ddpart

Contents

Introduction

This package is an implementation of the particle dry deposition model of Zhang et al. (2001) including the parametrization from Emerson et al. (2020). This vignette contains an example application of the package and validation against the original publication (Emerson et al. (2020)).

Usually, the wrapper functions CalculateDepositionVelocity() or CalculateDepositionVelocity2() would be used. These functions include the calculation of atmospheric stability conditions (Monin-Obukhov-Length) and friction velocity via the Pasquill class approach. This has the advantage that widely available meteorological measurements (or modelled data) can be used as an input (e.g. wind speed, solar radiation, etc.). If this is not required because direct data on friction velocity is available, the underlying functions to calculate the resistance components can be called directly to skip the Pasquill class approach. See section "Validation" for examples where this is the case.

Package installation and dependencies

Install the package:

```
devtools::install_github(
  repo = "https://github.com/AndSchmitz/ddpart"
)
library(ddpart)
```

Dependencies for this vignette:

```
library(tidyr)
library(knitr)
library(scales)
library(ggplot2)
```

Usage example

The following example shows how to use the package to calculate the dry deposition velocity of particles with 5 um diameter to grassland for a 48 h time series of meteorological data.

Load and show example meteo time series

```
meteo_time_series_long <- meteo_time_series %>%
  pivot_longer(
    cols = -c(TimeStampUTC)
  )

ggplot(
  data = meteo_time_series_long,
  mapping = aes(
    x = TimeStampUTC,
    y = value
  )
) +
  geom_line() +
  facet_wrap(~name, scales = "free_y", ncol = 3) +
  scale_x_datetime(guide = guide_axis(n.dodge = 2))
```

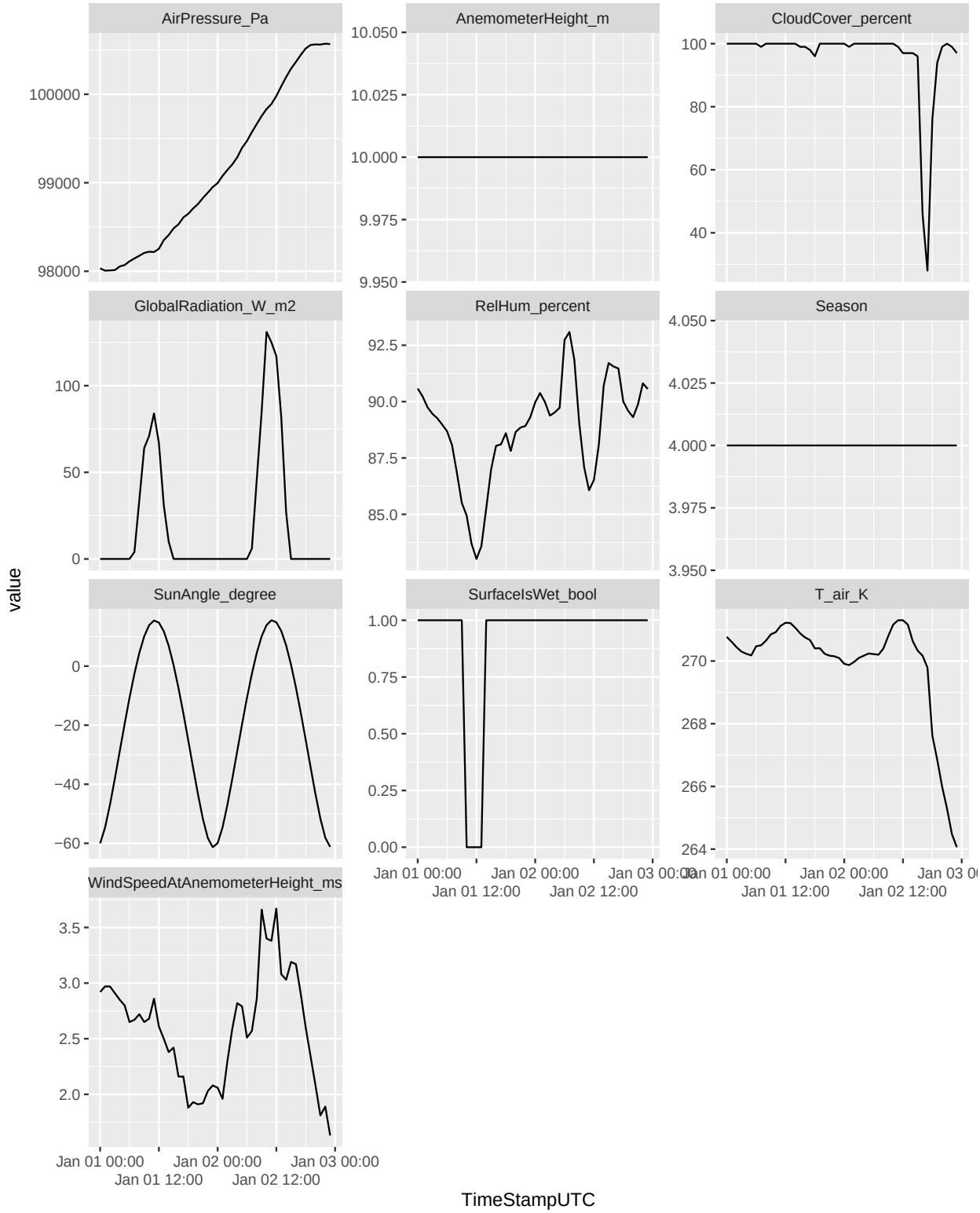

Figure 1: Example time series of meteorological data.

Calculate dry deposition velocity

In the following code block, the function CalculateDepositionVelocity() is used. This function requires a dataframe as input ("InputTable"). For each row of InputTable, a deposition velocity is calculated. The InputTable must contain all information required to calculate aerodynamic resistance and surface resistance (e.g. roughness length, zero plane distance height, wind speed and anemomemter height, hight of measured (or modelled) concentration data (=reference height), etc.). See ?CalculateDepositionVelocity for a description.

The function CalculateDepositionVelocity() is designed for the usual setup where both, wind speed measurements and concentrations measurements, are available over the same land use (e.g. both measured over grassland). For other cases, see ?CalculateDepositionVelocity2. For example, if wind speed data refers to grassland conditions (e.g. from meteorological models) but the dry deposition velocity should be calculated for other land uses (e.g. forest).

```
InputTable <- meteo_time_series %>%
  mutate(
    LUCZhang2001 = 6, # grassland
    RoughnessLength_m = 0.03,
    ZeroPlaneDisplacementHeight_m = 7 * RoughnessLength_m,
    ReferenceHeight_m = 20, # Height of concentration measurements
    DryParticleDiameter_m = 5 * 1e-6, # 5 um particles
    ParticleDensity_kgm3 = 2000,
    AerosolType = "SeaSalt", #Relevant for hygroscopic swelling
    Parametrization = "Emerson20"
  )


Results <- CalculateDepositionVelocity(
  InputTable = InputTable
)


ResultsLong <- Results %>%
  mutate(
    DryDepositionVelocity_cms = V_d_RefHeight_ms * 100
  ) %>%
  select(TimeStampUTC, DryDepositionVelocity_cms) %>%
  pivot_longer(
    cols = -c(TimeStampUTC)
  )

ggplot(
  data = ResultsLong,
  mapping = aes(
    x = TimeStampUTC,
    y = value
  )
) +
  geom_line() +
  facet_wrap(~name, scales = "free_y", nrow = 3) +
  scale_x_datetime(guide = guide_axis(n.dodge = 2))
```

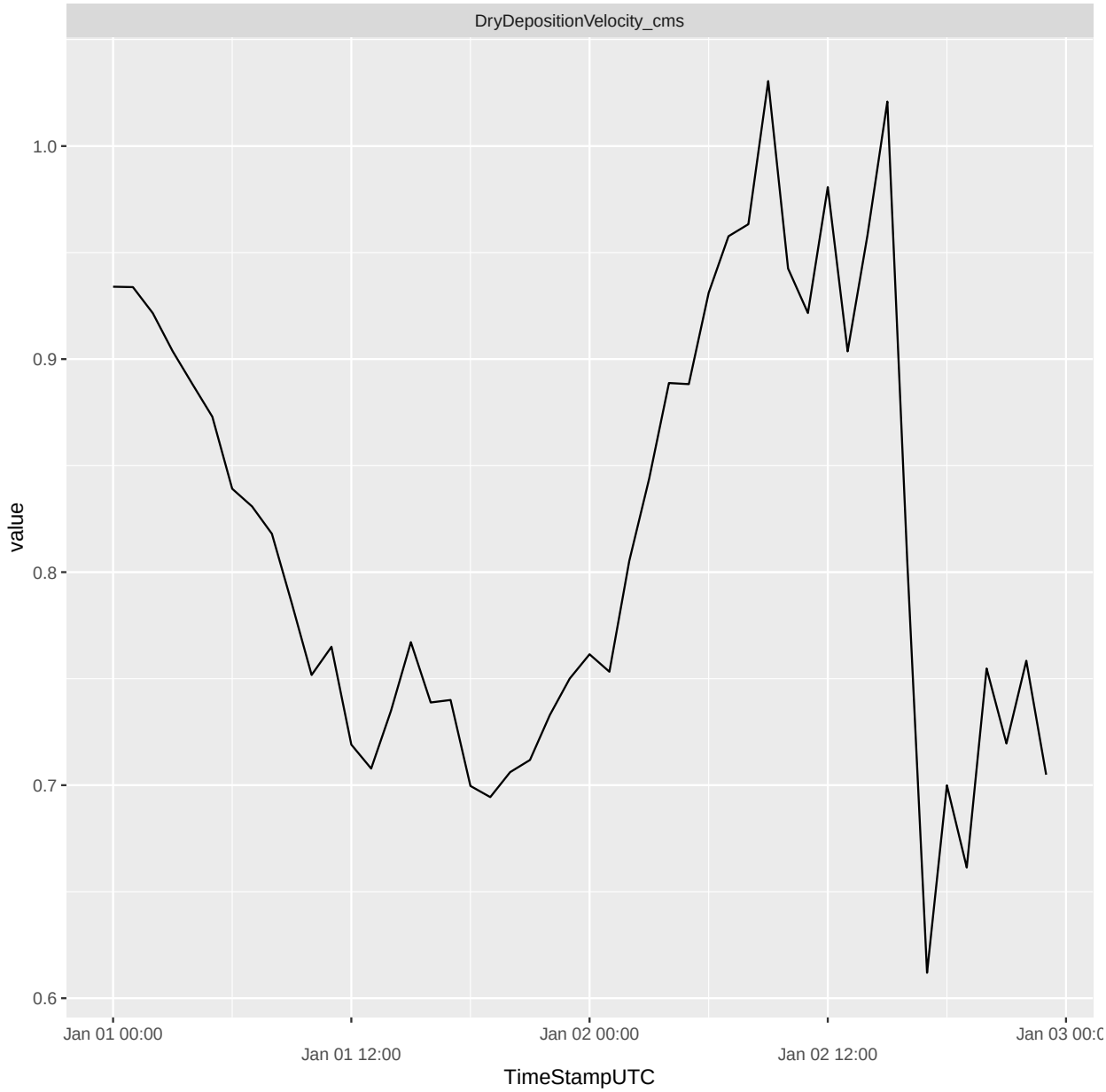

Figure 2: Dry deposition velocity calculated for the 48 hours of input data.

Validation

Emerson et al. (2020) figures 1 and 2 show the dry deposition velocity as a a function of particle diameter for different settings (i.e. for different land use types, friction velocity, etc.). In the following, these figures are reproduced as a validation of the functions in the ddpart package.

1. Dry deposition velocity vs. particle size

Emerson et al. (2020) figure 1 shows the dry deposition velocity for different land use classes.

Definition of parameters

```
# Vector of particle diameters
d_p_vec <- 10^seq(-2, 2, length.out = 5)
tmp <- expand.grid(1:9, d_p_vec)
d_p_vec <- tmp[, 1] * tmp[, 2]
d_p_vec <- d_p_vec[d_p_vec <= 100]
d_p_vec <- d_p_vec * 1e-6

InputPars <- data.frame(
  R_a_sm = 0,
  Parametrization = "Emerson20",
  LUC = 1,
  Season = 1,
  ParticleDensity_kgm3 = 1200,
  FrictionVelocity_ms = 0.2,
  AerosolType = "Rural",
  RelHum_percent = 85,
  # These parameters are guessed
  # Standard temperature and pressure
  T_air_K = 293.15,
  AirPressure_Pa = 101325,
  SurfaceIsWet_bool = TRUE
)

# Duplicate input for Zhang01 parametrization
tmp <- InputPars %>%
  mutate(
    Parametrization = "Zhang01"
  )
InputPars <- bind_rows(InputPars, tmp)

# Duplicate input for broadleaf and grassland
tmp1 <- InputPars %>%
  mutate(
    LUC = 2
  )
tmp2 <- InputPars %>%
  mutate(
    LUC = 6
  )
InputPars <- bind_rows(InputPars, tmp1, tmp2)

Input <- expand_grid(
```

```
  InputPars,
  DryParticleDiameter_m = d_p_vec
) %>%
  mutate(
    ParticleDiameter_m = CalculateHygroscopicSwelling(
      DryParticleDiameter_m = DryParticleDiameter_m,
      RelHum_percent = RelHum_percent,
      AerosolType = AerosolType
    )
  )
```

Calculations

```
Output <- Input %>%
  mutate(
    # Meteo basics
    DynamicViscosityAir_kgms = CalculateDynamicViscosityOfAir(
      T_air_K = T_air_K
    ),
    AirDensity_kgm3 = CalculateAirDensity(
      AirPressure_Pa,
      T_air_K
    ),
    KinematicViscosityOfAir_m2s = CalculateKinematicViscosityOfAir(
      DynamicViscosityAir_kgms = DynamicViscosityAir_kgms,
      AirDensity_kgm3 = AirDensity_kgm3
    ),
    MeanFreePathOfAirMolecule_m = CalculateMeanFreePath(
      T_air_K = T_air_K,
      AirPressure_Pa = AirPressure_Pa,
      DynamicViscosityAir_kgms = DynamicViscosityAir_kgms
    ),
    # Deposition processes------
    SettlingVelocity_ms = CalculateSettlingVelocity(
      ParticleDensity_kgm3 = ParticleDensity_kgm3,
      ParticleDiameter_m = ParticleDiameter_m,
      MeanFreePathOfAirMolecule_m = MeanFreePathOfAirMolecule_m,
      DynamicViscosityAir_kgms = DynamicViscosityAir_kgms
    ),
    # _Schmidt number------
    SchmidtNumber = CalculateSchmidtNumber(
      DynamicViscosityAir_kgms = DynamicViscosityAir_kgms,
      KinematicViscosityOfAir_m2s = KinematicViscosityOfAir_m2s,
      T_air_K = T_air_K,
      ParticleDiameter_m = ParticleDiameter_m
    ),
    # _E_b-----
    # Loss efficiency by Brownian diffusion
    BrownianDiffusionParameterGamma = GetLandUseParameters(
      LUCs = LUC,
      Seasons = Season,
      TargetPar = "gamma",
      Parametrization = Parametrization
```

```
  ),
  E_b = CalculateLossEfficiencyBrownianDiffusion(
    SchmidtNumber = SchmidtNumber,
    BrownianDiffusionParameterGamma = BrownianDiffusionParameterGamma,
    Parametrization = Parametrization
  ),
  # _Stokes number-----
  SurfaceIsVegetated = GetLandUseParameters(
    LUCs = LUC,
    Seasons = Season,
    TargetPar = "SurfaceIsVegetated",
    Parametrization = Parametrization
  ),
  CharacteristicRadius_m = GetLandUseParameters(
    LUCs = LUC,
    Seasons = Season,
    TargetPar = "A_mm",
    Parametrization = Parametrization
  ) * 1e-3,
  StokesNumber = CalculateStokesNumber(
    FrictionVelocity_ms = FrictionVelocity_ms,
    SettlingVelocity_ms = SettlingVelocity_ms,
    CharacteristicRadius_m = CharacteristicRadius_m,
    KinematicViscosityOfAir_m2s = KinematicViscosityOfAir_m2s,
    SurfaceIsVegetated = SurfaceIsVegetated
  ),
  # _E_Im----
  # Loss efficiency by impaction
  ImpactionParameterAlpha = GetLandUseParameters(
    LUCs = LUC,
    Seasons = Season,
    TargetPar = "alpha",
    Parametrization = Parametrization
  ),
  E_Im = CalculateLossEfficiencyImpaction(
    StokesNumber = StokesNumber,
    ImpactionParameterAlpha = ImpactionParameterAlpha,
    Parametrization = Parametrization
  ),
  # _E_In----
  # Loss efficiency by interception
  E_In = CalculateLossEfficiencyInterception(
    ParticleDiameter_m = ParticleDiameter_m,
    CharacteristicRadius_m = CharacteristicRadius_m,
    Parametrization = Parametrization
  ),
  # _R_s-----
  # Surface resistance
  R_s_sm = CalculateSurfaceResistance(
    SurfaceIsWet = SurfaceIsWet_bool,
    FrictionVelocity_ms = FrictionVelocity_ms,
    StokesNumber = StokesNumber,
    E_b = E_b,
```

```
    E_Im = E_Im,
    E_In = E_In,
    Parametrization = Parametrization
  ),
  # _V_d---------
  V_d_s = 1 / (R_a_sm + R_s_sm),
  V_g = SettlingVelocity_ms,
  V_d_ms = V_d_s + V_g,
  Type = "Results from ddpart"
)
```

Summary of parameters

```
TableData <- Output %>%
  select(
    Parametrization, LUC, Season,
    T_air_K, AirPressure_Pa, SurfaceIsWet_bool,
    SurfaceIsVegetated, CharacteristicRadius_m,
    ImpactionParameterAlpha, ParticleDensity_kgm3,
    BrownianDiffusionParameterGamma,
    R_a_sm
  ) %>%
  distinct() %>%
  rename(
    Gamma = BrownianDiffusionParameterGamma
  ) %>%
  mutate(
    Gamma = round(Gamma, 3),
    C_b = GetParameters(Parametrization = Parametrization, TargetParameter = "C_b"),
    nu = GetParameters(Parametrization = Parametrization, TargetParameter = "nu"),
    C_In = GetParameters(Parametrization = Parametrization, TargetParameter = "C_In"),
    beta = GetParameters(Parametrization = Parametrization, TargetParameter = "beta"),
    C_Im = GetParameters(Parametrization = Parametrization, TargetParameter = "C_Im")
  ) %>%
  pivot_longer(
    cols = -c("Parametrization", "LUC"),
    values_transform = as.character,
    names_to = "Parameter"
  ) %>%
  mutate(
    ParametrizationShort = case_when(
      Parametrization == "Emerson20" ~ "E20",
      Parametrization == "Zhang01" ~ "Z01",
      T ~ "ERROR"
    ),
    Parametrization_LUC = paste0(ParametrizationShort, "_LUC", LUC)
  ) %>%
  select(-Parametrization, -ParametrizationShort, -LUC) %>%
  pivot_wider(
    names_from = "Parametrization_LUC",
    values_from = "value"
  )

kable(
  x = TableData
)
```

Parameter	E20_LUC1	Z01_LUC1	E20_LUC2	Z01_LUC2	E20_LUC6	Z01_LUC6
Season	1	1	1	1	1	1
T_air_K	293.15	293.15	293.15	293.15	293.15	293.15
AirPressure_Pa	101325	101325	101325	101325	101325	101325
SurfaceIsWet_bool	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
SurfaceIsVegetated	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
CharacteristicRadius_m	0.002	0.002	0.005	0.005	0.002	0.002

Parameter	E20_LUC1	Z01_LUC1	E20_LUC2	Z01_LUC2	E20_LUC6	Z01_LUC6
ImpactionParameterAlpha	1	1	0.6	0.6	1.2	1.2
ParticleDensity_kgm3	1200	1200	1200	1200	1200	1200
Gamma	0.667	0.56	0.667	0.58	0.667	0.54
R_a_sm	0	0	0	0	0	0
C_b	0.2	1	0.2	1	0.2	1
nu	0.8	2	0.8	2	0.8	2
C_In	2.5	0.5	2.5	0.5	2.5	0.5
beta	1.7	2	1.7	2	1.7	2
C_Im	0.4	1	0.4	1	0.4	1

Plots

```
# Load data extracted from Emerson et al. (2020)
ValidationData <- dd_validation %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("Zhang01", "Emerson20")
    ),
    Type = "Extracted from Emerson et al. (2020)",
    LUCLabel = case_when(
      LUC == 1 ~ "Needleleaf forest",
      LUC == 2 ~ "Broadlead forest",
      LUC == 6 ~ "Grassland",
      T ~ "ERROR"
    ),
    LUCLabel = factor(
      x = LUCLabel,
      levels = c("Needleleaf forest", "Broadlead forest", "Grassland")
    )
  )
Output <- Output %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("Zhang01", "Emerson20")
    ),
    Type = "Calculated with ddpart",
    LUCLabel = case_when(
      LUC == 1 ~ "Needleleaf forest",
      LUC == 2 ~ "Broadlead forest",
      LUC == 6 ~ "Grassland",
      T ~ "ERROR"
    ),
    LUCLabel = factor(
      x = LUCLabel,
      levels = c("Needleleaf forest", "Broadlead forest", "Grassland")
    )
  )

Shapes <- c(3)
names(Shapes) <- c("Calculated with ddpart")
LineTypes <- c("solid")
names(LineTypes) <- c("Extracted from Emerson et al. (2020)")

ggplot() +
  geom_line(
    data = ValidationData,
    mapping = aes(
      x = ParticleDiameter_um,
      y = DepositionVelocity_cms,
      color = Parametrization,
      linetype = Type
    )
```

```
  ) +
  geom_point(
    data = Output,
    mapping = aes(
      x = DryParticleDiameter_m * 1e6,
      y = V_d_ms * 100,
      color = Parametrization,
      shape = Type
    ),
    size = 2
  ) +
  scale_shape_manual(values = Shapes) +
  scale_linetype_manual(values = LineTypes) +
  scale_x_log10(
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),
    expand = expansion(mult = c(0, 0))
  ) +
  scale_y_log10(
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),
    limits = c(1e-3, 1e2),
    expand = expansion(mult = c(0, 0))
  ) +
  annotation_logticks() +
  xlab("Particle diameter (um)") +
  ylab("Deposition velocity (cm/s)") +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),
    legend.box = "horizontal",
    panel.spacing = unit(1, "lines")
    # legend.margin=margin()
  ) +
  facet_wrap(~LUCLabel)
```

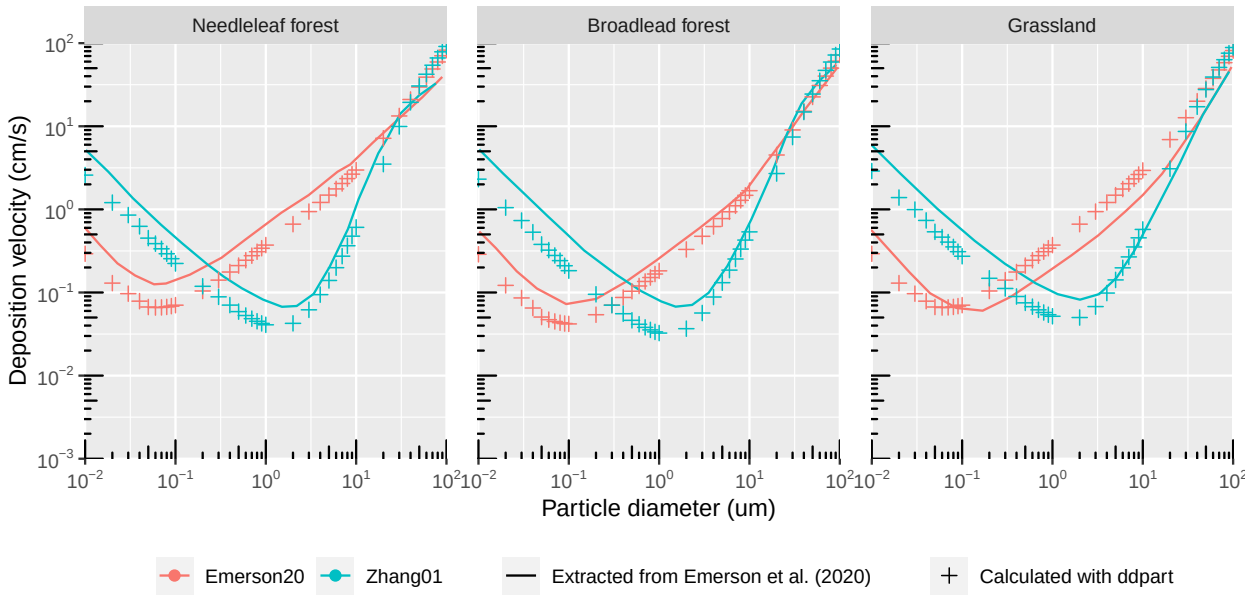

Figure 3: Dry deposition velocity as a function of particle diameter for needleleaf forest (LUC 1), evergreen broadleaf forest (LUC 2) and grassland (LUC 6) according to the Zhang et al. (2001) parametrization and the Emerson et al. (2020) parameterization. Lines represent data extracted from fig. 1 in Emerson et al. (2020). Results from the ddpart R package are indicated by "+". The figures show results at 0.2 m/s friction velocity and a particle density of 1200 kg/m3. Data from ddpart is generated for season 1 with aerodynamic resistance set to zero, wet surface, 85% relative humidity and corresponding hygroscopic swelling for aerosol of type "rural" (Zhang et al. (2001) eq. 10). Particle diameter refers to the dry particle diameter (i.e. before accounting for hygroscopic swelling).

FIXME - these deviations might be explained by some differences in the formulations of hygroscopic growth or the slip correction factors between Emerson et al. (2020) using the GEOS-CHEM model and the R implementation using hygroscopic growth from Zhang et al. (2001) and the slip correction from Seinfeld and Pandis (2006).

2. Contribution from dry deposition sub-processes

Emerson et al. (2020) figure 2 shows the contribution of the four dry deposition sub-processes (Brownian motion, interception, impaction and gravitational settling) to the total dry deposition velocity. This figure is reconstructed with functions from the ddpart package.

Definition of parameters

```
# Vector of particle diameters
d_p_vec <- 10^seq(-2, 2, length.out = 5)
tmp <- expand.grid(1:9, d_p_vec)
d_p_vec <- tmp[, 1] * tmp[, 2]
d_p_vec <- d_p_vec[d_p_vec <= 100]
d_p_vec <- d_p_vec * 1e-6

InputPars <- data.frame(
  R_a_sm = 0,
  Parametrization = "Emerson20",
  LUC = 1,
  Season = 1,
  ParticleDensity_kgm3 = 1500,
  FrictionVelocity_ms = 0.4
) %>%
  mutate(
    # These parameters are guessed
    # Standard temperatur and pressure
    T_air_K = 293.15,
    AirPressure_Pa = 101325,
    SurfaceIsWet_bool = F
  )

# Duplicate input for Zhang01 parametrization
tmp <- InputPars %>%
  mutate(
    Parametrization = "Zhang01"
  )
InputPars <- bind_rows(InputPars, tmp)

Input <- expand_grid(
  InputPars,
  ParticleDiameter_m = d_p_vec
)
```

Calculations

```
Output <- Input %>%
  mutate(
    # Meteo basics
    DynamicViscosityAir_kgms = CalculateDynamicViscosityOfAir(
      T_air_K = T_air_K
    ),
    AirDensity_kgm3 = CalculateAirDensity(
      AirPressure_Pa,
```

```
    T_air_K
  ),
  KinematicViscosityOfAir_m2s = CalculateKinematicViscosityOfAir(
    DynamicViscosityAir_kgms = DynamicViscosityAir_kgms,
    AirDensity_kgm3 = AirDensity_kgm3
  ),
  MeanFreePathOfAirMolecule_m = CalculateMeanFreePath(
    T_air_K = T_air_K,
    AirPressure_Pa = AirPressure_Pa,
    DynamicViscosityAir_kgms = DynamicViscosityAir_kgms
  ),
  # Deposition processes------
  SettlingVelocity_ms = CalculateSettlingVelocity(
    ParticleDensity_kgm3 = ParticleDensity_kgm3,
    ParticleDiameter_m = ParticleDiameter_m,
    MeanFreePathOfAirMolecule_m = MeanFreePathOfAirMolecule_m,
    DynamicViscosityAir_kgms = DynamicViscosityAir_kgms
  ),
  # _Schmidt number------
  SchmidtNumber = CalculateSchmidtNumber(
    DynamicViscosityAir_kgms = DynamicViscosityAir_kgms,
    KinematicViscosityOfAir_m2s = KinematicViscosityOfAir_m2s,
    T_air_K = T_air_K,
    ParticleDiameter_m = ParticleDiameter_m
  ),
  # _E_b-----
  # Loss efficiency by Brownian diffusion
  BrownianDiffusionParameterGamma = GetLandUseParameters(
    LUCs = LUC,
    Seasons = Season,
    TargetPar = "gamma",
    Parametrization = Parametrization
  ),
  E_b = CalculateLossEfficiencyBrownianDiffusion(
    SchmidtNumber = SchmidtNumber,
    BrownianDiffusionParameterGamma = BrownianDiffusionParameterGamma,
    Parametrization = Parametrization
  ),
  # _Stokes number-----
  SurfaceIsVegetated = GetLandUseParameters(
    LUCs = LUC,
    Seasons = Season,
    TargetPar = "SurfaceIsVegetated",
    Parametrization = Parametrization
  ),
  CharacteristicRadius_m = GetLandUseParameters(
    LUCs = LUC,
    Seasons = Season,
    TargetPar = "A_mm",
    Parametrization = Parametrization
  ) * 1e-3,
  StokesNumber = CalculateStokesNumber(
    FrictionVelocity_ms = FrictionVelocity_ms,
```

```
      SettlingVelocity_ms = SettlingVelocity_ms,
      CharacteristicRadius_m = CharacteristicRadius_m,
      KinematicViscosityOfAir_m2s = KinematicViscosityOfAir_m2s,
      SurfaceIsVegetated = SurfaceIsVegetated
    ),
    # _E_Im----
    # Loss efficiency by impaction
    ImpactionParameterAlpha = GetLandUseParameters(
      LUCs = LUC,
      Seasons = Season,
      TargetPar = "alpha",
      Parametrization = Parametrization
    ),
    E_Im = CalculateLossEfficiencyImpaction(
      StokesNumber = StokesNumber,
      ImpactionParameterAlpha = ImpactionParameterAlpha,
      Parametrization = Parametrization
    ),
    # _E_In----
    # Loss efficiency by interception
    E_In = CalculateLossEfficiencyInterception(
      ParticleDiameter_m = ParticleDiameter_m,
      CharacteristicRadius_m = CharacteristicRadius_m,
      Parametrization = Parametrization
    ),
    # _R_s-----
    # Surface resistance
    R_s_sm = CalculateSurfaceResistance(
      SurfaceIsWet = SurfaceIsWet_bool,
      FrictionVelocity_ms = FrictionVelocity_ms,
      StokesNumber = StokesNumber,
      E_b = E_b,
      E_Im = E_Im,
      E_In = E_In,
      Parametrization = Parametrization
    ),
    # _V_d---------
    V_d_s = 1 / (R_a_sm + R_s_sm),
    V_g = SettlingVelocity_ms,
    V_d_ms = V_d_s + V_g,
    Type = "Results from ddpart"
  )

# Plot separate dry deposition processes contributions
ProcessContributions <- Output %>%
  mutate(
    BounceCorrectionTerm = ifelse(
      test = SurfaceIsWet_bool,
      yes = 1,
      no = exp(-sqrt(StokesNumber))
    ),
    # Calculation of resistances and corresponding vd's for each process
    epsilon_0 = GetParameters(TargetParameter = "epsilon_0", Parametrization = Parametrization),
```

```
  R_s_E_b_only = 1 / (epsilon_0 * FrictionVelocity_ms * BounceCorrectionTerm * E_b),
  V_d_E_b_only = 1 / (R_a_sm + R_s_E_b_only),
  R_s_E_Im_only = 1 / (epsilon_0 * FrictionVelocity_ms * BounceCorrectionTerm * E_Im),
  V_d_E_Im_only = 1 / (R_a_sm + R_s_E_Im_only),
  R_s_E_In_only = 1 / (epsilon_0 * FrictionVelocity_ms * BounceCorrectionTerm * E_In),
  V_d_E_In_only = 1 / (R_a_sm + R_s_E_In_only)
) %>%
select(Parametrization, ParticleDiameter_m, V_d_E_b_only, V_d_E_Im_only, V_d_E_In_only, V_d_ms, Settli
pivot_longer(
  cols = -c("Parametrization", "ParticleDiameter_m"),
  names_to = "Component",
  values_to = "v"
) %>%
mutate(
  ComponentName = case_when(
    Component == "V_d_E_b_only" ~ "Brownian",
    Component == "V_d_E_Im_only" ~ "Impaction",
    Component == "V_d_E_In_only" ~ "Interception",
    Component == "SettlingVelocity_ms" ~ "Settling",
    Component == "V_d_ms" ~ "Total"
  )
)
```

Summary of parameters

```
TableData <- Output %>%
  select(
    Parametrization, LUC, Season,
    T_air_K, AirPressure_Pa, SurfaceIsWet_bool,
    SurfaceIsVegetated, CharacteristicRadius_m,
    ImpactionParameterAlpha, ParticleDensity_kgm3,
    BrownianDiffusionParameterGamma,
    R_a_sm
  ) %>%
  distinct() %>%
  rename(
    Gamma = BrownianDiffusionParameterGamma
  ) %>%
  mutate(
    Gamma = round(Gamma, 3),
    C_b = GetParameters(Parametrization = Parametrization, TargetParameter = "C_b"),
    nu = GetParameters(Parametrization = Parametrization, TargetParameter = "nu"),
    C_In = GetParameters(Parametrization = Parametrization, TargetParameter = "C_In"),
    beta = GetParameters(Parametrization = Parametrization, TargetParameter = "beta"),
    C_Im = GetParameters(Parametrization = Parametrization, TargetParameter = "C_Im")
  ) %>%
  pivot_longer(
    cols = -c("Parametrization", "LUC"),
    values_transform = as.character,
    names_to = "Parameter"
  ) %>%
  mutate(
    Parametrization_LUC = paste0(Parametrization, "_LUC", LUC)
  ) %>%
  select(-Parametrization, -LUC) %>%
  pivot_wider(
    names_from = "Parametrization_LUC",
    values_from = "value"
  )

kable(
  x = TableData
)
```

Parameter	Emerson20_LUC1	Zhang01_LUC1
Season	1	1
T_air_K	293.15	293.15
AirPressure_Pa	101325	101325
SurfaceIsWet_bool	FALSE	FALSE
SurfaceIsVegetated	TRUE	TRUE
CharacteristicRadius_m	0.002	0.002
ImpactionParameterAlpha	1	1
ParticleDensity_kgm3	1500	1500
Gamma	0.667	0.56
R_a_sm	0	0
C_b	0.2	1

Parameter	Emerson20_LUC1	Zhang01_LUC1
nu	0.8	2
C_In	2.5	0.5
beta	1.7	2
C_Im	0.4	1

Plots

```
# Load data extracted from Emerson et al. (2020)
ValidationData <- dd_subprocess_validation %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("Zhang01", "Emerson20")
    ),
    Type = "Extracted from Emerson et al. (2020)"
  )
ProcessContributions <- ProcessContributions %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("Zhang01", "Emerson20")
    ),
    Type = "Calculated with ddpart"
  )

Colors <- c("black", "blue", "orange", "darkred", "purple")
names(Colors) <- c("Total", "Brownian", "Settling", "Interception", "Impaction")
Shapes <- c(3)
names(Shapes) <- c("Calculated with ddpart")
LineTypes <- c("solid")
names(LineTypes) <- c("Extracted from Emerson et al. (2020)")

ggplot() +
  geom_line(
    data = ValidationData,
    mapping = aes(
      x = ParticleDiameter_um,
      y = DepositionVelocity_cms,
      color = Process,
      linetype = Type
    )
  ) +
  geom_point(
    data = ProcessContributions,
    mapping = aes(
      x = ParticleDiameter_m * 1e6,
      y = v * 100,
      color = ComponentName,
      shape = Type
    ),
    size = 2
```

```
  ) +
  scale_shape_manual(values = Shapes) +
  scale_color_manual(values = Colors) +
  scale_linetype_manual(values = LineTypes) +
  scale_x_log10(
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),
    expand = expansion(mult = c(0, 0))
  ) +
  scale_y_log10(
    breaks = trans_breaks("log10", function(x) 10^x),
    labels = trans_format("log10", math_format(10^.x)),
    limits = c(1e-3, 1e2),
    expand = expansion(mult = c(0, 0))
  ) +
  annotation_logticks() +
  xlab("Particle diameter (um)") +
  ylab("Deposition velocity (cm/s)") +
  theme(
    legend.position = "bottom",
    legend.title = element_blank(),
    legend.box = "vertical",
    panel.spacing = unit(1, "lines")
    # legend.margin=margin()
  ) +
  facet_wrap(~Label)
#> Warning: Removed 1 row(s) containing missing values (geom_path).
#> Warning: Removed 108 rows containing missing values (geom_point).
```

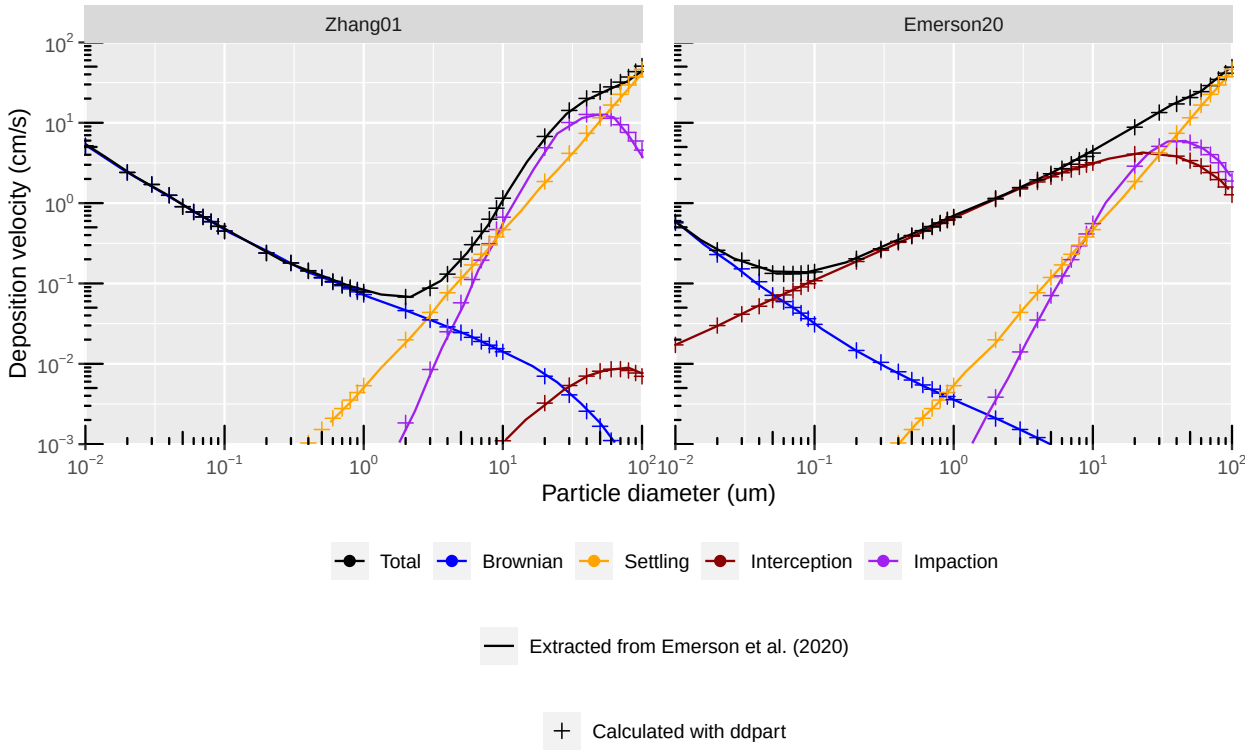

Figure 4: Contribution of Brownian diffusion, Gravitational settling, Interception and Impaction to the total dry deposition velocity according to the Zhang et al. (2001) parametrization (left panel) and the Emerson et al. (2020) parameterization (right panel). Lines indicate data extracted from fig. 2 in Emerson et al. (2020). Results from the ddpart R package are indicated by "+". The figures show results at 0.4 m/s friction velocity for needleleaf forest (LUC 1) and a particle density of 1500 kg/m3. Data from ddpart is generated for season 1 with aerodynamic resistance set to zero, no hygroscopic swelling and dry surface.

References

- Seinfeld JH, Pandis SN. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. Second edition. Wiley; 2006.
- Zhang L, Gong S, Padro J, Barrie L. A size-segregated particle dry deposition scheme for an atmospheric aerosol module. Atmospheric Environment 2001;35:549–560.
- Emerson EW, Hodshire AL, DeBolt HM, Bilsback KR, Pierce JR, McMeeking GR, Farmer DK. Revisiting particle dry deposition and its role in radiative effect estimates. Proceedings of the National Academy of Sciences 2020;117:26076–26082.