Comparing averages between Fortran scripts and PFTools Python functions

Post-Processed Outputs – MM’s CONUS1 paper

All of MMs CONUS1 outputs were processed using fortran scripts that can be found on `/glade/p/univ/ucsm0002/CONUS_modern/CONUS.WY2003/scripts`

These consist of daily, monthly, and yearly averages of the following variables (variable and var name in Fortran script in parentheses)
ParFlow Averages:

- Streamflow (flow) ✅
- Soil moisture (SM) - average ✅
- Water table depth (WTd) - accumulated ✅
- Total Storage (storage)- accumulated ❌
- GW storage (GWstor)❌
- Soil moisture storage (SMstor)❌

Storage sums:

- Surface Water Storage (surf_wat)✅
- Total water storage (tws) = subsurface + surface + SWE storages ❌ *(don't know which PFTools "subsurface storage" this is and how SWE storage is different than just the average calculated in the clm_averages.f90)*

CLM Averages:

- Latent heat (LH) – CLM out layer 1 [W/m^2] - average
- Sensible heat flux (SH) – CLM out layer 3 [W/m^2] - average
- ground evaporation without condensation (qflx_grnd) – CLM out layer 6 [mm/s] - accumulated
- Vegetation transpiration (qflx_trans) – CLM out layer 9 [mm/s] - accumulated
- Snow water equivalent (SWE) – CLM out layer 11 [mm] - average??
- Ground temperature (Tgrnd) – CLM out layer 12 [K] skin temp - average
- Soil temperature (Tsoil) – CLM out layer 14 [K] @5cm - average

Other:

- Evapotranspiration (calculated from PFTools, but not in fortran)❌

ParFlow Averages

Streamflow

Fortran:

```
flow(i,j) =  flow(i,j) +         &
1000.d0*((max(press(i,j,5),0.0d0))**(5.0/3.0))* &
sqrt(max(abs(Sx(i,j,1)),abs(Sy(i,j,1))))/mannings      !acc
um
```

PFTools Python:
`calculate_overland_flow_grid(pressure, slopex, slopey, mannings, dx, dy, flow_method='OverlandKinematic', epsilon=1e-5, mask=None)`

```
    """
    Calculate overland outflow per grid cell of a domain
    :param pressure: A nz-by-ny-by-nx ndarray of pressure v
alues (bottom layer to top layer)
    :param slopex: ny-by-nx
    :param slopey: ny-by-nx
    :param mannings: a scalar value, or a ny-by-nx ndarray
    :param dx: Length of a grid element in the x direction
    :param dy: Length of a grid element in the y direction
    :param flow_method: Either 'OverlandFlow' or 'OverlandK
inematic'
        'OverlandKinematic' by default.
    :param epsilon: Minimum slope magnitude for solver. Onl
y applicable if kinematic=True.
        This is set using the Solver.OverlandKinematic.Epsi
lon key in Parflow.
    :param mask: A nz-by-ny-by-nx ndarray of mask values (b
ottom layer to top layer)
        If None, assumed to be an nz-by-ny-by-nx ndarray of
1s.
    :return: A ny-by-nx ndarray of overland flow values
    """
```

Soil Moisture

Fortran:

```
sm(:,:) = sm(:,:) + sat(:,:,5)*porosity(:,:,5) / 24.0d0
!avg
```

PFTools Python:
`soil_moisture += saturation * porosity`

Water Table Depth

Fortran:

```
WTd(:,:) = WTd(:,:) + (52.0d0 - press(:,:,1))/ 24.0d0
!avg
```

PFTools Python:
`calculate_water_table_depth(pressure, saturation, dz)`

```
    """
    Calculate water table depth from the land surface
    :param pressure: A nz-by-ny-by-nx ndarray of pressure v
alues (bottom layer to top layer)
    :param saturation: A nz-by-ny-by-nx ndarray of saturati
on values (bottom layer to top layer)
    :param dz: An ndarray of shape (nz,) of thickness value
s (bottom layer to top layer)
    :return: A ny-by-nx ndarray of water table depth values
(measured from the top)
    """
```

Total Storage

I ***think*** what's going on with "Total Storage" in fortran is that it is computing all storage from all subsurface layers (i.e., k = 1,5). This differs from the fortran "Groundwater Storage" which ***appears*** to be only subsurface storage from the deepest layer. Soooooo, I think that `calculate_subsurface_storage` is the function that aligns with "Total Storage" because it should calculate all layers.
Fortran:

```
do k = 1, 5
storage(i,j) = storage(i,j) + 1000.d0*1000.d0*dz(k)*    &
               (press(i,j,k)*Ss*Sat(i,j,k) +     &
               Sat(i,j,k)*porosity(i,j,k))/24.0d0
end do !k - total storage
```

PFTools Python:
`calculate_subsurface_storage(porosity, pressure, saturation, specific_storage, dx, dy, dz, mask=None)`

```
    """
    Calculate gridded subsurface storage across several lay
ers.
    For each layer in the subsurface, storage consists of t
wo parts
      - incompressible subsurface storage
        (porosity * saturation * depth of this layer) * dx
* dy
      - compressible subsurface storage
        (pressure * saturation * specific storage * depth o
f this layer) * dx * dy
    """
```

Groundwater Storage

I don't think there is a function in PFTools for this, because If I'm understanding the fortran code for Groundwater Storage, it is only the deepest subsurface layer that is considered. **How do we want to treat this in CONUS2? How many layers (or how deep)?**

Fortran:

```
do k = 1, 1
GWstor(i,j) = GWstor(i,j) + 1000.d0*1000.d0*dz(k)*    &
               (press(i,j,k)*Ss*Sat(i,j,k) +     &
               Sat(i,j,k)*porosity(i,j,k))/24.0d0
end do !k - groundwater storage
```

PFTools Python:
Calculate subsurface storage for deep layers

EL: To me Groundwater storage is storage below the WTD. We could easily implement this by getting subsurface storage and setting it to 0 below wtd

Soil Moisture Storage

Similarly to the "Total Storage" and "Groundwater Storage," if I understand the `k` correctly, this should be calculating the subsurface storage in the top 4 soil layers (layers 2-5), so everything except the "Groundwater Storage". **Fortran**:

```
do k = 2, 5
SMstor(i,j) = SMstor(i,j) + 1000.d0*1000.d0*dz(k)*    &
               (press(i,j,k)*Ss*Sat(i,j,k) +     &
               Sat(i,j,k)*porosity(i,j,k))/24.0d0
end do !k - soil storage
```

PFTools Python:
Calculate subsurface storage for soil layers

EL: to me this would be compressible and incompressible above the wtd. So we could do complementary of the gw storage

******If the above logic is correct, then Groundwater Storage (layer 1) + Soil Storage (layers 2-5) = Total Storage, (which is different than Total Water Storage (tws)!!!)***

Storage Sums

Surface Water Storage

This computes any positive pressure?

Fortran:

```
surf_wat(i,j) = surf_wat(i,j) + max(press(i,j,5),0.0d0)/ 24
.0d0        !avg
```

PFTools Python:
`calculate_surface_storage(pressure, dx, dy, mask=None)`

```
    """
    Calculate gridded surface storage on the top layer.
    Surface storage is given by:
      Pressure at the top layer * dx * dy (for pressure val
ues > 0)
    :param pressure: A nz-by-ny-by-nx ndarray of pressure v
alues (bottom layer to top layer)
    :param dx: Length of a grid element in the x direction
    :param dy: Length of a grid element in the y direction
    :param mask: A nz-by-ny-by-nx ndarray of mask values (b
ottom layer to top layer)
        If None, assumed to be an nz-by-ny-by-nx ndarray of
1s.
    :return: An ny-by-nx ndarray of surface storage values
    """
```

Total Water Storage

I think that I have this correct, but I am not sure if SWE Storage needs to be calculated differently than it is in the clm_averages.f90, where the daily average is found. Maybe an accumulation?

Fortran:

```
! subsurface storage (portion for 1 hour)
do k = 1, 5
substorage = substorage + dz(k)*    &
               (press(i,j,k)*Ss*sat(i,j,k) +     &
               sat(i,j,k)*porosity(i,j,k))
end do !k - total storage

! surface storage (portion for 1 hour)
surfstorage = max(press(i,j,5),0.0d0)

! swe storage (portion for 1 hour)
swestorage = CLM(i,j,11)

! total storage (add the storage for this hour, divide by 24 for daily mean)
storage(i,j) = storage(i,j) + (substorage*1000 + surfstorage*1000 + swestorage)/24.0d0
```

PFTools Python:
`calculate_subsurface_storage + calculate_surface_storage + SWE (STORAGE?) = TWS`

EL: SWE storage would be swe from clm output * 1000 (mm->m)*dx*dy

CLM Averages

Variables as direct CLM outputs

- Latent heat (LH) – CLM out layer 1 [W/m^2] - average
- Sensible heat flux (SH) – CLM out layer 3 [W/m^2] - average
- ground evaporation without condensation (qflx_grnd) – CLM out layer 6 [mm/s] - accumulated
- Vegetation transpiration (qflx_trans) – CLM out layer 9 [mm/s] - accumulated
- Snow water equivalent (SWE) – CLM out layer 11 [mm] - average
- Ground temperature (Tgrnd) – CLM out layer 12 [K] skin temp - average
- Soil temperature (Tsoil) – CLM out layer 14 [K] @5cm - average

```
SWE(:,:) = SWE(:,:) + CLM(:,:,11) / 24.0d0      !avg
Tgrnd(:,:) = Tgrnd(:,:) + CLM(:,:,12) / 24.0d0      !avg
! should I multiply these two fluxes by one hour (3600 s)?
qflx_grnd(:,:) = qflx_grnd(:,:) + CLM(:,:,6)     !accum
qflx_trans(:,:) = qflx_trans(:,:) + CLM(:,:,9)   !accum
Tsoil(:,:) = Tsoil(:,:) + CLM(:,:,14) / 24.0d0      !avg
LH(:,:) = LH(:,:) + CLM(:,:,1)/ 24.0d0      !avg
SH(:,:) = SH(:,:) + CLM(:,:,3)/ 24.0d0      !avg
```

Evapotranspiration

Fortran:
Not sure this was computed in the fortran scripts I have available.

PFTools Python:
`calculate_evapotranspiration(et, dx, dy, dz, mask=None)`

```
    """
    Calculate gridded evapotranspiration across several lay
ers.
    :param et: A nz-by-ny-by-nx ndarray of evapotranspirati
on flux values with units 1/T (bottom layer to top layer)
    :param dx: Length of a grid element in the x direction
    :param dy: Length of a grid element in the y direction
    :param dz: Thickness of a grid element in the z directi
on (bottom layer to top layer)
    :param mask: A nz-by-ny-by-nx ndarray of mask values (b
ottom layer to top layer)
        If None, assumed to be an nz-by-ny-by-nx ndarray of
1s.
    :return: A nz-by-ny-by-nx ndarray of evapotranspiration
values (units L^3/T), spanning all layers (bottom to top)
    """
```

EL I don't think we should use evaptrans files (cause that's not et, and this function works only if you do use evaptrans)

I'd suggest using qflx_evap_tot * 3600(/s to /h) /1000 (mm to m) *dx *dy

Meteorological Variables

Vapor Pressure Deficit

- NLDAS Temperature
- NLDAS Air Pressure
- NLDAS Specific Humidity

In [ ]:

```

```