iemisc: Calculating the Friction Loss Examples

Irucka Embry, E.I.T. (EcoC²S)

2023-02-12

Contents

Replicate the R code

Note: If you wish to replicate the R code below, then you will need to copy and paste the following commands in R first (to make sure you have all the packages and their dependencies):

```r
install.packages(c("install.load", "iemisc", "data.table", "units", "flextable",
    "pracma"))
# install the packages and their dependencies


# load the required packages
install.load::load_package("iemisc", "data.table", "units", "flextable")
# load needed packages using the load_package function from the install.load
# package (it is assumed that you have already installed these packages)


import::from(pracma, newtonRaphson)
# import newtonRaphson from the pracma package
```

Example 1 Set

```r
# Please note that the f2, f3, f4, f5, f6, f7, f8, and the relerror functions
# are found within the iemisc R package created by Irucka Embry

# Example 1 -- Example 17.2 (Lindeburg Manual Reference)

Re <- 4e+05

eps <- 0.004

D <- 1

eps/D

## [1] 0.004

# Answers from the Lindeburg Manual Reference text: 0.028 - Moody diagram
# 0.0287 - Appendix 17.B 'Darcy Friction Factors (turbulent flow)' 0.0288 -
# Swamee-Jain approximation 0.0287 - Colebrook equation

f2 <- f2(eps = eps, D = D, Re = Re)
f2

## [1] 0.02944312

f3 <- f3(eps = eps, D = D, Re = Re)
f3

## [1] 0.0286854

f4 <- f4(eps = eps, D = D, Re = Re)
f4

## [1] 0.02867517

f5 <- f5(eps = eps, D = D, Re = Re)
f5

## [1] 0.02869798

f6 <- f6(eps = eps, D = D, Re = Re)
f6

## [1] 0.02881149

f7 <- f7(eps = eps, D = D, Re = Re)
f7

## [1] 0.02869798

f8 <- f8(eps = eps, D = D, Re = Re)
f8

## [1] 0.02867606

# determine the relative error
acc <- 0.0287

relerror(acc, f2)
```

```
## [1] 2.589255
relerror(acc, f3)

## [1] 0.05086994
relerror(acc, f4)

## [1] 0.08652119
relerror(acc, f5)

## [1] 0.00704152
relerror(acc, f6)

## [1] 0.3884771
relerror(acc, f7)

## [1] 0.007044366
relerror(acc, f8)

## [1] 0.08341057
```

Example 2 Set

```
# Please note that the f2, f3, f4, f5, f6, f7, and the f8 functions are found
# within the iemisc R package created by Irucka Embry

# Example 2 (EngineerExcel Reference)

eps <- 5e-05

D <- 0.0254

Re <- 6000

# f equal to 0.0375 from Microsoft Excel Goal Seek

f2(eps = eps, D = D, Re = Re)

## [1] 0.0379846
f3(eps = eps, D = D, Re = Re)

## [1] 0.03555518
f4(eps = eps, D = D, Re = Re)

## [1] 0.03549632
f5(eps = eps, D = D, Re = Re)

## [1] 0.03781183
```

```r
f6(eps = eps, D = D, Re = Re)
```

```
## [1] 0.03843287
```

```r
f7(eps = eps, D = D, Re = Re)
```

```
## [1] 0.03549702
```

```r
f8(eps = eps, D = D, Re = Re)
```

```
## [1] 0.03781382
```

Problem 1 Statement

Example 1 [Kudela]

"Oil, with $\rho = 900$ kg/m^3 and kinematic coefficient of viscosity $\nu = 0,00001$ m^2/s, flows at $\mathrm{q_v}$ = 0,2 m^3/s through 500 m of 200-mm diameter cast-iron pipe. Determine the head loss."

"Absolute roughness for iron-cast pipe is $\epsilon = 0.26$ mm."

Solution 1

```r
# Please note that the Re2, f2, f3, f4, f5, f6, f7, f8, and the colebrook
# functions are found within the iemisc R package created by Irucka Embry


# oil iron-cast pipe find the friction loss -- the head loss is 117 meters


# given the water flow of 0.2 m^3/s create a numeric vector with the units of
# cubic meters per second for the volumetric flow rate
Vdot <- set_units(0.2, m^3/s)
Vdot
```

```
## 0.2 [m^3/s]
```

```r
# given length of 500 m create a numeric vector with the units of meters
L_SI <- set_units(500, m)
L_SI
```

```
## 500 [m]
```

```r
g_SI <- set_units(9.80665, m/s^2)
g_SI
```

```
## 9.80665 [m/s^2]
```

```r
# given saturated liquid density of oil (SI units)
rho_SI <- set_units(900, kg/m^3)
rho_SI
```

```
## 900 [kg/m^3]
```

```r
# given kinematic viscosity of oil (SI units)
nu_SI <- set_units(1e-05, m^2/s)
nu_SI
```

```
## 1e-05 [m^2/s]
```

```r
# create a numeric vector with the units of millimeters for the given specific
# roughness
epsilon <- set_units(0.26, mm)
epsilon
```

```
## 0.26 [mm]
```

```r
# create a numeric vector with the units of meters for the given specific
# roughness
epsilon <- epsilon

units(epsilon) <- make_units(m)
epsilon
```

```
## 0.00026 [m]
```

```r
# create a numeric vector with the units of millimeters for the given internal
# pipe diameter
Di <- set_units(200, mm)
Di
```

```
## 200 [mm]
```

```r
# create a numeric vector with the units of meters for the given internal pipe
# diameter
units(Di) <- make_units(m)
Di
```

```
## 0.2 [m]
```

```r
# relative roughness (dimensionless) of the cast iron pipe
rel_roughness <- epsilon/Di
rel_roughness
```

```
## 0.0013 [1]
```

```r
# internal area of the cast iron pipe
Ai <- Di^2 * pi/4
Ai
```

```
## 0.03141593 [m^2]
```

```r
# average velocity of the flowing water
V <- Vdot/Ai
V
```

```
## 6.366198 [m/s]
```

```r
# Reynolds number using the kinematic viscosity
Re_SI <- Re2(D = drop_units(Di), V = drop_units(V), nu = drop_units(nu_SI))
Re_SI
```

```
## [1] 127324
```

```r
# Darcy friction factor (f) for cast iron pipe Moody equation
fr2_SI <- f2(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI)

# Romeo, et. al. equation
fr3_SI <- f3(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI)

# Žarko Ćojbašića and Dejan Brkić equation
fr4_SI <- f4(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI)

# Colebrook-White equation
fr5_SI <- f5(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI)

# Colebrook-White equation from Didier Clamond
colebrook_SI <- colebrook(Re_SI, K = drop_units(rel_roughness))

# Swamee-Jaine equation
fr6_SI <- f6(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI)

# Zigrang-Sylvester equation
fr7_SI <- f7(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI)

# Vatankhah equation
fr8_SI <- f8(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI)


# friction loss for cast iron pipe
hf_SI1 <- (f2(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI) * drop_units(L_SI) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_SI))

hf_SI2 <- (f3(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI) * drop_units(L_SI) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_SI))

hf_SI3 <- (f4(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI) * drop_units(L_SI) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_SI))

hf_SI4 <- (f5(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI) * drop_units(L_SI) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_SI))

hf_SI5 <- (colebrook(Re_SI, K = drop_units(rel_roughness)) * drop_units(L_SI) * drop_units(V)^2)/(2 *
    drop_units(Di) * drop_units(g_SI))

hf_SI6 <- (f6(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI) * drop_units(L_SI) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_SI))

hf_SI7 <- (f7(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI) * drop_units(L_SI) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_SI))

hf_SI8 <- (f8(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_SI) * drop_units(L_SI) *
```

```r
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_SI))


# result table
result_table_SI <- data.table(V1 = c("Moody equation", "Romeo, et. al. equation",
    "Žarko Ćojbašića and Dejan Brkić equation", "Colebrook-White equation",
    "Colebrook-White equation from Didier Clamond", "Swamee-Jaine equation", "Zigrang-Sylvester equatio
    "Vatankhah equation"), V2 = c(fr2_SI, fr3_SI, fr4_SI, fr5_SI, colebrook_SI, fr6_SI,
    fr7_SI, fr8_SI), V3 = c(hf_SI1, hf_SI2, hf_SI3, hf_SI4, hf_SI5, hf_SI6, hf_SI7,
    hf_SI8))

setnames(result_table_SI, c("Darcy friction factor equation", "Darcy friction factor (f) for cast iron p
    "Friction loss for cast iron pipe over total length"))


prettySI <- flextable(result_table_SI)
colkeys <- c("Darcy friction factor equation", "Darcy friction factor (f) for cast iron pipe",
    "Friction loss for cast iron pipe over total length")
prettySI <- colformat_num(x = prettySI, col_keys = colkeys, big.mark = ",", digits = 4,
    na_str = "N/A")
print(prettySI, preview = "pdf", align = "center")

## a flextable object.
## col_keys: `Darcy friction factor equation`, `Darcy friction factor (f) for cast iron pipe`, `Frictio
## header has 1 row(s)
## body has 8 row(s)
## original dataset sample:
##                  Darcy friction factor equation
## 1                                Moody equation
## 2                       Romeo, et. al. equation
## 3      Žarko Ćojbašića and Dejan Brkić equation
## 4                      Colebrook-White equation
## 5 Colebrook-White equation from Didier Clamond
##   Darcy friction factor (f) for cast iron pipe
## 1                                   0.02329232
## 2                                   0.01744932
## 3                                   0.01743092
## 4                                   0.02272431
## 5                                   0.01711496
##   Friction loss for cast iron pipe over total length
## 1                                          120.32680
## 2                                           90.14221
## 3                                           90.04714
## 4                                          117.39249
## 5                                           88.41489
```

Henryk Kudela calculated 117 meters for the head loss.

Problem 2 Statement

Example 1 [Subramanian]

Find the head loss due to the flow of 1,500 gpm of oil ($\nu = 1.15 \times 10^{-4}$ ft^2/s) through 1,600 feet of 8" diameter cast iron pipe. The density of the oil $\rho = 1.75$ slug/ft^3.

"For cast iron, $\epsilon = 8.5 \times 10^{-4}$ ft."

Solution 2

```
# Please note that the Re2, f2, f3, f4, f5, f6, f7, f8, and the colebrook
# functions are found within the iemisc R package created by Irucka Embry


# oil cast iron pipe find the head loss -- the head loss is 83.7 feet

# given the water flow of 1500 gpm (gal / min) create a numeric vector with the
# units of gallons per minute for the volumetric flow rate
Vdot <- set_units(1500, gallon/min)
Vdot
```

```
## 1500 [gallon/min]
```

```
# create a numeric vector with the units of cubic feet per second for the
# volumetric flow rate
units(Vdot) <- make_units(ft^3/sec)
Vdot
```

```
## 3.342014 [ft^3/s]
```

```
# given length of 1600 ft create a numeric vector with the units of feet
L_Eng <- set_units(1600, ft)
L_Eng
```

```
## 1600 [ft]
```

```
# create a numeric vector for gravity (US Customary units)
g_Eng <- set_units(9.80665 * (3937/1200), ft/sec^2)
g_Eng
```

```
## 32.17398 [ft/s^2]
```

```
# given saturated liquid density of oil (US Customary units)
rho_Eng <- set_units(1.75, slug/ft^3)
rho_Eng
```

```
## 1.75 [slug/ft^3]
```

```
# given kinematic viscosity of oil (US Customary units)
nu_Eng <- set_units(0.000115, ft^2/sec)
nu_Eng
```

```
## 0.000115 [ft^2/s]
```

```r
# create a numeric vector with the units of feet for the given specific
# roughness
epsilon <- set_units(0.00085, ft)
epsilon

## 0.00085 [ft]

# create a numeric vector with the units of inch for the given internal pipe
# diameter
Di <- set_units(8, inch)
Di

## 8 [inch]

# create a numeric vector with the units of feet for the given internal pipe
# diameter
units(Di) <- make_units(ft)
Di

## 0.6666667 [ft]

# relative roughness (dimensionless) of the cast iron pipe
rel_roughness <- epsilon/Di
rel_roughness

## 0.001275 [1]

# internal area of the cast iron pipe
Ai <- Di^2 * pi/4
Ai

## 0.3490659 [ft^2]

# average velocity of the flowing water
V <- Vdot/Ai
V

## 9.574165 [ft/s]

# Reynolds number using the kinematic viscosity
Re_Eng <- Re2(D = drop_units(Di), V = drop_units(V), nu = drop_units(nu_Eng))
Re_Eng

## [1] 55502.41

# Darcy friction factor (f) for cast iron pipe Moody equation
fr2_Eng <- f2(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng)

# Romeo, et. al. equation
fr3_Eng <- f3(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng)

# Žarko Ćojbašića and Dejan Brkić equation
fr4_Eng <- f4(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng)

# Colebrook-White equation
fr5_Eng <- f5(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng)

# Colebrook-White equation from Didier Clamond
colebrook_Eng <- colebrook(Re_Eng, K = drop_units(rel_roughness))
```

```r
# Swamee-Jaine equation
fr6_Eng <- f6(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng)

# Zigrang-Sylvester equation
fr7_Eng <- f7(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng)

# Vatankhah equation
fr8_Eng <- f8(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng)


# friction loss for cast iron pipe
hf_Eng1 <- (f2(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))

hf_Eng2 <- (f3(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))

hf_Eng3 <- (f4(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))

hf_Eng4 <- (f5(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))

hf_Eng5 <- (colebrook(Re_Eng, K = drop_units(rel_roughness)) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))

hf_Eng6 <- (f6(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))

hf_Eng7 <- (f7(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))

hf_Eng8 <- (f8(eps = drop_units(epsilon), D = drop_units(Di), Re = Re_Eng) * drop_units(L_Eng) *
    drop_units(V)^2)/(2 * drop_units(Di) * drop_units(g_Eng))


# result table
result_table_Eng <- data.table(V1 = c("Moody equation", "Romeo, et. al. equation",
    "Žarko Ćojbašića and Dejan Brkić equation", "Colebrook-White equation",
    "Colebrook-White equation from Didier Clamond", "Swamee-Jaine equation", "Zigrang-Sylvester equatio
    "Vatankhah equation"), V2 = c(fr2_Eng, fr3_Eng, fr4_Eng, fr5_Eng, colebrook_Eng,
    fr6_Eng, fr7_Eng, fr8_Eng), V3 = c(hf_Eng1, hf_Eng2, hf_Eng3, hf_Eng4, hf_Eng5,
    hf_Eng6, hf_Eng7, hf_Eng8))

setnames(result_table_Eng, c("Darcy friction factor equation", "Darcy friction factor (f) for cast iron
    "Friction loss for cast iron pipe over total length"))


prettyEng <- flextable(result_table_Eng)
colkeys <- c("Darcy friction factor equation", "Darcy friction factor (f) for cast iron pipe",
    "Friction loss for cast iron pipe over total length")

prettyEng <- colformat_num(x = prettyEng, col_keys = colkeys, big.mark = ",", digits = 4,
```

```
    na_str = "N/A")

print(prettyEng, preview = "pdf", align = "center")

## a flextable object.
## col_keys: `Darcy friction factor equation`, `Darcy friction factor (f) for cast iron pipe`, `Frictio
## header has 1 row(s)
## body has 8 row(s)
## original dataset sample:
##                 Darcy friction factor equation
## 1                                Moody equation
## 2                         Romeo, et. al. equation
## 3     Žarko Ćojbašića and Dejan Brkić equation
## 4                        Colebrook-White equation
## 5 Colebrook-White equation from Didier Clamond
##   Darcy friction factor (f) for cast iron pipe
## 1                                   0.02484564
## 2                                   0.02240349
## 3                                   0.02238161
## 4                                   0.02444434
## 5                                   0.02041294
##   Friction loss for cast iron pipe over total length
## 1                                           84.94316
## 2                                           76.59387
## 3                                           76.51904
## 4                                           83.57119
## 5                                           69.78847
```

R. Shankar Subramanian calculated 83.7 feet for the head loss.

Works Cited

Charlie Young, P.E., EngineerExcel, "Colebrook Equation Solver in Excel", https://www.engineerexcel.com/colebrook-equation-solver-in-excel/

Henryk Kudela, "Hydraulic losses in pipes", page 5, Wrocław University of Science and Technology Department of Mechanical and Power Engineering, http://fluid.itcmp.pwr.wroc.pl/~znmp/dydaktyka/fundam_FM/Lecture11_12.pdf

Michael R. Lindeburg, PE, *Civil Engineering Reference Manual for the PE Exam*, Twelfth Edition, Belmont, California: Professional Publications, Inc., 2011, pages 17-5 - 17-7.

R. Shankar Subramanian, "Pipe Flow Calculations", pages 6-7, Clarkson University Department of Chemical and Biomolecular Engineering, https://web2.clarkson.edu/projects/subramanian/ch330/notes/Pipe%20Flow%20Calculations.pdf

The NIST Reference on Constants, Units, and Uncertainty, Fundamental Constants Data Center of the NIST Physical Measurement Laboratory, "standard acceleration of gravity g_n", https://physics.nist.gov/cgi-bin/cuu/Value?gn.

Wikimedia Foundation, Inc. Wikipedia, 15 May 2019, “Conversion of units”, https://en.wikipedia.org/wiki/Conversion_of_units.

EcoC^{2}S Links

EcoC^{2}S Home – https://www.ecoccs.com/
About EcoC^{2}S – https://www.ecoccs.com/about_ecoc2s.html
Products – https://www.questionuniverse.com/products.html
EcoC^{2}S Media – https://www.ecoccs.com/media.html
EcoC^{2}S Resources – https://www.ecoccs.com/resources.html
R Trainings and Resources provided by EcoC^{2}S (Irucka Embry, E.I.T.) – https://www.ecoccs.com/rtraining.html

Copyright and License