Compartmental and Distributed Models
Session 1, July 15-19, 2009
Exercises
Supplementary Materials

Please note that all the models referenced here are contained in the Monday Files.
In order, they are
SynthesisModel.proj
ExchangeModel.proj
ConvertModel.proj
FlowModel.proj
AllProcesses.proj
AllProcessesPDE.proj

Compartmental Modeling:

Lesson I: Introduction to compartmental modeling and using JSim

1. A restricted compartmental definition which covers a few simple cases:

A compartment represents a physical volume, V. In the volume, there is an amount of material, Q. The concentration, C, is given by

$$C = Q / V.$$

A compartment is assumed to be instantaneously well mixed. In physical terms, this means that the diffusional process is infinitely fast.

Sources are processes which increase Q. Sinks are processes which decrease Q. The rate of change of Q with respect to time, the change in the amount is given by

$$dQ / dt = + Sources - Sinks.$$

This is called an ordinary differential equation (ODE).

In the simpler models, the volume is usually constant, and the equation is written as either

$$V * dC / dt = + Sources - Sinks$$

or

$$dC / dt = Sources / V - Sinks / V.$$

The complete specification of an ODE requires an initial condition (IC). The initial condition specifies the concentration at time equals 0. This is written as

$$C(0) = C_{0.}$$

2. Specifying units:

The units for the amount of material are usually moles (*mole*), millimoles (*mmol*), micromoles (*umol*), nanomoles (*nmol*) or picomoles (*pmol*). The units for concentration are usually molar (*M*), millimolar (*mM*), micromolar (*uM*), or nanomolar (*nM*), or picomolar (*pM*). The units for volume include liter (both *liter* and *L*), milliliter (ml), and cubic centimeters (cm^3). The units for time include seconds (both *s* and *sec*), millisecond (*ms* and *msec*), minute (*min*) and hour (*hr*). A 1 molar concentration is 1 mole of substance per liter. Water has a concentration of 55

moles/liter or 55 Molar.

3. A compartmental model for Synthesis, Constant Value, or Decay

The ODE and IC are given by

$$dC/dt = S*C/V,\ C(0) = C_{0.}$$

The analytic solution is

$$C(t) = C_0 * \exp(S*t/V).$$

If $S>0$, the right hand side of the equation is a source and the solution is exponentially growing. If $S=0$, the solution is a constant (see previous example). If $S<0$, the right hand side of the equation is a sink and the solution is exponentially decaying.

4. Coding the model in JSim:

```
* Note: Once something has been explained in the model code, the
 comment will be deleted from future models, e.g., a comment can be in
 block form beginning with forward slash asterisk and terminated with
 asterisk forward slash. Other comments can be in the form two forward
 slashes followed by the comment. */

import nsrunit;     // enables units in the model
unit conversion on; /* Checks equations for
                       dimensional compatibility. */


math SynConDecay {// 3 models in 1
// Independent Variable
realDomain t s; t.min=0; t.max=3; t.delta=0.1;

// Dependent Variable and constants
real C(t) mM,
     C0 = 1 mM,
     V  = 2 ml,
     S  = 1 ml/s; // Synthesis rate if S>0

// Initial Condition
when(t=t.min) C=C0;

// Ordinary Differential Equation
C:t=S*C/V;

// Analytic Solution
real analyticC(t) mM;
```

```
analyticC=C0*exp(S*t/V);
} // End of model
```

Model is in Synthesis.proj.

5. Standard instructions for running models (instructions when you start without a project)

a. Launch JSim. This will open a **Project** GUI and **Message** GUI.

b. Use the **Add** button to add a **New Model**. Name the new model SynConDecModel.

c. Use the **ConstantModel** button to open the model source code GUI.

d. Type the model code **in boldface (above)** into the large blank window (left handside).

e. Use the **Compile** button to compile the model. When compilation is successful the **Run Time** Gui will appear.

f. Use the **Run** button to run the model. Change the value of S from 1 to 0 to -1 and run the model each time.

g. Use the p**lotpage_1** button to switch from the **Message** GUI to the **plotpage_1** GUI.

h. There will be an area of the plot page which shows **Data [SynConDecModel] [ ] [▼] Curve [1] show [✓].**

Using the [▼] button will display available plotting choices. Select "C". Selecting the **[1]** button will let you change it to **Curve [2]**. Note that the color button changes from ■ (black) to ■ (red). Select the [▼] button again and plot "analyticC". Below the word **Curve** will be a button with a horizontal line. Use that button to change the line texture to dashes. Use the other buttons to add symbols, change line thickness, change colors, etc.

h. Label the plot by Clicking on **[Title]** and **[axis label]**.

i. Reposition the legend and title using click and drag on them.

j. Explore what is under the rest of all of the buttons by selecting them.

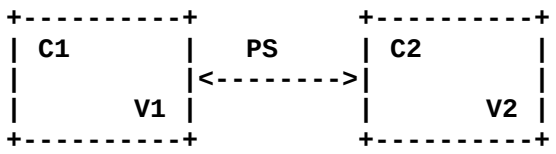
k. Go back to the **Project** button. Use the **File** button to **Save** the project as SynConDecModel.proj.

l. Use the **File** button again to **Close** or **Exit** the project.

With a project file (file ending in .proj, you can launch the project (varies in different operating systems).

Lesson II. Multiple compartments

1. Compartmental Model with exchange:



This is a model with exchange between two compartments. The exchange process is governed by an exchange coefficient. For reasons which will be explained in a later lesson, the exchange coefficient is designated *PS.* The governing equations are

$$V_1 * dC_1/dt = PS * C_2 - PS * C_1$$
$$V_2 * dC_2/dt = PS * C_1 - PS * C_2 \quad ,$$

which emphasizes the mass balance. They are usually written in the form

$$dC_1/dt = (PS/V_1) * (C_2 - C_1)$$
$$dC_2/dt = -(PS/V_2) * (C_2 - C_1) \quad .$$

Note that the right hand side (*RHS*) of the first equation is divided by V_1 and the RHS of the 2nd equation is divided by V_2. The two volumes are not necessarily equal.

2. Coding the model in JSim:

```
import nsrunit;     // enables units in the model
unit conversion on; /* Checks equations for
                                dimensional compatibility. */
/*
    +----------+          +----------+
    | C1       |   PS     | C2       |
    |          |<-------->|          |
    |       V1 |          |       V2 |
    +----------+          +----------+
*/
math ExchangeModel {// Exchange model
// Independent Variable
realDomain t s; t.min=0; t.max=3; t.delta=0.1;

// Dependent Variable and constants
real C1(t) mM, C2(t) mM,
     C10 = 1 mM, C20 = 0 mM,
     V1  = 2 ml, V2  = 10 ml,
     PS  = 1 ml/s; // Exchange rate

// Initial Conditions
when(t=t.min) {C1=C10; C2=C20;} // Note use of Brackets

// Ordinary Differential Equations
C1:t= (PS/V1)*(C2-C1);
C2:t=-(PS/V2)*(C2-C1);

/* Analytic Solutions from Maple(TM) */
real analyticC1(t) mM, analyticC2(t) mM,
```

```
     Exponential(t) dimensionless;

Exponential=exp(-PS*(V1+V2)*t/(V1*V2));
analyticC1(t) = (1/(V1+V2)) * (C20*V2+C10*V1
                              + V2*(C10-C20)*Exponential);
analyticC2(t) = (1/(V1+V2)) * (C20*V2+C10*V1
                              - V1*(C10-C20)*Exponential);

} // End of model
```

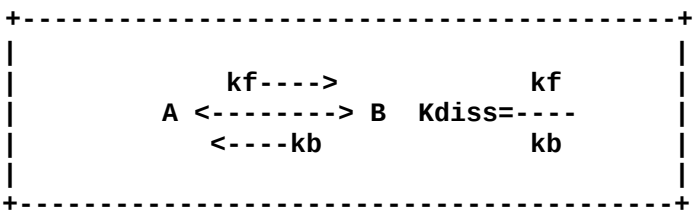
Model is in ExchangeModel.proj.

To run this model see instructions under Lesson I.5. Use the **View** tab to **reset # rows** to **2.** Use the **[plot 1]** button to change to **[plot 2]**. Instead of using the **[▼]** button, type C1−analyticC1 in the blank box where the plot variable would be. You can display plot values with the mouse by positioning in in the plot area and depressing the left hand button.

Question: Ordinary Differential Equations are solve dnumerically by many different methods. Access the ODE solvers from the **Run Time** GUI by using the **Pages** tab and selecting **Solvers**. Change the ODE method to **Euler** and set **nstep** to 1. ***Estimate*** the maximum error as a function of t.delta by changing t.delta (Run Time GUI) from 0.1 to 0.01and 0.001. Do the same for RK2 and RK4. Interpret your results.

Lesson III. Multiple substances in a compartment

1. Compartmental Model with conversion:



This is a model where substance A can become substance B and substance B can become substance A. The conversion is governed by a single rate constant and a dissociation constant, K_{diss}. The dissociation constant is the inverse of the equilibrium constant.

$$K_{diss} = k_f / k_b.$$

$$k_f = K_{diss} * k_b.$$

$$dA/dt = -k_f * A + k_b * B.$$

$$dB/dt = k_f * A - k_b * B.$$

Note that the volumes are not used in this equation because the volumes would appear on both sides of the equations.

2. Coding the model in JSim

```
import nsrunit;
unit conversion on;
/*
     +----------------------------------------+
     |                                        |
     |             kf---->            kf     |
     |         A <--------> B  Kdiss=----     |
     |            <----kb              kb     |
     |                                        |
     +----------------------------------------+
*/
math Convert { // Conversion model
// Independent Variable
realDomain t s; t.min=0; t.max=3; t.delta=0.1;

// Dependent Variable and constants
real A(t) mM, B(t) mM,
     A0 = 1 mM, B0 = 0 mM,
     Kdiss = 3 dimensionless, kb = 1 sec^(-1), kf sec^(-1);
     kf = Kdiss*kb;

// Initial Condition
when(t=t.min) {A=A0; B=B0;} // Note the use of brackets

// Ordinary Differential Equation
A:t = -kf*A + kb*B;
B:t =  kf*A - kb*B;

/* Analytic Solutions from Maple(TM) */
```

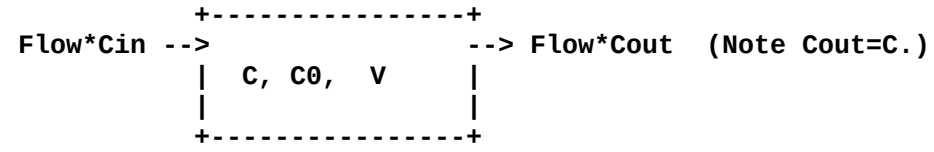
```
real analyticA(t) mM, analyticB(t) mM,
     Exponential(t) dimensionless;
Exponential = exp(-(Kdiss+1)*kb*t);
analyticA = (1/(Kdiss+1))*(B0+A0 + (Kdiss*A0-B0)*Exponential);
analyticB = (1/(Kdiss+1))*(Kdiss*(B0+A0)-(Kdiss*A0-B0)*Exponential);

}  /* End of Model*/
```

Model is in ConvertModel.proj.

To run this model see instructions under Lesson I.5. Explore the model by changing values for the various parameters and running solutions. You may have have noticed that the volume is missing from this model. When everything happens in the same volume, and flow is not involved, the volume is usually excluded from the model.

Lesson IV: Compartment Model with flow:

1. Inflow and Outflow



The next model is the inflow of a concentration and the outflow of a concentration in a single compartment. Recall that in a compartmental model, mixing occurs instantaneously. The governing ODE and IC are:

$dC/dt = Flow/Volume*(Cin(t)-Cout(t));$ where *Cin(t)* is the inflowing concentration and *Cout(t)* is the outflowing concentration, and $C(0)=0;$

Since the compartment is instantaneously well mixed, $Cout(t)=C(t)$

and we can rewrite the ODE as

$dC/dt = Flow/Volume*(Cin(t)-C).$

The analytic solution involves an integral. For a constant inflowing concentration, *Cin(t)=Cin*, the analytic solution is given by $C(t)=Cin-(Cin-C_0)*\exp(-Flow*t/Volume).$

The time constant for the system is defined as $\tau = Volume/Flow.$

2. Coding the model in JSim:

```
import nsrunit; unit conversion on;
/*
                +----------------+
     Flow*Cin -->                --> Flow*Cout  (Note Cout=C.)
                |  C, C0,  V     |
                |                |
                +----------------+
*/
math ExchangeModel {realDomain t s; t.min=0; t.max=10; t.delta=0.1;

// Dependent Variable and constants
real C(t) mM,
     C0     = 0 mM,
     Volume = 2 ml,
     Flow   = 1 ml/s; // Flow rate
extern real Cin(t) mM; // Cin will be defined at run time using a
                       // function generator
// Initial Condition
when(t=t.min) C=C0;

// Ordinary Differential Equation
C:t= (Flow/Volume)*(Cin-C);

// Analytic Solution only works when Cin is a constant
real analyticC(t) mM;
real tau=Volume/Flow; //Note that the dimensionality of tau will be
                      // assumed from this equation.
analyticC=Cin-(Cin-C0)*exp(-t/tau);
```

```
} // End of model
```

Model is in FlowModel.proj.

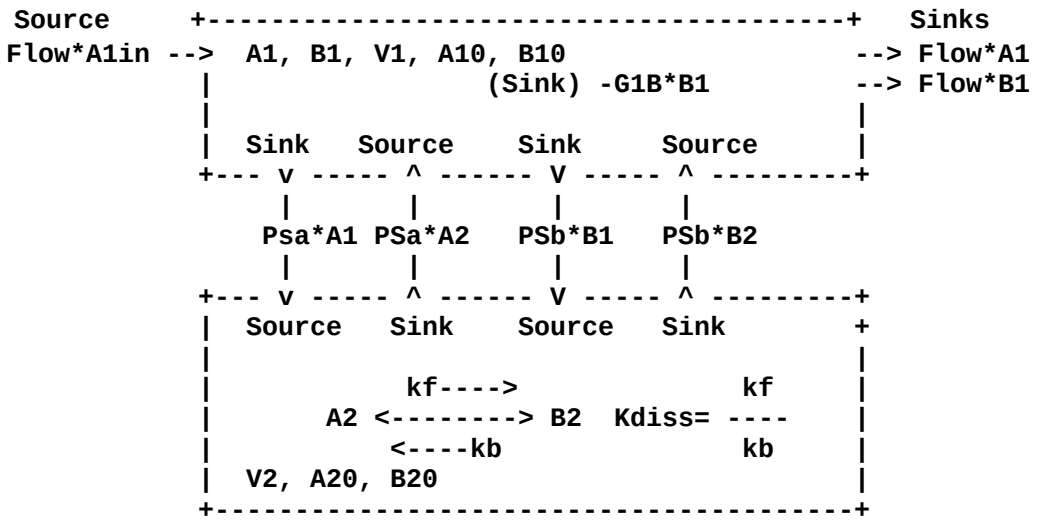
To run this model see instructions under Lesson I.5. In addition you will have to set the input concentration. In the **Run Time** Gui, enter **0** for **Cin(t),** and run the model. Plot out **C, analyticC**, and **Cin**. Vary **C0** and **Cin**.

3. The function generator

The function generator for **Cin(t)** can be accessed by the small sine wave enclosed in a circle **(~)** button to the right of **Cin(t)** . Selecting this button will bring up a menu for creating a function generator. You can accept the default name fgen_1 or create a new function generator with a different name. Select "Use existing fgen_1." Selecting the Pulse 1 button gives you access to all of the predefined functions. Note that the analytic solution based on constant input will no longer be valid as a comparison. You may wish to increase t.max to 30. Compartment flow models will be revisited when flow compartment in series are considered.

Lesson V: Putting it all together:

1. Creating a new model



We next construct a model for two species with inflow of species A, exchange of A between two compartments, conversion of A to Bin non-flowing compartment, exchange of B with the flowing compartment, consumption of B in the flowing compartment and outflow of A and B.

The governing equations are

$K_{diss} = k_f / k_b.$

$k_f = K_{diss} * k_b.$

$$dA_1/dt = (Flow/V_1)*(Ain - A_1) - (PS_A/V_1)*(A_1 - A_2);$$

$$dA_2/dt = (PS_A/V_2)*(A_1 - A_2) - k_f*A_2 + k_b*B_2;$$

$$dB_1/dt = (Flow/V_1)*(-B_1) - (PS_B/V_1)*(B_1 - B_2) - (G_{1B}/V_1)*B_1;$$

$$dB_2/dt = (PS_B/V_2)*(B_1 - B_2) + k_f*A_2 - k_b*B_2;$$

2. Write this model for JSim, using the previous models as examples. Don't forget to include the initial conditions. A correct version of this model is included but do not use it unless all else fails. Learning how to write and debug models is part of the exercise. With A10=A20=A30=A40=0, Ain = 1 mM (constant), Kdiss=2, kb = 4/sec, G1B=2 ml/sec, V1=1 ml, V2=2 ml, PSa=PSb=3 ml/sec, and F= 1 ml/sec, what is the value of the outflow of B1 at 20 seconds? (Answer: B1out(20 seconds) =.209 mM.)

Question: Identify the sources and sinks in all four equations. It may be necessary to multiply the terms out to separate sources and sinks. (Hint: Sources have plus signs, sinks have minus signs.)

Question: What are some parameter and initial condition combinations to make this model into the (1) Synthesis model, (2) the Exchange model, (3) the Conversion Model and (4) the Flow Model. Indicate what the output variables are for each.

```
import nsrunit; unit conversion on;
/*
    Source      +---------------------------------------+   Sinks
    Flow*A1in -->  A1, B1, V1, A10, B10                 --> Flow*A1
                |                 (Sink) -G1B/V1*B1     --> Flow*B1
                |                                       |
                |  Sink   Source    Sink     Source     |
                +--- v ----- ^ ------ v ----- ^ ---------+
                     |       |        |       |
                    Psa*A1 PSa*A2   PSb*B1   PSb*B2
                     |       |        |       |
                +--- v ----- ^ ------ v ----- ^ ---------+
                |  Source   Sink    Source   Sink        +
                |                                       |
                |            kf---->              kf     |
                |       A2 <--------> B2  Kdiss= ----    |
                |           <----kb               kb     |
                |  V2, A20, B20                         |
                +---------------------------------------+
*/
math AllProcessesODE {realDomain t s; t.min=0; t.max=20; t.delta=0.1;

// Dependent Variable and constants
real A1(t) mM, A2(t) mM, B1(t) mM, B2(t) mM, A1out(t) mM, B1out(t) mM,
     A10 = 0 mM, A20 = 0 mM, B10 = 0 mM, B20 = 0 mM,
```

```
     V1  = 1 ml, V2  = 2 ml,
     Flow   = 1 ml/s, // Flow rate
     PSa    = 3 ml/s, // Passive exchange rate for A
     PSb    = 3 ml/s, // Passive exchange rate for B
     G1B    = 2 ml/s, // Consumption rate for B in flowing compartment
     Kdiss = 3 dimensionless, //Dissociation constant
     kb = 4 sec^(-1), // Backwards rate constant
     kf sec^(-1);     // Forward rate constant
     kf = Kdiss*kb;      //
extern real A1in(t) mM; // A1in will be defined at run time using a
                        // function generator
// Initial Condition
when(t=t.min) {A1=A10; A2=A20; B1=B10; B2=B20;}

// Ordinary Differential Equation
A1:t = (Flow/V1)*(A1in-A1) + (PSa/V1)*(A2-A1);
A2:t =                       (PSa/V2)*(A1-A2)     -kf*A2+kb*B2;
A1out = A1;
B1:t = (Flow/V1)*( -B1)    + (PSb/V1)*(B2-B1)     -(G1B/V1)*B1;
B2:t =                       (PSb/V2)*(B1-B2)     +kf*A2-kb*B2;
B1out=B1;

// No analytic solutions are given for the general case. Compare
// Analytic solutions for special cases where they exist

} // End of model
```

Model is in AllProcesses.proj.

Lesson VI: Equations for distributed regions
(Partial Differential Equations)

Consider a flow carrying a concentration of a metabolite through a tube. In a small region of the tube, we write the equation for mass conservation in one dimension as:

$$d(V*C)/dt=\partial(V*C)/\partial t+U*\partial(V*C)/\partial x=+Sources-Sinks+V*Diffusion.$$

where V is the volume, C is the concentration, U is the velocity, t is the time and x is the spatial dimension.

$$d(V*C)/dt=\partial(V*C)/\partial t+U*\partial(V*C)/\partial x$$

means that the total derivative (also called the material derivative) is equal to the local rate of change plus the advection of a gradient. See any standard text of fluid dynamics for a fuller explanation. To the observer positioned at a particular place along the tube, what is observed is the local rate of change from sources and sinks and the advection of a gradient. The gradient term is moved to the right hand side, and we customarily write,

$$\partial(V*C)/\partial t=-U*\partial(V*C)/\partial x+Sources-Sinks+V*Diffusion.$$

Assuming the volume is constant, the volume can be placed outside the partial derivatives.

$$V*\partial C/\partial t=-U*V*\partial C/\partial x+Sources-Sinks+V*Diffusion.$$

Let $U*V$ the velocity multiplying the volume be rewritten as

$$-U*V=(-Flow/Area)*(Area*L)=-Flow*L,$$

where $Flow$ is the flow rate, commonly given in ml/unit time or (ml/gram of tissue)/unit time, and $Area*L=Volume$ of the capillary, and L is the length. This yields

$$V*\partial C/\partial t=-Flow*L*\partial C/\partial x+Sources-Sinks+V*Diffusion.$$

Dividing both sides by the capillary volume, we have

$$\partial C/\partial t=-(Flow*L/V)*\partial C/\partial x+Sources/V-Sinks/V+Diffusion.$$

Note that if a region has no flow,

$$dC/dt = \partial C/\partial t,$$

and there is no advective term.

How does the partial differential equation compare with the ordinary differential equation for a stirred tank? Consider the simple case with advection and no diffusion, sources or sinks. The ordinary differential equation is given as

$$dC/dt = Flow/V * (Cin - Cout).$$

With slight rearrangement, this is the same as

$$dC/dt = -Flow * L/V * (Cout - Cin)/(L).$$

But

$$(Cout - Cin)/L = (C(L) - C(0))/(L - 0)$$

which is approximately equal to dC/dx.

Axial diffusion is neither a source nor a sink. It redistributes the molecules in the region.
In a region with no flow, the diffusion equation is

$$\partial C/\partial t = D * \partial^2 C/\partial x^2$$

where D is the diffusion coefficient in cm^2/sec .
When advection is included, we have the classic advection-diffusion equation,

$$\partial C/\partial t = -(F * L/V) * \partial C/\partial x + D * \partial^2 C/\partial x^2 \quad .$$

Sources and Sinks:
We encounter the same sources and sinks in distributed models as in compartmental models: synthesis and decay, exchange with other regions, conversion of one chemical species to another.

Recall the complex problem of Section V:

$$K_{diss} = k_f / k_b.$$
$$k_f = K_{diss} * k_b.$$

$$dA_1/dt = (Flow/V_1)*(Ain - A_1) - (PS_A/V_1)*(A_1 - A_2);$$

$$dA_2/dt = (PS_A/V_2)*(A_1 - A_2) - k_f * A_2 + k_b * B_2;$$

$$dB_1/dt = (Flow/V_1)*(-B_1) - (PS_B/V_1)*(B_1 - B_2) - (G_{1B}/V_1)*B_1;$$

$$dB_2/dt = (PS_B/V_2)*(B_1 - B_2) + k_f * A_2 - k_b * B_2;$$

In a distributed model with axial diffusion, the last four equations are given by

$$\partial A_1/\partial t = -(Flow * L/V_1)*\partial A_1/\partial x - (PS_A/V_1)*(A_1 - A_2) + D_{A1}*\partial A_1^2/\partial x^2;$$

$$\partial A_2/\partial t = (PS_A/V_2)*(A_1 - A_2) - k_f * A_2 + k_b * B_2 + D_{A2}*\partial A_2^2/\partial x^2;$$
$$\partial B_1/\partial t = -(Flow * L/V_1)*\partial B_1/\partial x - (PS_B/V_1)*(B_1 - B_2) - (G_{1B}/V_1)*B_1 + D_{B1}*\partial B_1^2/\partial x^2;$$

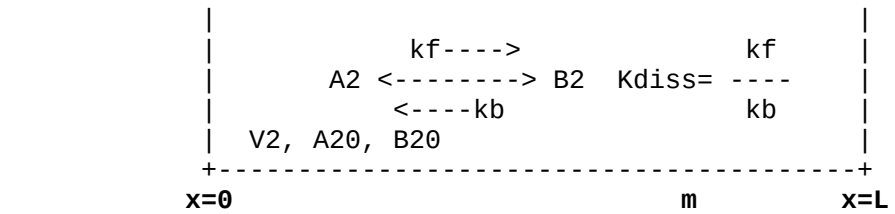
$$\partial B_2/\partial t = (PS_B/V_2)*(B_1 - B_2) + k_f * A_2 - k_b * B_2 + D_{B2}*\partial B_2^2/\partial x^2;$$

With compartment Models we had initial conditions and inflow and outflow were incorporated in the ordinary differential equations. With partial differential equations, we have initial conditions, and inflows, outflows, and regions with neither incorporate this information in boundary conditions on the left and right hand side of the regions. It is customary to place the inflowing material at the left boundary and outflowing material at the right boundary.

The boundary conditions will be covered in detail in the next section.

The main differences in the code are highlighted in boldface. Code for the PDE version is given as

```
iimport nsrunit; unit conversion on;
/*
    Source      +---------------------------------------+   Sinks
    Flow*A1in -->  A1, B1, V1, A10, B10                  --> Flow*A1
                |                 (Sink) -G1B/V1*B1      --> Flow*B1
                |                                        |
                |  Sink   Source    Sink     Source      |
                +--- v ----- ^ ------ V ----- ^ ---------+
                     |       |        |       |
                    Psa*A1 PSa*A2   PSb*B1   PSb*B2
                     |       |        |       |
                +--- v ----- ^ ------ V ----- ^ ---------+
                |  Source   Sink    Source   Sink        +
```




```
*/
math AllProcessesPDE {
real L=0.1 cm, Ngrid=31;
realDomain x cm; x.min=0; x.max=L; x.ct=Ngrid;
realDomain t s; t.min=0; t.max=20; t.delta=0.1;

// Dependent Variable and constants
real A1(x,t) mM, A2(x,t) mM, B1(x,t) mM, B2(x,t) mM, A1out(t) mM, B1out(t) mM,
     A10 = 0 mM, A20 = 0 mM, B10 = 0 mM, B20 = 0 mM,
     V1  = 1 ml, V2  = 2 ml,
     Flow   = 1 ml/s,      // Flow rate
     PSa    = 3 ml/s,      // Passive exchange rate for A
     PSb    = 3 ml/s,      // Passive exchange rate for B
     G1B    = 2 ml/s,      // Consumption rate for B in flowing compartment
     D1     = 1e-5 cm^2/s, // Axial diffusion in V1
     D2     = 1e-5 cm^2/s, // Axial diffusion in V2
     Kdiss = 3 dimensionless, //Dissociation constant
     kb = 4 sec^(-1), // Backwards rate constant
     kf sec^(-1);     // Forward rate constant
     kf = Kdiss*kb;      //
extern real A1in(t) mM; // A1in will be defined at run time using a
                        // function generator
// Initial Condition
when(t=t.min) {A1=A10; A2=A20; B1=B10; B2=B20;}
// Boundary conditions
when (x=x.min) {-(Flow*L/V1)*(A1-A1in)+D1*A1:x=0;
                -(Flow*L/V1)*(B1-0   )+D2*B1:x=0;
                D2*A2:x=0; D2*B2:x=0;}
when (x=x.max) {D1*A1:x=0; D1*B1:x=0;
                A1out=A1;  B1out=B1;
                D2*A2:x=0; D2*B2:x=0; }
// Partial Differential Equation
A1:t = (-Flow*L/V1)*A1:x +D1*A1:x:x + (PSa/V1)*(A2-A1)  ;
A2:t =                   +D2*A2:x:x + (PSa/V2)*(A1-A2)     -kf*A2+kb*B2;

B1:t = (-Flow*L/V1)*B1:x +D1*B1:x:x + (PSb/V1)*(B2-B1)     -(G1B/V1)*B1;
B2:t =                   +D2*B2:x:x + (PSb/V2)*(B1-B2)     +kf*A2-kb*B2;

// No analytic solutions are given for the general case. Compare
// Analytic solutions for special cases where they exist
// Making D1 and D2 large ( 0.2 cm^2/sec) will make this model behave like
// the compartmental version

} // End of model
```

The ODE and PDE versions are in the same project AllProcessesPDE.proj