R Notebook

Calculate VLP using Hagedorn-Brown (no heat transfer)

- vertical well

```
library(rNodal)

# Example from C.13 in Brown
# P2 (pressure at end point is given).
# The question is: what is the length of the tubing.
# P2 = 1000 psia

input.example.C13 <- setWellInput(
                        field.name = "HAGBR.MOD",
                        well.name = "Brown_C13",
                        depth.wh = 0, depth.bh = 2670,
                        diam.in = 1.995,
                        GLR = 500,
                        liq.rt = 1000,
                        wcut = 0.6,
                        thp = 500,
                        tht = 120,
                        bht = 150,
                        API = 22,
                        gas.sg = 0.65,
                        wat.sg = 1.07,
                        if.tens = 30)


well.model <- setVLPmodel(vlp.model = "hagbr.mod",
                          segments = 11,
                          tol = 0.000001)

# c13_output <- runVLP(well.input = input.example.C13, well.model)
# c13_output
```

VLP calculation with Heat transfer calculations

- Vertical well
- Angle constant

It will grab only few variables from the VLP final table above.

```
c("depth", "dL", "pres", "temp")
```

```
# calculate the fluid temperature in the well
# input: deviation survey and well calculated parameters: uses new functions:
#        get_well_parameters
#        rNodal:::temp.gradient

library(rNodal)

# the standard well input
```

```r
# Note that U (heat transfer coefficient is added at the end)
input.example.C13 <- setWellInput(
                        field.name = "HAGBR.MOD",
                        well.name = "Brown_C13",
                        depth.wh = 0, depth.bh = 2670,
                        diam.in = 1.995,
                        GLR = 500,
                        liq.rt = 1000,
                        wcut = 0.6,
                        thp = 500,
                        tht = 120,
                        bht = 150,
                        API = 22,
                        gas.sg = 0.65,
                        wat.sg = 1.07,
                        U = 17)

# well model
well.model <- setVLPmodel(vlp.model = "hagbr.mod",
                          segments = 11,
                          tol = 0.000001)

# run VLP only to get the initial part of the table
# TODO: maybe we need to add a function that only return the simple stuff
# such as depth, dL
vlp_output <- runVLP(well.input = input.example.C13,
                     well.model)

names(vlp_output)
#>  [1] "i"           "depth"       "dL"          "temp"        "pres"
#>  [6] "p_avg"       "t_avg"       "segment"     "GOR"         "Rs"
#> [11] "gas.fvf"     "gas.free"    "liq.dens"    "z"           "gas.dens"
#> [16] "oil.visc"    "wat.visc"    "mixL.visc"   "oil.fvf"     "wat.fvf"
#> [21] "liq.svel"    "gas.svel"    "NL"          "CNL"         "NLV"
#> [26] "NGV"         "A"           "B"           "BA"          "ND"
#> [31] "X2"          "HL.psi"      "X2.mod"      "psi"         "HL"
#> [36] "Re.TP"       "ff"          "mix.dens"    "mixL.volume" "mixL.dens"
#> [41] "mixTP.dens"  "mixTP.svel"  "elev.grad"   "fric.grad"   "dp.dz"
```

```r
# get only the variables we need for heat transfer. But what we really want
# is the deviation survey: MD, TVD
vlp_output <- vlp_output[, c("depth", "dL", "pres", "temp")]
```

```r
# get initial well parameters including basic calculations
well_parameters <- get_well_parameters(input.example.C13)
names(well_parameters)
#>  [1] "field.name"   "well.name"    "depth.wh"     "tht"
#>  [5] "depth.bh"     "bht"          "diam.in"      "ed"
#>  [9] "thp"          "liq.rt"       "wcut"         "API"
#> [13] "oil.visc"     "gas.sg"       "GLR"          "wat.sg"
#> [17] "salinity"     "if.tens"      "U"            "oil.cp"
#> [21] "gas.cp"       "wat.cp"       "angle"        "temp.grad"
#> [25] "diam"         "area"         "diam.ft"      "oil.sg"
#> [29] "oil.fraction" "wat.fraction" "WOR"          "oil.rt"
#> [33] "gas.rt"       "wat.rt"       "mass.total"   "GOR"
#> [37] "mass.rt"      "mass.rate"    "cp.avg"
```

```
# heat capacity is a scalar from basic calculations
well_parameters$cp.avg
#> [1] 0.6766667
```

```
# temp.gradient calculates the fluid temperature coming from the wellbore
# inputs are depth, dL, pres, temp plus basic calculations
rNodal:::temp.gradient(vlp_output, well_parameters)
#> # A tibble: 12 x 6
#>    depth    dL  pres  temp     L    Ti
#>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#>  1     0     0   500   120  2670   136.
#>  2  243.  243.  548.  123. 2427.  139.
#>  3  485.  243.  596.  125. 2185.  142.
#>  4  728.  243.  644.  128. 1942.  144.
#>  5  971.  243.  693.  131. 1699.  146.
#>  6 1214.  243.  742.  134. 1456.  147.
#>  7 1456.  243.  791.  136. 1214.  148.
#>  8 1699.  243.  841.  139.   971.  149.
#>  9 1942.  243.  891.  142.   728.  150.
#> 10 2185.  243.  941.  145.   485.  150.
#> 11 2427.  243.  992.  147.   243.  150.
#> 12 2670   243. 1043.  150.     0   150
```

Notes.

1. We don't need VLP calculations for the function temp_gradient(). We only need the depth table which is only one line.
2. We don't need either the whole `well_parameters` list. #' from well_parameters we need: U, angle, depth.bh, depth.wh, bht, tht, diam.ft, mass.rate, cp.avg. All are scalars
3. We should separate the well inputs from the initial calculations instead of merging them in `well_parameters`. It would be clearer and more usable. Have more sense.
4. We should create a function that only returns the depth points. Also another one that add the ground temperature. The standalone function could be called from different scripts.
5. The calculations in temp.gradient are independent of pressure.
6. We could suply the table of depths and fluid temperature to the VLPControl function and let the algorithm calculate the average temperature from the new table that considers heat transfer.
7. Avoid using `temp.grad` which only considers the earth temperature not the fluid's. The temperature gradient should not be a scalar but a vector as calculated by `temp.gradient`.

Same as above but calculating fluid temperature. Angle constant

Added new input parameters:

```
.  U = double 1= 8
.  oil.cp = double 1= 0.53
.  gas.cp = double 1= 0.5
.  wat.cp = double 1= 1
```

Added new calculated objects:

```
.  mass.rt = double 1= 378585
.  mass.rate = double 1= 378585
.  cp.avg = double 1= 0.67667
```

```r
# this tests if the new function get_well_parameters() returns all what's needed for heat transfer
library(rNodal)

well_table <- runVLP(well.input = input.example.C13,
                     well.model)[, c("depth", "dL", "pres", "temp")]

input.example.C13 <- setWellInput(field.name = "HAGBR.MOD",
                                   well.name = "Brown_C13",
                                   depth.wh = 0, depth.bh = 2670, diam.in = 1.995,
                                   GLR = 500, liq.rt = 1000, wcut = 0.6,
                                   thp = 500, tht = 120, bht = 150,
                                   API = 22, gas.sg = 0.65, wat.sg = 1.07,
                                 U = 17)
# input.example.C13
# getBasicCalcs(input.example.C13)
well_params <- get_well_parameters(input.example.C13)
Hmisc::list.tree(well_params, maxcomp = 40)
#>  well_params = list 39 (5416 bytes)
#> .  field.name = character 1= HAGBR.MOD
#> .  well.name = character 1= Brown_C13
#> .  depth.wh = double 1= 0
#> .  tht = double 1= 120
#> .  depth.bh = double 1= 2670
#> .  bht = double 1= 150
#> .  diam.in = double 1= 1.995
#> .  ed = double 1= 6e-04
#> .  thp = double 1= 500
#> .  liq.rt = double 1= 1000
#> .  wcut = double 1= 0.6
#> .  API = double 1= 22
#> .  oil.visc = double 1= 5
#> .  gas.sg = double 1= 0.65
#> .  GLR = double 1= 500
#> .  wat.sg = double 1= 1.07
#> .  salinity = double 1= 0
#> .  if.tens = double 1= 30
#> .  U = double 1= 17
#> .  oil.cp = double 1= 0.53
#> .  gas.cp = double 1= 0.5
#> .  wat.cp = double 1= 1
#> .  angle = double 1= 1.5708
#> .  temp.grad = double 1= 0.011236
#> .  diam = double 1= 0.16625
#> .  area = double 1= 0.021708
#> .  diam.ft = double 1= 0.16625
#> .  oil.sg = double 1= 0.92182
#> .  oil.fraction = double 1= 0.4
#> .  wat.fraction = double 1= 0.6
#> .  WOR = double 1= 1.5
#> .  oil.rt = double 1= 400
#> .  gas.rt = double 1= 5e+05
#> .  wat.rt = double 1= 600
#> .  mass.total = double 1= 378.59
#> .  GOR = double 1= 1250
#> .  mass.rt = double 1= 378585
#> .  mass.rate = double 1= 378585
#> .  cp.avg = double 1= 0.67667

# temp.gradient calculates the fluid temperature coming from the wellbore
rNodal:::temp.gradient(well_table, well_parameters)
#> # A tibble: 12 x 6
#>    depth    dL  pres  temp     L    Ti
#>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#>  1     0     0   500   120  2670   136.
#>  2  243.  243.  548.  123. 2427.  139.
#>  3  485.  243.  596.  125. 2185.  142.
#>  4  728.  243.  644.  128. 1942.  144.
#>  5  971.  243.  693.  131. 1699.  146.
#>  6 1214.  243.  742.  134. 1456.  147.
#>  7 1456.  243.  791.  136. 1214.  148.
#>  8 1699.  243.  841.  139.  971.  149.
#>  9 1942.  243.  891.  142.  728.  150.
#> 10 2185.  243.  941.  145.  485.  150.
#> 11 2427.  243.  992.  147.  243.  150.
#> 12 2670   243. 1043.  150.     0   150
```

old version using spelled-out parameters

```
# this in an old version where all well parameters had to be spelled out
# parameters necessary to calculate the fluid temperature
well_table <- runVLP(well.input = input.example.C13,
                     well.model)[, c("depth", "dL", "pres", "temp")]

theta   <-  pi /2
diam.in <- 1.995
diam.ft <- diam.in / 12
tht     <- 120
bht     <- 150
depth   <- 2670
ge      <- (bht - tht) / depth
mass.rate <- 228145
U <-  17
# U <- 4
cp.avg <- (0.53 + 0.5 + 1 ) / 3

# calculate dT/dx for the well
rNodal:::temp.fluid(well_table, theta, depth, bht, tht, U, cp.avg, diam.ft, mass.rate)
#> # A tibble: 12 x 6
#>    depth    dL  pres  temp     L    Ti
#>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#>  1     0     0   500   120  2670   129.
#>  2  243.  243.  548.  123. 2427.  133.
#>  3  485.  243.  596.  125. 2185.  137.
#>  4  728.  243.  644.  128. 1942.  140.
#>  5  971.  243.  693.  131. 1699.  143.
#>  6 1214.  243.  742.  134. 1456.  145.
#>  7 1456.  243.  791.  136. 1214.  147.
#>  8 1699.  243.  841.  139.  971.  148.
#>  9 1942.  243.  891.  142.  728.  149.
#> 10 2185.  243.  941.  145.  485.  150.
#> 11 2427.  243.  992.  147.  243.  150.
#> 12 2670   243. 1043.  150.    0   150
# we don't want all parameters spelled out      ^   ^      ^     ^      ^      ^
```

APPENDIX

previous data (unit testing)

```
p30 = 1043.8745
p30 = 1043.8793
p30 = 1045.1834
p30 = 1043.1091 (after using p.avg and t.avg)
p30 = 1043.1094 (after using p.avg, t.avg, p0 = p.calc)
```

where the HDF5 file is

link to hdf5 file

hdf5 in inst/extdata

h5

results for unit testing

```
  MD      TVD     Pres  Temp
   0        0    500.0  135.8
 242.7   242.7   563.1  137.9
 485.5   485.5   627.5  139.8
 728.2   728.2   693.1  141.6
 970.9   970.9   759.8  143.4
1213.6  1213.6   827.6  144.9
1456.4  1456.4   896.5  146.3
1699.1  1699.1   966.4  147.6
1941.8  1941.8  1037.3  148.6
2184.5  2184.5  1109.3  149.3
2427.3  2427.3  1182.2  149.8
2670.0  2670.0  1255.9  150.0
```

```
# convert text table to dataframe in preparation for unit test
results_unit_test <- "
      MD      TVD     Pres  Temp
       0        0    500.0  135.8
     242.7   242.7   563.1  137.9
     485.5   485.5   627.5  139.8
     728.2   728.2   693.1  141.6
     970.9   970.9   759.8  143.4
    1213.6  1213.6   827.6  144.9
    1456.4  1456.4   896.5  146.3
    1699.1  1699.1   966.4  147.6
    1941.8  1941.8  1037.3  148.6
    2184.5  2184.5  1109.3  149.3
    2427.3  2427.3  1182.2  149.8
    2670.0  2670.0  1255.9  150.0
"

library(readr)
read_table2(results_unit_test, col_names = TRUE)
#> # A tibble: 12 x 4
#>       MD   TVD  Pres  Temp
#>    <dbl> <dbl> <dbl> <dbl>
#>  1    0     0   500   136.
#>  2  243.  243.  563.  138.
#>  3  486.  486.  628.  140.
#>  4  728.  728.  693.  142.
#>  5  971.  971.  760.  143.
#>  6 1214. 1214.  828.  145.
#>  7 1456. 1456.  896.  146.
#>  8 1699. 1699.  966.  148.
#>  9 1942. 1942. 1037.  149.
#> 10 2184. 2184. 1109.  149.
#> 11 2427. 2427. 1182.  150.
#> 12 2670  2670  1256.  150
```

```
library(rNodal)

# separate the columns with tabs
well_as_text <- "
    MD  TVD
       0         0
     242.7   242.7
     485.5   485.5
     728.2   728.2
     970.9   970.9
    1213.6  1213.6
    1456.4  1456.4
    1699.1  1699.1
    1941.8  1941.8
    2184.5  2184.5
    2427.3  2427.3
    2670.0  2670.0
"

deviation_survey <- set_deviation_survey(well_as_text)
deviation_survey
#>        MD    TVD
#> 1     0.0    0.0
#> 2   242.7  242.7
#> 3   485.5  485.5
#> 4   728.2  728.2
#> 5   970.9  970.9
#> 6  1213.6 1213.6
#> 7  1456.4 1456.4
#> 8  1699.1 1699.1
#> 9  1941.8 1941.8
#> 10 2184.5 2184.5
#> 11 2427.3 2427.3
#> 12 2670.0 2670.0
```

```
# rNodal:::calc_angle_deviation_survey(deviation_survey)
```