Code Edits and General Use

The following supplementary information documents all code changes made to the original SWAT source code (revision 664) in order to incorporate alternative soil water routing routines. All changes in the source code are denoted by initials GWP. Copies of the modified executable (Windows or Linux) or modified source code are available upon request by emailing gpignotti@gmail.com.

General approach

The main focus of modifying the SWAT code was to incorporate alternative equations to calculate soil water percolation based off the Campbell and van Genuchten approximations of hydraulic conductivity as described in the body of the paper. Additionally, this necessitated: 1) calculating the parameters for the equations based on soil properties, 2) creating an hourly loop for percolation, 3) better constraining maximum and minimum percolation and soil water content, and 4) printing new output files.

In general the approach is summarized as:

```
Time loop [1,24]
        Layer loop [1,number layers]
                Soil storage = Soil storage + Percolation from layer above
                If Soil storage > Wilting Point
                        Calculate Lateral flow and Percolation (user selected equation)
                        Soil storage = Soil storage - Percolation - Lateral flow
                        Redistribute any soil water above saturation
                        Check Soil storage is within maximum and minimum values
                End If
        Next Layer
Next Time step
```

Files modified

File	Brief Summary of Changes
allocate_parms.f	Create new variables for new equations and new output files
header.f	Add new column headers for HRU output file
hruaa.f	Record average annual HRU volumetric soil water content
hruday.f90	Record daily HRU volumetric soil water content
hrumon.f	Record monthly HRU volumetric soil water content
hruyr.f	Record yearly HRU volumetric soil water content
modparm.f	Allocate and provide dimensions of new variables
percmain.f	Create time loop, modify how percmicro is called, add min/max check
percmicro.f	Add new soil water percolation equations
readbsn.f	Create new flag that determines which soil water percolation equation is used and sets exponential parameter value multiplicative factor
readfile.f	Create new output files for soil water variables and flag to turn printing on or off
sat_excess.f	Change the code to better handle saturated soil layers
soil_phys.f	Calculate the soil parameters for new soil percolation equations
subbasin.f	Modify the call structure to accommodate new saturation excess calculations
sumv.f	Calculate volumetric soil water content for each layer and store the values
surface.f	Modify the call structure to accommodate new saturation excess calculations
writed.f	Write new soil water variables to new output files
zero2.f	Zero the new soil water variables

Edits to allocate_parms.f

Added new variables for soil water storage and new equations:

```
!!  SWC edits by GWP
!new variables for soil water storage
allocate(sol_dg(mlyr,mhru))
allocate(sol_sep(mlyr,mhru))
allocate(sol_lat(mlyr,mhru))
allocate(infl_print(mlyr,mhru))
allocate(sol_sepp(mlyr,mhru))
allocate(sol_exw(mlyr,mhru))
allocate(sol_ule(mlyr,mhru))
!PTF parameters
allocate(wo_thr(mlyr,mhru))
allocate(rb_thr(mlyr,mhru))
allocate(ca_rb_b(mlyr,mhru))
allocate(vg_rb_m(mlyr,mhru))
allocate(ca_co_b(mlyr,mhru))
allocate(vg_co_m(mlyr,mhru))
allocate(ca_sx_b(mlyr,mhru))
allocate(vg_sx_m(mlyr,mhru))
allocate(ca_wo_b(mlyr,mhru))
allocate(vg_wo_m(mlyr,mhru))

allocate(vswc(mlyr+1,mhru))
!!End GWP edits
```

Change initialized variable that stores the number of possible HRU outputs to include new volumetric soil water content values:

```
!GWP edit
mhruo = 89
!end GWP edit
```

Edits to header.f

Added additional headers for printing new volumetric soil water content values:

```
!!    column headers for HRU output file - edited by GWP
      heds = (/"  PRECIPmm"," SNOFALLmm"," SNOMELTmm","     IRRmm",
     &         "     PETmm","      ETmm"," SW_INITmm","  SW_ENDmm",
     &         …,
     &         " VSM1mm/mm"," VSM2mm/mm"," VSM3mm/mm"," VSM4mm/mm"," VSM5mm/mm",
     &         " VSM6mm/mm"," VSM7mm/mm"," VSM8mm/mm"," VSM9mm/mm", "VSM10mm/mm")
```

Edits to hruaa.f

Store average annual HRU volumetric soil water content:

```
# GWP edits
        pdvas(80) = hruaao(73,j) / 365.4
        pdvas(81) = hruaao(74,j) / 365.4
        pdvas(82) = hruaao(75,j) / 365.4
        pdvas(83) = hruaao(76,j) / 365.4
```

```
        pdvas(84) = hruaao(77,j) / 365.4
        pdvas(85) = hruaao(78,j) / 365.4
        pdvas(86) = hruaao(79,j) / 365.4
        pdvas(87) = hruaao(80,j) / 365.4
        pdvas(88) = hruaao(81,j) / 365.4
        pdvas(89) = hruaao(82,j) / 365.4

# end GWP edits
```

Edits to hruday.f90

Store daily HRU volumetric soil water content:

```
!!    GWP edits to print volumetric soil water content
      pdvas(80) = vswc(1,j)
      pdvas(81) = vswc(2,j)
      pdvas(82) = vswc(3,j)
      pdvas(83) = vswc(4,j)
      pdvas(84) = vswc(5,j)
      pdvas(85) = vswc(6,j)
      pdvas(86) = vswc(7,j)
      pdvas(87) = vswc(8,j)
      pdvas(88) = vswc(9,j)
      pdvas(89) = vswc(10,j)

!!    #End GWP edits
```

Modify output printing format:

```
!! SWC edits by GWP
3333    format(i5,1x,i5,1x,i2,1x,i2,1x,i4,1x,i1,1x,f8.3)
4447    format (i5,1x,i5,1x,i3,1x,f7.1,1x,f8.3,1x,f8.3,1x,f8.3)
!!    End GWP edits
```

Edits to hrumon.f

Store monthly HRU volumetric soil water content:

```
# GWP edits
        pdvas(80) = hrumono(73,j) / Real(days)
        pdvas(81) = hrumono(74,j) / Real(days)
        pdvas(82) = hrumono(75,j) / Real(days)
        pdvas(83) = hrumono(76,j) / Real(days)
        pdvas(84) = hrumono(77,j) / Real(days)
        pdvas(85) = hrumono(78,j) / Real(days)
        pdvas(86) = hrumono(79,j) / Real(days)
        pdvas(87) = hrumono(80,j) / Real(days)
        pdvas(88) = hrumono(81,j) / Real(days)
        pdvas(89) = hrumono(82,j) / Real(days)

# end GWP edits
```

Edits to hruyr.f

Store yearly HRU volumetric soil water content:

```
# GWP edits
        pdvas(80) = hruyro(73,j) / Real(366 - leapyr)
        pdvas(81) = hruyro(74,j) / Real(366 - leapyr)
        pdvas(82) = hruyro(75,j) / Real(366 - leapyr)
        pdvas(83) = hruyro(76,j) / Real(366 - leapyr)
        pdvas(84) = hruyro(77,j) / Real(366 - leapyr)
        pdvas(85) = hruyro(78,j) / Real(366 - leapyr)
        pdvas(86) = hruyro(79,j) / Real(366 - leapyr)
        pdvas(87) = hruyro(80,j) / Real(366 - leapyr)
        pdvas(88) = hruyro(81,j) / Real(366 - leapyr)
        pdvas(89) = hruyro(82,j) / Real(366 - leapyr)

# end GWP edits
```

Edits to modparm.f

Specify the type, dimension, and allocation of new soil water variables:

```
      !!GWP edits for soil water changes
      integer :: iperc, iwriteperc ! flags
      ! can use these for debugging
      real, dimension(:,:), allocatable :: sol_sepp, sol_exw, sol_ule
      real, dimension (:,:), allocatable :: sol_dg, sol_sep, sol_lat
      real, dimension (:,:), allocatable :: infl_print
      !Parameters for PTFs
      real, dimension(:,:), allocatable :: wo_thr,
     &  rb_thr, ca_rb_b, vg_rb_m, ca_co_b, vg_co_m,
     &  ca_sx_b, vg_sx_m, ca_wo_b, vg_wo_m
      real, dimension(:,:), allocatable :: vswc
      real :: swcexp

      !!End GWP edits
```

Edits to percmain.f

Add new local variables:

```
      real :: ndt, tidt, dt_sep

      ndt = 24
```

Initiate hourly time loop for percolation, where incoming infiltration is divided into 24 equal hourly intervals (~line 134):

```
      dt_sep = sepday / ndt

      do tidt = 1, ndt
          do j1 = 1, sol_nly(j)
            !! add water moving into soil layer from overlying layer
            if (j1 == 1) then
                sol_st(j1,j) = sol_st(j1,j) + dt_sep
            else
                sol_st(j1,j) = sol_st(j1,j) + sepday
            endif
```

Comment out old soil water code and include new code which runs percmicro whenever excess soil water content is above 0. Then remove seepage and lateral flow from soil water storage and check mass balance (~line 197):

```
    !        if (sw_excess > 1.e-5) then
    !          !! calculate tile flow (lyrtile), lateral flow (latlyr) and
    !          !! percolation (sepday)
    !          call percmicro(j1)
    !
    !         sol_st(j1,j) = sol_st(j1,j) - sepday - latlyr - lyrtile
    !          sol_st(j1,j) = Max(1.e-6,sol_st(j1,j))
    !
    !          !! redistribute soil water if above field capacity (high water table)
    !          call sat_excess(j1)
    !!          sol_st(j1,j) = sol_st(j1,j) - lyrtilex
    !!          sol_st(j1,j) = Max(1.e-6,sol_st(j1,j))
    !        end if

     !SWC edits by GWP (also shut off lines above, double !! means was already off)
                  ! REMOVE FIELD CAPACITY FLAG
              !if (sw_excess > 1.e-5) then
              if (sol_st(j1,j) > 1.e-5) then
                  call percmicro(j1) ! calculate latlyr and sepday
                  !This code no longer needed b/c now updating sol_st and sepage in
percmicro
 !                 sol_st(j1,j) = sol_st(j1,j) - sepday -
 !    &                latlyr - lyrtile
                  !sol_st(j1,j) = Max(1.e-6, sol_st(j1,j))
                  !sol_st(j1,j) = sol_st(j1,j) - latlyr - lyrtile

                  !!GWP EDITS
         !Remove the seepage from the soil water content depending upon saturation level
                  if (sol_st(j1,j) > sol_fc(j1,j)) then
                      if (sepday + latlyr > sw_excess) then
                          ratio = 0.
                          ratio = sepday / (latlyr + sepday)
                          sepday = 0.
                          latlyr = 0.
                          sepday = sw_excess * ratio
                          latlyr = sw_excess * (1. - ratio)
                          sol_st(j1,j) = sol_fc(j1,j)
                      else
                          sepday = sepday
                          latlyr = latlyr
                          sol_st(j1,j) = sol_st(j1,j) - sepday - latlyr
                      end if
                  else
                      if (sepday > sol_st(j1,j)) then
                          sepday = sol_st(j1,j)
                          sol_st(j1,j) = 0.
                      else
                          sol_st(j1,j) = sol_st(j1,j) - sepday
                      end if
                  end if

                  !New mass balance check; already subtracted out sepday and latlyr
                  if (lyrtile < sepday) then
```

```
                sepday = sepday - lyrtile
            else
                sol_st(j1,j) = sol_st(j1,j) - (lyrtile - sepday)
                sepday = 0.
            end if


            !! redistribute soil water if above field capacity (high water table)
            !only isep =/= 0 options now
            call sat_excess(j1)

            !check if below minimum values
            if (sol_st(j1,j) < 0.) then
                       sepday = sepday + sol_st(j1,j) ! sol_st will be negative
                sol_st(j1,j) = 0.
            end if

        end if
        !End GWP edits
```

Update summary calculations to include new routine and include new variables and end time loop (~line270):

```
          !! summary calculations
          if (j1 == sol_nly(j)) then
            sepbtm(j) = sepbtm(j) + sepday
          endif
          latq(j) = latq(j) + latlyr
          qtile = qtile + lyrtile
          flat(j1,j) = flat(j1,j) + latlyr + lyrtile
          sol_prk(j1,j) = sol_prk(j1,j) + sepday
           if (latq(j) < 1.e-6) latq(j) = 0.
          if (qtile < 1.e-6) qtile = 0.
          if (flat(j1,j) < 1.e-6) flat(j1,j) = 0.
          !store these for debugging
          sol_sep(j1,j) = sol_prk(j1,j)
          sol_lat(j1,j) = flat(j1,j)
          infl_print(j1,j) = inflpcp
        end do
      end do
```

Edits to percmicro.f

Modify lateral flow calculation from daily to hourly. Iperc 5-8 are to date untested:

```
        !the factor must be changed from 0.024 to 0.001 because we changing from daily to
hourly calculation
!        latlyr = adjf * ho * sol_k(ly1,j) * hru_slp(j) / slsoil(j)
!     &                                                           * .024
        latlyr = adjf * ho * sol_k(ly1,j) * hru_slp(j) / slsoil(j)
     &                                                           * .001
```

Read iperc flag from basins.bsn and calculate appropriate soil water percolation:

```
!!SWC edits by GWP
      if (iperc == 0) then ! original calculation
          sol_hk(ly1,j) = Max(2., sol_hk(ly1,j))
```

```
!!  compute seepage to the next layer
    sepday = 0.
    sepday = sw_excess * (1. - Exp(-24. / sol_hk(ly1,j)))

else if (iperc == 1) then ! Campbell-Rawls
    if (ly1 == 1) then
              z = sol_z(ly1,j)
    else
              z = sol_z(ly1,j) - sol_z(ly1-1,j)
       endif
    sepday = 0.
    if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
        sepday = sol_k(ly1,j) * 1.
    else
        theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
        se = theta  /
&           sol_por(ly1,j)
        if (se > 1.) then
            se = 1.
        end if
        n = 3. + 2. * ca_rb_b(ly1,j)
        sol_kun = sol_k(ly1,j) * se ** n
        sepday = sol_kun * 1.
   end if
    !sepday = min(sw_excess, sepday)
    !sepday = min(sepday, sol_st(ly1,j))

else if (iperc == 2) then ! VanGenuchten-Rawls
    if (ly1 == 1) z = sol_z(ly1,j)
    if (ly1 > 1) z = sol_z(ly1,j) - sol_z(ly1-1,j)
    sepday = 0.
    if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
        sepday = sol_k(ly1,j) * 1.
    else
        theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
        se = (theta - rb_thr(ly1,j)) /
&           (sol_por(ly1,j) - rb_thr(ly1,j))
        m = vg_rb_m(ly1,j)
        if (m > 0) then
            minv = 1/m
        else
            m = 0
        end if
        factor_1 = 1. - se ** minv
        factor_2 = factor_1 ** m
        factor_3 = 1. - factor_2
        factor_4 = factor_3 ** 2.
        if (se > 0) then
            sol_kun = sol_k(ly1,j) * factor_4 * se ** 0.5
        else
            sol_kun = 0.
        endif
        sepday = sol_kun * 1.
   end if
     !sepday = min(sepday, sol_st(ly1,j))

else if (iperc == 3) then ! Campbell-Cosby
```

```
      if (ly1 == 1) z = sol_z(ly1,j)
      if (ly1 > 1) z = sol_z(ly1,j) - sol_z(ly1-1,j)
      sepday = 0.
      if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
          sepday = sol_k(ly1,j) * 1.
      else
          theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
          se = theta  /
&             sol_por(ly1,j)
          n = 3. + 2. * ca_co_b(ly1,j)
          sol_kun = sol_k(ly1,j) * se ** n
          sepday = sol_kun * 1.
     end if
       !sepday = min(sepday, sol_st(ly1,j))


  else if (iperc == 4) then ! VanGenuchten-Cosby
      if (ly1 == 1) z = sol_z(ly1,j)
      if (ly1 > 1) z = sol_z(ly1,j) - sol_z(ly1-1,j)
      sepday = 0.
      if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
          sepday = sol_k(ly1,j) * 1.
      else
          theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
          se = theta  /
&             sol_por(ly1,j)
          m = vg_co_m(ly1,j)
          if (m > 0) then
              minv = 1/m
          else
              m = 0
          end if
          factor_1 = 1. - se ** minv
          factor_2 = factor_1 ** m
          factor_3 = 1. - factor_2
          factor_4 = factor_3 ** 2.
          sol_kun = sol_k(ly1,j) * factor_4 * se ** 0.5
          sepday = sol_kun * 1.
     end if
       !sepday = min(sepday, sol_st(ly1,j))


  else if (iperc == 5) then ! Campbell-Saxton
      if (ly1 == 1) z = sol_z(ly1,j)
      if (ly1 > 1) z = sol_z(ly1,j) - sol_z(ly1-1,j)
      sepday = 0.
      if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
          sepday = sol_k(ly1,j) * 1.
      else
          theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
          se = theta  /
&             sol_por(ly1,j)
          n = 3. + 2. * ca_sx_b(ly1,j)
          sol_kun = sol_k(ly1,j) * se ** n
          sepday = sol_kun * 1.
     end if
       !sepday = min(sepday, sol_st(ly1,j))
```

```fortran
 else if (iperc == 6) then ! VanGenuchten-Saxton
     if (ly1 == 1) z = sol_z(ly1,j)
     if (ly1 > 1) z = sol_z(ly1,j) - sol_z(ly1-1,j)
     sepday = 0.
     if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
         sepday = sol_k(ly1,j) * 1.
     else
         theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
         se = theta /
&            sol_por(ly1,j)
         m = vg_sx_m(ly1,j)
         if (m > 0) then
             minv = 1/m
         else
             m = 0.
         end if
         factor_1 = 1. - se ** minv
         factor_2 = factor_1 ** m
         factor_3 = 1. - factor_2
         factor_4 = factor_3 ** 2.
         sol_kun = sol_k(ly1,j) * factor_4 * se ** 0.5
         sepday = sol_kun * 1.
    end if
      !sepday = min(sepday, sol_st(ly1,j))


 else if (iperc == 7) then ! Campbell-Wosten
     if (ly1 == 1) z = sol_z(ly1,j)
     if (ly1 > 1) z = sol_z(ly1,j) - sol_z(ly1-1,j)
     sepday = 0.
     if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
         sepday = sol_k(ly1,j) * 1.
     else
         theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
         se = theta  /
&            sol_por(ly1,j)
         n = 3. + 2. * ca_wo_b(ly1,j)
         sol_kun = sol_k(ly1,j) * se ** n
         sepday = sol_kun * 1.
    end if
      !sepday = min(sepday, sol_st(ly1,j))


 else if (iperc == 8) then ! VanGenuchten-Wosten
     if (ly1 == 1) z = sol_z(ly1,j)
     if (ly1 > 1) z = sol_z(ly1,j) - sol_z(ly1-1,j)
     sepday = 0.
     if (sol_st(ly1,j) > 0.99 * sol_ul(ly1,j)) then
         sepday = sol_k(ly1,j) * 1.
     else
         theta = sol_st(ly1,j) / z + sol_wp(ly1,j)
         se = (theta - wo_thr(ly1,j)) /
&            (sol_por(ly1,j) - wo_thr(ly1,j))
         m = vg_wo_m(ly1,j)
         if (m > 0) then
             minv = 1/m
         else
```

```
                m = 0.
            end if
            factor_1 = 1. - se ** minv
            factor_2 = factor_1 ** m
            factor_3 = 1. - factor_2
            factor_4 = factor_3 ** 2.
            if (se > 0) then
                sol_kun = sol_k(ly1,j) * factor_4 * se ** 0.5
            else
                sol_kun = 0.
            endif
            sepday = sol_kun * 1.
       end if
         !sepday = min(sepday, sol_st(ly1,j))

      end if
!!End GWP edits
```

Update maximum biozone layer seepage to hourly:

```
          !sepday = min(sepday,sol_k_sep *24.)
          sepday = min(sepday,sol_k_sep *1.)
```

Comment out mass balance since this now performed in percmain (~line 343):

```
      ! this is now done in percmain
            !! check mass balance moved to percmain
!       if (sepday + latlyr > sw_excess) then
!         ratio = 0.
!         ratio = sepday / (latlyr + sepday)
!         sepday = 0.
!         latlyr = 0.
!         sepday = sw_excess * ratio
!         latlyr = sw_excess * (1. - ratio)
!       endif
!       if (sepday + lyrtile > sw_excess) then
!         sepday = 0.
!         sepday = sw_excess - lyrtile
!       endif
```

Edits to readbsn.f

Read in iperc flag (determines which soil percolation equation to run - ~line 593):

```
      !!      SWC edits by GWP
      if (eof < 0) exit
      read (103,*,iostat=eof) iperc
!!    End GWP edits
```

Read in swcexp value which applies a percentage change to the default exponential value in the Campbell and van Genuchten equations, which is set using PTFs. This value needs to be added to the basins.bsn file on line 17, after FFCB. Specifying as zero means the default value is used (~line 407):

```
!GWP edits
      read (103,*) swcexp
!end edits
```

Edits to readfile.f

Read in new soil water variable printing code, iwriteperc (~line 788):

```
!!      GWP
      read (101,5101) titldum
      read(101,*,iostat=eof) iwriteperc
!!     GWP
```

Open and format new soil water variable output files (~line 677):

```
!!SWC edits by GWP - create depth and hk files (probably want off in most scenarios)
        open (1293,file='output.dg')
        write (1293,5005)
5005    format (t20,'Soil Depth (mm)',/,t15,'Layer #',/,t3,'Day',t13,
     *  'HRU',t28,'1',t40,'2',t52,'3',t64,'4',t76,'5',t87,'6',t100,
     * '7',t112,'8',t124,'9',t135,'10')
        open (1295,file='output.sep')
        write (1295,5007)
5007    format (t20,'Soil Sep',/,t15,'Layer #',/,t3,'Day',t13,
     *  'HRU',t28,'1',t40,'2',t52,'3',t64,'4',t76,'5',t87,'6',t100,
     *  '7',t112,'8',t124,'9',t135,'10')
        open (1296,file='output.lat')
        write (1296,5008)
5008    format (t20,'Soil Lat',/,t15,'Layer #',/,t3,'Day',t13,
     *  'HRU',t28,'1',t40,'2',t52,'3',t64,'4',t76,'5',t87,'6',t100,
     *  '7',t112,'8',t124,'9',t135,'10')
        open (1297,file='output.inflpcp')
        write (1297,5009)
5009    format (t20,'InflPcp',/,t15,'Layer #',/,t3,'Day',t13,
     *  'HRU',t28,'1',t40,'2',t52,'3',t64,'4',t76,'5',t87,'6',t100,
     *  '7',t112,'8',t124,'9',t135,'10')
!!End GWP edits
```

Edits to sat_excess.f

Modify saturation excess procedure to add soil water content to layer above if saturated or to surface runoff if top layer. Default SWAT code is commented out as this moves saturation excess to seepage and then back up layers. This code was created primarily by Claire Baffaut (~line 142):

```
           !! GWP Edits to check max and min soil moisture and resdistribute if saturated
          !check if greater than saturation
          if (sol_st(j1,j) - sol_ul(j1,j) > 1.e-4) then
              ul_excess = sol_st(j1,j) - sol_ul(j1,j)
              sol_st(j1,j) = sol_ul(j1,j)
              !check if is first layer
              if (j1 == 1) then
                  if (pot_fr(j) > 0.) then
                      pot_vol(j) = pot_vol(j) + ul_excess
                  else
                      surfq(j) = surfq(j) + ul_excess
                  end if
                  ul_excess = 0.
              else !not top layer
                  nn = j1
                  do ly = nn-1,1,-1 !redistribute to next layer up
```

```
                      sol_st(ly,j) = sol_st(ly,j) + ul_excess
                      if (sol_st(ly,j) > sol_ul(ly,j)) then
                          ul_excess = sol_st(ly,j) -
     &                        sol_ul(ly,j)
                          sol_st(ly,j) = sol_ul(ly,j)

                              if (j1 == 1 .and. ul_excess >
     &                            1.e-4) then !if now in upper layer
                                  if (pot_fr(j) > 0.) then
                                      pot_vol(j) = pot_vol(j) +
     &                                    ul_excess
                                  else
                                      surfq(j) = surfq(j) +
     &                                    ul_excess
                                  end if
                                  ul_excess = 0.
                              end if
                      else
                          ul_excess = 0.
                          exit
                      end if
                    end do
              end if
          end if
      end if


!!old code
!      if (j1 < sol_nly(j)) then
!        if (sol_st(j1,j) - sol_ul(j1,j) > 1.e-4) then
!          sepday = sepday + (sol_st(j1,j) - sol_ul(j1,j))
!          sol_st(j1,j) = sol_ul(j1,j)
!        end if
!      else
!
!        if (sol_st(j1,j) - sol_ul(j1,j) > 1.e-4) then
!          ul_excess = sol_st(j1,j) - sol_ul(j1,j)
!          sol_st(j1,j) = sol_ul(j1,j)
!          nn = sol_nly(j)
!          do ly = nn - 1, 1, -1
!            sol_st(ly,j) = sol_st(ly,j) + ul_excess
!            if (sol_st(ly,j) > sol_ul(ly,j)) then
!              ul_excess = sol_st(ly,j) - sol_ul(ly,j)
!              sol_st(ly,j) = sol_ul(ly,j)
!            else
!              ul_excess = 0.
!              exit
!            end if
!            if (ly == 1 .and. ul_excess > 0.) then
!              !! add ul_excess to depressional storage and then to surfq
!              pot_vol(j) = pot_vol(j) + ul_excess
!            end if
!          end do
!          !compute tile flow again after saturation redistribution
!!         if (ldrain(j) > 0.) then
!!           ul_excess = sol_st(ldrain(j),j) - sol_fc(ldrain(j),j)
!!           if (ul_excess > 0.) then
!!             lyrtilex = ul_excess * (1. - Exp(-24. / tdrain(j)))
```

```
!!          end if
!!        end if
!       end if
!     end if
!!     end if
```

Edits to soil_phys.f

Calculate new soil water percolation equation parameter values based on soil properties (~line 243):

```
!!    SWC edits by GWP
      if (iperc > 0) then
          do ilyr = 1, sol_nly(i)
              if (sol_sand(ilyr,i) > 65) then
                  wo_thr(ilyr,i) = 0.025
              else
                  wo_thr(ilyr,i) = 0.01
              end if
              rb_thr(ilyr,i) = -0.0182482 +
     &            0.00087269 * sol_sand(ilyr,i)  +
     &            0.00513488 * sol_clay(ilyr,i) +
     &            0.02939286 * sol_por(ilyr,i) -
     &            0.00015395 * sol_clay(ilyr,i) ** 2. -
     &            0.0010827 * sol_sand(ilyr,i) * sol_por(ilyr,i) -
     &            0.00018233 * sol_clay(ilyr,i) ** 2. *
     &            sol_por(ilyr,i) ** 2. +
     &            0.00030703 * sol_clay(ilyr,i) ** 2. * sol_por(ilyr,i)
     &            - 0.0023584 * sol_por(ilyr,i) ** 2. * sol_clay(ilyr,i)
              ca_rb_lam = Exp(-0.7842831 +
     &            0.0177544 * sol_sand(ilyr,i) -
     &            1.062498 * sol_por(ilyr,i) -
     &            0.00005304 * sol_sand(ilyr,i) ** 2. -
     &            0.00273493 * sol_clay(ilyr,i) ** 2. +
     &            1.11134946 * sol_por(ilyr,i) ** 2. -
     &            0.03088295 * sol_sand(ilyr,i) * sol_por(ilyr,i) +
     &            0.00026587 * sol_sand(ilyr,i) ** 2. *
     &            sol_por(ilyr,i) ** 2. -
     &            0.00610522 * sol_clay(ilyr,i) ** 2. *
     &            sol_por(ilyr,i) ** 2. -
     &            0.00000235 * sol_sand(ilyr,i) ** 2. *
     &            sol_clay(ilyr,i) +
     &            0.00798746 * sol_clay(ilyr,i) ** 2. * sol_por(ilyr,i)
     &            - 0.00674491 * sol_por(ilyr,i) ** 2. *
     &            sol_clay(ilyr,i))
              ca_rb_b(ilyr,i) = 1 / ca_rb_lam
              vg_rb_m(ilyr,i) = ca_rb_lam / (ca_rb_lam + 1.)
              ca_co_b(ilyr,i) = 2.91 + 0.159 * sol_clay(ilyr,i)
              vg_co_m(ilyr,i) = 1 / (1 + ca_co_b(ilyr,i))
              ca_sx_b(ilyr,i) = 3.14 + 0.00222 * sol_clay(ilyr,i) ** 2.
     &            + 0.00003484 * sol_sand(ilyr,i) ** 2. *
     &            sol_clay(ilyr,i)
              vg_sx_m(ilyr,i) = 1. / (1. + ca_sx_b(ilyr,i))
              if (ilyr > 2) then
                  wo_factor = 0.
              else
                  wo_factor = 1.
              end if
              ca_wo_n = 1. + Exp(-25.23 - 0.02195 * sol_clay(ilyr,i) +
```

```
     &             0.0074 * sol_silt(ilyr,i) -
     &             0.194 * sol_cbn(ilyr,i) +
     &             45.5 * sol_bd(ilyr,i) -
     &             7.24 * sol_bd(ilyr,i) ** 2. +
     &             0.0003658 * sol_clay(ilyr,i) ** 2. +
     &             0.002885 * sol_cbn(ilyr,i) ** 2. -
     &             12.81 * sol_bd(ilyr,i) ** -1. -
     &             0.1524 * sol_silt(ilyr,i) ** -1. -
     &             0.01958 * sol_cbn(ilyr,i) ** -1. -
     &             0.2876 * Log(sol_silt(ilyr,i)) -
     &             0.0709 * Log(sol_cbn(ilyr,i)) -
     &             44.6 * Log(sol_bd(ilyr,i)) -
     &             0.02264 * sol_bd(ilyr,i) * sol_clay(ilyr,i) +
     &             0.0896 * sol_bd(ilyr,i) * sol_cbn(ilyr,i) +
     &             0.00718 * wo_factor * sol_clay(ilyr,i))
           ca_wo_b(ilyr,i) = 1. / (ca_wo_n - 1)
           vg_wo_m(ilyr,i) = 1. - 1. / co_wo_n
         end do
       end if
!!     End GWP edits
```

Use multiplicative factor (swcexp) set in basins.bsn file to modify default value by percentage (~line 281)

```
!Use swcexp multiplicative factor in calibration
              ca_rb_b(ilyr,i) = ca_rb_b(ilyr,i) * (1. + swcexp)
              vg_rb_m(ilyr,i) = vg_rb_m(ilyr,i) * (1. + swcexp)
              ca_co_b(ilyr,i) = ca_co_b(ilyr,i) * (1. + swcexp)
              vg_co_m(ilyr,i) = vg_co_m(ilyr,i) * (1. + swcexp)

!End change w/ swcexp
```

Edits to subbasin.f

Since saturation excess can now add to runoff, the order of code must be modified. Several routines are called within surface rather than in subbasin to allow this possible addition to runoff. Routines called in surface are commented out of subbasin.f. Code was primarily created by Claire Baffaut (~line 207):

```
          !! add surface flow that was routed across the landscape on the previous day
       !!   qday = qday + surfq_ru(j)
       !!   surfq_ru(j) = 0.

          !! compute effective rainfall (amount that percs into soil)
!          inflpcp = Max(0.,precipday - surfq(j))
!!        end if

!        !! perform management operations
!        if (yr_skip(j) == 0) call operatn

!        if (auto_wstr(j) > 1.e-6 .and. irrsc(j) > 2) call autoirr

!        !! perform soil water routing
!        call percmain
```

Edits to sumv.f

Calculate daily layer volumetric soil water content for printing in output.hru (~line 453):

```
!Edits by GWP get mm/mm SWC by layer to print in output.hru
z = 0.
ly = 0.

do ly = 1, sol_nly(j)
    if (ly == 1) then
        z = sol_z(ly,j)
    else
        z = sol_z(ly,j) - sol_z(ly-1,j)
    end if
    vswc(ly,j) = sol_st(ly,j) / z + sol_wp(ly,j)
end do
!End of GWP edits
```

Add calculated volumetric soil water content to monthly sums (~line 541):

```
# GWP edits

        !hval = 73

        !do ly = 1, sol_nly(j)
        !    hrumono(hval,j) = hrumono(hval,j) + vswc(ly,j)
        !    hval = hval + 1
        !end do

        hrumono(73,j) = hrumono(73,j) + vswc(1,j)
        hrumono(74,j) = hrumono(74,j) + vswc(2,j)
        hrumono(75,j) = hrumono(75,j) + vswc(3,j)
        hrumono(76,j) = hrumono(76,j) + vswc(4,j)
        hrumono(77,j) = hrumono(77,j) + vswc(5,j)
        hrumono(78,j) = hrumono(78,j) + vswc(6,j)
        hrumono(79,j) = hrumono(79,j) + vswc(7,j)
        hrumono(80,j) = hrumono(80,j) + vswc(8,j)
        hrumono(81,j) = hrumono(81,j) + vswc(9,j)
        hrumono(82,j) = hrumono(82,j) + vswc(10,j)
# end edits
```

Edits to surface.f

Moved code from subbasin.f to allow for the possibility of saturation excess addition to runoff (~line 120):

```
!!GWP - Moved code from subbasin to here to allow for returned sat flow to runoff
inflpcp = Max(0., precipday - surfq(j))

!! perform management operations
if (yr_skip(j) == 0) call operatn

if (auto_wstr(j) > 1.e-6 .and. irrsc(j) > 2) call autoirr

call percmain

!again for inflpcp just in case it was adjusted by surfq in percmain
!*not sure how this changes anything yet
inflpcp = Max(0., precipday - surfq(j))

!!End of code move GWP
```

Edits to writed.f

Write new soil water variables to output files as well as add wilting point soil moisture to the available water content that is printed in output.swr (~line 110):

```
!! SWC edits by GWP - add WP back into printed out SW storage
          write (129,5000) iida, j, (sol_st(j1,j) + sol_wpmm(j1,j),
     &                 j1 = 1, sol_nly(j))
          write (1293,5000) iida, j, (sol_dg(j1,j),
     &                 j1 = 1, sol_nly(j))
          write (1295,5000) iida, j, (sol_sep(j1,j),
     &                 j1 = 1, sol_nly(j))
          write (1296,5000) iida, j, (sol_lat(j1,j),
     &                 j1 = 1, sol_nly(j))
          write (1297,5000) iida, j, (infl_print(j1,j),
     &                 j1 = 1, sol_nly(j))
   !!End GWP edits
```

Edits to zero2.f

Initialize new soil water variables with zero (~line 383):

```
      !! SWC edits by GWP
      iimp = 0
      iperc = 0
      iwriteperc = 0
      sol_sep = 0
      sol_exw = 0
      sol_ule = 0
      sol_dg = 0.
      sol_sep = 0.
      sol_lat = 0.
      infl_print = 0.
      !PTF parameters
      wo_thr = 0.
      rb_thr = 0.
      ca_rb_b = 0.
      vg_rb_m = 0.
      ca_co_b = 0.
      vg_co_m = 0.
      ca_sx_b = 0.
      vg_sx_m = 0.
      ca_wo_b = 0.
      vg_wo_m = 0.

      swcexp = 0.
      vswc = 0.

      !!End GWP edits
```

New flags needed to run the model

There are two new flags that can be used to specify which soil water equation to use and to turn on printing of extra variables.

(1) In the **basins.bsn** file an two extra lines are needed, one on line 17 and one at the bottom line of the file to specify percentage change to the exponential value of the Campbell and van Genuchten equations and the soil water percolation equation used by SWAT as follows:

Line 17:

```
0.000    | SWCEXP : Multiplicative factor used for calibration of b/m parameter in CA and VG equations
```

Last line:

```
1        | IPERC: Changes the percolation method (0=default, 1=CA-RA, 2=VG-RA, 3=CA-CO, 4=VG-CO)
```

IPERC: A flag value of 0 will use the default SWAT soil percolation equation. This is not recommended, rather the user should simply run an officially released SWAT executable if the default is preferred. Flag values of 1, 2, 3, and 4 indicate the Campbell-Rawls, van Genuchten-Rawls, Campbell-Cosby, and van Genuchten-Cosby equations respectively. Although, other options have been coded (5-8), users are cautioned against their use as they have not undergone testing to date.

SWCEXP: A flag value of zero will not change the exponent value and will simply use the default value calculated by the PTFs as specified from the IPERC flag. Non-zero values will apply a percentage change to the exponent, either positive or negative as specified by the user.

(2) The second flag is in **file.cio** and turns on and off printing of additional soil water variables and is similarly added as a line at the bottom of the file as follows:

```
1        | IWRITEPERC: Code for printing new output soil files
```

Where 0 will not print the variables and 1 will print files for layer-based lateral flow, percolation, depth, and infiltration precipitation.

Changes to output data

For simpler comparison of output soil water values, the values in the **output.swr** file have been modified to print total soil water content, rather than available water content as specified by the default SWAT output. That is, the **output.swr** file now records available soil water plus wilting point water, not soil water content without wilting point water content.

Additionally, changes made to the code provide the user the ability to print volumetric soil water content for specified HRUs and associated layers in **file.cio**. HRU output variables 80-89 will print volumetric soil water content for soil layers 1-10 respectively for user-specified HRUs. Volumetric soil output is useful when comparing to observed soil moisture information which is generally recoded volumetrically as opposed to the equivalent water depth used in SWAT.