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))
```

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(
    LUCNames = "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 = "Zhang01",
    SurfaceIsVegetated_bool = T
  )


Results <- CalculateDepositionVelocity(
  InputTable = InputTable
)
```

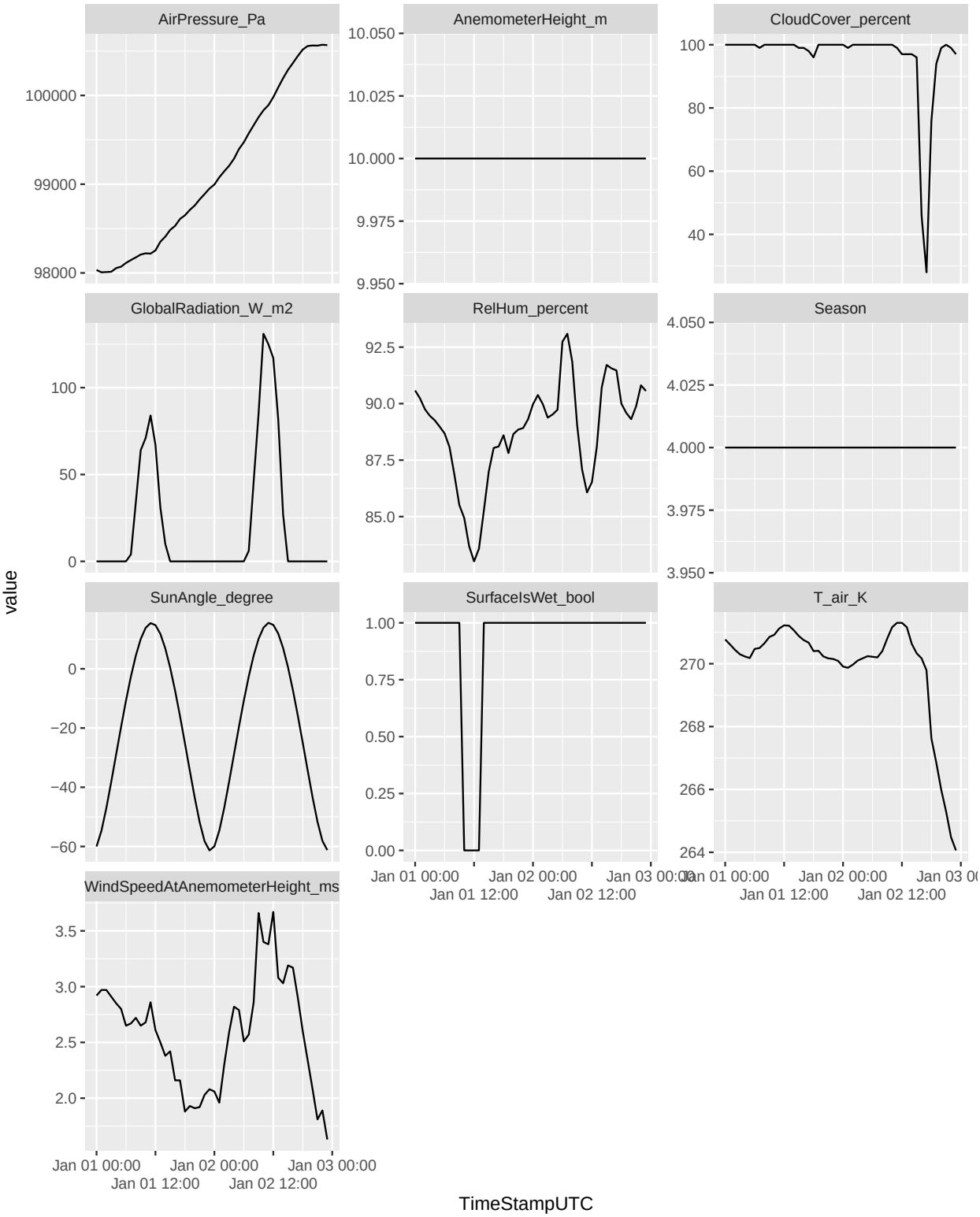

Figure 1: Example time series of meteorological data.

```r
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))
```

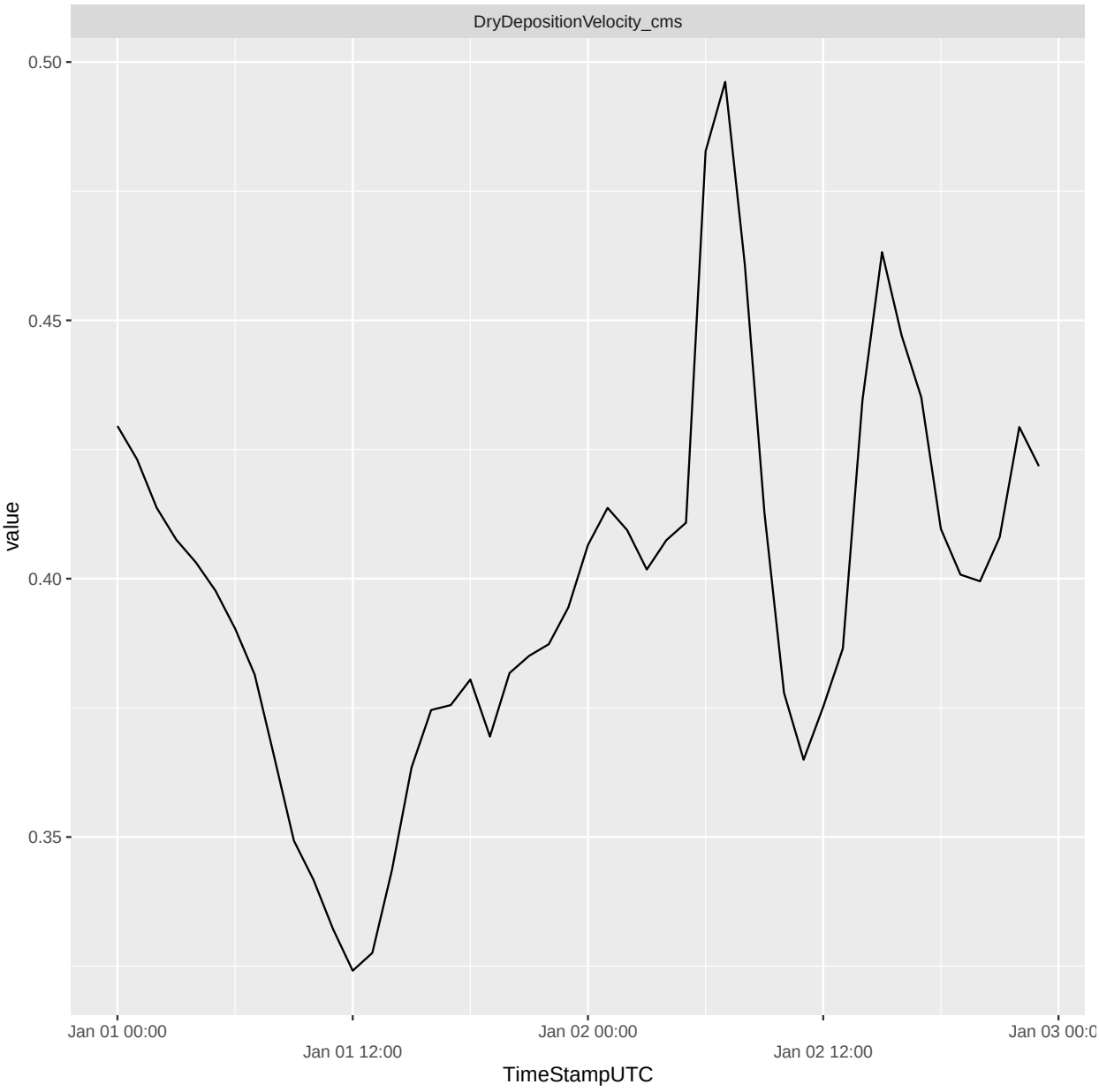

Figure 2: Dry deposition velocity calculated for the 48 hours of input data.

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. Emerson et al. (2020) used a specific implementation of the Zhang01 approach in the GEOS-Chem model (https://github.com/geoschem/geos-chem/). This implementation differs from the original formulation by Zhang01 in some aspects. In total, there are 4 variants of parametrizations:

A) "Zhang01": The original model according to Zhang et al. (2001). Not used for the figures in Emerson et al. (2020). Implemented in ddpart as parametrization "Zhang01".

B) "GEOS-Chem old": This is what Emerson et al. (2020) refer to as "Zhang et al. (2001)" in their Fig. 1 and 2. Implemented in ddpart as parametrization "GCOld". It is an adaption of parametrization A) with two differences:

- First, the land use classes in the GEOS-Chem model differ from those in A). The mapping between land use classes is not always straightforward. For example, "Grassland" in B) uses parameters for "shrubs and interrupted woodland" from A). The complete mapping of LUCs is defined in file "drydep_mod.F90" starting in lines 3143: https://github.com/geoschem/geos-chem/blob/main/GeosCore/drydep_mod.F90.
- Second, no differentiation according to seasons exists. Instead, the average over seasons 1-5 is used for each parameter. See for example "drydep_mod.F90" line 3226.

C) "GEOS-Chem new": This is the "revised parametrization" in Emerson et al. Fig. 1 and 2. Implemented in ddpart as parametrization "GCNew". It uses the same assignment of LUCs as in B) and also averages land use parameters over seasons, but some parameters have been re-calibrated by Emerson et al. to better match measurement data. Note that although Emerson applies the parameters for "Shrubs and interrupted wood-lands" (LUC10 from Zhang01) for grassland, the parametrization is valid for grassland. This is because Emerson20 *calibrated* the other parameters, such that the resulting dry deposition velocity matches measurement data for grassland.

D) "GCNewSeason": Same as C) but with season-specific land use parameters. E.g. parameter "characteristic receptor radius" (A in mm) varies between seasons for deciduous broadleaf forest. This is implemented in ddpart as parametrization "GCNewSeason".

In order to reproduce Fig. 1 and Fig. 2 from Emerson et al. (2020), variants B) and C) must be used.

1. 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

```
# Land use parameters for variants B) and C)
LUCPars <- expand_grid(
  LUCNames = "Needleleaf",
  Season = 1, # irrelevant for GCOld and GCNew parametrizations
  Parametrization = c("GCOld", "GCNew") #see explanation text above
) %>%
  mutate(
    A_mm = GetLandUseParameters(
      LUCNames = LUCNames,
      Seasons = Season,
      Parametrizations = Parametrization,
      TargetPar = "A_mm"
    ),
    alpha = GetLandUseParameters(
      LUCNames = LUCNames,
      Seasons = Season,
      Parametrizations = Parametrization,
      TargetPar = "alpha"
    ),
    gamma = GetLandUseParameters(
      LUCNames = LUCNames,
      Seasons = Season,
      Parametrizations = Parametrization,
      TargetPar = "gamma"
    )
  )

# 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

# Other parameters
OtherPars <- data.frame(
  R_a_sm = 0,
  ParticleDensity_kgm3 = 1500,
  FrictionVelocity_ms = 0.4,
  SurfaceIsVegetated_bool = T
) %>%
  mutate(
    # These parameters are guessed
    # Standard temperatur and pressure
    T_air_K = 293.15,
    AirPressure_Pa = 101325,
```

```
    SurfaceIsWet_bool = F
  )

# Repeat input rows for each particle diameter
Input <- bind_cols(LUCPars, OtherPars) %>%
  expand_grid(
    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 = gamma,
    E_b = CalculateLossEfficiencyBrownianDiffusion(
      SchmidtNumber = SchmidtNumber,
      BrownianDiffusionParameterGamma = BrownianDiffusionParameterGamma,
      Parametrization = Parametrization
    ),
    # _Stokes number-----
    CharacteristicRadius_m = A_mm / 1000,
    StokesNumber = CalculateStokesNumber(
      FrictionVelocity_ms = FrictionVelocity_ms,
      SettlingVelocity_ms = SettlingVelocity_ms,
      CharacteristicRadius_m = CharacteristicRadius_m,
      KinematicViscosityOfAir_m2s = KinematicViscosityOfAir_m2s,
      SurfaceIsVegetated = SurfaceIsVegetated_bool
    ),
    # _E_Im----
    # Loss efficiency by impaction
```

```
    ImpactionParameterAlpha = alpha,
    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,
      ParticleDiameter_m = ParticleDiameter_m,
      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, Settl
  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, LUCNames,
    T_air_K, AirPressure_Pa, SurfaceIsWet_bool,
    SurfaceIsVegetated_bool, 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"),
    values_transform = as.character,
    names_to = "Parameter"
  ) %>%
  pivot_wider(
    names_from = "Parametrization",
    values_from = "value"
  )

kable(
  x = TableData
)
#> Warning: 'xfun::attr()' ist veraltet.
#> Benutzen Sie stattdessen 'xfun::attr2()'
#> Siehe help("Deprecated")
#> Warning: 'xfun::attr()' ist veraltet.
#> Benutzen Sie stattdessen 'xfun::attr2()'
#> Siehe help("Deprecated")
```

Parameter	GCOld	GCNew
LUCNames	Needleleaf	Needleleaf
T_air_K	293.15	293.15
AirPressure_Pa	101325	101325
SurfaceIsWet_bool	FALSE	FALSE
SurfaceIsVegetated_bool	TRUE	TRUE
CharacteristicRadius_m	0.002	0.002
ImpactionParameterAlpha	1	1
ParticleDensity_kgm3	1500	1500
Gamma	0.56	0.667

Parameter	GCOld	GCNew
R_a_sm	0	0
C_b	1	0.2
nu	2	0.8
C_In	0.5	2.5
beta	2	1.7
C_Im	1	0.4

Plots

```
# Load data extracted from Emerson et al. (2020)
ValidationData <- dd_subprocess_validation %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("GCOld", "GCNew")
    ),
    Type = "Extracted from Emerson et al. (2020)"
  )
ProcessContributions <- ProcessContributions %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("GCOld", "GCNew")
    ),
    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 containing missing values or values outside the scale range
#> (`geom_line()`).
#> Warning: Removed 108 rows containing missing values or values outside the scale range
#> (`geom_point()`).
```

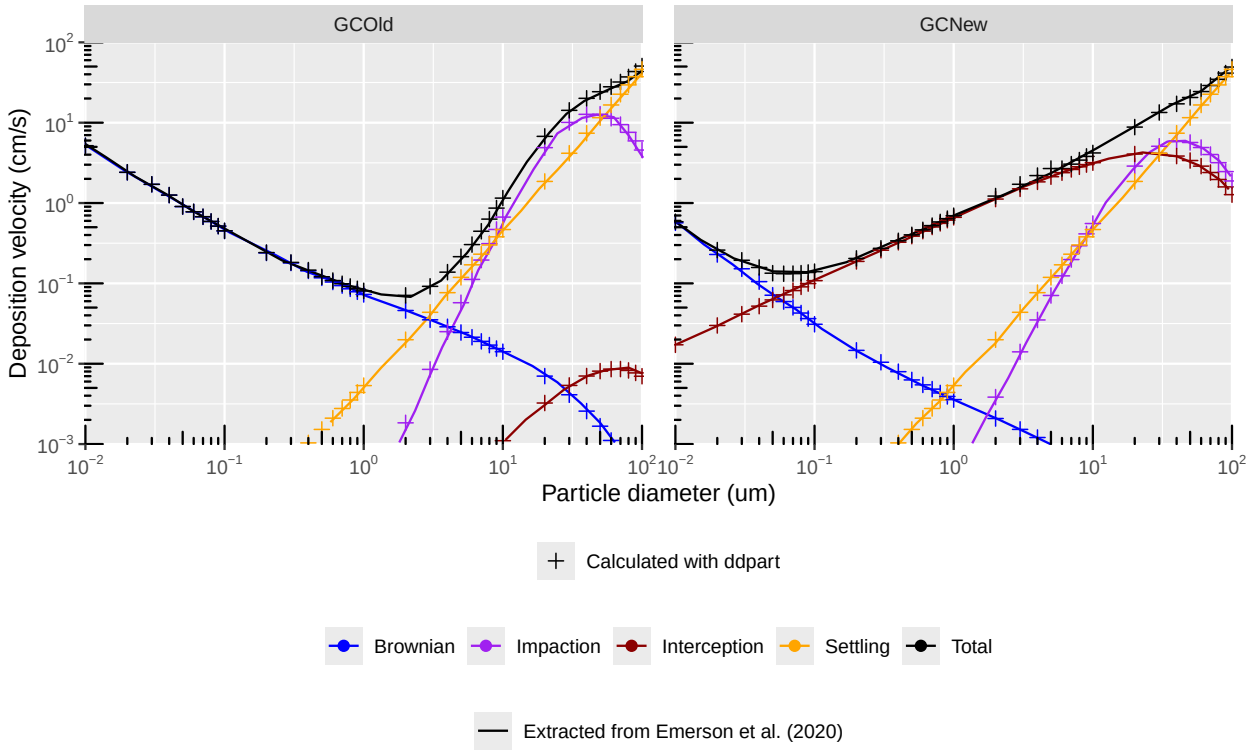

Figure 3: Contribution of Brownian diffusion, Gravitational settling, Interception and Impaction to the total dry deposition velocity. Lines indicate data extracted from fig. 2 in Emerson et al. (2020). Results from the ddpart R package are indicated by +. The Left panel shows parametrization 'GEOS-Chem old' which is similar to the Zhang et al. (2001) parametrization. The right panel shows parametrization 'GEOS-Chem new' which is referred to as the 'revised' parametrization in Emerson et al. (2020). The figures show results at 0.4 m/s friction velocity for needleleaf forest and a particle density of 1500 kg/m3. Data from ddpart is generated with aerodynamic resistance set to zero, no hygroscopic swelling and dry surface. See parameter table above for full information.

2. Dry deposition velocity vs. particle size for 3 different land use classes

Emerson et al. (2020) figure 1 shows the dry deposition velocity for different land use classes. Note that Emerson et al. state that their Fig. 1 has been generated with a friction velocity of 0.2 m/s. Three observations strongly suggest that this is not the case and that instead the same friction velocity as in Emerson et al. Fig. 2 (0.4 m/s) has been used by Emerson to generate Fig. 1. These observations are:

- The relation between dry deposition velocity and particle size for needleleaf forest in Emerson Fig. 1 is *exactly* identical to Emerson Fig. 2. This is shown below based on the data extracted from the Emerson paper. Emerson Fig. 2 has been generated with friction velocity of 0.4 m/s according to its figure caption.
- Dry deposition velocity is directly dependend on friction velocity (e.g. via surface resistance, see Emerson eq. 2). I.e. a change of friction velocity from 0.4 m/s to 0.2 m/s would create obvious differences in the relation between dry deposition velocity and particle size (which is not seen in Emerson Fig. 1 vs. Emerson Fig. 2).

Therefore, a friction velocity of 0.4 m/s is used to reproduce Emerson Fig. 1.

Definition of parameters

```
# LUC pars
LUCPars <- expand_grid(
  LUCNames = c("Grassland", "Needleleaf", "DecBroadleaf"),
  Season = 1, # irrelevant for GCOld and GCNew parameterizations
  Parametrization = c("GCOld", "GCNew") #see explanation text above
) %>%
  mutate(
    A_mm = GetLandUseParameters(
      LUCNames = LUCNames,
      Seasons = Season,
      Parametrizations = Parametrization,
      TargetPar = "A_mm"
    ),
    alpha = GetLandUseParameters(
      LUCNames = LUCNames,
      Seasons = Season,
      Parametrizations = Parametrization,
      TargetPar = "alpha"
    ),
    gamma = GetLandUseParameters(
      LUCNames = LUCNames,
      Seasons = Season,
      Parametrizations = Parametrization,
      TargetPar = "gamma"
    )
  )

# 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
```

```r
# Other parameters
OtherPars <- data.frame(
  # See above why FrictionVelocity_ms is 0.4 and not 0.2 m/s as stated in caption
  # of Emerson20 Fig1.
  FrictionVelocity_ms = 0.4,
  ParticleDensity_kgm3 = 1200,
  SurfaceIsVegetated = TRUE,
  # These parameters are guessed:
  # "article water uptake corresponding to ambient relative humidity"
  RelHum_percent = 80,
  AerosolType = "Rural",
  # Standard temperature and pressure
  T_air_K = 293.15,
  AirPressure_Pa = 101325,
  R_a_sm = 0,
  SurfaceIsWet_bool = FALSE
)

# Repeat input rows for each particle diameter
Input <- bind_cols(LUCPars, OtherPars) %>%
  expand_grid(
    DryParticleDiameter_m = d_p_vec
  )
```

Calculations

```
Output <- Input %>%
  mutate(
    ParticleDiameter_m = CalculateHygroscopicSwelling(
      DryParticleDiameter_m = DryParticleDiameter_m,
      RelHum_percent = RelHum_percent,
      AerosolType = AerosolType
    ),
    # 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-----
    E_b = CalculateLossEfficiencyBrownianDiffusion(
      SchmidtNumber = SchmidtNumber,
      BrownianDiffusionParameterGamma = gamma,
      Parametrization = Parametrization
    ),
    StokesNumber = CalculateStokesNumber(
      FrictionVelocity_ms = FrictionVelocity_ms,
      SettlingVelocity_ms = SettlingVelocity_ms,
      CharacteristicRadius_m = A_mm / 1000,
      KinematicViscosityOfAir_m2s = KinematicViscosityOfAir_m2s,
      SurfaceIsVegetated = SurfaceIsVegetated
    ),
    # _E_Im----
```

```
  # Loss efficiency by impaction
  E_Im = CalculateLossEfficiencyImpaction(
    StokesNumber = StokesNumber,
    ImpactionParameterAlpha = alpha,
    Parametrization = Parametrization
  ),
  # _E_In----
  # Loss efficiency by interception
  E_In = CalculateLossEfficiencyInterception(
    ParticleDiameter_m = ParticleDiameter_m,
    CharacteristicRadius_m = A_mm / 1000,
    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,
    ParticleDiameter_m = ParticleDiameter_m,
    Parametrization = Parametrization
  ),
  # _V_d---------
  V_d_surface_ms = 1 / (R_a_sm + R_s_sm),
  V_g = SettlingVelocity_ms,
  V_d_ms = V_d_surface_ms + V_g,
  Type = "Results from ddpart"
)
```

Summary of parameters

```
TableData <- Output %>%
  select(
    Parametrization, LUCNames,
    T_air_K, AirPressure_Pa, SurfaceIsWet_bool,
    SurfaceIsVegetated, A_mm,
    alpha, ParticleDensity_kgm3,
    gamma, R_a_sm
  ) %>%
  distinct() %>%
  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", "LUCNames"),
    values_transform = as.character,
    names_to = "Parameter"
  ) %>%
  mutate(
    LUCShort = case_when(
      LUCNames == "Needleleaf" ~ "NL",
      LUCNames == "DecBroadleaf" ~ "BL",
      LUCNames == "Grassland" ~ "GR",
      T ~ "ERROR"
    ),
    Parametrization_LUC = paste0(Parametrization, "_", LUCShort)
  ) %>%
  select(-Parametrization, -LUCNames, -LUCShort) %>%
  pivot_wider(
    names_from = "Parametrization_LUC",
    values_from = "value"
  )

kable(
  x = TableData
)
#> Warning: 'xfun::attr()' ist veraltet.
#> Benutzen Sie stattdessen 'xfun::attr2()'
#> Siehe help("Deprecated")
#> Warning: 'xfun::attr()' ist veraltet.
#> Benutzen Sie stattdessen 'xfun::attr2()'
#> Siehe help("Deprecated")
```

Parameter	GCOld_GR	GCNew_GR	GCOld_NL	GCNew_NL	GCOld_BL	GCNew_BL
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	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Parameter	GCOld_GR	GCNew_GR	GCOld_NL	GCNew_NL	GCOld_BL	GCNew_BL
SurfaceIsVegetated	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
A_mm	10	10	2	2	7	7
alpha	1.3	1.3	1	1	0.8	0.8
ParticleDensity_kgm3	1200	1200	1200	1200	1200	1200
gamma	0.54	0.667	0.56	0.667	0.56	0.667
R_a_sm	0	0	0	0	0	0
C_b	1	0.2	1	0.2	1	0.2
nu	2	0.8	2	0.8	2	0.8
C_In	0.5	2.5	0.5	2.5	0.5	2.5
beta	2	1.7	2	1.7	2	1.7
C_Im	1	0.4	1	0.4	1	0.4

Plots

```
# Load data extracted from Emerson et al. (2020)
ValidationData <- dd_validation %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("GCOld", "GCNew")
    ),
    Type = "Extracted from Emerson et al. (2020)",
    LUCLabel = factor(
      x = LUCName,
      levels = c("Needleleaf", "DecBroadleaf", "Grassland")
    )
  )
Output <- Output %>%
  mutate(
    Label = factor(
      x = Parametrization,
      levels = c("GCOld", "GCNew")
    ),
    Type = "Calculated with ddpart",
    LUCLabel = factor(
      x = LUCNames,
      levels = c("Needleleaf", "DecBroadleaf", "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)
```

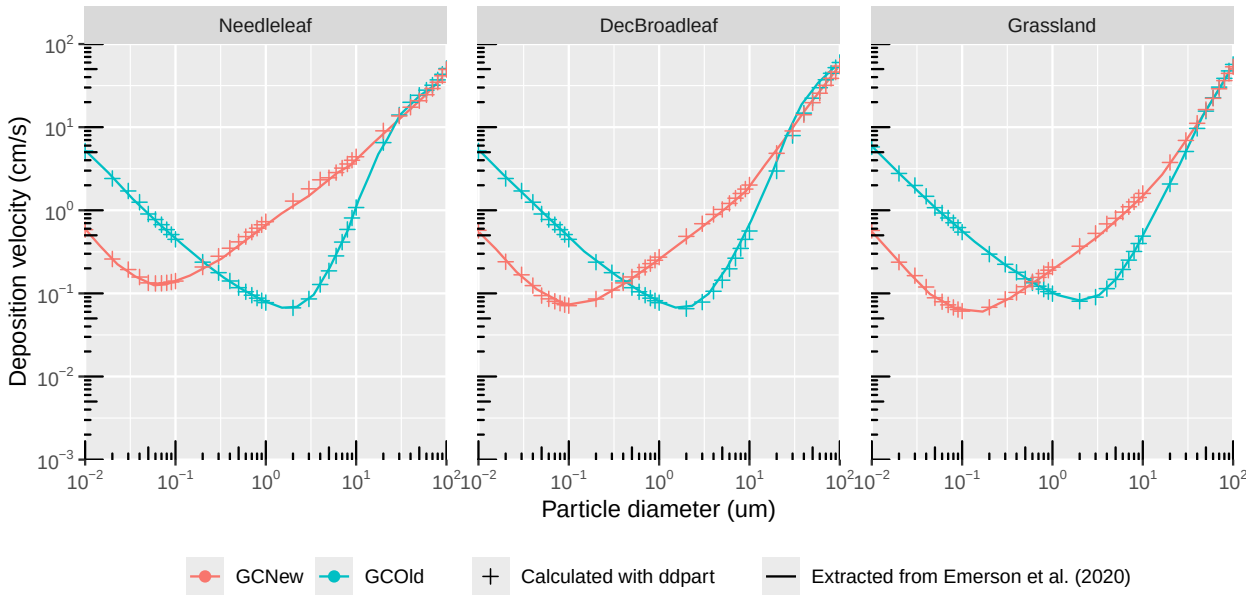

Figure 4: Dry deposition velocity as a function of particle diameter for needleleaf forest, decidous broadleaf forest and grassland. Lines represent data extracted from fig. 1 in Emerson et al. (2020). Results from the ddpart R package are indicated by +. The figures show results at 0.4 m/s friction velocity and a particle density of 1200 kg/m3. Data from ddpart is generated with aerodynamic resistance set to zero, dry surface, 80% 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).

Emerson Fig 1. vs. Emerson Fig. 2

The following graph shows that the relation between dry deposition velocity and particle size are exactly identical in Emerson Fig. 1 and Emerson Fig. 2. This strongly suggests that Emerson used the same value of friction velocity to generate these graphs.

- Variable "dd_validation" contains the modelled dry deposition velocity for needleleaf forest, deciduous broadleaf forest and grassland extracted from Emerson Fig 1. Here, only the data for needleleaf forest is used.
- Variable "dd_subprocess_validation" contains the modelled dry deposition velocity for needleleaf forest extracted from Emerson Fig. 2 ("total dry deposition" is used).
- In both cases, data has been extracted using a plot digitalizer tool.

```
EmersonFig1 <- dd_validation %>%
  filter(
    LUCName == "Needleleaf"
  ) %>%
  select(-LUCName) %>%
  mutate(
    DataSource = "Emerson et al. (2020) Fig1"
  )

EmersonFig2 <- dd_subprocess_validation %>%
  filter(
    Process == "Total"
  ) %>%
  select(-Process) %>%
  mutate(
    DataSource = "Emerson et al. (2020) Fig2"
  )

EmersonCmp <- bind_rows(EmersonFig1, EmersonFig2) %>%
  mutate(
    Parametrization = factor(
      x = paste0("Parametrization = ", Parametrization),
      levels = c("Parametrization = GCOld", "Parametrization = GCNew")
    )
  )

ggplot(
  data = EmersonCmp,
  mapping = aes(
    x = ParticleDiameter_um,
    y = DepositionVelocity_cms,
    color = DataSource,
    linetype = DataSource
  )
) +
  geom_line(
    size = 2
  ) +
  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))
  ) +
  facet_wrap(~Parametrization) +
  theme(
    legend.position = "bottom"
  )
#> Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
#> i Please use `linewidth` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
```

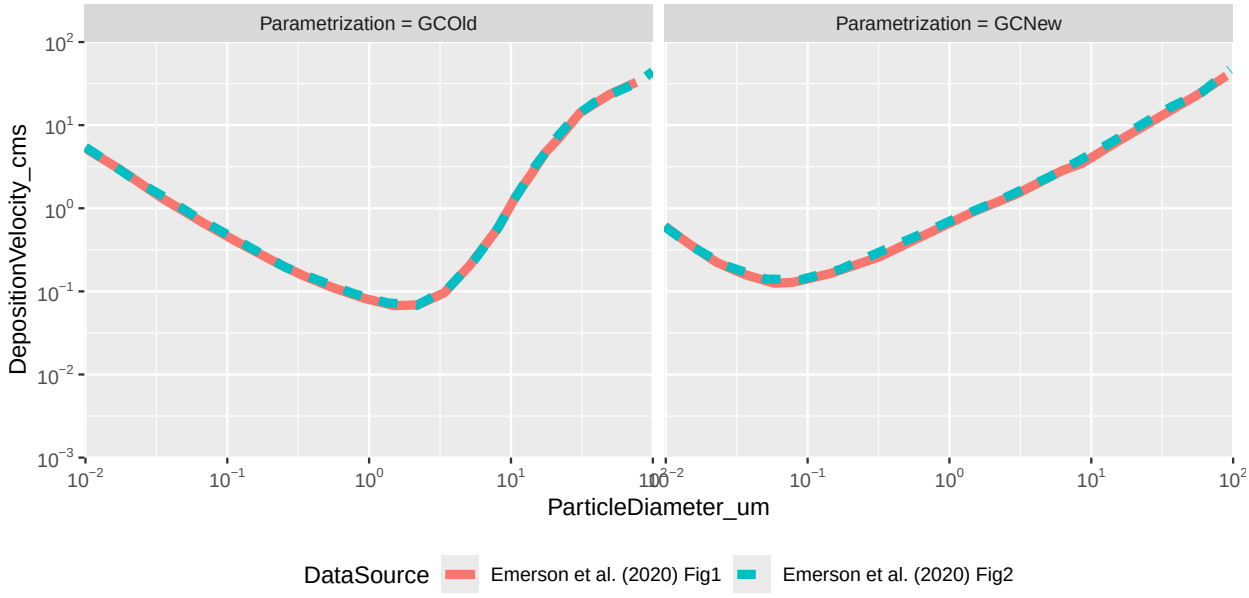

Figure 5: Modelled dry deposition velocity extracted from Emerson Fig. 1 and Fig. 2 is identical.

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.