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).

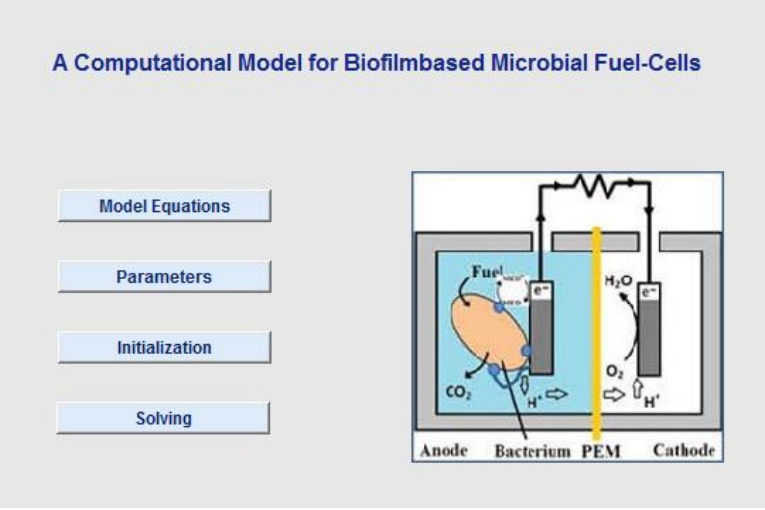
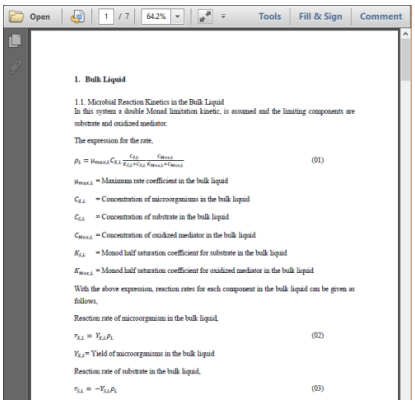


Figure 1: Main Graphical User Interface

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.

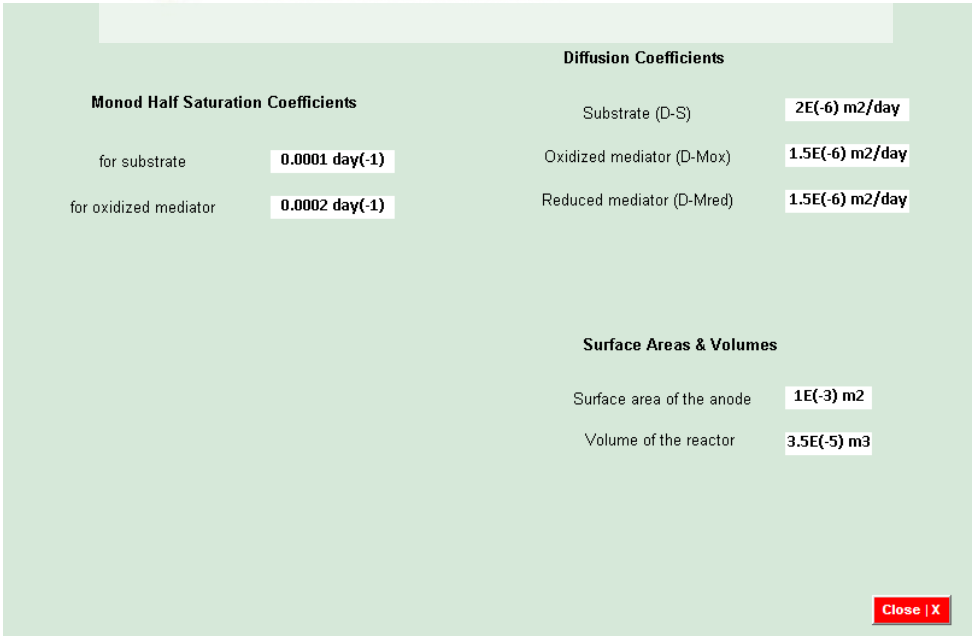


Figure 3: Parameters GUI

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.

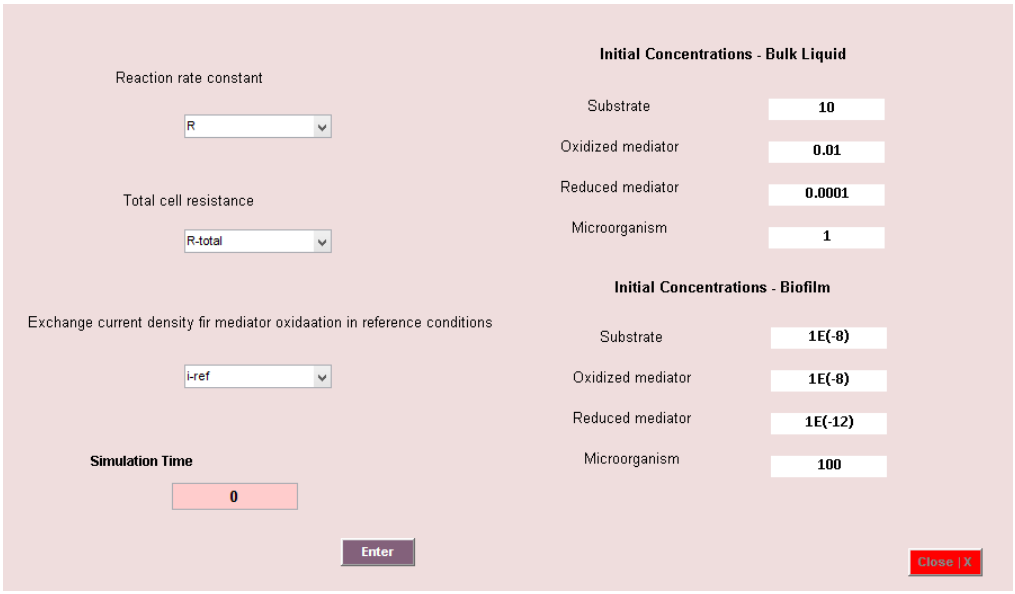


Figure 4: Initialization GUI

2.4. Solving GUI

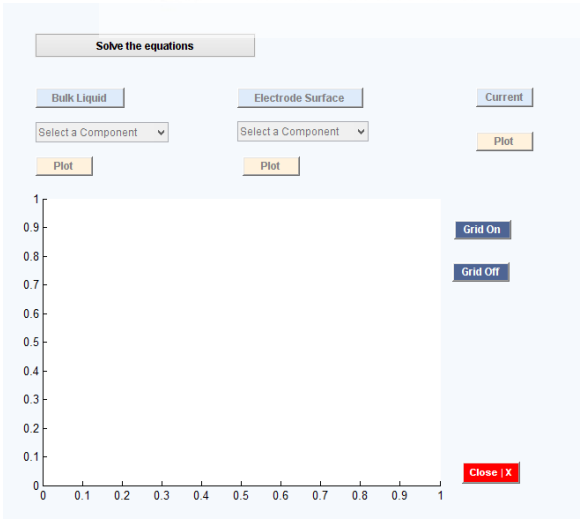


Figure 5: Solving GUI

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.

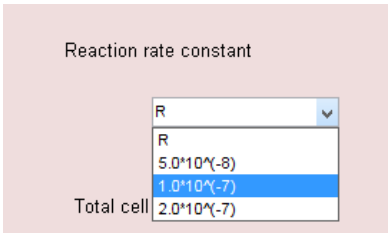


Figure 6: Selection of reaction rate coefficient

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.

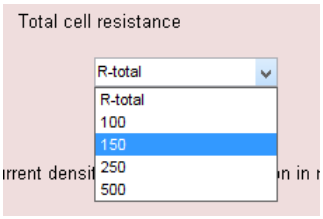


Figure 7: Selection of total cell resistance

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).

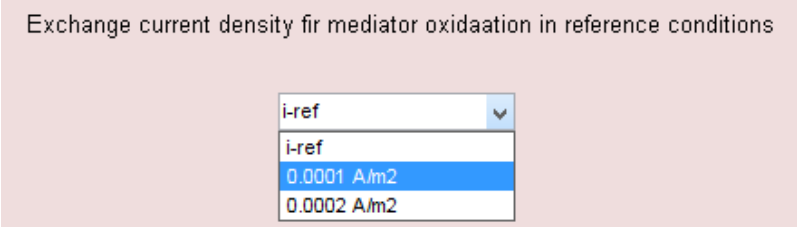


Figure 8: Selection of exchange current density at reference conditions

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).

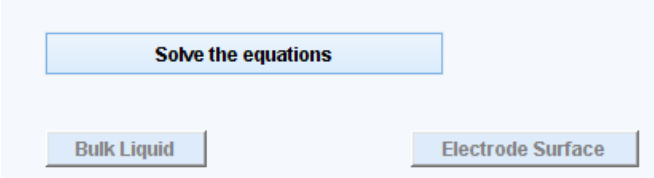


Figure 10: Press the push button 'Solve the equations'

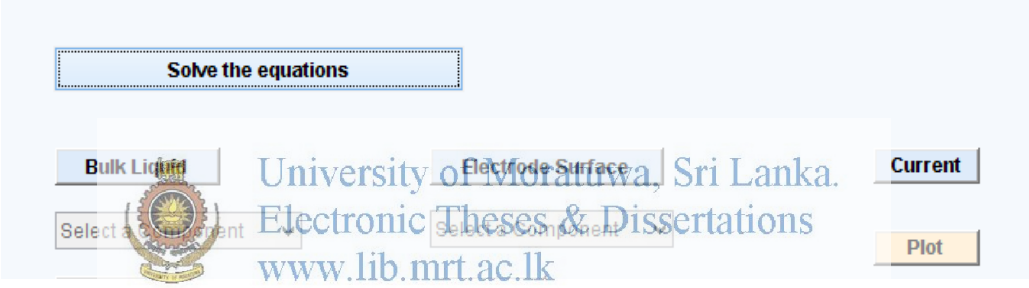


Figure 11: Solving of Equations is complete - Push buttons are active

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.

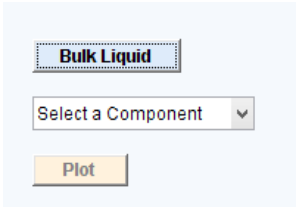


Figure 12: Press 'Bulk Liquid' push button - activate the below popup menu

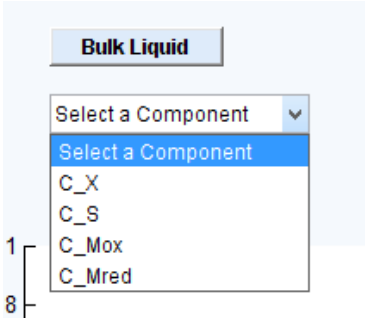


Figure 13: Selection of a component form the menu - bulk liquid

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.

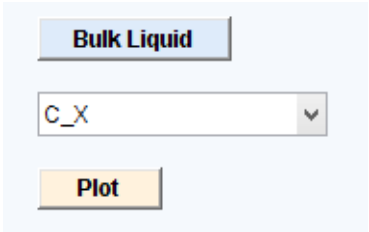


Figure 14: 'Plot' button is active - bulk liquid

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.

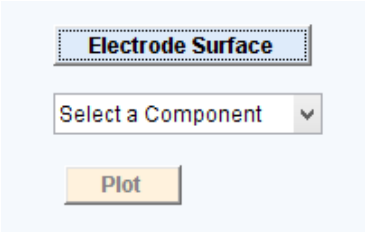


Figure 15: Press the 'Electrode Surface' push button - activate the below popupmenu

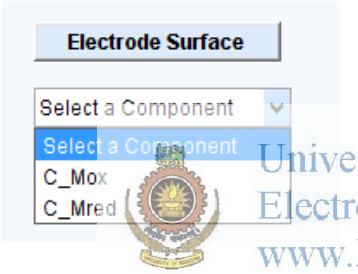


Figure 16: Selection of a component form the menu - electrode surface

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

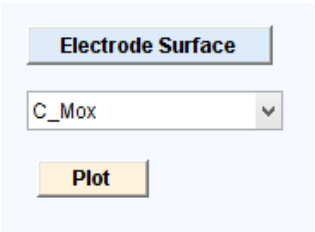


Figure 17: Plot' button is active - electrode surface

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).

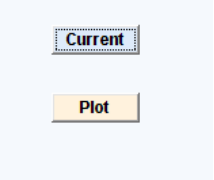


Figure 18: Press 'Current' push button - activate 'Plot' push button

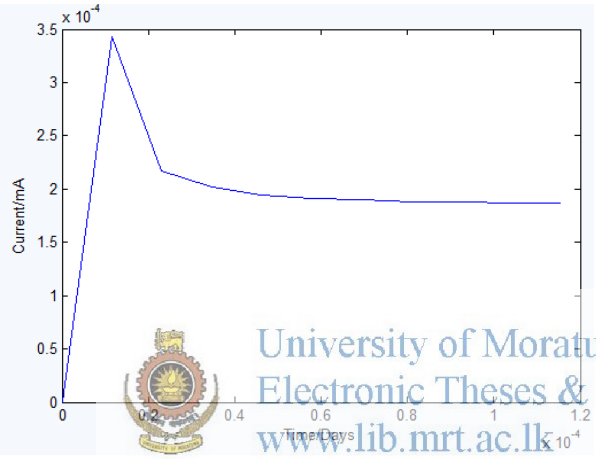


Figure 19: Variation of Current with time

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).

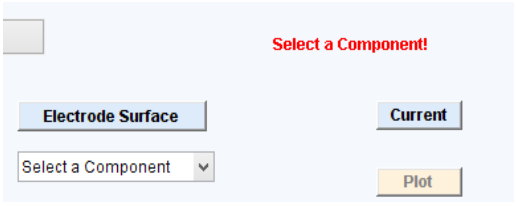


Figure 20: A message with the necessary instructions

- 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'

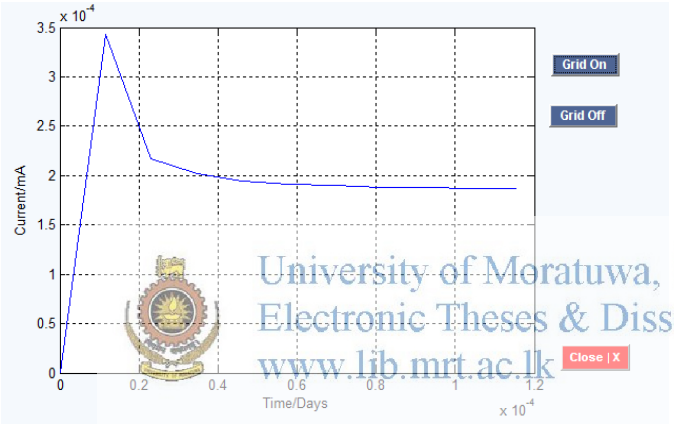


Figure 22: Plot with grid lines on it

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');
```