LGARTO Map

LGARTO is a vadose zone model that simulates soil moisture profiles and fluxes relevant for the vadose zone, in particular AET, infiltration, recharge, and saturation or infiltration excess runoff. It does not simulate lateral vadose fluxes. It allows for the representation of an arbitrary number of soil layers.

Input Files (standalone mode)

Config File (txt)
Provides parameters independent of soil type and file paths for forcing data and soils data

Forcing File (csv OR txt)
Forcing data to run the model including PET and precipitation

Soil Parameter file (dat)
Provides Van Genuchten parameters for various soil types

main (in bmi_main_lgar.cxx)
Initializes the model's state, runs Update over each timestep and produces model outputs

initialize
Using config and soil parameter file, alters model state to contain initial soil moisture profile and initializes mass balance variables

Update

repeat for each timestep

Determine number of subcycles for this timestep

repeat next box for each substep

calc_aet
Calculates AET from PET for each surface WF

Checks if a new wetting front should be created via boolean create_surficial_front

if surficial front must be created

lgar_move_wetting_fronts
see page 2
updates model soil moisture profile and fluxes

lgar_create_surficial_front
creates a new wetting front
updates soil moisture profile and fluxes

if no new surficial front will be created

there is ponded depth

lgar_insert_water
calculates infiltration capacity and reduces ponded head by allowed infiltration, also simulates runoff

lgar_move_wetting_fronts
see page 2
updates soil moisture profile

there is no ponded depth

lgar_dzdt_calc
calculates velocity (dzdt) for all wetting fronts
updates dzdt for each wetting front in linked list

ensures mass balance closes within tolerance

giuh_convolution_integral
calculate giuh runoff

update global mass balance variables

Finalize

prints cumulative fluxes for entire simulation, clears LGAR variables from memory

Output Files (technically updated after each call to Update, from main)

data_layers.csv
Soil moisture profiles for each timestep

data_variables.csv
Fluxes for each timestep.
[precipitation, PET, AET, surface_runoff, giuh_runoff, soil_storage, total_discharge, infiltration, percolation, groundwater_to_stream_recharge, mass_balance]

functions
files

LGAR video: https://www.hydroshare.org/resource/46747d77d0ce4995b1e1d58384e15a09/
LGARTO video: https://github.com/NOAA-OWP/LGAR-C/tree/LGARTO/tests/videos

lgar_move_wetting_fronts function: surface WFs

This function updates the positions of all wetting fronts for a substep, updates various fluxes, and returns recharge through the lower boundary. It updates wetting front depths based on their dzdt and / or AET values, updates moistures of eligible WFs, adds infiltration to an eligible region of the vadose zone if there is water to add, and subtracts AET demand from eligible regions. It also enforces that capillary head is the same on either side of a soil layer boundary with iterative mass balance calculations.
In the context of this code, a wetting front is defined as a region of soil water that shares a volumetric water content and capillary head value, and each wetting front is therefore represented as a theta-depth pair in the linked list representing the soil moisture profile. Adjacent wetting fronts not in the same soil layer must have the same capillary head. Two types of wetting fronts are defined: surface WFs, which are in contact with the soil surface, and groundwater or Talbot-Ogden (GW or TO, interchangeably) WFs, which are in contact with the water table.

Inputs

- TO_enabled: boolean that determines if the model is running in LGARTO mode (true) or LGAR mode (false)
- timestep_h: sub time step in hours
- free_drainage_subtimestep_cm: free drainage in cm, leaving the model lower boundary. Only relevant if free_drainage_enabled is set to true in the config file and TO_enabled is set to false
- PET_timestep_cm: the potential evapotranspiration for the sub time step in cm
- wilting_point_psi_cm: wilting point in cm, specified in the config file. Note that there is a single value for the entire model domain.
- field_capacity_psi_cm: field capacity in cm, specified in the config file. Note that there is a single value for the entire model domain.
- root_zone_depth_cm: the depth above which AET is extracted from the soil moisture profile in most cases, specified in the config file.
- volin_cm: the mass of water, in cm, that enters (infiltrates to) the vadose zone in this substep
- wf_free_drainage_demand: this is an integer which indicates which wetting front will be augmented by infiltration, and by free drainage if it is enabled. Note that all surface wetting fronts above this one will share its capillary head value and also will be affected by infiltration and free drainage if enabled.
- old_mass: the mass of water in the soil at the start of the time step in cm, which is necessary because mass conservation should be enforced
- num_layers: the integer number of soil layers in the model domain
- surf_frac_rz: the fraction of the root zone occupied by surface wetting fronts
- AET_demand_cm: the mass of water, in cm, that will be extracted via AET from surface wetting fronts, if in LGAR mode
- cum_layer_thickness_cm: an array that contains the cumulative thickness, from the soil surface downward, to the bottom of each soil layer
- soil_type: an array that contains the integers corresponding to the soil type for each layer
- frozen_factor: a factor that augments K_s and therefore ultimately dzdt for wetting fronts, if the model is coupled to SFT
- head: a pointer to the first element in the linked list describing the soil moisture profile, is necessary in the event that multiple instances of CASAM are run at the same time when using the NextGen
- state_previous: the soil moisture profile from the previous sub time step, will be necessary for mass balance calculations
- soil_properties: contains soil parameters for each layer
- surf_AET_vec: the vector containing the contribution to AET of each surface WF, only relevant if in LGARTO mode

Iterate over each surface WF, from lowest to highest

Case 1: WF is not in the same layer as the one below it and is not the last WF

In this simple case, the wetting front is the lowest one in its layer. Therefore it must have the same capillary head as the wetting front directly below it, and its dzdt value will always be 0. In this step, the capillary head is set to be equal to the capillary head directly below, and theta is updated accordingly.

Case 2: WF is the last WF in the list and there is exactly 1 WF per layer (special case)

In this special case, there is exactly 1 WF per layer, meaning that there is exactly 1 value for capillary head throughout the vadose zone. AET and infiltration contribute to changing the mass but not the depth of this WF, and a modified version of the mass balance code in case 3 is used because there is no theta value below this WF. The WFs above this one will later have their psi and theta values adjusted as part of case 1. Case 2 is the only instance in which a wetting front that is at the bottom of its layer will be updated without considering the water content below (because in this case there is no water content below).

prior_mass and new_mass are calculated. In this case, prior_mass and new_mass should only differ by any relevant AET and infiltration fluxes that will affect WF moisture.

→ lgar_theta_mass_balance: this function will calculate the theta value for the WF that conserves mass, considering all WFs above (because in this case they will extend to the bottom of their layers) and considering any AET, free drainage, or infiltration fluxes.

→ Capillary head is updated based on theta.

Case 3: WF is in the same layer as the WF below it and is not the last WF

In this case, the WF does not extend to the bottom of its layer. Therefore, it is free to move downwards, and this is the only case for surface WFs for which dzdt is not 0. The wetting front will be extended downward equal to a distance of its dzdt value times the sub timestep. Then, its mass will be updated. If the WF is in the top layer, the mass balance is fairly straightforward, as the mass of the WF at the start of the time step is known, as well as any relevant AET or infiltration fluxes, and given the new depth, the theta (and psi) value can be directly updated to conserve mass. In the event that this WF is not in the top layer, then its conservation of mass calculation must additionally take into account the total mass of water in layers above the current WF with the same capillary head value as the current WF. The WF's capillary head and theta values are updated such that mass is conserved in the entire region including itself and in layers above with the same psi value. The theta and capillary head values of wetting fronts above which extend to the bottom of their layers are later updated via case 1.

WF is in the top layer

prior_mass is calculated. This will be the sum of the mass of the wetting front at the start of the sub time step and any relevant AET, infiltration, or free drainage fluxes.

→ WF depth is increased via dzdt and the sub time step.

→ WF moisture is updated such that it conserves mass with prior_mass.

WF is not in the top layer

WF depth is increased via dzdt and the sub time step.

→ prior_mass and new_mass are calculated. prior_mass will be the mass for this wetting front, considering moisture in layers above with the same capillary head as the current one, in addition to any relevant AET, infiltration, or free drainage fluxes. new_mass will be the current mass of the wetting front (and relevant water in layers above with the same capillary head) not yet with an altered theta value, but with an altered depth value.

→ lgar_theta_mass_balance: this function will calculate the theta value for the WF that conserves mass, considering all relevant soil water above (which will share a capillary head with the current WF and extend to the bottom of their layers) and considering any AET, infiltration, or free drainage fluxes.

Capillary head is updated based on theta.

→ In the event that infiltration required that theta would become greater than theta_e, theta was corrected to theta_e. Therefore in this case the WF to which water was added must be made slightly deeper, or a small amount of water must be added to the boundary bottom flux, to conserve mass.

→ Now that all surface WFs have been moved, there are cases in which:
- One WF has surpassed another and the two must merge (function lgar_merge_wetting_fronts);
- A WF has exceeded the depth for its layer and must be moved to the next layer (function lgar_wetting_fronts_cross_layer_boundary);
- A WF has exceeded the depth of the model lowest boundary and its depth must be corrected (lgar_wetting_fronts_cross_domain_boundary)
- The WF from which AET is extracted has become drier than the one below it, and so it must merge with the one below (lgar_fix_dry_over_wet_wetting_fronts)

All surface WFs will be iteratively updated until none of these condirions are met.

→ All surface WFs have been moved. Next the function will move GW WFs, see next page.

lgar_move_wetting_fronts: GW WFs

note that this page will only run if TO_enabled is set to true in the config file

first, the function lgarto_resolve_TO_WF_beteen_surf_WF addresses the rare case where multiple surface WFs advance and pass a TO WF in such a way that a TO WF would occur between two surface WFs. In this case, the TO WF between surface WFs is deleted.

Because extracting AET from GW WFs usually changes their height, the AET component from each GW WF is calculated and is used to augment GW WFs. This is achieved with lgarto_calc_aet_from_TO_WFs. Most GW WFs with a nonzero AET will satisfy that demand by losing water from the top part of the WF, becoming shorter. The single exception is the TO WF with the capillary head closest to 0 that is also in contact with the soil surface when surface WFs are present and no other TO WFs are in the root zone. In this case, AET demand is extracted by making the TO WF, and all TO WFs with the same capillary head, drier. After that, lgarto_ensure_rooting_zone_population makes sure that the number of wetting fronts within the root zone is never too low by inserting new TO WFs at the soil surface if there are too few WFs in the root zone.

Next, the dzdt value for a TO WF is used to update the TO WF depth. Then, the recharge, boundary_bottom_flux_cm is updated based on TO movement. This process iterates over each TO WF, from low to high. Unlike with surface WFs, there are not three separate cases that need to be separately addressed. Because the water content of each TO WF does not change in this step, there is no need for iterative calculations to ensure conservation of mass across multiple layers. It is sufficient to simply make each TO WF shallower by a distance equal to its dzdt value multiplied by the sub time step, and subsequently increment the recharge by this distance multiplied by the change in water content between two adjacent TO WFs.

In the event that there are any TO WFs that are above surface WFs (meaning the depth of these TO WFs will be 0), their dzdt value was not used to increment their depth. Instead, contributions to groundwater from TO WFs that are above surface WFs (or thinking in terms of a soil moisture profile, to the left of surface WFs) are satisfied by taking water from surface wetting fronts, which will have a capillary head closer to 0. The function lgarto_extract_TO_GW_flux_from_surface_WFs accomplishes this.

An interesting emergent property of many LGARTO model runs is that a sudden large change in capillary head with respect to depth can occur at the bottom of the root zone when AET is large enough. In order to ensure that capillary rise resulting in negative recharge is sufficient, a TO WF is split into two when the difference in capillary head between two adjacent TO WFs is sufficiently large.

Now that TO WFs have moved, some of them might have moved in such a way that requires correction. For example, if a TO WF moved faster than the one below it such that the WF that was previously higher is now lower, merging is required. Further, corrections including surface WFs are generally required again, because extracting recharge from surface fronts changed their water contents. In total, eight corrections are checked for iteratively until no more remain, handled by the functions lgar_merge_surface_and_TO_wetting_fronts, lgarto_TO_WFs_merge_via_depth, lgarto_TO_WFs_merge_via_theta, lgar_TO_wetting_fronts_cross_layer_boundary, lgar_merge_wetting_fronts, lgar_wetting_fronts_cross_layer_boundary, lgar_wetting_front_cross_domain_boundary, and a case where the deepest surface WF becomes a TO WF if it is sufficiently dry.

the process of creating and correcting wetting fronts can sometimes yield a number of TO WFs at the top of the model domain that adds little accuracy to the model but increases computational expense. The function lgarto_consolidate_excessive_fronts ensures that there are never too many TO WFs at the top of the model domain.

Finally, it is ensured that all GW WFs above surface WFs have a depth of 0; if this condition fails the model crashes with an error message. After this point, TO WF movement and merging is complete, and the next page describes general checks or corrections that the model makes in LGAR or LGARTO mode.

lgar_move_wetting_fronts: checks and corrections after WFs have moved

First, WFs that have a depth of 0 and are drier than their hydrostatic position are deleted, with the idea that these WFs add to computational expense but do not greatly impact the simulation. This is similar in effect to lgarto_consolidate_excessive_fronts.

→ The case where dry surface WFs are above wetter surface WFs in the same layer is addressed with lgar_fix_dry_over_wet_wetting_fronts. This happened previously in the LGAR component of the code, but might be necessary again depending on how much water was extracted from surface WFs during TO WF movement.

→ Extra check to make sure all psi values have been correctly as a result of updated theta values

→ check very specific case where if there is just 1 layer and 1 WF, this WF has the depth of the entire model domain

→ In some rare cases, mass conservative WF merging yields a WF that is higher than the model domain. This is checked and corrected.

→ GW WFs that go below the lower boundary of the model domain are truncated and the resulting mass is added to recharge.

→ miscellaneous checks that prevent physically impossible scenarios, or cause the model to crash if an unfixable physically impossible scenario is detected, including:

- Causes a crash if two adjacent WFs are in layers that are physically impossible (e.g. two adjacent wetting fronts are in layers 1 and 3)
- Enforce that capillary head is the same on either side of a soil layer boundary when the WF on both sides of the soil layer boundary is a GW WF, in the rare event that this was not implemented after TO WF movement
- Causes a crash if: a wetting front depth is not a number, the volumetric water content of a WF is less than 0, the WF capillary head is not a number, or if the volumetric water content is less than theta_r or greater than theta_e for its layer
- Enforce very specific case where if there is a number of WFs exactly equal to the number of layers, then there must be exactly one capillary head value throughout the model domain
- Any WF that is below the lower domain of the model is truncated and the excess water is added to recharge. This has already happened for both TO and surface WFs, so this process might be redundant.
- Causes a crash if the GW recharge is not a number

→ end of lgar_move_weting fronts; return recharge to groundwater