# APPENDIX B

## Modifications and New Source Code for MT3D

Two of the existing source files were modified (note: all modifications are labeled with a c gp start and c gp end within the code):

- mt3dms4.for
- mt_ssm4.for

Three new source files were created:

- ade.for
- erfc.for
- river_flow.for

### MT3DMS4.FOR - Main Program

Lines 76 – 83:

```
c gp start
c added additional variables for river calcualtions
      real              :: wash_flow(0:1000),wash_up_conc
      real              :: conc_river_outlet,conc_aq_output
      real              :: dec_par(3,5)
      character(len=18) :: dum
      integer           :: imont,igp,jgp
c gp end
```

Lines 117 -136:

```
c gp start
c open required files
        open(210,file='monte.dat',status='old')
        read(210,*) imont
        close(210)
        write(dum,888) imont
888     format('output\outlet.',i4.4)
        open(211,file=dum,status='unknown')
        open(212,file='wash.in',status='old')
c read ade input data
c paleochannel section first
        do jgp=1,3
         do igp=1,5
          read(212,*) dec_par(jgp,igp)
         enddo
        enddo
      read(212,*) wash_flow(0)
      read(212,*) wash_up_conc
      close(212)
c gp end
```

Lines 406 – 410:

```
c gp start
c calculating river flows at each river cell
      call river_flow(x(lcss),x(lcdelr),x(lcdelc),x(lcdh),ncol,nrow,
     1                nlay,ntss,mxss,wash_flow)
c gp end
```

Lines 516 – 525:

```
c gp start
      IF(TRNOP(3) .AND. ICOMP.LE.MCOMP)
     & CALL SSM4FM(NCOL,NROW,NLAY,NCOMP,ICOMP,IX(LCIB),
     & X(LCDELR),X(LCDELC),X(LCDH),IX(LCIRCH),X(LCRECH),X(LCCRCH),
     & IX(LCIEVT),X(LCEVTR),X(LCCEVT),MXSS,NTSS,X(LCSS),X(LCSSMC),
     & X(LCQSTO),X(LCCNEW),ISS,X(LCA),X(LCRHS),NODES,UPDLHS,MIXELM,
     & wash_flow,wash_up_conc,
     & conc_river_outlet,conc_aq_output,
     & dec_par,time2)
c gp end
```

Lines 596 – 600:

```
c gp start
c save concentrations at outlet to unit 211
      write(211,3011) time2,conc_river_outlet,conc_aq_output
3011  format(f8.2,1x,4(e15.7,1x))
c gp end
```

MT_SSM4.FOR - SUBROUTINE SSM4FM

(note: line number refer to file line numbers not individual subroutine line numbers)

Lines 600 – 607:

```
c gp start
c initial new variables for river concentration calculations
      real    :: wash_flow(0:1000),wash_up_conc,wash_conc(0:1000)
      real    :: conc_river_outlet,conc_aq_output,qsss
      real    :: dec_par(3,5)
      real    :: time2,c1,c2,c3
      integer :: icnt
c gp end
```

Lines 663 – 739:

```
c gp start
c calculate concentrations on flux boundaries based on ade solution
c note that c1 is paleochannel conc and c2 is pond conc

c calculate river concentrations
   20 continue
c     reach 2
       call ade(dec_par(1,1),dec_par(1,2),dec_par(1,3),dec_par(1,4),
     1          dec_par(1,5),time2,c1)
c     reach 3
       call ade(dec_par(2,1),dec_par(2,2),dec_par(2,3),dec_par(2,4),
     1          dec_par(2,5),time2,c2)
c     reach 4
```

```
       call ade(dec_par(3,1),dec_par(3,2),dec_par(3,3),dec_par(3,4),
     1          dec_par(3,5),time2,c3)
      icnt=0
      wash_conc(0)=wash_up_conc
      DO NUM=1,NTSS
        K=SS(1,NUM)
        I=SS(2,NUM)
        J=SS(3,NUM)
        QSS=SS(5,NUM)
        qsss=ss(5,num)*DELR(J)*DELC(I)*DH(J,I,K)
        IQ=SS(6,NUM)
        if(IQ==4) then
         icnt=icnt+1
c calculate wash concentration for river cells only
c   flow from aquifer to river (use mixing equation)
c   note that ctmp is not used in this case
          if(QSS<0.0) then
           if(j==69.and.i==74) then
            wash_conc(icnt)=(wash_flow(icnt-1)*wash_conc(icnt-1)+
     1                   abs(qsss)*cnew(j,i,k,1)+138240.*20.)/
     2                   wash_flow(icnt)
           else
            wash_conc(icnt)=(wash_flow(icnt-1)*wash_conc(icnt-1)+
     1                          abs(QSSS)*cnew(j,i,k,1))/
     2                      wash_flow(icnt)
           endif
          else
c   flow from river to aquifer (use upstream concentration)
           wash_conc(icnt)=wash_conc(icnt-1)
           ctmp=wash_conc(icnt)
          endif
c save downstream river and aquifer concentrations for printing in main
          conc_river_outlet=wash_conc(icnt)
        conc_aq_output=cnew(j,i,k,1)
         else
c concentration for flux boundaries
          if(ss(4,num)>49000..and.ss(4,num)<51000.) then
c paleochannels
           ctmp=c1
        elseif(ss(4,num)>999..and.ss(4,num)<1001.) then
c pond section
           ctmp=c2
          elseif(ss(4,num)>49..and.ss(4,num)<51.) then
c central section
           ctmp=c3
        else
c western boundary -- constant 10 micrograms per liter
         ctmp=ss(4,num)
        endif
         endif

        IF(NCOMP.GT.1) CTMP=SSMC(ICOMP,NUM)

        IF(IQ.EQ.15) QSS=1./(DELR(J)*DELC(I)*DH(J,I,K))
        IF(ICBUND(J,I,K,ICOMP).GT.0.AND.IQ.GT.0) THEN
          N=(K-1)*NCOL*NROW+(I-1)*NCOL+J
          IF(QSS.LT.0) THEN
```

```
            IF(UPDLHS) A(N)=A(N)+QSS*DELR(J)*DELC(I)*DH(J,I,K)
          ELSE
            RHS(N)=RHS(N)-QSS*CTMP*DELR(J)*DELC(I)*DH(J,I,K)
          ENDIF
        ENDIF
      ENDDO
c gp end
```

## ADE.FOR

```
      subroutine ade(L,v,dl,c1,c0,t,c)
        real      :: L
        t1=(L-v*t)/sqrt(4.0*dl*t)
        c=0.5*erfc(t1)*(c1-c0)+c0
      return
      end subroutine ade
```

## ERFC.FOR – Taken from Numerical Recipes

```
      FUNCTION erfc(x)
      REAL erfc,x
CU    USES gammp,gammq
      REAL gammp,gammq
      if(x.lt.0.)then
        erfc=1.+gammp(.5,x**2)
      else
        erfc=gammq(.5,x**2)
      endif
      return
      END
      FUNCTION gammp(a,x)
      REAL a,gammp,x
CU    USES gcf,gser
      REAL gammcf,gamser,gln
      if(x.lt.0..or.a.le.0.)pause 'bad arguments in gammp'
      if(x.lt.a+1.)then
        call gser(gamser,a,x,gln)
        gammp=gamser
      else
        call gcf(gammcf,a,x,gln)
        gammp=1.-gammcf
      endif
      return
      END
      FUNCTION gammq(a,x)
      REAL a,gammq,x
CU    USES gcf,gser
      REAL gammcf,gamser,gln
      if(x.lt.0..or.a.le.0.)pause 'bad arguments in gammq'
      if(x.lt.a+1.)then
        call gser(gamser,a,x,gln)
        gammq=1.-gamser
      else
```

```
        call gcf(gammcf,a,x,gln)
        gammq=gammcf
      endif
      return
      END
      SUBROUTINE gcf(gammcf,a,x,gln)
      INTEGER ITMAX
      REAL a,gammcf,gln,x,EPS,FPMIN
      PARAMETER (ITMAX=100,EPS=3.e-7,FPMIN=1.e-30)
CU    USES gammln
      INTEGER i
      REAL an,b,c,d,del,h,gammln
      gln=gammln(a)
      b=x+1.-a
      c=1./FPMIN
      d=1./b
      h=d
      do 11 i=1,ITMAX
        an=-i*(i-a)
        b=b+2.
        d=an*d+b
        if(abs(d).lt.FPMIN)d=FPMIN
        c=b+an/c
        if(abs(c).lt.FPMIN)c=FPMIN
        d=1./d
        del=d*c
        h=h*del
        if(abs(del-1.).lt.EPS)goto 1
11    continue
      pause 'a too large, ITMAX too small in gcf'
1     gammcf=exp(-x+a*log(x)-gln)*h
      return
      END
      SUBROUTINE gser(gamser,a,x,gln)
      INTEGER ITMAX
      REAL a,gamser,gln,x,EPS
      PARAMETER (ITMAX=100,EPS=3.e-7)
CU    USES gammln
      INTEGER n
      REAL ap,del,sum,gammln
      gln=gammln(a)
      if(x.le.0.)then
        if(x.lt.0.)pause 'x < 0 in gser'
        gamser=0.
        return
      endif
      ap=a
      sum=1./a
      del=sum
      do 11 n=1,ITMAX
        ap=ap+1.
        del=del*x/ap
        sum=sum+del
        if(abs(del).lt.abs(sum)*EPS)goto 1
11    continue
      pause 'a too large, ITMAX too small in gser'
1     gamser=sum*exp(-x+a*log(x)-gln)
```

```
      return
      END
      FUNCTION gammln(xx)
      REAL gammln,xx
      INTEGER j
      DOUBLE PRECISION ser,stp,tmp,x,y,cof(6)
      SAVE cof,stp
      DATA cof,stp/76.18009172947146d0,-86.50532032941677d0,
     *24.01409824083091d0,-1.231739572450155d0,.1208650973866179d-2,
     *-.5395239384953d-5,2.5066282746310005d0/
      x=xx
      y=x
      tmp=x+5.5d0
      tmp=(x+0.5d0)*log(tmp)-tmp
      ser=1.000000000190015d0
      do 11 j=1,6
        y=y+1.d0
        ser=ser+cof(j)/y
11    continue
      gammln=tmp+log(stp*ser/x)
      return
      END
```

## RIVER_FLOW.FOR

```
      subroutine river_flow(ss,delr,delc,dh,ncol,nrow,nlay,
     1                      ntss,mxss,wash_flow)
c subroutine to read in wash flow directions and
c to calculate wash flows in all river cells
c notes: 1. positive ss(5,i) represents flow from river to aquifer (i.e.
lossing reach)
c           2. negative ss(5,i) represents flow from aquifer to river (i.e.
gaining reach)
c        3. ss(1,i) = river layer number (z)
c        4. ss(2,i) = river row number (y)
c        5. ss(3,i) = river column number (x)
c          6. ss(4,i) = river concentration, note that this will be updated
within SSM4FM subroutine
c        7. ss(5,i) = volumetric flow rate from river to aquifer
c         8. ss(6,i) = integer flag noting source/sink type, river cells must
be = "4"
c        9. wash_flow = array of wash volumetric flows (cubic feet per day)
c            10. wash_up_conc = upstream wash concentration (micrograms per
liter)
c        11. iwash = number of river cells in modflow model
c        12. current version allows for a maximum of 1000 river cells
c        13. wash_index: 1=reach #, 2=row#(y),3=col#,4=layer#
c
c written by: greg pohll - desert research institute
c
c date: July 9, 2003
c
c initialize variables
      real    :: ss(6,mxss)
      real    :: delr(ncol)
      real    :: delc(nrow)
      real    :: dh(ncol,nrow,nlay)
      real    :: wash_flow(0:1000)
      integer :: wash_index(3,1000)
      icnt=0
      do num=1,ntss
       K=SS(1,NUM)
       I=SS(2,NUM)
       J=SS(3,NUM)
       if(ss(6,num)==4) then
        icnt=icnt+1
        VOLAQU=DELR(J)*DELC(I)*DH(J,I,K)
        if(j==69.and.i==74) then
         wash_flow(icnt)=wash_flow(icnt-1)-ss(5,num)*volaqu+138240.
        else
         wash_flow(icnt)=wash_flow(icnt-1)-ss(5,num)*volaqu
        endif
       endif
      enddo
10    continue
      return
      end
```