Labeling the axis
xlabel('Time/Days');
ylabel(bulkliquid_axislabel);

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Plot the corressponding graph
global column_electrode;
global M_electrode;
global electrode_axislabel
%Readout the simulated time
time = M_electrode(:,1);
time_new = time/(24*3600);
%Read out the selected component's concentration values from the
matrix
%using the obtained column number
concentrations = M_electrode(:,column_electrode);
%Plot the graph
```

```
clear figure
hold off
plot(time_new,concentrations)
%Labeling the axis
xlabel('Time/Days');
ylabel(electrode_axislabel);


% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%
%Plot the corressponding graph
global M_current;
%Readout the simulated time
time = M_current(:,1);
time_new = time/(24*3600);
%Read out the current values from the matrix
current_density = M_current(:,3);
current = current_density;
%Plot the graph
clear figure
hold off
plot(time_new,current)
%Labeling the axis
xlabel('Time/Days');
ylabel('Current/mA');
```