

Package ‘ddpart’

September 26, 2025

Type Package
Title Simulation of particle dry deposition
Version 0.1.1
Author Andreas Schmitz
Maintainer Andreas Schmitz <andreas.schmitz@lanuk.nrw.de>
Description Implementation of the particle dry deposition model of Zhang et al. (2001) including the parametrization from Emerson et al. (2020).
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Suggests devtools,
ggplot2,
knitr,
rmarkdown,
scales,
tidyr
VignetteBuilder knitr
RoxygenNote 7.3.2
Depends dplyr,
R (>= 2.10)

R topics documented:

CalculateAerodynamicResistance	2
CalculateAirDensity	4
CalculateDepositionVelocity	4
CalculateDepositionVelocity2	6
CalculateDynamicViscosityOfAir	9
CalculateFrictionVelocity	10
CalculateHygroscopicSwelling	11
CalculateKinematicViscosityOfAir	13
CalculateLossEfficiencyBrownianDiffusion	13

CalculateLossEfficiencyImpaction	14
CalculateLossEfficiencyInterception	15
CalculateMeanFreePath	16
CalculateMoninObukhovLength	17
CalculateSchmidtNumber	18
CalculateSettlingVelocity	20
CalculateStokesNumber	21
CalculateSurfaceResistance	22
CalculateWindSpeedAtTargetHeight	23
dd_subprocess_validation	24
dd_validation	24
diffusion_validation	25
GetConstants	26
GetLandUseParameters	26
GetParameters	28
GetPasquillClass	29
meteo_time_series	30
Index	31

CalculateAerodynamicResistance
CalculateAerodynamicResistance

Description

Calculates aerodynamic resistance according to Erisman and Draaijers (1995) page 58. equation 3.4.

Usage

```
CalculateAerodynamicResistance(
    FrictionVelocity_ms,
    ReferenceHeight_m,
    ZeroPlaneDisplacementHeight_m,
    RoughnessLength_m,
    MoninObukhovLength_m
)
```

Arguments

FrictionVelocity_ms
 Friction velocity in in m/s.

ReferenceHeight_m
 Reference height in m. Aerodynamic resistance will be calculated between ReferenceHeight_m and the sum of ZeroPlaneDisplacementHeight_m + RoughnessLength_m.

ZeroPlaneDisplacementHeight_m
Displacement height in m.

RoughnessLength_m
Roughness length in m.

MoninObukhovLength_m
Monin-Obkukhiv length in m. Monin-Obkukhiv length for neutral stratification (Pasquill class D) is "infinity". This case is encoded by a value defined in GetConstants()\$InfLength in this package.

Value

Aerodynamic resistance in s/m.

References

Erismann JW, Draaijers GPJ. Atmospheric Deposition In Relation to Acidification and Eutrophication. 1995.

Examples

```
# Aerodynamic resistance for extremely unstable stratification
# over grassland
PasquillClass <- "A"
RoughnessLength_m <- 0.03
WindSpeed_ms <- 1.5
AnemometerHeight_m <- 10
ReferenceHeight_m <- 10
ZeroPlaneDisplacementHeight_m <- 7 * RoughnessLength_m

MOL_m <- CalculateMoninObukhovLength(
  PasquillClass = PasquillClass,
  RoughnessLength_m = RoughnessLength_m
)
FrictionVelocity_ms <- CalculateFrictionVelocity(
  WindSpeed_ms = WindSpeed_ms,
  AnemometerHeight_m = AnemometerHeight_m,
  ZeroPlaneDisplacementHeight_m = ZeroPlaneDisplacementHeight_m,
  RoughnessLength_m = RoughnessLength_m,
  MoninObukhovLength_m = MOL_m
)
CalculateAerodynamicResistance(
  FrictionVelocity_ms = FrictionVelocity_ms,
  ReferenceHeight_m = ReferenceHeight_m,
  ZeroPlaneDisplacementHeight_m = ZeroPlaneDisplacementHeight_m,
  RoughnessLength_m = RoughnessLength_m,
  MoninObukhovLength_m = MOL_m
)
```

CalculateAirDensity	<i>CalculateAirDensity</i>
---------------------	----------------------------

Description

Calculates the density of air according to Seinfeld and Pandis (2006) page 735 eq. 16.36.

Usage

```
CalculateAirDensity(AirPressure_Pa, T_air_K)
```

Arguments

AirPressure_Pa Air pressure in Pa.
T_air_K Air temperature in K.

Value

Density of air in kg/m³.

References

Seinfeld JH, Pandis SN. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. Wiley; 2006.

CalculateDepositionVelocity	<i>CalculateDepositionVelocity</i>
-----------------------------	------------------------------------

Description

Calculates the deposition velocity of particle according to the Zhang et al. (2001) and the Emerson et al. (2020) publications.

This is a wrapper function over the various dry deposition sub-processes. It is designed for use cases where information on wind speed is available for the same land use type where deposition velocities should be calculated (e.g. wind speed measured at 10 m anemometer height and concentrations at 1.5 m, both over grassland.) Use CalculateDepositionVelocity2() for situations where wind speed data is available for one land use but dry deposition velocities should be calculated for another land use.

Required inputs are:

- (1) basic meteorological data (wind speed, global radiation, relative humidity, cloud cover, air temperature, air pressure, surface wetness)
- (2) information on the surface properties (land use class, roughness length and displacement height) and reference height (height of concentration measurements)

(3) particle properties (dry particle diameter, density, optional: aerosol type to calculate hygroscopic swelling)

The calculation steps in CalculateDepositionVelocity() are:

1. Calculate meteorological basics (e.g. viscosity and density of air) and whether its day of night (based on parameter SunAngle_degree).
2. Calculate the Pasquill class (general classification of atmospheric stability).
3. Calculate other meteorological parameters (Monin-Obukhov length, friction velocity, stability corrections).
4. Calculate the aerodynamic resistance (Ra) between the reference height and the effective height of the receptor (displacement height plus roughness length). Calculate surface resistance (Rs) and gravitation settling.
5. Calculate dry deposition velocity.

Usage

CalculateDepositionVelocity(InputTable)

Arguments

InputTable	<p>A data frame with the following columns:</p> <ul style="list-style-type: none"> - SunAngle_degree: The sun angle in degree. Only used to determine if its "day" ($> 0^\circ$) or "night" ($< 0^\circ$) which is required for the determination of the Pasquill stability class (GetPasquillClass()). The sun angle can easily be calculated with <code>oce::sunAngle()</code>. - T_air_K: Air temperature in Kelvin. - AirPressure_Pa: Air pressure in Pa. Required for the calculation of the mean free path of an air molecule, which is required for the calculation of the particle settling (sedimentation) velocity. - GlobalRadiation_W_m2: Global radiation in W/m². Required for the determination of the Pasquill stability class (GetPasquillClass()). - CloudCover_percent: Cloud cover in percent. Required for the determination of the Pasquill stability class (GetPasquillClass()). - WindSpeedAtAnemometerHeight_ms: Wind speed (m/s) at the the anemometer height. Required for GetPasquillClass() and CalculateFrictionVelocity(). - RoughnessLength_m: Roughness length (m) of the land cover for which the anemometer wind speed is provided. Required for multiple functions. - ZeroPlaneDisplacementHeight_m: Zero plane displacement height (m) of the land cover for which the anemometer wind speed is provided. Required for multiple functions. - AnemometerHeight_m: Height to which the wind speed data refers - SurfaceIsWet_bool: Boolean indicating whether the surface is wet, in which case the particle rebound effect is disabled. Can be set e.g. depending on relative humidity and/or precipitation events. - SurfaceIsVegetated_bool: Boolean indicating whether the surface is vegetated. Should be TRUE for all LUCs currently implemented in ddpert.
------------	--

- LUCNames: Land use classes (character). See ?GetLandUseParameters for information.
- ReferenceHeight_m: Aerodynamic resistance is calculated between ReferenceHeight_m and the effective receptor height (RoughnessLength_m + ZeroPlaneDisplacementHeight_m).
- RoughnessLength_m: Roughness length of the land use class.
- ZeroPlaneDisplacementHeight_m: Zero-plane displacement height of the land use class.
- Season: Integer determining the season according to Zhang et al. (2001) table 2.
- DryParticleDiameter_m: Particle diameter in m before the growth particles due to water uptake is taken into account (see parameter "AerosolType").
- ParticleDensity_kgm3: Density of the particles in kg/m³.
- AerosolType: 'Set parameter AerosolType to "Dry" to disable calculation of hygroscopic swelling. Set parameter "AerosolType" to one of "SeaSalt", "Urban", "Rural", "AmmoniumSulfate" to enable hygroscopic swelling according to Zhang et al. (2001). See CalculateHygroscopicSwelling() for further details.
- RelHum_percent: Relative humidity (particles).
- Parametrization: A character indicates which parametrization should be used. See ?GetLandUseParameters for allowed values.

Value

A data frame repeating the InputTable plus additional columns with calculated values.

References

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, Bilback 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.

Examples

See vignette.

CalculateDepositionVelocity2

CalculateDepositionVelocity2

Description

Calculates the deposition velocity of particle according to the Zhang et al. (2001) and the Emerson et al. (2020) publications.

This is a wrapper function over the various dry deposition sub-processes. It is designed for use cases where information on wind speed is available for one land use but dry deposition velocities should be calculated for another land use. For example, wind speed is available from modelled data for standard WMO conditions (10 m over grassland) but the dry deposition velocity should be calculated for forest. Use CalculateDepositionVelocity() for the more typical case where wind speed information is available for the same land use type where dry deposition should be calculated.

The approach therefore differentiates between two sites:

- "anemometer site": Location/land use class where the wind speed measurements are available. Characterized by parameters "RoughnessLengthAnemometer_m" and "ZeroPlaneDisplacementHeightAnemometer_m"
- "target land use site": Location/land use class where the dry deposition velocity should be calculated. Characterized by parameters "RoughnessLengthTargetLUC_m", "ZeroPlaneDisplacementHeightTargetLUC_m", "Season"

CalculateDepositionVelocity2() vertically extrapolates the wind speed at "anemometer site" to a "blending height" (e.g. 50 m) where atmospheric conditions are somewhat independent of the underlying surface. From the blending height, the parameters relevant for particle dry deposition to the target land use site (friction velocity, aerodynamic resistance, etc.) are calculated. The main calculation steps in CalculateDepositionVelocity() are:

1. Calculate meteorological basics (e.g. viscosity and density of air) and whether its day or night (based on parameter SunAngle_degree)
2. Calculate the Pasquill class (general classification of atmospheric stability) based on data at the anemometer location (e.g. grassland land use class). This classification is also applied to the target land use type.
3. Calculate other meteorological parameters for the anemometer site (Obukhov length, friction velocity, stability corrections) and extrapolate wind speed to the blending height, using CalculateWindSpeedAtTargetHeight()
4. Calculate meteorological parameters for the target land use type (Obukhov length, friction velocity, stability corrections), based on the wind speed at blending height and the surface properties of the target land use type.
5. Calculate the aerodynamic resistance (R_a) between the reference height and the effective height of the receptor / target land use type. Calculate surface resistance (R_s) and gravitation settling.
6. Calculate dry deposition velocity to the target land use class.

Usage

```
CalculateDepositionVelocity2(InputTable)
```

Arguments

InputTable	A data frame with the all columns required by CalculateDepositionVelocity(), but:
------------	---

- RoughnessLengthAnemometer_m: Roughness length (m) of the land cover for which the anemometer wind speed is provided. Required for multiple functions.
- ZeroPlaneDisplacementHeightAnemometer_m: Zero plane displacement height (m) of the land cover for which the anemometer wind speed is provided. Required for multiple functions.
- WindSpeedBlendingHeight_m: Height in m to which the wind speed is extrapolated. At this height, the wind speed is assumed to be independent of the underlying land use.
- TargetLUCNames: Land use classes (character). See ?GetLandUseParameters for information.
- RoughnessLengthTargetLUC_m: Roughness length (m) of the land cover for which the dry deposition velocities are calculated.
- ZeroPlaneDisplacementHeightTargetLUC_m: Zero plane displacement height (m) of the land cover for which the dry deposition velocities are calculated.

Value

A data frame repeating the InputTable plus additional columns with calculated values.

References

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.

Examples

```
if (require("tidyr")) {

# Define some standard conditions
Basics <- data.frame(
  SunAngle_degree = 30,
  T_air_K = 293.15,
  AirPressure_Pa = 101325,
  SurfaceIsWet_bool = FALSE,
  SurfaceIsVegetated_bool = TRUE,
  GlobalRadiation_W_m2 = 100,
  RelHum_percent = 80,
  CloudCover_percent = 80,
  RoughnessLengthAnemometer_m = 0.03, # WMO grassland
  ZeroPlaneDisplacementHeightAnemometer_m = 0.21,
  AnemometerHeight_m = 10,
  WindSpeedAtAnemometerHeight_ms = 5,
  WindSpeedBlendingHeight_m = 50, # WS no longer dependent on LUC at 50m
  Season = 1,
  DryParticleDiameter_m = 5e-6,
  ParticleDensity_kgm3 = 2000,
  AerosolType = "Dry", # disable hygroscopic swelling
```



```

    Parametrization = "GCNew",
    ReferenceHeight_m = 50 # assuming concentrations are known at blending height
  )

  # Define two land use classes for which to calculate dry deposition velocity
  TargetLUCs <- data.frame(
    TargetLUCNames = c("Needleleaf", "DecBroadleaf"),
    RoughnessLengthTargetLUC_m = c(0.8, 2.65) # Zhang01 table 3
  ) %>%
    mutate(
      ZeroPlaneDisplacementHeightTargetLUC_m = 7 * RoughnessLengthTargetLUC_m
    )

  # Run calculations
  InputTable <- merge(Basics, TargetLUCs)
  Output <- CalculateDepositionVelocity2(
    InputTable = InputTable
  )

}

# Show results.
print(Output %>% select(TargetLUCNames, V_d_RefHeight_ms))

```

CalculateDynamicViscosityOfAir

CalculateDynamicViscosityOfAir

Description

Calculates Dynamic viscosity of air according to Seinfeld and Pandis (2006) page 909.

Usage

```
CalculateDynamicViscosityOfAir(T_air_K)
```

Arguments

T_air_K Air temperature in Kelvin.

Value

Dynamic viscosity of air in kg/(m*s).

References

Seinfeld JH, Pandis SN. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. Wiley; 2006.

```
CalculateFrictionVelocity
    CalculateFrictionVelocity
```

Description

Calculates friction velocity according to Erisman and Draaijers (1995) page 67 equation 3.25.

Usage

```
CalculateFrictionVelocity(
  WindSpeed_ms,
  AnemometerHeight_m,
  ZeroPlaneDisplacementHeight_m,
  RoughnessLength_m,
  MoninObukhovLength_m
)
```

Arguments

WindSpeed_ms Wind speed in m/s.

AnemometerHeight_m
 Height in m to which WindSpeed_ms refers.

ZeroPlaneDisplacementHeight_m
 Displacement height in m.

RoughnessLength_m
 Roughness length in m.

MoninObukhovLength_m
 Monin-Obkukhiv length in m. Monin-Obkukhiv length for neutral stratification (Pasquill class D) is "infinity". This case is encoded by a value defined in GetConstants()\$InfLength in this package.

Value

A friction velocity value in m.

References

Erisman JW, Draaijers GPJ. Atmospheric Deposition In Relation to Acidification and Eutrophication. 1995.

Examples

```
# Friction velocity over grassland for extremely unstable stratification
# (Pasquill class A)
MOL_m <- CalculateMoninObukhovLength(
  PasquillClass = "A",
```

```

    RoughnessLength_m = 0.03
  )
  CalculateFrictionVelocity(
    WindSpeed_ms = 1.5,
    AnemometerHeight_m = 10,
    ZeroPlaneDisplacementHeight_m = 0.21,
    RoughnessLength_m = 0.03,
    MoninObukhovLength_m = MOL_m
  )

# Friction velocity over grassland for neutral stratification
# (Pasquill class D)
MOL_m <- CalculateMoninObukhovLength(
  PasquillClass = "D",
  RoughnessLength_m = 0.03
)
CalculateFrictionVelocity(
  WindSpeed_ms = 7,
  AnemometerHeight_m = 10,
  ZeroPlaneDisplacementHeight_m = 0.21,
  RoughnessLength_m = 0.03,
  MoninObukhovLength_m = MOL_m
)

```

CalculateHygroscopicSwelling

CalculateHygroscopicSwelling

Description

Calculates increase in particle diameter due to water uptake according to Zhang et al. (2001) eq. 10. All parameters must be vectors of same lengths. Note that a correction has been applied to Zhang et al. (2001) eq. 10: The whole equation is raised to the power of 1/3, following the original publication mentioned by Zhang: Gerber (1985). Without this correction, the wet diameter is often smaller compared to the dry diameter.

Usage

```

CalculateHygroscopicSwelling(
  DryParticleDiameter_m,
  AerosolType,
  RelHum_percent
)

```

Arguments

DryParticleDiameter_m

The diameter of the particles before application of hygroscopic swelling (dry) in m.

AerosolType Hygroscopic swelling differs depending on aerosol type. Implemented types are (1) "Dry" for no swelling, (2) "SeaSalt", (3) "Urban", (4) "Rural" and (5) "AmmoniumSulfate".

RelHum_percent Relative humidity in percent.

Value

A vector of particle diameters after accounting for hygroscopic swelling in m.

References

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.

Gerber HE. 1985. Relative - Humidity Parameterization of the Navy Aerosol Model (NAM).

NAVAL RESEARCH LAB WASHINGTON DC; December 30, 1985. Available at: <https://apps.dtic.mil/sti/citations/ADA16>

Examples

```
DryParticleDiameter_m <- c(0.01, 1, 5, 10) * 1e-6
AerosolType <- c("SeaSalt", "Rural", "Dry")
RelHum_percent <- seq(60, 100, by = 0.5)
Input <- expand.grid(
  DryParticleDiameter_m = DryParticleDiameter_m,
  AerosolType = AerosolType,
  RelHum_percent = RelHum_percent
)

Output <- Input %>%
  mutate(
    WetParticleDiameter_m = CalculateHygroscopicSwelling(
      DryParticleDiameter_m = DryParticleDiameter_m,
      AerosolType = AerosolType,
      RelHum_percent = RelHum_percent
    )
  )

if (require("ggplot2")) {
  ggplot(
    data = Output,
    mapping = aes(
      x = RelHum_percent,
      y = WetParticleDiameter_m,
      color = as.factor(DryParticleDiameter_m),
      linetype = AerosolType
    )
  ) +
  geom_line()
}
```

```
CalculateKinematicViscosityOfAir
    CalculateKinematicViscosityOfAir
```

Description

Calculates kinematic viscosity of air according to Dixon (2007) appendix B

Usage

```
CalculateKinematicViscosityOfAir(DynamicViscosityAir_kgms, AirDensity_kgm3)
```

Arguments

```
DynamicViscosityAir_kgms
    Dynamic viscosity of air in kg/(m*s).
AirDensity_kgm3
    Density of air in kg/m3.
```

Value

Kinematic viscosity of air in kg/(m*s).

References

Dixon JC. The Shock Absorber Handbook. John Wiley & Sons; October 22, 2007.

```
CalculateLossEfficiencyBrownianDiffusion
    CalculateLossEfficiencyBrownianDiffusion
```

Description

Calculates loss efficiency by brownian diffusion according to Emerson et al. (2020) eq. 3.

Usage

```
CalculateLossEfficiencyBrownianDiffusion(
    SchmidtNumber,
    BrownianDiffusionParameterGamma,
    Parametrization
)
```

Arguments

SchmidtNumber Schmidt number. E.g. provided by CalculateSchmidtNumber()
 BrownianDiffusionParameterGamma
 Empirical parameter. Land use specific in Zhang et al. (2001) and constant at a value of 2/3 in Emerson et al. (2020)
 Parametrization
 A character defining which parametrization to use. See ?GetLandUseParameters for a list of valid values.

Value

Loss efficiency by brownian diffusion

References

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.

CalculateLossEfficiencyImpaction

CalculateLossEfficiencyImpaction

Description

Calculates loss efficiency by impaction according to Emerson et al. (2020) eq. 4.

Usage

```
CalculateLossEfficiencyImpaction(  
  StokesNumber,  
  ImpactionParameterAlpha,  
  Parametrization  
)
```

Arguments

StokesNumber Stokes number
 ImpactionParameterAlpha
 A land-use specific empirical paramete
 Parametrization
 A character defining which parametrization to use. See ?GetLandUseParameters for details.

Value

Stokes number

References

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.

CalculateLossEfficiencyInterception
CalculateLossEfficiencyInterception

Description

Calculates the loss efficiency by interception (E_{In}) according to Emerson et al. (2020)

Usage

```
CalculateLossEfficiencyInterception(  
    ParticleDiameter_m,  
    CharacteristicRadius_m,  
    Parametrization,  
    SurfaceIsVegetated  
)
```

Arguments

ParticleDiameter_m
Particle diameter in m.

CharacteristicRadius_m
Characteristic radius of the receptor surfaces in m.

Parametrization
A character defining which parametrization to use. See ?GetLandUseParameters for a list of valid values.

SurfaceIsVegetated
A boolean indicated whether the surface is vegetated. Interception is zero for non-vegetated surfaces.

Value

Loss efficiency by interception.

References

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.

`CalculateMeanFreePath` *CalculateMeanFreePath*

Description

Calculates the mean free path of an air molecule according to Seinfeld and Pandis (2006) page 399 eq. 9.6.

Usage

```
CalculateMeanFreePath(T_air_K, AirPressure_Pa, DynamicViscosityAir_kgms)
```

Arguments

`T_air_K` Air temperature in Kelvin.
`AirPressure_Pa` Air pressure in Pa.
`DynamicViscosityAir_kgms` Dynamic viscosity of air in kg/(m*s). E.g. provided by `CalculateDynamicViscosityOfAir()`.

Value

Mean free path of an air molecule in m.

References

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.

Examples

```
# Validation against example Seinfeld and Pandis eq 9.7 page 399
T_air_K <- 298
AirPressure_Pa <- 101325
DynamicViscosityAir_kgms <- 1.8e-5
MeanFreePath_m <- CalculateMeanFreePath(T_air_K, AirPressure_Pa, DynamicViscosityAir_kgms)
MeanFreePath_um <- MeanFreePath_m * 1e6
print(MeanFreePath_um)
# 0.0651 um as given in SP06
```

CalculateMoninObukhovLength
CalculateMoninObukhovLength

Description

Calculates Monin-Obukhov length according to Seinfeld and Pandis (2006) page 751 eq. 16.83.

Usage

```
CalculateMoninObukhovLength(PasquillClass, RoughnessLength_m)
```

Arguments

PasquillClass A single character indicating the Pasquill stability class. For example according to Seinfeld and Pandis (2006) page 750 as implemented in GetPasquillClass().

RoughnessLength_m
Roughness length in m.

Value

A numeric value for Monin-Obukhov length. Monin-Obukhov length for neutral stratification (Pasquill class D) is "infinity". This case is encoded by a value defined in GetConstants()\$InfLength in this package.

References

Seinfeld JH, Pandis SN. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change 2006.

Examples

```
# For Pasquill class A ("extremely unstable" conditions) and grassland
CalculateMoninObukhovLength(
  PasquillClass = "A",
  RoughnessLength_m = 0.03
)

# For Pasquill class D ("neutral" conditions) and grassland
CalculateMoninObukhovLength(
  PasquillClass = "D",
  RoughnessLength_m = 0.03
)
GetConstants()$InfLength
```

 CalculateSchmidtNumber

CalculateSchmidtNumber

Description

Calculates the Schmidt number according to Seinfeld and Pandis (2006) page 574

Usage

```
CalculateSchmidtNumber(
  DynamicViscosityAir_kgms,
  KinematicViscosityOfAir_m2s,
  T_air_K,
  ParticleDiameter_m
)
```

Arguments

DynamicViscosityAir_kgms	Dynamic viscosity of air in kg/(m*s). E.g. provided by CalculateDynamicViscosityOfAir().
KinematicViscosityOfAir_m2s	Kinematic viscosity of air in m2/2. E.g. provided by CalculateKinematicViscosityOfAir().
T_air_K	Air temperature in Kelvin.
ParticleDiameter_m	Particle diameter in m.

Value

Schmidt number

References

Seinfeld JH, Pandis SN. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. Wiley; 2006.

Examples

```
# Reproduce relation between brownian diffusivity and particle diameter as shown
# in Seinfeld and Pandis (2006) page 417 Figure 9.8

data(diffusion_validation)
PlotData <- data.frame(
  # Set standard atmospheric conditions
  T_air_K = 20,
  AirPressure_Pa = 101325,
```

```

# Set particle diameter according to Seinfeld and Pandis (2006)
ParticleDiameter_m = 10^seq(-9, -5, length.out = 50)
) %>%
# Calculation of quantities related to diffusion
mutate(
  DynamicViscosityAir_kgms = CalculateDynamicViscosityOfAir(T_air_K),
  AirDensity_kgm3 = CalculateAirDensity(
    AirPressure_Pa = AirPressure_Pa,
    T_air_K = T_air_K
  ),
  KinematicViscosityOfAir_m2s = CalculateKinematicViscosityOfAir(
    DynamicViscosityAir_kgms = DynamicViscosityAir_kgms,
    AirDensity_kgm3 = AirDensity_kgm3
  ),
  SchmidtNumber = CalculateSchmidtNumber(
    DynamicViscosityAir_kgms = DynamicViscosityAir_kgms,
    KinematicViscosityOfAir_m2s = KinematicViscosityOfAir_m2s,
    T_air_K = T_air_K,
    ParticleDiameter_m = ParticleDiameter_m
  ),
  # Calculate brownian diffusivity from intermediate results
  BrownianDiffusivity_m2s = KinematicViscosityOfAir_m2s / SchmidtNumber,
  # For plotting:
  BrownianDiffusivity_cm2s = BrownianDiffusivity_m2s * 1e4,
  ParticleDiameter_um = ParticleDiameter_m * 1e6,
  Type = "Results from ddpart"
)

# Plot
if (require("ggplot2")) {
  ggplot() +
    geom_line(
      data = diffusion_validation %>%
        mutate(
          Type = "Validation data from Seinfeld and Pandis (2006)"
        ),
      mapping = aes(
        x = log10(ParticleDiameter_um),
        y = log10(DiffusionCoefficient_cm2s),
        color = Type
      )
    ) +
    geom_point(
      data = PlotData,
      mapping = aes(
        x = log10(ParticleDiameter_um),
        y = log10(BrownianDiffusivity_cm2s),
        color = Type
      ),
      shape = "x"
    ) +
    annotate(
      geom = "text",

```

```

        x = -0,
        y = -3,
        label = "Small differences likely related\nto errors from extraction of
validation data from PDF figure."
    ) +
    theme(
        legend.position = "bottom"
    )
}

```

CalculateSettlingVelocity

CalculateSettlingVelocity

Description

Calculates settling velocity of particles by gravitation according to Zhang et al. (2001).

Usage

```

CalculateSettlingVelocity(
  ParticleDensity_kgm3,
  ParticleDiameter_m,
  MeanFreePathOfAirMolecule_m,
  DynamicViscosityAir_kgms
)

```

Arguments

ParticleDensity_kgm3
Particle density in kg/m3.

ParticleDiameter_m
Particle diameter in m.

MeanFreePathOfAirMolecule_m
Mean free path of an air molecule in m. E.g. provided by CalculateMeanFreePath().

DynamicViscosityAir_kgms
Dynamic viscosity of air in kg/(m*s). E.g. provided by CalculateDynamicViscosityOfAir().

Value

Gravitation settling velocity in m/s.

References

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.

CalculateStokesNumber	<i>CalculateStokesNumber</i>
-----------------------	------------------------------

Description

Calculates the stokes number according to Zhang et al. (2001)

Usage

```
CalculateStokesNumber(  
    FrictionVelocity_ms,  
    SettlingVelocity_ms,  
    CharacteristicRadius_m,  
    KinematicViscosityOfAir_m2s,  
    SurfaceIsVegetated  
)
```

Arguments

- FrictionVelocity_ms
Friction velocity in m/s
- SettlingVelocity_ms
Settling velocity in m/s
- CharacteristicRadius_m
Characteristic radius of receptor surface in m
- KinematicViscosityOfAir_m2s
Kinematic viscosity of air in m2/s
- SurfaceIsVegetated
Boolean value indicating whether the receptor surface is a vegetation surface

Value

Stokes number

References

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.

CalculateSurfaceResistance
CalculateSurfaceResistance

Description

Calculates the surface resistance (R_s) according to Zhang et al. (2001).

Usage

```
CalculateSurfaceResistance(  
    SurfaceIsWet,  
    FrictionVelocity_ms,  
    StokesNumber,  
    E_b,  
    E_Im,  
    E_In,  
    ParticleDiameter_m,  
    Parametrization  
)
```

Arguments

SurfaceIsWet	Indicator whether the receptor surface is wet (for bounce correction term), boolean.
FrictionVelocity_ms	Friction velocity in m/s.
StokesNumber	Stokes number.
E_b	Loss efficiency by brownian diffusion.
E_Im	Loss efficiency by impaction.
E_In	Loss efficiency by interception.
ParticleDiameter_m	Particle diameter in m.
Parametrization	A character defining which parametrization to use. See ?GetLandUseParameters for a list of valid values.

Value

Surface resistance in s/m.

References

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.

CalculateWindSpeedAtTargetHeight
CalculateWindSpeedAtTargetHeight

Description

Calculate wind speed according to stability-corrected log wind profile. This function is rearrangement of the formula for friction velocity after Erisman and Draaijers (1995) page 67. It is in line with https://en.wikipedia.org/wiki/Log_wind_profile

Usage

```
CalculateWindSpeedAtTargetHeight(  
    FrictionVelocity_ms,  
    TargetHeight_m,  
    ZeroPlaneDisplacementHeight_m,  
    RoughnessLength_m,  
    MoninObukhovLength_m  
)
```

Arguments

FrictionVelocity_ms
Friction velocity in m/s.

TargetHeight_m Height at which the wind speed should be calculated in m.

ZeroPlaneDisplacementHeight_m
Zero-plane displacement height in m.

RoughnessLength_m
Roughness length in m.

MoninObukhovLength_m
Monin-Obukhov-length in m.

Value

Wind speed at target height in m/s.

References

Erisman JW, Draaijers GPJ. Atmospheric Deposition In Relation to Acidification

dd_subprocess_validation

Data for validation of dry deposition subprocesses

Description

Data extracted from Emerson et al. (2020) fig. 2. Shows the dependency of dry deposition subprocesses to particle size according to the old and the old and the revised parametrization of the GEOS-Chem model. Note that the old parametrization is not identical to the Zhang et al. (2001) parametrization although it is termed "Zhang (2001)" in the Emerson et al. (2020) paper. See ?GetLandUseParameters for details.

Usage

```
data(dd_subprocess_validation)
```

Format

A data frame with 153 rows and 4 columns:

ParticleDiameter_um Particle diameter in um

DepositionVelocity_cms Dry deposition velocity resulting from the respective dry-deposition subprocess in cm²/s

Process Dry-deposition sub-process

Parametrization Parametrization (GEOS-Chem old (GCold) or GEOS-Chem new (GCNew))

References

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.

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.

dd_validation

Data for validation of dry deposition

Description

Data extracted from Emerson et al. (2020) fig. 1. Shows the dependency of dry deposition velocity to particle size for different land use types according to the old and the revised parametrization of the GEOS-Chem model. Note that the old parametrization is not identical to the Zhang et al. (2001) parametrization although it is termed "Zhang (2001)" in the Emerson et al. (2020) paper. See ?GetLandUseParameters for details.

Usage

```
data(dd_validation)
```

Format

A data frame with 106 rows and 4 columns:

ParticleDiameter_um Particle diameter in um

DepositionVelocity_cms Dry deposition velocity in cm²/s

Parametrization Parametrization (GEOS-Chem old (GCold) or GEOS-Chem new (GCNew))

LUCName Land use class name

References

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.

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.

diffusion_validation *Data for validation of the diffusion term*

Description

Shows the dependency of the diffusion coefficient and particle diameter. Data extracted from Seinfeld and Pandis (2006) page 417 fig. 9.8.

Usage

```
data(diffusion_validation)
```

Format

A data frame with 38 rows and 2 columns:

ParticleDiameter_um Particle diameter in um

DiffusionCoefficient_cm2s Diffusion coefficient in cm²/s

References

Seinfeld JH, Pandis SN. *Atmospheric Chemistry and Physics: From Air Pollution to Climate Change*. Wiley; 2006.

GetConstants

GetConstants

Description

This function returns some constants.

Usage

```
GetConstants()
```

Value

A named list of constants.

Examples

```
GetConstants()
```

GetLandUseParameters

GetLandUseParameters

Description

Get land use parameters according to Zhang et al. (2001) table 3 or Emerson et al. (2020). There are 4 variants of parametrization for the Zhang01 approach:

- 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" in a section starting with the string "LUC [Zhang et al., 2001]": https://github.com/geoschem/geos-chem/blob/main/GeosCore/drydep_mod.F90.

For the LUCs relevant for ddpart, the mapping Emerson20 <-> Zhang01 is as follows:

- "Needleleaf" <-> "Needleleaf" (LUC1)
- "Broadleaf" <-> "Deciduous broadleaf" (LUC4)
- "Grassland" <-> "Shrubs and interrupted wood-lands" (LUC10)

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. (2020) 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 the parameters for "Shrubs and interrupted wood-lands" (LUC10 from Zhang01) are used 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".

Usage

```
GetLandUseParameters(LUCNames, Seasons, Parametrizations, TargetParameter)
```

Arguments

LUCNames	A vector of land use class names (character). Currently, allowed values are - "Grassland" - "Needleleaf" - "DecBroadleaf" (deciduous broadleaf)
Seasons	A vector of season codes (integer values 1-5 according to Zhang et al. (2001); 1: Midsummer with lush vegetation, 2: Autumn with cropland that has not been harvested, 3: Late autumn after frost, no snow., 4: Winter, snow on ground and subfreezing, 5: Transitional spring with partially green short annuals)
Parametrizations	A character, one of "GCNewSeason", "Zhang01", "GCold", "GCNew".
TargetParameter	A character indicating which parameter to return ("z_0_m", "A_mm", "alpha" or "gamma").

Value

A vector of values for parameter "TargetParameter".

References

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.

Examples

```
if (require("tidyr")) {  
  
  GetLandUseParameters(  

```

```
LUCNames = c("Grassland", "Needleleaf", "DecBroadleaf"),
Seasons = rep(x = 4, times = 3),
Parametrizations = rep(x = "Zhang01", times = 3),
TargetParameter = "A_mm"
)

GetLandUseParameters(
  LUCNames = c("Grassland", "Needleleaf", "DecBroadleaf"),
  Seasons = rep(x = 4, times = 3),
  Parametrizations = rep(x = "GCNewSeason", times = 3),
  TargetParameter = "A_mm"
)

}
```

GetParameters	<i>GetParameters</i>
---------------	----------------------

Description

This function returns empirical constants for dry deposition sub-processes according to Zhang et al. (2001) or according to the re-paramtrization by Emerson et al. (2020). This function covers only parameters that are not land-use specific (see GetLandUseParameters()).

Usage

```
GetParameters(Parametrization, TargetParameter)
```

Arguments

- Parametrization
A character defining which parametrization to use. See ?GetLandUseParameters for a list of valid values.
- TargetParameter
A character indicating which parameter value to return. Valid options are "C_b", "beta", "C_Im", "nu", "C_In" and "epsilon_0". See Emerson et al. (2020) table S1.

Value

A named list of parameters.

References

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.

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.

Examples

```
if ( require("tidyr") ) {  
  GetParameters(Parametrization = "Zhang01", TargetParameter = "C_b")  
  GetParameters(Parametrization = "GCNewSeason", TargetParameter = "C_b")  
}
```

GetPasquillClass*GetPasquillClass*

Description

Calculates Pasquill stability class according to Seinfeld and Pandis (2006) page 750.

Usage

```
GetPasquillClass(  
  SurfaceWindSpeed_ms,  
  DayOrNight,  
  IncomingSolarRadiation_Wm2,  
  CloudCover_percent  
)
```

Arguments

SurfaceWindSpeed_ms	
	Wind speed at surface in m/s.
DayOrNight	Either "Day" or "Night" (character)
IncomingSolarRadiation_Wm2	
	Solar radiation in W/m2.
CloudCover_percent	
	Cloud cover in percent.

Value

Chracter string indicating the Pasquill stability class (A - F)

References

Seinfeld JH, Pandis SN. Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. 2006.

meteo_time_series	<i>Example 48 h meteorological time series</i>
-------------------	--

Description

Meteorological data required for dry deposition modelling (e.g. to determine atmospheric stability via Pasquill classes). Hourly data is for example available from the [ERA5 model](#).

Usage

```
data(meteo_time_series)
```

Format

A data frame with 48 rows and 11 columns

Index

* datasets

- dd_subprocess_validation, [24](#)
- dd_validation, [24](#)
- diffusion_validation, [25](#)
- meteo_time_series, [30](#)

- CalculateAerodynamicResistance, [2](#)
- CalculateAirDensity, [4](#)
- CalculateDepositionVelocity, [4](#)
- CalculateDepositionVelocity2, [6](#)
- CalculateDynamicViscosityOfAir, [9](#)
- CalculateFrictionVelocity, [10](#)
- CalculateHygroscopicSwelling, [11](#)
- CalculateKinematicViscosityOfAir, [13](#)
- CalculateLossEfficiencyBrownianDiffusion, [13](#)
- CalculateLossEfficiencyImpaction, [14](#)
- CalculateLossEfficiencyInterception, [15](#)
- CalculateMeanFreePath, [16](#)
- CalculateMoninObukhovLength, [17](#)
- CalculateSchmidtNumber, [18](#)
- CalculateSettlingVelocity, [20](#)
- CalculateStokesNumber, [21](#)
- CalculateSurfaceResistance, [22](#)
- CalculateWindSpeedAtTargetHeight, [23](#)

- dd_subprocess_validation, [24](#)
- dd_validation, [24](#)
- diffusion_validation, [25](#)

- GetConstants, [26](#)
- GetLandUseParameters, [26](#)
- GetParameters, [28](#)
- GetPasquillClass, [29](#)

- meteo_time_series, [30](#)