Semi-automated Modular Program Construction for Physiological Modeling

By
Gary M. Raymond and
James B. Bassingthwaighte

MPC

- Designed to work with Jsim's Mathematical Modeling Language (MML).
- Processes FileName.mpc producing Filename.mod.
- Combines MML with directives which begin with //% .
- There are 13 directives.
- Directives control identification, fetching, relabeling code variables, parameters, and equations, combining equations into new equations.

Directives simple as 1, 2, & 3

- //%COM (comment not copied to model file
- //%START codeBlockName
- //%END codeBlockName

  Identifies a block of code in a file that is available to MPC. Usually stored in code libraries.

Code Libraries: INSERT 4 & 5

```
Built with MPC using
//%INSERTSTART codeBlockName
//%GET FileName codeBlockName( )
//%INSERTEND codeBlockName
e.g.
CodeLibrary.mpc contains
/*
//%INSERTSTART flowPDECalc
//%GET FlowPDE.mod flowDiffCalc()
//%INSERTEND flowPDECalc
*/

This code generates
```

java MPC CodeLibrary.mpc

```
CodeLibrary.mod which will contain
/*
//%START flowPDECalc
C:t = -(F*L/V)*C:x
      + D*C:x:x;
//%END flowPDECalc
*/
a JSim comment block when file FlowPDE.mod
contains

//%START flowDiffCalc
C:t = -(F*L/V)*C:x
      + D*C:x:x;
//%END flowDiffCalc
```

Alternatively DELETE (6 & 7)

- Instead of extracting code from existing models (just illustrated), the relevant code from other models or CodeLibraries can be appended to a .mpc file and surrounded by

  //%STARTDELETE

  … some code with START and END directives

  //%ENDDELETE

The CONTINUE directive (8)

- //%
- Used to continue the specification of the

  //%GET

  //%REPLACE and

  //%COLLECT directives.
- ALL CONTINUED DIRECTIVES ARE INTERNALLY CONVERTED INTO SINGLE LINES.

The GET directive (9)

- The GET directive gets a block of code from a file and relabels the names of the variables and parameters.
- `//%GET FileName codeBlockName (`
  `//% "oldName1=newName1", . . .`
  `//% "oldNameN=newNameN")`
  `oldName1 can include blanks, but EVERYTHING between the first quote and the equal sign must match.`
- `The GET directive can get code which also contains other directives`

Replacers and Replacements

- Two directives use replacers and replacements, the SETGLOBALVAL and REPLACE.
- Basic Format is

```
%replacer%=("replacement")
Example
//%REPLACE %F%=("Flow")
%F% = 1 ml/(g*min);
becomes
Flow = 1 ml/(g*min);
```

- A replacer can occur anywhere in the .mpc file, e.g. in another directive.

Two Kinds of Replacements (10)

- //%SETGLOBALVAL %replacer% =("replacement")
- Specifies only one replacer with only one replacement.
- The SETGLOBALVAL instructions are the first instructions executed. They apply to all the code in the .mpc file. They can occur anywhere in the .mpc file.
- Often used in conjunction with the //%REPLACE directive.

SETGLOBALVAL Example

```
/%SETGLOBALVAL %N%=("5")
//%REPLACE (%im1%=("#0#%N%-2"),%i%=("#1
//% #%N%-1"), %ip1%=("#2#%N%") )
dx = L/%N%;
C0:t=0;
C%i%:t = D*(C%im1%-2*C%i%+C%ip1%)/dx^2;
C%N%:t=0;

produces
```

```
dx=L/5;
C0:t=0;
C1:t = D*(C0-2*C1+C2)/dx^2;
C2:t = D*(C1-2*C2+C3)/dx^2;
C3:t = D*(C2-2*C3+C4)/dx^2;
C4:t = D*(C3-2*C4+C5)/dx^2;
C5:t = 0;
```

The REPLACE directive (11)

- The REPLACE directive is similar to the SETGLOBALVAL directive, except it can have multiple replacers and replacements.
- Multiple replacers must all have the same number of replacements.
- Each replacement causes the line containing the replacer to be duplicated. If the replacement is blank, no line duplication occurs.

```
//%REPLACE %S%=("A","B")
//%REPLACE %N%=("1","2")
%S%%N%
produces
A1
B1
A2
B2

//%REPLACE (%S%=("A","B"), %N%=("1","2") )
produces
A1
B2
```

The ENDREPLACE directive (12)

- //%ENDREPLACE is used to terminate the scope of the last //%REPLACE encountered.
- If any matching //%ENDREPLACE directives are omitted, they are internally placed at the end of the .mpc file.
- Note that the REPLACE directive can have limited scope, but the SETGLOBALVAL applies to the entire .mpc file.

The COLLECT directive (13)

- //%COLLECT("VariableName")

  collects all the equations which begin with

  VariableName = …

  and adds the right hand sides together.
- The usual form is

  //%COLLECT("VariableName:t") to produce new differential equations.

Order of Processing Directives

Process all SETGLOBALVAL directives in the file in the order in which they occur.

While there are remaining GET, REPLACE, and COLLECT directives, compress them into single lines of code.

Perform all REPLACE directives from most embedded to least embedded and from left to right for replacers and replacements.

Perform all GET directives from first to last; if GET directives have returned more GET, REPLACE, or COLLECT directives repeat until no more directives remain.

Removal of Duplicated lines of Code

Remove all duplicate lines of generated code starting with JSim's “math” declaration to the end of the file, or until a line containing DIAGRAM or DETAILED is reached. Duplicated lines must exactly match. Retain duplicate lines if they contain /*, //, or */ or if the first non-blank character is + or -.

For example,

```
A = if(C>D) C else
    0.0;
B = if(D>C) D else
    0.0;
```

produces

```
A = if(C>D) else
        0.0;
B = if(D>C) D else
```

However there are fixes: inserting leading blanks or adding a comment to the matching line.

Example: Input File eg.mpc

```
//%REPLACE %S%=("Ado","Ino")
//%REPLACE %R%=("p","i")
//%GET eg.mpc flowDCalc("C=%S%p","F=Flow","V=Vp",
//%                          "D=%S%Dp")
//%GET eg.mpc exchangeCalc("C1=%S%p","V1=Vp",
//%                   "PS=PS%S%","C2=%S%i","V2=Vi")
//%GET eg.mpc diffusionCalc("C=%S%i","D=%S%Di")
//%COLLECT("%S%%R%:t")
```

Example: Input File eg.mpc

```
//%STARTDELETE
//%START flowDCalc
C:t = -(F*L/V)*C:x
      + D*C:x:x;
//%END flowDCalc

//%START diffusionCalc
C:t = D*C:x:x;
//%END diffusionCalc

//%START exchangeCalc
C1:t = PS/V1*(C2-C1);
C2:t = PS/V2*(C1-C2);
//%END exchangeCalc
//%ENDDELETE
```

Example: Output in eg.mod file

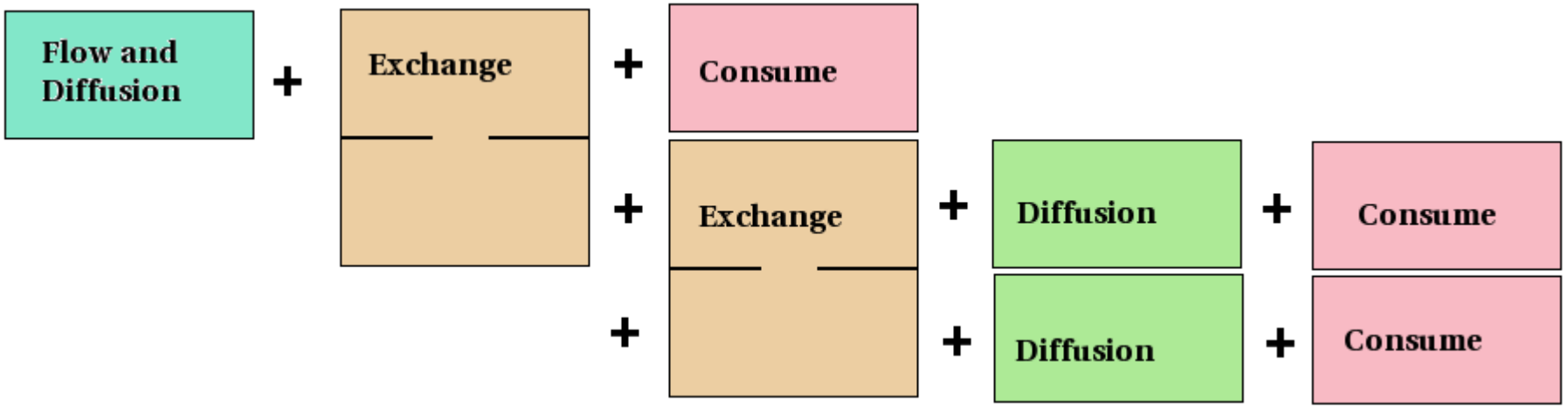
```
>java MPC eg.mpc
>cat eg.mod

Adop:t = -(Flow*L/Vp)*Adop:x
       + AdoDp*Adop:x:x
       + PSAdo/Vp*(Adoi-Adop);
Inop:t = -(Flow*L/Vp)*Inop:x
       + InoDp*Inop:x:x
       + PSIno/Vp*(Inoi-Inop);
Adoi:t = PSAdo/Vi*(Adop-Ado1)
       + AdoiDi*Ado1:x:x;
Inoi:t = PSIno/Vi*(Inop-Inoi)
       + InoiDi*Inoi:x:x;
// This MML file generated from eg.mpc using MPC.
```

ALL 13 Directives begin with //%

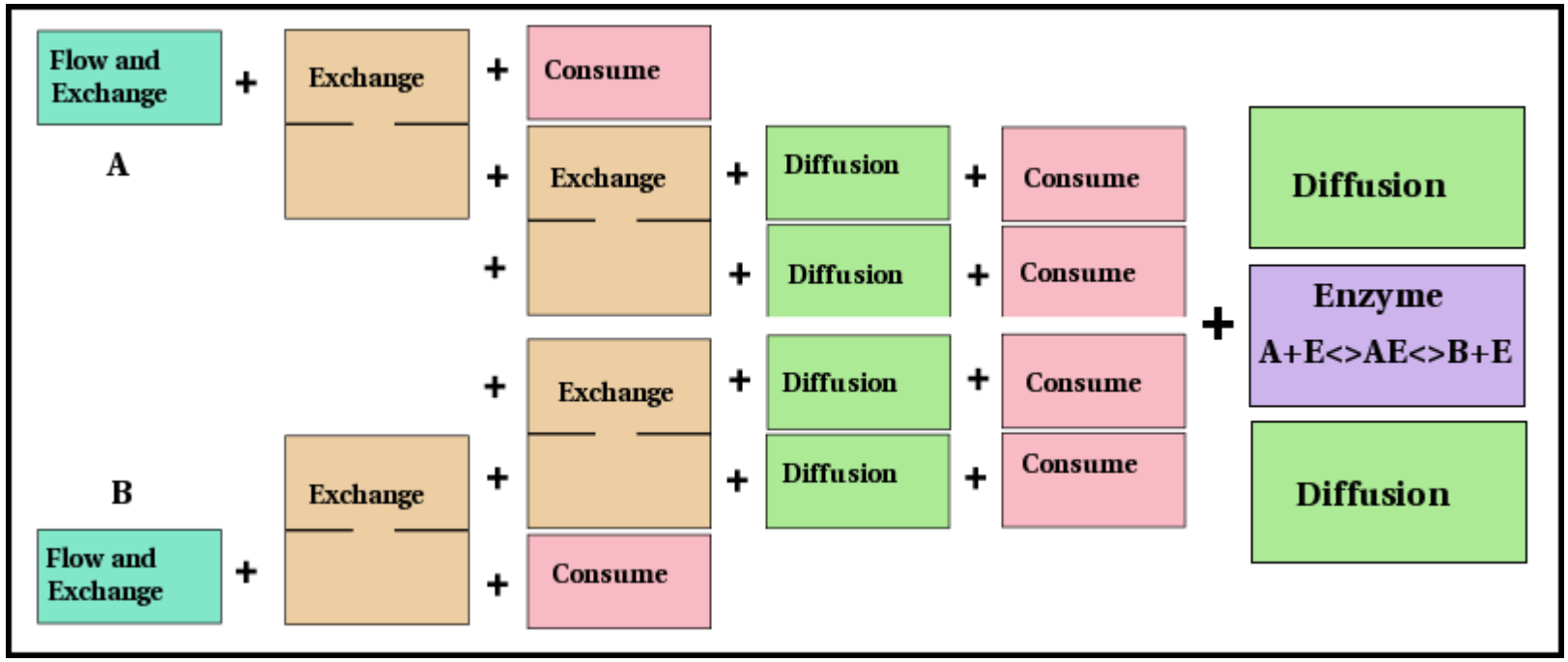
(1) **COM** for comments in .mpc NOT written to output. (2) **START** and (3) **END** for identifying code blocks. (4) **INSERTSTART** and (5)**INSERTEND** for marking code blocks. (6) **STARTDELETE** and (7) **ENDDELETE** primarily for removing code libraries appended to a .mpc file. (8) **CONTINUATION** directives used only for GET, REPLACE, and COLLECT directives. (9) **GET** for getting blocks of MML code and relabeling variables and parameters. (10) **SETGLOBALVAL** used for global setting of a value. (11) **REPLACE** used for duplicating lines of code. (12) **ENDREPLACE** for terminating scope of REPLACE directive. (13) **COLLECT** for making new equations.

Examples

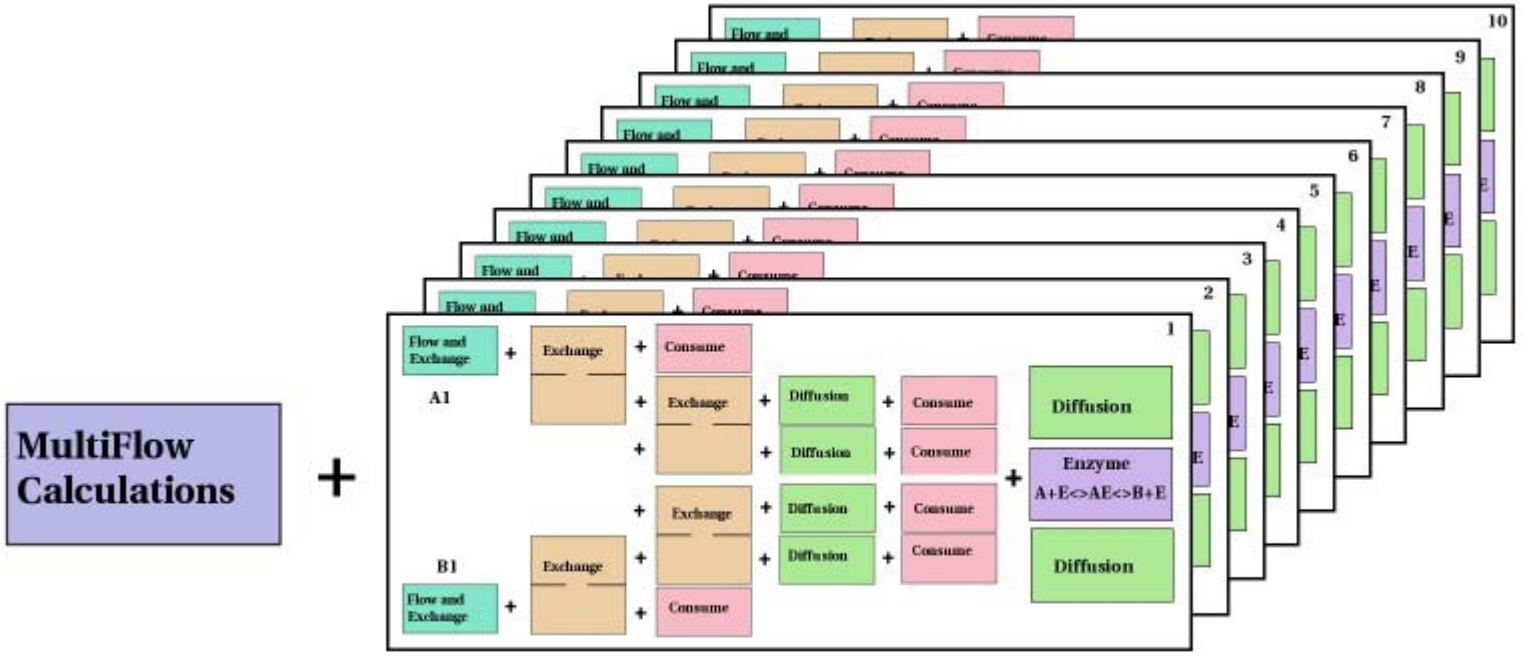
- Using 8 modules to build a three region blood tissue exchange model (BTEX30)



- Using 2 BTEX30 modules + 2 more diffusion modules + enzyme conversion module to make BTEX30A2B.



Using 10 copies of BTEX30A2B module + MultiFlow Calculations (flow heterogeneity) module to make BTEX30A2B10paths, simulating organ level kinetics.

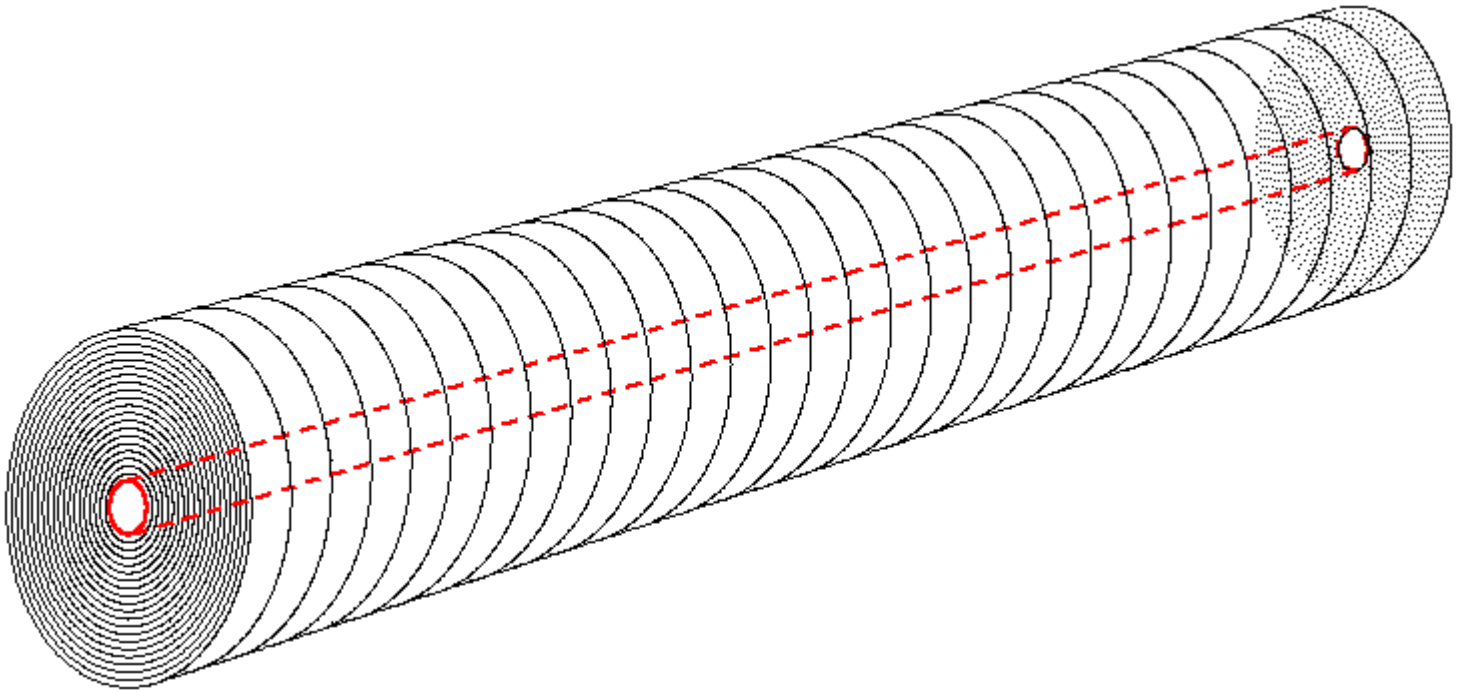


Lines in .mpc vs .mod

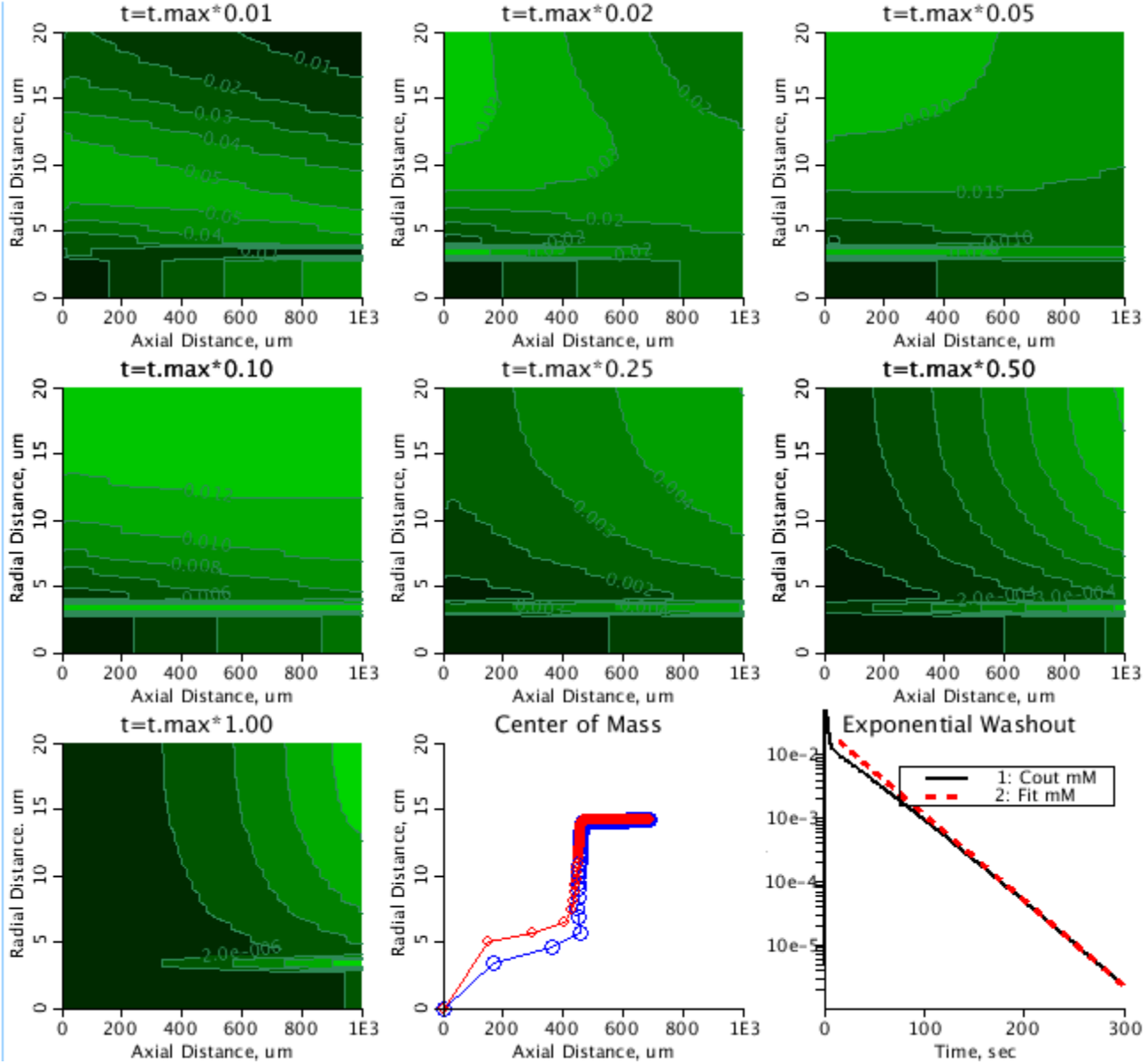
MODEL	LINES IN File.mpc	LINES in File.mod
BTEX30	47 (28)	93
BTEX30A2B	37 (17)	148
BTEX30A2B10paths	50 (19)	977

1-d Flow with 2-d diffusion

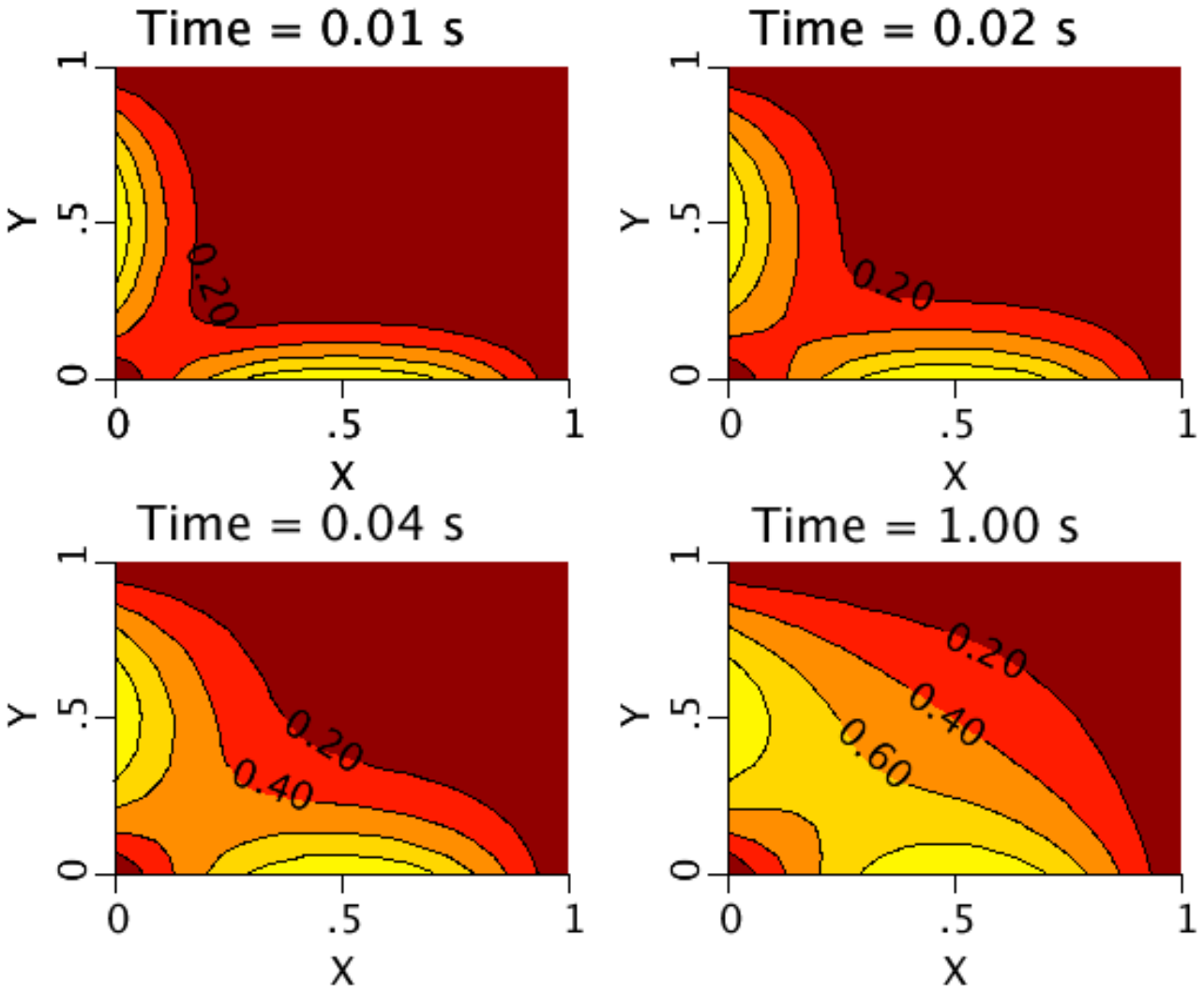
BTEX20 with radial and axial diffusion



Flow limited Case, t.max=300 sec



Solving the time-dependent 2-d Laplace's equation with Dirichlet boundary conditions



Comparison of two methods

MODEL	LINES IN File.mpc	LINES IN File.mod	RUNNING TIME
Laplace2DODE	26	1698	~31 seconds
Laplace2DPDE	26	135	~1 second

MORE DETAILS AT

physiome.org/jsim/models/webmodel/NSR/MPC/index.html

Questions?

garyr@uw.edu