MODIFICATIONS OF RATE.FOR

Tingyue Gu
gu@ohio.edu

Explanation for the one-dimensional concentration array u.

In the top section of the source code:

mnelemb = maximum number of finite element set by the programmer. In the input date file, nelement value must be smaller than this maximum value.

mnc = maximum number of interior collocation point. In the input date file, nc value must be smaller than this maximum value.

mnsp= maximum number of species. In the input date file, nsp value must be smaller than this maximum value.

mnndb=maximum. number of nodes used to discretize the z-axis for the bulk-fluid phase.
mnndb=2*mnelemb+1 (because quadratic elements are used)

mnnt=maximum number of nodes

nndb=2*nelemb+1= actual number of finite element nodes used to discretize the z-axis for the bulk-fluid phase. #1 node is at z=0 (column inlet).

nc=actual number of interior collocation points. The exterior collocation point (at r=1, or $R=R_p$) is collocation point number nc+1.

nnt=nndb+2*nndb*nc= total actual number of concentration (in the bulk-fluid phase and in the macropore fluid) points for a component. For each finite element node, there are nc number of concentration points in macropore fluid. The c_{pi} and c^*_{pi} values at the exterior collocation point, i.e., $(c_{pi})_{nc+1}$ and $(c^*_{pi})_{nc+1}$ values, are not included in 2*nndb*nc. They are calculated in the code using Eq. 3-30 on p. 15 (in my book) and Eq. 3-17 on p. 11. Obviously, $(c_{pi})_{nc+1}$ and $(c^*_{pi})_{nc+1}$ values are not included in array u.

In the main program, a one-dimensional array u is used to combine c_{bi}, c_{pi} and c^*_{pi} values. At each integration time point (for the ODE solver/integretor, i.e. DVODE), there is a new array of u. It is desirable to have different multidimensional matrices for c_{bi}, c_{pi} and c^*_{pi} values to make the code more transparent and easier to manage. However, these matrices cannot be passed to subroutines via their arguments unless their dimensions are fixed and declared explicitly (using numbers) in the main program. Since RATE.FOR allows for different nelement, nc values, etc. in the input data, these matrices cannot have fixed dimensions. To solve this problem, a one-dimensional array u is used to cover c_{bi}, c_{pi} and c^*_{pi} values. Array u always has a size of mntt (=mnsp*mnnt) number of values. Based on input data, ntt (=nsp*nnt) number of values in array u are actually useful. The rest of space in u are filled with zero. This one-dimensional array u with a fixed dimension (mnnt) is easily passed to subroutines. Once it is read by a subroutine, it is decomposed to give c_{bi}, c_{pi} and c^*_{pi} values in multidimensional arrays. The first few active lines of code in subroutine fcn perform this job.

Array u is divided into Component 1, Component 2, Component 3, ... Each component's section is subdivided into three subsections: c_{bi} values first, then c_{pi} values, then c^*_{pi} values. In array u, each component takes up nnt (=nndb+2*nndb*nc) number of values. Among them, there are nndb number of c_{bi} values, nndb*nc number of c_{pi} values and nndb*nc number of c^*_{pi} values. The actual space occupied by c_{bi}, c_{pi} and c^*_{pi} values for nsp number of components is ntt=nsp*nnt. The rest of the space (mntt minus ntt) in array u is filled with zero.

In the main program, array u is initialized so that all of its values are set to zero first. This means that initially, the column is free of any binding species. If your column initially is not "empty," you need to assign c_{bi}, c_{pi}, c^*_{pi} values to array u based on its internal structure.

Examples of u values:
u(1) = c_{bi} value at column inlet for component 1
u(2*nnt+1)=c_{bi} value at column inlet for component 3. (2*nnt for the first two components.)
u(nndb) = c_{bi} value at column exit for component 1.
u(nnt+nndb) = c_{bi} value at column exit for component 2.
u(nndb+1) = c_{pi} value for component 1 at column inlet for the first component. And it is at the first interior collocation point, the one next to r=1 (the particle surface).
u(nndb+nndb*nc+1)=c^*_{pi} corresponding to the u(nndb+1) = c_{pi} value. In u(nndb+nndb*nc+1), nndb takes care of all =c_{bi} values, nndb*nc takes care of all c_{pi} values.

As a matter of fact, concentrations of any component anywhere in the column (in the bulk-fluid, in the particle's macropore fluid, or in the stationary phase) at any time can be printed out. However, the values are too numerous. The current printout statements in the source code only prints out dimensionless effluent concentration profiles. They can be rewritten to print out any concentration values at any time of the chromatographic operation.

If the RATE.FOR Fortran source code you have does not contain an option to print out concentration profiles in the column bulk-phase at difference z-axis positions at different time, you can insert the following lines

```
      print*, 'Dimensionless bulk-phase concentrations inside column'
      print*, 'at different z-axis positions for dimensionless time t='
      write(*,1003) tend
      print*, '   z, c1, c2, c3, c4'
                  do 104      i = 1, nndb
                  write (*, 1003)   float(i-1)/float(nndb-1), u(i),
     #u(nnt+i), u(2*nnt+i), u(3*nnt+i)
104               continue
      goto 100
```

right before

```
if(nsp.eq.1) write(*,1003) tend,u(nndb)
if(nsp.eq.2) write(*,1003) tend,u(nndb),u(nnt+nndb)
if(nsp.eq.3) write(*,1003)tend,u(nndb),u(nnt+nndb),u(2*nnt+nndb)
```

Isotherm Section (Subroutine getdgdc) in the Source Code

If a different isotherm other than Langmuir isotherm (p. 11) or the stoichiometric ion-exchange isotherm (Eq. 3-19 on p. 12. Correction: last C_{0j} should be C_{0i}.), subroutine getdgdc (c, dgdc, nsp)
needs to be modified. This subroutine calculates $\partial g_i/\partial cp_j$ values for given c_p values for each component. In the subroutine, sf=1 for Langmuir isotherm. sf=0 for the stoichiometric ion-exchange isotherm.

$g_i=(1-\varepsilon_p)c^*_{pi}+\varepsilon_p c_{pi}$ in which c^*_{pi} carries the isotherm expression. Or, we can write,

$$g_i = (1-\varepsilon_p)\frac{a_i c_{pi}}{1+\sum_{n=1}^{nsp} b_n c_{pn} C_{0n}} + \varepsilon_p c_{pi}$$

For $i \neq j$,

$$\frac{\partial g_i}{\partial c_{pj}} = (1-\varepsilon_p) a_i c_{pi} \frac{-b_j C_{0j}}{(1+\Sigma)^2} + 0$$

For i=j,

$$\frac{\partial g_i}{\partial c_{pj}} = (1-\varepsilon_p)\frac{a_i \cdot 1 \cdot (1+\Sigma) - a_i c_{pi} b_j C_{0j}}{(1+\Sigma)^2} + \varepsilon_p$$ where Σ is the summation term in the g_i expression above.

In the subroutine, cf(j) is the maximum feed concentration of component j. It is the same as C_{0j}. ssum=Σ. epsip=ε_p. consta(i) = a_i, constb(j)=b_j for the Langmuir isotherm.

If you have a different isotherm, follow this example to write down $\partial g_i/\partial cp_j$ expressions for $i \neq j$ and i=j situations and then change the subroutine accordingly. You must fully understand how Langmuir isotherm is implemented in the subroutine first before you start modifying it. If this section is coded wrong, the dimensionless effluent concentration profiles calculated will not satisfy "mass balance" meaning the dimensionless area under a peak is not equal to its τ_{imp} value. You can download a small windows program in my chromatography web page to calculate peak areas from raw data generated from RATE.EXE. If the peak areas always satisfy mass balance, chances are your modification of the subroutine is right. The utility calculates peak areas one peak at a time. For multicomponent data, you need to use Excel to generate time-concentration files for individual components.

Changing Column Feed Profiles

This program allows you have any feed profiles (concentration time courses). However, you need to express them in the subroutine fcn (nttttt, t, u, uprime). The existing relevant section is shown below.

```
c      apply NBC. c2=-1  in flux=c1*U+c2   at node #1   !!!
c      no need to restore afb later, cuz, afb is set 0 at start
c      BTW, c1 was already applied to akb in subr. bulk
c      if #ns species' feed is zero, no change here, goto do 233
      if(cf(ns).eq.0.0d0) goto 233

        if(index.eq.1) afb(1) = afb(1) + (+1.0d0)
      if(index.eq.2) then
            if(t.le.timp)  afb(1) = afb(1) + 1.0d0
      endif

c        --- elution again, but the mobile phase #nsp is absorbable
      if(index.eq.6) then
            if(ns.eq.nsp) afb(1) = afb(1) + 1.0d0
            if(ns.lt.nsp.and.t.le.timp)  afb(1) = afb(1) + 1.0d0
      elseif(index.eq.7) then
            if(ns.eq.nsp.and.t.gt.timp) afb(1) = afb(1) + 1.0d0
            if(ns.lt.nsp.and.t.le.timp)  afb(1) = afb(1) + 1.0d0
      endif

c      --- displacement.
c     the last species #nsp is the displacer
      if (index.eq.3.and.nsp.gt.1) then
            if(ns.eq.nsp) afb(1) = afb(1) + 1.0d0
      endif

c      --- BT then shift to displ.
      if(index.eq.4.or.index.eq.5) then
          if(nsp.gt.1) then
            if(t.le.tshift.and.ns.ne.nsp) afb(1) = afb(1) + 1.0d0
            if(t.gt.tshift.and.ns.eq.nsp) afb(1) = afb(1) +1.0d0
          elseif(nsp.eq.1) then
            if(t.le.tshift) afb(1) = afb(1) + 1.0d0
          endif
      endif
```

The feed profiles must be written in dimensionless concentration form (as a function of dimensionless time which is t in the code section above). In the code section above, the dimensionless feed concentration of a component is either 0 or 1. We are talking about only step changes (rectangular pulses) here. If the feed concentration is 0, nothing needs to be done. If it is 1, afb(1) value increases by 1. The afb array is a one-dimensional array in the finite element discretization of z-axis for the bulk-fluid phase PDE. afb(1) is the afb value at the column inlet because the #1 finite element node is at the column inlet.

Example:
You want to have a dimensionless concentration profile for component 2 (ns=2) like this,
$C_{f2}(\tau)/C_{02} = a1 + a2\cdot(\tau-\tau_{imp}) + a3\cdot(\tau-\tau_{imp})^2 + a4\cdot(\tau-\tau_{imp})^3 + a5\cdot\exp[a6\cdot(\tau-\tau_{imp})]$
between dimensionless time frame st1 < τ< st2, and $C_{f2}(\tau)/C_{02} = 0$, other otherwise. You can use index=8 in the input data file to signal this. a1,...,a6, st1, st2 values must be in the input data file and be read by the main program and passed to subroutine fcn. You can use something like this to pass the values:

```
      common /gradient/ a1, a2, a3, a4, a5, a6, st1, st2
```

You need to add the following lines in the code section above.

```
c            -- nonlinear feed profile for component 2. Index=8
      if(ns.eq.2.and.t.gt.st1.and.t.lt.st2)  then
      tt = t -timp
      afb(1) = afb(1) + a1 + a2*tt + a3*tt*tt+ a4*tt*tt*tt + a5*dexp(a6*tt)
```

The code statement `“(index.eq.8.and.ns.eq.2.and.t.gt.st1.and.t.lt.st2)”` means the component number is 2, and st1 < τ< st2 for operation index 8.

No action needs to be taken when the dimensionless feed concentration for component 2, ($C_{f2}(\tau)/C_{02}$, is 0.

Compilation Using Microsoft FORTRAN PowerStation V.4.0 (Windows 95)
The following message will be generated:

```
--------------------Configuration: rate - Win32 Debug--------------------
Compiling Fortran...
C:\Temp1\rate.for
C:\Temp1\rate.for(568): warning FOR4269: unused dummy argument RPAR
C:\Temp1\rate.for(568): warning FOR4269: unused dummy argument NTTTTT
C:\Temp1\rate.for(568): warning FOR4269: unused dummy argument IPAR
C:\Temp1\rate.for(3705): warning FOR4269: unused dummy argument VSAV
C:\Temp1\rate.for(4251): warning FOR4269: unused dummy argument IDUM
C:\Temp1\rate.for(4319): warning FOR4269: unused dummy argument NERR
C:\Temp1\rate.for : warning FOR4227: argument RTOL (number 7) in reference to
procedure DVODE from procedure main incorrect: is not an array
C:\Temp1\rate.for : warning FOR4227: argument ATOL (number 8) in reference to
procedure DVODE from procedure main incorrect: is not an array
C:\Temp1\rate.for : warning FOR4227: argument JAC (number 16) in reference to
procedure DVODE from procedure main incorrect: has the wrong data type
C:\Temp1\rate.for : warning FOR4227: argument RPAR (number 18) in reference to
procedure DVODE from procedure main incorrect: is not an array
C:\Temp1\rate.for : warning FOR4227: argument IPAR (number 19) in reference to
procedure DVODE from procedure main incorrect: is not an array
C:\Temp1\rate.for : warning FOR4227: argument MSG (number 1) in reference to
procedure XERRWD from procedure DVODE incorrect: is not an array
C:\Temp1\rate.for : warning FOR4227: argument MSG (number 1) in reference to
procedure XERRWD from procedure DVINDY incorrect: is not an array
Linking...
rate.exe - 0 error(s), 13 warning(s)
```

All these warnings are harmless. They refer to undeclared arrays or unused variables used in the ODE solve, DVODE. I did not bother to correct them.

Compilation Using Microsoft FORTRAN PowerStation V.1.0a (Windows 3.1)
The following message will be generated:

```
Initializing...
Compiling...
Microsoft (R) FORTRAN PowerStation Optimizing Compiler Version 1.0
Copyright (c) Microsoft Corp 1984-1993. All rights reserved.
```

```
C:\TEMP1\RATE.FOR
C:\TEMP1\RATE.FOR(568) : warning F4202: FCN : formal argument NTTTTT : never
used
C:\TEMP1\RATE.FOR(568) : warning F4202: FCN : formal argument RPAR : never
used
C:\TEMP1\RATE.FOR(568) : warning F4202: FCN : formal argument IPAR : never
used
C:\TEMP1\RATE.FOR(1919) : warning F4016: DVODE : formal argument JAC : type
mismatch
C:\TEMP1\RATE.FOR(3705) : warning F4202: DVNLSD : formal argument VSAV : never
used
C:\TEMP1\RATE.FOR(3705) : warning F4202: DVNLSD : formal argument PDUM : never
used
C:\TEMP1\RATE.FOR(4251) : warning F4202: D1MACH : formal argument IDUM : never
used
C:\TEMP1\RATE.FOR(4319) : warning F4202: XERRWD : formal argument NERR : never
used
Linking...
Microsoft (R) 32-Bit Executable Linker Version 1.0F
Copyright (C) Microsoft Corp 1992-93. All rights reserved.

-machine:i386 -base:0x00010000 -subsystem:console -entry:mainCRTStartup
-debug:full -debugtype:cv
RATE.OBJ
-out:RATE.EXE
libf.lib
libc.lib
kernel32.lib
ntdll.lib
Binding...
bindmsf2: MSOFT1 -- Copyright (C) 1986-93 Phar Lap Software, Inc.

Replacing .EXE stub in application: RATE.EXE with one
from file: C:\F32\BIN\bindmsf2.exe.

INPUT FILES:
    C:\F32\BIN\bindmsf2.exe        (stub .EXE at offset 47599) (16512 bytes)
    RATE.EXE                       (409528 bytes)

OUTPUT FILE:
    RATE.exe                       (426552 bytes)
Microsoft (R) FORTRAN PowerStation (MS-DOS) Fix for Windows 95
Copyright (C) Microsoft Corp. 1995.  All Rights reserved.

RATE.EXE -- Fixed.
RATE.EXE - 0 error(s), 8 warning(s)
```

(Nov. 1999)