# Package: kwb.vs2dh (via r-universe)

October 12, 2024

**Version** 0.0.0.9000

**Title** Interface to VS2DH

**Description** Interface to the open-source model USGS model VS2DH for simulation of water and energy transport in variable-saturated porous media (for more information see: http://wwwbrr.cr.usgs.gov/projects/GW_Unsat/vs2di1.3/).

**Depends** R (>= 3.0.0), stringr, lattice, colorRamps, kwb.utils, kwb.db

**Remotes** github::kwb-r/kwb.utils, github::kwb-r/kwb.db

**BinaryFiles** inst/extdata/engine/linux/vs2dh3_3.exe, inst/extdata/engine/win/vs2dh3_3.exe, inst/extdata/engine/win/vs2dt3_3.exe

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyLoad** yes

**URL** https://github.com/KWB-R/kwb.vs2dh/

**BugReports** https://github.com/KWB-R/kwb.vs2dh/issues

**RoxygenNote** 6.1.0

**Suggests** testthat

**Repository** https://kwb-r.r-universe.dev

**RemoteUrl** https://github.com/KWB-R/kwb.vs2dh

**RemoteRef** HEAD

**RemoteSha** e57d86cad48e66a1093db001ae94e8c52fe170c8

## Contents

---

`checkOperatingSystem` *Helper function: checks whether operating system is windows or linux*

---

## Description

Helper function: checks whether operating system is windows or linux

## Usage

```
checkOperatingSystem()
```

## Examples

```
os <- checkOperatingSystem() ### if function shell() is not available -> linux"
os ### return operating system
```

---

`convBasic` *Helper function: convert basic config to FORTRAN style*

---

## Description

Helper function: convert basic config to FORTRAN style

## Usage

```
convBasic(baseConf, dbg = TRUE)
```

## Arguments

| | |
|---|---|
| `baseConf` | with basic parameterisation, i.e. conf$basic |
| `dbg` | If TRUE debug messages are printed on the screen (default: TRUE) |

## Value

Basic configuration in FORTRAN style

`convertStringVectorToMatrix` *Helper function: convert string vector to numeric matrix*

## Description

Helper function: convert string vector to numeric matrix

## Usage

```
convertStringVectorToMatrix(values, splitSep = " ", byrow = TRUE,
  rows = length(values), toNumeric = TRUE)
```

## Arguments

| | |
|---|---|
| `values` | character vector to be converted to matrix |
| `splitSep` | seperator used as split parameter in function strsplit (Default: " ") |
| `byrow` | If TRUE matrix is filled by rows first. Default: TRUE |
| `rows` | for constructing matrix: Default: length(values) |
| `toNumeric` | If true values will be converted to numeric, if FALSE not |

## Value

Returns matrix with nrows equals to length of values input vector

`convertToWindowsPath` *Helper function: converts string to windows path*

## Description

Helper function: converts string to windows path

## Usage

```
convertToWindowsPath(path, sep = "\\/")
```

## Arguments

| | |
|---|---|
| `path` | path string to be converted to windows path |
| `sep` | seperator which default: "/", which is used on unix and for R) |

## Examples

```
unixPath <- "/usr/model/vs2dh"
winPath <- convertToWindowsPath(path=unixPath)
winPath
```

convInitial *Helper function: converting initial conditions to FORTRAN style*

## Description

Helper function: converting initial conditions to FORTRAN style

## Usage

```
convInitial(iniConf, dbg = TRUE)
```

## Arguments

| | |
|---|---|
| iniConf | config with soils parameterisation, i.e. conf$initial |
| dbg | If TRUE debug messages are printed on the screen (default: TRUE) |

## Value

Soil configuration in FORTRAN style

convMatrixByRowToString *Helper function: converts matrix to one string*

## Description

Helper function: converts matrix to one string

## Usage

```
convMatrixByRowToString(matr, colSep = " ", rowSep = "\n")
```

## Arguments

| | |
|---|---|
| matr | matrix with character data |
| colSep | seperator used for merging data from all columns for each row (default: " ") |
| rowSep | seperator used for separating the data of different rows (default: "newline") |

## Value

One string with each row separated with rowSep

## convRecharge *Helper function: converting recharge periods to FORTRAN style*

### Description

Helper function: converting recharge periods to FORTRAN style

### Usage

```
convRecharge(rechConf, dbg = TRUE)
```

### Arguments

| | |
|---|---|
| rechConf | config with recharge period parameterisation, i.e. conf$recharge |
| dbg | If TRUE debug messages are printed on the screen (default: TRUE) |

### Value

Recharge periods configuration in FORTRAN style

## convSoils *Helper function: converting soils structure to FORTRAN style*

### Description

Helper function: converting soils structure to FORTRAN style

### Usage

```
convSoils(conf, dbg = TRUE)
```

### Arguments

| | |
|---|---|
| conf | config with R model parameterisation, i.e. conf |
| dbg | If TRUE debug messages are printed on the screen (default: TRUE) |

### Value

Soil configuration in FORTRAN style

## cutStringByPattern *Helper function: Cut string by pattern*

### Description

Helper function: Cut string by pattern

### Usage

```
cutStringByPattern(pattern, endString = TRUE, txt)
```

### Arguments

| | |
|---|---|
| pattern | pattern to be searched for in string txt; if pattern is a vector of two the string is cut between pattern[1] and pattern[2] |
| endString | if TRUE the string after the search pattern is returned, if FALSE from the beginning until the search pattern (only evaluated if pattern pattern is vector of length 1!) |
| txt | string or vector of strings to be searched for in string |

### Value

Returns only partial string(s) depending on the cut pattern and until the end of each string

### Examples

```
### Path to your vs2dh model directory
string = "A B C D"
pattern = "B "
cutStringByPattern(pattern=pattern, txt=string)
```

## filterLines *Helper function: filtering lines*

### Description

Helper function: filtering lines

### Usage

```
filterLines(pattern, patternNameCol = "txt", data, startLineOffset = 0,
  input, lineCol = "Line")
```

### Arguments

| | |
|---|---|
| pattern | pattern which lines are searched |
| patternNameCol | for pattern like "/B25" use: "txt", for para names use "paraNames", (Default: "txt") |
| data | as retrieved by prepareImport()$res |
| startLineOffset | offset of starting line: 0 (if parName is in same row as parVal, 1 if parVal is one row after parName) |
| input | as retrieved by prepareImport()$input |
| lineCol | Default "Line" |

---

## fortranFormat *Helper function: formatting R elments to FORTRAN style*

---

### Description

Helper function: formatting R elments to FORTRAN style

### Usage

```
fortranFormat(conf)
```

### Arguments

| | |
|---|---|
| conf | as retrieved by vs2dhConfigure() |

### Value

config with prepared FORTRAN formatting style (only TRUE/FALSE values will be replaced during vs2dh.writeConfig()

---

## importMatrices *Helper function: imports matrices (jtex, hvalues, tvalues)*

---

### Description

Helper function: imports matrices (jtex, hvalues, tvalues)

### Usage

```
importMatrices(prepData, dbg = TRUE)
```

### Arguments

| | |
|---|---|
| prepData | object as retrieved by prepareImport() |
| dbg | prints debug information on the screen |

### Value

Matrix values (always: soil, if available: initial pressure head & temperature distribution) model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh") inp <- prepareImport(model.path) grid <- importMatrices(inp) #### Soil properties matrix vs2dh.plotMatrix(data = grid$jtex) #### Initial temperature distribution matrix vs2dh.plotMatrix(data = grid$tvalues)

---

## importMatrix *Helper function: imports matrix values of vs2dh.dat*

---

### Description

Helper function: imports matrix values of vs2dh.dat

### Usage

```
importMatrix(prepData, pattern = "JTEX", patternNameCol = "parNames",
  startLineOffset = 0, msg = "B9: Importing soil matrix...",
  nrow = NULL, dbg = TRUE)
```

### Arguments

| | |
|---|---|
| prepData | object as retrieved by prepareImport() |
| pattern | as used in filterLines() |
| patternNameCol | as used in filterLines() |
| startLineOffset | as used in filterLines() |
| msg | message which parameter is imported. used if DBG == TRUE |
| nrow | number of rows of matrix: Default: nly |
| dbg | If TRUE prints debug information on the screen (Default: TRUE) |

### Value

Matrix values (always: soil, if available: initial pressure head & temperature distribution)

## importRechargePeriods *Helper function: imports recharge periods*

### Description

xxxxxx

### Usage

```
importRechargePeriods(prepData)
```

### Arguments

| | |
|---|---|
| prepData | object as retrieved by prepareImport() |

### Value

Recharge periods (i.e. PART C) with naming parameter names in R style

## importSingleLineParas *Helper function: imports single line parameters*

### Description

FORTRAN parameters with always correspond to only one value

### Usage

```
importSingleLineParas(prepData, dbg = TRUE)
```

### Arguments

| | |
|---|---|
| prepData | object as retrieved by prepareImport() |
| dbg | prints debug information on the screen |

### Value

Single line values with naming parameter names in R style (before part C, i.e. recharge period 1 starts)

`labeledList` *Helper function: replace a list index with a name*

## Description

Helper function: replace a list index with a name

## Usage

```
labeledList(inpLst, labelTxt = "id")
```

## Arguments

| | |
|---|---|
| `inpLst` | input list |
| `labelTxt` | user defined txt used for naming list |

## Value

Labeled list

`leafValues` *Helper function: names of all sublists of a list returns the names of all sublists of* x *in the "$"-notation, e.g. list$sublist$subsublist$subsubsublist*

## Description

Helper function: names of all sublists of a list returns the names of all sublists of *x* in the "$"-notation, e.g. list$sublist$subsublist$subsubsublist

## Usage

```
leafValues(x, values = list())
```

## Arguments

| | |
|---|---|
| `x` | R list. |
| `values` | name to be used as prefix for all names found. default: "" |

## Value

leafs of sublists into one dimensional list

## multipleLineValues *Helper function: multipleLineValues*

### Description

Writes one vector for values in multiple lines that beginn in FORTRAN "in the next line" (e.g. DELZ, DXR)

### Usage

```
multipleLineValues(parName = "DELZ", data, patternNameCol = "parNames",
  startLineOffset = 1, toNumeric = TRUE)
```

### Arguments

| | |
|---|---|
| parName | e.g. "PLTIM", "DELZ" or "DXR" |
| data | as retrieved by prepareImport() |
| patternNameCol | for pattern like "/B25" use: "txt", for para names use "paraNames", (Default: "txt") -> used by function: filterLines() |
| startLineOffset | offset of starting line: 0 (if parName is in same row as parVal, 1 if parVal is one row after parName), -> used by function: filterLines() |
| toNumeric | If TRUE result is converted to numeric (Default: TRUE) |

### Value

List with one vector of multiple lines values for each parameter

## nodePairs *Helper function: node pairs*

### Description

Helper function: node pairs

### Usage

```
nodePairs(j, n)
```

### Arguments

| | |
|---|---|
| j | Row of each cell |
| n | Column of each cell |

### Value

data.frame with node pairs

patternSelect *Helper function: selects pattern contained in different lines of a string vector (e.g. imported using readLines())*

## Description

Helper function: selects pattern contained in different lines of a string vector (e.g. imported using readLines())

## Usage

```
patternSelect(pattern, data)
```

## Arguments

| | |
|---|---|
| pattern | index of row to be split (valid values: 1-3) |
| data | vector with strings |

## Value

data.frame with columns lines (indicating the lines where pattern was found and txt (string contained in that lines)

prepareImport *Helper function: prepareImport (imports and prepares input file)*

## Description

Helper function: prepareImport (imports and prepares input file)

## Usage

```
prepareImport(model.path, mdb.path = system.file("extdata",
  "InputDescription.xls", package = "kwb.vs2dh"), dbg = TRUE)
```

## Arguments

| | |
|---|---|
| model.path | full path to folder containing vs2dh.dat |
| mdb.path | full path to the folder containing the InputDescription.xls which is used as lookup table for converting FORTRAN to R parameter names (Default: system.file("extdata", "InputDescription.xls", package = "kwb.vs2dh")) |
| dbg | if TRUE text output is printed on the screen |

## Value

Read vs2dhi.dat and does some preparations steps which are stores in a list which contains the sub-lists "input" (raw imported vs2dh.dat file), "inpDescription" (lookup table defined in file "mdb.path"), "res" (prepared table with parameters names and values) and "rechargePeriods" (summary of start/endlines of file "input" for each recharge period)

## Examples

```
## Not run:
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
prepareImport(model.path)

## End(Not run)
```

---

`splitHeader` *Helper function: splits header of output file with (energy, fluid) mass balance time series (balance.out)*

---

## Description

Helper function: splits header of output file with (energy, fluid) mass balance time series (balance.out)

## Usage

```
splitHeader(row, header)
```

## Arguments

| | |
|---|---|
| `row` | index of row to be split (valid values: 1-3) |
| `header` | first three lines of balance.out containing header |

## Value

splitted header

## vs2dh.Configure *Configure complete vs2dh parameterisation*

### Description

Configure complete vs2dh parameterisation

### Usage

```
vs2dh.Configure(basic = vs2dh.ConfigureBasic(),
  soils = vs2dh.ConfigureSoils(), initial = vs2dh.ConfigureInitial(),
  recharge = vs2dh.ConfigureRechargePeriods())
```

### Arguments

| | |
|---|---|
| basic | parameterisation (title, units, time, output, solver, grid, options), as retrieved by vs2dh.ConfigureBasic(), |
| soils | retrieved by vs2dh.ConfigureSoils() |
| initial | initial conditions for flow (pressure head or moisture contents and transport (temperatures), as retrieved by vs2dh.ConfigureInitial() |
| recharge | recharge periods as retrieved by vs2dh.ConfigureRechargePeriods() |

### Value

Complete vs2dh parameterisation in R style

## vs2dh.ConfigureBalance *Configure balance output file ("balance.out")*

### Description

Configure balance output file ("balance.out")

### Usage

```
vs2dh.ConfigureBalance(mb9 = c(1:72), outputEachTimeStep = TRUE)
```

### Arguments

| | |
|---|---|
| mb9 | The index number of each mass balance component to be written to file "balance.out".(See table 7, from p. 66, in Healy (1990) |
| outputEachTimeStep | FALSE if output to file 9 ("balance.out") is desired only at selected output times rather than at each time step (default: TRUE) |

## Value

Selected components and temporal resolution for "balance.out" file

---

`vs2dh.ConfigureBasic` *Configure basic model parameters*

---

## Description

Configure basic model parameters

## Usage

```
vs2dh.ConfigureBasic(titl = "My title",
  units = vs2dh.ConfigureBasicUnits(),
  time = vs2dh.ConfigureBasicTime(),
  output = vs2dh.ConfigureBasicOutput(),
  solver = vs2dh.ConfigureBasicSolver(),
  grid = vs2dh.ConfigureBasicGrid(),
  options = vs2dh.ConfigureBasicOptions())
```

## Arguments

| | |
|---|---|
| `titl` | 80-character problem description (formatted read, 20A4) |
| `units` | as retrieved by vs2dh.ConfigureBasicUnits() |
| `time` | as retrieved by vs2dh.ConfigureBasicTime() |
| `output` | as retrieved by vs2dh.ConfigureBasicTime() |
| `solver` | retrieved by vs2dh.ConfigureBasicSolver() |
| `grid` | retrieved by vs2dh.ConfigureBasicGrid() |
| `options` | as simulation options as retrieved by vs2dh.ConfigureBasicOptions (default: TRUE) |

## Value

Basic configuration

## vs2dh.ConfigureBasicGrid

*Configure Grid*

### Description

Configure Grid

### Usage

```
vs2dh.ConfigureBasicGrid(nxr = 46, nly = 34, ang = 0, rad = FALSE,
  dx = 0.5, dy = 0.5)
```

### Arguments

| | |
|---|---|
| nxr | Number of cells in horizontal or radial direction. (default: 46) |
| nly | Number of cells in vertical direction. (default: 34) |
| ang | Angle by which grid is to be tilted (Must be between -90 and +90 degrees, ang = 0 for no tilting, see Healy (1990) for further discussion), degrees. (default: 0) |
| rad | Logical variable. TRUE if radial coordinates are used; otherwise = FALSE (default: FALSE) |
| dx | Constant value for grid spacing in horizontal or radial direction. (default: 0.5) |
| dy | Constant value for grid spacing in vertical direction. (default: 0.5) |

### Value

Grid parameterisation

## vs2dh.ConfigureBasicOptions

*Configure basic simulation options: energy transport, evaporation or evapotranspiration should be simulated? (FUNCTIONALITY NOT IMPLEMENTED YET! DO NOT CHANGE default PARAMETERISATION)*

### Description

Configure basic simulation options: energy transport, evaporation or evapotranspiration should be simulated? (FUNCTIONALITY NOT IMPLEMENTED YET! DO NOT CHANGE default PARAMETERISATION)

### Usage

```
vs2dh.ConfigureBasicOptions(trans = TRUE, bcit = FALSE,
  etsim = FALSE)
```

### Arguments

| | |
|---|---|
| trans | Should energy transport be simulated? (default: TRUE), Option FALSE currently NOT IMPLEMENTED YET!!!!!!! |
| bcit | NOT IMPLEMENTED YET!!!!!!! If TRUE evaporation is to be simulated for this simulation;if FALSE no evaporation is simulated for this simulation (default: FALSE) |
| etsim | NOT IMPLEMENTED YET!!!!!!! If TRUE evapotanspiration (plant-root extraction) is to be simulated for this simulation; if FALSE no evaporation is simulated for this simulation (default: FALSE). |

### Value

Return basic simulation options

---

## vs2dh.ConfigureBasicOutput

*Configure model output*

---

### Description

Configure model output

### Usage

```
vs2dh.ConfigureBasicOutput(times = vs2dh.ConfigureBasicOutputTimes(),
  main = vs2dh.ConfigureBasicOutputMain(), variables = TRUE,
  balance = vs2dh.ConfigureBalance(),
  obsPoints = vs2dh.ConfigureObsPoints(),
  boundaryFluxes = vs2dh.ConfigureBoundaryFluxes(),
  enhancedPrecision = FALSE)
```

### Arguments

| | |
|---|---|
| times | as retrieved by `vs2dh.ConfigureBasicOutputTimes()` |
| main | as retrieved by `vs2dh.ConfigureBasicOutputMain()` |
| variables | If TRUE output of pressure heads (and temperatures if TRANS = T) to file "variables.out" is desired at selected observation times (default: TRUE) |
| balance | IF TRUE one-line mass balance summary for each time step is written to file "balance.out"; (default: TRUE) |
| obsPoints | as retrieved by vs2dh.ConfigureObsPoints(), default: no output of observation points to file "obsPoints.out" |
| boundaryFluxes | as retrieved by vs2dh.ConfigureBoundaryFluxes(), default: no output of boundary fluxes to file "boundaryFluxes.out" |
| enhancedPrecision | If TRUE results are written with enhanced precision to files9 ("variables.out") and file11 ("obsPoints.out"), (default: FALSE) |

## Value

Configuration of model output

---

`vs2dh.ConfigureBasicOutputMain` *Configure main output file ("vs2dh.out")*

---

## Description

Configure main output file ("vs2dh.out")

## Usage

```
vs2dh.ConfigureBasicOutputMain(f6p = TRUE, thpt = TRUE, spnt = TRUE,
  ppnt = TRUE, hpnt = TRUE, vpnt = TRUE)
```

## Arguments

| | |
|---|---|
| `f6p` | If TRUE mass balance is to be written to file "vs2dt.out" for each time step; if FALSE mass balance is to be written to file "vs2dt.out" only at observation times and ends of recharge periods (default: TRUE) |
| `thpt` | If TRUE volumetric moisture contents are to be written to file "vs2dt.out", otherwise: FALSE (default: TRUE) |
| `spnt` | If TRUE saturation is written to file "vs2dt.out", otherwise: FALSE (default: TRUE) |
| `ppnt` | If TRUE pressure heads are written to file "vs2dt.out", otherwise: FALSE (default: TRUE) |
| `hpnt` | If TRUE total heads are written to file "vs2dt.out", otherwise: FALSE (default: TRUE) |
| `vpnt` | If TRUE velocities are written to file "vs2dt.out", otherwise: FALSE (default: TRUE) |

## Value

Output config of vs2dh.out file

## vs2dh.ConfigureBasicOutputTimes

*Configure model output times*

### Description

Configure model output times

### Usage

```
vs2dh.ConfigureBasicOutputTimes(pltim = seq(0, 10, 0.5), numt = 1000,
  enhancedPrecision = FALSE, dbg = TRUE)
```

### Arguments

| | |
|---|---|
| pltim | Elapsed times at which pressure heads and temperatures are to be written to file "variables.out", and heads, temperatures, saturations, velocities, and/or moisture contents to file "vs2dt.out", T. (default: seq(0,10,0.5)) |
| numt | Maximum number of time steps. (default: 1000), will be set automatically if called from higher level function vs2dh.ConfigureBasic() |
| enhancedPrecision | if TRUE enhanced results are written with enhanced precision to files6 (vs2dh.out) & file9 (variables.out), (default: FALSE) |
| dbg | if TRUE additional information printed on screen (default: TRUE) |

### Value

Model output times

## vs2dh.ConfigureBasicSolver

*Basic solver configuration*

### Description

Basic solver configuration

### Usage

```
vs2dh.ConfigureBasicSolver(cis = TRUE, cit = TRUE, numt = 1000,
  minit = 2, itmax = 80, eps = 1e-04, eps1 = 0.001, eps2 = 0.001,
  hmax = 0.7, itstop = TRUE)
```

### Arguments

| | |
|---|---|
| cis | If TRUE spatial discretisation is realised by centered-in-space differencing; if FALSE backward-in-space differencing is to be used for transport equation. (default: TRUE) |
| cit | If TRUE temporal discretisation is realised by centered-in-time differencing; if FALSE backward-in-time or fully implicit differencing is to be used. (default: TRUE) |
| numt | Maximum number of time steps.(default: 1000). (NOTE: if enhanced precision in print out to file "balance.out" and file 11 "obsPoints.out", is desired set NUMT equal to a negative number. That is, multiply actual maximum number of time steps by -1)1 |
| minit | Minimum number of iterations per time step. (default: 2) |
| itmax | Maximum number of iterations per time step. (default: 80) |
| eps | Head closure criterion for iterative solution of flow equation, L. (default: 0.0001) |
| eps1 | Temperature closure criterion for iterative solution of transport equation, C. (default: 0.001) |
| eps2 | Velocity closure criterion for outer iteration loop at each time step, L/T. (default: 0.001) |
| hmax | Relaxation parameter for iterative solution. See discussion in Lappala and others (1987) for more detail. Value is generally in the range of 0.4 to 1.2. (default: 0.7) |
| itstop | If TRUE simulation is terminated after ITMAX iterations in one time step; otherwise = F. (default: TRUE) |

### Value

Configuration of basic solver

---

## vs2dh.ConfigureBasicTime

*Configure time parameters*

---

### Description

Configure time parameters

### Usage

```
vs2dh.ConfigureBasicTime(stim = 0, tmax = 10.1)
```

### Arguments

| | |
|---|---|
| stim | Initial time (usually set to 0). (default: 0) |
| tmax | Maximum simulation time. (default: 10.1) |

## Value

Time parameters

---

`vs2dh.ConfigureBasicUnits` *Configure unit parameters*

---

## Description

Configure unit parameters

## Usage

```
vs2dh.ConfigureBasicUnits(zunit = "m", tunit = "day", cunx = "J")
```

## Arguments

| | |
|---|---|
| `zunit` | Units used for length, "m" for meters. (default: "m") |
| `tunit` | Units used for time, "sec" for seconds, "day" for day (default: "day") |
| `cunx` | Units used for heat, "J" for Joules. (default: "J") |

## Value

Model units

---

`vs2dh.ConfigureBoundaryCondition` *Configure boundary conditions*

---

## Description

Configure boundary conditions

## Usage

```
vs2dh.ConfigureBoundaryCondition(jj = 5, nn = 17:26, ntx = 2,
  pfdum = 0.009778035, ntc = 0, cf = 10, importData = NULL,
  ibc = 0)
```

### Arguments

| | |
|---|---|
| jj | vector with row numbers of nodes |
| nn | vector with column numbers of nodes |
| ntx | vector with node type identifier for boundary conditions. 0 (for no specified boundary (needed for resetting some nodes after initial recharge period); 1 (for specified pressure head); 2 (for specified flux per unit horizontal surface area in units of L/T); 3 (for possible seepage face); 4 (for specified total head); 5 (for evaporation, Note: is not implemented yet!); 6 (for specified volumetric flow in units of L3/T). 7 (for gravity drain). (The gravity drain boundary condition allows gravity driven vertical fow out of the domain assuming a unit vertical hydraulic gradient. Flow into the domain cannot occur.)" |
| pfdum | vector with specified head for NTX = 1 or 4 or specified flux for NTX = 2 or 6. If codes 0, 3, 5, or 7 are specified, the line should contain a dummy value for PFDUM or should be terminated after NTX by a blank and a slash (/). |
| ntc | vector with node type identifier for transport boundary conditions. 0 (for no specified boundary); 1 (for specified temperatures)" |
| cf | vector with specified temperature for NTC = 1 or NTX = 1, 2, 4, 6, or 7. Present only if TRANS = T. |
| importData | Optionally a data.frame with the colnames (jj,nn,ntx,pfdum,ntc,cf) can be used. In this case the input for the function parameters (jj,nn,ntx,pfdum,ntc,cf) will be ignored! |
| ibc | Code for reading in boundary conditions: 0 (by individual node), 1 (by row/column). (default: 0), Note: currently only option 0 is implemented |

### Value

Boundary conditions

---

## vs2dh.ConfigureBoundaryFluxes

*Configure Boundary fluxes*

---

### Description

Configure Boundary fluxes

### Usage

```
vs2dh.ConfigureBoundaryFluxes(nodes = NULL)
```

### Arguments

| | |
|---|---|
| nodes | data.frame with columns "idbf" (id of boundary face), "bf_j" (grid row) and "bf_n" (grid column) |

## Value

Boundary fluxes parameterisation

## Examples

```
nodes <- data.frame(idbf = c(rep(1,5), rep(2,6)),
                    bf_j = 1:11,
                    bf_n = rep(1,11))
vs2dh.ConfigureBoundaryFluxes(nodes = nodes)
```

---

`vs2dh.ConfigureGenuchten`

*Configure Genuchten flow parameters*

---

## Description

Configure Genuchten flow parameters

## Usage

```
vs2dh.ConfigureGenuchten(ratioKzKh = 1, satKh = 750, ss = 0,
  porosity = 0.39, alpha = 2.3, rmc = 0.05, beta = 5.8)
```

## Arguments

| | |
|---|---|
| `ratioKzKh` | Ratio of hydraulic conductivity in the z-coordinate direction to that in the x-coordinate direction (Default:1) |
| `satKh` | Saturated hydraulic conductivity (K) at 20 C in the x-coordinate direction for class ITEX, L/T. (Default: 750 m/d) |
| `ss` | Specific storage (Ss), L^-1. (Default: 0) |
| `porosity` | Porosity (f), (Default: 0.39) |
| `alpha` | van Genuchten alpha. NOTE: alpha is as defined by van Genuchten (1980) and is the negative reciprocal of alpha' used in earlier versions (prior to version 3.0) of VS2DT, L., (Default: 2.3/m) |
| `rmc` | Residual moisture content, (Default: 0.05) |
| `beta` | van Genuchten parameter, beta' in Healy (1990) and Lappala and others (1987), (Default:5.8) |

## Value

Genuchten flow parameters

## See Also

http://pubs.usgs.gov/circ/2003/circ1260/pdf/Circ1260.pdf p.87 for default parameterisation

## vs2dh.ConfigureInitial

*Configure initial conditions*

### Description

Configure initial conditions

### Usage

```
vs2dh.ConfigureInitial(flow = vs2dh.ConfigureInitialFlow(),
  temp = vs2dh.ConfigureInitialTemp())
```

### Arguments

| | |
|---|---|
| flow | initial flow conditions (i.e. pressure head or saturation distribution for model domains, as retrieved by vs2dh.ConfigureInitialFlow() |
| temp | initial temperature distribution as retrieved by vs2dh.ConfigureInitialTemp() |

### Value

Initial model conditions

## vs2dh.ConfigureInitialFlow

*Configure initial head/soil moisture distribution*

### Description

Configure initial head/soil moisture distribution

### Usage

```
vs2dh.ConfigureInitialFlow(values = NA, dwtx = 3.7291667,
  hmin = -3.98, asPressureHeads = TRUE)
```

### Arguments

| | |
|---|---|
| values | If NA values are supplied (default): initial conditions are defined in terms of pressure head, and an equilibrium profile is specified above a free-water surface at a depth of DWTX until a pressure head of HMIN is reached. All pressure heads above this are set to HMIN. If one value is supplied: all initial conditions in terms of pressure head (if asPressureHead=TRUE) or moisture(if asPressure-Head=FALSE) are set equal to value of "hvalues". |

| | |
|---|---|
| dwtx | Depth to free-water surface above which an equilibrium profile is computed, L. All All pressure heads above DWTX this are set to HMIN. (default: 3.7291667) |
| hmin | Minimum pressure head to limit height of equilibrium profile, L. Must be negative. (default: -3.98) matrix (with columns equal to NXR and rows equals to NLY) needs to be supplied |
| asPressureHeads | If TRUE, initial flow conditions are pressure heads, else: moisture content (default: TRUE) |

## Value

Initial flow conditions

---

`vs2dh.ConfigureInitialTemp` *Configure initial temperature distribution:*

---

## Description

Configure initial temperature distribution:

## Usage

```
vs2dh.ConfigureInitialTemp(values = 10)
```

## Arguments

| | |
|---|---|
| values | either constant (for constant initial temperature for whole model domain, or matrix with number of rows (equals to nxr) and columns (equals to nly), with user defined initial temperature distribution (default: 10 degree C, constant temperature) |

## Value

Initial temperature distribution parameterisation

## vs2dh.ConfigureObsPoints

*Configure Observation points*

### Description

Configure Observation points

### Usage

```
vs2dh.ConfigureObsPoints(obs_n = NULL, obs_j = NULL,
  outputEachTimeStep = TRUE)
```

### Arguments

| | |
|---|---|
| obs_n | grid column "n" (if NULL no observation points will be written to file obsPoints.out) |
| obs_j | grid row "j" (if NULL no observation points will be written to file obsPoints.out) |
| outputEachTimeStep | If FALSE output to obsPoints.out is desired only at selected output times rather than at each time step (default: TRUE) |

### Value

Observation point parameterisation

## vs2dh.ConfigureRechargePeriod

*Configure recharge period*

### Description

Configure recharge period

### Usage

```
vs2dh.ConfigureRechargePeriod(tper = 0.1,
  options = vs2dh.ConfigureRechargePeriodOptions(),
  solver = vs2dh.ConfigureRechargePeriodSolver(),
  boundary = vs2dh.ConfigureBoundaryCondition())
```

### Arguments

| | |
|---|---|
| tper | Length of this recharge period, T. (default: 0.1) |
| options | as retrieved by vs2dh.ConfigureRechargePeriodOptions() |
| solver | as retrieved by vs2dh.ConfigureRechargePeriodSolver() |
| boundary | as retrieved by vs2dh.ConfigureBoundaryCondition() |

### Value

Recharge period

---

## vs2dh.ConfigureRechargePeriodOptions

*Configure recharge period options*

---

### Description

Configure recharge period options

### Usage

```
vs2dh.ConfigureRechargePeriodOptions(prnt = FALSE, rbcit = FALSE,
  retsim = FALSE, seepage = vs2dh.ConfigureSeepage())
```

### Arguments

| | |
|---|---|
| prnt | If TRUE, = T if heads, temperature, moisture contents, and/or saturations are to be printed to file "vs2dt.out" after each time step; If FALSE they are to be written to file "vs2dt.out" only at observation times and ends of recharge periods. (default: FALSE) |
| rbcit | NOT IMPLEMENTED YET!!!!!!! If TRUE evaporation is to be simulated for this recharge period;if FALSE no evaporation is simulated for this recharge period (default: FALSE) |
| retsim | NOT IMPLEMENTED YET!!!!!!! If TRUE evapotanspiration (plant-root extraction) is to be simulated for this recharge period; if FALSE no evaporation is simulated for this recharge period (default: FALSE). |
| seepage | If TRUE are to be simulated for this recharge period; seepage face not simulated for this recharge period. (default: vs2dh.ConfigureSeepage()) |

### Value

Recharge period

## vs2dh.ConfigureRechargePeriods

*Configure recharge periods*

### Description

Configure recharge periods

### Usage

```
vs2dh.ConfigureRechargePeriods(periods = list(vs2dh.ConfigureRechargePeriod(),
 vs2dh.ConfigureRechargePeriod(tper = 10)))
```

### Arguments

| | |
|---|---|
| periods | list with one or multiple recharge periods with structure as retrieved by vs2dh.ConfigureRechargePeriod() |

### Value

List with parameterisation of recharge periods

## vs2dh.ConfigureRechargePeriodSolver

*Configure recharge period solver*

### Description

Configure recharge period solver

### Usage

```
vs2dh.ConfigureRechargePeriodSolver(delt = 1e-04, tmlt = 1.2,
  dltmx = 1, dltmin = 1e-04, tred = 0.1, dsmax = 10, sterr = 0,
  pond = 0)
```

### Arguments

| | |
|---|---|
| delt | Length of initial time step for this period, T. (default: 1.0E-4) |
| tmlt | Multiplier for time step length. (default: 1.2) |
| dltmx | Maximum allowed length of time step, T. (default: 1) |
| dltmin | Minimum allowed length of time step, T. (default: 1.0E-4) |
| tred | Factor by which time-step length is reduced if convergence is not obtained in IT-MAX iterations. Values usually should be in the range 0.1 to 0.5. If no reduction of time-step length is desired, input a value of 0.0. (default: 0.1) |

| | |
|---|---|
| dsmax | Maximum allowed change in head per time step for this period, L. (default: 10) |
| sterr | Steady-state head criterion; when the maximum change in head between successive time steps is less than STERR, the program assumes that steady state has been reached for this period and advances to next recharge period, L. (default: 0) |
| pond | Maximum allowed height of ponded water for constant flux nodes. See Lappala and other (1987) for detailed discussion of POND, L. (default: 0) |

## Value

Recharge period times

---

`vs2dh.ConfigureSeepage` *Configure seepage*

---

## Description

Configure seepage

## Usage

```
vs2dh.ConfigureSeepage(seepFaces = list(vs2dh.ConfigureSeepageFace(),
  vs2dh.ConfigureSeepageFace(j = 30:20, n = 45)))
```

## Arguments

| | |
|---|---|
| seepFaces | list of seepage faces as retrieved by vs2dh.ConfigureSeepageFace() |

## Value

Seepage config

---

`vs2dh.ConfigureSeepageFace` *Configure seepage face*

---

## Description

Configure seepage face

## Usage

```
vs2dh.ConfigureSeepageFace(j = c(30:20), n = 2, jlast = 0)
```

### Arguments

| | |
|---|---|
| j | Row and column of each cell on possible seepage face, in order from the lowest to the highest elevation; JJ pairs of values are required. |
| n | Column of each cell on possible seepage face, in order from the lowest to the highest elevation; JJ pairs of values are required. |
| jlast | Number of the node which initially represents the highest node of the seep; value can range from 0 (bottom of the face) up to JJ (top of the face). (default: 0) |

### Value

Seepage face

---

## vs2dh.ConfigureSoil *Configure soil parameters*

---

### Description

Configure soil parameters

### Usage

```
vs2dh.ConfigureSoil(hk = vs2dh.ConfigureGenuchten(),
  ht = vs2dh.ConfigureTrans())
```

### Arguments

| | |
|---|---|
| hk | flow properties `vs2dh.ConfigureGenuchten()` |
| ht | transport properties `vs2dh.ConfigureTrans()` |

### Value

(Genuchten) flow & transport parameters of soil

---

## vs2dh.ConfigureSoilGrid *Configure soil grid*

---

### Description

Configure soil grid

### Usage

```
vs2dh.ConfigureSoilGrid(ntex = 2, grid = vs2dh.ConfigureBasicGrid(),
  irow = 0)
```

### Arguments

| | |
|---|---|
| ntex | number of different textual classes (default: 2) |
| grid | grid matrix as retrieved by vs2dh.ConfigureBasicGrid() |
| irow | Textural classes are read for each row (irow=0). This option is preferable if many |

### Value

Grid with soil indexes

---

## vs2dh.ConfigureSoils *Configure Soils*

---

### Description

Configure Soils

### Usage

```
vs2dh.ConfigureSoils(props = list(vs2dh.ConfigureSoil(),
  vs2dh.ConfigureSoil(hk = vs2dh.ConfigureGenuchten(ratioKzKh = 100))),
  grid = vs2dh.ConfigureSoilGrid(ntex = length(props)), wus = 0.5)
```

### Arguments

| | |
|---|---|
| props | Soil (flow & transport) properties as retrieved by vs2dh.ConfigureSoil() ordered by itex number, i.e. first list element = itex1, second = itex2 |
| grid | grid matrix as retrieved by vs2dh.ConfigureSoilGrid |
| wus | Weighting option for intercell relative hydraulic conductivity: 1 (for full upstream weighting), 0.5 (for arithmetic mean) and 0.0 (for geometric mean), (default: 0.5) |

### Value

Configuration of soils for flow and energy transport (only hydraulic function type 1 vanGenuchten model is implemented!)

## vs2dh.ConfigureTrans *Configure soil transport parameters*

### Description

Configure soil transport parameters

### Usage

```
vs2dh.ConfigureTrans(alphaL = 1, alphaT = 0.1, cs = 2180000,
  ktRmc = 129600, ktSat = 155520, cw = 4180000)
```

### Arguments

| | |
|---|---|
| alphaL | Longitudinal dispersivity, L. (Default: 1 m) |
| alphaT | Transverse dispersivity, L. (Default: 0.1 m) |
| cs | Heat capacity of dry solids (Cs), Q/L3 C. (Default: 2180000.0 J/m3C) |
| ktRmc | Thermal conductivity of water sediment at residual moisture content, Q/LTC. (default: 129600.0) |
| ktSat | Thermal conductivity of water sediments at full saturation, Q/LC. (Default: 155520.0) |
| cw | Heat capacity of water (Cw), which is the product of density times specific heat of water, Q/L3 C. (default: 4180000.0) |

### Value

Soil transport parameters

### See Also

http://pubs.usgs.gov/circ/2003/circ1260/pdf/Circ1260.pdf p.87 for default parameterisation

## vs2dh.plotMassBalance *Plotting of mass balance time series*

### Description

Plotting of mass balance time series

### Usage

```
vs2dh.plotMassBalance(paras = c("TOTAL__FLOW_IN", "TOTAL__FLOW_OUT",
  "FLUID__STORAGE"), paraUnit = "TIMESTEP", data,
  mainLabel = "Flow mass balance", ...)
```

## Arguments

| | |
|---|---|
| paras | vector with at least one or multiple parameters in model results, for checking available paras run (see example: validParas). The parameter "TIME" is not allowed! |
| paraUnit | TIMESTEP RATE or |
| data | as returned by vs2di.run()$balance |
| mainLabel | a text to be written above the plot |
| ... | further parameters passed to xyplot() |

## Value

Temporal time series plot of mass balance variables

## Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
res <- vs2di.run(model.path = model.path)
#### Checking available parameter names for "paras":
validParas <- gsub(pattern="__TOTAL|__RATE|__TIMESTEP|TIME",
                   replacement = "",
                   colnames(res$balance))
validParas
### Flow mass balance components (inflow, outflow, storage):
vs2dh.plotMassBalance(data=res$balance)
### Only resulting flow mass balance error (per timestep):
vs2dh.plotMassBalance(paras="FLUID__VOL_BAL",
                      data = res$balance,
                     mainLabel = "Flow mass balance error")
### Energy mass balance components (inflow, outflow, storage):
vs2dh.plotMassBalance(paras=c("TOTAL__ENERGY_IN",
                              "TOTAL__ENERGY_OUT",
                              "ENERGY__STORAGE"),
                              data=res$balance,
                              mainLabel="Energy mass balance")
### Only resulting energy mass balance error (per timestep):
vs2dh.plotMassBalance(paras="ENERGY__BALANCE",
                      data = res$balance,
                    mainLabel = "Energy mass balance error")
```

---

# vs2dh.plotMatrix *Plot vs2dh.plotMatrix results*

---

## Description

Plot vs2dh.plotMatrix results

### Usage

```
vs2dh.plotMatrix(data, nodes = TRUE, maxCol = 100,
  ignoreZeroValues = FALSE, ...)
```

### Arguments

| | |
|---|---|
| data | matrix as returned by vs2di.run(model.path, returnOutput=TRUE) |
| nodes | if TRUE node numbers are used as x-/y-axis; if false spatial extent |
| maxCol | maximum number of colors to be used for colorramp |
| ignoreZeroValues | if TRUE 0 values will be set to NA (Default: FALSE) |
| ... | further parameters passed to `levelplot()` is derived returned in model units (currently not implemented!) |

### Value

Plot matrix values

### Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
res <- vs2di.run(model.path = model.path)
vs2dh.plotMatrix(data=res$variables$PressureHead[[1]])
```

---

## vs2dh.plotObservationPoints

*Spatial plotting of observation point time series*

---

### Description

Spatial plotting of observation point time series

### Usage

```
vs2dh.plotObservationPoints(paras = "Temp", data, paraLabel = NULL,
  maxCol = 100, ...)
```

### Arguments

| | |
|---|---|
| paras | vector with at least one of the following parameters: H_m (total head), P_m (pressure head), THETA (moisture content), SAT (saturation), TEMP (for temperature), VX (velocity in x-direction), VZ (velocity in z-direction), ET (evapotransporation) |
| data | as returned by vs2di.run() in sublist "obsPoint" |

| | |
|---|---|
| paraLabel | optional parameter for changing the label text of the parameters defined in paras (if not used the value in "paras" is used for labelling) |
| maxCol | maximum number of colors to be used for colorramp |
| ... | further parameters passed to levelplot() |

## Value

Spatial plots of observation point time series

## Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
res <- vs2di.run(model.path = model.path)
vs2dh.plotObservationPoints (paras="TEMP",
                             data=res$obsPoints,
                             paraLabel="Temperature (\u00B0\ C)")
```

---

# vs2dh.plotVariables *Plot vs2dh.plotVariables matrix time series*

---

## Description

Plot vs2dh.plotVariables matrix time series

## Usage

```
vs2dh.plotVariables(para = "Temp", data, ignoreBoundary = TRUE,
  ignoreZeroValues = TRUE, paraLabel = NULL, maxCol = 100,
  main = "", nodes = TRUE, ...)
```

## Arguments

| | |
|---|---|
| para | either "Temp" (for temperature) or "PressureHead" |
| data | as returned by vs2di.run() in sublist "variables" |
| ignoreBoundary | should boundary nodes (first/last column/row) be ignored? Default: TRUE |
| ignoreZeroValues | if TRUE 0 values will be set to NA (Default: TRUE) |
| paraLabel | optional parameter for changing the label text of the parameter (if not used the value in "para" is used for labelling) |
| maxCol | maximum number of colors to be used for colorramp |
| main | Additional text for main label (e.g. model name) |
| nodes | if TRUE node numbers are used as x-/y-axis; if false spatial extent is derived returned in model units (currently not implemented!) |
| ... | further parameters passed to levelplot() |

## Value

Time series matrix plot

## Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
modelName <- "tutorial2"
model.path <- system.file("extdata", "vs2dh_example", modelName, package = "kwb.vs2dh")
res <- vs2di.run(model.path = model.path)
vs2dh.plotVariables ( para = "Temp",
                      data = res$variables,
                      paraLabel = "Temperature (\u00B0\ C)",
                      main = modelName)
vs2dh.plotVariables (para = "PressureHead",
                      data = res$variables,
                      paraLabel = "Pressure head (m)",
                      main = modelName
                     )
```

---

# vs2dh.readBoundaryFluxes

*Read VS2dh model output file with fluxes at boundary faces*

---

## Description

Read VS2dh model output file with fluxes at boundary faces

## Usage

```
vs2dh.readBoundaryFluxes(model.path, fileName = "boundaryFluxes.out",
  dbg = TRUE)
```

## Arguments

| | |
|---|---|
| model.path | full path to folder containing model output files |
| fileName | name of file containing fluxes at boundary faces (default: "boundaryFluxes.out") |
| dbg | if true text output |

## Value

read model results of boundary faces are imported in a R object

## Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
vs2dh.readBoundaryFluxes(model.path)
```

## vs2dh.ReadConfig *vs2dh.ReadConfig()*

### Description

Imports the configuration from an existing vs2dh.dat file and stores it in an R list data structure as retrieved by vs2dh.Configure()

### Usage

```
vs2dh.ReadConfig(model.path, dbg = TRUE)
```

### Arguments

| | |
|---|---|
| model.path | Full path to the folder containing the vs2dh.dat |
| dbg | prints debug information on the screen |

### Value

Imported vs2dh configuration model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh") vs2dh.ReadConfig(model.path)

## vs2dh.readObsPoints *Read VS2dh model output file with observation point time series*

### Description

Read VS2dh model output file with observation point time series

### Usage

```
vs2dh.readObsPoints(model.path, fileName = "obsPoints.out", dbg = TRUE)
```

### Arguments

| | |
|---|---|
| model.path | full path to folder containing model output files |
| fileName | name of file containing observation point results (default: "obsPoints.out") |
| dbg | if true text output |

### Value

read model results at observation points aare imported in a R object

### Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
vs2dh.readObsPoints(model.path)
```

## vs2dh.readVariables *Read VS2dh model output file with time series of variables pressure head and temperature*

### Description

Read VS2dh model output file with time series of variables pressure head and temperature

### Usage

```
vs2dh.readVariables(model.path, fileName = "variables.out", dbg = TRUE)
```

### Arguments

| | |
|---|---|
| model.path | full path to folder containing model output files |
| fileName | name of file containing variable results (default: "variables.out") |
| dbg | if true text output |

### Value

variable model results are imported in a R object

### Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
res <- vs2di.run(model.path = model.path)
vs2dh.readVariables(model.path)
```

## vs2dh.writeConfig *Write configuration*

### Description

Write configuration

### Usage

```
vs2dh.writeConfig(conf)
```

### Arguments

| | |
|---|---|
| conf | model config as retrieved by vs2dhConfigure() |

### Value

Write vs2dh.dat in folder model.path

## Examples

```
conf <- vs2dh.Configure()
inpDat <- vs2dh.writeConfig(conf)
write(inpDat, file.path(getwd(), "inp.dat"))
```

---

`vs2di.createFileDat` *Create vs2dh.fil or vs2dt.fil file*

---

## Description

Create vs2dh.fil or vs2dt.fil file

## Usage

```
vs2di.createFileDat(engine, model.path, dbg = TRUE)
```

## Arguments

| | |
|---|---|
| `engine` | model engine either 'vs2dh' or 'vs2dt' |
| `model.path` | full path to folder containing vs2dh.dat |
| `dbg` | if true text output |

## Value

Saves/Overwrites vs2dh.fil in folder `model.path`

## Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
vs2di.createFileDat(engine = "vs2dh",
                    model.path = model.path)
```

---

`vs2di.readBalance` *Read VS2dh model output file with (energy, fluid) mass balance time series*

---

## Description

Read VS2dh model output file with (energy, fluid) mass balance time series

## Usage

```
vs2di.readBalance(model.path, engine, fileName = "balance.out",
  dbg = TRUE)
```

### Arguments

| | |
|---|---|
| model.path | full path to folder containing model output files |
| engine | model engine 'vs2dh' (for flow & heat modelling) or 'vs2dt' (for flow & solute transport) (Default: no default) |
| fileName | name of file containing (energy and fluid) balance results (default: "balance.out") |
| dbg | if true text output |

### Value

balance model results are imported in a R object

### Examples

```
### Location of kwb.vs2dh package on your computer
wDir <- system.file(package = "kwb.vs2dh")
model.path <- file.path(wDir, "extdata/vs2dh_example/tutorial2")
res <- vs2di.run(model.path = model.path)
vs2di.readBalance(model.path = model.path, engine = "vs2dh")
```

---

## vs2di.readMain *Read main VS2dh model output file "vs2dh.out"*

---

### Description

Read main VS2dh model output file "vs2dh.out"

### Usage

```
vs2di.readMain(model.path, engine = "vs2dh")
```

### Arguments

| | |
|---|---|
| model.path | full path to folder containing model output files |
| engine | model engine 'vs2dh' (for flow & heat modelling) or 'vs2dt' (for flow & solute transport) (Default: "vs2dh") |

### Value

main model results are imported in a R object

### Examples

```
### Location of example vs2dh model contained in "kwb.vs2dh package"
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
res <- vs2di.run(model.path = model.path)
vs2di.readMain(model.path = model.path, engine = "vs2dh")
```

vs2di.run *Run VS2dh model*

## Description

Run VS2dh model

## Usage

```
vs2di.run(engine = "vs2dh",
  engineDirectoryWin = this_extdata_file("engine/win"),
  engineDirectoryLinux = this_extdata_file("engine/linux"),
  model.path = this_extdata_file("vs2dh_example/tutorial2"),
  returnOutput = TRUE, showWarnings = TRUE, openTargetDir = FALSE,
  dbg = TRUE)
```

## Arguments

| | |
|---|---|
| engine | model engine 'vs2dh' (for flow & heat modelling) or 'vs2dt' (for flow & solute transport) (Default: "vs2dh") |
| engineDirectoryWin | default directory on Windows OS containing vs2dh3_3.exe and vs2dt3_3.exe (Default: system.file("extdata/engine/win", package = "kwb.vs2dh")) |
| engineDirectoryLinux | default directory on Linux OS containing vs2dh3_3.exe and vs2dt3_3.exe (Default: system.file("extdata/engine/linux", package = "kwb.vs2dh")) |
| model.path | full path to folder containing vs2dh.dat (default: system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")) |
| returnOutput | if TRUE model output files variables.out, balance.out and obsPoints.out are imported into R completely; vs2dh.out (only warnings imported) |
| showWarnings | if TRUE print warning messages during simulation on screen (default: TRUE) |
| openTargetDir | If TRUE path containing model files will be opened in explorer (Default: FALSE) |
| dbg | if true text output on screen on model run progress |

## Value

Run VS2dh model and saves model output files in folder `model.path`. If `returnOutput` is TRUE the output is imported in R object

## Examples

```
### Running model with default model.path and engine.path:
model.path <- system.file("extdata", "vs2dh_example/tutorial2", package = "kwb.vs2dh")
res <- vs2di.run(model.path = model.path)
```

vs2di.runConfig *Run configuration with VS2dh model*

## Description

Run configuration with VS2dh model

## Usage

```
vs2di.runConfig(conf, engine = "vs2dh", tDir = tempdir(),
  returnOutput = TRUE, openTargetDir = TRUE, showWarnings = TRUE,
  dbgRun = TRUE, dbg = TRUE)
```

## Arguments

| | |
|---|---|
| conf | as retrieved by vs2dh.ReadConfig (i.e. importing a working VS2DH configuration from an vs2h.dat file) or completley in R using function vs2dh.Configure() (ATTENTION: DEFAULT PARAMETERISATION FOR vs2dh.Configure() CURRENTLY YIELDS NO RUNNING VS2DH MODEL CONFIGURATION) |
| engine | model engine 'vs2dh' (for flow & heat modelling) or 'vs2dt' (for flow & solute transport) (Default: "vs2dh") |
| tDir | target directory where vs2dh model input/output files should be stored. (Default: tempdir()) |
| returnOutput | if TRUE model output files variables.out, balance.out and obsPoints.out are imported into R completely; vs2dh.out (only warnings imported) (Default: TRUE) |
| openTargetDir | If TRUE path containing model files will be opened in explorer (Default: FALSE) |
| showWarnings | if TRUE print warning messages during simulation on screen (default: TRUE) |
| dbgRun | if true text output on screen on model run progress |
| dbg | if true text output on screen on additional model run progress |

## Value

Import & write VS2dh model results in R object

## Examples

```
## Not run:
### Testing import and writing functions with
### Folder which contains the subfolders with the different models to test:
model.main.path <- system.file("extdata", "vs2dh_example", package = "kwb.vs2dh")
#### Using vs2dh.dat file contained in subfolders example1, example2, tutorial1
#### for testing (i.e. ignore example3 due to long run time, but this also works!)
exampleModels <- c(paste0("example", 1:2), "tutorial1")
### Create result list in R for storing model outputs for each model in a separate
### sublist
res <- list()
```

```
#### Loop through all example model input files & plot temperature distribution
for (testModel in exampleModels)
{
 model.path <- file.path(model.main.path, testModel)
 cat(sprintf("Testing model in %s\n", model.path ))
 cat("1.Step: importing configuration to R.....")
 conf <- vs2dh.ReadConfig(model.path = model.path)
 cat("Done!")
 cat("2.Step: Run vs2di.runConfig():")
 tDir <- file.path(tempdir(), testModel)
 res[[testModel]] <- vs2di.runConfig(conf = conf, tDir = tDir)
 cat("3.Step: Plot temperature distribution...")
 vs2dh.plotVariables(para = "Temp",
                     data = res[[testModel]]$variables,
                     main = testModel  ### nice label by using model folder name
                    )
  cat("Done!")
}

## End(Not run)
```

# Index