Reworking CTSM’s Water Flux Calculations

Bill Sacks
CESM Software Engineering Group (CSEG)

with many contributions from Mat Rothstein (CSEG)

and also: Mike Barlage, Martyn Clark, Dave Lawrence,
David Noone, Sean Swenson, Mariana Vertenstein
Outline

• What is the problem?

• Why is this a problem?

• What are we doing?

• Two metaphors that inspire me
subroutine SurfaceRunoff (...) 

! Inputs: [cut for brevity] 
!
! Input/outputs: 
! - h2osoi_liq: liquid water (mm) 
! - qflx_top_soil: net water input into soil from top (mm/s) 
!
! Outputs: 
! - qflx_surf: surface runoff (mm/s) 
! - fcoov: fractional impermeable area 
! - fsat: fractional area with water table at surface 
! - fracice: fractional impermeability 
! - icefrac: fraction of ice 
! - max_infil: maximum infiltration capacity in VIC (mm) 
! - i_0: column average soil moisture in top VIC layers (mm)
What is the problem? (2)

doj = 1,nlevsoi
    donum_hydrologyc
        c = filter_hydrologyc(fc)
        ! --- Set icefrac and fracice ---
    end do
end do


dofc = 1, num_hydrologyc
    c = filter_hydrologyc(fc)
    if (use_vichydro) then
        ! --- Set a bunch of auxiliary VIC variables ---
        fsat(c) = ...
    else
        fsat(c) = ...
    end if
What is the problem? (3)

```plaintext
if (frost_table(c) > zwt(c)) then
    if (use_vichydro) then
        fsat(c) = ...
    else
        fsat(c) = ...
    end if
else
    if (frost_table(c) > zwt_perched(c)) then
        fsat(c) = ...
    end if
end if
end do
```
do fc = 1, num_hydrology
    c = filter_hydrology(fc)
    qflx_surf(c) = fsat(c) * qflx_top_soil(c)
end do

do fc = 1, num_urban
    c = filter_urban(fc)
    ! --- For urban: set qflx_surf ----
    ! --- and update h2osoi_liq(c,1) ----
end do

do fc = 1, num_hydrology
    c = filter_hydrology(fc)
    ! --- Add some terms to qflx_top_soil ----
end do

end subroutine SurfaceRunoff
What is the problem? (Full routine)
Why is this a problem? (Right brain)
Why is this a problem?
(Right brain)
Why is this a problem? (Right brain)
Why is this a problem? (Right brain)
Why is this a problem? (Left brain)

• Hard to understand and modify the code

• Hard to bring in alternative parameterizations
  ▸ Incorporating Noah-MP parameterizations into CTSM

• Numerics (state updates) entangled with physics: can’t implement better numerical solutions

• Hard to implement water tracers / isotopes
Why is this a problem?
Water tracers / isotopes

$$Flux_{tracer} = Flux_{bulk} \cdot \frac{Source_{tracer}}{Source_{bulk}}$$
Why is this a problem?
Water tracers / isotopes

\[ \text{Flux}_{\text{tracer}} = \text{Flux}_{\text{bulk}} \cdot \frac{\text{Source}_{\text{tracer}}}{\text{Source}_{\text{bulk}}} \]

if \( \text{xs}(c) > 0. \) then

\[ \text{h2osoi_liq}(c,1) = \text{pondmx}_\text{urban} \]
Why is this a problem?  
Water tracers / isotopes

\[ \text{Flux}_{\text{tracer}} = \text{Flux}_{\text{bulk}} \cdot \frac{\text{Source}_{\text{tracer}}}{\text{Source}_{\text{bulk}}} \]

\[
\text{if } (\text{x}\text{s}(c) > 0) \text{ then } \text{h2o}\text{sbi}_{\text{liq}}(c) = \text{pondm}\text{x}_{\text{urban}}
\]
What are we doing? – schematic

**Before**

<table>
<thead>
<tr>
<th>some_subroutine</th>
</tr>
</thead>
<tbody>
<tr>
<td>calc. flux 1</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>update state 1</td>
</tr>
<tr>
<td>calc. flux 2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>calc. flux 3</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>update state 2</td>
</tr>
</tbody>
</table>
What are we doing? – schematic

**Before**

- `some_subroutine`
  - `calc. flux 1`
  - ...
  - `update state 1`
  - `calc. flux 2`
  - ...
  - `calc. flux 3`
  - ...
  - `update state 2`

**After**

- `sub_driver`
  - `calc. flux 1`
    - select case …
    - case `XXX`
    - case `YYY`
    - extra code
    - common to all
  - `calc. flux 2`
    - select case …
    - case `AAA`
    - case `BBB`
  - `calc. flux 3`
    - `calc. flux 3 using single available method`

- `update states`
  - `update state 1`
  - `update state 2`

- `flux1_XXX` (calc. flux 1 using method `XXX`)
- `flux1_YYY` (calc. flux 1 using method `YYY`)
- `flux2_AAA` (calc. flux 2 using method `AAA`)
- `flux2_BBB` (calc. flux 1 using method `BBB`)

**Benefits**

- Partitions complexity
- Easier to plug in alternative parameterizations
- Separating flux calculations from state updates facilitates:
  - Adding water tracers / isotopes
  - Separating physics from numerics
What are we doing? – code (1)

```plaintext
subroutine SaturatedExcessRunoff (...)
! Inputs: [cut for brevity]
!
! Outputs:
! - qflx_sat_excess_surf: surface runoff due to saturated surface (mm/s)
! - fcov: fractional impermeable area
! - fsat: fractional area with water table at surface
```
What are we doing? – code (2)

```fortran
select case (this%fsat_method)
  case (FSAT_METHOD_TOPMODEL)
    call this%ComputeFsatTopmodel(..., fsat)
  case (FSAT_METHOD_VIC)
    call this%ComputeFsatVic(..., fsat)
  case default
    call endrun(subname//' ERROR: Unrecognized fsat_method')
end select
```
What are we doing? – code (3)

```plaintext
do fc = 1, num_hydrologyc
    c = filter_hydrologyc(fc)
    qflx_sat_excess_surf(c) = fsat(c) * qflx_rain_plus_snomelt(c)
end do

end subroutine SaturatedExcessRunoff
```
subroutine ComputeFsatTopmodel(..., fsat)
! [Input argument declarations cut for brevity]
real(r8), intent(inout) :: fsat(:) ! fractional area with water table at surface

do fc = 1, num_hydrology
  c = filter_hydrology(fc)
  if (frost_table(c) > zwt_perched(c) .and. frost_table(c) <= zwt(c)) then
    fsat(c) = ...
  else
    fsat(c) = ...
  end if
end do

end subroutine ComputeFsatTopmodel
Data structure rework for water tracers and isotopes

**Before**

| waterstate_inst | var1 | var2 | var3 | var4 | var5 | var6 | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>waterflux_inst</td>
<td>flux1</td>
<td>flux2</td>
<td>flux3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data structure rework for water tracers and isotopes

Before

<table>
<thead>
<tr>
<th>waterstate_inst</th>
<th>waterstate_bulk_inst</th>
</tr>
</thead>
<tbody>
<tr>
<td>var1</td>
<td>var1</td>
</tr>
<tr>
<td>var2</td>
<td>var2</td>
</tr>
<tr>
<td>var3</td>
<td>var3</td>
</tr>
<tr>
<td>var4</td>
<td>...</td>
</tr>
<tr>
<td>var5</td>
<td></td>
</tr>
<tr>
<td>var6</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>waterflux_inst</th>
<th>waterflux_bulk_inst</th>
</tr>
</thead>
<tbody>
<tr>
<td>flux1</td>
<td>flux1</td>
</tr>
<tr>
<td>flux2</td>
<td>flux2</td>
</tr>
<tr>
<td>flux3</td>
<td>flux3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>waterstate_inst</th>
<th>waterstate_bulk_inst</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>var4</td>
</tr>
<tr>
<td></td>
<td>var5</td>
</tr>
<tr>
<td></td>
<td>var6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>waterdiag_inst</th>
<th>waterdiag_bulk_inst</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>var4</td>
</tr>
<tr>
<td></td>
<td>var5</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>waterflux_inst</th>
<th>waterflux_bulk_inst</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>flux1</td>
</tr>
<tr>
<td></td>
<td>flux2</td>
</tr>
<tr>
<td></td>
<td>flux3</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Principles:

1. Avoid needing to change lots of science code
2. Scientists can do initial development without needing to know about tracers
Two metaphors that inspire me
Preparatory refactoring
for each desired change, make the change easy (warning: this may be hard), then make the easy change
Livable code


You have to live here
Livable code


You GET to live here