design.R

sam

2021-03-13

```r
# Propeller design
# Script created by Sam Hince
# 02/04/2021

rm(list = ls())

library(caTools)
library(tictoc)
library(rjson)
library(ggplot2)

##########################################################################
tic()

setwd("/home/sam/Documents/classGitRepos/MAE195")

#givens
name <- "Cessna_150_design"
D <- 5.75 #ft
D_hud <- 1 #ft
B <- 2 #number of blades
V <- 110 #mph
power <- 70 #BHp
# thrust <- 0
RPM <- 2400 #3200 # 1250
rho <- 0.001267 # sea level ft
kinetic_viscosity <- 0.002377
gamma <- 1.4
gas_const <- 1718
temp <- 464.514 # atmospheric temperature (currently: standard atm at 20k ft)
airfoil <- "./propSpecs/NACA4415_RN500K_NCRIT9.csv" # airfoil data

# settings
steps <- 21 # number of blade stations
Cl <- rep(0.7, steps) # desired Cl at each blade station
sucessThreshold <- 0.001 # criteria for convergence

##########################################################################
# conversions
Omega <- RPM * (1/60) * (2*pi)
R <- D/2
```

```
mu <- kinetic_viscosity * rho # N s/m^2 dynamic viscosity of the fluid
stations <- seq(from = (D_hud/2), to = R, length.out = steps)

V <- V * 1.467 # convert to ft/s
power <- power * 550 # convert to ft*lb/s

##########################################################################
# data read
coef <- read.csv(file = airfoil, header = TRUE)

##########################################################################

#step 1 # starting value for zeta
zeta <- 0

### loop start ###
while(TRUE){
  #step 2 # determine F and phi at each station
  df <- data.frame(Xi=numeric(), f=numeric(), varF=numeric(), phit=numeric(), phi=numeric())

  for(r in stations){
    Xi <- r/R

    phit <- atan((V/(Omega*R)) * (1 + (zeta/2))) # atan or atan2?

    f <- (B/2)*((1-(Xi)) / (sin(phit))) # ((sin(phit))^2))
    varF <- (2/pi) * atan((exp(2*f)-1)^(1/2)) # Ideal version: varF <- (2/pi) * acos(exp(-f))

    # useing r*tan(phi) = R*tan(phit) = const
    phi <- atan2(tan(phit),Xi) # phi <- atan2(R*tan(phit),r)

    # save results
    df <- rbind(df, data.frame(Xi, f, varF, phit, phi))
  }

  #step 3
  X <- (Omega * stations)/V # (Omega * r)/V
  G <- df$varF*X*sin(df$phi)*(cos(df$phi))

  Wc <- (4*pi*(V^2)*G*zeta)/(B*Omega*Cl)
  Rn <- (rho * Wc) / mu

  #step 4
  alpha <- approx(x = coef$CL, y = coef$ALPHA, xout = Cl, method="linear", ties=min)$y
  Cd <- approx(x = coef$CL, y = coef$CD, xout = Cl, method="linear", ties=min)$y

  alpha <- alpha * (pi/180) # convert to radians
  epsilon <- Cd/Cl # viscous effects

  #step 5
  a <- (zeta / 2) * (cos(df$phi)^2) * (1 - (epsilon*tan(df$phi)))
  aprime <- (zeta / (2*X)) * cos(df$phi) * sin(df$phi) * (1 + (epsilon / tan(df$phi)))
```

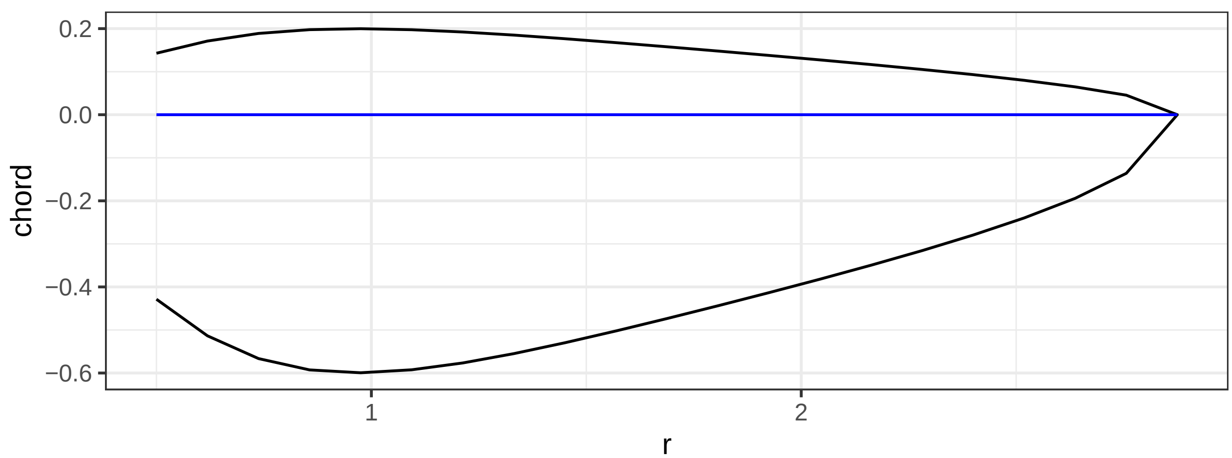
```r
  #step 6
  W <- (V * (1+a))/sin(df$phi)

  #step 7
  c <- Wc/W
  beta <- alpha + df$phi

  #step 8
  lambda <- V/(Omega*R)
  I1_integrand <- 4*df$Xi*G*(1-(epsilon*tan(df$phi)))
  I2_integrand <- lambda*(I1_integrand/(2*df$Xi))*(1+(epsilon/tan(df$phi)))*sin(df$phi)*cos(df$phi)
  J1_integrand <- 4*df$Xi*G*(1+(epsilon/tan(df$phi)))
  J2_integrand <- (J1_integrand/2)*(1-(epsilon*tan(df$phi)))*(cos(df$phi)^2)

  #integrate
  I1 <- trapz(df$Xi, I1_integrand)
  I2 <- trapz(df$Xi, I2_integrand)
  J1 <- trapz(df$Xi, J1_integrand)
  J2 <- trapz(df$Xi, J2_integrand)

  # step 9
  if(exists("thrust")){
    # use thrust definition
    Tc <- (2*thrust)/(rho*(V^2)*pi*(R^2))
    zeta_new <- ((I1/2*I2)) - (((I1/(2*I2))^2)-(Tc/I2))^(1/2)
    Pc <- (zeta*J1) + ((zeta^2)*J2)

  }else{
    # use power definition
    Pc <- (2*power)/(rho*(V^3)*pi*(R^2))
    zeta_new <- (-1 * (J1/(2*J2))) + (((J1/(2*J2))^2)+(Pc/J2))^(1/2)
    Tc <- (zeta*I1) - ((zeta^2)*I2) # (4*zeta*I1) - (2*(zeta^2)*I2)

  }

  # step 10 - is zeta is no within 0.1% start back at step 2
  zeta_change <- abs((zeta_new - zeta) / zeta)
  zeta <- zeta_new
  print(zeta)
  if(zeta_change < sucessThreshold){
    break
  }

} # end of loop
```

```
## [1] 0.3779419
## [1] 0.357586
## [1] 0.3584427
## [1] 0.3584062
```

```r
# step 11
if(exists("thrust")){
  power <- Pc * (rho*(V^3)*pi*(R^2)) * (1/2)
```

```
}else{
  thrust <- Tc * (rho*(V^2)*pi*(R^2)) * (1/2)
}

n <- RPM / 60
advance_ratio <- V / (n * D)
efficiency <- Tc / Pc

#sigma <- (B * c) / (2 * pi * stations) # local solidity
#solidity <- trapz(df$Xi, sigma)
solidity <- (B * trapz(stations, c)) / (pi * (R^2))

AF <- (100000/16)*trapz(df$Xi, ((c/D)*(df$Xi^3)))
mach <- (Omega * stations) / (sqrt(gamma * gas_const * temp))

# step 12
# output geom
ls <- list(propName = name,
           diameter = D,
           hubDiameter = D_hud,
           blades = B,
           airfoil = airfoil,
           Cl = mean(Cl),
           velocity = V, # leave in ft/s
           J = advance_ratio,
           RPM = RPM,
           power = power / 550, # convert to HP
           thrust = thrust,
           Cp = "",
           Ct = "",
           solidity = solidity,
           AF = AF,
           alt = list(density = rho,
                      kinematicViscosity = kinetic_viscosity,
                      speedofsound = sqrt(gamma * gas_const * temp)),
           radialStation = stations,
           chord = c,
           beta = beta * (180/pi))

exportJson <- toJSON(ls)
write(exportJson, file = './propSpecs/DesignOutput.json')

##########################################################################
# plotting code

prop_geom_le <- data.frame(chord = (c * (1/4)), r = stations)
prop_geom_te <- data.frame(chord = (c * (-3/4)), r = stations)
prop_geom_te <- prop_geom_te[seq(dim(prop_geom_te)[1],1),]

prop_geom <- rbind(prop_geom_le, prop_geom_te)

geom_plot <- ggplot(prop_geom, aes(x = r, y = chord)) + geom_path() + coord_fixed() + theme_bw() +
  geom_line(data = data.frame(chord = rep(0, length(stations)), r = stations), colour = "blue")
```

```r
print(geom_plot)
```




```r
################################################################################

toc()
```

```
## 0.303 sec elapsed
```