Package 'DMSTAr'

August 2, 2026

Type Package

Title Dynamic Model for Stormwater Treatment Areas

Version 0.1.1

Date 2026-07-23

Maintainer Paul Julian <pjulian@evergladesfoundation.org>

URL https://github.com/SwampThingPaul/DMSTAr

BugReports https://github.com/SwampThingPaul/DMSTAr/issues

Description Performs treatment wetland modeling consistent with the original 'Excel' spreadsheet and VBA code version of DMSTA2 developed by Dr. Bill Walker <http://wwwalker.net/dmsta/>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports stats, utils

Depends R (>= 4.1.0)

Suggests knitr, rmarkdown, kableExtra, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author Paul Julian [aut, cre] (ORCID: <https://orcid.org/0000-0002-7617-1354>)

Repository CRAN

Date/Publication 2026-08-02 16:30:18 UTC

Contents

`build_P_kinetics` *Build phosphorus kinetics parameters for a registered model*

Description

Builds a standardized parameter list for a selected phosphorus model type (e.g., `"STA"`, `"PSTA"`, `"RES"`) using a registered model-builder function, then derives kinetic coefficients `K1`, `K2`, `K3` and the kinetic model indicator `PModel` via `compute_DMSTA_kvals()`.

Usage

```
build_P_kinetics(mod_type, Dpy = 365.25, DutyCycle = NULL, pparams, ...)
```

Arguments

`mod_type`	Character scalar. Model identifier (e.g., `"STA"`).
`Dpy`	Numeric scalar. Time steps per year (default `365.25`).
`DutyCycle`	Numeric scalar in $[0, 1]$. Duty-cycle multiplier applied to time-scaled rate parameters.
`pparams`	Named `list`. Raw parameters expected by the chosen model builder.
`...`	Additional arguments passed through to the underlying model builder.

Details

Model builders are obtained from the package phosphorus model registry.

The selected model builder should return a list containing at minimum `C1000`, `Cstar`, and `Ks` (on the target time step). Additional fields returned by the builder (e.g., `Z1`, `Z2`, `Chalf`) are preserved. The function appends `K1`, `K2`, `K3`, and `PModel`.

The result is assigned class `"P_kinetics"` (prepended to any existing classes).

Value

A named `list` of standardized parameters with elements `K1`, `K2`, `K3`, and `PModel`. The result has class `"P_kinetics"`.

Examples

```
# Example assumes DMSTAr ships with a registered "STA" builder.
pparams <- list(
  C1000 = 1, Cstar = 0.2, Ks_per_yr = 0.5,
  Z1 = 10, Z2 = 30, Z3 = 60,
  K2Coef1 = 0.1, Chalf = 0.2, SeasonalFactor = 1,
  DutyCycle = 0.95
)
pars <- build_P_kinetics("STA", Dpy = 365.25, pparams = pparams)
pars$K1
pars$PModel
```

---

build_P_kin_slots *Build multi-slot phosphorus kinetics parameter vectors*

Description

Builds phosphorus kinetics for multiple model "slots" and assembles the results into K-length vectors (where `K = length(mods)`).

Usage

```
build_P_kin_slots(
  mods,
  registry = NULL,
  pparams,
  Dpy = 365.25,
  DutyCycle = NULL,
  derive_PModel = TRUE,
  default_PModel = 1L,
  ...
)
```

Arguments

mods	Character vector of model identifiers (e.g., c("STA","RES")).
registry	Optional model registry (named list of builder functions). If NULL (default), the package's default registry is used.
pparams	Named list of raw parameters. The same pparams is passed to every slot.
Dpy	Numeric scalar. Time steps per year (default 365.25).
DutyCycle	Numeric scalar in $[0, 1]$. Duty cycle multiplier.
derive_PModel	Logical; reserved for future use.
default_PModel	Integer scalar; reserved for future use.
...	Additional arguments passed to build_P_kinetics().

Details

Each slot is constructed by calling build_P_kinetics() with the same parameter list pparams. This is useful when a simulation uses multiple phosphorus modules that differ by model type but share the same raw parameter set.

Value

A named list containing:

K1, K2, K3 Numeric vectors of length K.

Chalf, Z_1, Z_2, Z_3, K2Coef Numeric vectors of length K.

Kslots Integer scalar giving K.

SeasonalFactor, Ytrans, Ysigma, Czero, PModel Global scalars.

mods Character vector of model identifiers.

Examples

```
pparams <- list(
  # shared / STA
  C1000 = 22, Cstar = 3, Ks_per_yr = 16,
  Z1 = 40, Z2 = 100, Z3 = 200,
  K2Coef1 = 0.1, Chalf = 50, SeasonalFactor = 1,
  # PSTA
  Ytrans = 1, Ysigma = 1, C1000_2 = 50, ks_2 = 20, zh_2 = 10,
  # RES
  k_depth_penalty = 0.5,
  DutyCycle = 0.95
)

out <- build_P_kin_slots(
  mods = c("STA", "PSTA", "RES"),
  pparams = pparams,
  Dpy = 365.25
)
out$K1
```

```
out$Z_1
```

`cfs_to_hm3d` *Convert Cubic Feet per Second to Cubic Hectometers per Day*

Description

Converts volumetric flow from cubic feet per second (cfs) to cubic hectometers per day (hm^3/day).

Usage

```
cfs_to_hm3d(x)
```

Arguments

`x`	Numeric vector of flow values in cubic feet per second.

Value

Numeric vector of flow values in cubic hectometers per day.

`dmstar_default_params` *Create a default DMSTAr parameter list*

Description

Constructs a named list of model parameters used by DMSTAr. Most arguments are scalar numeric values (or logical flags). Additional named parameters may be supplied via `...`; these will be appended to the returned list and will override any existing defaults with the same name.

Usage

```
dmstar_default_params(
  MT = FALSE,
  Qin_Frac = 0,
  A_cell = 0,
  Width = 0,
  Ntanks = 1,
  Zrelease = 0,
  Q_zmin = 0,
  Zweir = 0,
  Q_b = 0,
  Q_a = 0,
  Bypass_elev = 0,
```

```
Qomax = 0,
Qimax = 0,
Seepin_Rate = 0,
Seepin_Elev = 0,
seepin_conc = 0,
Seepout_Rate = 0,
Seepout_Elev = 0,
seepage_c = 20,
C_init_ppb = 0,
Y_init_mgm2 = 0,
Zinit = 40,
Cstar = 3,
C1000 = 22,
Chalf = 300,
Ks_per_yr = 0,
Z1 = 40,
Z2 = 100,
Z3 = 200,
K2Coef1 = 0,
SeasonalFactor = 0,
Ytrans = 0,
Ysigma = 0,
Czero = 0,
ks_2 = 0,
zh_2 = 0,
k_depth_penalty = 1,
C_rain = 10,
DryDepo = 20,
ShutdownET = TRUE,
force_Q_out = FALSE,
DutyCycle = 0.95,
Zmin = 2,
Cmax = 2000,
enable_P_release = FALSE,
K_release = 0,
dmsta_version = "2E",
offline_trigger = FALSE,
offline_start = NULL,
offline_freq = NULL,
offline_dur = NULL,
frac_1 = 0,
frac_2 = 0,
frac_3 = 0,
frac_4 = 0,
frac_5 = 0,
frac_6 = 0,
IsaNode = NULL,
...
```

```
)
```

Arguments

MT	Logical. If TRUE, replaces all numeric parameter values in the returned list with 0. Defaults to FALSE.
Qin_Frac	Numeric. Fraction of flow entering cell from basin.
A_cell	Numeric. Cell effective treatment are (units: km2).
Width	Numeric. mean width of flow path (units: km.
Ntanks	Integer-ish numeric. Number of tanks in series.
Zrelease	Numeric. Release elevation (units: cm).
Q_zmin	Numeric. Zc; no outflow below this water level , added to control depth specified in input series (units: cm).
Zweir	Numeric. Zw; fixed weir depth, use for reservoirs with outflow hydraulics controlled by outlet structure; = 0 for shallow systems when outflow is usually controlled by vegetation resistance (units: cm).
Q_b	Numeric. Rating curve parameter b; q / w = a (Z - Zw) ^b for Z >= Zc; 'typicallly ~ 3 to 4 for marsh control; 1.5 for weir control (reservoirs & other deep cells).
Q_a	Numeric. Rating curve parameter a; flow/width at water depth of 1 m (a); typically ~ 0.5 to 2.
Bypass_elev	Numeric. depth at which bypass begins (0 = no limit).
Qomax	Numeric. Inflow capacity (triggers bypass) (0 = no limit) (units: hm3/d).
Qimax	Numeric. Outflow capacity (0 = no limit) (units: hm3/d).
Seepin_Rate	Numeric. centimeters per day per centimeter of head, reflects transmissivity of soils (units: cm/d/cm).
Seepin_Elev	Numeric. 'drives inflow seepage rate, depth relative to mean ground surface elev (units: cm).
seepin_conc	Numeric. Concentration associated with seepage inflow (units: mg/m3).
Seepout_Rate	Numeric. centimeters per day per centimeter of head, reflects transmissivity of soils (units: cm/d/cm).
Seepout_Elev	Numeric. drives outflow seepage rate, relative to mean ground surface elev, can be < 0 (units: cm).
seepage_c	Numeric. Seepage concentration term (units: mg/m3).
C_init_ppb	Numeric. Initial concentration (units: mg/m3).
Y_init_mgm2	Numeric. Initial biomass P storage (units: mg/m^2).
Zinit	Numeric. Initial water column depth relative to mean ground elevation (units: cm).
Cstar	Numeric. Reference concentration parameter also defined as water column conc at storage = 0 mg/m2 at steady-state (units: mg/m3).
C1000	Numeric. water column conc at storage = 1000 mg/m2 at steady-state (units: mg/m3).

`Chalf`	Numeric. Half-saturation / half-response concentration parameter or water column concentration at 1/2 maximum uptake (units: mg/m3).
`Ks_per_yr`	Numeric. net settling rate in steady state in K/C* model (units: m/yr).
`Z1`	Numeric. Uptake rate decreases below this depth; flat between Z1 and Z2; =0 ignored (units: cm).
`Z2`	Numeric. uptake rate starts to decrease above this depth (30-day average); decreases linearly between Z2 & Z3; 0 = ignored; reflects damage to vegetation at high depths (units: cm)
`Z3`	Numeric. upper end of depth penalty range; K = 1 m/yr at depths above this value, regardless of calibration; 0 = ignored (units: cm).
`K2Coef1`	Numeric. Coefficient for secondary rate term (model-specific).
`SeasonalFactor`	Numeric. Seasonal factor multiplier (model-specific).
`Ytrans`	Numeric. Transform parameter for Y term (model-specific).
`Ysigma`	Numeric. Sigma/spread parameter for Y term (model-specific).
`Czero`	Numeric. Baseline concentration offset (model-specific).
`ks_2`	Numeric. Secondary coefficient (model-specific).
`zh_2`	Numeric. Secondary depth/elevation parameter (model-specific).
`k_depth_penalty`	Numeric. Depth penalty multiplier/parameter.
`C_rain`	Numeric. Rain concentration (model-specific units).
`DryDepo`	Numeric. Dry deposition loading term (model-specific units).
`ShutdownET`	Logical. If `TRUE`, ET is shut down under model-specific conditions.
`force_Q_out`	Logical. If `TRUE`, forces outflow behavior (model-specific).
`DutyCycle`	Numeric. Duty cycle (0–1) applied to relevant process(es).
`Zmin`	Numeric. Minimum elevation (model-specific).
`Cmax`	Numeric. Maximum concentration cap (model-specific).
`enable_P_release`	Logical. If `TRUE` P release from sediment is active
`K_release`	Numeric. Currently a kg/day rate. For future implementation
`dmsta_version`	Text. Version of DMSTA to implement default is set to "2E",
`offline_trigger`	Logical. If `FALSE`, STA rest period not implement.
`offline_start`	Date. Implement STA rest period start date. Default is `as.Date("1965-03-15)`
`offline_freq`	Numeric. frequency of rest period. Default `3L`
`offline_dur`	Numeric. Duration of rest period. Default `45L`
`frac_1`	Numeric. value used to update Qfrac during rest period implement
`frac_2`	Numeric. value used to update Qfrac during rest period implement
`frac_3`	Numeric. value used to update Qfrac during rest period implement
`frac_4`	Numeric. value used to update Qfrac during rest period implement

frac_5	Numeric. value used to update Qfrac during rest period implement
frac_6	Numeric. value used to update Qfrac during rest period implement
IsaNode	Logical or NULL. If NULL, derived as (A_cell <= 0).
...	Additional named parameters to add to the returned list, or to override existing defaults by name. All must be named (e.g., foo = 1).

Details

- **Derived** IsaNode**:** If IsaNode is NULL, it is set to (A_cell <= 0); otherwise the provided IsaNode value is used.
- MT **behavior:** When MT = TRUE, all numeric values in the returned parameter list are replaced with 0, providing an empty list. Logical and non-numeric entries are left unchanged.
- **Extra parameters via** ...**:** All arguments passed in ... must be named. New names trigger a warning and are added to the returned list. Names that already exist in the defaults override the default value.

Value

A named list of DMSTAr parameters.

Examples

```
# Get defaults
p <- dmstar_default_params()

# Override a few defaults
p <- dmstar_default_params(Zmin = 5, DutyCycle = 0.9, Ntanks = 3)

# Add a new parameter via ...
p <- dmstar_default_params(NewParam = 123)

# Derive IsaNode automatically when IsaNode is NULL
p <- dmstar_default_params(A_cell = 0)   # IsaNode becomes TRUE
p <- dmstar_default_params(A_cell = 10)  # IsaNode becomes FALSE

# MT = TRUE zeroes numeric parameters
p0 <- dmstar_default_params(MT = TRUE)
```

---

dmsta_build_tanks *Build DMSTA tank partitioning for a cell*

Description

Internal helper that partitions a single cell area into a series of conceptual "tanks" used by DMSTA. The number of tanks is derived from ttankS; if ttankS is fractional, the final tank receives the fractional area share and all preceding tanks receive equal shares.

Usage

```
dmsta_build_tanks(A_cell, ttankS, snap_last = TRUE)
```

Arguments

A_cell	Numeric scalar > 0. Cell area (units consistent with the rest of the DMSTA implementation).
ttankS	Numeric scalar > 0. Effective number of tanks. May be fractional.
snap_last	Logical; if TRUE, force the last cumulative fraction to exactly 1.0. Default is TRUE.

Details

The function also returns per-tank area fractions and cumulative fractions (useful for mapping depth/area relationships). Optionally, the last cumulative fraction can be "snapped" to exactly 1.0 for numerical stability.

Value

A named list with elements:

Ntanks Integer number of tanks.

A_Tank Numeric vector of tank areas, length Ntanks.

F_Tank Numeric vector of tank area fractions (A_Tank / A_cell).

Fcum Numeric vector of cumulative area fractions (cumsum(F_Tank)).

See Also

dmsta_p_init_state() for initializing state vectors compatible with the returned Ntanks.

---

dmsta_cals *Parameter Sets for Stormwater Treatment Area Dynamic Model*

Description

Internal lookup table of calibration sets used by the Dynamic Model for Stormwater Treatment Areas (DMSTA). Each row corresponds to a vegetation / compartment set (e.g., emergent marsh, SAV, PSTA, reservoir) and provides model coefficients, depth/flow/concentration ranges, and variability terms.

Usage

```
dmsta_cals
```

Format

A data frame with 5 rows and 25 variables:

Set Character. Parameter set identifier (e.g., `"EMG_3"`).

Descript Character. Human-readable description of the set.

C0 Integer. Concentration parameter C0 = Conc at 0 g/m2 P Storage (ug/L).

C1 Integer. Concentration parameter C1 = Conc at 1 g/m2 P storage (ug/L).

C2 Integer. Concentration parameter C2 = Conc at Half-Max Uptake.

Ks Numeric. Net Settling Rate at Steady State (m/yr).

Z1 Integer. Saturated Uptake Depth (cm).

Z2 Integer. Lower Penalty Depth (cm).

Z3 Integer. Upper Penalty Depth (cm).

K1 Numeric. First Order Removal Rate (m/yr).

C0_NEWS_SF Numeric. Periphyton for NEWS or Seasonal Adjustment for P Uptake (ug/L).

C1_Peri Numeric. Periphyton system - concentration parameter (ug/L).

Ks_Peri Numeric. Periphyton system - settling rate (1/yr).

Zx_Peri Numeric. Periphyton system - saturated uptake depth (cm).

Sm Numeric. Transition Storage Midpoint (mg/m2).

Sb Numeric. Transition Storage Bandwidth (mg/m2).

MinDepth Integer. Minimum depth observed/allowed for the set.

MaxDepth Integer. Maximum depth observed/allowed for the set.

MinQW Integer. Minimum flow (QW) observed/allowed for the set.

MaxQW Integer. Maximum flow (QW) observed/allowed for the set.

MinConc Numeric. Minimum concentration observed/allowed for the set.

MaxConc Numeric. Maximum concentration observed/allowed for the set.

MinFreqZ_LT10cm Numeric. Minimum frequency of Z < 10 cm.

MaxFreqZ_LT10cm Numeric. Maximum frequency of Z < 10 cm.

K_CV Numeric. Coefficient of variation for Ks (or related K term).

Source

dmsta xlsm file, sheet `Calibrations`

Examples

```
data(dmsta_cals)
dmsta_cals
```

`dmsta_flowP_case` *Run a networked DMSTA hydrology–phosphorus simulation*

Description

Simulates coupled hydrology and phosphorus dynamics for a network of interconnected DMSTA cells over a daily time series. Each cell is simulated sequentially within each day, with treated outflows routed downstream according to network topology, splitter rules, and recycle indices.

Usage

```
dmsta_flowP_case(
  series,
  cells,
  Nsteps = 4L,
  N_plant = 30L,
  Qmethod = c("RK4", "Euler", "RKF45", "custom"),
  Pmethod = c("RK4", "Euler"),
  integrator_fun = NULL,
  interp_option = 2L,
  max_iter = 1L,
  conv_tol = 0.01,
  return_cell_series = TRUE,
  keep_Q17 = TRUE,
  ...
)
```

Arguments

`series`	Data frame of daily watershed inputs. Must include `Date`, `Qi`, `Ci`, `Rain`, `Et`, and `Zcontrol`.
`cells`	List of cell definitions created by `dmsta_make_cell()` and validated with `dmsta_validate_cells()`.
`Nsteps`	Integer. Number of hydrology sub-steps per day.
`N_plant`	Integer. Window length (days) for rolling mean depth used in reservoir penalty blending.
`Qmethod`	Character string specifying the hydrology integrator (`"RK4"`, `"Euler"`, `"RKF45"`, or `"custom"`).
`Pmethod`	Character string specifying the phosphorus integrator (`"RK4"` or `"Euler"`).
`integrator_fun`	Optional custom hydrology integrator function.
`interp_option`	Control-depth interpolation option (DMSTA semantics).
`max_iter`	Integer. Maximum number of network convergence iterations.
`conv_tol`	Numeric. Relative convergence tolerance on external phosphorus loads.
`return_cell_series`	Logical. If `TRUE`, return per-cell daily output series.
`keep_Q17`	Logical. If `TRUE`, retain seep-recycle bookkeeping streams (Q17/L17/C17).
`...`	Additional arguments passed to daily hydrology integration.

Details

This function manages network-level state, routing, lagged recycle bookkeeping, and optional convergence iteration, while delegating per-cell daily physics to `dmsta_flowP_day()`.

Network routing follows DMSTA conventions. Treated outflows are routed either to a downstream cell or out of system, while bypass, release, and seepage discharge streams leave the system immediately. Lagged seepage recycle is tracked explicitly as an internal transit reservoir.

For strict DMSTA parity, use `Qmethod = "RK4"` and `Pmethod = "RK4"` with no operational overrides.

Value

An object of class `"dmsta_network_result"` containing:

results Case-level and optional per-cell daily output series.

budgets Water and phosphorus mass budgets at case and cell level.

meta Convergence diagnostics and model configuration metadata.

Examples

```
# Read data (internal)
data(series)
series <- series[1:370,]; # for example, limit input file

# Data formatting
series$Qi <- cfs_to_hm3d(series$Flow) # cfs to hm3/d
series$Rain <- in_to_m(series$Rainfall) # inches to meters per day
series$Et <- in_to_m(series$ET)
series$Zcontrol <- 0/100 # meters; setting to zero to see what happens
# If you have release series; otherwise set to 0
series$Qr0 <- 0   # constrained outflow (forced Q) if used
series$Qr1 <- 0   # release 1
series$Qr2 <- 0   # release 2
series$Ci <- series$Conc

# input parameters
# --- 1) Base hydrology params (shared structure) ---
hydro_base <- list(
  A_cell = 2.19, # km2
  # depths in cm (engine converts /100 to meters internally)
  Zmin   = 2,             # cm
  Zinit  = 40,            # cm
  Zweir  = 0,             # cm
  Q_zmin = 38,            # cm
  Zrelease = 0,           # cm
  Bypass_elev = 121.92,    # cm (approx 1.2192 m)
  # hydraulics
  Q_a = 1.0,
  Q_b = 4.0,
  Width = 1.55,           # km
  Qomax = 0.0,            # hm3/day; 0 disables max cap in this implementation
  Qimax = 0.0,            # hm3/day; 0 disables inflow cap
```

```
  # seepage (rates in m/day per m head; elevations in cm)
  Seepout_Rate = 0.0,
  Seepout_Elev = 0.0,      # cm
  Seepin_Rate  = 0.0,
  Seepin_Elev  = 0.0,      # cm
  ShutdownET = TRUE,
  force_Q_out = FALSE,
  DutyCycle = 0.95,
  Cmax = 2000
)

# 2) Base P params (shared structure)
P_base <- list(
  # STA module
  C1000 = 22,
  Cstar = 3,
  Ks_per_yr = 16.8,
  Z1 = 40,
  Z2 = 100,
  Z3 = 200,
  Chalf = 300,
  K2Coef1 = 0,
  SeasonalFactor = 0,    # keep 0 for base parity
  # PSTA (NEWS transition)
  Ytrans = 0,
  Ysigma = 0,
  Czero = 0,
  C1000_2 = NULL,
  ks_2 = 0,
  zh_2 = 0,
  # RES depth penalty
  k_depth_penalty = 1,
  # atmos + seepage water quality
  C_rain = 10,           # ppb (ug/L)
  DryDepo = 20,          # mg/m2-yr
  seepage_c = 20,        # ppb cap for seep outflow
  seepin_conc = 0,       # ppb
  # initial P state
  C_init_ppb = 30,
  Y_init_mgm2 = 1000
  )

  #  3) Cell-specific params
  params_cell1 <- modifyList(hydro_base, modifyList(P_base, list(
  Qin_Frac = 0.22,
  Seepout_Rate = 0.00789,
  Ks_per_yr = 16.8,
  Y_init_mgm2 = 3387.67297548954
  )))

  params_cell2 <- modifyList(hydro_base, modifyList(P_base, list(
  Qin_Frac = 0,
  Seepout_Rate = 0.00155,
```

```
Ks_per_yr = 52.5,
Y_init_mgm2 = 768.480041186681
)))

# 4) Build cells
cells <- list(
dmsta_make_cell(
label = "CELL1",
params = params_cell1,
ttankS = 3.0,
DownCell = 2L,
Qin_Frac = params_cell1$Qin_Frac,
RecycleIndex = 1L  # self; can omit if your validator maps NA/0 -> self
),
dmsta_make_cell(
label = "CELL2",
params = params_cell2,
ttankS = 3.0,
DownCell = 0L,
Qin_Frac = params_cell2$Qin_Frac,
RecycleIndex = 2L  # self
)
)

cells <- dmsta_validate_cells(cells)

# Run case
out <- dmsta_flowP_case(
 series = series,
 cells  = cells,
 Nsteps = 4L,
 max_iter = 1L,
 return_cell_series = TRUE,
 keep_Q17 = TRUE
)
head(out$results$case)
head(out$results$cells[[1]])
head(out$results$cells[[2]])
```

`dmsta_flowP_series` *Run a coupled hydrology–phosphorus simulation over a time series*

Description

Simulates DMSTA hydrology and phosphorus dynamics over a multi-day time series for a single cell. This function manages time-series iteration, initialization, rolling diagnostics (e.g., `Z_plant`), and aggregation of daily results, while delegating daily integration to `dmsta_flowP_day()`.

Usage

```
dmsta_flowP_series(
  series,
  params,
  pparams = NULL,
  ttankS = 3,
  Nsteps = 4L,
  N_plant = 30L,
  Qmethod = c("RK4", "Euler", "RKF45", "custom"),
  Pmethod = c("RK4", "Euler"),
  integrator_fun = NULL,
  interp_option = 2L,
  ppar = NULL,
  constants = NULL,
  tanks = NULL,
  V_init = NULL,
  init_P_state = NULL,
  return_steps = FALSE,
  ...
)
```

Arguments

series	Data frame containing daily forcing inputs. Must include at least Date, Qi, Ci, Rain, Et, and Zcontrol. Optional columns include release and recycle terms.
params	List of hydrologic and phosphorus parameters.
pparams	Optional list of phosphorus parameters to merge into params.
ttankS	Numeric. Number of tanks in series (may be fractional).
Nsteps	Integer. Number of hydrology sub-steps per day.
N_plant	Integer. Window length (days) used to compute rolling mean depth for Z_plant.
Qmethod	Character string specifying the hydrology integrator.
Pmethod	Character string specifying the phosphorus integrator.
integrator_fun	Optional custom hydrology integrator function.
interp_option	Integer. Control-depth interpolation option.
ppar	Optional precomputed phosphorus kinetic parameter list.
constants	Optional list of constants used by phosphorus derivatives.
tanks	Optional pre-built tanks-in-series configuration.
V_init	Optional initial volume (hm^3). If NULL, derived from depth initialization rules.
init_P_state	Optional initial phosphorus state. If NULL, initialized from depth and concentration parameters.
return_steps	Logical. If TRUE, store daily hydrology sub-step outputs in the result metadata.
...	Additional arguments passed to daily hydrology integration.

Details

Series-level operational semantics (e.g., HydroIndex-style presence flags, optional release warm-up periods) are applied here and passed downstream as daily inputs.

Value

An object of class `"dmsta_result"` with elements:

results Data frame of daily hydrology and phosphorus outputs.

budgets List containing water and phosphorus mass budget data frames.

meta Metadata describing model configuration, initialization, and methods used.

`dmsta_flow_series` *Run a DMSTA hydrology simulation over a time series*

Description

Simulates hydrologic behavior for a single cell over a multi-day time series, managing initialization, rolling diagnostics, and result aggregation.

Usage

```
dmsta_flow_series(
  series,
  params,
  V_init = NULL,
  Qmethod = c("RK4", "Euler", "RKF45", "custom"),
  Nsteps = 4L,
  integrator_fun = NULL,
  interp_option = 2L,
  ...
)
```

Arguments

`series`	Data frame of daily inputs.
`params`	List of hydrologic parameters.
`V_init`	Optional initial volume.
`Qmethod`	Character. Hydrology integrator.
`Nsteps`	Integer. Sub-steps per day.
`integrator_fun`	Optional custom integrator.
`interp_option`	Control-depth interpolation mode.
`...`	Additional options.

Value

Object of class "dmsta_result".

Examples

```
# Read data (internal)
data(series)
series <- series[1:370,]; # for example, limit input file

# Data formatting
series$Qi <- cfs_to_hm3d(series$Flow) # cfs to hm3/d
series$Rain <- in_to_m(series$Rainfall) # inches to meters per day
series$Et <- in_to_m(series$ET)
series$Zcontrol <- 0/100 # meters; setting to zero to see what happens
# If you have release series; otherwise set to 0
series$Qr0 <- 0   # constrained outflow (forced Q) if used
series$Qr1 <- 0   # release 1
series$Qr2 <- 0   # release 2
series$Ci <- series$Conc

# input parameters
params <- list(
 A_cell = 2.19, # km2
 Zmin   = 2,              # cm
 Vmin   = 0,              # hm3
 Q_a = 1.0, # qcoef_a; discharge coef
 Q_b = 4.0, # qcoef_b; discharge exponent
 Zweir = 0, # cm; qcoef_offset; depth offset for outflow computation
 Q_zmin = 38,  # cm; qcoef_zmin
 Qomax = 0.0, # maximum discharge  hm3/day
 Qimax = 0, # maximum inflow  (hm3/day)
 Width = 1.55, # km
 Bypass_elev = 121.92, # z_byp; mean depth at which bypass begins (m) from input (cm)
 Seepout_Rate = 0.00789, # outflow seepage rate per unit head (m/day)/m from input cm/d/cm
 Seepout_Elev = 0.0, # elevation controling outflow seepage rate (input cm)
 Seepin_Rate  = 0.0,
 Seepin_Elev  = 0.0,
 ShutdownET = TRUE,
 force_Q_out = FALSE,
 wrap_interp = TRUE,
 Zinit = 40,  # cm; initial water column depth
 Qin_Frac = 0.22, # inflow_frac; fraction of basin flows going into this cell
 Zrelease = 0,   # cm; z_release; minimum depth for releases
 RecycleQ = 0,
 IsaNode = NULL,
 enable_P_release = FALSE,
 K_release = 0,
 IsaNode = NULL
)

 V_init <- (cm_to_m(params$Zinit) * params$A_cell)
 out <- dmsta_flow_series(V_init, series, params, Nsteps = 4)
```

```
out$results
```

`dmsta_init_case_state` *Initialize per-cell state for a network/case run*

Description

Builds tank geometry and initializes hydrologic and phosphorus state for each cell using the cell parameters:

- `V[ic]` is initialized as `A_cell * (Zinit/100)` (depth cm -> m),
- tank geometry is built using `dmsta_build_tanks(A_cell, ttankS)`,
- phosphorus states `M` and `S` are initialized with `dmsta_p_init_state()` using `C_init_ppb` and `Y_init_mgm2`.

Usage

```
dmsta_init_case_state(cells)
```

Arguments

`cells`	A validated list of cell definitions. Each cell must contain `$params` (with `A_cell`, `Zinit`, `C_init_ppb`, `Y_init_mgm2`) and `$ttankS`.

Value

A list with elements:

V Numeric vector of initial volumes, length = number of cells.

tanks List of per-cell tank geometry objects.

Pstate List of per-cell phosphorus state objects (each with `M` and `S` vectors).

`dmsta_make_cell` *Create a DMSTA network cell definition*

Description

Constructs a cell definition used by `dmsta_flowP_case()` network/case simulations. The function attaches:

- per-cell tank specification `ttankS`,
- routing fields (`DownCell`, `Qin_Frac`, `RecycleIndex`),
- optional splitter specification (`SplitterFrac`),
- per-cell kinetics parameters (`ppar`) using built-in modules (`STA`, `PSTA`, `RES`),
- per-cell constants list used by the phosphorus derivative.

Usage

```
dmsta_make_cell(
  label,
  params,
  ttankS,
  DownCell = 0L,
  Qin_Frac = 0,
  RecycleIndex = NULL,
  SplitterFrac = NULL
)
```

Arguments

`label`	Character scalar. Cell label (use `"SPLITTER"` for the splitter cell if your case uses one).
`params`	Named list of per-cell parameters (must include at least `A_cell`, `DutyCycle`, `Zinit`, `C_init_ppb`, `Y_init_mgm2`, plus fields required by the selected kinetics builders).
`ttankS`	Numeric scalar > 0. Effective number of tanks (may be fractional).
`DownCell`	Integer index of the downstream cell (0 means terminal/out of system).
`Qin_Frac`	Numeric scalar. Fraction of watershed inflow assigned to this cell in network routing (case runner may override internally for "already allocated" inflows).
`RecycleIndex`	Integer index of the cell receiving recycled seepage flow/mass. `0` or `NA` indicates default handling (often self).
`SplitterFrac`	Optional. Splitter routing fractions (only used if `label` is `"SPLITTER"`). May be a numeric vector (length = number of cells) or a named vector/list mapping downstream indices to fractions.

Value

A named list representing a cell definition, suitable for inclusion in the `cells` argument of `dmsta_flowP_case()`.

See Also

`dmsta_flowP_case()` for running a multi-cell simulation.

---

dmsta_p_init_state *Initialize DMSTA phosphorus state vectors for tanks*

Description

Internal helper that initializes per-tank state vectors for the DMSTA phosphorus module given tank geometry and initial conditions.

Usage

```
dmsta_p_init_state(tanks, Z_init_m, C_init_ppb, Y_init_mgm2)
```

Arguments

`tanks`	A list as returned by `dmsta_build_tanks()` containing `Ntanks` and `A_Tank`.
`Z_init_m`	Numeric scalar. Initial water column depth (meters).
`C_init_ppb`	Numeric scalar. Initial concentration (ppb).
`Y_init_mgm2`	Numeric scalar. Initial areal mass/loading (mg/m^2).

Details

For each tank `i`, the function computes:

- `M[i]` = `C_init_ppb` * `A_Tank[i]` * `Z_init_m`
- `S[i]` = `Y_init_mgm2` * `A_Tank[i]`

where `A_Tank[i]` is the area of tank `i`. Units are assumed to be consistent with the DMSTA implementation (e.g., depth in meters).

Value

A named `list` with elements:

M Numeric vector (length `tanks$Ntanks`) of initialized water-column masses.

S Numeric vector (length `tanks$Ntanks`) of initialized sediment/areal stores.

See Also

`dmsta_build_tanks()` to generate tank geometry.

dmsta_validate_cells *Validate and normalize network cell definitions (internal)*

Description

Checks that `cells` is a non-empty list of cell definitions and that each cell contains the minimum required fields for a coupled hydrology + phosphorus simulation.

Usage

```
dmsta_validate_cells(cells)
```

Arguments

`cells`	A list of cell definitions (as created by `dmsta_make_cell()` or assembled manually).

Details

The validator enforces:

- required structural fields: `params`, `ttankS`, `ppar`, `constants`
- required `ppar` fields used by the phosphorus derivative
- required `constants` fields used by the phosphorus derivative
- DMSTA convention for recycle indices (0 means self)
- at most one splitter cell labeled `"SPLITTER"` (case-insensitive)

The function also fills defaults for some routing fields if missing.

Value

The normalized `cells` list (with defaults filled in). Invisibly returns `cells` but typically used as `cells <- dmsta_validate_cells(cells)`.

model_parameter_builders *Build parameter lists for supported DMSTAr models*

Description

These functions convert a raw parameter list (`pparams`) into a standardized parameter list used by DMSTAr model implementations.

Usage

```
build_STA(Dpy, DutyCycle, pparams, ...)

build_PSTA(Dpy, DutyCycle, pparams, ...)

build_RES(Dpy, DutyCycle, pparams, ...)
```

Arguments

Dpy	Number of time steps per year (e.g., 365 for daily).
DutyCycle	Fraction of time active in each step (0–1).
pparams	A named list of raw parameters for the selected model.
...	Reserved for future extensions; ignored.

Details

Builders perform unit conversions and time-step scaling:

- rate parameters per year are converted to per-timestep using Dpy
- depth parameters in centimeters are converted to meters
- duty cycle is applied to rate parameters where applicable

Value

A named list of standardized model parameters.

Functions

- build_STA(): Build parameters for the STA model.
- build_PSTA(): Build parameters for the PSTA model.
- build_RES(): Build parameters for the RES model.

---

series *Example DMSTA daily forcing time series*

Description

A example time series of daily hydrology and phosphorus forcing inputs suitable for demonstrating DMSTAr case and cell simulations.

Usage

```
series
```

Format

A data frame with daily records and the following columns:

Date Date for the record. Stored as a character vector in "YYYY-MM-DD" format in the raw object shown; you may convert to Date with as.Date().

Flow Daily inflow/flow rate (cfs).

Conc Daily inflow concentration (e.g., ug/L).

Rainfall Daily rainfall rate (e.g., in/day).

ET Daily evapotranspiration rate (e.g., in/day).

Details

This dataset is intended as a lightweight example for testing and documentation. For use with functions that expect columns named Qi, Ci, and Rain, you may need to rename columns (e.g., Flow -> Qi, Conc -> Ci, Rainfall -> Rain).

Source

Source workbook: PROJECT_SFWMD_EC_01MAR2012_NET_EAA_STA1E.xls

sheet Series_Input

Examples

```
data(series)

# Convert Date column (if needed)
series$Date <- as.Date(series$Date)

# If your simulation expects Qi/Ci/Rain column names:
sim_series <- within(series, {
  Qi   <- Flow
  Ci   <- Conc
  Rain <- Rainfall
})
sim_series$Flow <- NULL
sim_series$Conc <- NULL
sim_series$Rainfall <- NULL

head(sim_series)
```

---

validate_P_paramsK *Validate alignment of K-length phosphorus parameter vectors*

Description

Internal helper that checks whether the K-length parameter vectors in params are aligned with params$Kslots.

Usage

```
validate_P_paramsK(params)
```

Arguments

params	A named list containing Kslots and K-length vectors: K1, K2, K3, Chalf, Z_1, Z_2, Z_3, and K2Coef.

Value

Invisibly returns TRUE if validation passes; otherwise throws an error.

Index