CellML in PKPD Modelling

1 Contents

1 Contents 1
2 Creating a basic model in OpenCell 2
2.1 Starting a model 2
2.2 Creating a new CellML model 3
2.2.1 Making new components 4
2.2.2 Creating variables 4
2.2.3 Defining units 5
2.2.4 Creating equations 7
2.2.5 Creating connections 8
2.2.6 Validating the model 9
2.2.7 Running a simulation 9
3 Creating a modular PKPD model 11
3.1 Running the Tham 2008 OpenCell session file 11
3.2 Creating the effect compartment in OpenCell 12
3.3 Running the effect compartment model 14
3.4 Using CellML imports to swap compartments 16
4 Conclusion 17

Auckland Bioengineering Institute CellML OpenCell Tutorial

2 Creating a basic model in OpenCell

This part of the tutorial will teach you how to create a simple, functional CellML model from scratch using OpenCell, and how to display the results of simulating this model on a graph.

2.1 Starting a model

We'll begin by defining some equations and parameters that we would like to use CellML to describe. Let's say we want to create a CellML model the simple chemical reaction shown below:

$$A + B \underset{k_b}{\overset{k^f}{\leftrightarrow}} C$$

Using mass action kinetics, we might model this using the following equations:

$$J = k_f[A].[B] - k_b.[C]$$

$$\frac{d[A]}{dt} = -J$$

$$\frac{d[B]}{dt} = -J$$

$$\frac{d[C]}{dt} = J$$

Where *[A]*, *[B]* and *[C]* are the concentrations of the reaction species, k_f and k_b are rate constants and *J* is a flux.

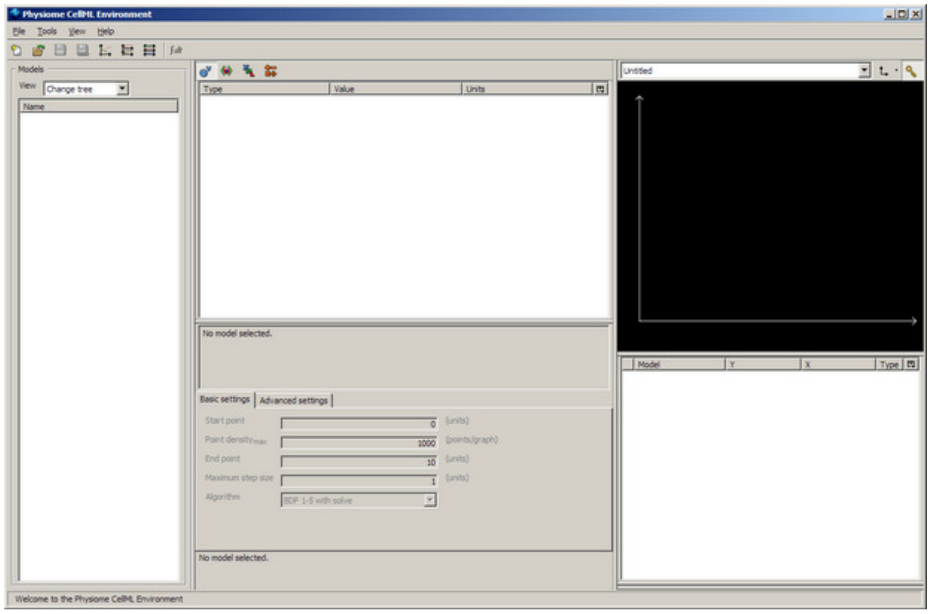
To create this model in CellML, we will use two *components*: an *environment* component to contain the *time* variable, and a *flux* component to contain the reaction. We will use seven variables: *time*, the reaction species *A, B,* and *C*, the rate constants k_f and k_b, and the flux *J*. There will be a connection between the two components to pass the *time* variable, allowing the flux component to make use of the value of *time* from the *environment* component.

Although the best description of elements within a CellML model can be found within the CellML 1.1 Specification, a brief comment on components and connections is in order here.

Every variable in a CellML model is defined within a component. These components allow a modeller to organize and group variables and equations into modules. There is no ordained way to do this, so a modeller may in fact choose to create only one component and put everything in the model within it, but usually the components represent discrete processes or entities within the model, such as protein interactions in a signal transduction model. This is useful both because it keeps the model neat and aids conceptualization, and because it facilitates the creation of reusable modules.

Connections allow the modeller to map variables within different components to each other. This means that a variable may be defined by an equation in one component and mapped to another component to be used. Because the derivative of each ODE (ordinary differential equation) in this model needs to be taken with respect to time, the variable that tracks time needs to be defined once and mapped to each component.

When you first load OpenCell, it will look something like this (depending on your operating system and monitor resolution):



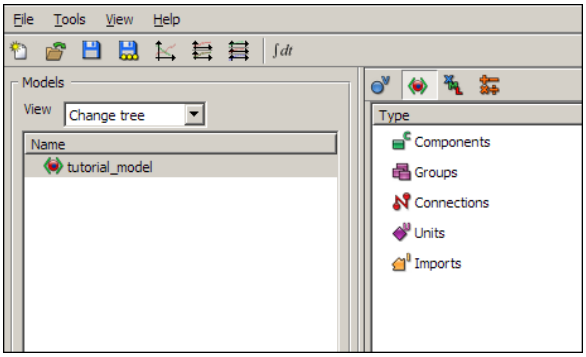
The window is divided into four sections; the model selector pane (far left), tree view pane (top middle), interaction pane (bottom middle), and the graph pane (far right). For more information about the OpenCell interface, refer to the manual - available from the *Help* menu from within OpenCell.

2.2 Creating a new CellML model

Click on the first button on the toolbar (*new model*) or go to the *File* menu and select *New*. A window titled *Create new model* will appear, asking for a name and a CellML version for your new model. Type in `tutorial_model`, and select *CellML 1.1* for the *CellML version*. Press enter, or click on *OK*. You will now have a model called *tutorial_model* visible in the model selector pane.

Now switch to the *show complete model structure* view in the *tree view pane*, using the second button on the tree view pane toolbar. This view is the main view you will use to edit models in OpenCell, and displays all the elements in a CellML model.

You will see a list of items in the tree view pane; *Components, Groups, Connections, Units,* and *Imports*.



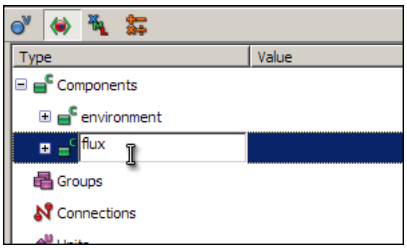
Click on the *Save* button (the diskette icon) or select *Save* from the *File* menu. Give your file the name `tutorial_model.cellml` and save it in your chosen folder. Keep the *Save as type* option as *CellML Models*.

Save regularly during this tutorial.

2.2.1 Making new components

You are going to create two components; one called *environment* to contain the global variable *time*, and another called *flux* to contain the variables and equations for the reaction.

To create a new component, right click on the green icon labelled *Components*, and select *New* from the context menu. A component called *New_component* will appear below *Components* - select the name of this component, type in `environment` (in place of `New_component`) and press enter. Now create another component, and rename it to `flux`.

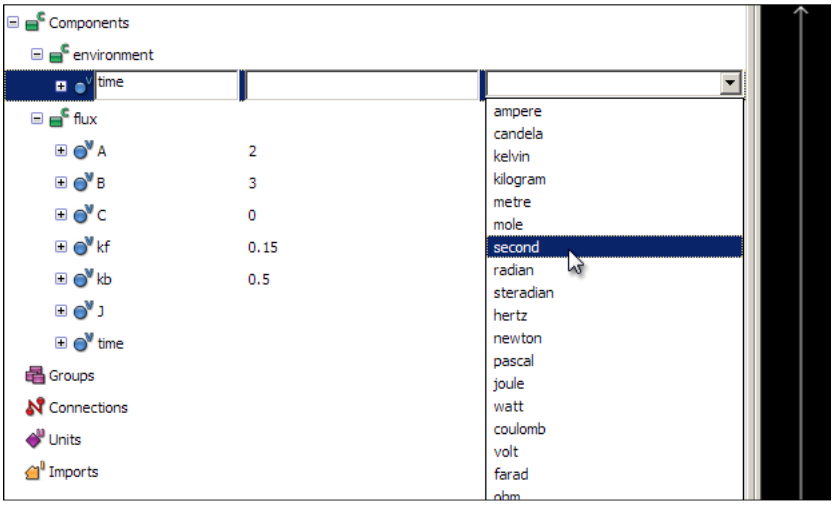


2.2.2 Creating variables

You will now create the variables for the model, within the appropriate components. First create the *time* variable: right click on the *environment* component's icon in the tree view (the green rectangle with a C, labelled *environment*) and select *New variable* from the context menu. Re-name the variable from *New_variable* to `time`.

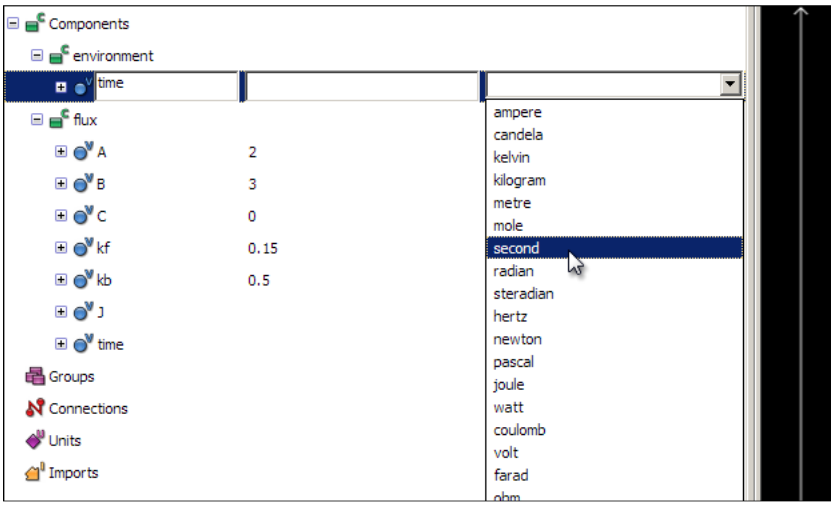
The other six variables should be created in the flux component. Create the variables *A, B, C, kf, kb* and *J* under the *flux* component. Some of these variables will require *initial values*. These can be entered in the *Value* field of the tree view. Enter values of `2`, `3`, and `0` for the variables *A, B*, and *C* respectively. These values are the initial concentrations of these three species. The rate constant variables *kf* and *kb* should be given values of `0.15` and `0.5`.

You will also need to create the variable *time* in the flux component, which will later be connected to the *time* variable in the *environment* component. Variables that are connected must appear in both of the connected components.



2.2.3 Defining units

All variables and numbers in a CellML model must be associated with units. OpenCell has a library of pre-defined units, which includes all base and derived SI units. First, associate the unit second with the *time* variable - click on the *Units* field of the time variable in the tree view, and select *second* from the list of pre-defined units.

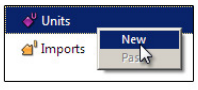


The other units used in this model will have to be defined, as they do not appear in the list of pre-defined units. The concentrations of the three reaction species are in units of micromoles per litre, or micromolar; the flux will be in units of micromolar per second. The two reaction rates will have units of per micromolar per second (for *kf*) and per second (for *kb*).

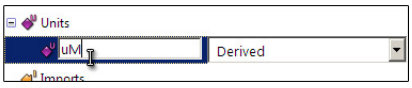
These units are made up of combinations of SI units, modified by *prefixes*, *multipliers*, *offsets*, and *exponents*. For example, the unit micromolar is made up of *mole* with a *prefix* of -6 (micro) combined with *litre* with an exponent of -1; micromoles per litre.

To define the unit micromolar, right click on the *Units* icon in the tree view and select *New* from the context menu (1). Rename this unit `uM`, and make sure that the type of unit is *Derived* (2). Now right click on the icon for *uM* and select *New unit* again from the context menu. Select *mole* from the list of units next to the yellow triangular unit node icon (3). To make these micromoles, enter a value of `-6` into the *Prefix* text field, or select *micro* from the list of prefixes. Right click again on the *uM* icon, and select *New unit* again. For this new unit, select *litre* from the list of pre-defined units. To make this per litre, enter a value of `-1` into the *exponent* field of the litre unit (4). When you now select the named unit node in the tree view, the unit definition will be displayed in the information pane (5).

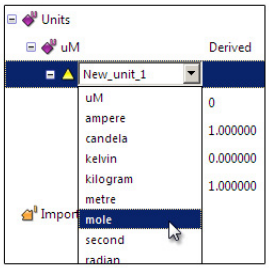
1



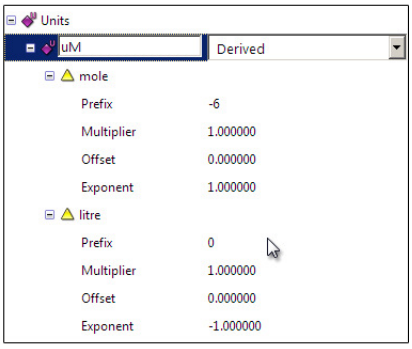
2



3



4



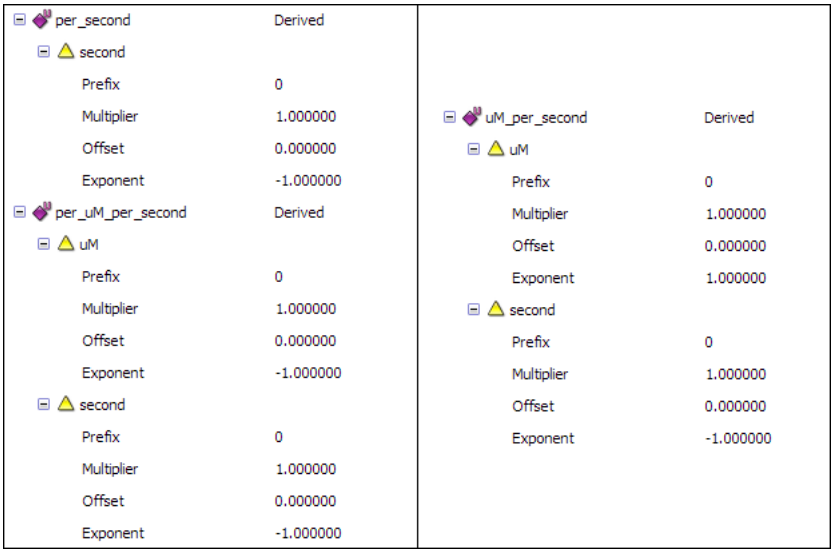
5

$$uM = \mu mol \times L^{-1}$$

To define the remaining units follow the procedure above, using the components of each unit.

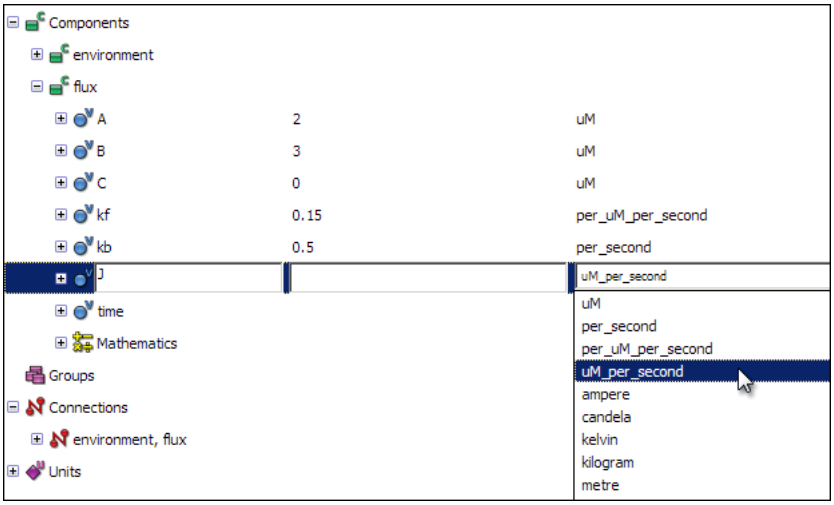
- The *per_second* unit is defined as a second unit with an exponent of -1 .
- *uM_per_second* is defined using *uM* combined with the second unit with an exponent of -1.
- *per_uM_per_second* is defined using *uM* with an exponent of -1, combined with the second unit with an exponent of -1.

This will give you the following unit definitions:



Now that you have defined all the required units, you can associate them with the variables. Click on the units field of each variable, and select the appropriate unit for each of the variables.

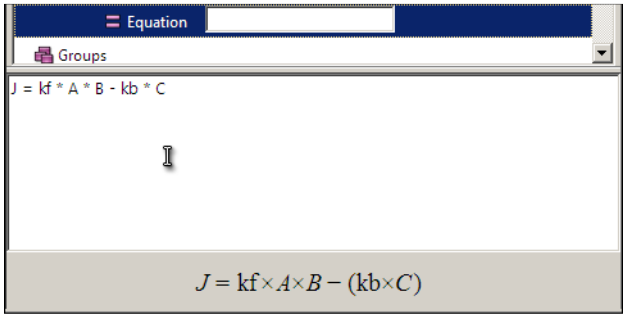
A: uM
B: uM
C: uM
kf: per_uM_per_second
kb: per_second
J: uM_per_second



2.2.4 Creating equations

Now you will enter the model equations. These are created within the appropriate component, in this case the flux component. Right click on the icon for the component in the tree view, and select *New maths* from the context menu. An icon labelled *Mathematics* will appear - right click on it and select *New equation* from the context menu. Click on the equation element, and in the text box (which will initially read *<no operator yet>*) enter the following equation:

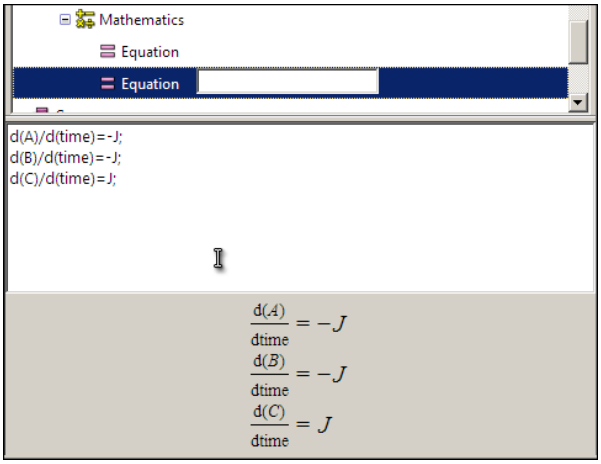
```
J=kf*A*B-kb*C
```



For the second, third, and fourth equations, create another equation and enter the following in the text box:

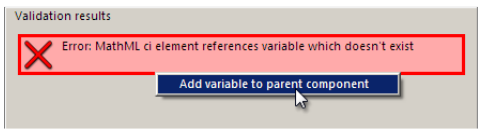
```
d(A)/d(time)=-J;
d(B)/d(time)=-J;
d(C)/d(time)=J;
```

OpenCell allows you to create multiple equations at once by entering a number of equations separated by semicolons. When you press enter, the multiple equations will appear as separate elements in the tree view. You may also enter your equations directly into the *Mathematics* tree-view element text box.



After you have entered the equations, selecting them will cause a formatted version of the equation to appear in the interaction pane below the tree view. This will only work if you have entered an equation in a way that OpenCell is able to interpret. The input format is an editable form of MathML and is similar to the format for a basic command-line calculator tool.

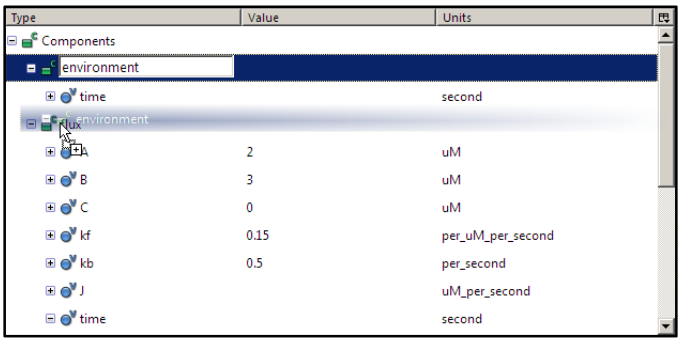
It is worth noting that if you create equations without having first created the appropriate variables, OpenCell will show errors when the model is validated. You can right click on these errors and select *Add variable to parent component* to automatically fix these errors, creating the required variable elements within the component that the equation is in. These automatically created variables will have no initial value and their units will be set to *dimensionless*. You will have to edit these fields if required.



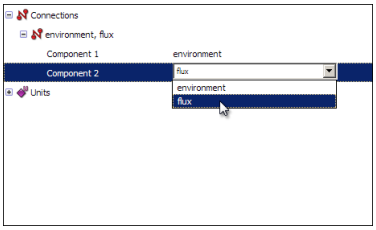
2.2.5 Creating connections

Finally, you will need to create a connection between the two components, to pass the *time* variable from the *environment* component to the *flux* component. As we mentioned previously, connections allow variables which have been defined in one component to be used within another component.

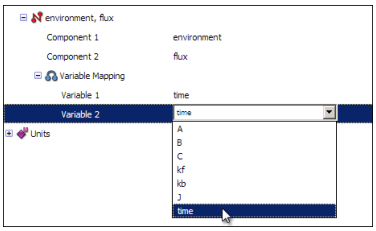
There are two ways to create connections in OpenCell. The easiest is to use the drag & drop method. To connect time in environment to time in flux, simply hold down the control key, drag the *environment* component, and drop it on the *flux* component. All of the common variables will be connected, which in this case is only *time*.



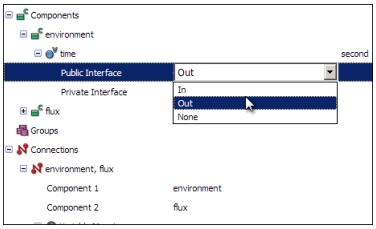
Alternatively, you can create connections using the context menus in the tree view. To create a connection, right click on the red *Connections* icon in the tree view, and select New from the context menu. Expanding the newly created connection will show you two items named *Component 1* and *Component 2*. Select *environment* and *flux* respectively from the drop-down menus.



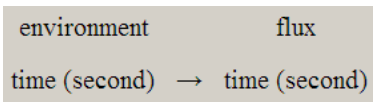
Now, right click on the connection labelled *environment*, *flux* and select *New variable mapping* from the context menu. Expanding the new variable mapping will show you two items labelled *Variable 1* and *Variable 2*. You want to map the variable time from the environment component onto time from the *flux* component, so select *time* in both of the fields.



For any newly created connection to function (created using either method), the direction of the *Public Interface* of the *time* variables must be set correctly for each component. Do this by first expanding the *time* variable in the *environment* component, and then selecting *Out* from the *Public Interface* drop-down menu. Setting up the interfaces can be done before connections are created, at the time when the variables are created.

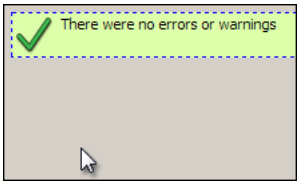


In the same way, change the *Public interface* of the *time* variable in the *flux* component to *In*. This sets the direction of the connection, as shown in the information pane when a correctly created connection is selected.



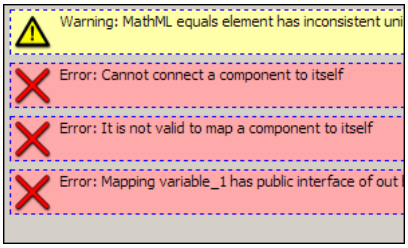
2.2.6 Validating the model

Validating the model will help you to identify and locate any errors you might have made in building your model.



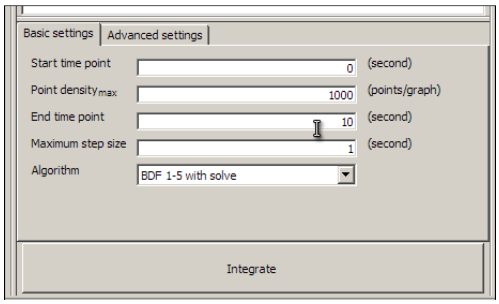
To validate the model, right click on the model name in the model selector panel on the left, and select *Validate* from the context menu. If there are no errors, a message will appear in the far right hand pane of the OpenCell window, the graph pane.

If there were errors, messages will appear describing the errors found. Double-clicking on the error messages will take you to the part of the model where the error was detected. Take note that you must be in the *complete model structure view* for this error highlighting function to work correctly. Shown on the left is an example of what these error messages might look like:



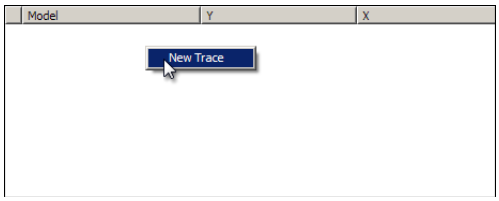
2.2.7 Running a simulation

Our model is now validated and ready to simulate. Save the model now, and then click on the Display integration pane button (the last button on the toolbar, with the letters *dt*,) to make the integration pane visible. You have to close and re-load the model before the integration pane becomes available. The integration pane should look something like this:

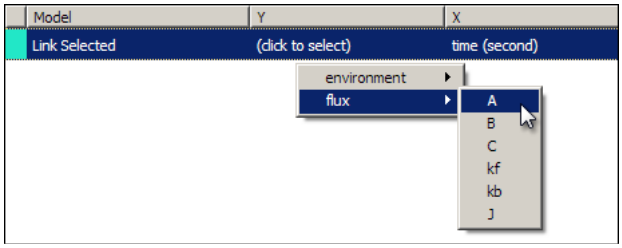


Click on the large *Integrate* button, and wait until the message *Integration complete* appears in the integration pane. The *Integrate* button will change to the *Export CSV button*. This button allows you to save the results of the simulation as comma separated values, for loading into other applications. You now have results for the model which you can display in the graph pane.

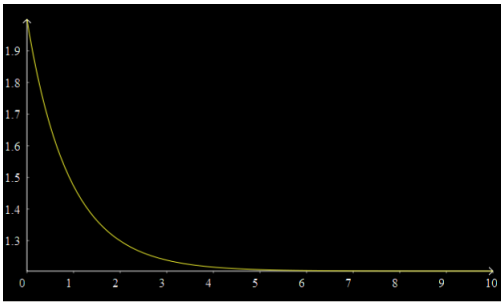
Right click in the graph trace control area in the graph pane, and select *New Trace* from the context menu.



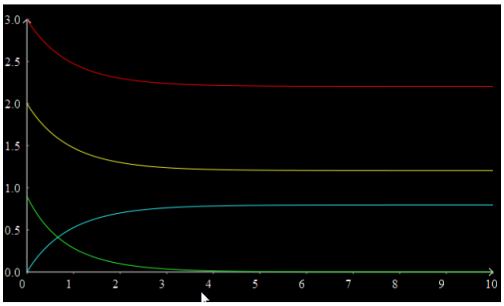
A new trace entry will appear in the listing. Click on the X column where it says *(click to select)* and select *environment*, *time* from the cascading menu. Do the same for the Y column and select *flux, A*.



A graph should now appear above, showing the concentration of *A* decaying over time to its equilibrium value.



Add more traces to the graph - another way to add graph traces is to simply drag variables from the tree view and drop them on the graph pane. Try doing this with the variables *B*, *C*, and *J* from the *flux* component. Additional new traces will automatically have time selected as the X axis variable, although you can of course change this if you wish by clicking on it and selecting a different variable. You should now have a graph that looks something like this:



You can change the colour of individual traces by clicking on the coloured block in the graph controls and selecting a colour from the pop up palette.

This concludes the first part of the tutorial. Please refer to the manual and other tutorials for more information.

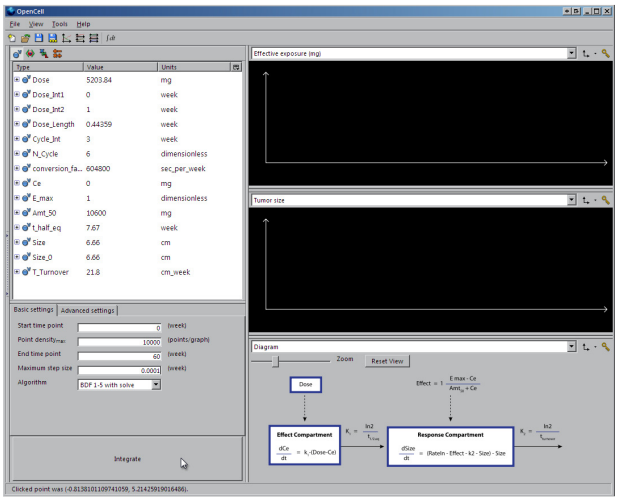
3 Creating a modular PKPD model

This part of the tutorial will guide you through the creation of a simple modular PKPD model based on the existing Tham *et al.* 2008 tumor response model. The first part of this section will involve looking at the Tham 2008 session file from the CellML model repository, running the model, and viewing the results on graphs. Next you will be instructed on how to build the effect compartment of this model from scratch using the OpenCell software. Finally you will run a modularized version of this model in OpenCell, importing your newly created effect compartment, and also swapping out the environment component for one with a different drug dosage regime.

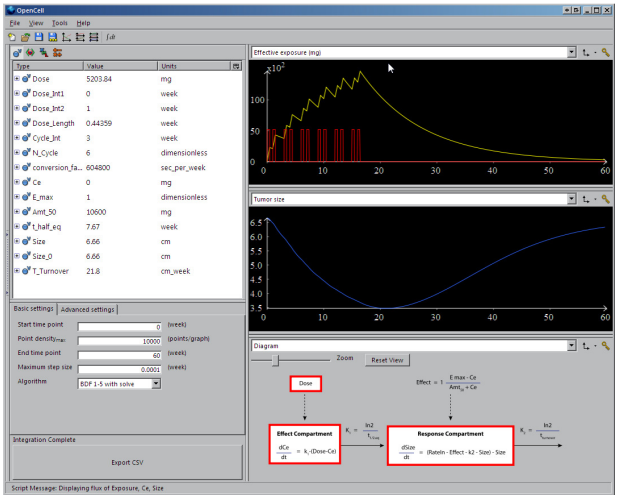
3.1 Running the Tham 2008 OpenCell session file

Start OpenCell, and open the file `tham_2008.session.xml` contained in the `tham_2008` folder. This is the OpenCell session file for this model, also available in the CellML model repository on the exposure page, http://models.cellml.org/exposure/1985/tham_2008.cellml/view. To load the session file from the repository exposure page, click on the *Simulate using OpenCell* link under *Views available* on the right hand side.

The OpenCell application window will look something like this when the session file is loaded:



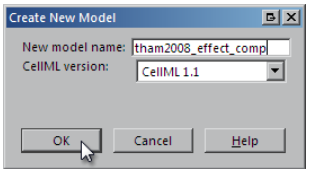
Click on the Integrate button at the bottom left of the window to generate results to graph. Once you have integration results, clicking on the compartments in the diagram at the bottom right will cause the appropriate graphs to appear in the graph panels, as shown below.



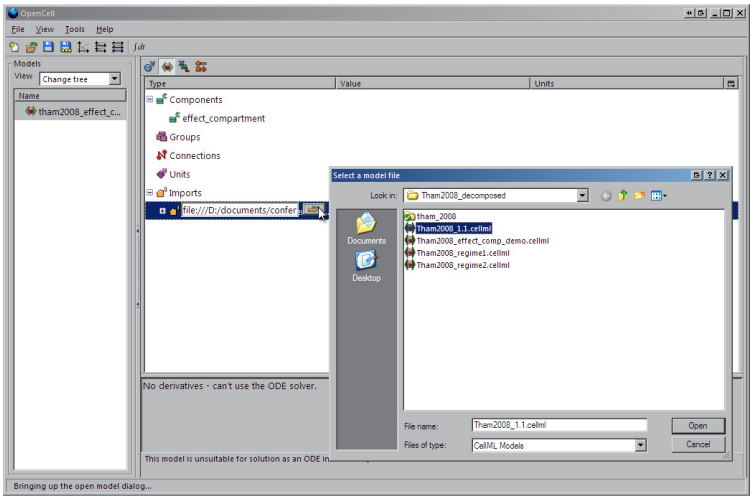
3.2 Creating the effect compartment in OpenCell

In this section of the tutorial, you will create the effect compartment of the Tham et al. 2008 model using OpenCell, and then import the environment component from the modular version of the model so that the effect component can be integrated. Unit definitions for the effect compartment model will also be imported to save time.

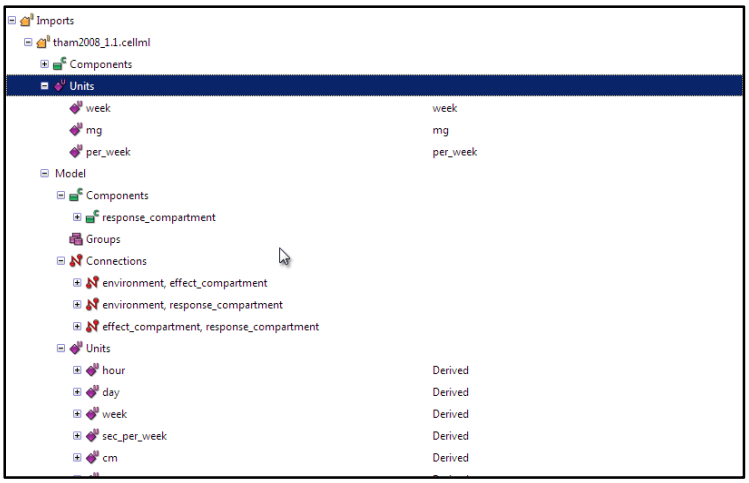
Load OpenCell, and create a new 1.1 model. Name it `tham2008_effect_comp`. Save the model in the `Tham2008_decomposed` folder, using the filename `tham2008_effect_comp.cellml`



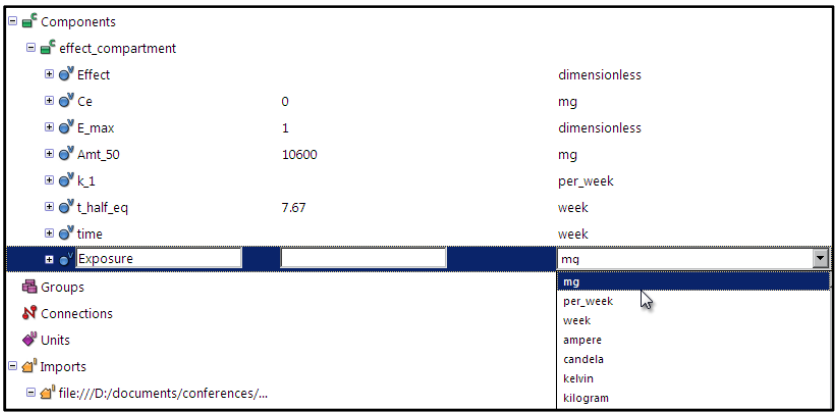
Instead of creating all the unit definitions required in the effect compartment, we can import them from an existing model. Enter the complete model structure tree view, and create a new import. Click on the file browsing button beside the import component, and select the model `tham2008_1.1.cellml`.



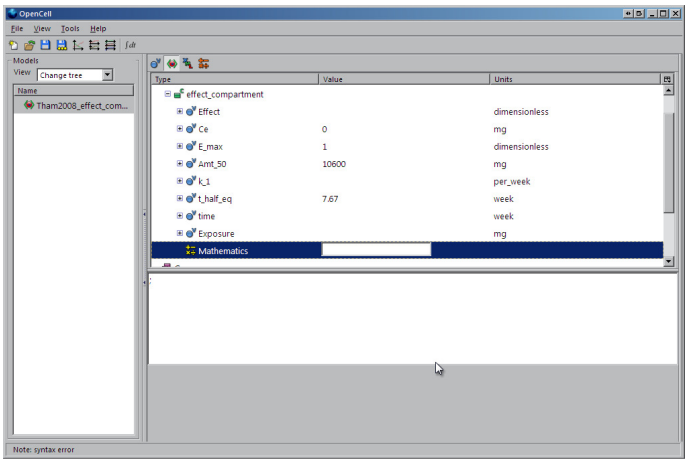
Expand the import element, and find the *model - units* branch. You will need to create an entry under the *import - units* branch for each of the unit definitions required, copying the name of the units into both text entry boxes as shown below. You will need the unit definitions `mg`, `week`, and `per_week`.



Now create a new component in the model tree, and name it `effect_compartment`. Within this component, create the following variables, with their initial values and units as shown below:



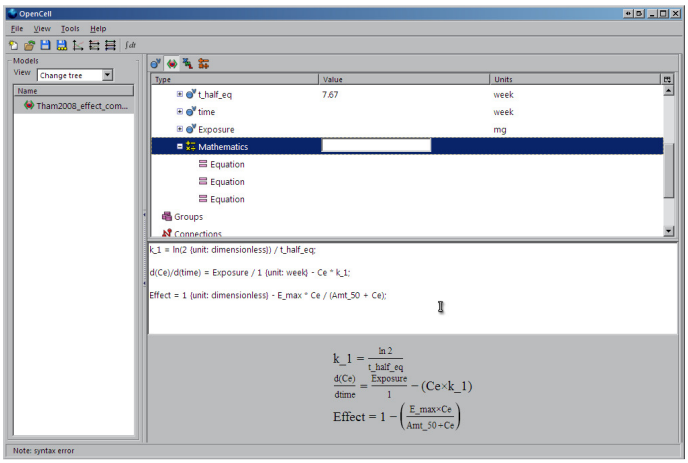
To enter the mathematical equations for the effect compartment, first create a new Mathematics element under the `effect_compartment` element.



Now, paste the following text into the text entry box provided.

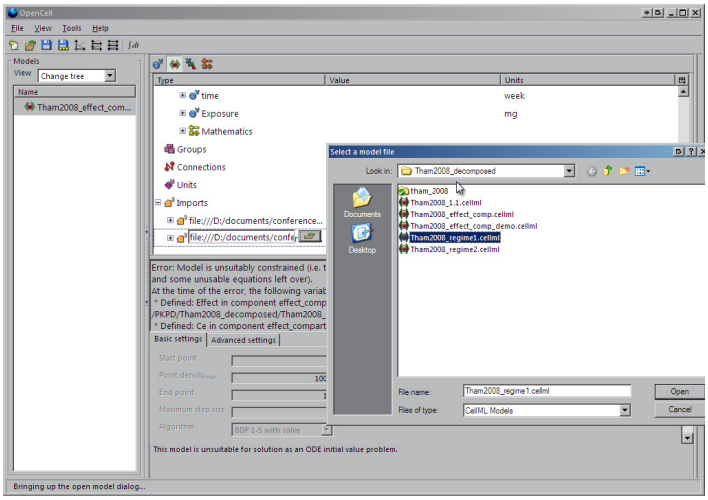
```
k_1 = ln(2 {unit: dimensionless}) / t_half_eq;
d(Ce)/d(time) = Exposure / 1 {unit: week} - Ce * k_1;
Effect = 1 {unit: dimensionless} - E_max * Ce / (Amt_50 + Ce);
```

Each equation separated by a semi-colon will automatically become a separate Equation element under the Mathematics element, as shown below.

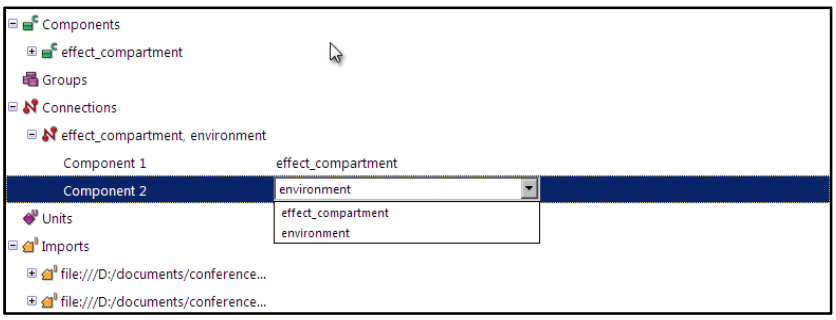


3.3 Running the effect compartment model

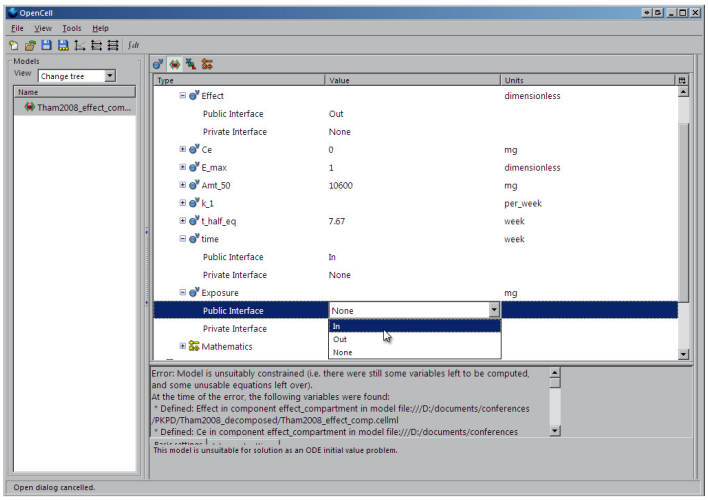
In order to be able to run the effect compartment model, we now need to import the environment component to provide the variable of integration and the drug dosage regime. Create another Import element, and select the model `Tham2008_regime1.cellml` with the file selection button.



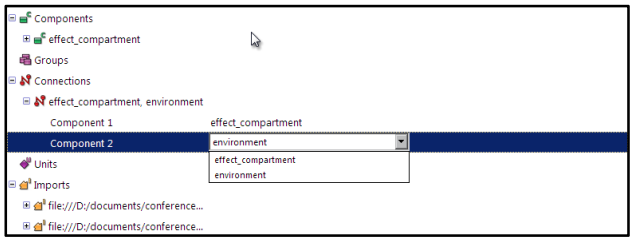
Create a new component under the import, and enter `environment` into both the text boxes to import the environment component from the regime model.



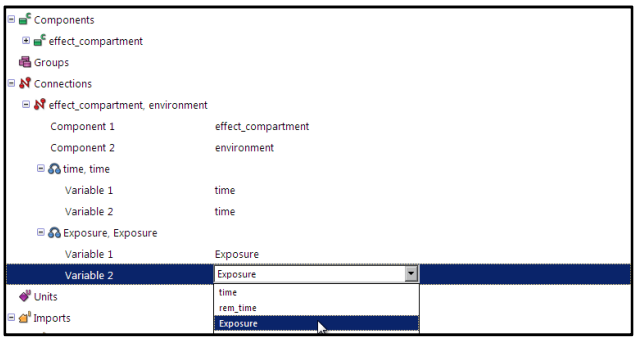
Now we need to connect the variables that pass values between the `environment` and `effect_compartment` components. First we need to create interfaces on the variables in the `effect_compartment` that we created. Expand the variables `Effect`, `time`, and `Exposure`. Set their public interface values to Out, In, and In respectively.



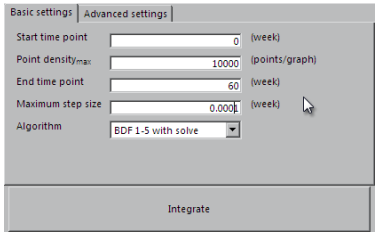
We must now create connections between the variables in the environment and effect_compartment components. Create a new connection, and select the components.



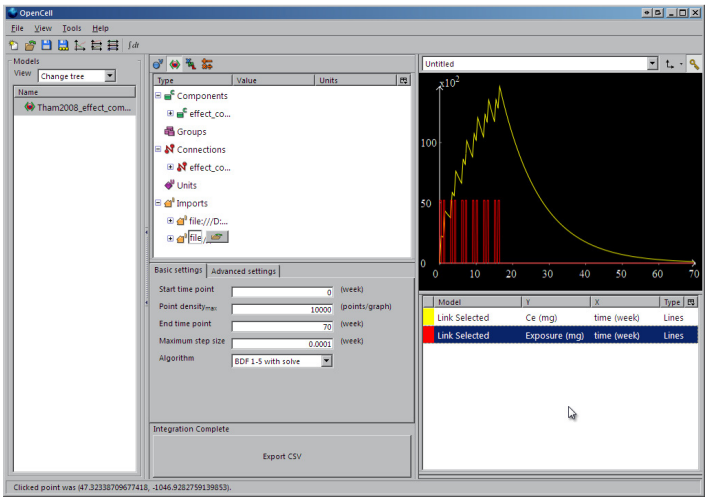
Now create variable mappings for the variables `time` and `Exposure`.



Save you model, then right click on its name in the model listing and select *Reload*. The integration button should now be available for running the model. Before running the model, set the integration settings to appropriate values as shown below.



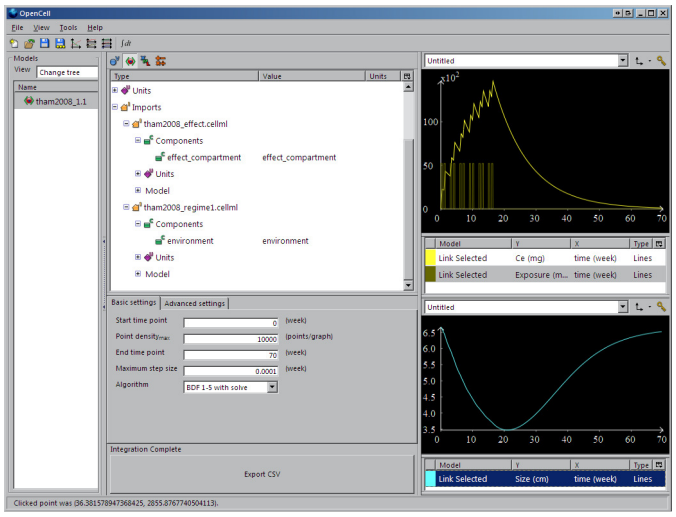
Now click on integrate, and graph the variables `Ce` and `Exposure` against `time`. The results should display as shown below.



3.4 Using CellML imports to swap compartments

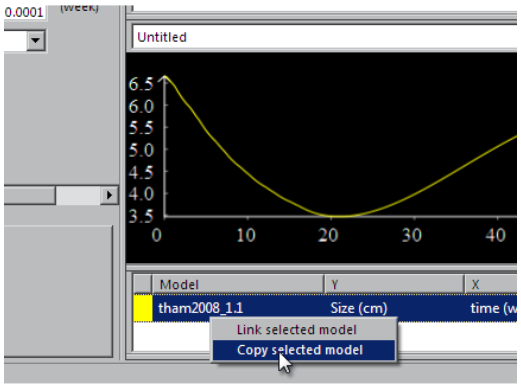
As you have already seen, CellML 1.1 allows the re-use of models and parts of models through the use of imports. For this tutorial I have created a very basic demonstration of this which will show you a simple example of how this might be used.

Load OpenCell, and open the `tham2008_1.1.cellml` model from the `tham2008_decomposed` folder. Integrate the model, and create the graphs shown below:

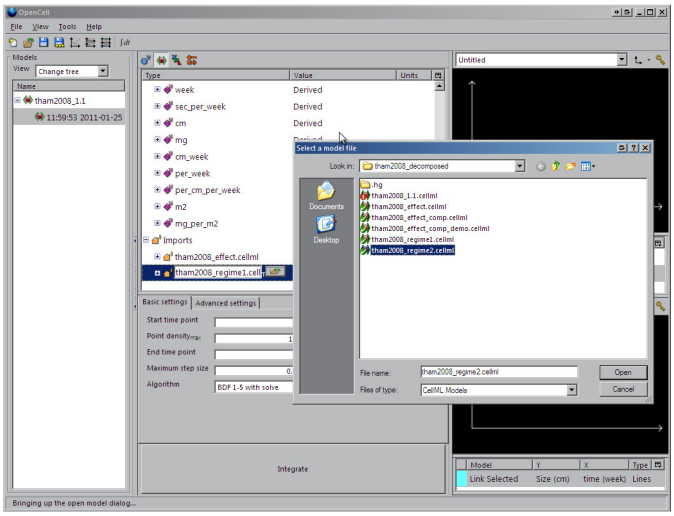


This model imports both the effect component and the environment component from separate CellML files. As the environment component contains the parameters for drug dosage (Exposure), it is possible to change the imported CellML file in order to instantly select a new drug exposure regime.

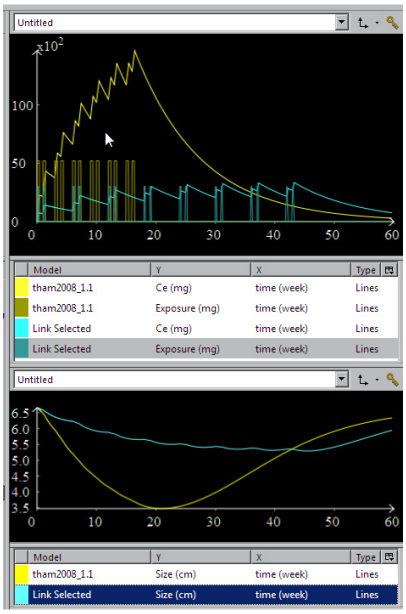
We will now create a clone of the model, which we can alter by importing a different `environment` component. Before we do so, we want to lock our existing graphs in place, so that we can compare them to the graphs from our altered model. To do this, click on the graph traces where it says *Link Selected* and choose *Copy selected model*. Do this for all three traces. The text *Link Selected* will change to the name of the model which you copied the integration results from. These graphs will now be maintained even when new integration results are generated.



To create your altered model, first right click on the model name in the model listing panel, and select Clone. This will create a new model underneath the original, named by the time of the clone's creation. Select the cloned model, and then find the import which is importing the `tham2008_regime1.cellml` model. Click on the file selection button next to this import, and select the `tham2008_regime2.cellml` model as shown below.



Integrate the model with the new import loaded. Now create a new copy of each trace you created for the original model. You should now be able to see results for the original and altered model on the same graphs, as shown below.



4 Conclusion

In this tutorial we have introduced you to some of the basic concepts of CellML 1.1, in particular the modularity and re-use features of the language. For further tutorials and information, please visit the *getting started* section at http://www.cellml.org/getting-started/.

We encourage you to get involved with the CellML project via the mailing lists and the tracker. You will find links to the mailing lists on the *getting involved* section of the website. You can discuss features of and issues with the software on the CellML sections of the Physiome Tracker, at https://tracker.physiomeproject.org/ .