Table of Contents

Erosion

Simulates daily soil erosion, and (optionally) the effect of soil loss on the soil profile. Code and this document are derived from PERFECT.

Methods:

The calculation of daily soil loss is performed by one of **two** submodels:

Sub Model 1: Freebairn

A modified MUSLE (Freebairn and Wockner) cover concentration function determined from QDPI field data to predict soil movement from the inter-contour bank area for clay soils in situations where peak discharge cannot be adequately predicted.

It accounts for variation in soil loss with cover and runoff volume (the main factors that can be managed), and uses the MUSLE slope-length, erodibility and practise factors to provide generality.

The model has the following form:

{Equation 1}

$$E = (16.52 - 0.46COV + 0.0031COV^2)\frac{LS \cdot K \cdot P \cdot Q}{10} \quad COV < 50\%$$

$$E = (-0.0254COV + 2.54)\frac{LS \cdot K \cdot P \cdot Q}{10} \quad COV > 50\%$$

Where;

- E: Event soil loss (t/ha)
- COV: Cover (%)
- Q: Event runoff (mm)
- K: Soil erodibility factor (t/ha/EI30)
- LS: Slope length and steepness factor
- P: Supporting practice factor

The LS factor from Wishmeier and Smith (1978) is related to catchment slope and length by:

{Equation 2}

$$LS = (65.41S^2 + 4.56S + 0.065)\left(\frac{L}{22.1}\right)^{0.6(1-e^{-35.835S})}$$

Where;

- LS: Slope length and steepness factor
- S: Sine of slope angle
- L: Length of catchment (m)

Dimensional Correctness of Modified MUSLE

Note that the equation for E is not dimensionally correct (the units of E (t/ha) do not match those on the right hand side of the equation). To be dimensionally correct E should be in units of (mm t)/(ha EI30) instead of t/ha. The existance of the EI30 unit can be traced back to the original USLE(Wischmeier and Smith, 1978) equation on which this equation is based. In the USLE EI30 unit is cancelled out because both R (Soil Erosivity) and K (Soil Erodibility) have EI30 as units. The unit remains in the new equation because E (Erosion) is calculated using Q (Runoff) not R (Soil Erosivity). The new equation still uses K however, which means the EI30 unit from the K is no longer cancelled out. For consistency, K is kept in the same units in the new equation. so that users who have measured the values of K for use in USLE can use the same measured values (in the same units) in the new equation. However, keeping these same units for consistency results in the new equation being dimensionally incorrect. The other extra dimension, the mm unit, is also gained as a consequence of using Q (Runoff) instead of R.

To resolve these dimensional issues we firstly define K as a being unit less however with the same value as for K used in USLE.

From first principles the equation above starts out as,

```
In USLE equation it is R x K which gives us Erosion in t/ha.
Therefore we need to replace R such that (R_runoff x Q) x K still equals t/ha
of erosion.
K is now unitless however,
Therefore R_runoff must be in t/(ha x mm) which means R_runoff x Q(mm) x K =
t/ha
R_runoff is therefore the tons of soil lost per ha per millimeter of runoff.

E = R_runoff x Q x K x LS x P
OR
E = (sediment erosion in t/(ha x mm)) x Q x K x LS x P



From R_runoff, we can remove a unit conversion for (grams of water as runoff)
to (mm of water as runoff)
and reduce R_runoff to a fraction of (grams of soil lost(sediment)) per (gram
of water as runoff).

sediment concentration = 1 gram of sediment / 1 gram of water

1 gram of sediment = (1/10^6) tons of sediment
```

```
1000Kg of Water = 1m^3
10^6 grams of water = 1000mm x 1m^2
1gram of water = (1000 x (mm x m^2))/10^6

1 ha = 100m x 100m = 10^4m^2
1m^2 = 1/(10^4)

1 gram of water = (1000 x (mm x ha)) / (10^6 x 10^4)
1 gram of water = (1/10^7) x (mm x ha)

frac_to_(t/ha mm) =  1 gram of sediment/ 1 gram of water = (tonnes/(10^6)) x
(10^7 / (mm x ha) = 10 t/(mm x ha)

We are now left with

E = sediment concentration function x (10 t/(ha mm)) x Q(mm) x K x LS x P



The remaining sediment concentration function however is going to be a
function of the cover fraction.
Rather than using cover fraction in the function it would be nicer to use the
cover percentage and
just move this conversion out of the function so that the function now has
cover as a percentage.

E =  [sediment concentration function x percent_to_frac x frac_to_(t/(ha mm))]
x Q(mm) x K x LS x P

E  = [sediment concentration function x (1/100) x (10 t/(ha mm))] x Q(mm) x K
x LS x P

E  = [sediment concentration function x (1/10) t/ha/mm] x Q(mm) x K x LS x P
```

This matches the simpler form at the start of this section {Equation 1}
where,

```
 sediment concentration function = 16.52 - 0.46 * cover + 0.0031 cover^2
```

and (1/10) actually has its units removed,

```
(1/10) t/(ha mm) = (1/10)
```

We now see Equation 1 in its full, dimensionally correct form:

```
E (t/ha) = 16.52 - 0.46 * cover + 0.0031 cover^2 x (1/10)(t/(ha mm)) x LS x K
x P x Q(mm)

(when cover is less than 50%)

AND

E (t/ha) = -0.0254 * cover + 2.54 x (1/10)(t/(ha mm)) x LS x K x P x Q(mm)

(when cover is greater than 50%)
```

QDPI field data was used to obtain the form and the coefficients for sediment concentration function. This is possible because E, Q, K, LS, P are all known via measurement.

As mentioned above, We do not publish the full dimensionally correct form.
For consistency sake we replace the unitless K with the same dimensions as the USLE definition of K.
We then ignore the t/(ha mm) units on the (1/10) as well.
This results in a dimensionally incorrect form of the equation but one that is easier to understand for users of the USLE.

Right Units for K

In the section above about dimensional correctness we stated,
"To resolve these dimensional issues we firstly define K as a being unitless however with the same value as for K used in USLE."

Even though when using the modified MUSLE (Freebairn and Wockner) we don't care about the **units** of K
(because to keep modified MUSLE (Freebairn and Wockner) dimensionally correct we drop off the units)
the **value** of K still needs to be the **value in the correct units**.

As a result of the history of the USLE there are 3 possible units for K.
Originally the USLE equation was in Imperial units. Or rather the units of R and K were in Imperial units. (because all the other variables in USLE are unitless)
The Equation was later converted to Metric units. Or rather the units for R and K were converted to Metric units.
Finally the Equation was converted to SI units (which are metric but are in the Official SI metric units). So once again R and K were converted into SI units.

At this point we need to state that the **correct units of K** for the modified MUSLE (Freebairn and Wockner) is K in **Metric** units.

So if you have K in either Imperial units or SI units,**YOU MUST CONVERT IT TO METRIC BEFORE USING IT**.
Note: There is no R in the modified MUSLE (Freebairn and Wockner) so we don't need to worry about converting the units of R, like we would if we were using the USLE.

To convert K(SI) to K(Metric) simply multiply the K(SI) by Gravity (9.81 m/s^2).

The sensible range of values for K(SI)
is **between 0.001 and 0.09**

The sensible range of values for K(Metric)
is **between 0.01 and 0.88**

You can derive the above conversion from SI to Metric as follows,

According to the USLE,
the units of K (the Erodability of the Soil) are in t/(ha EI30)
where EI30 is the unit of R (the Erosivity of the Soil).
The idea of this being that the units of R which are EI30 will be cancelled out by the EI30 in K and just leave the t/ha for the result of the equation (because all the other variables in USLE are unitless).

Therefore if we wish to convert K(SI) into K(Metric)
we first must work out the conversion of units of EI30(SI) to EI30(Metric) (or to state another way, we must first work out the conversion of units of R(SI) to R(Metric))
because EI30(SI) are part of the units of K(SI).

Since,
R(SI) = EI30(SI) = (MJ/ha) x (mm/h)
R(Metric) = EI30(Metric) = 100m x (t/ha) x (cm/h)

EI30 is a measure of Kinetic Energy of falling rain and Rainfall Intensity.
Joules is the unit of the Kinetic Energy of the falling rain.
We divide this Energy over the area it falls which is why we get MJ/ha.
The mm/h is the Maximum Rainfall intensity falling over 30min period.

Joules = Work = Force x Distance = (Mass x Acceleration) x Distance = Mass x Gravity x Distance = Kg x 9.81 (m/s^2) x m

Starting with R(SI) we want to get to R(Metric)
R(SI) = EI30(SI) = (MJ/ha) x (mm/h)

Therefore,

using
Joules = Kg x 9.81 (m/s^2) x m

R(SI) = 10^6 x Kg x 9.81 (m/s^2) x m x (1/ha) x (mm/h)

R(SI)/(9.81 m/s^2) = 10^6 x Kg x m x (1/ha) x (mm/h)

= 10^3 x t x m x (1/ha) x (mm/h)
= 10^2 x m x (t/ha) x (10 x mm/h)
= 100m x (t/ha) x (cm/h)
= R(Metric)

hence,
R(SI)/ (9.81 m/s^2) = R(Metric)
or
R(SI)/Gravity = R(Metric)

or
EI30(SI)/Gravity = EI30(Metric)

Now that we have worked out the conversion of R(SI) to R(Metric) we can now work out the conversion of K(SI) to K(Metric).

```
K(SI) = t/(ha x EI30(SI))
K(Metric) = t/(ha x EI30(Metric))

using, EI30(Metric) = EI30(SI)/Gravity

K(Metric) = t/(ha (EI30(SI)/Gravity) )

K(Metric) = (t x Gravity)/(ha x EI30(SI))

K(Metric) = Gravity x t/(ha x EI30(SI))

K(Metric) = Gravity x K(SI)

K(Metric) = 9.81 (m/s^2) x K(SI)
```

Sub Model 2: Rose

A simplified version of the sediment concentration function contained in GUESS (Carroll).

The equation is given as:

$$E = 2700 \cdot S \cdot (1.0 - cover)\lambda \frac{Q}{100}$$

Where;

- E: Event soil loss (t/ha)
- S: Sine of slope angle
- cover: Fractional surface cover (0-1)
- Q: Event runoff (mm)
- λ Factor approximating efficiency of entrainment

The cover term *cover* in the rose equation does not fully account for the effect of cover. Therefore, the efficiency of entrainment λ is further modified by cover by Rose (1985):

$$\lambda = \lambda_{bare} e^{-0.15COV}$$

Where;

- λ Factor approximating efficiency of entrainment
- λ_{bare} Efficiency of entrainment (bare surface)
- COV Surface cover (%)

The submodels parameters are λ_{bare} (rose_lambda), and the '-0.15' exponential term (rose_b2).

Feedback from erosion on the soil profile
The module is capable of eroding the soil profile as soil loss occurs (see profile_reduction).
The module remains ignorant of what other modules may be doing through the profile, all it does is send a delta to whichever module owns dlayer - typically soilwat2.
It is the responsibility of other modules to ensure they remain consistent with this new profile.
The calculation of "dlt_depth_mm" describing the change in each profile layer for the given amount of soil loss is given by:
dlt_depth_mm(i) = (100.0*soil_loss) / (1000.0*bd(i)) Where;

- soil_loss Event soil loss (t/ha)
- bd Bulk density of the respective profile layer (g/cc)

The bulk density for each layer is used to maintain mass balance of soil movement up through the (constant) soil layer thicknesses.

Erosion continues replacing the lowest profile layer with “imaginary” material of the same properties until a limit (see bed_depth) is reached.

Once the lowest profile layer rests against bedrock, further soil erosion reduces the thickness of this layer until it is a fraction (see profile_layer_merge) of its original thickness.

Once this fraction is reached, the layer is merged into the next highest layer.

(Currently, many apsim modules cannot sensibly deal with the last scenario.)

This process continues until the parameter minimum_depth is reached, at which the simulation stops with a fatal error.

Splitting soil loss into suspended and bed loads.

There is provision to split soil loss into two components: suspended and bed loads. This reflects how soil loss is measured.
Each soil loss submodel is performed twice: once for suspended ediment, and once for heavy particles.

For the modified MUSLE model, the parameter K is replaced by K_bed and K_susp
(NB. It's important to remove or comment out the original K parameter, otherwise the module assumes you are still trying to use a single soil loss equation)

For the rose model, parameters lambda and b2 are replaced by lambda_bed, lambda_susp, b2_bed and b2_susp.
Once again, you must remove the original definitions of lambda and b2. Reporting the separate losses is through soil_loss_bed, soil_loss_susp, both in units of t/ha, and sed_conc_bed, sed_conc_susp (g/l).
soil_loss returns total soil loss, and sed_conc returns total sediment concentration.

Communicating with water balance modules

As the erosion module was derived substantially from PERFECT (ie. a daily timestep), it does not use the advanced features of APSIM's apswim and surface modules (eg. hydrographs or peak runoff rate).
SWIM users should be aware that peak runoff rate is not used in calculation of soil erosion at this time.
Surface users should be aware that surface roughness factors do not enter erosion or runoff.
The calculation of cover used in the soil loss equation should be the same cover as used in runoff prediction.
When soilwat provides water balance, this cover is exported to the system (and thus erosion) as "total_cover". However, SWIM does not provide this variable.
If the surface module is active, it will provide a variable "surf_cover", which is the combined crop and residue covers on the soil surface.
To use this cover as the cover term in the soil loss equation, add the following to the manager's rules:

```
[xxx.manager.init]

total_cover = 0.0

[xxx.manager.start_of_day]

total_cover = surface.surf_cover
```

To use erosion with apswim, add the following rules:

```
[xxx.manager.init]

total_cover = 0.0

[xxx.manager.start_of_day]

total_cover = 1.0 - (1.0 - crop1.cover_tot) * (1.0 - crop2.cover_tot) * … *
(1.0 - residue2.residue_cover)
```

Procedures:

The module follows these steps at initialisation:

1. All variables are set to zero.
2. The parameter file is read, if necessary, ls factor is calculated for the freebairn model.
3. The static parameters are written to the summary file.

During daily processing, the following procedures occur:

1. The daily state variables are reset.
2. Today's inputs are requested from the system.
3. Soil loss is calculated as a function of cover and runoff.
4. If there is soil loss, the profile is changed and sent back to its owner.
5. The new depth to bedrock is calculated.
6. Control returns to the system.

System interface names

Name	Description	Units
model		Either **'rose'** or **'freebairn'**
profile_reduction		Either 'on' or 'off'
profile_layer_merge	Fraction of original size below which the lowest layer is merged into the layer above	(0-1)
minimum_depth	If the profile erodes to this depth, the simulation is stopped	(mm)
slope	Slope of plot	(%)
slope_length	length of plot	(m)
bed_depth	depth to bedrock	(mm)
cover_extra (optional)	fudge factor added to total_cover	(-1.0,1.0)

Freebairn specific parameters (model = 'freebairn')

Name	Description	Units
p_factor	Supporting practise factor	(unitless)

Either

Name	Description	Units
k_factor	Soil erodibility factor	(t/ha/EI 30)

or

Name	Description	Units
k_factor_bed	Soil erodibility factor for bed load	(t/ha/EI 30)
k_factor_susp	Soil erodibility factor for suspended load	(t/ha/EI 30)

Rose specific parameters (model = 'rose')

Either

Name	Description	Units
entrain_eff	Efficiency of entrainment - bare surface	()

or

Name	Description	Units
entrain_eff _bed	Efficiency of entrainment - bare surface - bed load	()
entrain_eff _susp	Efficiency of entrainment - bare surface - suspended load	()

Either

Name	Description
eros_rose_b2	??

or

Name	Description
eros_rose_b2_bed	??
eros_rose_b2_susp	??

Inputs from other modules on a daily timestep:

Name	Description	Units
day		
year		
runoff	daily runoff	(mm)
total_cover	combined crop and residue cover	(0 - 1)
bd(mxlayr)	moist bulk density of soil	(g/cm^3)
dlayer(mxlayr)	thickness of soil layer i	(mm)

Outputs to other modules as requested:

Name	Description	Units
soil_loss	todays soil loss	(t/ha)
soil_loss_bed	todays soil loss in bedload	(t/ha)
soil_loss_susp	todays soil loss in suspension	(t/ha)
soil_loss_mm	todays soil loss from the topmost profile layer	(mm)
sed_conc	todays sediment concentration	(g/l)
sed_conc_bed	todays sediment concentration in bedload	(g/l)
sed_conc_susp	todays sediment concentration in suspended load	(g/l)
bed_depth	todays depth to bedrock	(mm)
erosion_cover	todays cover used to drive soil loss equations	(0 - 1)

Resets to other modules:

Name	Description	Units
dlt_dlayr()	thickness of soil layer i	(mm)

dlt_dlayr is delta layer depth or change in layer depth.

It allows you to alter the layer depths from their current values by a certain amounts specified in this array variable.

ie. if
```
dlayer = 10 20 30 40 50
```
and
```
dlt_dlayr = 6 -7 -8 9 10
```
then dlayer would become,
```
dlayer = 16 13 22 49 60
```

Typical, sample parameters

Dataset	slope %	Slope length m	P factor (t/ha/EI30)	soil erodibility K	Lambda_bare	b2
Greenmount	6.5	60	1.0	0.38 (1)*	0.77 (2)*	15 (3)*
Greenwood	4.5	37	1.0	0.38 (1)*	0.70 (3)*	15 (3)*

* See the corresponding examples below.

The following are examples of Apsim Erosion parameters in the Published Scientific Literature:
(1) Freebairn, Silburn & Loch (1989). Aust.J.Soil Res. 27: 199-211
```
[gmtf.erosion.parameters]

    model = freebairn (1)

    slope = 6.5 (%)

    slope_length = 60.0 (m)

    bed_depth = 1500. (mm)

    profile_reduction = off

    profile_layer_merge = 0.05 ()

    minimum_depth = 100.0 (mm)

    k_factor = 0.38 ()

    p_factor = 1.0 ()
```

(2) Silburn & Loch (1992) 5th Aust. Soil con. Conf.

```
[gmtr.erosion.parameters]

    model = rose (2)

    slope = 6.5 (%)

    slope_length = 60.0 (m)

    bed_depth = 1500. (mm)

    profile_reduction = off

    profile_layer_merge = 0.05 ()

    minimum_depth = 100.0(mm)

    entrain_eff = 0.77 ()

    eros_rose_b2 = 0.15 ()
```

(3) Rose (1985) Adv. in Soil Science Vol 2

```
[gwd.erosion.parameters]

    model = rose (3)

    slope = 4.5 (%)

    slope_length = 37.0 (m)

    bed_depth = 1500. (mm)

    profile_reduction = off

    profile_layer_merge = 0.05 ()

    minimum_depth = 100.0(mm)

    entrain_eff = 0.7 ()

    eros_rose_b2 = 0.15 ()
```

Bibliography

- Carroll, C., Rose, C.W. and Crawford, S.J. (1986). GUESS - Theory Manual. School of Australian Environmental Studies, Griffith University , Queensland ,
- Freebairn, D.F. and Wockner, G.H. (1986). A study of soil erosion on Vertisols of the Eastern Darling Downs, Australian Journal of Soil Research , 19 , 133-46
- Littleboy, M., Silburn, D.M., Freebairn, D.M., Woodruff, D.R., and Hammer, G.L. (1989) PERFECT - A computer simulation model of Productivity Erosion Runoff Functions to Evaluate Conservation Techniques. QDPI, Brisbane.
- Rose, C.W. (1985). Developments in soil erosion and deposition models. Advances in Soil Science , Volume 2 , 1-63.
- Wischmeier, W.H. and Smith, D.D. (1978). Predicting rainfall erosion losses, a guide to conservation planning. Agricultural Handbook number 537 , USDA, 58pp.

Programmers notes

The calculation of soil loss ends up in two variables, soil_loss_bed and soil_loss_susp.

When the user describes a single soil loss equation, the result is left in soil_loss_bed, with soil_loss_susp left at 0.

All reporting is done as the sum of these two variables.

Nutrient reduction:

Nutrient reduction is represented by "moving" the soil layers downwards through the soil profile (the same as moving the nutrient pools of the soil layers upwards through the soil).

Layers take on the nutrient characteristics of the layer below in proportion to the depth increment gained from that layer.

If there is no "bedrock", all layers would eventually all end up with the nutrient properties of the subsoil (bottom layer).

If the accumulated depth loss due to erosion exceeds the depth between the bottom of the profile and the depth to bedrock (and profile reduction is on),
the bottom layer does not contain nutrients from below and its nutrient content will diminish (as nutrients are exported upwards through the soil)

The Algorithm for "Top Down" Removal

1. Determine the loss for each layer.

1.1 For the top layer, this is given by:

```
  conc_kg_kg = variable(1)/(1000.0*bd(1)*dlayr(1)*10.0)
  loss_kg_ha = soil_loss * 1000.0* enr * conc_kg_kg
```

where: conc_kg_kg = kg of nutrient per kg of soil (kg/kg)
variable = content of particular nutrient (kg/ha),
bd = bulk density in (g/cm^3)
dlayr = depth of layer in (mm)
soil_loss = soil loss (t/ha)
enr = enrichment ratio.
"enr" is calculated from

```
  enr = enr_a_coeff * (1000.0 * soil_loss)**(-1.0 * enr_b_coeff)
```

and is bounded to (1.0 < enr < enr_a_coeff):

```
     enr = amin1(enr_a_coeff, enr)
     enr = amax1(enr, 1.0)
```

1.2 For the remaining layers, the loss from each layer is that which moved into the layer above

```
  layer_loss = variable(i) * layer_divide(dlt_depth_mm(i), dlayr(i))
```

where
i is the index of the layer we are working on, and
layer_divide(a,b) bounds the result of (a/b) to 0.0 -> 1.0.
This prevents taking away more than is present in the layer. "dlt_depth_mm(i)" is an array of deltas describing the change in each profile depth for the given amount of soil loss, given by:

```
  dlt_depth_mm(i) = (100.0*soil_loss) / (1000.0*bd(i))
```

The bulk density for each layer is used to maintain mass balance of soil movement up through the (constant) soil layer thicknesses.

2. The gain to each layer is found in a similar manner, ie, the fraction of the layer below that moves into this layer:

2.2

```
  layer_gain = variable(i+1) *  layer_divide(dlt_depth_mm(i+1), dlayr(i+1))
```

2.3 Obviously, the lowest layer cannot use this formula, as there is no layer beneath, and we have to cater for bedrock.

The procedure is to assume that the lowest layer gains a fraction of itself, so long as it is not resting on bedrock:

```
  layer_gain = variable(i) * layer_divide(dlt_depth_mm(i), dlayr(i))

  ! check we're not going into bedrock
  if (suml(dlayr, mxlayr) + dlt_depth_mm(n_layrs) .gt. bed_depth .and.
profile_reduction .eq. on ) then
        layer_gain = 0.0
```

3. At this point, we find the new value of the variable for this layer in the new profile:

3.1

```
  variable(i) = variable(i) + layer_gain - layer_loss
```

Two more steps follow,:-

3.2 If profile reduction is on, check to see if the bottom layer is too thin, and merge it into the next layer up if so.

3.3 Perform a mass balance check:

```
\Sum{yesterdays variable} + loss from layer 1 - gain to lowest layer  =
\Sum{Todays variable}
```

3.4 This procedure is repeated for all the nitrogen / carbon variables:

snh4(mxlayr)
inert_c(mxlayr)
biom_c(mxlayr)
biom_n(mxlayr)
hum_c(mxlayr)
hum_n(mxlayr)
fom_n(mxlayr)
fpool1(mxlayr)
fpool2(mxlayr)
fpool3(mxlayr)

The total amounts of N and C lost on eroded soil is calculated by summing losses from the relevant N and C pools:

```
  n_loss_in_sed = snh4_loss + biom_n_loss + hum_n_loss + fom_n_loss

  c_loss_in_sed = biom_c_loss + hum_c_loss + fpool1_loss + fpool2_loss +
fpool3_loss
```

4. Finally, reducing the profile layer depth (dlayr) takes place following a similar method to before; by reducing from the bottom until the profile rests against bedrock:

```
tot_depth = suml(dlayr, n_layrs) + dlt_depth_mm(n_layrs)

if (tot_depth .gt. bed_depth ) then
  overrun = tot_depth - bed_depth
  do 2000 i = n_layrs, 1, -1
  if (overrun .gt. 0.0) then
    if(overrun .le. dlayr(i)) then
      dlayr(i) = dlayr(i) - overrun
      overrun = 0.0
    else
      overrun = overrun - dlayr(i)
      dlayr(i) = 0.0
    endif
  endif
  2000 continue
endif
```

and the lowest profile layer is merged if necessary.

The threshold for merging layers is specified in the parameter file as a proportion of the original layer,
so after a merge, this threshold must be recalculated as a proportion of the next upper layer.

APSIM
AGRICULTURAL PRODUCTION SYSTEMS SIMULATOR

Fertiliser

The APSIM fertiliser module allows the user to specify the application of solid fertilizer to an APSIM "system" using a schedule spanning multiple years.
There is also total flexibility for user specification of fertiliser components.

Requesting fertiliser application from other modules

Any module can request fertiliser application using standard APSIM messages.
The only data required to specify the operation is amount of fertiliser to apply, the type of fertiliser, and the depth at which it is applied.
The format of a message to apply 50 kg/ha of urea at a depth of 50 mm would be as follows.

```
fertiliser apply amount = 50(kg/ha), depth = 50 (mm), type = urea()
```

In this case "fertiliser" is the name of the module and "apply" is the action that it responds to.".

An alternate approach is to use a keyword naming convention to "set" a fertiliser application property in the fertiliser module. The fertiliser module will interpret any set command variable name starting with "fert_" as a fertiliser command and will use this information to apply fertiliser to the surface layer.

```
if das = 1 then
   soilno3 = no3(1) + no3(2)
   if (soilno3 < 50) then
       fertiliser.fert_no3_n = 50 - soilno3
   endif
endif
```

In this example manager logic the fertiliser module will apply adequate nitrate nitrogen to bring the total nitrogen content of the top 2 surface layers to 50 kg/ha on the day of sowing.

Specifying Fertiliser Constituents

The user has the total flexibility to configure (or describe) within the module's constants (.ini) file what a certain fertiliser contains. To do this the user specifies the fractional (on weight basis) component of the fertiliser for any substance known by the system. A full name can also be specified for use in model outputs.

For example:-

```
[standard.fertiliz .NO3_N]
full_name = Nitrogen as Nitrate
components = no3
fraction = 1.0
```

This tells us that the fertiliser 'NO3_N' is made up of one substance called no3.

More complex examples may be:-

To specify Urea in terms of N per unit area:-

```
[standard.fertiliz .Urea_N]
full_name = Nitrogen as Urea
components = urea
fraction = 1.0
```

To specify Urea in terms of weight of fertiliser:-

```
[standard.fertiliz .Urea]
full_name = Urea
components = urea
fraction = 0.46
```

(the component called urea is described in APSIM in terms of weight of nitrogen and so the fraction must represent the fractional weight of nitrogen in urea)

To specify the relevant components of a compost:-

```
[standard.fertiliz .compost]
fullname = Organic Matter (Compost)
components = no3 nh4 org_n org_c_pool3
fraction = .05 .05 .10 .50
```

Where org_c_pool3 is the carbon pool least susceptible to decomposition and org_n is the nitrogen content of fresh organic matter.
See the SoilN2 documentation for further details.

Once you have specified the composition of a fertiliser the module will find any module that owns each of the components and will send an increment to them respectively.
There is a current limit of twenty components per specification.

Fertiliser Module Outputs

The Fertiliser module has one output, fertiliser, which is the total amount of fertiliser added on any day (kg/ha).

APSIM
AGRICULTURAL PRODUCTION SYSTEMS SIMULATOR

Irrigation

The APSIM irrigate module allows the user to:

- specify irrigation schedules spanning multiple years
- configure an automatic irrigation schedule calculated on soil moisture
- specify both schedules to be turned on or off at any time in a simulation
- apply solutes in irrigation water for redistribution via the water balance module.

Apply Command

The 'apply' command allows you to apply an irrigation in a management component in your simulation (Manager module).

The simplest way to apply an irrigation is to use the 'apply' command of the irrigation module.

nb. This is very similar to the 'apply' command of the fertiliser module.

```
irrigation apply amount = 50 (mm)
```

There are some optional arguments that you can use with the apply command.

Irrigation to runoff like rain does

By default all the irrigation enters the soil.
It is left up to the user to take the efficiency of irrigation (water lost due to method of irrigation used) into account by decreasing the amount applied when irrigating using an apply command.

However there is an option to allow an irrigation to runoff in exactly the same way that rain does. This runoff amount is added to the existing output variable for rain called 'runoff'.

You can specify this runoff to occur with the irrigation by using the 'will_runoff' argument. 0 means no runoff, 1 means runoff like rain does.

```
irrigation apply amount = 50 (mm), will_runoff = 1
```

If this argument is not provided, it defaults to 0 (the irrigation does not runoff)

Subsurface Irrigation¶

You can apply an irrigation to be a subsurface irrigation by using the optional 'depth' argument.

```
irrigation apply amount = 50 (mm), depth = 3000 (mm)
```

The SoilWater module will work out what layer of the soil that this depth corresponds to, and will put the water directly into this soil layer.

Solutes in Irrigation

You can apply solutes with the irrigation by using the optional 'no3', 'nh4' and 'cl' arguments.

```
   irrigation apply amount = 50 (mm), no3 = 50 (kg/ha), nh4 = 60 (kg/ha), cl
= 70 (kg/ha)
```

nb. cl is chloride.

You can use this in conjunction with the depth argument.

```
   irrigation apply amount = 50 (mm), depth = 3000 (mm), no3 = 50 (kg/ha),
nh4 = 60 (kg/ha), cl = 70 (kg/ha)
```

nb. each one of these solutes is an optional argument. So you do not need to provide all three. You can provide just one or two of them if you want.

Resetting Schedules

Manual and automatic schedules can be enabled and disabled within a simulation via the message system. The syntax of a standard manager message to turn off the manual irrigation schedule would be as follows:

```
    irrigation set manual_irrigation = off
```

Other related data is unaffected by this switch resetting and so, for example, the automatic irrigation scheduling can be set on and off for set windows in time using a pair of if statements in the manager file.
Parameters for the automatic irrigation calculations can be reset similarly:

```
    irrigation set crit_fr_asw = 0.9 (0-1)
    irrigation set asw_depth = 150 (mm)
```

Working with Irrigation Allocation Budgets

The following example shows how the allocation mechanisms are utilized in the APSIM Irrigate module.
Note that only automatic irrigation scheduling and remote scheduling (eg via the manager or operations modules) is taken into account.
The mechanism cannot be used in conjunction with a manual irrigation schedule.
The example shows an annual allocation set on the first of July each year, which is applied using automatic irrigation scheduling.

```
user_data_group.irrigate .parameters

automatic_irrigation = on             (on/off) ! switch schedule on or off
crit_fr_asw = 0.66                       (0-1) ! critical fraction of
                                               ! available soil water
                                               ! to trigger irrigation
asw_depth = 600                           (mm) ! depth for available
                                               ! soil water calculations

user_data_group.manager .start_of_day
if today = date('1_jul') then
  irrigation set allocation = 1000 (mm)
endif
```

Working with Irrigation Efficiency

The following example shows how the irrigation efficiency is utilized in the APSIM Irrigate module.
The example shows an irrigation_efficiency setting of 75%.
This means that only 75% of the irrigation is actually being applied due to approximated losses due to evaporation, wind loss or runoff.

```
user_data_group.manager .start_of_day

irrigation.irrigation_efficiency = 0.75

if today = date('1_jul') then
  irrigation apply amount = 50
endif
```

Using APSIM Irrigate with APSIM WaterStorage

There is an option to use APSIM Irrigate in conjunction with the APSIM WaterStorage module. Please also read the documentation for APSIM WaterStorage.

APSIM Irrigate works on a ‘mm' basis, whereas APSIM WaterStorage works on real volumes (Ml). Hence, when an irrigation application is specified in mm, an ‘area of application' must be provided in order to calculate the required volume of water from the specified source instance of WaterStorage. As mentioned previously, whenever WaterStorage is used in a simulation for supply of irrigation water, a variable called ‘crop_area' (ha) must be specified in the manager logic.

Irrigations using water from WaterStorage can only be initiated by using the ‘irrigation apply' action in manager. A new optional argument called ‘source' is added to the ‘apply' command line to trigger the use of water from WaterStorage. The required syntax is as follows:

```
sample.manager.start_of_day
if day = 10 then
  irrigation apply amount=10 (mm), source = dam bore dam2 ()
endif
```

The argument ‘source' specifies the sources from which to obtain the irrigation water, in preferential order. In other words, in the above example, if the dam cannot fully supply the required water, the balance will be taken from the bore. If there is still a shortage of water, then dam2 will be asked next to supply water. There is no limit to the number of sources which can be specified.
When the irrigation water is applied to the soil, it will carry the solutes makeup of the water source being used.

Irrigation Module Outputs

The Irrigation module outputs the following variables.

Variable Name	Description
irrigation	Total amount of irrigation added to profile during any timestep (mm)
manual_irrigation	Current state of the fixed irrigation schedule (on/off)
automatic_irrigation	Current state of the automatic irrigation schedule (on/off)
crit_fr_asw	Critical fraction of available soil water (ie. above a 15 bar lower limit) below which irrigation is automatically applied. (0-1)
asw_depth	Depth to which available soil water fraction is calculated. (mm)
allocation	Current amount of irrigation allocation available for use (mm)
allocation_ml	Current amount of irrigation allocation available for use (ML)
carry_over	Amount of irrigation allocation unused as at reset of allocation (mm) (Value will be zero for days on which allocation is not reset)
carry_over_ml	Amount of irrigation allocation unused as at reset of allocation (ML)
irr_fasw	Fraction of available soil water within the critical irrigation soil depth
irr_deficit	Deficit of soil water within the critical irrigation soil depth (mm)
irrig_loss	Losses resulting from and application of irrigation in conjunction with the irrigation_efficiency mechanism.
irrig_tot	Total irrigation specified, not including losses due to irrigation efficiency (mm)
irrigation_XXX	Applied quantity of solute XXX, (kg/ha)

Map

How to Use Map

The Map module maps simulation soil layers onto output layers.
Numerical simulation of water and solute is likely to require many more layers than the user either wants to know about in the outputs or has data to compare against.
Map groups simulation layers into more useful output layers using an array of coefficients.
Output arrays may either be expressed as sums, averages, or concentrations of the contributing simulation layers.
Map can map the same array in several different ways (eg. Below nitrate, no3n, is converted to a sum, a concentration, and a concentration in the soil water) but there can be only one layer structure.

Example

```
[test.map.parameters]
arrays2sum_names = no3n nh4n dlayer
arrays2ave_names = soil_temp
arrays2conc_names = no3n
arrays2concsw_names = no3n
arrays2satpaste_names =
core_start = 0 200 400 600 800
core_end = 200 400 600 800 900
```

To Use Supply

arrays2sum A list of array names owned by other modules whose elements are to be summed into a more useful output range.
The name of the mapped array will be that of the original array but preceded by ‘map_'.
Currently a maximum of 10 arrays can be summed.

arrays2ave A list of array names owned by other modules whose elements are to be averaged into a more useful output range.
The name of the mapped array will be that of the original array but preceded by ‘map_'.
Currently a maximum of 10 arrays can be averaged.

arrays2conc A list of array names owned by other modules whose elements are to be converted into concentrations in the soil volume.
The name of the mapped array will be that of theoriginal array but preceded by ‘conc_'.
Currently a maximum of 10 arrays can be converted to concentrations.

arrays2concsw As above except the concentrations will be expressed as the concentration in the soil water.

The name of the mapped array will be that of the original array but preceded by ‘concsw_'.
Currently a maximum of 10 arrays can be converted to concentrations.

arrays2satpaste As above except the concentrations will be expressed as the concentration in a saturation paste.
Because there is no standard method of estimating the saturation paste water content from the soil's hydraulic properties the user must supply that water content.
Map will look for an array, satpaste_wc – the saturation paste gravimetric water content – units of g /g, to be provided from another module (e.g. Manager, Input).
The name of the mapped array will be that of the original array but preceeded by ‘satpaste_'.
Currently a maximum of 10 arrays can be converted to saturation paste concentrations.

core_start/core_end A list of the start and end depths (in mm) for each layer for which you want values reported.
There can be only one if you want - this is a simple way of finding the total amount of a substance to some depth which is not the simulation depth.
The depths do not have to be contiguous, there can be gaps on the layers, the layers do not have to be in increasing depth order, point depths (core_start = core_end) are acceptable (and in this case all Map does is to find the appropriate layer).
The only requirements are that there are pairs of core_start and core_end and that for any pair core_start <= core_end.

SoilN

Description

The SoilN module describes the dynamics of **both carbon and nitrogen** in soil.

The transformations considered in each layer are shown diagramatically in Figure 1.

The major difference from the CERES model is that the soil organic matter is divided into two pools (**biom** and **hum**),
the **biom** pool notionally representing the more labile, soil microbial biomass and microbial products, whilst **hum** comprises the rest of the soil organic matter.

The flows between the different pools are calculated in terms of carbon, the corresponding nitrogen flows depending on the C:N ratio of the receiving pool.

The C:N ratios of the various pools are assumed to be constant through time; C:N for **biom** is specified in the ini file, whilst the C:N of **hum** is derived from the C:N ratio of the soil which is an input.

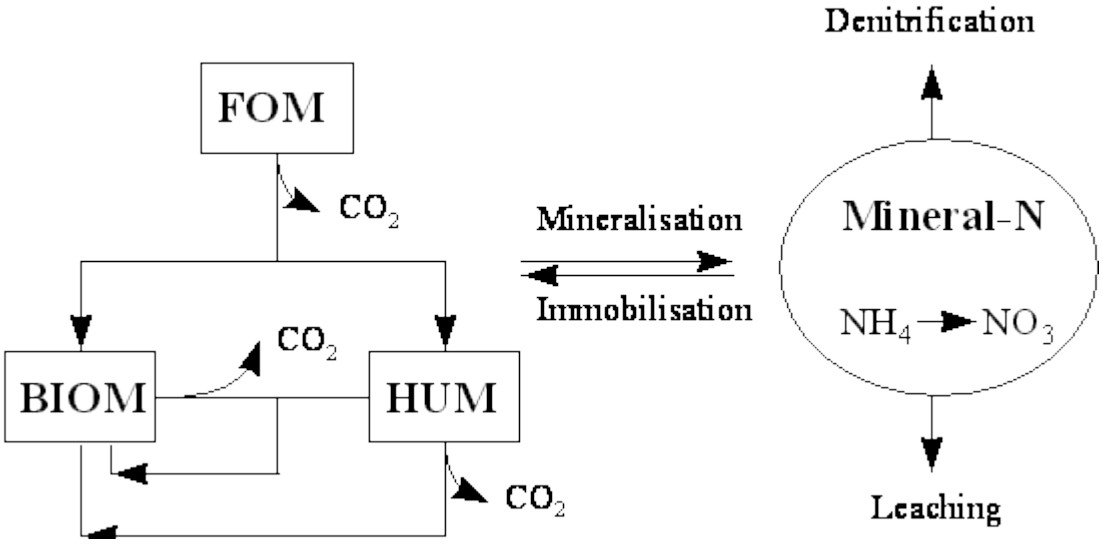


Figure 1. Diagram of transformations occuring in each soil layer

Decomposition of **biom** and **hum** pools are calculated as first-order processes with the rate constants being modified by factors involving soil temperature and moisture in the layer.

The fresh organic matter pool (**fom**) is treated as in CERES-Maize (Jones and Kiniry, 1986), and its rate of decomposition also depends on a C:N ratio factor.

Mineralisation or immobilisation of mineral-N is determined as the balance between the release of nitrogen during decomposition and immobilisation during microbial synthesis and humification.

An inadequate supply of mineral-N to satisfy the immobilisation demand results in a slowing of the decomposition.

Both ammonium- and nitrate-N are available for immmobilisation, though ammonium-N is used preferentially.

Decomposition of any organic matter pool results in evolution of carbon dioxide to the atmosphere and transfers of carbon to the **biom** and **hum** pools.

The flows are defined in terms of efficiency coefficients, representing the proportion of carbon retained in the system,
and the fraction of the retained carbon that is synthesised into the **biom** pool (see Table 1 for parameter definitions and values).

When **biom** decomposes there is an internal cycling of carbon (microbes feeding on microbial products).

At initialisation, the amounts of **hum** and **biom** in each layer are calculated from inputs (oc, finert, fbiom).

To allow for slower rates of decomposition of soil organic matter in the deeper soil layers, part of the **hum** pool is considered to be non-susceptible to decomposition; this is specified as finert, which typically will increase with depth.

Fbiom specifies the **biom** pool carbon as a fraction of the **hum** carbon that is subject to decomposition,
i.e. fbiom = **biom**/(**hum** - inert_c)

Table 1.

Model parameters determining the flows of carbon during the decomposition of the soil organic matter pools and surface residues

Parameter	Value	Definition
mcn	8.0	C:N ratio of **biom** pool
ef_fom	0.4	efficiency of carbon retention when **fom** decomposes
fr_fom_biom	0.9	proportion of retained carbon from **fom** synthesised into **biom**
ef_biom	0.4	efficiency of carbon retention when **biom** decomposes
fr_biom_biom	0.6	proportion of retained carbon from **biom** resynthesised into **biom**
ef_hum	0.4	efficiency of carbon retention when **hum** decomposes
ef_res	0.4	efficiency of carbon retention when residues decompose
fr_res_biom	0.9	proportion of retained carbon from residues synthesised into **biom**
rd_carb	0.2/day	maximum decomposition rate for carbohydrate-like C in **fom**.

Parameter	Value	Definition
rd_cell	0.05/day	maximum decomposition rate for cellulose-like C in **fom**
rd_lign	0.0095/day	maximum decomposition rate for lignin-like C in **fom**
rdbiom	0.0081/day	maximum decomposition rate for **biom**
rdhum	0.00015/day	maximum decomposition rate for **hum**

Soil Temperature

Daily average soil temperature of each soil layer is calculated based on the soil temperature model of EPIC (Williams et al., 1984).
A sinusoidal function of day of year is assumed, with subsurface temperature changes lagging behind those at the soil surface.
The sinusoidal function is based on mean annual air temperature (tav), annual amplitude in mean monthly air temperature (amp) and latitude (included in met file).
The actual surface temperature is calculated from maximum and minimum temperatures, solar radiation and soil albedo.
Changes with depth are obtained from an exponential function of the ratio of depth and a temperature damping depth.
This damping depth is a function of the average bulk density of the soil and the amount of water above the lower limit.

Decomposition of Soil Organic Matter Pools

fom decomposition = Fpool (carbohydrate,cellulose or lignin fraction) x decay rate for a given fraction (rd_carb, rd_cell, rd_lign) x Soil water factor x Soil temperature factor x C:N ratio factor

biom decomposition = **biom** x rd_biom x Soil water factor x Soil temperature factor

hum decomposition = (**hum** - inert_C) x rd_hum x Soil water factor x Soil temperature factor

The factors affecting the individual decay rates are as follows:

(i) Soil Water

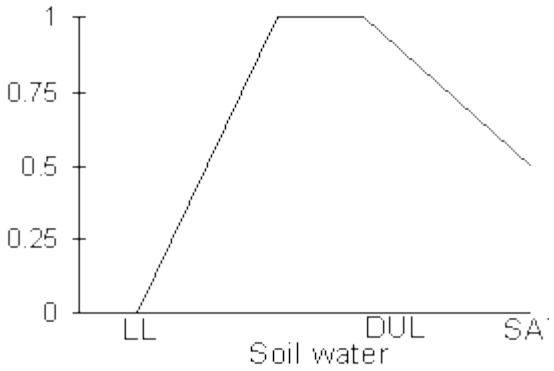


Figure 2. Water factor affecting mineralisation rates of the various soil organic matter pools.

(ii) Soil Temperature

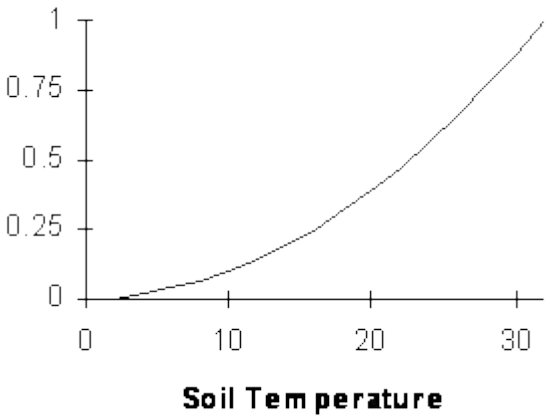


Figure 3. Temperature factor affecting mineralisation rates of the various soil organic matter pools.

(iii) C:N ratio

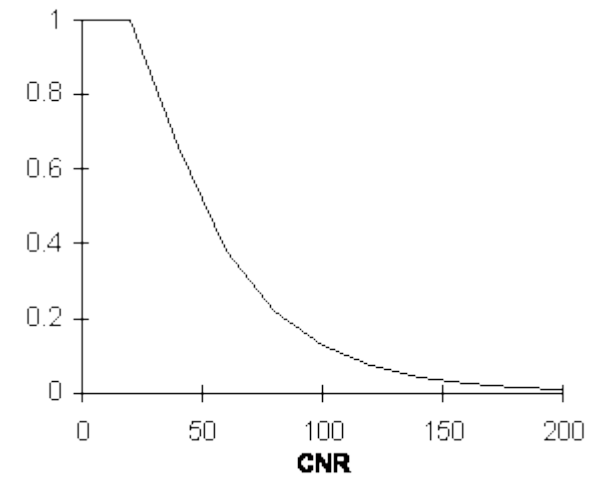


Figure 4. C:N ratio factor affecting mineralisation rate of soil FOM pools is calculated using a modified C:N ratio that includes the mineral nitrogen in the soil layer.

CNR = fom_C / (fom_N + min_N)

Nitrification

Nitrification is assumed to follow Michaelis-Menton kinetics (See Godwin and Jones 1991, though there is error in their equation 14).

potential rate = nitrification_pot x NH4 ppm / (NH4 ppm + NH4_at_half_pot)

where nitrification_pot (mg N/kg soil/ day) and NH4_at_half_pot (ppm) are specified in the SOILN ini file. Actual daily nitrification is reduced to allow for sub-optimal water, temperature and pH conditions.

nitrification rate = potential rate x min (water factor, temperature factor, pH factor)

(Unlike CERES, there is no provision for the potential rate of nitrification to change with time to represent a changing microbial population)

(i) Soil Water

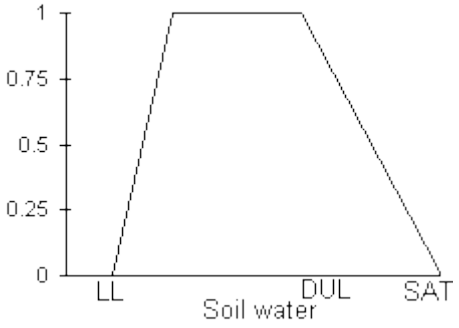


Figure 5. Water factor affecting the nitrification rate of ammonium in each soil layer.

(ii) Soil Temperature

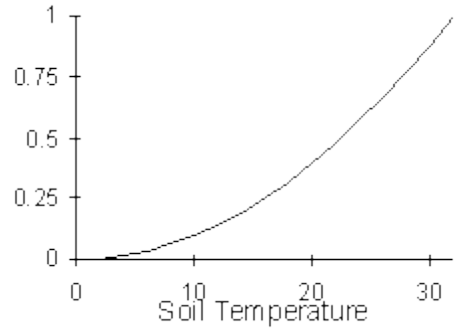


Figure 6. Temperature factor affecting the nitrification rate of ammonium in each soil layer.

(iii) pH

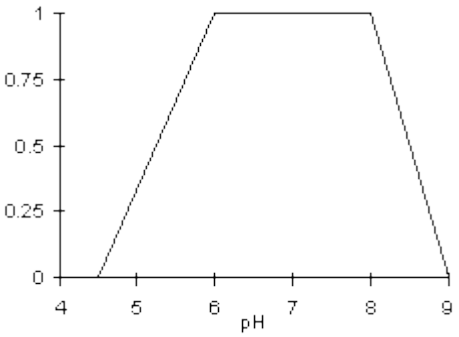


Figure 7. pH factor affecting the nitrification rate of ammonium in each soil layer.

Nitrous oxide from nitrification:

Note: Nitrous oxide predictions has been tested under a limited range of soil/climate and production systems
and although this testing produced sensible results there should still be a higher degree of user caution attached to the predictions.

Nitrous oxide (N2O) emission during nitrification (N2Onit) is calculated as a proportion (k2) of nitrified N (Li, 2000; Parton et al., 2001; Li et al., 2007):

N2Onit = k2 * Rnit

where Rnit is the rate of nitrification (kg N/ha/day).

A wide range of values have been adopted for k2 in agricultural soils in other models, potentially reflecting the uncertainty in the process resulting in N2O emissions during nitrification

1. Li, 2000; Li et al., 2000; Parton et al., 2001; Li et al., 2007). Following Li et al.(2007) we adopt a value of 0.002, but acknowledge that values may be soil-specific.

2. Li, C.S., 2000. Modeling trace gas emissions from agricultural ecosystems, Nutr. Cyc. Agroecosys. 58, 259-276.
3. Li, C., Aber, J., Stange, J., Butterbach-Bahl, K., Papen, H., 2000. A process-oriented model of N2O and NO emissions from forest soils: 1 Model development. J. Geophys. Res. 105(D4). 4369–4384.
4. Parton, W.J., Holland, E.A., Del Grosso, S.J., Hartman, M.D., Martin, R.E., Mosier, A.R., Ojima, D.S., Schimel, D.S., 2001. Generalized model for NOx and N2O emissions from soils. J. Geophys. Res. 106(D15),17403-17419.
5. Li, Y., White, R.E., Chen, D.L., Zhang, J.B., Li, B.G., Zhang, Y.M., Huang, Y.F. Edis, R., 2007. A spatially referenced water and nitrogen management model (WNMM) for (irrigated) intensive cropping systems in the North China Plain. Ecol. Mod. 203, 395-423.

Denitrification

The code comes from CERES-Maize V1 and has not been modified to allow for biom_c.
Under most situations, simulated denitrification rates are small.
The sensibleness of the simulations where significant amounts of denitrification occur has not been verified.

denitrification rate = 0.0006 x NO 3 x active carbon ppm x water factor x temperature factor

where,
active carbon ppm = 0.0031 x (hum_C ppm + FOM_C ppm) + 24.5

(i) Soil Water

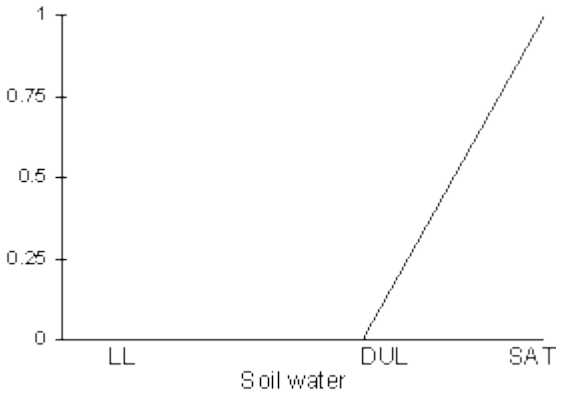


Figure 8. Water factor affecting the denitrification of nitrate in each soil layer.

(ii) Soil Temperature¶

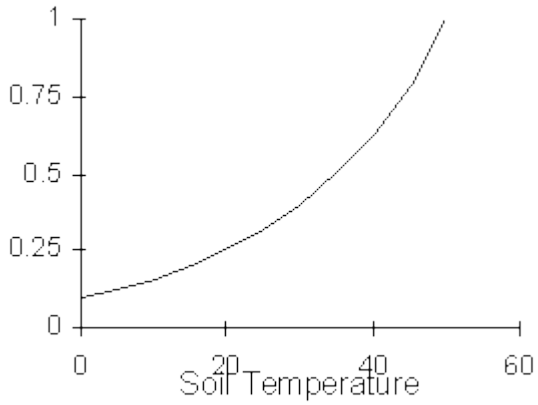


Figure 9. Temperature factor affecting the denitrification of nitrate in each soil layer.

Nitrous oxide from denitrification:

Note: Nitrous oxide predictions has been tested under a limited range of soil/climate and production systems
and although this testing produced sensible results there should still be a higher degree of user caution attached to the predictions.

Nitrous oxide (N2O) emission during denitrification (N2Odenit) is calculated by combining predictions of denitrification with the ratio of N2 to N2O emitted during denitrification predicted by the model of Del Grosso et al. (2000):

N2/N2Odenit = Max {(0.16 k1), (k1 exp(-0.8 NO3 ppm/CO2)) } * Max {0.1, ((1.5 WFPS) - 0.32) }

where,
k1 is related to gas diffusivity in soil at field capacity,
NO3ppm (µg/g) is the nitrate concentration of the soil on a dry weight basis, and
CO2 is the heterotrophic CO2 respiration (µgC/g soil / day).

1. Del Grosso, S.J., Parton, W.J., Mosier, A.R., Ojima, D.S., Kulmala, A.E., Phongpan, S., 2000. General model for N2O and N-2 gas emissions from soils due to denitrification. Glob. Biogeochem. Cycl. 14, 1045-1060.

Urea hydrolysis:

potential hydrolysis fraction = -1.12 + 1.31xOC + 0.203xpH - 0.155xOCxpH

This fraction is bound between 0 and 1. For OC=1% and pH=7 the fraction=0.526

hydrolysis rate = Urea x potential hydrolysis fraction x min (temperature factor, water factor)

(i) Soil Water

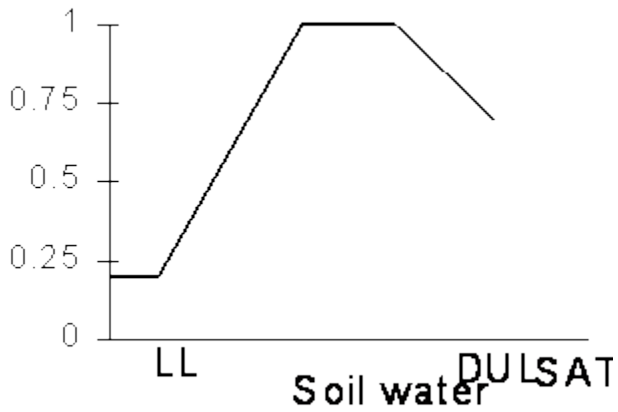


Figure 10. Water factor affecting the hydrolysis of urea in each soil layer.

(ii) Soil Temperature

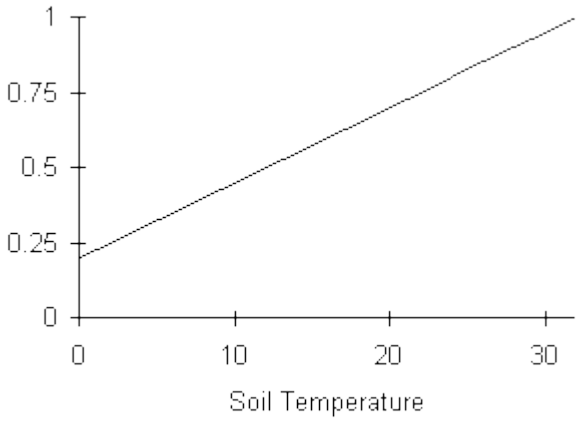


Figure 11. Temperature factor affecting hydrolysis of urea in each soil layer.

SoilN module actions

Reset

The reset action can be invoked to reset the module to the state specified within the module's input data,
which includes the soil's organic carbon components (biom and hum), nitrate, ammonium and urea N, and FOM (specified as root weight).

The Reset action is identical to the initialise action used by the simulation engine at the start of the simulation except that a description of the reinitialised state is not printed in the simulation summary file.
Where only mineral N concentrations need to be reinitialised the 'set' action is used (See below).

APSIM Manager Example:

```
[sample.manager.start_of_day]

! reinitialise soil nitrogen at the beginning of each sowing window

If day = 100 then
   soiln2 reset
endif
```

Initialise

The initialise action has been replaced by the reset action (see above).

Set

The set action is used to reinitialise single mineral N pools within the module (Nitrate, Ammonium or Urea). Only the particular mineral pools are reset.

APSIM Manager Example:

```
[sample.manager.start_of_day]

! reinitialise soil mineral nitrogen at given date

If day = 100 then
   soiln2 set no3 = 10 5 1 (kg/ha)
   soiln2 set nh4 = 10 5 1 (kg/ha)
endif
```

or in terms of concentration

```
[sample.manager.start_of_day]

If day = 100 then
   soiln2 set no3ppm = 1.0 1.0 1.0 (ppm)
   soiln2 set nh4ppm = 1.0 1.0 1.0 (ppm)
endif
```

Summary Report

At initialisation, at series of tables and useful information is printed to the simulation summary file for perusal by the user.
These tables can be printed to the summary file at any point during the simulation as a detailed record of the system state at a particular time.

APSIM Manager Example:

```
[sample.manager.start_of_day]

! Print out a summary of module state to the summary file

If day = 100 then
    soiln2 sum_report
endif
```

Incorporate FOM

Fresh organic matter can be incorporated into the soil using the incorp_fom action.
The amounts of FOM and FON are specified on a layered basis .
An optional FOM type can be specified to use any preset fractionations of FOM as specified in the module ini file.
A default fractionation is used if type is not specified.
FON content can be specified as either an amount or a C:N ratio.

APSIM Manager Example:

NOTE: actions must be specified as a single line in the manager file

```
If day = 100 then
    soiln2 incorp_fom dlt_fom_type = manure, dlt_fom_wt = 100 50 0 (kg/ha),
dlt_fom_n = 4 2 0 (kg/ha)
endif
```

or assuming that FOM is 40% carbon we can use a C:N ratio.

```
If day = 100 then
   soiln2 incorp_fom dlt_fom_type = manure, dlt_fom_wt = 100 50 0 (kg/ha),
dlt_fom_cnr = 10 10 0 (kg/ha)
endif
```

SoilN module outputs

Name	Units	Description
no3	kg/ha	amount of NO3 in each layer
no3ppm	ppm	concentration of NO3 in each layer
no3_min	ppm	minimum amount of NO3 allowed in each layer
nh4	kg/ha	amount of NH4 in each layer
nh4ppm	ppm	concentration of NH4 in each layer
nh4_min	ppm	minimum amount of NH4 allowed in each layer
urea	kg/ha	amount of urea in each layer
fom_n	kg/ha	amount of N in FOM in each layer
hum_n	kg/ha	amount of humic N in each layer
biom_n	kg/ha	amount of biomass N in each layer
fom_c	kg/ha	amount of C in FOM in each layer
hum_c	kg/ha	amount of humic C in each layer
biom_c	kg/ha	amount of biomass C in each layer
carbon_tot	kg/ha	Total amount of C in each layer
nit_tot	kg/ha	Total amount of N in each layer
dlt_n_min	kg/ha	Today's net N mineralisation in each layer from soil organic matter pools
dlt_n_min_res	kg/ha	Today's net N mineralisation in each layer from surface residues
dlt_n_min_tot	kg/ha	Today's net N mineralisation in each layer from soil organic matter pools and surface residues.
dnit	kg/ha	Today's denitrification loss for each layer
dlt_c_loss_in_sed	kg/ha	Loss of C in sediment due to erosion
dlt_n_loss_in_sed	kg/ha	Loss of N in sediment due to erosion

SoilP

SoilP Module Scope

SoilP is a representation of the availability of phosphorus in soil that provides an index for the soil's ability to supply P to crops that can be used in crop modules to modify growth processes under P limiting conditions.

The module is designed to handle inputs of fertiliser that are either immediately available or slow acting (e.g. rock phosphate), and the improved effectiveness of fertiliser due to placement effects.

To link SoilP with the MANURE model, the MANURE module handles the release of P from manure on the soil surface as a function of time and moisture content.
This released P then becomes an input to the SoilP modules in the same manner that fertiliser P is an input.
Similarly the corresponding release of N from manure is transferred to the appropriate pools in SoilN.

The dominant processes considered by SoilP are:

- loss of availability through time
- removal by crops
- addition by crop residues
- mineralisation / immobilisation of soil organic P.

SoilP is a multi-layer module that uses the same soil layer structure as SoilN.
This approach has been adopted in order that there can be a full accounting of the organic P components in the system.

The availability of P in soil is dominantly influenced by the near-surface layers,
and crop response to fertiliser is often adequately described by soil tests, P sorption etc of surface soil samples, i.e. without consideration of subsoils.

In most situations,
extractable P decreases with depth,
P sorption increases,
and root density decreases;
all three factors contribute to the relative unimportance of the subsoil for P nutrition.

The multi-layer nature of the module requires assumptions to be made as to how P uptake by crops is to be partitioned between layers in the soil profile.

SoilP Module History

Within the CARMASAT Project, there is a need to develop a capability for modelling the nutritional effects of manure,
and in particular to accommodate the effects of both nitrogen and phosphorus in manure on crop growth.

In order that this can be achieved, an essential stepping stone is the incorporation into crop modules of routines that constrain growth under conditions of limiting P supply.

This, in turn, requires a module that specifies the P supply from the soil, namely SoilP.

Citations

1. Barrow, N.J. (1974). The slow reactions between soil and anions. 1. Effects of time, temperature, and water content of a soil on the decrease in effectiveness of phosphate for plant growth. *Soil Science* 118, 380-386.
2. Probert, M.E. (1985). A conceptual model for initial and residual responses to phosphorus fertilizers. *Fertilizer Research* **6**, 131-8. *(This paper contains the broad principles of how the availability of P is conceptualised in SOILP)*
3. Probert, M.E. and Okalebo, J.R. (1992). Effects of phosphorus on the growth and development of maize. *In* "A search for strategies for sustainable dryland cropping in semi-arid eastern Kenya". (Ed M.E. Probert). ACIAR Proceedings No 41. pp 55-62.

SoilP Module Structure

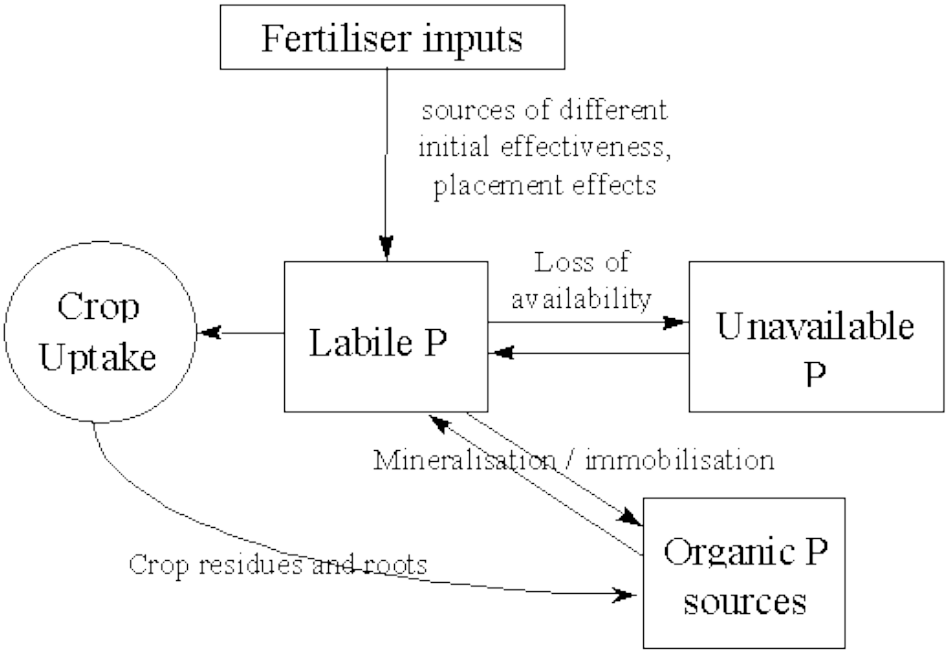


Figure 1. Schematic representation of the SoilP module showing the principal processes considered.

SoilP Module Components

Initialising SoilP

Inputs required by the SoilP module are:

- Labile_p in each layer (ppm)
- Sorption in each layer. These values are the total P sorbed at an equilibrium concentration of 1 ppm in solution, that is the value of *a* when the Freudlich equation is used to describe P sorption (*Psorbed = ac b*).
- Residue_cp which is the C:P ratio of the initial above ground crop residues.
- Root_cp which is the C:P ratio of the initial roots.

- Rate_dissol_rock_p which is the rate of dissolution of any rock phosphate present at initialisation.
- The unavailable_p, banded_p and rock_p pools on a layer basis are optional inputs. In the absence of any of values, banded_p and rock_p will be initialised to zero and the unavailable_p pool will be set such that the labile_p « unavailable_p system is at a steady state.

Currently labile_p is the only permitted input. In the future, input of extractable soil P (with options for different extractants) could be input and labile_p calculated.

Adding P as Fertiliser

The nature of the P source and how it is placed determines how it modelled:

- water soluble and broadcast P is added into labile_p,
- water soluble and placed P is added into banded_p,
- any non water soluble P (eg rock phosphate) is added into rock_p

Applied P fertiliser can be any combination of these three forms.

Processes of P in Soil

1. **Loss of availability with time**.
   The rate for labile_p to unavail_P (and banded_p to unavail_p) is dependent on temperature and moisture (Barrow).
   The reverse process, unavail_p to labile_p will have a rate constant commensurate with the relative sizes of the two pools under steady state conditions.
2. **Dissolution of rock_p** to labile_p.
   The rate coefficient depends on the source and the soil type (soil pH, calcium status, etc.).
   It is an input for any source/soil combination.
   A typical value is about 0.2 yr^{-1} .
3. **Decrease in effect of placement with time**, ie banded_p to labile_p.
   The effectiveness of placed P is assumed to decrease with time.
   Typically 50% of effect is lost in one year.
   A **tillage event** results in total transfer of banded_p to labile_p.
4. **Mineralisation of organic P sources**to labile P.
   This is linked to decomposition of carbon from HUM, BIOM and FOM pools of SoilN module and surface residues in Residue module.
   SoilP assumes constant C:P ratios in BIOM and HUM but tracks C:P ratios of FOM and surface residues as crops add residues of varying P concentration.
5. **Removal of P by crop uptake**.
   P uptake is taken from labile_p and banded_p in proportion to their contributions to effective_p (see below).
6. **Addition of P by crop death/senescence**.
   P in above and below ground parts of the plant are returned to the Residue and FOM pools respectively.

Effective P in Soil

When P is banded, especially on soils of high sorptivity, it is more effective in increasing P uptake than a corresponding broadcast application.

This effect is simulated in SoilP by placing a premium on banded-p.

Hence,
effective_p = labile_p + eff_band * banded_p

where,
eff_band is the relative effectiveness of banded_p (>1) and would depend on the band spacing and the sorptivity of the soil.
Typically it is expected that eff_band has value of 2-3.

Module Interaction

An index of soil P status, as a function of effective_p, sorption, soil water, and rooting depth is made available to crop modules to determine whether the daily demand for P uptake can be met.

SoilP receives back the actual P uptake for the day.

SoilP Module Parameterisation

Variable / Parameter	Units	Definition
labile_p	kg/ha	soil P that is available for crop uptake
unavail_p (optional)	kg/ha	soil P that is not available for crop uptake but can become available with time to replenish P removal by crops. If not specified, then initialised to steady state.
banded_p (optional)	kg/ha	water soluble fertiliser P applied as a band. If not specified then initialised to zero.
rock_p (optional)	kg/ha	non water soluble fertiliser P. If not specified then initialised to zero.
rate_dissol_rock_p	yr^{-1}	rate coefficient determining release of available P from non water soluble source
availP_ratio	-	ratio of available P : unavailable P at a steady state
sorption	mg/kg	soil's P sorption characteristic
root_cp	-	C:P ratio of roots at initialisation
residue_cp	-	C:P ratio of surface residues at initialisation

SoilP Module Dependencies

SOILP uses an APSIM water balance module (eg SOILWAT) to determine the soil water status.

The dynamics of P in organic matter provided by SOILP requires the soil nitrogen (SOILN) and residue (RESIDUE) modules for carbon cycling.

SoilP Module Outputs

Variable Name	Units	Description
labile_p	kg/ha	soil P that is available for crop uptake
unavail_p	kg/ha	soil P that is not available for crop uptake but which can become available with time to replenish P removal by crops
banded_p	kg/ha	water soluble fertiliser P applied as a band
biom_p	Kg/ha	
hum_p	kg/ha	
fom_p	kg/ha	
uptake_p_***cropname***		P uptake for crop ***cropname***

SoilP Module Special Considerations

Simulation of crop growth under conditions of limiting P supply requires crop modules to provide the following:

1. inclusion of P stress factors (derived from P content of the crop or part of crop, optimum and minimum P concentrations which are functions of stage of growth)
   to restrict growth rates and modify phenological development and partitioning of biomass between tops and roots
2. calculation of daily P uptake by crop which is passed to SoilP so that P can be taken up from appropriate pools/layers
3. when crop residues (tops or roots) cease to be part of the plant and are added to the soil or surface residues (e.g. at harvest or due to senescence),
   their P content has to be returned, along with mass and N content, to maintain the P balance for the crop/soil system.

SoilP Module Validation

The qualitative response of SoilP in terms of decreasing availability with time, more rapid loss of availability at higher temperature, effect of P sorption on crop uptake (by maize) has been verified.

The only data set where SoilP and a modified Maize module (which includes routines to enable the crop to respond to P limitations) have been tested against observed data is for an experiment in Kenya described by Probert and Okalebo (1992).

SoilTemp

General description

Simulates soil temperature given minimal input information using a numerical scheme.

Usage

Soiltemp requires

1. water content information from a water module,
2. air temperature from the input module,
3. and where available will use soil water evaporation and incident net radiation as part of the upper boundary condition.

Theory

The node/element scheme for the numerical simulation is shown in Figure 1 where the circles denote the nodes and the zigzag's show the elements.
All heat storage is assumed to occur at the nodes and all resistance to heat transfer is assumed to take place within the elements.

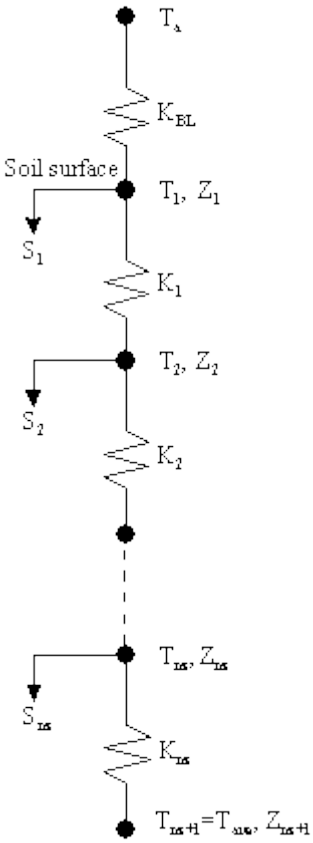


***Figure 1**. A diagrammatic representation of the node structure of the numerical simulation.*

where in Figure 1. ,
T is temperature,
Z is depth,
K is thermal conductance,
S is heat storage,
nz is the number of nodes in the simulation,
BL stands for boundary layer,
T_a is the air temperature, and
T_{ave} is the annual average soil temperature.

We begin by defining the heat balance at a node i, where i ≠ 1 or nz.
The decrease in storage of heat at the node over a particular timestep is equal to the amount of heat leaving the node minus the amount of heat coming in all divided by the timestep so that,

$$\frac{-\Delta \text{storage}}{\Delta t} = \frac{\text{heat out} - \text{heat in}}{\Delta t} \quad , [1]$$

where in Equation 1. ,
Delta stands for a change and
t is time (s).

Also In Equation 1. ,
Storage of heat is calculated from the change in temperature with time and the volumetric specific heat of the soil so that,

$$\Delta \text{storage} = \left(T_i^{j+1} - T_i^j\right) C_i A \frac{\left(z_{i+1} - z_{i-1}\right)}{2} \quad , [2]$$

in Equation 2. ,
i stands for a particular node,
j refers to the current timestep and j+1 to the next timestep,
T is temperature (^oC),
C is the volumetric specific heat of the soil (J K^{-1} m^{-3}),
A is the cross-sectional area and is normally taken as 1 m^2 , and
z is depth (m). Because all heat storage is assumed to occur at the nodes, the appropriate depth interval is half the distance to the node on either side of *i* .

In Equation 2. ,
C can be calculated from the specific heats of the soil constituents so that,

$$C_i = C_{clay}\left(1 - \frac{\rho_i}{\rho_s}\right) + C_{water}\theta_i + C_{air}\left(1 - \frac{\rho_i}{\rho_s} - \theta_i\right) \quad , [3]$$

in Equation 3. ,
Rho is the soil bulk density (Mg m^{-3}),
Rho_s is the particle density and is usually taken as 2.65 Mg m^{-3} ,
Theta is the volumetric water content (m^3 m^{-3}),
the subscripts on *C* indicate the substance.
C_{air} is usually assumed to be negligible while appropriate values of C for clay minerals and water are 2.39 and 4.18 MJ m^{-3} K^{-1} .
Where more precise values are required, the equation can be expanded to account for organic matter and other minerals.
See Campbell (1985) or Jury *et al.* (1991) for details on how to calculate C in more detail.

Also back in Equation 1.,
Heat transfer is calculated from the standard Fickian equation where the rate of transfer is proportional to the temperature difference divided by the distance,

$$\text{heat out}_i = \frac{-\lambda_i\left(\overline{T}_{i+1} - \overline{T}_i\right)\Delta t}{\left(z_{i+1} - z_i\right)} \quad , [4]$$

in Equation 4. ,
Lambda is the thermal conductivity (J s^{-1} m^{-1} K^{-1}) and
the overbar on T indicates that an appropriate average in time of temperature should be used.

in Equation 4.,
Lamda can either be calculated from measurements of temperature changes in the soil (see Jury *et al.* , 1991 for an example) or can be approximated from regression equations.

Below is given the scheme from Campbell (1985).
First four parameters dependent on density, water content, and the amount of clay are calculated. These are,

$$A = 0.65 - 0.78\rho + 0.60\rho^2$$
$$B = 1.06\rho\theta$$
$$C = 1 + 2.6\frac{1}{\sqrt{f_{clay}}}$$
$$D = 0.03 + 0.1\rho^2 \quad , [5]$$

in Equation 5. ,
f_{clay} is the fraction of clay in the soil.

The parameters in Equation 5 are combined into the equation for Lamda so that,

$$\lambda = A + B\theta - (A - D)e^{-(C\theta)^4} \quad . [6]$$

We are now able to put the elements of the equation together and rewrite Equation 1. as follows,

$$\frac{-C_i\left(T_i^{j+1}-T_i^{j}\right)A\frac{\left(z_{i+1}-z_{i-1}\right)}{2}}{\Delta t}=\frac{\frac{-\lambda_i\left(\overline{T}_{i+1}-\overline{T}_i\right)\Delta t}{\left(z_{i+1}-z_i\right)}-\frac{-\lambda_{i-1}\left(\overline{T}_i-\overline{T}_{i-1}\right)\Delta t}{\left(z_i-z_{i-1}\right)}}{\Delta t}$$. [7]

Now, in order to make some notational simplifications, define

$$H_i=\frac{C_iA\left(z_{i+1}-z_{i-1}\right)}{2\Delta t}$$, [8]

and

$$K_i=\frac{\lambda_i}{\left(z_{i+1}-z_i\right)}$$, [9]

and multiply by -1, so that Equation 7. can be rewritten as,

$$0=K_i\left(\overline{T}_{i+1}-\overline{T}_i\right)-K_{i-1}\left(\overline{T}_i-\overline{T}_{i-1}\right)-H_i\left(T_i^{j+1}-T_i^{j}\right)$$. [10]

The average temperature is defined as

$$\overline{T}=\eta T^{j+1}+(1-\eta)T^{j}$$, [11]

in Equation 11. ,
Eta is an operator controlling the forward/backward averaging of temperature.

Expanding the heat balance equation (Equation 10.),

$$\begin{aligned}0=&K_i\eta T_{i+1}^{j+1}+K_i(1-\eta)T_{i+1}^{j}-K_i\eta T_i^{j+1}-K_i(1-\eta)T_i^{j}\\&-K_{i-1}\eta T_i^{j+1}-K_{i-1}(1-\eta)T_i^{j}+K_{i-1}\eta T_{i-1}^{j+1}+K_{i-1}(1-\eta)T_{i-1}^{j}\\&-H_iT_i^{j+1}+H_iT_i^{j}\end{aligned}$$, [12]

and collecting on the terms of T, we can rewrite Equation 12. ,

$$
\begin{aligned}
0 = {} & T_{i-1}^{j+1}\left(K_{i-1}\eta\right) \\
& + T_{i}^{j+1}\left(-K_{i}\eta - K_{i-1}\eta - H_{i}\right) \\
& + T_{i+1}^{j+1}\left(K_{i}\eta\right) \\
& + T_{i-1}^{j}\left(K_{i-1}(1-\eta)\right) \\
& + T_{i}^{j}\left(-K_{i}(1-\eta) - K_{i-1}(1-\eta) + H_{i}\right) \\
& + T_{i+1}^{j}\left(K_{i}(1-\eta)\right)
\end{aligned}
\quad , [13]
$$

we can now put the unknown, or 'future', terms of T on the left hand side, we can rewrite Equation 13. ,

$$
\begin{aligned}
& -T_{i-1}^{j+1}\left(K_{i-1}\eta\right) \\
& -T_{i}^{j+1}\left(-K_{i}\eta - K_{i-1}\eta - H_{i}\right) \\
& -T_{i+1}^{j+1}\left(K_{i}\eta\right)
\end{aligned}
=
\begin{aligned}
& T_{i-1}^{j}\left(K_{i-1}(1-\eta)\right) \\
& +T_{i}^{j}\left(-K_{i}(1-\eta) - K_{i-1}(1-\eta) + H_{i}\right) \\
& +T_{i+1}^{j}\left(K_{i}(1-\eta)\right)
\end{aligned}
\quad , [14]
$$

the Equation 14. can now be expressed in matrix format,

$$
\begin{bmatrix}
\left(-K_{i-1}\eta\right) & \left(K_{i}\eta + K_{i-1}\eta + H_{i}\right) & \left(-K_{i}\eta\right)
\end{bmatrix}
\begin{bmatrix}
T_{i-1}^{j+1} \\ T_{i}^{j+1} \\ T_{i+1}^{j+1}
\end{bmatrix}
=
\begin{Bmatrix}
T_{i-1}^{j}\left(K_{i-1}(1-\eta)\right) \\
+T_{i}^{j}\left(K_{i}(1-\eta) + K_{i-1}(1-\eta) - H_{i}\right) \\
+T_{i+1}^{j}\left(K_{i}(1-\eta)\right)
\end{Bmatrix}
\quad [15]
$$

in Equation 15., if the **three** elements of the left-most matrix are expressed as
a_i ,
b_i ,
and c_i ,
and the right hand side as
d_i ,
Then Equation 15. can be expanded as an example for a simulation of nz nodes,

$$
\begin{bmatrix}
b_1 & c_1 & 0 & \cdots & 0 & 0 \\
a_2 & b_2 & c_2 & \cdots & 0 & 0 \\
0 & a_3 & b_3 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & b_{nz-1} & c_{nz-1} \\
0 & 0 & 0 & \cdots & a_{nz} & b_{nz}
\end{bmatrix}
\begin{bmatrix}
T_1^{j+1} \\ T_2^{j+1} \\ T_3^{j+1} \\ \vdots \\ T_{nz-1}^{j+1} \\ T_{nz}^{j+1}
\end{bmatrix}
=
\begin{bmatrix}
d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{nz-1} \\ d_{nz}
\end{bmatrix}
. [16]
$$

<u>We now have a tri-diagonal matrix which can be solved using the Thomas algorithm (Carnahan *et al.* , 1969).</u>

The lower boundary condition is taken as a zero flux condition, ie. constant temperature, in which T_{nz+1} equals a constant which is usually taken as the average annual temperature.

$$d_{ns} = \left\{ T^j_{ns-1}\left(K_{ns-1}(1-\eta)\right) + T^j_{ns}\left(-K_{ns}(1-\eta) - K_{ns-1}(1-\eta) + H_{ns}\right) + T^j_{ns+1}\left(K_{ns}(1-\eta)\right)\right\} + \\ + \eta K_{ns} T^j_{ns+1} \qquad [17]$$

The upper boundary condition is more complex.
Usually the known temperature is the air temperature at some height above the soil and d_1 is expressed as,

$$d_1 = \left\{ T^j_{i-1}\left(K_{i-1}(1-\eta)\right) + T^j_i\left(-K_i(1-\eta) - K_{i-1}(1-\eta) + H_i\right) + T^j_{i+1}\left(K_i(1-\eta)\right)\right\} + \\ + \eta K_{BL} T_{air} - R_n + E_{soil} \qquad , [18]$$

in Equation 18. ,
K_{BL} is the boundary layer conductance (J s^{-1} m^{-2} K^{-1}),
T_{air} is the temperature at the top of the boundary layer,
R_n is the net radiative input (J s^{-1} m^{-2}), and
E_{soil} is the evaporative loss of energy from the soil surface (J s^{-1} m^{-2}).

Implementation

Soiltemp is designed to independent of the Apsim timestep.

To allow for the numerical solution, the equations above ([16],[17] and [18]) are solved 48 times within each Apsim timestep.

When Apsim is running on its customary daily step (24 hours) this allows for a **half-hourly** (24 hours/48) internal time step,
which should be more than sufficient for numerical stability.

Estimating upper boundary condition (Change in air temperature)

When the Apsim timestep is daily(24 hours)

Soiltemp estimates the **changes in air temperature**, the **upper boundary condition (Equation 18.)**, in the following manner:

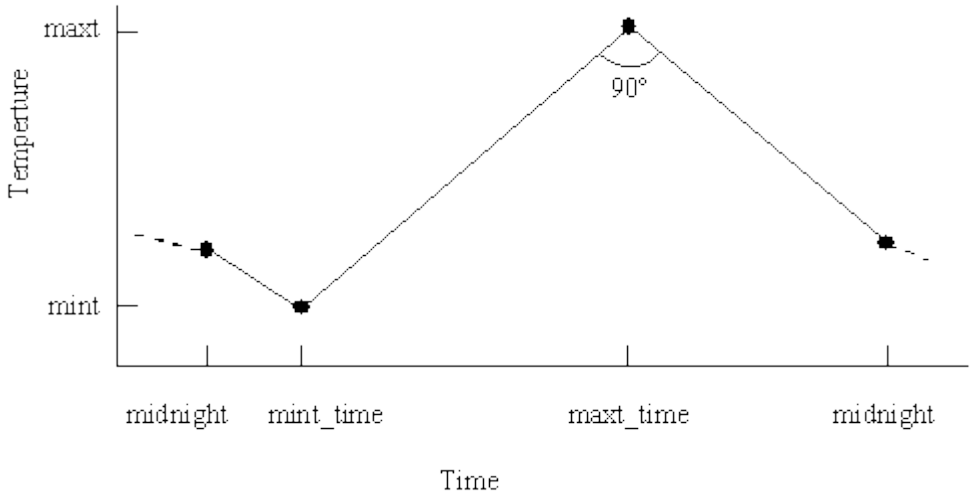


***Figure 2**. Diagram showing the interpolation of air temperature within a day based on minimum and maximum temperature.*

Air temperatures occurring between midnight and mint_time are linearly interpolated from the air temperature at midnight,
calculated at the end of the previous Apsim timestep, and mint.
There is a linear rise in temperture from the day's minimum to maximum.
After maxt_time until midnight, air temperature is calculated as decreasing at the same rate at which it rose.

If the Apsim timestep is less than 24 hours

it is assumed that the user is supplying enough detail of the diurnal **changes in air temperature** that such interpolation is not required.
In that case air temperture is taken as the average of mint and maxt for each Apsim timestep.

Initialisation

There are two sections required for initialisation of the Soiltemp module; constants and parameters.

Constants

Name	Unit	Description	Value
nu	-	forward/backward differencing	0.6
vol_spec_heat_om	J m^{-3} K^{-1}	volumetric specific heat of organic matter	5.00e6
vol_spec_heat_water	J m^{-3} K^{-1}	volumetric specific heat of water	4.18e6
vol_spec_heat_clay	J m^{-3} K^{-1}	volumetric specific heat of clay minerals	2.39e6

Parameters

Where a variable name is followed by "[nz]" the variable is an array and the appropriate number of values must be supplied.

Name	Unit	Description	Range
clay[nz]	-	proportion of clay	0.0 - 1.0
bound_layer_cond	J s^{-1} m^{-2} K^{-1}	boundary layer conductance	0.0 - 100.0

The higher the value of **bound_layer_cond** the greater the difference between air and soil surface temperature.
If its value is unknown, Campbell (1986) suggests that a value of 20 J s^{-1} m^{-2} K^{-1} is an appropriate initial estimate.

A further, optional, parameter is,

Name	Unit	Description	Range
soil_temp[nz]	^{o}C	initial soil temperature	-100.0 - 100.0

which used to initialise soil temperature.
If it is not supplied the soil temperature array is initialised to the average annual temperature.
Simulations will eventually 'forget' the effect of poor initial guesses of soil temperature, but this may take some time.
Testing of this module showed that it took approximately 40 days for the temperature at 1.5 m deep to converge to within 0.5 ^{o}C of the analytical solution when the initial temperature difference was 7 ^{o}C.
The discrepancy will be greatest deeper in the soil profile, and where *C* is high or l is low.
In general, where soil temperature is only important in the soil surface layers the convergence will occur within the first 10 days or so.

Where the time taken to 'forget' the initial conditions might cause significant error, there are two strategies for overcoming this problem.
The first is to run a dummy simulation prior to the start of the real simulation to estimate the starting soil temperature.
The second option is to estimate the initial soil temperatures from an analytical solution.
A solution for the heat flow equation assuming a sinusoidal upper boundary condition, which might for example be the annual cycle in air temperature, is (Carslaw and Jaeger, 1959),

$$T(z,t) = T_{ave} + T_{amp} \exp(-\frac{z}{z_d}) \sin(\omega t - \frac{z}{z_d})$$, [19]

in Equation 19. ,

$$z_d = \sqrt{\frac{2\lambda}{\omega C}}$$, [20]

and,
T_{ave} is the average annual temperature (^{o}C),
T_{amp} is the annual amplitude in temperature (^{o}C), and
w or Omega is the angular frequency (radians).
Thermal conductivity and heat capacity can be estimated, using soil profile averages, from equations 3, 5, and 6.

Time step inputs from other modules

Soiltemp must be accompanied by the input module and a soil water module in order that other inputs are supplied.

These inputs are:

Variables from the Met(Input) module

Name	Unit	Description
tav	^{o}C	Average annual temperature, used to set the initial soil temperature if soil_temp is not supplied. Also determines the temperature at the lower boundary.
mint	^{o}C	Minimum air temperature.
maxt	^{o}C	Maximum air temperature.

Variables from the Clock module

Name	Unit	Description
timestep	min	Simulation timestep, converted to seconds internally.

Variables from the soil water module

Where a variable name is followed by "[nz]" the variable is an array and the appropriate number of values must be supplied.

Name	Unit	Description
dlayer[nz]	mm	Array of layer depths used to specify the nodes, converted to m internally.
sw[nz]	$m^3 m^{-3}$	Volumetric soil water content.
bd[nz]	$Mg\ m^{-3}$	Soil bulk density.
eos	mm	Potential soil water evaporation, or the water-depth equivalent of the net radiation reaching the soil surface, converted to J s^{-1} m^{-2} K^{-1} internally.
es	mm	Actual soil water evaporation, or the water-depth equivalent of the evaporation from the soil surface, converted to J s^{-1} m^{-2} K^{-1} internally.

To allow compatibility with modules where changes in the soil occur with time, dlayer and bd are requested at every Apsim time step. If eos and es are not available they are assumed equal to zero.

Time step outputs

Where a variable name is followed by "[nz]" the variable is an array and the appropriate number of values will be outputted.

Name	Unit	Description
final_soil_temp_surface	^oC	Soil surface temperature at the end of the final internal Soiltemp timestep.
final_soil_temp[nz]	^oC	Soil temperature at the end of the final internal Soiltemp timestep.
Ave_soil_temp_surface	^oC	Average soil surface temperature during the Apsim timestep.
Ave_soil_temp[nz]	^oC	Average soil temperature during the Apsim timestep.
mint_soil_surface	^oC	Minimum soil surface temperature found in each layer during the Apsim timestep.

Name	Unit	Description
mint_soil[nz]	^{o}C	Minimum soil temperature found in each layer during the Apsim timestep.
maxt_soil_surface	^{o}C	Maximum soil surface temperature found in each layer during the Apsim timestep.
maxt_soil[nz]	^{o}C	Maximum soil temperature found in each layer during the Apsim timestep.
therm_cond[nz]	$J\ s^{-1}\ m^{-1}\ K^{-1}$	Thermal conductivity for each layer.
heat_store[nz]	$J\ m^{-3}\ K^{-1}$	Volumetric specific heat for each layer.

References

Campbell G.S., 1985, *Soil Physics with Basic* , Elsevier, Amsterdam.
Carnahan B., Luther H.A., Wilkes J.O., 1969, *Applied Numerical Methods* , 1 st Ed. Wiley, New York.
Carslaw H.S., Jaeger J.C., 1959, *Conduction of Heat in Solids* , Oxford University Press, London.
Jury W.A., Gardner W.R., Gardner W.H., 1991, *Soil Physics* , 5 th Ed. Wiley, New York.

SoilWat

Description

The SoilWater module is a cascading water balance model that owes much to its precursors in CERES (Jones and Kiniry, 1986) and PERFECT(Littleboy *et al*, 1992).

The algorithms for redistribution of water throughout the soil profile have been inherited from the CERES family of models.

The water characteristics of the soil are specified in terms of the lower limit (ll15), drained upper limit (dul) and saturated (sat) volumetric water contents.

Water movement is described using separate algorithms for saturated or unsaturated flow.
It is notable that redistribution of solutes, such as nitrate- and urea-N, is carried out in this module.

Modifications adopted from PERFECT include
(i) the effects of surface residues and crop cover on modifying runoff and reducing potential soil evaporation,
(ii) small rainfall events are lost as first stage evaporation rather than by the slower process of second stage evaporation, and
(iii) specification of the second stage evaporation coefficient (cona) as an input parameter, providing more flexibility for describing differences in long term soil drying due to soil texture and environmental effects.

The module is interfaced with the RESIDUE and crop modules so that simulation of the soil water balance responds to change in the status of surface residues and crop cover (via tillage, decomposition and crop growth).

Enhancements beyond CERES and PERFECT include

- the specification of swcon for each layer, being the proportion of soil water above dul that drains in one day
- isolation from the code of the coefficients determining diffusivity as a function of soil water (used in calculating unsaturated flow). Choice of diffusivity coefficients more appropriate for soil type have been found to improve model performance.
- Unsaturated flow is permitted to move water between adjacent soil layers until some nominated gradient in soil water content is achieved, thereby accounting for the effect of gravity on the fully drained soil water profile.

SoilWater is called by APSIM on a daily basis, and typical of such models the various processes are calculated consecutively
(This contrasts with models such as SWIM that solve simultaneously a set of differential equations that describe the flow processes).

Figure 1. The subroutine structure of SoilWater.

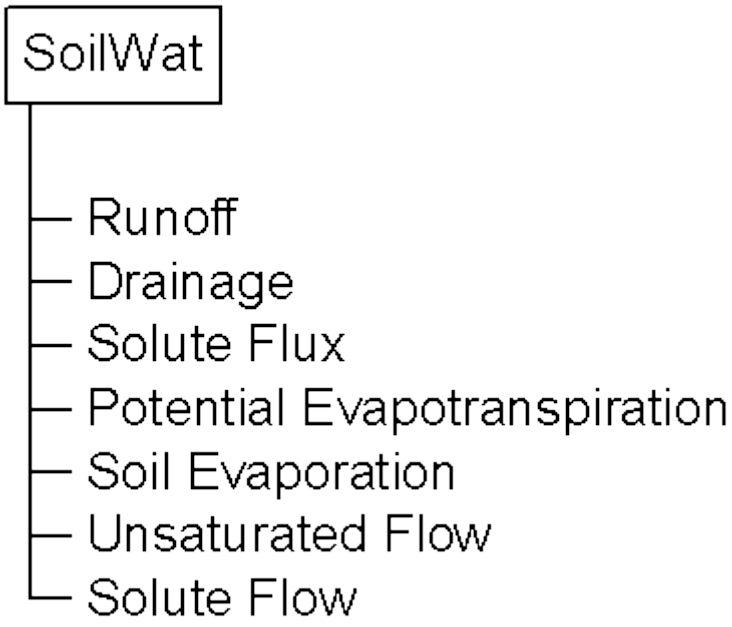
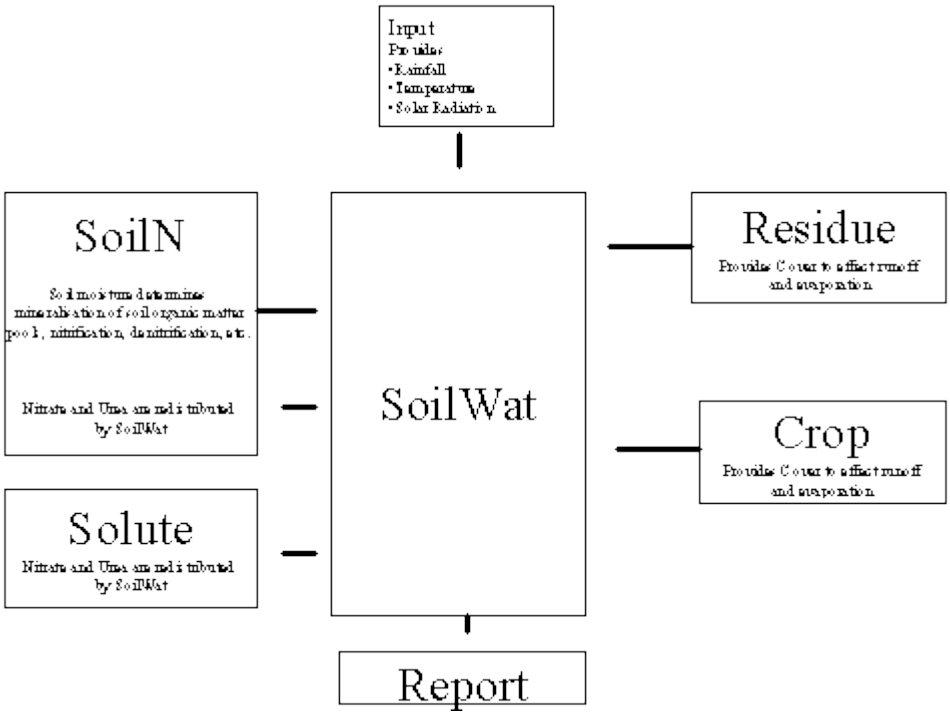
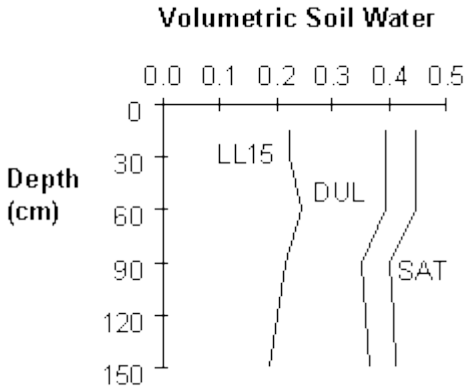


Figure 2. A diagram showing the communication between SoilWater and other APSIM modules.



Soil Moisture Properties

An example some soil properties used to configure the SoilWater soil water "bucket" is as follows.



- LL15 is the 15Bar lower limit of soil water content . It is approximately the driest water content achievable by plant extraction. This defines the “bottom of the bucket”.
- DUL is the drained upper limit of soil water content . It is the content of water retained after gravitational flow. DUL is sometimes referred to as “Field Capacity”.
- SAT is the Saturated water content . This defines the “top of the bucket”

Initial Soil Water

There are **five** ways to parameterise initial soil water in SoilWater.

These can also be specified in the Manager module sections using the ‘set' command
e.g

```
SoilWater set insoil = 0.5 ()
```

1. Initial Soil Water as User Specified Soil Water Content

The user can initialise soil water content to any initial volumetric soil water content for each layer.
This can be done by setting the soil water parameter (sw) for each layer to a value between SAT and LL15.
To use this option the **insoil parameter must be set to > 1.**

2. Initial Soil Water as a Fraction of Available Soil Water distributed evenly down the profile

Initial soil water can be set to a fraction of the maximum available soil water in each layer. Setting the insoil parameter to a fraction (0 = LL15, 1 = DUL) initialises the soil water parameter (initial sw values are ignored) to this fraction for each layer.

ie. If **0.0 <= Insoil <= 1.0**, then
in each layer
Soil water (sw) = LL15 + ((DUL-LL15) x Insoil)

3. Initial Soil Water as a Fraction of Available Soil Water filling the profile from the top.

Initial soil water can be set to a fraction of the maximum extractable soil water in the whole profile.
The profile is filled from the top down until the **fraction** is reached. Setting the profile_fesw parameter to a fraction of total extractable soil water initialises the soil water for layers starting at the top of the profile to DUL, until the **fraction** is reached.
The remaining layers are set to LL15. This parameter is exclusive of the others.

eg. If
the soil profile has layers of 150, 150, 300 and 300 mm,
giving a total profile depth of 900 mm
with corresponding DUL of 67.5, 67.5, 135, 120 mm and
corresponding LL15 of 34.5, 34.5, 72, 75 mm giving
ESW of 33, 33, 63, 45 mm with a
Total ESW in the profile of 174 mm.
And
profile_fesw = 0.5,
Then
The ESW of each layer is set to 33, 33, 21, 0 mm giving
A total of 87 mm in the profile.

4. Initial Soil Water as a depth of available Soil Water from the top of the profile.

Initial soil water can be set to a depth of available soil water in the whole profile.
The profile is filled from the top down until the **soilwater** depth is reached.
Setting the profile_esw_depth parameter to a total available soil water initialises the soil water for layers starting at the top of the profile to DUL, until the **amount of water** is reached.
The remaining layers are set to LL15. This parameter is exclusive of the others.

eg. Using the soil parameters of the previous example and
profile_esw_depth = 87mm,
Then
The ESW of each layer is set to 33, 33, 21, 0 mm.

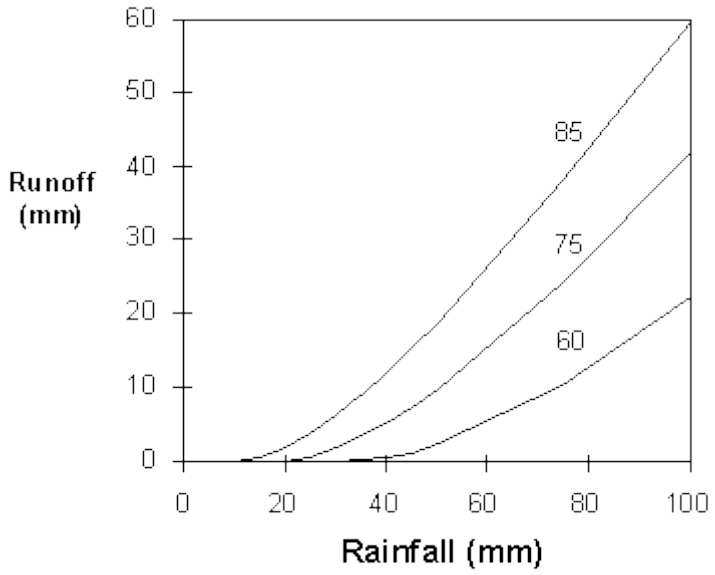
5. Initial Soil Water as a depth of wet soil, filled to field capacity.

Initial soil water can be set to a depth of wet soil.
The profile is filled from the top down until the **soil** depth is reached.
Setting the wet_soil_depth parameter to a depth of soil filled to field capacity (DUL) initialises the soil water for layers starting at the top of the profile to DUL, until the **soil depth** is reached.
The remaining layers are set to LL15. This parameter is exclusive of the others.

e.g. Using the soil parameters of the previous example and
wet_soil_depth = 400 mm,
Then
The ESW of each layer is set to 33, 33, 21, 0 mm giving
A total of 87 mm in the profile.

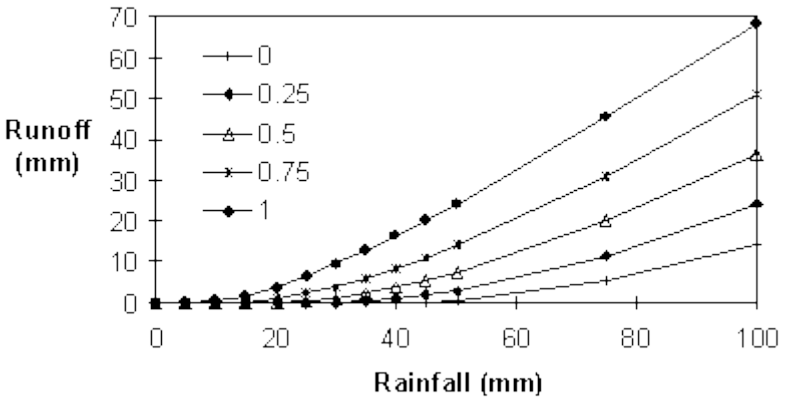
Runoff

Runoff from rainfall is calculated using the USDA-Soil Conservation Service procedure known as the curve number technique.
The procedure uses total precipitation from one or more storms occurring on a given day to estimate runoff.
The relation excludes duration of rainfall as an explicit variable, and so rainfall intensity is ignored.
When irrigation is applied it is assumed to be at low intensity and therefore no runoff is calculated.

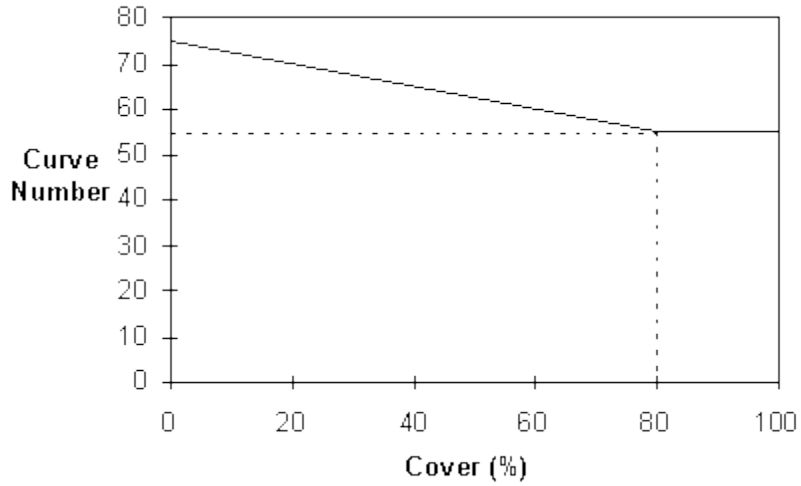


Runoff response curves (ie runoff as a function of total daily rainfall) are specified by numbers from 0 (no runoff) to 100 (all runoff).
Response curves for three runoff curve numbers for rainfall varying between 0 and 100 mm per day

The user supplies a curve number for average antecedent rainfall conditions (CN2Bare).
From this value the wet (high runoff potential) response curve and the dry (low runoff potential) response curve are calculated.
The SoilWater module will then use the family of curves between these two extremes for calculation of runoff depending on the daily moisture status of the soil.
The effect of soil moisture on runoff is confined to the effective hydraulic depth as specified in the module's ini file and is calculated to give extra weighting to layers closer to the soil surface.



Runoff response curve for average antecedent rainfall condition curve number (CN2) of 75 for a range of soil moisture conditions
(0 - dry, 1 - wet).



Residue cover effect on runoff curve number where bare soil curve number is 75 and total reduction in curve number is 20 at 80% cover.

Surface residues inhibit the transport of water across the soil surface during runoff events and so different families of response curves are used according to the amount of crop and residue cover.
The extent of the effect on runoff is specified by:
a threshold surface cover (CNCov), above which there is no effect,
and the corresponding curve number reduction (CNRed).

Tillage of the soil surface also reduces runoff potential, and a similar modification of Curve Number is used to represent this process.
A tillage event is directed to the module, specifying cn_red, the CN reduction, and cn_rain, the rainfall amount required to remove the tillage roughness.
CN2 is immediately reduced and increases linearly with cumulative rain, ie. roughness is smoothed out by rain.

The effect of these processes is seen in the daily output variable CN2_new - the curve number used to calculate runoff on any particular day.

Evaporation

Soil evaporation is assumed to take place in two stages: the constant and the falling rate stages.

In the **first stage** the soil is sufficiently wet for water to be transported to the surface at a rate at least equal to the potential evaporation rate.
Potential evapotranspiration is calculated using an equilibrium evaporation concept as modified by Priestly and Taylor(1972).

Once the water content of the soil has decreased below a threshold value the rate of supply from the soil will be less than potential evaporation (**second stage evaporation**).
These behaviors are described in SoilWater through the use of two parameters: **U** and **cona**.

The parameter **U** (as from CERES) represents the amount of cumulative evaporation before soil supply decreases below atmospheric demand.

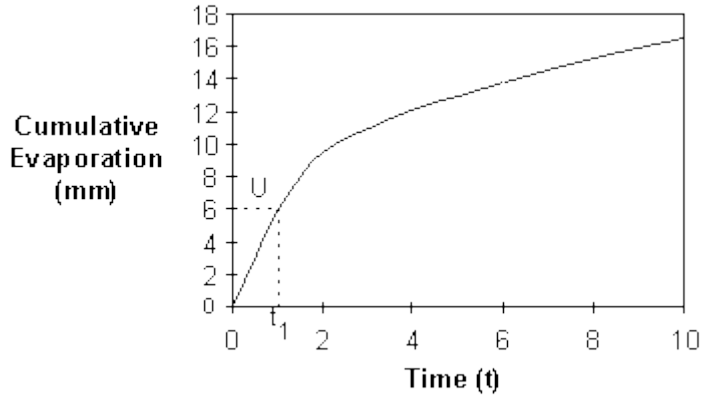
The rate of soil evaporation during the second stage is specified as a function of time since the end of first stage evaporation.

The parameter **Cona** (from PERFECT) specifies the change in cumulative second stage evaporation against the square root of time.

ie. $E_s = \text{cona } t^{1/2}$

Water lost by evaporation is removed from the surface layer of the soil profile thus this layer can dry below the wilting point or lower limit (LL) to a specified air-dry water content (air_dry).

SoilWater uses two parameters, **U** and **CONA**, to specify soil evaporation.
U(as in CERES) is the amount of cumulative evaporation, since soil wetting, before soil supply becomes limiting.
Subsequently soil evaporation is a fraction of the square root of time since the end of first stage evaporation, using the regression coefficient **CONA** (from PERFECT).



Cumulative Soil Evaporation through time for U = 6 mm and CONA = 3.5.

For $t <= t_1$
$E_s = E_{os}$

For $t > t_1$

$$\Sigma E_s = U.t + CONA.\sqrt{t - t_1}$$

Saturated Water Flow (between SAT and DUL)

When water content in any layer is below SAT but above DUL, a fraction of the water drains to the next deepest layer each day.

Flux = SWCON x (SW - DUL)

Infiltration or water movement into any layer that exceeds the saturation capacity of the layer automatically cascades to the next layer.

Unsaturated Water Flow (below DUL)

For water contents below DUL, movement depends upon the water content gradient between adjacent layers and the diffusivity,
which is a function of the average water contents of the two layers.

Unsaturated flow may occur both towards the surface and downwards, but cannot move water out of the bottom of the deepest layer in the profile.
Flow between adjacent layers ceases at a soil water gradient (gravity_gradient) specified in the SoilWater ini file.

The diffusivity is defined by two parameters set by the user (diffus_const, diffus_slope) in the SoilWater parameter set

(Default values, from CERES, are 88 and 35.4, but 40 and 16 have been found to be more appropriate for describing water movement in cracking clay soils).

Diffusivity = diffus_const x exp (diffus_slope x thet_av)

where,
thet_av is the average of SW - LL15 across the two layers.
Flow = Diffusivity x Volumetric Soil Water Gradient

Solute Movement

SoilWater can move any solute within the APSIM simulation.
Solutes are defined to be either mobile or immobile, according to specifications in the SoilWater module's ini file.

The saturated and unsaturated flows of soil water are used to calculate the redistribution of solutes throughout the soil using a “mixing” algorithm which assumes that all water and solute entering or leaving a layer is completely mixed.
This means that solute movement can simply be calculated as the product of the water flow and the solute concentration in that water.

Fluxes of solutes are associated with both saturated and unsaturated water fluxes.
In both cases a simple mixing algorithm is used whereby incoming water and solute is fully mixed with that already present in any layer to obtain concentrations for solutes that are applied to the water leaving the layer.

Efficiency factors (flux_eff and flow_eff) are specified in the SoilWater ini file to adjust the effectiveness of mixing for either saturated or unsaturated flows.

Leaching

The amount of solute dissolved in water can move to the layer below. This is considered by the model to be leaching.
Individual solutes leached from each layer can be tracked by reporting the solute leaving the layer.
An example may include reporting no3_leach(7).
If there are 7 layers defined in the soil profile then this variable will represent the leaching of no3 out of the root zone.

Solutes in Rainfall

Solutes can be applied to the soil via rainfall using the optional parameter 'rainfall_XXX_conc' in parts per million,
where 'XXX' is the solute name.

As many solutes as desired can be applied in this way, however any solutes used must first be initialised in the system with the SOLUTE module.

Above Saturation Flow (above SAT)

When the soil water rises above Saturation there are two different parameters that can be specified: MWCON or KS.
Both of these are layered values, so you need to enter a value for each layer in the soil.

MWCON allows you to specify either 1 or 0 for each layer in the soil.

A value of 1 means that all water above saturation for that layer immediately flows straight into the layer below.
If this layer below is filled above saturation by water coming in from above and MWCON for that layer is 1 then this water then flows to the next layer and so forth for each of the layers below.
Eventually water will run out of the bottom layer as drainage (report "drain" to see this).
This is the tipping or cascading part of the classical tipping bucket or cascading flow analogy.

A value of 0 means that all water above saturation is NOT allowed to flow into the layer below. ie it is an impermeable layer.
Water will then begin to backup.
Note that depending on how SWCON is set, water will still be able to be lost from the layer via the process of Saturated Flow (the flow that occurs between DUL and SAT).
If both MWCON and SWCON prevent the water from flowing out of the layer then the water will begin to back up and will either become a pond on the surface or a water table beneath the surface.

KS allows you to specify **a number of millimeters per day** that is allowed to drain from the layer when the the soil water is above saturation.
Once again depending upon how SWCON is set, Saturated Flow may still occur, and once again if soil water backs up a pond or water table is generated.

nb. If <u>neither</u> MWCON or KS is set, then an MWCON value of 1 for each layer is assumed. ie. any water above saturation flows straight into the layer below.
If <u>both</u> MWCON and KS are set, then KS overrides MWCON.

Surface Ponding and Water Table Functionality

Soilwat has the functionality of simulating an impermeable layer(s) within the soil profile, above which a water table may perch.
The impermeable layer(s) is specified by setting either MWCON or KS (as mentioned in the previous section).

If the water-table happens to reach the surface, then Soilwat also has the ability to simulate a surface-storage (or ‘ponding') pool.
The surface-water storage capacity is specified by the parameter **max_pond**, the maximum available surface water storage.
This **max_pond** is applied to both runoff from rainfall events and also to runoff generated by water backing-up above an impermeable layer (mxcon or ks) below.
Any surface water backing-up over ‘max_pond' is added to the runoff pool.

There is also an output value called "pond_evap" which tells you the evaporation from the pond. This is separate from es which does NOT include pond evaporation in it.

MWCON, KS and max_pond are all optional parameters and if not provided, they are set to defaults (0.0 for max_pond).

Here is an example of setting up a pond:

```
[black_earth.soilwat2.parameters]
!layer 1 2 3 4 5 6 7
dlayer = 150 150 300 300 300 300 300        ! layer thickness mm soil
bd  = 1.30 1.30 1.29 1.31 1.35 1.36 1.36    ! bulk density gm dry soil/cc
moist soil
sat = .500 .509 .510 .505 .490 .480 .480    ! saturation mm water/mm soil
dul = .450 .459 .45 .44 .42 .41 .41         ! drained upper limit mm water/mm
soil
sw = .280 .364 .43 .43 .40 .41 .41          ! initial soil water mm water/mm
soil
ll15 = .230 .240 .240 .250 .260 .270 .280   ! lower limit mm water/mm soil
air_dry = .10 .20 .20 .20 .20 .20 .20       ! air dry mm water/ mm soil
swcon = 0.02 0.02 0.02 0.02 0.02 0.02 0.02  ! drainage coefficient
mwcon = 1.0 0.0 1.0 1.0 1.0 1.0 1.0
max_pond = 15.0     ! maximum depth of surface storage
```

nb. The two lines,

```
mwcon = 1.0 0.0 1.0 1.0 1.0 1.0 1.0
max_pond = 15.0     ! maximum depth of surface storage
```

Note, in layer 2 of mwcon, mwcon is set to 0.0 **not** 1.0 like all the other layers and therefore this layer is impermeable to ‘cascading' flow, hence water cascading down the soil profile will reach this layer and begin to back-up towards the surface. Drainage is still allowed through this layer unless ‘swcon' is also set to zero.

If the impermeable layer is specified deep down in the soil, then the water table might not reach the surface and pond. The output variable ‘water_table' reports the depth of the water-table below the surface as measured from the ground surface. If there is no water table, ‘water_table' = 10000 by default.

NB. There is a Pond module in APSIM. This Pond module calculates all the solute and nitrogen effects of having a pond above your soil. It grows algae etc and modfies the chemical processes going on in the soil (The processes that normally occur in the Soil Nitrogen module SoilN). The ponding we are discussing here are just the parts that effect the water balance and hence are dealt with in this Soil Water module.

Runon, Lateral Inflow and Outflow on a layer basis

Runon Capability

Surface 'runon' can now be input from a data file.
Soilwat2 does a daily 'get' to retrieve this information from the input module.
From a user's point of view this means that 'runon' can be provided on a daily basis from the met file or from a separate input file.

For example :

```
[runon.sparse.data]

runon = 0
allow_sparse_data = true
 year  day  runon
 ()    ()   (mm)
 1988  1    20
 1988  5    17
 1988  6    22
 1988  10   25
 1988  11   22
```

Lateral Inflow Capability

Lateral Inflow can be input as a layer-based array via the same method as 'runon'. The parameter name is 'inflow_lat(layer)'.

For example:

```
[runon.sparse.data]
runon = 0
allow_sparse_data = true
year  day runon  inflow_lat(1)  inflow_lat(2) inflow_lat(3)
 ()    ()  (mm)  (mm)           (mm)          (mm)
 1988  1   20     1             2             2
 1988  5   17     2             1             2
 1988  6   22     0             0             0
 1988  10  25     1             1             1
 1988  11  22     1             2             2
```

Both 'runon' and 'inflow_lat(layer)' are optional inputs to the model.

NB. with Lateral Inflow it is assumed that ALL the water goes straight into the layer. Irrespective of the layers ability to hold it. It is like an irrigation. Klat has no effect and does not alter the amount of water coming into the layer. Klat only alters the amount of water flowing out of the layer

Lateral Outflow Capability

Lateral Outflow is the flow that occurs as a result of the soil water going above DUL and the soil being on a slope. So if there is no slope and the water goes above DUL there is no lateral outflow. KLAT is just the lateral resistance of the soil to this flow. It is a soil water conductivity.

The calculation of lateral outflow on a layer basis is now performed using the equation:

Lateral flow for a layer = Klat * d * s/(1+s^2)^0.5 * L/A * unit conversions.

Where:
Klat = lateral conductivity (mm/day)

d = depth of saturation in the layer (mm)= dlayer*(sw-dul)/(sat-dul) if sw > dul.
Note this allows lateral flow in any "saturated" layer, not just those inside a water table.

s = slope (m/m)
L= catchment discharge width. Basically, it's the width of the downslope boundary of the catchment. (m)
A = catchment area. (m2)

An example of the new soilwat2 input parameters (optional - if not included, lateral outflow = zero) are :

```
[black_earth.soilwat2.parameters]
slope =0.1
discharge_width = 50
catchment_area = 1000
klat = 0.5
```

There is a new reportable variable called outflow_lat(layer). This can be summed by reporting outflow_lat().

NB. Lateral outflow only occurs when the soil water for the layer is above Drained Upper Limit (DUL). The above calculation involving klat, slope, discharge width, and catchment area is used to calculate the Lateral Outflow from each layer.

SoilWater Module Outputs

Name	Units	Description
Es	mm	Daily soil evaporation
Eo	mm	Daily potential evapotranspiration
Eos	mm	Daily potential soil evaporation
cn2_new		Daily value CN2 adjusted for surface cover
Runoff	mm	Daily runoff
Drain	mm	Daily drainage from bottom of soil profile
infiltration	mm	Daily infiltration across the soil surface
eff_rain	mm	Effective Rainfall (Rainfall - Runoff - Drainage).
salb		Bare soil albedo
bd	g/cm^3	Bulk density of the soil for each layer
esw	mm	Extractable soil water in each layer (ie. sw_dep - ll15_dep)
sw_dep	mm	Amount of water in each layer
sw	mm^3/mm^3	Volumetric water content in each layer
dlayer	mm	Thickness of each soil layer
ll15_dep	mm	Amount of water corresponding to a soil potential of 15 bar
ll15	mm^3/mm^3	Volumetric water content for each layer corresponding to a soil potential of 15 bar
dul_dep	mm	Amount of water at drained upper limit for each soil layer
dul	mm^3/mm^3	Volumetric water content at drained upper limit for each soil layer
sat_dep	mm	Amount of water in each layer at saturation
sat	mm^3/mm^3	Volumetric water content at saturation for each soil layer
air_dry_dep	mm	Amount of water retained at air_dry for each layer
air_dry	mm^3/mm^3	Volumetric water content for air dry soil in each layer
flux	mm	Saturated water flux from each layer to the layer below
flow	mm	Unsaturated water movement between layers (+ve up)
nnnn_leach	kg/ha	Amount of solute nnnn in saturated water movement from each layer to the layer below (where 'nnnn' is the name of the solute)(eg. no3_leach)

Name	Units	Description
nnnn_up	kg/ha	Amount of solute nnnn in unsaturated water movement for each layer (+ve up). (where 'nnnn' is the name of the solute)(eg. no3_up)
water_table	mm	Depth of the water table (10000 if no water table present)
pond	mm	Surface ponding
outflow_lat	mm	Lateral outflow from each layer

SoilWater module actions

Reset

The reset action can be invoked to reset the module to the state specified within the module's input data, which includes the soil moisture characteristics, runoff and evaporation parameters and the initial soil water profile.

The Reset action is identical to the initialise action used by the simulation engine at the start of the simulation except that a description of the reinitialised state is not printed in the simulation summary file.

APSIM Manager Example:

```
[sample.manager.start_of_day]

! reinitialise residues at the beginning of each sowing window
If day = 100 then
   SoilWater reset
endif
```

Initialise

The initialise action has now been replaced by the reset action (see above).

Summary Report

At initialisation, at series of tables and useful information is printed to the simulation summary file for perusal by the user. These tables can be printed to the summary file at any point during the simulation as a detailed record of the system state at a particular time.

APSIM Manager Example:

```
[sample.manager.start_of_day]


! Print out a summary of module state to the summary file
If day = 100 then
   SoilWater sum_report
endif
```

References

1. Jones, C.A., and J.R. Kiniry. 1986. CERES-Maize: A simulation model of maize growth and development. Texas A&M University Press, College Station, Texas.
2. Littleboy, M., D.M. Silburn, D.M. Freebairn, D.R. Woodruff, G.L. Hammer, and J.K. Leslie. 1992. Impact of soil erosion on production in cropping systems. I. Development and validation of a simulation model. Aust. J. Soil Res. 30, 757-774.
3. Priestly, C.H.B., and Taylor, R.J. (1972) On the assessment of surface heat and evaporation using large-scale parameters. *Monthly Weather Review*. **100**, 81
4. Ritchie, J.T. (1972) Model for predicting evaporation from a row crop with incomplete cover. *Water Resources Research.* **8** , 1204.
5. Soil Conservation Service (1972) National Engineering Handbook Section 4: Hydrology, Soil Conservation Service, USDA, Washington.

Solute

Description

The Solute module is an APSIM plug-in-pull-out module that keeps a solute balance of up to five different solutes.
The Solute module itself does not change the state of the solutes, but lets other modules change the solute information according to their own processes.
For example, a solute may have its state changed by a soil water balance module as a result of its leaching calculations.

Things to note when using the Solute module

The Solute module can keep track of Nitrate and Ammonium, but must do so independent of any other soil balance module (e.g. SoilN2).
The Solute module is designed primarily to trace solutes such as Bromide and Chloride through soil layers.

Solute module outputs

The outputs for this module flow on from the input parameters.
The module will provide the amount of solute in kg/ha on a layered basis for each of the specified solutes.

For the sample illustrated above, the possible outputs would be "br" or "cl".
Additional outputs are:

1. ‘max_xxxx’ (the maximum amount of solute xxxx in any layer (kg/ha)),
2. ‘maxppm_xxxx’ (the maximum amount of solute xxxx in any layer (ppm)), and
3. ‘maxlayer_xxxx’ (the soil layer number corresponding to the maximum solute load for solute xxxx’).

For Chlorine in the above example, these outputs would be

1. max_cl,
2. maxppm_cl, and
3. maxlayer_cl.

Solute Diffusion

Whilst the mass flow or advection of solutes is provided by the equations within the soil water modules, solute diffusion can be calculated within the solute module by supplying a diffusivity coefficient for an individual solute as follows.

```
[sample.solute.parameters]


solute_names = cl

cl =   0 0 0 0 0 0 0 0 (kg/ha) ! Initial Cl profile

d0_cl = 108
```

The solute flux is calculated as

Flux = D0 * θ*τ*dc/dx

Where D0 is the diffusivity in free water, θ is water content, τ is the soil water pore space tortuosity (=θ2/ θs2) and dc/dx is the concentration gradient within the soil water solution.

Surface

What does the SURFACE module do?

The Surface module is a small plug-in-pull-out APSim module that communications with the APSwim module to simulate changes in surface seal conductance through time
(See the APSwim documentation for more information regarding surface seals within the SWIM water balance model).

The Surface module is able to communicate with APSwim upon each internal SWIM time step to update surface conductance during high intensity rainfall events where SWIM time steps are of short duration.

Essentially, the SURFACE calculations "overwrite" the internal SWIM calculations of surface seal just prior to SWIM's attempt at solving its equations for the time step.

Things to note when using the SURFACE module.

- APSwim must be parameterised to use the option for a surface conductance function for the top boundary condition.
- There are 2 surface seal models available within SURFACE.
    - The first model replicates the internal surface seal model of SWIM and this option can be used to regression test the operation of the SURFACE module.
    - The second model is based upon the work of Silburn & Connolly (Journal of Hydrology , 172 (1995) 87-104.)

SurfaceOM

Description

The processes included in the SurfaceOM module are depicted in Figure 1.

Briefly, the above ground material can be burnt (or removed from the system in some other way, e.g. baling), incorporated into the soil during tillage operations, or decomposed.

Above ground residues are considered as consisting of a mixture of one or more different materials (or component parts), each of which is defined in terms of:

- Mass (kg/ha)
- Overall C:N ratio ()
- Overall C:P ratio ()
- Standing Fraction (0-1)
- Type (eg wheat, lucerne, eucalyptus leaves etc) – from which SURFACEOM will determine the following information:
    - Overall Carbon fraction (0-1)
    - Specific Area (ha/kg)
    - Potential Decomposition Rate (/day)
    - Mineral Composition (nh4, no3,and po4 (in ppm))
    - C,N,P fractions in each of the fresh organic matter (FOM) pools (i.e. carbohydrate, cellulose, lignin)

SURFACEOM module outputs can either refer to the entire mixture of surface materials, or to specific components.

When new material is added (e.g. at harvest), the material will either enter an existing surface organic matter component (eg wheat) or may start a new component, if that residue is a new addition to what is already present in the system (for example, wheat trash being added to existing lucerne residues, at harvest).

Each component is kept separate for calculations of C:N ratio, decomposition, and specific area.

An overall effective cover value (0-1) is calculated using all surface organic matter components present, for the purposes of subsequently calculating surface material effect on soil evaporation and runoff.

If a particular surface organic matter type has soluble inorganic N components (NO3-N and NH4-N), these may be transferred to the respective soil pools by leaching due to rainfall or irrigation.

The cumulative amount of rain and irrigation to transfer all of the soluble components is specified as an input; the amount leached is proportional to cumulative rain.

It is possible to specify that a certain proportion of any particular surface organic matter is 'standing', that is, inert from the processes of decomposition.
The default value for this proportion is zero, in other words, all the material is decomposable.
See section below on 'Standing/Lying Components'

Tillage results in a transfer of some surface OM into the soil FOM pool.
With tillage, surfaceOM N and C is incorporated into soil layers to the nominated tillage depth, and added to the respective soil mineral N and the fresh organic matter pools (FOM).
Incorporated surfaceOM C and N is partitioned into the rapid, medium and slowly decomposing FOM pools according to nominated fractions.
This fractionation is dependant upon the type of OM and so differences between crop residues and animal manures, for example, can be specified.

Decomposition results in loss of some carbon as CO_2 and transfer of carbon and nitrogen to the soil.
Decomposition of residues with a high C:N ratio creates an immobilisation demand, which is satisfied from mineral-N in the uppermost soil layers;
in extreme situations, inadequate mineral-N in soil restricts decomposition of residues.

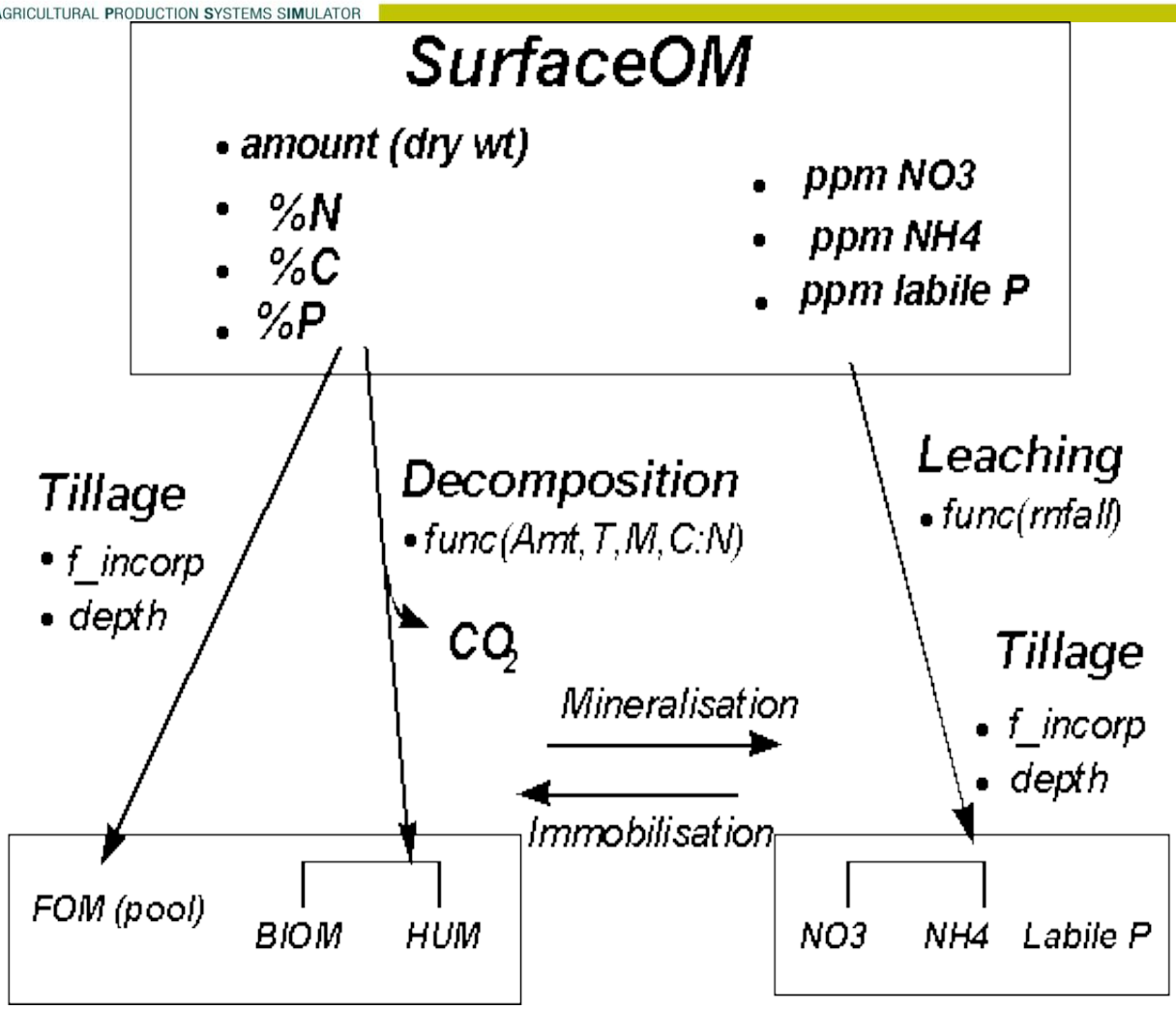


Figure 1. Schematic representation of the processes in the SURFACEOM module. (Note that default values of zero are used for mineral N and P components in plant residues)

Surface Organic Material Decomposition

Decomposition of surface OM's is calculated using a simple exponential decay algorithm where the fraction of each component decaying on any day (F_{decomp}) is calculated as follows:

F_{decomp} = Potential Decomposition Rate x **Moisture Factor** x **Temperature Factor** x **C:N ratio Factor** x **Contact Factor**

From this fraction, potential amounts of carbon and nitrogen to move into the soil are calculated for each component.
Any module responsible for soil organic matter pools (such as the SOILN module) can use this potential supply of carbon and nitrogen in its calculations.
The actual value of decomposition (that is a final soil limited value) for each component is passed back to the SURFACEOM module and above-ground component pools are updated using this value of decomposition.

The moisture factor for decomposition

The moisture factor affecting decomposition in the SURFACEOM module uses cumulative potential soil evaporation (E_{OS}) to capture the effect that dry residues decompose more slowly than wet residues.

It is assumes that residues dry out at a rate proportional to E_{os}, and that a critical cumulative evaporation (cum_eos_max) results in residues becoming so dry that decomposition ceases.

Moisture Factor = 1.0 - SE_{OS}/cum_eos_max

The factor is constrained to values between 0 and 1.

The temperature factor for decomposition

The effect of temperature on residue decomposition is described by:

Temperature Factor = (average air temperature / opt_temp)^2

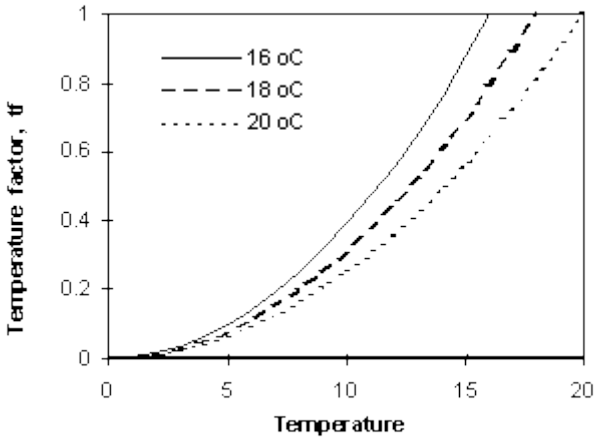
where: average air temperature = (maxt + mint) / 2

This factor is then constrained to values between 0 and 1.

The resultant relationship is shown in the following figure for three values of optimum temperature.

(tf = 20 is the default).

If average temperature is less than zero, the temperature factor is zero.



The C:N ratio factor for decomposition

$$C:N\ \ factor = \exp \frac{-k.(CN - CN_{opt})}{CN_{opt}}$$

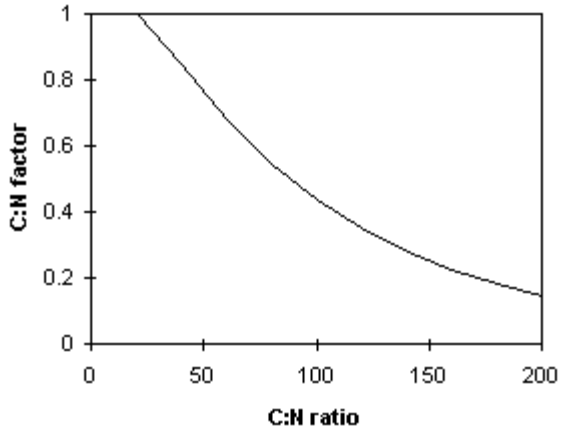
where

- CN = C:N ratio of surface residues
- CN_{opt} = Optimum C:N ratio for decomposition
- k = coefficient determining slope of curve.

This factor is calculated for individual residue types rather than for the entire mixture and is constrained to values between 0 and 1.

The standard values used with SURFACEOM are k = 0.277 and CN_{opt} = 25.

The resultant curve is as follows:

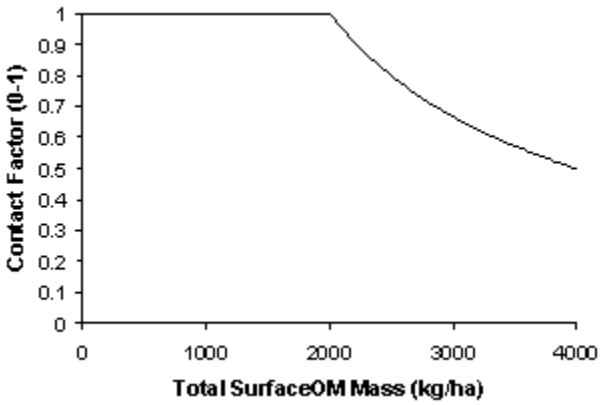


The soil contact factor on decomposition

Where large amounts of surface residues are present, overall rates of decomposition will be lower.
It is presumed that the material in immediate contact with the soil decomposes more rapidly than that piled on top (a "haystack" effect).

To account for this a **Contact Factor** discounts decomposition according to the amount of residue.
The relation currently used is based on work by Thorburn et al.(2001) , and involves the concept of a critical mass of surface organic matter at which the "haystack effect" is deemed to become relevant.
It may be summarized as:

- If surfaceom_wt < critical mass then cf = 1.0
  or
- If surfaceom_wt > critical mass then cf = critical mass/surfaceom_wt



Not all surface residues contribute to this "haystack effect".
Standing residues are excluded, and some residues on the soil surface can be specified to only contribute partially to the effect.
For example, course woody debris would not contribute but fine leaf litter would.

Tillage of above-ground residues

Residues are incorporated into the soil profile using a tillage command.
For example, the following manager command will incorporate 50% of residues into the top 100mm of soil.
(Note: the following information needs to be specified on a single line in APSIM manager logic)

```
surfaceom tillage type = ''user_defined'', f_incorp = 0.5 (0-1),
tillage_depth = 100. (mm)
```

Alternatively, the user can specify tillage operations in terms of an implement, with the corresponding values of f_incorp and tillage_depth being defined in the SURFACEOM module's constants file.
Some examples are:

```
surfaceom tillage type = disc ()
```

or

```
surfaceom tillage type = burn ()
```

In the "burn" example, the tillage type is specified to incorporate to a zero depth, and the fraction of organic matter specified to be incorporated will be lost from the system.

Each time a "tillage" is specified, all surface organic materials are effected.

If the user is wanting to simulate a situation where only one of the surface materials is incorporated (eg. Digging manure into row spaces between existing crop stubbles) leaving the remainder intact, there is a specific command available called "tillage_single".
The syntax is as follows:

```
surfaceom tillage_single name = manure , type = ''disc''()
```

The "name"refers to the specific name in the system of the surfaceom to be tilled.
Subsequent arguments are as for the standard "tillage" command (see above).

SurfaceOM Cover

SurfaceOM cover is calculated by combining the individual masses of surface OM types and their specific areas (i.e. an area of cover per unit mass)
Currently, both "standing" and "lying" fractions are considered to contribute to cover.

However, increasing amounts of surface OM have diminishing effects due to the additional residue overlaying other residues rather than covering bare soil.
This can be described as:

$dC/dS = 1 - C$ (1)

where
C is the effective fractional residue cover, and
S is the total surface area of residues per unit area

and hence

$C = 1 - e^{S}$ (2)

SurfaceOM Input Parameters

Name	Units	Description
name	-	Individual residue name
type	-	Specific residue type (referenced to .ini file)
mass	kg/ha	Initial amount of surface OM
cnr	-	Initial C:N ratio of surface OM
cpr (optional)*	-	Initial C:P ratio of surface OM
standing_fraction (optional)*	(0-1)	Standing or inert component of surface material

* - if optional parameters are not supplied then defaults from constants file are used.

All SurfaceOM module input parameters are arrays, hence several surface OM types can be initially specified as existing in the simulation.
For example:

```
[all.surfaceom.parameters]
name = wheat_old wheat_new lucerne  ! name of surface OM material
type = wheat     wheat     lucerne  ! type of surface OM material
mass = 350.0     2000.0    400.0    ! mass of surface OM (kg/ha)
cnr =  100.0     75.0      35.0 ()  ! C:N ratio of surface OM
```

If desired, only a single surface OM may be specified (in similar fashion to the old APSIM RESIDUE2 module), or up to a maximum of 100 materials.

The "name" of the surfaceom is a unique identifier of that surfaceom pool.
The "type" is the specific type of material, used to reference further information for that type.

In other words, there can be several surfaceom's of the same "type" in the system, but only one instance of each "name" is allowed.
See the above example, where residues from an old wheat crop are present together with new residues from a more recent wheat crop.
They are both of the same "type", but are considered as separate pools, each with an individual "name", *wheat_old* and *wheat_new*.

When a crop discards leaves/stems etc, they are added to the SURFACEOM module in a pool with the "name" and "type" equal to the crop name, i.e. wheat, chickpea.

SurfaceOM Standing/Lying Components

For individual surface materials, it is possible to specify how much of that material is "standing" using an optional input parameter called "standing_fraction".

If it is not supplied, the default value of 0.0 is applied.

The standing fraction is considered to be the inert component of the material, i.e. not subject to daily decomposition.

It is however, subject to incorporation during tillage events.

Currently there is no algorithm describing the movement of material from the "standing" pool to the "lying" (decomposable) pool, however in future it is planned to convert standing material to lying material on a daily basis as a function of such parameters as time, rainfall, stocking rates, field operations. etc.

Information on the "standing" component can be supplied to the SurfaceOM module as follows:

```
[all.surfaceom.parameters]
name = wheat_old wheat_new ! name of surface OM material
type = wheat     wheat     ! type of surface OM
mass = 350.0     2000.0    ! mass of surface OM (kg/ha)
cnr  = 100.0     75.0      ! C:N ratio of surface OM
standing_fraction = 0.0        0.4      ! standing (or inert) fraction
```

SurfaceOM Phosphorus

It is possible to trace the addition and decomposition of phosphorus through surface residues in conjunction with the APSIM SoilP module.

The user is required to provide an extra parameter to state the initial C:P ratio for surface residues.

```
[all.surfaceom.parameters]
name = maize          ! name of surface OM material
type = maize          ! type of surface OM material
mass = 2000 (kg/ha)   ! mass of surface OM material
cnr  = 75.0 ()        ! C:N ratio surface OM material
cpr  = 250.0 ()       ! C:P ratio surface OM material
```

The addition of this extra parameter will result in the SurfaceOM module becoming aware of the need for maintaining a phosphorus balance, and daily interactions with the SoilP module will occur.

(Note, if the "cpr" parameter is provided, the SoilP module must be included in the simulation).

In the above example, 2000 kg/ha of maize residue is added with a C:P ratio of 250.
This will result in 2000kg x 0.4 (kg C/kg biomass)/250 (kg C/kg P) = 3.2 kg P/ha in surface OM.

Resetting SurfaceOM

The reset action can be invoked to reset the module to the state specified within the module's input data, which includes the surfaceom weight, nitrogen and phosphorus contents, and cover.

This is identical to the initialise action used by the simulation engine at the start of the simulation except that a description of the reinitialised state is not printed in the simulation summary file.

APSIM Manager Example:

```
[sample.manager.start_of_day]

! reinitialise residues at the beginning of each sowing window

If day = 100 then
    surfaceom reset
endif
```

Summary Report

At initialisation, at series of tables and useful information is printed to the simulation summary file for perusal by the user.

These tables can also be printed to the summary file at any instance during the simulation as a detailed record of the system state at a particular time.

For the surfaceom module, this output consists of surfaceom names, types, weights, organic carbon, nitrogen, and phosphorus, mineral components, and standing fraction.

APSIM Manager Example:

```
[sample.manager.start_of_day]

! Print out a summary of module state to the summary file

If day = 100 then
      surfaceom sum_report
endif
```

Addition of Surface Organic Materials

Organic materials can be added to the soil surface using the add_surfaceom action.

APSIM Manager Example:

NOTE: actions must be specified as a single line in the manager file rather than as shown below.

```
If day = 100 then
      surfaceom add_surfaceom name = wheat, type = wheat, mass = 1000.
(kg/ha), n = 5 (kg/ha)
endif
```

or

to specify nitrogen content using a C:N ratio

```
If day = 100 then
      surfaceom add_surfaceom name = wheat, type = wheat, mass = 1000.
(kg/ha), cnr = 80 ()
endif
```

If the simulation is to contain a phosphorus balance then the phosphorus content of the added residue must also be added.
For the examples above the user would add the extra information as follows:

```
If day = 100 then
      surfaceom add_surfaceom name = wheat,type = wheat,mass = 1000.
(kg/ha),n = 5 (kg/ha),p = 2 (kg/ha)
endif
```

or

to specify phosphorus content using a C:P ratio

```
If day = 100 then
      surfaceom add_surfaceom name = wheat,type = wheat,mass = 1000.
(kg/ha),n = 5 (kg/ha),cpr = 200 (kg/ha)
endif
```

An example of how a source of manure (named FYM which will have its C content specified in the surfaceOM.ini file)
would be applied at 10 t ha^{-1} and incorporated using the C:N and C:P ratios to specify the N and P contents is:

```
If day = 100 then
       surfaceom add_surfaceom name = fym1,type = fym,mass = 10000.
(kg/ha),cnr = 15 (),cpr = 30 ()
       surfaceom tillage_single name = fym1, type = hoe ()
endif
```

Note that the "tillage_single" action will only incorporate the surface OM specified by "name" as fym1.

Guidelines on characterization of manures are provided in Appendix 1.

Use of variables in user-defined manager commands

APSIM Manager Example:

```
if surfaceom.surfaceom_wt > 4.0 then

    remove_amount = surfaceom.surfaceom_wt - 4.0

    remove_fraction = remove_amount/surfaceom.surfaceom_wt

    surfaceom tillage type=user_defined, f_incorp = remove_fraction ,
tillage_depth=0.0

endif
```

SurfaceOM module outputs

TOTAL Surface OM

Name	Units	Description
surfaceom_wt	kg/ha	Total mass of all surface organic materials
surfaceom_c	kg/ha	Total mass of organic carbon
surfaceom_n	kg/ha	Total mass of organic nitrogen
surfaceom_p	kg/ha	Total mass of organic nitrogen
surfaceom_no3	kg/ha	Total mass of nitrate
surfaceom_nh4	kg/ha	Total mass of ammonium
surfaceom_labile_p	kg/ha	Total mass of labile phosphorus
surfaceom_cover	0-1	Fraction of ground covered by all surface OM's
tf	0-1	Temperature factor for decomposition
wf	0-1	Water factor for decomposition
cf	0-1	Contact factor for decomposition

INDIVIDUAL Surface Om's

Name	Units	Description
surfaceom_wt_*xxxx*	kg/ha	Mass of the SurfaceOM named "*xxxx*" *
surfaceom_c_*xxxx*	kg/ha	Mass of organic carbon in "*xxxx*"
surfaceom_n_*xxxx*	kg/ha	Mass of organic nitrogen in "*xxxx*"
surfaceom_p_*xxxx*	kg/ha	Mass of organic nitrogen in "*xxxx*"
surfaceom_no3_*xxxx*	kg/ha	Mass of nitrate in "*xxxx*"
surfaceom_nh4_*xxxx*	kg/ha	Mass of ammonium in "*xxxx*"
surfaceom_labile_p_*xxxx*	kg/ha	Mass of labile phosphorus in "*xxxx*"
surfaceom_c1_*xxxx*	kg/ha	Mass of organic carbon in "*xxxx*"
surfaceom_c2_*xxxx*	kg/ha	Mass of organic carbon in "*xxxx*" in fpool2
surfaceom_c3_*xxxx*	kg/ha	Mass of organic carbon in "*xxxx*" in fpool3
surfaceom_n1_*xxxx*	kg/ha	Mass of organic nitrogen in "*xxxx*" in fpool1
surfaceom_n2_*xxxx*	kg/ha	Mass of organic nitrogen in "*xxxx*" in fpool2
surfaceom_n3_*xxxx*	kg/ha	Mass of organic nitrogen in "*xxxx*" in fpool3
surfaceom_p1_*xxxx*	kg/ha	Mass of organic phosphorus in "*xxxx*" in fpool1
surfaceom_p2_*xxxx*	kg/ha	Mass of organic phosphorus in "*xxxx*" in fpool2
surfaceom_p3_*xxxx*	kg/ha<	Mass of organic phosphorus in "*xxxx*" in fpool3
pot_c_decomp_*xxxx*	kg/ha	Potential organic C decomposition in "*xxxx*"
pot_n_decomp_*xxxx*	kg/ha	Potential organic N decomposition in "*xxxx*"
pot_p_decomp_*xxxx*	kg/ha	Potential organic P decomposition in "*xxxx*"
standing_fraction_*xxxx*	0-1	Fraction of "*xxxx*" which is inert, ie not in contact with the ground
surfaceom_cover_*xxxx*	0-1	Fraction of ground covered by "*xxxx*"
cnrf_*xxxx*	0-1	C:Nratio factor for decomposition for "*xxxx*"

* - *"xxxx"* is the "name" of an individual surface organic material,
for example "wheat" , "lucerne", "ox_manure" etc.

References

Probert M.E., Dimes J.P., Keating B.A., Dalal R.C., Strong W.M. *APSIM's water and nitrogen modules and simulation of the dynamics of water and nitrogen in fallow systems,* Agric. Syst. 56 (1998), pp 1-28.

Thorburn P.J., Probert M.E., Robertson F.A. *Modelling decomposition of sugar cane surface residues with APSIM-Residue,* Field Crops Research 70 (2001), pp 223-232.

APPENDIX

Guidelines for characterizing manure

In APSIM the organic forms of C, N and P in OM additions are distributed between three pools. Upon incorporation these pools are added to the corresponding FPOOLs (i.e. carbohydrate, cellulose, lignin) that comprise FOM (fresh organic matter) in the SoilN module.

To date all efforts to simulate the effects of manure have been for situations where the manure has been fully incorporated soon after application.

The minimum data required to specify a manure source are the same as those required for any other OM source:
a name (to specify a particular batch of manure),
a type (to distinguish the type of manure), its composition (C, N and P) and
the allocation of C, N and P to the three pools.

The fraction of C in the manure and the allocation of C, N and P between the pools is stipulated for the particular manure type in the SurfaceOM.ini file
(as in the example below)
but the contents of N and P are input as part of the manager command when manure is added to the system (either as amounts (kg/ha) or as C:N and C:P ratios)
as shown in the previous sections of this document.

```
[standard.surfaceom.fym]
fom_type = fym
fraction_C = 0.30
! The fraction of carbon in HQM (0-1)
pot_decomp_rate = 0.01 ! Decomposition rate (day-1) for manure on soil surface
fr_c = 0.1 0.5 0.4! The fractional allocation of carbon to each of the three
pools

fr_n = 0.1 0.5 0.4 ! The fractional allocation of nitrogen to each of the
three pools
fr_p = 0.1 0.5 0.4 ! The fractional allocation of phosphorus to each of the
three pools
po4ppm = 10.0      ! labile P concentration (ppm)
nh4ppm = 100.0     ! ammonium-N concentration (ppm)
no3ppm = 10.0      ! nitrate-N concentration (ppm)
specific_area = 0.0001   ! specific area (ha/kg)
cf_contrib =  1
```

In this example,
the allocation of C, N and P to the three pools is identical
and so the C:N and C:P ratios of all three pools will be equal to those based on the total C, N and P concentrations.

Modelling short-term dynamics

Using data from laboratory incubation studies, it has been shown that the assumption of the same allocation of C and N across all three pools is not consistent with the observed pattern of mineralization for a range of manures (Probert et al. 2005).

The initial period of immobilization of N (in the first few weeks) and the period before the system showed net mineralization could be simulated only if the FPOOLs had different C:N ratios.

Probert et al. (2005) used information from proximate analysis of the manures to distribute C and N between the pools.
They identified FPOOL1 with the soluble C and N components, FPOOL3 with the lignin carbon (ADL), and FPOOL3 by difference.

An example:

```
fr_c = 0.10 0.70 0.20  ! The fractional allocation of carbon to each of the
three pools

fr_n = 0.04 0.86 0.10  ! The fractional allocation of nitrogen to each of the
three pools
```

In this example there is relatively less N than C in pool 1 so that this pool will have a wider C:N ratio than the whole manure.
Because pool 1 is the fraction that decomposes fastest, there will be greater initial immobilization than if C:N ratio was the same in all three pools.

Modelling long-term multi-season dynamics

To date, modelling exercises have not been done to investigate whether distribution of C, N and P between the pools affects the effectiveness of manures as sources of nutrients for crops in the longer–term.
Experimental data to explore such effects are also lacking.

Simulation of short-term effects suggest that the consequence of different allocation of C and N between the pools diminishes over time
so that the longer-term effects can be expected to be dependent more on the overall C:N ratio than the C:N ratios of the different pools.

In modelling field studies of the response by crops to inputs of manure (Dimes and Revanuru 2004),
the quality aspect has been limited to varying the distribution of OM between the three pools, with the C:N being the same in each pool.

Lower quality manures are assumed to have a greater proportion of C in FPOOL3 thereby releasing its nutrients more slowly.
The values used to simulate the high (HQM) and low (LQM) quality manures were as shown in the Table.

Table. Values used by Dimes and Revanuru (2004) to simulate high and low quality manures.

Quality	fract_C	cnr	Allocation to pools
HQM	0.16	22	fr_c = 0.0 0.20
			fr_n = 0.0 0.20
LQM	0.25	35	fr_c = 0.0 0.01
			fr_n = 0.0 0.01

- HQM - High Quality Manure
- LQM - Low Quality Manure

References

Dimes, J.P. and Revanuru, S. (2004).
Evaluation of APSIM to simulate plant growth response to applications of organic and inorganic N and P on an Alfisol and Vertisol in India.
In "Modelling Nutrient Management in Tropical Cropping Systems" (eds R.J. Delve and M.E. Probert) pp 118-125. ACIAR Proceedings No. 114. (ACIAR: Canberra).

Probert, M.E., Delve, R.J., Kimani, S.K. and Dimes, J.P. (2005).
Modelling nitrogen mineralization from manures: representing quality aspects by varying C:N ratios of sub-pools.
Soil Biology & Biochemistry ***37***, *279-287*.

Introduction

1.1 What is APSIM SWIM?

APSIM SWIM is the result of adapting SWIM Version 2.0 to communicate with APSIM.

SWIM (Soil Water Infiltration and Movement) is a soil water and nutrient balance model written by P. J. Ross of C.S.I.R.O. Division of Land and Water.

APSIM (Agricultural Production Systems Simulator) is a cropping systems modelling environment specially designed to allow a plug-in-pull-out approach for the integration of various simulation models.

It is a product of the Agricultural Production Systems Research Unit (APSRU).

APSIM SWIM is designed to run within APSIM and calculate all flows of water and nutrients through, in, and out, of soil for a given simulation.
These flows include infiltration, runoff, plant uptakes, movement through soil, etc, and the related nutrient flows.

1.2 How does SWIM operate within APSIM?

To simultaneously solve the water flow equations SWIM increments its way through time using time steps small enough to allow the solution of its equations within given tolerance levels.

APSIM uses a fixed time step, usually one day in duration.

For these two approaches to exist together SWIM is forced to perform its equations for a time frame within the current APSIM time step.

As APSIM cycles through all the active modules,
allowing each to perform its own time step processes,
SWIM will be allowed, during its own process step,
to (including infiltration, runoff, drainage, and crop water uptake) and all solute flows (including solutes in infiltration, uptake of all solutes by all crops in the system).
Transformational flows of solutes, such as with nitrogen, will take place on a daily time step in the module that owns that variable, such as the soil nitrogen balance.

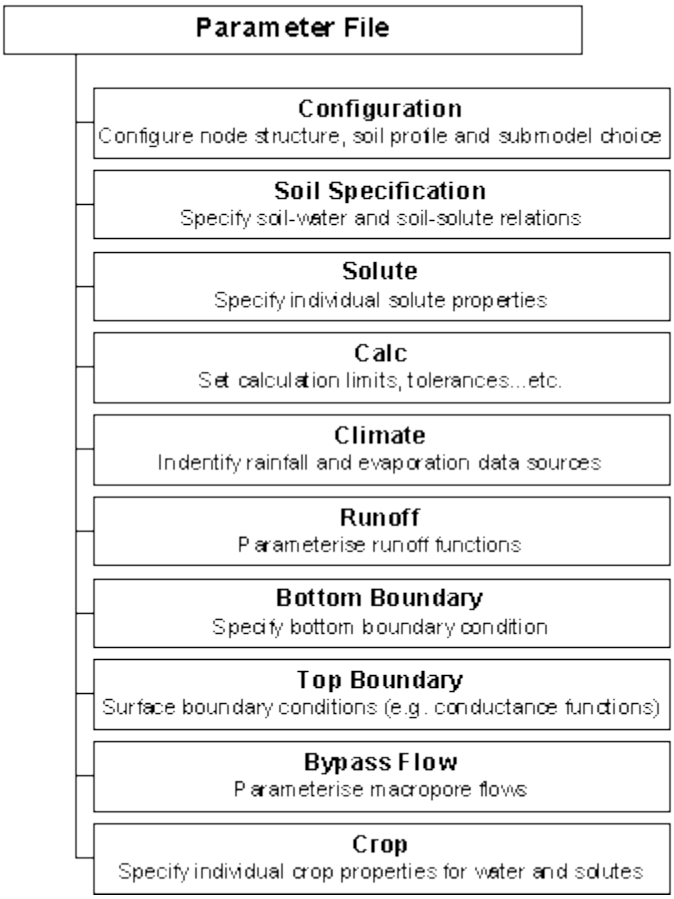
2 Initialisation

APSIM SWIM uses the standard APSIM input file capabilities and structure.

Please refer to the APSIM documentation for specifics on the formatting of APSIM input files.

2.1 The APSIM SWIM Parameter File

The structure of an APSIM SWIM data set is shown on the following page.
There are a series of predefined sections containing related data.
The structure shown here could be reproduced many times within one file as each data set lies within the one logical grouping of data sections.



2.2 Configuration Section

The main section of the APSIM SWIM parameter file(s) is the configuration section. Contained in this are all configuration switches, for specifying all SWIM sub-models, and descriptions of the soil profile required for model initialisation.

```
[sample.apswim.init]
! -------------------- initial layer information ----------------------

           !-------------------------------------------------------------
           !     1     2     3     4     5     6     7     8     9
           !-------------------------------------------------------------
        x =    0.0  50.0 100.0 300.0 500.0 750.0 1000.  1600  2000  (mm)
soil_type =  soil1     -     -     -     -     -     -     - soil2
      psi =  -1500 -1500 -1500 -1500 -1500 -1000 -1000 -1000  -500 (cm)

           !-------------------------------------------------------------

slmin = -3.0 ()                    ! slmin and slmax are the minimum and
maximum
slmax =  7.0 ()                    ! log suction values used for specifying
                                   ! moisture characteristic and hydraulic
                                   ! conductivity curves in the soil
descriptions
bypass_flow =on (on/off)           ! Bypass Flow  - On/Off
                                   !
runoff      = 2 ()                 ! Runoff Flag  - 0) No ponding (all runoff)
                                   ! -----------    1) All ponded (no runoff)
                                   !                2) Use runoff functions
                                   !
top_boundary_condition  = 2 ()     ! Surface Flag - 0) infinite surface
conductance
                                   ! ------------   1) constant potential
                                   !                2) conductance function
                                   !
bottom_boundary_condition = 0 ()   ! Bottom Flag  - 0) constant gradient
                                   ! -----------    1) water table
                                   !                2) zero flux
                                   !                3) free drainage (seepage)
vapour_conductivity = off (on/off) ! Vapour Conductivity flag (0=off, 1=on)
                                   ! ------------------------

run_solutes = br cl no3            ! List of solutes SWIM is required to move.

solute_exclusion = on (on/off)     ! switch to specify if solute in soil
water is
                                   ! taken in excess of the crop solute
demand.
extra_solute_supply = on (on/off)  ! switch to specify if extra solute, above
that
                                   ! received in the extracted soil water, is
to taken
                                   ! up using a simple empirical method)
echo_directives = on (on/off)      ! choice to echo the receipt of directives
such
                                   ! as irrigation or tillage.
```

The SWIM module contains many components and sub-models that can be specified to behave in different ways.
Examples of this may be the equations for runoff or behaviour at the top and bottom soil boundaries.
Switches for these are contained in the main model configuration section above and the in-line documentation in the example above explains the meaning of each one.

When configuring APSIM SWIM one is required to specify all solutes to be redistributed within the APSIM simulation.
To do this we specify, in this section, the names of all solutes that are to be moved (referred to above as “run_solutes”).
APSIM SWIM will then find out the states of each of these pools and communicate with their owner modules.
Some modules may potentially own solutes that will play no part within the simulation.
The user is informed during initialisation, via the summary file, of any solutes owned by individual modules that are not included in the list of "run_solutes".
The user can decide if any of these should also be added to the list of solutes to be parameterised and used by APSIM SWIM.
The user can alternatively choose to have no "run_solutes" by setting the parameter to "none".

The soil profile specification (at the beginning of the section - though order is not important) describes the nodal structure for the current simulation and the soil type at each of these nodes.
Each soil type is referred to by a unique user-defined name.
The user can also specify a gradual change in soil type from one node to another by not defining nodes between them.
A single dash (“-”) is used for this purpose.
The above example will linearly interpolate, according to depth, the soil for each node so that the soil properties will gradually change from soil type one at the surface to soil type two at the bottom boundary.
The actual properties of each soil type are described in a soil specification section for each soil type.
The name for each section is the same as the unique name given in the soil profile description above.

The parameters extra_solute_supply and solute_exclusion allow the user to specify the response to situations where total uptake of solute does not match the solute demand of the crop.
Extra solute can be excluded or taken up empirically, depending on the situation.
This functionality is not part of the original SWIMV2 model but has been included to allow flexibility in responses to crop modules depending upon APSwim for uptake of solutes.
The specification of these parameters is optional and default to “off”.

The last parameter, echo_directives, is used to verify the communications between APSwim and other modules within APSIM.
If this value is set to on, all tillage and application of surface water messages will be reported to the screen and to the simulation summary file.

2.3 Soil Type Description Sections

For SWIM to calculate all water and solute movements it needs to know certain soil-water and soil-solute relations.

This information is given as follows:

```
! ----------------------soil type information -------------------------
[sample.apswim.soil1]
sl    = -
3.000000  0.400000  1.000000  1.386233  1.538325  4.355526   7.000000 ()
wc    =  0.255000  0.254882  0.253123  0.243884  0.232800  0.078951   0.028610
 (cc/cc)
wcd   = -0.000000 -0.000545 -0.008644 -0.051190 -0.089358 -0.030304  -
0.010982 ()
hkl   = -0.551294 -0.554319 -0.599416 -0.841595 -1.144555 -8.187840 -14.799289
()
hkld = -0.000000 -0.013938 -0.222431 -1.367128 -2.500100 -2.500100  -
2.500100 ()

 bulk_density = 1.0 (g/cc)
                   ! ---------------------------
                   solute_name =   cl  br  no3            ()
                  !---------------------------
                          exco =   0    0    0            ()
                           fip =   1    1    1            ()
                           dis =   0    0    0            ()
                  !---------------------------
```

sl is log suction where suction has units of cm.

For each value of sl there are corresponding values of volumetric water content (wc).
These values define a series of points on the moisture characteristic curve for this soil type.
SWIM uses these points to interpolate all values on this characteristic curve using piece-wise cubic approximations.
To enable this we must also supply the slope of the moisture characteristic curve at each of these points as well (wcd).
wcd is the derivative of the log suction vs water content curve at this point.
A similar approach is used for the specification of the hydraulic conductivity curve.
Here log hydraulic conductivity (hkl) (conductivity in units of cm/h) is supplied as well as the corresponding slope (hkld).
These soil characteristic curves can be specified using the HYPROPS program.

Solute-Soil interactions are specified for each solute in a separate table.
For each solute SWIM requires:-

1. **exco** - freundlich exchange isotherm coefficient
2. **fip** - freundlich exchange isotherm power term such that adsorbed concentration = **exco** xsolute_conc$^{\mathbf{fip}}$
3. **dis** - is the dispersivity of the solute in the soil.

Bulk Density is to be expressed as g soil per cubic centimetre.

2.4 Solute Specific Information Section

There are some solute parameters required by SWIM that are not related to soil type but are only specific to a particular solute.

```
! ---------------------- solute information --------------------------
[sample.apswim.solute]

               ! ----------------------------
                 solute_name =   cl   br  no3
               ! ----------------------------
                       slupf =    0    0    1
                        slos = 0.001 0.001 0.001
                          d0 =    0  .072   0
                           a =    0    1    0
                        dthc =    0    0    0
                        dthp =    1    1    1
                        disp =    1    1    1
          ground_water_conc =    0    0    0 (ppm)
               ! ----------------------------
```

- **slos** is defined as the osmotic pressure per unit solute concentration. (NOTE: All solute concentrations are expressed as ppm. ie. µg solute per g water)
- **d0** is the diffusion coefficient.

Tortuosity is calculated with SWIM as:
a. (theta - **dthc**)$^{\mathbf{dthp}}$
where theta is volumetric soil water content.

Hydrodynamic dispersion is:
dispersivity. |velocity|$^{\mathbf{disp}}$
where dispersivity is the value of **dis** given for each soil type.

Ground water concentrations are specified for use during simulation whenever the bottom boundary condition is set to allow water entry via the bottom of the profile.

2.5 APSIM SWIM Calculation Parameters

```
! ------------------ swim calculation parameters ---------------------
[sample.apswim.calc]
dtmin = 0.0               (min)   ! min time step
dtmax = 1440.             (min)   ! max time step
dtmax_sol = 60            (min)   ! max time step during solute uptake
max_water_increment=1.    (mm)    ! max water increment

slcerr = 0.000001         ()      !
ersoil= 0.000001          ()      !
ernode= 0.000001          ()      !
errex = 0.01              ()      !

dppl  = 2                 ()      !
dpnl  = 1                 ()      !

swt = 0.0                 ()      ! Space Weighting Factor (gravity flow)
                                  ! ------------------------------------
                                  ! 0.5 -> 1.0 (central to fully upstream)
                                  ! < -1 (central diffs by factor of -1*SWF)

slswt  = 0.0              ()
```

The only point of note for this section that is not covered in the SWIMV2 documentation is the parameter dtmax_sol.
This parameter was not part of the original SWIMV2 but works in tandem with the crop-solute interaction parameter, solute_exclusion (included in the "init" section).
This maximum timestep specification determines the maximum timestep allowable when the solute_exclusion flag is set to true and there is a non-zero crop solute demand for the current APSwim timestep.
Specifying a small maximum timestep forces APSwim to work slowly through time whilst solute demand is not satisfied, thus minimising the magnitude of any over-supply of solute as calculated via the flow of solute with soil water extraction.
Once the supply of solute is met this extra constraint is removed and the normal maximum timestep (dtmax) is used.
These two timestep constraints allow for differing levels of precision depending upon the processes taking place.

2.6 Climatic Inputs Section

At present there are three climatic inputs - soil albedo and the rainfall and potential evapotranspiration data sources.
If the user specifies either source as "file" they will then need to provide a file name (see example below).
If these files are to be found in a directory other than the current directory then a full file name including path will be required.

The user can also direct apswim to get rainfall and evaporation information from other modules within APSIM, such as the input module,
by simply typing in the name of the variable to use for that data (e.g. 'rain' or 'pan' from met module).

The user can also direct apswim to calculate evaporation data from daily meteorological data. This is achieved by using the keyword “calc” for the evaporation data source.

```
! ------------------------ climatic inputs ---------------------------
[sample.climate]
salb     = 0.23                          ()
rainfall_source = rain                   ()  ! use apsim
variable called 'rain'
evap_source = calc                       ()  ! calculate own potential
evaporation rate
```

or

```
[sample.climate]
salb     = 0.23                          ()
rainfall_source = file                   ()  ! get rainfall
from a file
rainfall_file = c:\work\myrain.dat
evap_source = file                       ()
evap_file = c:\work\myevap.dat
```

2.7 Runoff Functions Section

As explained in the configuration section information earlier the runoff sub-model can be configured as follows:-

```
runoff      = 2          ()     ! Runoff Flag  - 0) No ponding (all runoff)
                                ! -----------    1) All ponded (no runoff)
                                !                2) Use runoff functions
```

If the user chooses option number two then runoff functions are to be used for the calculation of runoff.
In this case, extra information needs to be supplied to APSIM SWIM and this is to be placed in the runoff section.

```
! ------------------------ runoff functions -------------------------
[sample.apswim.runoff]
maximum_surface_storage  = 20 ! (mm)
minimum_surface_storage  = 10 ! (mm)
initial_surface_storage  = 15 ! (mm)
precipitation_constant   = 50 ! (mm)
runoff_rate_factor       = .2 ! (mm/h)/mm^P
runoff_rate_power        =  2 ! =P ______/
```

Runoff occurs when the surface water depth is greater than the surface storage.
For a water depth that is dH above the storage, the runoff rate is equal to

Runoff Rate = Runoff Rate Factor . dH^P

To allow for a reduction of surface roughness due to rainfall,
the storage decreases exponentially with precipitation energy from the given initial value towards the given minimum.
The equation for the exponential decrease in storage is of the form

$$S = S_{min} + (S_{initial} - S_{min}).\exp(-E/E_{spc})$$

where
E is cumulative rainfall energy and E_{spc} is energy in an amount of rain equal to the given storage **precipitation constant** falling at 25mm/h.

The user can specify the initial surface storage available to rainfall as any value between the maximum and minimum surface storages.

2.8 Bottom Boundary Conditions Section

There is only one parameter required for one of the possible settings for bottom boundary conditions.
If the user chooses to set a constant gradient for the bottom boundary then it is input as follows.

```
! ------------------- bottom boundary conditions --------------------
[sample.bottom_boundary]
constant_gradient    = 0
```

or

```
constant_potential   = 0
```

If the user chooses zero flux then no inputs are required from this section.

If a the bottom boundary is to be a user defined potential (eg water table is said to exist at the bottom boundary) then constant potential will need to be supplied in this section.

If the user wishes to enforce a constant gradient at the base of the profile then this parameter needs to be specified.

If the user specifies free drainage at the bottom boundary then a value for a constant potential is required to describe the flow at the bottom of the profile.

2.9 Top Boundary Conditions Section

The soil may have a thin surface layer that impedes water entry.

The water flux through this layer is equal to the surface conductance multiplied by the matric potential difference across this layer.

A soil layer of thickness dx and saturated hydraulic conductivity K_s would represent a conductance of K_s/dx at saturation.

To allow for a reduction of surface conductance due to formation of a crust caused by rainfall, the conductance decreases exponentially with cumulative precipitation energy from the given initial value towards the given minimum.

```
! --------------------- top boundary conditions ---------------------
[sample.apswim.top_boundary]
maximum_conductance    = 4.0  (/h)  ! initial soil surface conductance
minimum_conductance    = .02  (/h)  ! minimum soil surface conductance
initial_conductance    = 1.0  (/h)  ! initial soil surface conductance
precipitation_constant = 2.5  (mm)  ! used to define rate of surface sealing
```

The equation for the exponential decrease in conductance is of the form

$G = G_{min} + (G_{initial} - G_{min}).\exp(-E/E_{cpc})$

where,
E is cumulative rainfall energy and E_{cpc} is energy in an amount of rain equal to the given conductance **precipitation constant** falling at 25mm/h.

2.10 Bypass Flow Section

If bypass flow has been enabled, within the initialisation section, the following parameters need to be specified.

```
! -------------------------- bypass flow --------------------------
[sample.apswim.bypass_flow]
depth       = 4              ! (node number)
conductance = .10            ! (/h)
storage     = .10            ! (cm water/cm of +ve Psi)
```

Depth is, of course, the macropore depth within the soil profile. It is expressed in terms of node number.

2.11 Crop Parameters Section

```
[sample.apswim.crop]

! ------------------------------------------------------
               crop_name =   wheat   barley    maize
! ------------------------------------------------------
     min_xylem_potential = -15000.  -15000.  -15000.(cm)
     root_radius         =   0.25     0.25   0.25   (mm)
     root_conductance    =  .4d-7   1.4d-7   1.4d-7 (cm3/h)
! ------------------------------------------------------
```

At present the only crop specifications required from the user are the minimum xylem potential, root radius and root conductance for particular crops.
Other day-to-day values such as soil water demands and root distribution are provided by APSIM during the simulation run.

2.12 Rainfall and Potential Evaporation Data

As chosen by the user(see climate data specification section above),
APSwim allows two methods of supplying specified rainfall and potential evaporation data.

The first method utilises the APSIM inter-module communications.
In this simple approach, the data for rainfall or evaporation is described as one homogeneous event.
An amount, a starting time and a duration or intensity is obtained via a general request of all modules for this information just prior to processing the timestep calculations.
This information is generally described in the APSIM weather file as follows.
The values returned to APSwim have been highlighted.
In this example rainfall will always fall at an intensity of 15 mm/h starting at 3:00 p.m. and 5 mm of potential evaporation is said to lie between 6:00 a.m. and 6:00 p.m. each day.

Alteratively the rainfall could be said to alway fall between 3:00 p.m. and 4:00 p.m. by using the alternative suggested below.

```
[user-defined..nput.weather]
site = Gatton
latitude = -27.0  (degrees)


rain_time = 15:00 (hh:mm)
rain_int = 15.0 (mm/h)            !(or  rain_durn = 60 (min))


eo = 5   (mm)
eo_durn =  720  (min)
eo_time = 6:00  (hh:mm)

    year   day      radn    maxt    mint    rain
     ()     ()       (MJ)    (oC)    (oC)    (mm)
    1995     1      20.0    25.0    15.0      5
    1995     2      20.0    25.0    15.0      0
    1995     3      20.0    25.0    15.0     10
    1995     4      20.0    25.0    15.0      0
```

The second method uses a simple data log file.
At the present stage of development APSIM SWIM uses a fixed format input file structure for rainfall and potential evaporation data.
Later releases will contain more flexible and more powerful mechanisms for the input of this data.
The input format for rainfall is as follows:-

```
 1991 121 00:00  5.   6
 1991 122 00:00 20. 120
 1991 122  2:00 10. 600
 1991 129 14:00 25. 600
```

The columns, from left to right, contain

- year (4 digit specification),
- day of year,
- time (24 hour notation),
- amount of rainfall, and the time duration* for this record.

For example the first line records an event at 12:00 am on the 121st day of 1991 in which 5 mm fell in 6 minutes.
The last line records an event in which 25mm fell at 2:00pm on 129th day of 1991 and lasted for 10 hours.
Evaporation data follow the exact same format.

* Time duration has highest resolution of 1 min.

2.13 Irrigation Data

There are two ways of accessing the one mechanism for application of irrigation.
APSIM SWIM responds to a low level "add_water" message.
The format of this is:-

```
apswim add_water amount=20(mm), time=12:00(hh:mm), duration=60 (min),
                                intensity = 20 (mm/h), no3=10.0(kg/ha)
```

(Note: implementation would require text on a single line)

This message tells SWIM that 20 mm of irrgation is to be added at midday today.
The irrigation takes 60 minutes to complete and ten kg/ha of nitrate was in that irrigation water.
SWIM will then incorporate all this information into its cumulative water and solute addition with time curves.
Any combination of amount, duration and intensity information will suffice as long as an overall amount and duration can be calculated from the information given.
The APSIM system's manager or operations manager modules will allow the user to set up various irrigation schedules using this message format.

However, the APSIM system also contains an irrigation module that has added features such as automatic irrigation rules and tabular input format.
It is highly recommended that users apply irrigation using the IRRIGATE module. This module will convert the irrigation information into the necessary format for APSIM SWIM.
See documentation of these modules for further assistance.

3 The effects of cover upon the water balance

3.1 Potential Soil Evaporation

The algorithm for cover effects on potential soil water evaporation is taken from that derived for the soilwat2 module.

$$E = Ep \times \exp(-canopy\_coeff \times Cover_{canopy}) \times (1 - Cover_{residue})^{residue\_coeff}$$

Where Ep is total potential evapotransipiration
Canopy_coeff is a coefficient for the effect of canopy cover on potential soil water evaporation
Residue_coeff is a coefficient for the effect of residue cover on potential soil water evaporation

3.2 Effective Rainfall Energy

Surface residues are said to protect the soil from rainfall energy by shielding the soil surface from rainfall impact according to the level of residue cover.

$$r_{KE} = \left(1 + Eff \ln\left(\frac{I}{Ir}\right)\right) \times (1 - Cover_{residue})$$

Where
$r_{KE}$is a measure of the energy in an amount of precipitation
if intensity I compared with that of the same amount of precipitation of intensity I_r.
Eff is an efficiency parameter
$Cover_{residue}$ is a 0 to 1 measure of surface residue cover.

4 Water and Solute flows for bottom boundary conditions in SWIM

The following describes the flow of water and solutes across the bottom boundary for the four boundary conditions supported by the SWIM model.
The relevant flow equations are as follows

Richards' Equation¶

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x} K\left(\frac{\partial \psi}{\partial x} + \frac{\partial z}{\partial x}\right) + S$$

- See equation 4 in Swimv2.1 manual

or

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial x}.Conductivity\left(matricgradient + gravity\right) + Sink$$

Advection-dispersion equation

$$\frac{\partial(\theta c)}{\partial t} + \frac{\partial(\rho s)}{\partial t} = \frac{\partial}{\partial x}\left(\theta D \frac{\partial c}{\partial x}\right) - \frac{\partial(qc)}{\partial x} + \phi$$

- See equation 50 in Swimv2.1 manual

or

$$\frac{\partial(liquid\ phase)}{\partial t} + \frac{\partial(adsorbed\ phase)}{\partial t} = \frac{\partial}{\partial x}(Dispersion) - \frac{\partial}{\partial x}(Advection) + Sink$$

See the SWIMv2.1 User manual for more information and definitions of terms.

We shall now look how the elements of the flow equations above are calculated for the four different bottom boundary conditions.

4.1 Specified matric potential gradient

Water Flow

The flow equations for water in the profile are solved with the matric potential gradient term, shown in the Richards equation above, set to the user specified value at the lowest node in the profile.
The actual value of the matric potential (ψ) may vary in time but that same gradient will apply until it is changed by the user.

- If the matric potential gradient is set to zero then drainage will proceed simply due to gravity ($\delta z/\delta x$ which is usually 1). This is often referred to as a unit (hydraulic) gradient. Flow can only proceed downward, at the rate of the hydraulic conductivity.
- If the matric potential gradient is set to -gravity (ie -1) the net results is one of zero flux.
- If matric gradient is negative (and greater than gravity,ie <-1) then water will pushed up into the profile. Note that, unless the user changes the gradient, the bottom boundary potential will continue to rise indefinitely until ultimately upward flow was offset by runoff!
- If matric gradient is positive then water will be drawn down out of the profile. Once again, unless the user alters the gradient, the potential of the lowest node will move toward extremely dry conditions.

Solute Flow

The solute flux calculations for the deepest soil node differ slightly to that for the rest of the profile.
As knowledge of space effectively ceases past this node, swim makes no assumptions regarding diffusion or dispersion.
As can be seen in the equations above, dispersion calculations would require some assumptions regarding the spatial gradient of solute concentration.
As a result, only convection is calculated for the bottom boundary.

The solute concentration in the water flowing across the bottom boundary in either direction is dependant on the direction of water flow.
Thus:

- If the water flow is downward, solutes will progress down across the deepest node. Once solute has passed the node it cannot return to the profile. Bulges may move across this node, but as we calculate convection only for this node the concentration gradient, or shape of the bulge, is not considered at this node.
- If the water flow is upward then it is assumed that the solute concentration in the water entering the profile is the same as the specified ground water concentration. This will act as a supply of solute into the profile in the same manner as the user-specified potential boundary condition below.

4.2 User-Specified potential

Water Flow

The flow equations for water are solved with the potential (hydraulic head = sum of matric and gravity components) specified within the solution.
This matric potential is specified by the user in the input file for SWIMv2.1.
In APSIM SWIM the potential is set initially and can be manipulated at run-time via the APSIM manager. The potential gradient above this node will fluctuate through time due to the influence of this fixed potential and the nodes above.
This boundary condition is often used for the simulation of the entry of water tables into the section of the soil profile being modelled.

- If the potential is zero or positive then there is effectively a water table entering the base of the profile. The magnitude of the potential would represent the height of water above the lowest node. Increasing this potential does not instantly 'flood' the soil to this height. It may take some time, depending upon the soil properties, for the upward flow to achieve this and for the soil to equilibrate. Similarly, it may take some time for drainage if there is any decrease in the specified potential. Note that drainage will still occur during the presence of the water table. Any infiltration that may tend to increase the height of the water table will cause a subsequent drainage such that the specified water table height is maintained.
- The flows of water across the bottom boundary for a negative boundary condition are similar to the case above. This boundary condition may perhaps be used for situations where a water table is known to drop a little below the specified profile, provided the conductivity is not too low.

Solute Flow

- In situations where the bottom boundary potential is maintained it is assumed that the solute concentration of the water at that boundary is also maintainedIf the potential is zero or positive then the solute concentration of the deepest node is maintained at a constant concentration. In APSIM SWIM this concentration is the concentration of solutes in ground water. In SWIMv2.1 the concentration is held at the value, csl(n),

specified by the user in the input file. This represents the fact that the ground water has entered the profile. It is assumed that the solute concentration within the ground water is spatially homogeneous. Any water that enters the profile from the ground water will contain solutes at this given concentration. The solute concentration within water draining out of the profile will be determined via the normal solute flux calculations. Note that if the bottom boundary is changed to such a condition there will be a sudden change to the solute balance as solute is removed or added (over the time step) to the deepest node as the new concentration is applied.

- If the potential is negative the solute concentration is held constant as described above. In this case though, the user will have to decide on the possible implications of the behavior of the solute balance at this base node.

4.3 Zero flux

There is no flux of water or solute across the bottom boundary when using this boundary condition.
The flow equations are solved with the flux term at the base of the profile set to zero.
The matric potential and potential gradients will fluctuate due to the influence of higher nodes.
Water may pond above this boundary.
Solutes will move in response to the water flows and concentration gradients.
No special conditions need to be considered by the user.

4.4 Seepage with threshold suction

The seepage boundary condition acts as a combination of the zero flux and constant potential boundary conditions.
This condition is useful for simulating the operation of some field and laboratory apparatus or the effective functioning of particular boundary interfaces (e.g. drainage into gravel).

Water Flow

If the potential at the deepest node is below a critical value specified by the user then the water balance calculations are identical to the zero flux bottom boundary condition.
No flux will occur out of the profile until the potential at the deepest node has reached the user-defined value.
When the potential does reach the user-defined value the boundary condition changes to follow the behaviour of the "user-specified potential" condition.
The flow equations are solved with the potential of the deepest node set to the specified value. No upward flux can occur.

- If the specified potential is zero or positive then drainage will not occur until the water table within the profile reaches the described height. This condition can be used to simulate apparatus where water tables can build up to a maximum height.
- If the specified potential is negative then water is removed from the profile until the deepest node equilibrates with the applied suction. This can be used to simulate core experiments where the base of the profile has an applied suction.

Solute Flow

Solute flows across the bottom boundary follow those for water. As implied by the nature of the boundary condition,
there can be no dispersion for flux of solutes out of the profile (the profile is discontinuous at this point).
Fluxes out of the profile, therefore, are calculated using convection only.

- If the specified potential is zero or positive solute will accumulate at the base of the profile as is the case for the accumulation of the perched water table. When drainage does occur, the solute concentration of the drainage water will equal that of the water at the deepest node.
- If the specified potential is negative then solute will be extracted with the soil water until the soil equilibrates. The absence of the dispersion term will lead to the extraction of water from the profile being 100% efficient at removing the solute in that water. Once again, the concentration of solute in drainage water will equal that of the water at the deepest node.

4.5 Responsive Water Table

The responsive water table condition acts like a water table with a user-specified level of 'responsiveness',
where responsiveness refers to the time required for the water table depth to equilibrate with the piezometric pressure.
This condition is useful for simulating the situations where local management might temporarily alter the effective water table depth.
For example, irrigation might temporarily raise a water table or plant water use might lower a water table from that which would correspond to the ground water pressure.

Water Flow

Water flow (in or out) is calculated based upon the difference between the piezometric pressure and the actual pressure at the base of the profile.
If the pressure is above the ground water pressure, water will flow out of the profile. If the pressure is below the ground water pressure,
water will flow into the profile.

The rate of water flow is calculated as follows:

$q = (W - (\psi - X)) \times g_w$

where,

- q is flow at the base of the profile,
- W is the depth to ground water,
- ψ is pressure head at the bottom of the profile,
- X is the depth to the bottom of the profile and
- g_w is a ground water table "conductance".

g_w has units of per hour.
Therefore, a value of 1 would provide an flow of 1 cm per hour for every cm of pressure difference at the base of the profile.
This same calculation is used to provide flows of water into or out of the profile.
The sign of the pressure difference term will dictate the direction of flow.

Solute Flow

Solute flows across the bottom boundary follow those for water. As implied by the nature of the boundary condition,
there can be no dispersion for flux of solutes out of the profile (the profile is discontinuous at this point).
Fluxes out of the profile, therefore, are calculated using convection only.

- In situations where a water table is maintained it is assumed that the solute concentration of the water at the bottom of the profile is also maintained at a constant concentration. In APSIM SWIM this concentration is the concentration of solutes in ground water. It is assumed that the solute concentration within the ground water is spatially homogeneous. Any water that enters the profile from the ground water will contain solutes at this given concentration.
- The solute concentration within water draining out of the profile will be determined via the normal solute flux calculations (convection only).

4.6 Runtime alterations to the bottom boundary condition

The user can change the bottom boundary condition and/or value at runtime via the setting of the state of the bottom boundary condition at any time via the messaging system.
The following manager code will change the bottom boundary to a water table for five days before reverting to a constant gradient at the end of this period.

```
If  day = 1 then
   Apswim.bbc_potential = 50
Elseif day = 6 then
   Apswim.bbc_gradient = 0
Endif
```

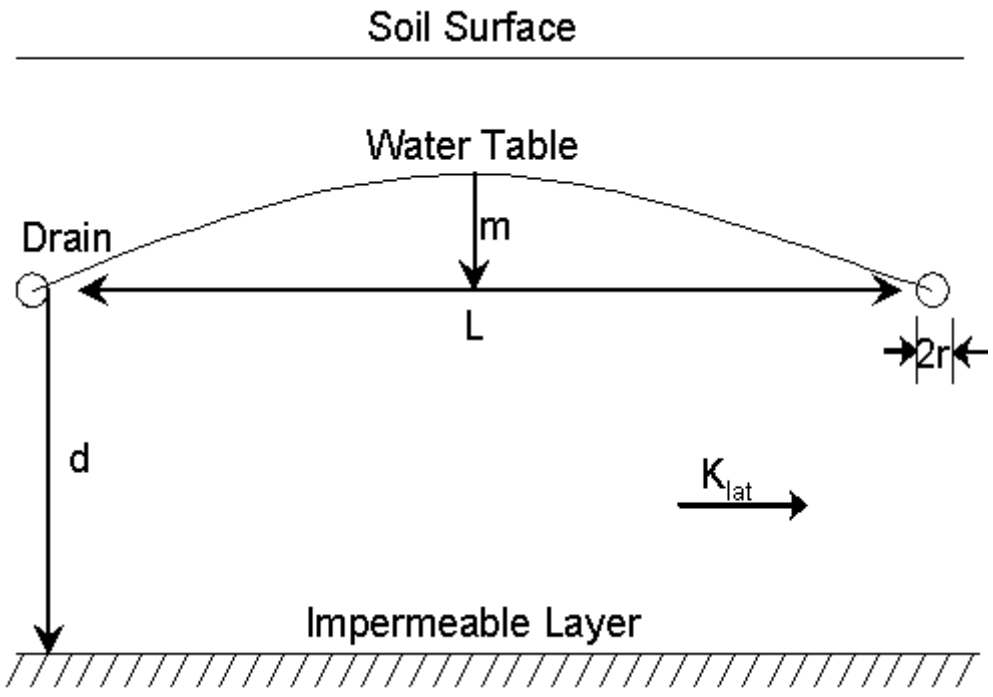
In this example days 1 to 5 will have a water table 50 cm above the bottom of the soil profile. Day 6 will be the start of a gradual drainage event out of this state.
Currently, these are the only two boundary condition state changes supported for alteration at runtime.

5 Subsurface Flows

5.1 Subsurface Drains

The flux of water and solutes into subsurface tile drains can be calculated using a formulation of the Hooghoudt equation as utilised in the DrainMod model.

The basic geometry of the system is as follows:



The flow into the drain (q) is calculated

$q = (8.0\ K_e\ d_e\ m + 4K_e\ m^2)/L^2$

where

- K_e is the effective lateral conductivity of the soil profile, m is the height of the water table above the drains,
- d is the height of the drains above the impermeable soil horizon,
- L is the drain spacing,
- r is the drain radius and d_e is the effective depth of impermeable soil horizon which takes into effect the convergence near to the drains.

d_e is calculated as:

$d_e = L*\pi/(8 \log (L/r) - 1.15)$, where $d/L >= 0.3$

$d_e = d/(1.0+d/L*(8\pi \log (d/r) - \alpha))$,

where $d/L < 0.3$, $\alpha = 3.55 - 1.6(d/L) + 2(d/L)^2$

The user specifies the system via input parameters for the various aspects of the geometry as follows:

```
[sample.apswim.drain]
drain_depth = 1000. (mm)       ! depth of the drain below the soil surface
drain_spacing = 30000 (mm)     ! distance between subsurface drains
drain_radius = 50 (mm)         ! radius of each tile drain
Klat = 1000. (mm/d)            ! lateral conductivity
imperm_depth = 3000.           ! depth of the impermeable soil profile below
the soil surface
```

6 APSIM SWIM Interface

6.1 Crop Interface to APSIM SWIM

All uptake of water and/or solutes by crops is handled by SWIM.
The uptake of these substances in built into the SWIM flow equations.
Competition between crops for resources such as water and solute is intrinsic to the solution of the flow equations and APSIM takes full advantage of this.
Competition of other resources such as light must be allowed for in another module.
Such modules are currently available within APSIM.

The crop module must also have been created from a template that allows for uptake calculation external to the module.

To enable SWIM to calculate crop uptakes each crop must be able to provide

Name	Units	Description
crop_type	-	name of crop type eg. wheat (not module name)
sw_demand	mm	potential plant transpiration
*nnnn*_demand	kg/ha	crop solute demand (where *nnnn* is the solute name)
cover_tot	0-1 fraction	total crop cover
rlv	mm/mm^3	root length volume(on a layer basis)

With this information SWIM will calculate the uptake of water and all solutes for each crop. These are available to the crop using the uptake_*nnnn_cccc* variable as described in the output section.
To be able to provide information such as potential plant transpiration for the current APSIM time step the crops will need to calculate these values in preparation for the time step (the PREPARE stage).
Also, SWIM will have to be included in the module list ahead of any crop or solute modules to allow SWIM to calculate ALL flows before the owner modules perform their daily time step.

6.2 Solute Interface to APSIM SWIM

The interface between solute owner modules and APSIM SWIM is very simple.
The APSIM SWIM parameter file must contain the names of all solutes the user is wanting SWIM to move for the current simulation.
APSIM SWIM will then ask all modules for the states of these pools, as does any other module requiring that information,
and will receive that information via the standard message passing system in APSIM.
Only the amount of solute in each layer (kg/ha) is required.
All movement parameters are included in the APSIM SWIM parameter file.
APSIM SWIM will notify the user if it has been parameterised to move solutes that do not exist within the simulation, or if solutes exist for which it is not parameterised.

To receive back updated values of solute, after solute flows are accounted for by SWIM, the solute owner module needs to be able to respond to the standard “SET” message.
This should also already be implemented in the solute owner module. **The solute module does not need to provide any solute flow information**.
As is shown, almost no effort is required to couple a solute module to APSIM SWIM.

NOTE: For those interested in the flow of simple solute tracers, without solute transformations, there is a simple solute module within APSIM that will allow the addition of any solute into the APSIM modelling system.

6.3 APSIM SWIM module actions

Reset

The reset action can be invoked to reset the module to the state specified within the module's input data,
which includes the soil moisture characteristics, runoff and boundary condition parameters and the initial soil water profile.
The Reset action is identical to the initialise action used by the simulation engine at the start of the simulation except that a description of the reinitialised state is not printed in the simulation summary file.

APSIM Manager Example:

```
[sample.manager.start_of_day]

! reinitialise apsim swim at the beginning of each sowing window

If day = 100 then
    apswim reset
endif
```

Initialise

The initialise action has now been replaced by the reset action (see above).

Summary Report

At initialisation, at series of tables and useful information is printed to the simulation summary file for perusal by the user.
These tables can be printed to the summary file at any point during the simulation as a detailed record of the system state at a particular time.

APSIM Manager Example:

```
[sample.manager.start_of_day]

! Print out a summary of module state to the summary file

If day = 100 then
    apswim sum_report
endif
```

APSIM
AGRICULTURAL PRODUCTION SYSTEMS SIMULATOR

WaterSupply

Description

APSIM WaterSupply is an instantiable module, capable of performing the role of water-source for the APSIM Irrigate module.
The module can be configured to simulate a dam, sump, river, bore or effluent source.
In any given APSIM simulation, different instances of WaterSupply can be configured to represent one or more of these water sources, at the same time.

For example:

```
version = 3.1
[apsim.sample_soilwat2]
title=Soilwat2 Sample Simulation
module = clock soilwat2.par [sample]
module = report soilwat2.par [sample]
module = Input(Met) %apsuite\apsim\met\sample\dalby.met[weather]
module = manager soilwat2.par [sample]
module = soilwat2 soilwat2.par [black_earth] soilwat2.ini [standard]
module = soiln2 soilwat2.par [black_earth]oiln2.ini [standard]
module = solute soilwat2.par [sample]solute.ini [standard]
module = Screen soilwat2.par [sample]
module = SummaryFile soilwat2.par [sample]
module = WaterSupply(dam) soilwat2.par [sample] dam.ini [standard]
module = WaterSupply(bore) soilwat2.par[sample]bore.ini [standard]
module = WaterSupply(sump) soilwat2.par[sample] sump.ini [standard]
module = irrigate
```

In the above simulation, WaterSupply is being instantiated to simulate a dam, a bore, and a sump.

Each instance of this module is capable of maintaining an available pool of water, with a dynamic solute composition, able to provide water or receive water as required.

Transfers of water between WaterSupply and Irrigate, or between WaterSupply and other instances of itself (eg dam and sump), are driven by commands from the APSIM Manager module.

The WaterSupply module is subject to various internal daily processes which produce a dynamic balance of both water and solutes in storage.
Depending on type of storage (eg dam or bore), these processes include direct capture of rainfall, capture of surface runoff (both local and catchment), evaporation and seepage losses, and renewal of allocations.

WaterSupply Parameterisation

The first and most critical piece of information required by the APSIM WaterSupply module during configuration is a parameter called "storage_type".

There are **six** discrete categories available, and each instance of this module must be configured as one of these:

1. dam_gully - a open surface storage formed by damming a watercourse
2. dam_ring - a open-surface ring-tank into which water is pumped for storage
3. dam_exc - an open-surface excavated storage
4. sump - an open sump which generally collects local and/or catchment runoff
5. river - a watercourse from which an annual pumping allocation is granted
6. bore - a underground water-source from which similar allocation is granted

nb. An Effluent Source would best be configured as type "bore", since it will not be subject to daily processes such as evaporation, seepage etc. (see later).

Depending on which "storage_type" is configured, the following parameters are required by APSIM WaterSupply:

For Dams and sumps

```
[sample.watersupply.parameters]

source_type = dam_ring          ! type of water source
receive_catchment_runoff = yes  ! use "yes" if dam collects runoff from
larger catchment area
catchment_area = 120.0          ! catchment area (ha) (required only if
"receive_catchment_runoff" equals "yes")
catchment_runoff_factor = 0.5   ! multiplier for soilwat2.runoff, applied to
catchment area (required only if "receive_catchment_runoff" equals "yes")
receive_crop_runoff = yes       ! use "yes" if dam collects runoff from
simulated crop area
max_available_water = 150.0     ! capacity (ML)
max_area = 20                   ! storage water surface area at capacity (ha)
init_available_water = 144.0    ! supply volume available at start of
simulation (ML)
max_pump = 20.0                 ! maximum pump delivery volume per day
(ML/day)
min_volume = 5.0                ! minimum volume in storage below which pump
cannot access
permeability = 0.00007          ! permeability of sealing layer (m/day)
seal_thickness = 0.5            ! thickness of low permeability seal (m)
init_br_conc = 100.0            ! initial bromide concentration (ppm)
```

For Bores and Rivers

```
[sample.watersupply.parameters]

source_type = bore           ! type of water supply
max_available_water = 400.0  ! maximum allocation including carry-overs (ML)
init_available_water = 200.0 ! volume available at start of simulation (ML)
max_pump = 20.0              ! maximum pump delivery volume per day (ML/day)
min_volume = 0.0             ! minimum volume in storage below which pump
cannot access
annual_allocation = 200.0    ! Annual Allocation in ML
allocation_renewal_day = 270 ! Day of year on which allocation is granted
init_br_conc = 100.0         ! initial bromide concentration (ppm)
init_cl_conc = 8.0           ! initial chloride concentration (ppm)
```

Parameterising Solutes

As can be seen in of the above examples, initial concentrations of solutes in the storage water can be specified using the parameter "init_xxx_conc" in parts-per-million,
where "xxx" is the name of the solute.

All solutes used in the system must also be initialized in the APSIM Solute module and given an initial distribution in the soil layers, even if this is zero in all layers.
As water is transferred between various elements in the simulation (sumps, dams, bores, irrigation, soil), solutes will be transferred accordingly.

WaterSupply Processes

Rainfall Capture

For open surface dams and sumps, rainfall results in a storage gain, expressed by:

rain_capture = (area* rain)/100 (1)

where,

- rain_capture = storage gain from rainfall event (Ml)
- area = capture surface area of dam (ha)
- rain = precipitation (mm)

Bore and river allocations are not affected by precipitation.

Runoff Capture

For open dams and sumps,

if the input parameter
`receive_crop_runoff = yes`
then
local runoff (soilwat2.runoff)
will be added to the storage water.

Soilwat2.runoff
is calculated in mm,
hence
crop_area (ha)
must be provided as a manager variable whenever WaterSupply is being used,
to convert to ML.

If the input parameter
`receive_catchment_runoff =  yes`
then the user must also supply two further parameters
catchment_area (ha)
and
catchment_runoff_factor (0-1).

The latter of these two subsequent inputs is a multiplier to the soilwat2.runoff parameter, effectively describing the water-shedding potential of the catchment in comparison to the cropped area.

If none of the instances of WaterSupply in a given simulation are configured to receive runoff, then runoff is lost from the system.

Runoff capture is calculated on a daily basis.

Evaporation

For open dams and sumps, loss of water to the atmosphere through evaporation is a daily occurrence. Evaporation is calculated in a two stage process as follows:
From CERES maize soil evaporation

soil_evaporation = radn*23.8846*(0.000204-(0.000183*0.1)) *(29+(0.6* maxt+0.4* mint)) (2)

where

- radn = incident solar radiation (Mj/m2)
- maxt = maximum daily temperature (oC)
- mint = minimum daily temperature (oC)

The evaporation from the surface of the dam/sump is then calculated according to:

evaporation = vol-(area*((d-(0.7*se/1000))** b)) (3)

where

- vol = current storage volume (Ml)
- area = dam surface area at capacity (ha)
- d = current storage depth (m)
- se = soil evaporation (as calculated above)
- b = geometry factor for dam type (from ini file)

Seepage

For open dams and sumps, loss of water through seepage out the base and sides is also a daily occurrence. Seepage is calculated as follows:

seepage = vol-(area*((d-(k*(d/st)/365.0))** b)) (4)

where

- vol = current storage volume (Ml)
- area = dam surface area at capacity (ha)
- d = current storage depth (m)
- k = permeability of base (m/day)
- st = seal thickness (m)
- b = geometry factor for dam type (from ini file)

Overflow

For open dams and sumps, incoming water which takes the storage volume above capacity is reported daily as "overflow" (ML).

Allocation Renewal

For bores and rivers, an allocation renewal check is made every day, and on the day specified as "renewal day" in the input parameters, the "annual_allocation" is cred to the available storage volume. The amount of carry-over allowed is specified by the user in setting "max_available water".

Using WaterSupply with APSIM Irrigate

APSIM Irrigate works on a "mm" basis, whereas APSIM WaterSupply works on real volumes (ML).
Hence, when an irrigation application is specified in mm, an "area of application" must be provided in order to calculate the required volume of water from the specified source instance of WaterSupply.

As mentioned previously, whenever WaterSupply is used in a simulation for supply of irrigation water, a variable called "crop_area" (ha) must be specified in the manager logic.

Irrigations using water from WaterSupply can only be initiated by using the "irrigate apply" action in manager.

A new optional argument called "source" is added to the "apply" command line to trigger the use of water from WaterSupply.
The required syntax is as follows:

```
[sample.manager.start_of_day]

if day = 10 then
     irrigate apply amount=10 (mm), source = dam bore dam2 ()
endif
```

The argument "source" specifies the sources from which to obtain the irrigation water, in preferential order.
In other words, in the above example, if the dam cannot fully supply the required water, the balance will be taken from the bore.
If there is still a shortage of water, then dam2 will be asked next to supply water.
There is no limit to the number of sources which can be specified.

When the irrigation water is applied to the soil, it will carry the solutes makeup of the water source being used.

Transfering Water between Instances of WaterSupply

In the general operation of an irrigation and water-storage system, there may be numerous occasions where the farmer wishes to transfer water between a sump and the dam, or to pump water from a bore or river into a dam.

This is easily achieved in APSIM WaterSupply by employing the "TOP_UP" command from manager:

To transfer 5 megalitres of water from a sump to a dam, for example:

```
[sample.manager.start_of_day]

if day = 15 then
     dam top_up amount=5 (Ml), source= sump
endif
```

Or else, the dam (say, capacity =150 Ml) may need to be filled from the bore, without knowing the exact amount required:

```
[sample.manager.start_of_day]

dam_deficit = 150-dam.available_water

if day = 15 then
    dam top_up amount=dam_deficit, source= bore
endif
```

Or else, the farmer may wish to pump to the dam whenever the sump is full:

```
[sample.manager.start_of_day]

sump_volume = sump.available_water
sump_deficit = sump.max_available_water - sump_volume

if sump_deficit = 0 then
    dam top_up amount=sump_volume, source= sump
endif
```

The preferential ordering of sources is available with the "top_up" command, as for the Irrigate "apply" command.
In all of these examples, the "top_up" command will transfer the specified water from the source to the destination and calculate the new solute figures in the destination pool.

WaterSupply Actions and Events

Figure 1: UML diagram for an Irrigate "apply" action (specifying "source = dam") showing responses from modules concerned.

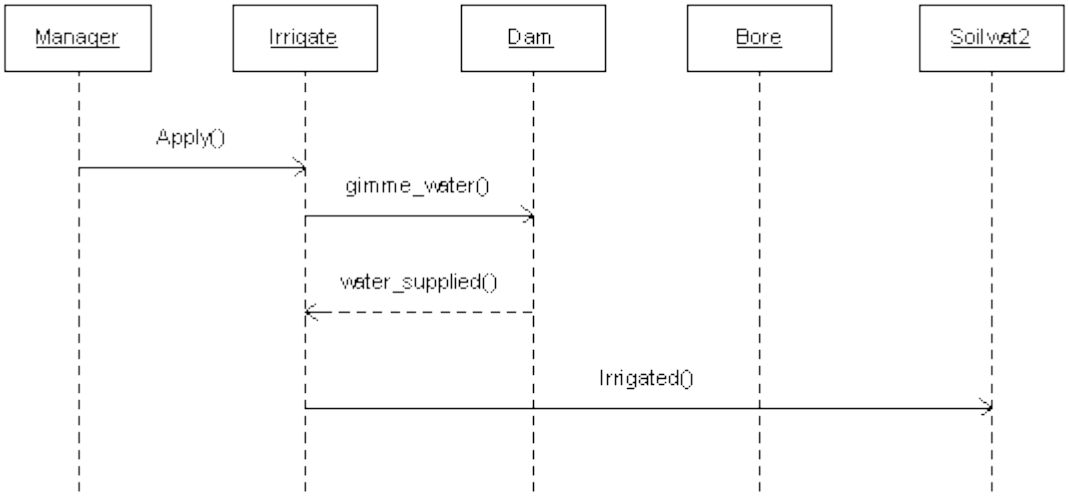


Figure 2: UML diagram for a WaterSupply (dam) "top_up" action (specifying "source = bore") showing responses from modules concerned.

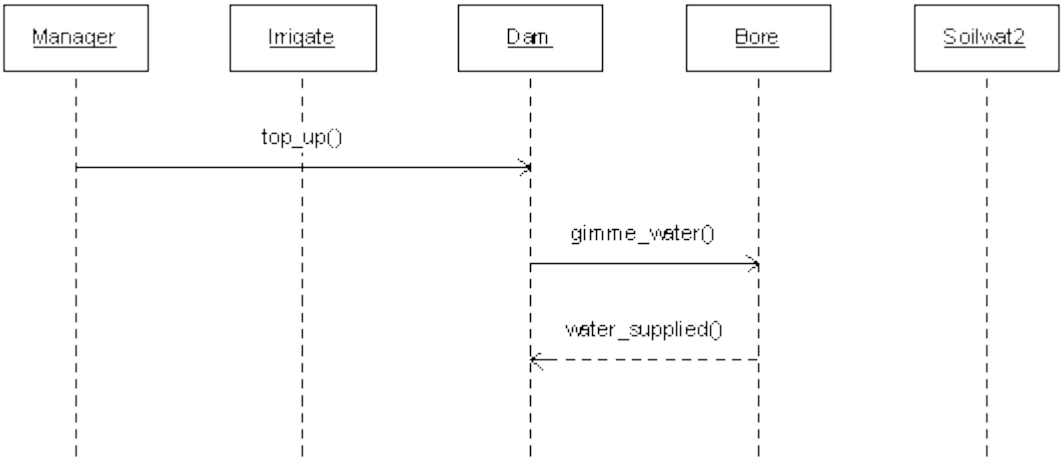
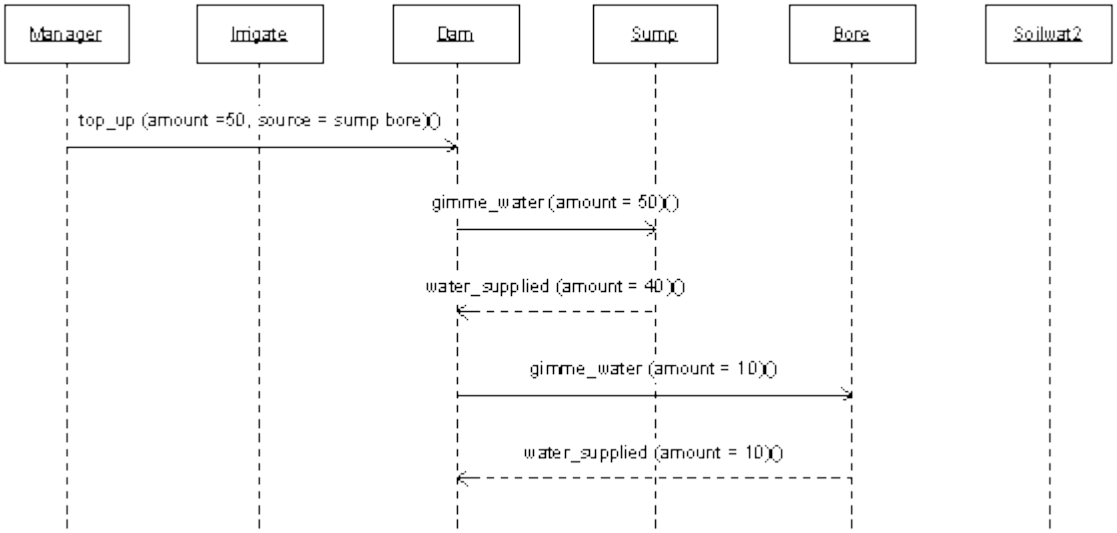


Figure 3: UML diagram for a WaterSupply (dam) "top_up" action (specifying "source = sump bore") in the situation where the first specified source is not capable of supplying all of the water required. In this case supplementary water is required from the second specified source.



WaterSupply Module Outputs

Name	Units	Description
available_water	Ml	Current storage volume
available_depth	m	Current storage depth
max_available_water	Ml	Storage capacity or max allocation carry-over
min_volume	Ml	Storage volume below which pumping not allowed
max_pump	Ml/day	Maximum daily pump delivery
annual_allocation	Ml	Annual water allocation (bores, rivers)
allocation_renewal_day	doy	Day of year for renewal of allocation (bores, rivers)
rain_capture	Ml	Rainfall captured by dam area
evaporation	Ml	Daily water loss to atmosphere
seepage	Ml	Dam seepage loss through base and sides
overflow	Ml	Input water above storage capacity
runoff_input	Ml	Daily runoff added to storage
storage_xxx	ppm	Storage concentration of solute "xxx", eg storage_cl
irrigation_water_supplied	ML	Water supplied from storage at request of APSIM Irrigate
full	(0-1)	Flag indicating whether a storage is full (1) or not full(0)
filling_event	(0-1)	Flag indicating whether a filling event occurred today

References

Jones, C.A., and J.R. Kiniry. 1986.
CERES-Maize: A simulation model of maize growth and development.
Texas A&M University Press, College Station, Texas.