BEACH Formalisms

Methodology

Hydrological Framework

Lateral/subsurface flow

Later flow occurs when the soil moisture content exceeds the field capacity and is represented as the difference between inflow and outflow at a cell j for each soil layer z according to Manfreda et al. [2005]:

$$\Delta LF_{j(t)} = \Big(\frac{W_j \sum_{i=1}^{N(t)} max[c_z(SW_i - SW_{fc,i}),\ 0]}{\sum_{i=1}^{N(t)} W_i}\Big) - max[c_z(SW_j - SW_{fc,j}),\ 0] \tag{1}$$

were c_z is the subsurface flow coefficient ($\approx 0.25\ d^{-1}$, if $dt \leq 1\ d$), SW and SW_{fc} are the soil water content at time t and at field capacity for the soil profile [mm], respectively. $N(t)$ is the number of cells i upstream exceeding field capacity and at steepest slope to cell j. The topographical wetness index (TWI) W, introduced by Beven and Kirby [1979], describes the tendency for water to accumulate spatially in regions characterized by a relatively low local slope and a large upstream drainage area. W_j is the TWI at cell j and W_i the cumulative TWI from upstream cells i. The TWI is given by:

$$TWI = ln\Big(\frac{a}{tan\beta}\Big) \tag{2}$$

were a is the drainage area per unit contour length (defined by the Local Drain Direction (LDD) network and cell area) and tan β, the local slope in radians derived from the Digital Elevation Model (DEM).

Old Plant height

$$h = \begin{cases} 0\ , & t \leq t_{sow-date} \\ \dfrac{h_{max} \cdot\ (t - t_{sow-date})}{\Delta t}, & t \leq t_{sow-date} \\ \dfrac{lnf}{-5 \cdot 10^{-5}}, & f > 0 \end{cases} \tag{3}$$

Soil Temperature

Temperture is an important parameter regulating chemical and biological processes. To simulate its evolution across time and space this model follows [Neitsch et al., 2009], where the average daily soil temperature (°C) at the center of each soil layer z is given by:

$$T_{soil_z}(t) = l \cdot T_{soil,k,z}(t-1) + (1.0 - l) \cdot \big[df \cdot [\overline{T}_{AAir} - T_{ssurf}] + T_{ssurf}\big] \tag{4}$$

where l is a lag coefficient (0 - 1.0) regulating the influence of the previous day's $(t-1)$ temperature, $\overline{T}_{AAir}$ is the average annual air temperature and T_{ssurf} is the soil surface temperature at time t. The depth factor df is given by,

$$df = \frac{zd}{zd + exp(-0.867 - 2.078 \cdot zd)} \tag{5}$$

where zd is the ratio of the depth at the center of the soil layer $D_{z/2}$ (mm) to the damping depth dd (mm) such that:

$$zd = \frac{D_{z/2}}{dd} \tag{6}$$

The damping depth dd is a function of the maximum damping depth dd_{max} (mm) and a soil water scaling factor φ (-).

$$dd = dd_{max} \cdot exp\Big[ln\Big(\frac{500}{dd_{max}}\Big) \cdot \Big(\frac{1-\varphi}{1+\varphi}\Big)^2\Big] \tag{7}$$

$$dd_{max} = 1000 + \frac{2500\rho_b}{\rho_b + 686 \cdot exp(-5.63\rho_b)} \tag{8}$$

$$\varphi = \frac{SW}{(0.356 - 0.144\rho_b) \cdot D_k} \tag{9}$$

where ρ_b is the soil bulk density ($mg\ m^{-3}$) and SW is water content ($mm\ H_2O$) in the soil profile D_z (mm).

The soil surface temperature T_{ssurf} in eq. 4, is a fucntion of the previous day's soil temperature, amount of ground cover captured by a crop factor bcv and the temperature of bare soil T_{bare}, such that:

$$T_{ssurf} = bcv \cdot T_{soil,k,t-1} + (1 - bcv) \cdot T_{bare} \tag{10}$$

$$bcv = \frac{CV}{(CV + exp(7.563 - 1.297 \cdot 10^{-4} \cdot CV))} \tag{11}$$

$$CV = \begin{cases} 0\ , & f \leq 0 \\ \dfrac{lnf}{-5 \cdot 10^{-5}}, & f > 0 \end{cases} \tag{12}$$

$$f = 1 - exp(-\mu * LAI) \tag{13}$$

where μ is light-use efficiency of the crop ($kg\ ha^{-1}\ m^2\ MJ^{-1}$).

Pesticide Fate

Pesticide distribution

The concentration of pesticide within the soil system (g/L) along each layer z is given as a function of its partitioning across the soil's gaseous, aqueous and adsorbed phases by:

$$C_{tot_z} = \theta_{gas_z}(t) C_{gas_z}(t) + \theta_z(t) C_{aq_z}(t) + \rho_{b_z}(t) C_{ads_z} \tag{14}$$

where $\theta_{gas_z}(t) = \theta_{sat_z}(t) - \theta_z(t)$, C_{gas_z} is the concentration in the gas phase (g/L), C_{aq_z} the concentration in aqueous phase (g/L) and C_{ads_z} the concentration in adsorbed phase ($g/Kg\ soil$) with the bulk soil density ρ_{b_z} in Kg/L.

Converting to mass (g) based on the cell area (A_i, m^2) and model layer depth (D_z, mm):

$$M_{tot_z} = A_i D_z \theta_{gas_z}(t) C_{gas_z}(t) + A_i D_z \theta_z(t) C_{aq_z}(t) + A_i D_z \rho_{b_z}(t) C_{ads_z} \tag{15}$$

simplifying to the respective fractions,

$$M_{tot_z} = V_{gas_z}(t) C_{gas_z}(t) + V_{H_2O}(t) C_{aq_z}(t) + M_{soil}(t) C_{ads_z} \tag{16}$$

and substituting phase concentrations for their equivalent in C_{aq} according to eq. 24 and eq. 34 yields,

$$M_{tot_z} = V_{gas_z} C_{aq_z}(t)/K_H + V_{H_2O}(t) C_{aq_z}(t) + M_{soil}(t) K_d C_{aq_z}(t) \tag{17}$$

Solving for C_{aq} ($g/L\ H_2O$) with K_d units in (L/Kg),

$$C_{aq} = \frac{M_{tot_z}}{V_{gas_z}(t)/K_H + V_{H_2O}(t) + M_{soil}(t) K_d} \tag{18}$$

$$C_{aq} = \frac{M_{tot_z}}{A_i D_z \Big(\theta_{gas_z}(t)/K_H + \theta_{aq}(t) + \rho_{b_z}(t) K_d \Big)} \tag{19}$$

and substituting the retardation factor from eq. 25,

$$C_{aq} = \frac{M_{tot_z}}{A_i D_z \Big(\theta_{gas_z}(t)/K_H + \theta_{aq}(t) R_z(t) \Big)} \tag{20}$$

Aqueous concentration is obtained with the following function. Note that the input parameter, `mass`, may be equal to the total concentration or either the heavy or light fractions, whichever output is required by the user.

```
def getConcAq(model, layer, theta_sat, mass,
              sorption_model="linear", gas=True):
    # Note that p_b (g/cm3) x k_d (L/Kg) -> unit-less
    if layer == 0:
        depth = model.z0
        theta_layer = model.theta_z0
    elif layer == 1:
        depth = model.z1
        theta_layer = model.theta_z1
    elif layer == 2:
        depth = model.z2
        theta_layer = model.theta_z2
    if sorption_model == "linear":
        # Retardation factor
        retard_layer = 1 + (model.p_b * model.k_d) / theta_layer
    else:
        print("No sorption assumed, Ret. factor = 1")
        retard_layer = 1  # No retardation.
    if gas:  # Leistra et al., 2001
        theta_gas = max(theta_sat - theta_layer, scalar(0))
        conc_aq = mass / ((cellarea() * depth) *  # m2 * mm = L
                          (theta_gas / model.k_h +
```

```
                                theta_layer * retard_layer))  # ug/L cell volume
    else:  # No gas phase
        # Whelan, 1987
        conc_aq = (mass / (cellarea() * depth * theta_layer * retard_layer))
    return conc_aq
```

An analogous re-arrangement is applied to obtained the concentration in gas C_{gas} $(g/L\ air)$ and solid (adsorbed) C_{ads} $(g/Kg\ soil)$ phases:

$$C_g = \frac{C_{aq}}{K_H} \tag{21}$$

$$C_{ads} = \frac{M_{tot_z}}{A_i \cdot D_z\Big(\theta_{gas_z}(t)/(K_H \cdot K_d) + \theta_{aq}(t)/K_d + \rho_{b_z}(t)\Big)} \tag{22}$$

To obtain the adsorbed concentration C_{ads} $(g/Kg\ soil)$, the following function is used:

```
def getConcAds(model, layer, theta_sat, mass, gas=True):
    # mass / Kg soil
    if layer == 0:
        depth = model.z0
        theta_layer = model.theta_z0
    elif layer == 1:
        depth = model.z1
        theta_layer = model.theta_z1
    elif layer == 2:
        depth = model.z2
        theta_layer = model.theta_z2
    if gas:
        theta_gas = max(theta_sat - theta_layer, scalar(0))
        # [mass pest/Kg soil]
        conc_ads = mass / ((cellarea() * depth) *
                           (theta_gas / (model.k_h * model.k_d) +
                            theta_layer / model.k_d +
                            model.p_b))
    else:
        print("No implementation without gas available")
        raise NotImplementedError
    return conc_ads
```

Sorption

Linear sorption

The partition of pesticide concentrations into the dissolved C_{aq} (g/L) and adsorbed phases C_{ads} (g/Kg) is determined by the pesticide dissociation coefficient K_d (L/Kg):

$$K_d = \frac{C_{ads}}{C_{aq}} \tag{23}$$

$$K_d = K_{oc} \cdot f_{oc} \tag{24}$$

where K_{oc} (L/Kg) is the pesticide octanol partition coefficient and f_{oc} is the fraction of organic carbon (-) in soil. The retardation factor R_z (-) is given by:

$$R_z(t) = 1 + \frac{\rho_{b_z}(t) \cdot K_d}{\theta_z(t)} \tag{25}$$

Volatilization

Pesticide volatilization follows Leistra *et al.* [2001], where a boundary air layer is conceptualized through which pesticide difusses before escaping into the atmosphere. The thickness (d_a, m) of this layer, assumed to be equivalent to the topmost soil layer's thickness or mixing layer, regulates the transport resistance (r_a, d/m) such that:

$$r_a(t) = \frac{d_a}{D_a(t)} \tag{26}$$

where D_a (m^2/d) is the diffusion coefficient in air for Metolachlor at the observed environmental temperature and adjusted relative to the reference diffusion coefficient ($D_{a,r}$, m^2/d) as:

$$D_a(t) = \Big(\frac{T(t)}{T_r}\Big)^{1.75} D_{a,r} \tag{27}$$

where T and T_r are the environmental temperature at time t and at the reference temperature at 293.15°K, respectively.

The total volatilization is given by the flux across the air layer boundary ($J_{v,air}$) and the flux across the topmost soil layer ($J_{v,soil}$) such that:

$$J_{v,air}(t) = -\frac{C_{gas,top}(t) - C_{air}(t)}{r_a} \tag{28}$$

$$J_{v,soil}(t) = -\frac{C_{gas,z_0}(t) - C_{gas,top}(t)}{r_s} \tag{29}$$

where $C_{gas,top}$ (mg/m^3) is the concentration in gas phase at the soil surface, C_{air} (mg/m^3) the concentration in air, C_{gas,z_0} (mg/m^3) the concentration in gas phase at the center of the uppermost soil layer and r_s (d/m) the diffusion resistance across the topmost soil layer and given by:

$$r_s(t) = \frac{0.5 D_z}{D_{rdiff,g}(t)} \tag{30}$$

To calculate the relative diffusion ($D_{rdiff,gas}$, m^2/d) the model provides two options. Under option 1 [Millington and Quirk, 1960],

$$D_{rdiff,gas} = \frac{D_a(t)\Big(\theta_{gas_z}(t)\Big)^a}{\Big(\theta_z(t)\Big)^b} \tag{31}$$

where Jin and Jury [1996] recommend that $a = 2$ and $b = 2/3$. Under option 2 [Currie, 1960],

$$D_{rdiff,gas} = D_a(t)\Big(a\Big)\Big(\theta_{gas_z}(t)\Big)^b \tag{32}$$

where Bakker *et al.* [1987] recommend $a = 2.5$ and $b = 3$ for moderately aggregated plough layers of loamy soils and humic sandy soils [Leistra et al., 2001].

Finally, it is assumed that flux across both layer boundaries is equivalent ($J_{v,soil} = J_{v,air}$) [Leistra et al., 2001]. Considering pesticide concentration in air to be negligble ($C_{air} = 0$), the concentration at the soil surface is:

$$C_{gas,top}(t) = \frac{r_a}{(r_a + r_s)} C_{gas,z_0(t)} \tag{33}$$

The gas concentration in the soil layer is related to the dimensionless Henry constant (K_H), where:

$$C_{gas,z_0}(t) = C_{aq,z_0}(t) K_H \tag{34}$$

Substituting eq. 33 into eq. 28 yields the mass flux lost to the atmosphere ($g/m^2 d$):

$$J_{v,air} = -\frac{C_{gas,z_0}}{(r_a + r_s)} \tag{35}$$

To obtain the pesticide mass volatilized at each time step, the following function is defined:

```
def getVolatileMass(model, temp_air, theta_sat, mass, frac,
                    rel_diff_model="option-1", sorption_model="linear",
                    gas=True, isotopes=True, ):
    # Volatilize only during peak volatilization time i.e., first 24 hrs, @Prueger2005.
    theta_layer = model.theta_z0
    theta_gas = max(theta_sat - theta_layer, scalar(0))
    # Convert to m (needed for final mass computation on cell basis)
    depth_m = model.z0 * 1 / 10 ** 3
    # Air boundary layer, assumed as 2m high
    thickness_a = scalar(1.0)  # m
    # Diffusion coefficient in air (cm^2/s); https://www.gsi-net.com
    #  D_ar (metolachlor) = 0.03609052694,  at reference Temp., in Kelvin, D_a,r)
    diff_ar = 0.03609052694 * 86400.0 * 1.0 / 10 ** 4  # m2/d
    # Diffusion coefficient adjusted to air Temp. in Kelvin, D_a
    diff_a = ((temp_air + 273.15) / 293.15) ** 1.75 * diff_ar  # m2/d
    if rel_diff_model == "option-1":
        # Millington and Quirk, 1960 (in Leistra, 2001, p.48)
        # a,b parameters: Jin and Jury, 1996 (in Leistra, 2001)
        diff_relative_gas = (diff_a * theta_gas ** 2 /
                             theta_sat ** (2 / 3))  # m2/d
    elif rel_diff_model == "option-2":
        # Currie 1960 (in Leistra, 2001)
        # a,b parameters: Baker, 1987 (in Leistra, 2001)
        diff_relative_gas = diff_a * 2.5 * theta_gas ** 3  # m2/d
    else:
        print("No appropriate relative diffusion parameter chosen")
        diff_relative_gas = diff_a  # m2/d
    # Transport resistance through air (r_a) and soil (r_s) layer
    r_a = thickness_a / diff_a  # d/m
    r_s = (0.5 * depth_m) / diff_relative_gas  # d/m
    conc_aq = getConcAq(model, 0, theta_sat, mass,
                        sorption_model=sorption_model, gas=gas, isotopes=isotopes)
    # Convert ug/L to ug/m3, as will be multiplying by cell's area in m2
```

```
    conc_aq *= 10 ** 3  # ug/L * 10^3 L/m3
    conc_gas = conc_aq / model.k_h  # ug/L air
    volat_flux = (conc_gas / (r_a + r_s)) * cellarea()  # ug/day
    return volat_flux
```

Run-off Loss

Non-uniform mixing-layer-model-runoff (nu-mlm-ro)

Multiple models are available to simulate mass transfer to overland flow. The first of these models, the *Non-uniform mixing-layer-model-runoff* (nu-mlm-ro) is adapted from Ahuja and Lehman, 1983 [see Shi et al., 2011, eq. 1 and p. 1217] and given by:

$$\frac{\partial(EDI\theta C_m)}{\partial t} = -RO\beta C_m \tag{36}$$

$$\beta = e^{(-bz)} \tag{37}$$

where the Effective Depth of Interaction (EDI) refers to the mixing layer depth (mm), θ is soil moisture (L/L), RO is run-off (mm) and C_m is concentration in the mixing layer (g/L). The paramater b is a calibration constant (assuming, $1 \geq b > 0$) and where z is the depth of the simulated top-soil layer. In this model, β accounts for an exponential decrease in the ability of overland flow to mix with soil water as depth increases.

Non-uniform mixing-layer (nu-mlm)

The adaptation above of the original model by Ahuja and Lehman replaces precipitation by RO, and thus considers the ability of rainfall water to mix with soil water instead. To test the original formulation, this second model is also made available as *Non-uniform mixing layer model* (nu-mlm).

Distributed mixing-layer model (d-mlm)

The *Distributed mixing-layer model* considered is adapted from Havis [1992], and mass transfer to overland flow based on a mass flux coefficient (K_L, mm/day). The change in mass in the overland flow (and consequently in the mixing layer) is given by:

$$\frac{\partial(h_{runoff}C_{runoff})}{\partial t} + \frac{\partial(qC_{runoff})}{\partial x} = K_L(C_m - C_{runoff}) \tag{38}$$

where h_{runoff} is overland flow height (mm), q is overland flow discharge rate per unit width (mm^2/day).

Instead of considering K_L as calibration parameter, the model defines K_L (mm/day) as the ratio of the mass flux (ϱ, mg/mm^2 day) from the soil surface to overland flow and the solute concentration difference between overland flow and the surface soil (i.e., $K_L = \varrho/(C_m - C_{runoff})$). Note that due to small concentations in overland flow, it may be assumed that $C_{runoff} \approx 0$. For laminar flow, based on Bennett and Myers [1982], K_L is given by:

$$K_L = 0.664\frac{D_w}{L}Re^{1/2}S_c^{1/3} \tag{39}$$

$$Re = \frac{\rho v L}{\mu} \tag{40}$$

$$S_c = \frac{\mu}{\rho D_w} \tag{41}$$

where the D_w is the solute diffusivity (cm^2/s), Re is the Reynolds number (-) and S_c is the Schmidt number (-). Parameters include the cell length (L, mm), the dynamic viscocity of water (μ at 25 $°C$, 8.9e-03 g/cm sec), the soil bulk density (ρ, g/cm^3) and the runoff velocity (v, mm/day) or amount of runoff generated in each cell per day.

To obtain the mass lost to run-off the following function is defined:

```
def getRunOffMass(model, theta_sat, precip, runoff_mm,
                  mass, frac,
                  transfer_model="simple-mt", sorption_model="linear",
                  gas=True, isotopes=True):
    # Aqueous concentration
    conc_aq = getConcAq(model, 0, theta_sat, mass,
                        sorption_model=sorption_model, gas=gas, isotopes=isotopes)
    if transfer_model == "simple-mt":
        mass_ro = conc_aq * runoff_mm * cellarea()
    elif transfer_model == "nu-mlm-ro":
        # non-uniform-mixing-layer-model-runoff (nu-mlm-ro)
        # Considers a decrease in effective transfer as mixing layer depth increases
        # Adapted from Ahuja and Lehman, 1983 in @Shi2011,
        # Adaptation replaces Precip by Runoff amount.
        b = 1  # [mm] Calibration constant, 1 >= b > 0 (b-ranges appear reasonable).
        # As b decreases, mass transfer increases, model.z0 in mm
        mass_ro = (runoff_mm * cellarea()) * exp(-b * model.z0) * conc_aq
    elif transfer_model == "nu-mlm":
        # non-uniform-mixing-layer-model (nu-mlm)
        # Original from Ahuja and Lehman, 1983 in @Shi2011
        b = 1  # [mm] Calibration constant, 1 >= b > 0 (b-ranges appear reasonable).
        # As b decreases, mass transfer increases, model.z0 in mm
        mass_ro = (precip * cellarea()) * exp(-b * model.z0) * conc_aq
    elif transfer_model == "d-mlm":
        # distributed mixing-layer-model (d-mlm)
        # Adapted from Havis et al., 1992, and
        # taking the K_L definition for laminar flow from Bennett and Myers, 1982.
        mass_ro = getKfilm(model, runoff_mm) * cellarea() * conc_aq
    else:
        print("Run-off transfer model not stated")
        return None
    return mass_ro
```

To obtain K_L the following function is defined and implemented when `transfer_model = 'd-mlm'` is declared in `getRunOffMass()` above:

```
def getKfilm(model, runoffvelocity):
    """
    Note: Model uses run-off (mm) per day (i.e. timestep) as runoff velocity.
    Chemical parameter source:
    http://www.gsi-net.com/en/publications/gsi-chemical-database/single/377.html
    """
    # Dynamic viscosity of water (\mu) @25 Celsius = 8.9e-04 [Pa s]
    #   1 Pa = 1 N/(m s^2) = 1 Kg/(m s^2)
    #   Convert to g/(cm s): dyn_visc = 8.9e-03 [g/cm s]
```

```
    dyn_visc = 8.9e-03  # [g/cm s] @25 degrees, [@Shi2011]:\mu
    # Solute diffusivity in water (D_w)
    # Metolachlor = 5.0967719112e-006 (cm2 / s)
    diff_solute = 5.0967719112e-006  # [cm2 / s], [@Shi2011]:D_w
    Sc = dyn_visc / (model.p_b * diff_solute)  # (-) Schmidt number, [@Shi2011]:S_c
    # Reynolds number (dimensionless), 86400s = 1 day
    cell_length = 2 * 10 ** 3  # mm
    # Reynolds (Re), [-] (Shi et al., 2011)
    re = (model.p_b * 1 / 10 ** 2 * runoffvelocity * cell_length) / (dyn_visc * 86400)
    kl = (0.664 * ((diff_solute * 86400 * 10 ** 2) / cell_length) *
          re ** (float(1) / 2) * Sc ** (float(1) / 3))
    return kl  # mm/day
```

Vertical mass flux (i.e., leaching)

Vertical flux can be computed differently across soil layers. Under the first approach, and only for the uppermost layer, the model follows McGrath [2008]:

$$C_{z_0,aq}(t+1) = C_{z_0,aq}(t) exp\Big(\frac{-P(t)}{\theta_{z_0}(t) \cdot R_{z_0}(t) \cdot D_{z_0}}\Big) \tag{42}$$

The mass leached (g) is thus given by:

$$M_{z_0,lch}(t) = D_{z_0} \cdot A_i \Big(\theta_{z_0}(t) C_{z_0,aq}(t) - \theta_{z_0}(t+1) C_{z_0,aq}(t+1)\Big) \tag{43}$$

where A is the area (m^2) for each cell i.

Under the second approach, available on all layers, mass leached is proportional to the aqueous concentration in percolated water such that:

$$M_{z,lch}(t) = DP_z(t) \cdot C_{z,aq}(t) \cdot A_i \tag{44}$$

```
def getLeachedMass(model, layer, theta_sat,
                   water_flux,
                   theta_after_percolate,
                   mass,
                   sorption_model=None,
                   leach_model=None, gas=True, isotopes=True):
    if layer == 0:
        depth = model.z0
        theta_layer = model.theta_z0
    elif layer == 1:
        depth = model.z1
        theta_layer = model.theta_z1
    elif layer == 2:
        depth = model.z2
        theta_layer = model.theta_z2
    # Aqueous concentration
    conc_aq = getConcAq(model, layer, theta_sat, mass,
                        sorption_model=sorption_model, gas=gas, isotopes=isotopes)
    if sorption_model == "linear":
```

```
        # Retardation factor
        retard_layer = scalar(1) + (model.p_b * model.k_d) / theta_layer
    else:
        print("No sorption assumed, Ret. factor = 1")
        retard_layer = scalar(1)  # No retardation.
    if leach_model == "mcgrath":
        if layer == 0:
            conc_aq_new = conc_aq * exp(-water_flux / (theta_layer * retard_layer * depth))
            mass_aq = conc_aq * (theta_layer * depth * cellarea())
            mass_aq_new = conc_aq_new * (theta_after_percolate * depth * cellarea())
            mass_leached = mass_aq - mass_aq_new
            if mapminimum(mass_aq_new) < 0:
                print("Error in Leached Model")
        else:
            # McGrath not used in lower layers,
            # as formulation accounts for rainfall impact
            mass_leached = conc_aq * water_flux * cellarea()
            mass_aq = conc_aq * (theta_layer * depth * cellarea())
            mass_aq_new = mass_aq - mass_leached
            if mapminimum(mass_aq_new) < 0:
                print("Error in Leached Model")
    else:
        mass_leached = conc_aq * water_flux * cellarea()
        mass_aq = conc_aq * (theta_layer * depth) * cellarea()
        mass_aq_new = mass_aq - mass_leached
        if mapminimum(mass_aq_new) < 0:
            print("Error in Leached Model")
    return mass_leached
```

Lateral mass flux

Later mass flux is proportional to the aqueous concentration in lateral water flow. It is obtained by adapting eq. 1, where the difference between mass loss in upstream cells i and mass loss at cell j is:

$$\Delta LMF_{j,z} = \sum_{i=1}^{N(t)} M_{loss,i(t)} - M_{loss,j(t)} \tag{45}$$

The mass gain at cell j is given by the mass loss from upstream cells i contributing to downstream cells and given by:

$$\sum_{i=1}^{N(t)} M_{loss,i(t)} = \frac{W_j \sum_{i=1}^{N(t)} max[C_{i,aq} \cdot \Big(c_z(SW_i - SW_{fc,i})\Big),\ 0]}{\sum_{i=1}^{N(t)} W_i} \tag{46}$$

Loss at cell j is then given by:

$$M_{loss,j(t)} = C_{j,aq} \cdot \Big(c_z(SW_j - SW_{fc,j})\Big) \tag{47}$$

Simplfying for the relative wetness index at cell j in eq. 46, we obtain:

$$W_{j/i} = \frac{W_j}{\sum_{i=1}^{N(t)} W_i} \tag{48}$$

```
def getLatMassFlux(model, layer, theta_sat, theta_fcap,
                   mass, sorption_model='linear', gas=True, isotopes=True):
    if layer == 0:
        depth = model.z0
        theta_layer = model.theta_z0
        c = model.c1
    elif layer == 1:
        depth = model.z1
        theta_layer = model.theta_z1
        c = model.c1
    elif layer == 2:
        depth = model.z2
        theta_layer = model.theta_z2
        c = model.c2
    # Aqueous concentration
    conc_aq = getConcAq(model, layer, theta_sat, mass,
                        sorption_model=sorption_model, gas=gas, isotopes=isotopes)
    # W(j/i)
    rel_wetness = model.wetness / accuflux(model.ldd_subs, model.wetness)
    # Cell mass loss/gain (to update only mass)
    mass_loss = max(conc_aq * (c * (depth * theta_layer - depth * theta_fcap)), scalar(0))
    mass_gain = rel_wetness * accuflux(model.ldd_subs, mass_loss)
    net_mass_latflux = mass_gain - mass_loss
    return {
        'mass_loss': mass_loss,
        'mass_gain': mass_gain,
        'net_mass_latflux': net_mass_latflux
    }
```

Degradation

Biodegradation

Biodegradation in the soil is assumed to occur only on the dissolved and sorbed fractions. Assuming a first-order rate law, the change in mass due to degradation is given by:

$$\frac{dM_{tot_z}}{dt} = -k_{b,aq}(V_{H_2O_z(t)}C_{aq_z}(t)) - k_{b,ads}(M_{soil_z}C_{ads_z}(t)) \tag{49}$$

where k_b (d^{-1}) is the biodegradation rate constant in the aquoeous (aq) and sorbed soil sites (ads). Considering degradation on each phase respectively equation 49 can subdivided and written in integrated for concentrations as:

$$C_{aq_z}(t+1) = C_{aq_z}e^{-k_{b,aq}\Delta t} \tag{50}$$

$$C_{ads_z}(t+1) = C_{ads_z}e^{-k_{b,ads}\Delta t} \tag{51}$$

where C_{aq} and C_{ads} are the contamimant concentrations in liquid and sorbed sites, respectively. Updating for total contaminant mass for all fractions:

$$M_{tot_z}(t+1) = V_{gas_z}(t)C_{gas_z}(t) + V_{H_2O}(t)C_{aq_z}(t+1) + M_{soil}(t)C_{ads_z}(t+1) \tag{52}$$

The biodegradation rate constant for the aqueous phase $k_{b,aq}$ is given by:

$$k_{b,aq} = \frac{ln(2)}{t_{\frac{1}{2}}} \tag{53}$$

where $t_{\frac{1}{2}}$ is the reference half-life (d) of the contaminant considered. The degradation in the sorbed fraction is assumed to be a fraction of that in the aqueous phase such that:

$$k_{b,ads} = s_f \cdot k_{b,aq} \tag{54}$$

where $0 \leq s_f \leq 1$ $(-)$ is an adjustment parameter for degradation rate. Although a decrease in degradation rates with increasing depth reltive to top soils may be associated to a decrease in microbial activity **(Rodríguez-Cruz et al., 2006; Si et al., 2009)**, currently no depth variation is considered. However, this could be easily implemented by assuming an exponential decrease in the degradation rate constant as a function of depth such that:

$$k_{b,aq_z} = k_{b,aq} \cdot exp(-\gamma \cdot D_{z/2}) \tag{55}$$

where γ $(-)$ is a calibration parameter $0 \leq \gamma \leq 1$ and $D_{z/2}$ (m) the depth to center of layer z.

Adapting the reference half-life to temperature and moisture changes the half-life becomes:

$$t_{\frac{1}{2}} = t_{\frac{1}{2}}^{ref} \cdot F_T \cdot F_\theta \tag{56}$$

where $t_{\frac{1}{2}}^{ref}$ is the half-life (days) at the reference moisture and temperature. F_T and F_θ are factor changes in degradation rates associated to temperature and moisture conditions, following Schroll et al. [2006] and the Arrhenius equation, respectively. To account for the influence of water content across a range of saturation conditions F_θ is given by:

$$F_\theta = \begin{cases} 0\ , & \theta_t \leq 0.5 \cdot \theta_{wp} \\ \left(\dfrac{\theta_t - 0.5 \cdot \theta_{wp}}{\theta_{fc} - \theta_{wp}}\right)^{\beta_\theta}, & \theta_t < \theta_{fc} \\ 1\ , & \theta_t > \theta_{fc} \end{cases} \tag{57}$$

$$F_T = e^{(E_a/R)(1/T_{ref} - 1/T)} \tag{58}$$

where T is the temperature in soil converted to Kelvin. T_{ref} is the reference temperature at which the experimental half-life is reported (i.e. typically 20 °C). E_a is the activation energy generally equal to 54,000 (KJ mol^{-1}) and R is the gas constant 8.314 (J mol^{-1} K^{-1}).

To compute mass degradation, the following method is used:

```
def getMassDegradation(model, layer,
                       theta_sat, theta_fcap, theta_wp,
                       mass, frac="L", sor_deg_factor=1,
                       sorption_model="linear", gas=True):
    # Get mass applied if top-soil
    if layer == 0:
        theta_aq_layer = model.theta_z0
        temp_layer = model.temp_z0_fin
        depth = model.z0
    elif layer == 1:
        theta_aq_layer = model.theta_z1
        temp_layer = model.temp_z1_fin
        depth = model.z1
    elif layer == 2:
        theta_aq_layer = model.theta_z2
        temp_layer = model.temp_z2_fin
        depth = model.z2
    # F_Theta_1
    theta_factor = ifthenelse(
        theta_aq_layer <= 0.5 * theta_wp, scalar(0),
        ifthenelse(theta_aq_layer <= theta_fcap,
                   (((theta_aq_layer - 0.5 * theta_wp) / (
                       theta_fcap - theta_wp)) ** scalar(model.beta_moisture)),
                   scalar(1)))
    # Temperature factor in biodegradation
    # F_T_1
    temp_factor = exp((model.act_e / model.r_gas) *
                      (1 / (model.temp_ref + 273.15) - 1 / (model.temp_air + 273.15)))
    # Half-life as a function of temperature and moisture
    dt_50 = max(model.dt_50_ref * theta_factor * temp_factor, scalar(0))
    # dt_50 = max(model.dt_50_ref)
    # Convert to degradation constant
    # Deg in dissolved phase
    k_b = ifthenelse(dt_50 > 0, ln(2) / dt_50,
                     scalar(0))
    # Deg in sorbed phase (now assumed equal)
    k_bs = k_b * sor_deg_factor
    # Step 0 - Obtain species concentration (all phases)
    conc_aq = getConcAq(model, layer, theta_sat, mass,
                        sorption_model=sorption_model, gas=gas)  # mass/L
    conc_ads = getConcAds(model, layer, theta_sat, mass, gas=gas)  # mass/g soil
    conc_gas = conc_aq / model.k_h
    mass_aq = conc_aq * (theta_aq_layer * depth * cellarea())
    mass_ads = conc_ads * (depth * cellarea() * model.p_b)  # pb = g/cm3
    # Step 1 - Degrade phase fractions
    # First order degradation kinetics
    theta_gas = max(theta_sat - theta_aq_layer, scalar(0))
    if frac == "H":
        conc_aq_new = conc_aq * exp(-1 * model.alpha_iso * k_b * scalar(model.jd_dt))
        conc_ads_new = conc_ads * exp(-1 * model.alpha_iso * k_bs * scalar(model.jd_dt))
    else:  # Same for total conc as for "L"
        conc_aq_new = conc_aq * exp(-1 * k_b * scalar(model.jd_dt))
        conc_ads_new = conc_ads * exp(-1 * k_bs * scalar(model.jd_dt))
```

```
    # Step 2 - Convert to mass (i.e., after degradation in each phase)
    mass_aq_new = conc_aq_new * (theta_aq_layer * depth * cellarea())
    mass_ads_new = conc_ads_new * (model.p_b * depth * cellarea())  # pb = g/cm3
    mass_gas = conc_gas * (theta_gas * depth * cellarea())
    mass_tot_new = mass_aq_new + mass_ads_new + mass_gas
    mass_deg_aq = mass_aq - mass_aq_new
    mass_deg_ads = mass_ads - mass_ads_new
    return {"mass_tot_new": mass_tot_new,
            "mass_deg_aq": mass_deg_aq,
            "mass_deg_ads": mass_deg_ads}
```

Isotope treatment

To obtain the masses of heavy M^h and light isotopes M^l, the isotope signature $\delta^{13}C$ is used where:

$$\delta^{13}C[\check{\mathrm{G}}] = \left(\frac{R_{smp} - R_{std}}{R_{std}}\right) \cdot 1000 \tag{59}$$

$$R_{smp} = \frac{M^h}{M^l} \tag{60}$$

$$R_{std} = 11237.2 \cdot 10^{-6} \tag{61}$$

and where the total mass $M_{tot} = M^h + M^l$, we obtain:

$$R_{std}\Big(\delta^{13}C/1000 + 1\Big) = \frac{M_{tot} - M^l}{M^l} \tag{62}$$

$$M^l = \frac{M_{tot}}{1 + R_{std}(\delta^{13}C/1000 + 1)} \tag{63}$$

$$M^h = M_{tot} - M^l \tag{64}$$

#{r test-child, child = 'Bioavailability.Rmd'} #

References

J.W. Bakker, F.R. Boone, and P. Boekel. Diffusie van gassen in grond en zuurstofdiffusiecoëfficiënten in Nederlandse akkerbouwgronden (Diffusion of gases in soil and oxygen diffusion coefficients in Dutch arable soils). Rapport 20. Technical report, ICW, Wageningen, The Netherlands, 1987.

C.O. Bennet and J.E. Myers. *Momentum, heat and mass transfer*. McGraw-Hill, New York, 3rd ed. edition, 1982.

K. J. Beven and M. J. Kirby. A physically based, variable contributing area model of basin hydrology. *Hydrological Sciences Bulletin*, 24(1):43–69, 1979. ISSN 0303-6936. doi: 10.1080/02626667909491834. URL http://www.tandfonline.com/doi/abs/10.1080/02626667909491834.

J.A. Currie. Gaseous diffusion in porous media. 2. Dry granular materials. *British J. Applied Physics*, 11: 318–324, 1960.

R N Havis, R E Smith, and D D Adrian. Partitioning solute transport between infiltration and overland flow under rainfall. *Water Resources Research*, 28(10):2569–2580, 1992. ISSN 0043-1397. doi: 10.1029/92WR01366.

Y. Jin and W. A. Jury. Characterizing the Dependence of Gas Diffusion Coefficient on Soil Properties. *Soil Sci. Soc. Am. J.*, 60:66–71, 1996. doi: 10.2136/sssaj1996.03615995006000010012x.

M. Leistra, A. M. A. van der Linden, J. J. T. I. Boesten, A. Tiktak, and F. van den Berg. PEARL model for pesticide behaviour and emissions in soil-plant systems; Description of the processes in FOCUS PEARL v 1.1.1. Technical report, 2001.

S. Manfreda, M. Fiorentino, and V. Iacobellis. DREAM: a distributed model for runoff, evapotranspiration, and antecedent soil moisture simulation. *Advances in Geosciences*, 2(August 2016):31–39, 2005. ISSN 1680-7359. doi: 10.5194/adgeo-2-31-2005.

Gavan S Mcgrath, Christoph Hinz, and Murugesu Sivapalan. Modeling the effect of rainfall intermittency on the variability of solute persistence at the soil surface. *Water Resources Research*, 44:1–10, 2008. doi: 10.1029/2007WR006652.

R.J. Millington and J.P. Quirk. Transport in porous media. *Transactions 7th Int. Congress Soil Sci.*, 1: 97–106, 1960.

S L Neitsch, J G Arnold, J R Kiniry, and J R Williams. Soil & Water Assessment Tool - Theoretical Documentation Version 2009. 2009. ISSN 2151-0040.

Reiner Schroll, Hans Heinrich Becher, Ulrike Dörfler, Sebastian Gayler, Sabine Grundmann, Hans Peter Hartmann, and Jürgen Ruoss. Quantifying the effect of soil moisture on the aerobic microbial mineralization of selected pesticides in different soils. *Environmental Science and Technology*, 40(10):3305–3312, 2006. ISSN 0013936X. doi: 10.1021/es052205j.

X N Shi, L S Wu, W P Chen, and Q J Wang. Solute Transfer from the Soil Surface to Overland Flow: A Review. *Soil Science Society of America Journal*, 75(4):1214–1225, 2011. ISSN 0361-5995. doi: DOI10.2136/sssaj2010.0433.