Package 'hydraulics'

March 6, 2024

Type Package

Title Basic Pipe and Open Channel Hydraulics

Version 0.7.0

Author Ed Maurer [aut, cre], Irucka Embry [aut, ctb] (iemisc code)

Maintainer Ed Maurer <emaurer@scu.edu>

Description Functions for basic hydraulic calculations related to water flow in circular pipes both flowing full (under pressure), and partially full (gravity flow), and trapezoidal open channels. For pressure flow this includes friction loss calculations by solving the Darcy-Weisbach equation for head loss, flow or diameter, plotting a Moody diagram, matching a pump characteristic curve to a system curve, and solving for flows in a pipe network using the Hardy-Cross method. The Darcy-Weisbach friction factor is calculated using the Colebrook (or Colebrook-White equation), the basis of the Moody diagram, the original citation being Colebrook (1939) <doi:10.1680/ijoti.1939.13150>. For gravity flow, the Manning equation is used, again solving for missing parameters. The derivation of and solutions using the Darcy-Weisbach equation and the Manning equation are outlined in many fluid mechanics texts such as Finnemore and Maurer (2024, ISBN:978-1-264-78729-6). Some gradually- and rapidly-varied flow functions are included. For the Manning equation solutions, this package uses modifications of original code from the 'iemisc' package by Irucka Embry.

URL https://github.com/EdM44/hydraulics, https://edm44.github.io/hydraulics/

License GPL (>= 3)

Depends R (>= 3.6.0)

Encoding UTF-8

Imports ggplot2, grid, gtools, pracma, purrr, reshape2, stats, tibble, units

Suggests formatdown, kableExtra, knitr, rmarkdown

RoxygenNote 7.3.1

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Repository CRAN

Date/Publication 2024-03-06 13:10:08 UTC

R topics documented:

`atmosprops` *Functions to calculate ICAO standard atmospheric properties: temperature, density, and pressure.*

Description

Functions to calculate ICAO standard atmospheric properties: temperature, density, and pressure.

Usage

```
atmtemp(alt = NULL, units = NULL, ret_units = FALSE)

atmpres(alt = NULL, units = NULL, ret_units = FALSE)

atmdens(alt = NULL, units = NULL, ret_units = FALSE)
```

Arguments

alt	the altitude (above mean sea level). If excluded, sea level is assumed [m or ft]
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units.
ret_units	If set to TRUE the value(s) returned are of class units with units attached to the value. [Default is FALSE]

Value

the temperature of air for the standard atmosphere for the atmtemp function [$^\circ C$ or $^\circ F$]

the absolute pressure of air for the standard atmosphere for the atmpres function [Nm^{-2} or $lbf\,ft^{-2}$]

the density of air for the standard atmosphere for the atmdens function [$kg\,m^{-3}$ or $slug\,ft^{-3}$]

Author(s)

Ed Maurer

Examples

```
#Find standard atmospheric temperature at altitude 8000 m
atmtemp(alt = 8000, units = 'SI')

#Find standard atmospheric pressure assuming default altitude of zero (sea-level)
atmpres(units = 'Eng', ret_units = TRUE)

#Find standard atmospheric density at altitude 15000 ft
atmdens(alt = 15000, units = 'Eng')
```

---

atmos_table *Tabulates into a tibble some properties of the standard atmosphere: temperature, density, and pressure.*

Description

Tabulates into a tibble some properties of the standard atmosphere: temperature, density, and pressure.

Usage

```
atmos_table(units = c("SI", "Eng"), ret_units = TRUE)
```

Arguments

units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units.
ret_units	If set to TRUE the value(s) returned are of class units with units attached to the value. [Default is TRUE]

Author(s)

Ed Maurer

Examples

```
atmos_table(units = 'SI')
```

---

colebrook *Calculates the Darcy-Weisbach Friction Factor f*

Description

This function calculates the Darcy-Weisbach friction factor and is only provided in this package for use with water in circular pipes while the equation is technically valid for any liquid or channel. As with many parts of this package, techniques and formatting were drawn from Irucka Embry's iemisc package, which includes some methods with similar functionality. Two utility functions are included for velocity and Reynolds Number.

Usage

```
velocity(D = NULL, Q = NULL)

reynolds_number(V = NULL, D = NULL, nu = NULL)

colebrook(ks, V, D, nu)
```

Arguments

D	numeric vector that contains the pipe diameter [m or ft] which should be D >=0.0025 m (0.0082 ft).
Q	(for velocity function only) numeric vector that contains the flow rate [$m^3 s^{-1}$ or $ft^3 s^{-1}$]
V	numeric vector that contains the average Velocity of flow in the pipe, equal to flow divided by area, $\frac{Q}{A}$ [$m\,s^{-1}$ or $ft\,s^{-1}$]
nu	numeric vector that contains the kinematic viscosity of water, [$m^2 s^{-1}$ or $ft^2 s^{-1}$]. Computed with a utility function in water_properties.R: kvisc(T=T, units=['SI' or 'Eng'])
ks	numeric vector that contains the 'equivalent sand roughness height sand roughness height. Units should be consistent with other input [m or ft]

Details

The Colebrook-White equation was developed to estimate the Darcy-Weisbach friction factor for commercial pipes under turbulent flow conditions. It is recommended for pipe diameters greater than 2.5 mm (0.1 inch). The equation is:

$$\frac{1}{\sqrt{f}} = -2\log\left(\frac{\frac{ks}{D}}{3.7} + \frac{2.51}{Re\sqrt{f}}\right)$$

where $Re = \frac{VD}{nu}$ is the unitless Reynolds Number.

Value

f Returns a numeric vector containing the Darcy-Weisbach friction factor

Author(s)

Ed Maurer

See Also

`kvisc` for kinematic viscosity, `velocity` for calculating $V = \frac{Q}{A}$, `reynolds_number` for Reynolds number

Examples

```
# A Type 1 problem (solve for hf): US units
D <- 20/12   #diameter of 20 inches
Q <- 4       #flow in ft^3/s
T <- 60      #water temperature in F
ks <- 0.0005 #pipe roughness in ft

f <- colebrook(ks=ks,V=velocity(D,Q), D=D, nu=kvisc(T=T, units="Eng"))
```

`darcyweisbach` *Solves the Darcy-Weisbach Equation for the either head loss (hf), flow rate (Q), diameter (D), or roughness height (ks).*

Description

This function solves the Darcy-Weisbach friction loss equation for with water in circular pipes. the function solves for either head loss (hf), flow rate (Q), diameter (D),or roughness height, (ks) whichever is missing (not included as an argument).

Usage

```
darcyweisbach(
  Q = NULL,
  D = NULL,
  hf = NULL,
  L = NULL,
  ks = NULL,
  nu = NULL,
  units = c("SI", "Eng"),
  ret_units = FALSE
)
```

Arguments

Q	numeric vector that contains the flow rate [m^3s^{-1} or ft^3s^{-1}]
D	numeric vector that contains the pipe diameter [m or ft]
hf	numeric vector that contains the head loss through the pipe section [m or ft]
L	numeric vector that contains the pipe length [m or ft],
ks	numeric vector that contains the equivalent sand roughness height. Units should be consistent with other input [m or ft]
nu	numeric vector that contains the kinematic viscosity of water, [m^2s^{-1} or ft^2s^{-1}].
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units. This is used for compatibility with iemisc package
ret_units	If set to TRUE the value(s) returned are of class units with units attached to the value. [Default is FALSE]

Details

The Darcy-Weisbach equation was developed to estimate the head loss, h_f, due to friction over a length of pipe. For circular pipes it is expressed as:

$$h_f = \frac{fL}{D}\frac{V^2}{2g} = \frac{8fL}{\pi^2 g D^5}Q^2$$

where f is the friction factor (calculated with the colebrook function and g is the gravitational acceleration ($9.81\frac{m}{s^2}$ or $32.2\frac{ft}{s^2}$).

Value

Returns a list including the missing parameter (hf, Q, D, or ks):

- Q - flow rate.
- V - flow velocity.
- L - pipe length.
- hf - head loss due to friction

- f - Darcy-Weisbach friction factor
- ks - roughness height
- Re - Reynolds number

See Also

colebrook for friction factor calculation

Examples

```
#Type 2 (solving for flow rate, Q): SI Units
D <- .5
L <- 10
hf <- 0.006*L
T <- 20
ks <- 0.000046
darcyweisbach(D = D, hf = hf, L = L, ks = ks, nu = kvisc(T=T, units='SI'), units = c('SI'))

#Type 3 (solving for diameter, D): Eng (US) units
Q <- 37.5     #flow in ft^3/s
L <- 8000     #pipe length in ft
hf <- 215     #head loss due to friction, in ft
T <- 68       #water temperature, F
ks <- 0.0008  #pipe roughness, ft
darcyweisbach(Q = Q, hf = hf, L = L, ks = ks, nu = kvisc(T=T, units='Eng'), units = c('Eng'))
```

---

direct_step *Uses the direct step method to find the distance between two known depths in a trapezoidal channel*

Description

This function applies the direct step method for a gradually-varying water surface profile for flow in an open channel with a trapezoidal shape.

Usage

```
direct_step(
  So = NULL,
  n = NULL,
  Q = NULL,
  y1 = NULL,
  y2 = NULL,
  b = NULL,
  m = NULL,
  nsteps = 1,
```

```
  units = c("SI", "Eng"),
  ret_units = FALSE
)
```

Arguments

`So`	numeric channel bed slope [unitless]
`n`	numeric vector that contains the Manning roughness coefficient
`Q`	numeric vector that contains the flow rate [m^3s^{-1} or ft^3s^{-1}]
`y1`	numeric vector that contains the initial water depth [m or ft]
`y2`	numeric vector that contains the final water depth [m or ft]
`b`	numeric vector that contains the channel bottom width [m or ft]
`m`	numeric vector that contains the side slope of the channel (m:1 H:V) [unitless]
`nsteps`	integer of the number of calculation steps between y1 and y2 [unitless]
`units`	character vector that contains the system of units [options are `SI` for International System of Units and `Eng` for English (US customary) units. This is used for compatibility with iemisc package.
`ret_units`	If set to TRUE the value(s) returned are of class `units` with units attached to the value. [Default is FALSE]

Details

The direct step method applies the energy equation to gradually-varied open channel flow conditions, assuming each increment is approximately uniform. This function works with a trapezoidal channel shape. The water depths at two locations are input with channel geometry and flow rate, and the distance between the two locations, ΔX, is calculated:

$$\Delta X = \frac{E_1 - E_2}{S_f - S_o}$$

where E_1 and E_2 are the specific energy values at the locations of y_1 and y_2, S_f is the slope of the energy grade line, and S_o is the slope of the channel bed.

Value

Returns a data frame (tibble) with the columns:

- x - cumulative distance from position of y1
- z - elevation of the channel bed at location x
- y - depth of the water at location x
- A - cross-sectional area at location x
- Sf - slope of the energy grade line at location x
- E - specific energy at location x
- Fr - Froude number at location x

Author(s)

Ed Maurer

Examples

```
#Solving for profile between depths 3.1 ft and 3.4 ft in a rectangular channel
#Flow of 140 ft^3/s, bottom width = 6 ft:
direct_step(So=0.0015, n=0.013, Q=140, y1=3.1, y2=3.4, b=6, m=0, nsteps=2, units="Eng")
```

---

hardycross *Applies the Hardy-Cross method to solve for pipe flows in a network.*

Description

This function uses the Hardy-Cross method to iteratively solve the equations for conservation of mass and energy in a water pipe network. The input consists of a data frame with the pipe characteristics and lists of the pipes in each loop (listed in a clockwise direction) and the initial guesses of flows in each pipe (positive flows are in a clockwise direction).

Usage

```
hardycross(
  dfpipes = dfpipes,
  loops = loops,
  Qs = Qs,
  n_iter = 1,
  units = c("SI", "Eng"),
  ret_units = FALSE
)
```

Arguments

`dfpipes`	data frame with the pipe data. Format is described above, but must contain a column named _ID_.
`loops`	integer list defining pipes in each loop of the network.
`Qs`	numeric list of initial flows for each pipe in each loop [m^3s^{-1} or ft^3s^{-1}]
`n_iter`	integer identifying the number of iterations to perform.
`units`	character vector that contains the system of units [options are `SI` for International System of Units and `Eng` for English (US customary) units.
`ret_units`	If set to TRUE the value(s) returned for pipe flows are of class `units` with units attached to the value. [Default is FALSE]

Details

The input data frame with the pipe data must contain a pipe ID column with the pipe numbers used in the loops input list. There are three options for input column of the pipe roughness data frame:

Column Name	Approach for Determining K
ks	f calculated using Colebrook equation, K using Darcy-Weisbach
f	f treated as fixed, K calculated using Darcy-Weisbach
K	K treated as fixed

In the case where absolute pipe roughness, ks (in m or ft), is input, the input pipe data frame must also include columns for the length, L and diameter, D, (both in m or ft) so K can be calculated. In this case, a new f and K are calculated at each iteration, the final values of which are included in the output. If input K or f columns are provided, values for ks are ignored. If an input K column is provided, ks and f are ignored. If the Colebrook equation is used to determine f, a water temperature of 20^oC or 68^oF is used.

The number of iterations to perform may be specified with the n_iter input value, but execution stops if the average flow adjustment becomes smaller than 1 percent of the average flow in all pipes.

The Darcy-Weisbach equation is used to estimate the head loss in each pipe segment, expressed in a condensed form as $h_f = KQ^2$ where:

$$K = \frac{8fL}{\pi^2 g D^5}$$

If needed, the friction factor f is calculated using the Colebrook equation. The flow adjustment in each loop is calculated at each iteration as:

$$\Delta Q_i = -\frac{\sum_{j=1}^{p_i} K_{ij} Q_j |Q_j|}{\sum_{j=1}^{p_i} 2K_{ij} Q_j^2}$$

where i is the loop number, j is the pipe number, p_i is the number of pipes in loop i and ΔQ_i is the flow adjustment to be applied to each pipe in loop i for the next iteration.

Value

Returns a list of two data frames:

- dfloops - the final flow magnitude and direction (clockwise positive) for each loop and pipe
- dfpipes - the input pipe data frame, with additional columns including final Q

See Also

colebrook, darcyweisbach

Examples

```
#             A----------B --> 0.5m^3/s
#             |\   (4)   |
#             | \        |
#             |  \       |
#             |   \(2)   |
#             |    \     |(5)
#             |(1)  \    |
#             |      \   |
```

```
#              |       \  |
#              |        \ |
#              |   (3)   \|
# 0.5m^3/s --> C----------D

#Input pipe characteristics data frame. With K given other columns not needed
dfpipes <- data.frame(
ID = c(1,2,3,4,5),                    #pipe ID
K = c(200,2500,500,800,300)           #resistance used in hf=KQ^2
)
loops <- list(c(1,2,3),c(2,4,5))
Qs <- list(c(0.3,0.1,-0.2),c(-0.1,0.2,-0.3))
hardycross(dfpipes = dfpipes, loops = loops, Qs = Qs, n_iter = 1, units = "SI")
```

`manningc` *Solves the Manning Equation for gravity flow in a circular pipe*

Description

This function solves the Manning equation for water flow in a circular pipe at less than full. Uniform flow conditions are assumed, so that the pipe slope is equal to the slope of the water surface and the energy grade line. This is a modification of the code prepared by Irucka Embry in his iemisc package. The iemisc::manningcirc function was adapted here for more limited cases commonly used in classroom exercises, additional checks were included to ensure the pipe is flowing less than full, and a cross-section figure is also available. The iemisc::manningcirc and iemisc::manningcircy functions were combined into a single function. Manning n supplied is assumed to be that for full pipe flow; an optional argument may be supplied to account for n varying with depth.

Usage

```
manningc(
  Q = NULL,
  n = NULL,
  Sf = NULL,
  y = NULL,
  d = NULL,
  y_d = NULL,
  n_var = FALSE,
  units = c("SI", "Eng"),
  ret_units = FALSE
)
```

Arguments

`Q`	numeric vector that contains the flow rate [$m^3 s^{-1}$ or $ft^3 s^{-1}$]
`n`	numeric vector that contains the Manning roughness coefficient (for full flow or fixed).

Sf	numeric vector that contains the slope of the pipe [unitless]
y	numeric vector that contains the water depth [m or ft]
d	numeric vector that contains the pipe diameter [m or ft]
y_d	numeric vector that contains the ratio of depth to diameter [unitless]
n_var	If set to TRUE the value of n for full flow is adjusted with depth. [Default is FALSE]
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units. This is used for compatibility with iemisc package]
ret_units	If set to TRUE the value(s) returned are of class units with units attached to the value. [Default is FALSE]

Details

The possible applications of this function for solving the Manning equation for circular pipes are:

Given	Solve for
y_d, Q, Sf, n	d
d, Sf, Q, n	y
y, d, Q, n	Sf
y, d, Sf, n	Q
d, Q, Sf, y	n

The Manning equation (also known as the Strickler equation) describes flow conditions in an open channel under uniform flow conditions. It is often expressed as:

$$Q = A\frac{C}{n}R^{\frac{2}{3}}{S_f}^{\frac{1}{2}}$$

where C is 1.0 for SI units and 1.49 for Eng (U.S. Customary) units. Critical depth is defined by the relation (at critical conditions):

$$\frac{Q^2 B}{g\,A^3} = 1$$

where B is the top width of the water surface. Since B equals zero for a full pipe, critical depth is set to the pipe diameter d if the flow Q exceeds a value that would produce a critical flow at $\frac{y}{d} = 0.99$.

Value

Returns a list including the missing parameter:

- Q - flow rate
- V - flow velocity
- A - cross-sectional area of flow
- P - wetted perimeter
- R - hydraulic radius (A/P)
- y - flow depth

- d - pipe diameter
- Sf - slope
- n - Manning's roughness (for full flow, or as adjusted if n_var is TRUE)
- yc - critical depth
- Fr - Froude number
- Re - Reynolds number
- Qf - Full pipe flow rate

Author(s)

Ed Maurer, Irucka Embry

See Also

xc_circle for a cross-section diagram of the circular channel

Examples

```
#Solving for flow rate, Q: SI Units
manningc(d = 0.6, n = 0.013, Sf = 1./400., y = 0.24, units = "SI")
#returns 0.1 m3/s

#Solving for Sf, if d=600 mm and pipe is to flow half full
manningc(d = 0.6, Q = 0.17, n = 0.013, y = 0.3, units = "SI")
#returns required slope of 0.003

#Solving for diameter, d when given y_d): Eng (US) units
manningc(Q = 83.5, n = 0.015, Sf = 0.0002, y_d = 0.9, units = "Eng")
#returns 7.0 ft required diameter

#Solving for depth, y when given Q: SI units
manningc(Q=0.01, n=0.013, Sf=0.001, d = 0.2, units="SI")
#returns depth  y = 0.158 m, critical depth, yc = 0.085 m

#Solving for depth, y when given Q: SI units, and n variable with depth
manningc(Q=0.01, n=0.013, Sf=0.001, d = 0.2, n_var = TRUE, units="SI")
#returns depth  y = 0.174 m, critical depth, yc = 0.085 m
```

`manningt` *Solves the Manning Equation for water flow in an open channel*

Description

This function solves the Manning equation for water flow in an open channel with a trapezoidal shape. Uniform flow conditions are assumed, so that the channel slope is equal to the slope of the water surface and the energy grade line. This is a modification of the code prepared by Irucka Embry in his iemisc package. Specifically the iemisc::manningtrap, iemisc::manningrect, and iemisc::manningtri were combined and adapted here for cases commonly used in classroom exercises. Some auxiliary variables in the iemisc code are not included here (shear stress, and specific energy), as these can be calculated separately. A cross-section figure is also available.

Usage

```
manningt(
  Q = NULL,
  n = NULL,
  m = NULL,
  Sf = NULL,
  y = NULL,
  b = NULL,
  units = c("SI", "Eng"),
  ret_units = FALSE
)
```

Arguments

`Q`	numeric vector that contains the flow rate [$m^3 s^{-1}$ or $ft^3 s^{-1}$]
`n`	numeric vector that contains the Manning roughness coefficient
`m`	numeric vector that contains the side slope of the channel (m:1 H:V) [unitless]
`Sf`	numeric vector that contains the slope of the channel [unitless]
`y`	numeric vector that contains the water depth [m or ft]
`b`	numeric vector that contains the channel bottom width [m or ft]
`units`	character vector that contains the system of units [options are `SI` for International System of Units and `Eng` for English (US customary) units. This is used for compatibility with iemisc package.
`ret_units`	If set to TRUE the value(s) returned are of class `units` with units attached to the value. [Default is FALSE]

Details

The Manning equation characterizes open channel flow conditions under uniform flow conditions:

$$Q = A\frac{C}{n}R^{\frac{2}{3}}S_f{}^{\frac{1}{2}}$$

where C is 1.0 for SI units and 1.49 for Eng (U.S. Customary) units. Using the geometric relationships for hydraulic radius and cross-sectional area of a trapezoid, it takes the form:

$$Q = \frac{C}{n}\frac{\left(by + my^2\right)^{\frac{5}{3}}}{\left(b + 2y\sqrt{1+m^2}\right)^{\frac{2}{3}}}S_f{}^{\frac{1}{2}}$$

Critical depth is defined by the relation (at critical conditions):

$$\frac{Q^2 B}{g\, A^3} = 1$$

where B is the top width of the water surface. For a given Q, m, n, and Sf, the most hydraulically efficient channel is found by maximizing R, which can be done by setting in the Manning equation $\frac{\partial R}{\partial y} = 0$. This produces:

$$y_{opt} = 2^{\frac{1}{4}} \left(\frac{Qn}{C \left(2\sqrt{1+m^2} - m\right) S_f^{\frac{1}{2}}} \right)^{\frac{3}{8}}$$

$$b_{opt} = 2y_{opt} \left(\sqrt{1+m^2} - m \right)$$

Value

Returns a list including the missing parameter:

- Q - flow rate
- V - flow velocity
- A - cross-sectional area of flow
- P - wetted perimeter
- R - hydraulic radius
- y - flow depth (normal depth)
- b - channel bottom width
- m - channel side slope
- Sf - slope
- B - top width of water surface
- n - Manning's roughness
- yc - critical depth
- Fr - Froude number
- Re - Reynolds number
- bopt - optimal bottom width (returned if solving for b)
- yopt - optimal water depth (returned if solving for y)

Author(s)

Ed Maurer, Irucka Embry

See Also

`spec_energy_trap` for specific energy diagram and `xc_trap` for a cross-section diagram of the trapezoidal channel

Examples

```
#Solving for flow rate, Q, trapezoidal channel: SI Units
manningt(n = 0.013, m = 2, Sf = 0.0005, y = 1.83, b = 3, units = "SI")
#returns Q=22.2 m3/s

#Solving for roughness, n, rectangular channel: Eng units
manningt(Q = 14.56, m = 0, Sf = 0.0004, y = 2.0, b = 4, units = "Eng")
#returns Manning n of 0.016

#Solving for depth, y, triangular channel: SI units
manningt(Q = 1.0, n = 0.011, m = 1, Sf = 0.0065, b = 0, units = "SI")
#returns 0.6 m normal flow depth
```

---

moody *Creates a Moody diagram with optional manually added points*

Description

This function plots a standard Moody diagram, and allows additional points to be added by including arguments Re and f.

Usage

```
moody(Re = NULL, f = NULL)
```

Arguments

Re	(optional) numeric vector that contains the Reynolds numbers of points to be manually added
f	(optional) numeric vector (same length as Re) that contains the Darcy-Weisbach friction factors corresponding to the points to be manually added

Value

a Moody diagram, with the optional added (Re, f) points

Author(s)

Ed Maurer

Examples

```
# Draw canonical Moody diagram
moody()

# Draw Moody diagram plotting two additional points
Re = c(10000, 100000)
f = c(0.04, 0.03)
moody( Re = Re, f = f )
```

`operpoint` *Uses input pump and system curves to find the operating point for a pump and create a plot.*

Description

Uses input pump and system curves to find the operating point for a pump and create a plot.

Usage

```
operpoint(pcurve = NULL, scurve = NULL)
```

Arguments

`pcurve`	A pumpcurve object
`scurve`	A systemcurve object

Value

Returns a list including:

- Qop - flow at the operating point [m^3s^{-1} or ft^3s^{-1}]
- hop - head at the operating point [m or ft]
- p - a plot object of the curves

Author(s)

Ed Maurer

See Also

`pumpcurve` and `systemcurve` for input object preparation

pumpcurve — *Fits a polynomial curve to three or more points from a pump characteristic curve to be used in solving for an operating point of the pump in a piping system.*

Description

Fits a polynomial curve to three or more points from a pump characteristic curve. This allows solving for an operating point of the pump in a piping system. A portion of this is based on https://github.com/PhDMeiwp/basicTrendline/blob/master/R/trendline.R

Usage

```
pumpcurve(Q = NULL, h = NULL, eq = "poly1", units = c("SI", "Eng"))
```

Arguments

Q	Numeric vector of flow rates for selected points on the pump curve [m^3s^{-1} or ft^3s^{-1}]
h	Numeric vector of heads for selected points on the pump curve [m or ft]
eq	Character vector identifying the for of equation to fit (see details)
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units.

Details

The form of the equation fit to the input points may be one of the following, as determined by the eq input parameter.

eq	equation form
poly1	$h = a + bQ + cQ^2$
poly2	$h = a + cQ^2$
poly3	$h = h_{shutoff} + cQ^2$

where $h_{shutoff}$ is the head on the pump curve associated with $Q = 0$. The shutoff head at $Q = 0$ should be included in the input vectors if the poly3 equation form is to be used.

Value

Returns an object of class pumpcurve consisting of a list including:

- curve - a function defining the curve that is fit to the data
- eqn - a character vector of the equation for the curve
- r2 - the coefficient of determination for the curve fit, R^2
- p - a plot object of the fit curve
- units - the units system passed to the function

Author(s)

Ed Maurer

Examples

```
#Input in Eng units - use \code{units} package for easy unit conversion
qgpm <- units::set_units(c(0, 5000, 7850), gallons/minute)
qcfs <- units::set_units(qgpm, ft^3/s)
hft <- c(81, 60, 20) #units are already in ft so setting units is optional
pumpcurve(Q = qcfs, h = hft, eq = "poly2", units = "Eng")
```

---

sequent_depth *Solves the Momentum Equation for sequent (or conjugate) depth in a trapezoidal channel*

Description

This function solves the Momentum equation for water flow in an open channel with a trapezoidal shape and determines the sequent (conjugate) depth. This is the flow depth either upstream or downstream of a hydraulic jump, whichever is not provided as input.

Usage

```
sequent_depth(
  Q = NULL,
  b = NULL,
  y = NULL,
  m = NULL,
  units = c("SI", "Eng"),
  ret_units = FALSE
)
```

Arguments

Q	numeric vector that contains the flow rate [m^3s^{-1} or ft^3s^{-1}]
b	numeric vector that contains the channel bottom width [m or ft]
y	numeric vector that contains the water depth [m or ft]
m	numeric vector that contains the side slope of the channel (m:1 H:V) [unitless]
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units. This is used for compatibility with iemisc package.
ret_units	If set to TRUE the value(s) returned are of class units with units attached to the value. [Default is FALSE]

Details

The Momentum equation for open channel flow conditions in a trapezoidal channel:

$$M = \frac{by^2}{2} + \frac{my^3}{3} + \frac{Q^2}{gy\,(b + my)}$$

where C is 1.0 for SI units and 1.49 for Eng (U.S. Customary) units. The momentum function is assumed to be the same on both sides of a hydraulic jump, allowing the determination of the sequent depth.

Value

Returns a list including:

- y - input depth
- y_seq - sequent depth
- yc - critical depth
- Fr - Froude number for input depth
- Fr_seq - Froude number for sequent depth
- E - specific energy for input depth
- E_seq - specific energy for sequent depth

Author(s)

Ed Maurer

Examples

```
#Solving for sequent depth: SI Units
#Flow of 0.2 m^3/s, bottom width = 0.5 m, Depth = 0.1 m, side slope = 1:1
sequent_depth(Q=0.2,b=0.5,y=0.1,m=1,units = "SI", ret_units = TRUE)
```

---

spec_energy_trap *Creates a specific energy diagram for a trapezoidal channel*

Description

This function plots a specific energy diagram of a trapezoidal (including rectangular and triangular) channel, with annotation of critical depth and minimum specific energy.

Usage

```
spec_energy_trap(
  Q = NULL,
  b = NULL,
  m = NULL,
  y = NULL,
  scale = 3,
  units = c("SI", "Eng")
)
```

Arguments

Q	flow rate [m^3s^{-1} or ft^3s^{-1}]
b	bottom width [m or ft]
m	side slope (H:1) [unitless]
y	depth(s) of flow (a numeric vector of length <= 2) [m or ft] (optional)
scale	multiplier (of yc) for axis scales (default is 3)
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units.

Details

Specific Energy, E, is the energy, expressed as a head (i.e., the mechanical energy per unit weight of the water, with units of length) relative to the channel bottom. It is calculated as:

$$E = y + \alpha \frac{Q^2}{2g\,A^2} = y + \alpha \frac{V^2}{2g}$$

where y is flow depth, A is the cross-sectional flow area, $V = \frac{Q}{A}$, and and α is a kinetic energy correction factor to account for non-uniform velocities across the cross-section; $\alpha = 1.0$ in this function (as is commonly assumed).

Value

a specific energy diagram

Author(s)

Ed Maurer

Examples

```
# Draw a specific energy diagram for a cross-section with flow 1, width 2, side slope 3:1 (H:V)
spec_energy_trap(Q = 1.0, b = 2.0, m = 3.0, scale = 4, units = "SI")

# Draw the same specific energy diagram adding lines for depths, y = 0.5 and 0.8 m
spec_energy_trap(Q = 1.0, b = 2.0, m = 3.0, scale = 4, y = c(0.5, 0.8), units = "SI")
```

`systemcurve` *Creates a system curve for a piping system using the static head and a coefficient.*

Description

Creates a system curve for a piping system using the static head and a coefficient.

Usage

```
systemcurve(hs = NULL, K = NULL, units = c("SI", "Eng"))
```

Arguments

`hs`	Numeric value of the static head [m or ft]
`K`	Numeric value of the coefficient in the equation $h = hs + KQ^2$ where Q has units of m^3s^{-1} or ft^3s^{-1}
`units`	character vector that contains the system of units [options are `SI` for International System of Units and `Eng` for English (US customary) units.

Value

Returns an object of class `systemcurve` consisting of a list including:

- curve - a function defining the system curve
- eqn - a character vector of the equation for the curve
- units - the units system passed to the function

Author(s)

Ed Maurer

Examples

```
#Input in Eng units. Coefficient can be calculated manually or using
#other package functions for friction loss in a pipe system using \eqn{Q=1}
ans <- darcyweisbach(Q = 1,D = 20/12, L = 3884, ks = 0.0005, nu = 1.23e-5, units = "Eng")
systemcurve(hs = 30, K = ans$hf, units = "Eng")
```

waterprops *Functions to calculate water properties: density, specific weight, dynamic and kinematic viscosity, saturation vapor pressure, surface tension, and bulk modulus.*

Description

This function calculates water properties that are used in other functions.

Usage

```
dvisc(T = NULL, units = NULL, ret_units = FALSE)

dens(T = NULL, units = NULL, ret_units = FALSE)

specwt(T = NULL, units = NULL, ret_units = FALSE)

kvisc(T = NULL, units = NULL, ret_units = FALSE)

svp(T = NULL, units = NULL, ret_units = FALSE)

surf_tension(T = NULL, units = NULL, ret_units = FALSE)

Ev(T = NULL, units = NULL, ret_units = FALSE)
```

Arguments

T	the water temperature [$^\circ C$ or $^\circ F$]
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units. This is used for compatibility with iemisc package
ret_units	If set to TRUE the value(s) returned are of class units with units attached to the value. [Default is FALSE]

Value

rho, the density of water for the dens function [$kg\,m^{-3}$ or $slug\,ft^{-3}$]

spwt, the specific weight of water for the specwt function [$N\,m^{-3}$ or $lbf\,ft^{-3}$]

mu, the dynamic viscosity of water for the dvisc function [$N\,s\,m^{-2}$ or $lbf\,s\,ft^{-2}$]

nu, the kinematic viscosity of water for the kvisc function [$m^2 s^{-1}$ or $ft^2 s^{-1}$].

svp, the saturation vapor pressure of water for the svp function [Nm^{-2} or $lbfft^{-2}$].

surf_tension, the surface tension of water for the surf_tension function [Nm^{-1} or $lbfft^{-1}$].

Ev, the bulk modulus of elasticity of water for the Ev function [Nm^{-2} or $lbfft^{-2}$].

Author(s)

Ed Maurer

Examples

```
#Find kinematic viscocity for water temperature of 55 F
nu = kvisc(T = 55, units = 'Eng')

#Find kinematic viscocity assuming default water temperature of 68 F
nu = kvisc(units = 'Eng')

#Find water density for water temperature of 25 C
rho = dens(T = 25, units = 'SI')

#Find saturation vapor pressure for water temperature of 10 C
vps = svp(T = 10, units = 'SI')

#Find surface tension for water temperature of 10 C
s_tens = surf_tension(T = 10, units = 'SI')
```

`water_table` *Tabulates into a tibble the basic water properties: density, dynamic and kinematic viscosity, saturation vapor pressure, surface tension, and bulk modulus.*

Description

Tabulates into a tibble the basic water properties: density, dynamic and kinematic viscosity, saturation vapor pressure, surface tension, and bulk modulus.

Usage

```
water_table(units = c("SI", "Eng"), ret_units = TRUE)
```

Arguments

`units`	character vector that contains the system of units [options are `SI` for International System of Units and `Eng` for English (US customary) units. This is used for compatibility with iemisc package
`ret_units`	If set to TRUE the value(s) returned are of class `units` with units attached to the value. [Default is TRUE]

Author(s)

Ed Maurer

Examples

```
water_table(units = 'SI')
```

`xc_circle` *Creates a cross-section plot for a partially filled pipe*

Description

This function plots a cross-section of a circular pipe, shaded as filled to the level indicated by the depth and diameter values passed to it.

Usage

```
xc_circle(y = NULL, d = NULL, units = c("SI", "Eng"))
```

Arguments

`y`	water depth [m or ft]
`d`	pipe diameter [m or ft]
`units`	character vector that contains the system of units [options are `SI` for International System of Units and `Eng` for English (US customary) units.

Value

a cross-section diagram

Author(s)

Ed Maurer

Examples

```
# Draw a cross-section with diameter 1.0 and depth 0.7
xc_circle(y = 0.7, d = 1.0, units = "SI")
```

xc_trap *Creates a cross-section plot for an open channel*

Description

This function plots a cross-section of a (trapezoid, rectangle, triangle), shaded as filled to the level indicated by the values passed to it.

Usage

```
xc_trap(y = NULL, b = NULL, m = NULL, units = c("SI", "Eng"))
```

Arguments

y	water depth [m or ft]
b	bottom width [m or ft]
m	side slope (H:1)
units	character vector that contains the system of units [options are SI for International System of Units and Eng for English (US customary) units.

Value

a cross-section diagram

Author(s)

Ed Maurer

Examples

```
# Draw a cross-section with depth 1, width 2, side slope 3:1 (H:V)
xc_trap(y = 1.0, b = 2.0, m = 3.0, units = "SI")
```

Index