Universidade Federal de Uberlândia
FEELT – Faculdade de Engenharia Elétrica

SISTEMAS DE CONTROLE

Trabalho Final: Modelo/Simulação
Professor: Dr. Éder Alves de Moura

Gabriel Cardoso Mendes de Ataide	11811ECP008
Guilherme Gabriel Ferreira Souza	11921ECP001
Renzo Prats Silva Souza	11921ECP004
Rodrigo Henrique Alves Ferreira	11811ECP001

Uberlândia
Dezembro 2023

SUMÁRIO

Introdução

O uso de drones na atualidade tem crescido muito. Hoje, sua aplicação varia da simples recreação até o uso em aplicações comerciais, industriais e militares. A Figura 1 apresenta um exemplo de um quadricóptero, uma das versões de veículos multirotores mais populares.

Figura 1 - Drone DJI Mavic 2

Este roteiro apresenta a modelagem matemática do bicóptero, simulando o movimento de um drone em um plano 2D. Esta simplificação visa simular um sistema de controle embarcado que controlará o movimento, simplificado, de um drone. Esse movimento consiste de sua posição e atitude (orientação).

Código reescrito em R

Python

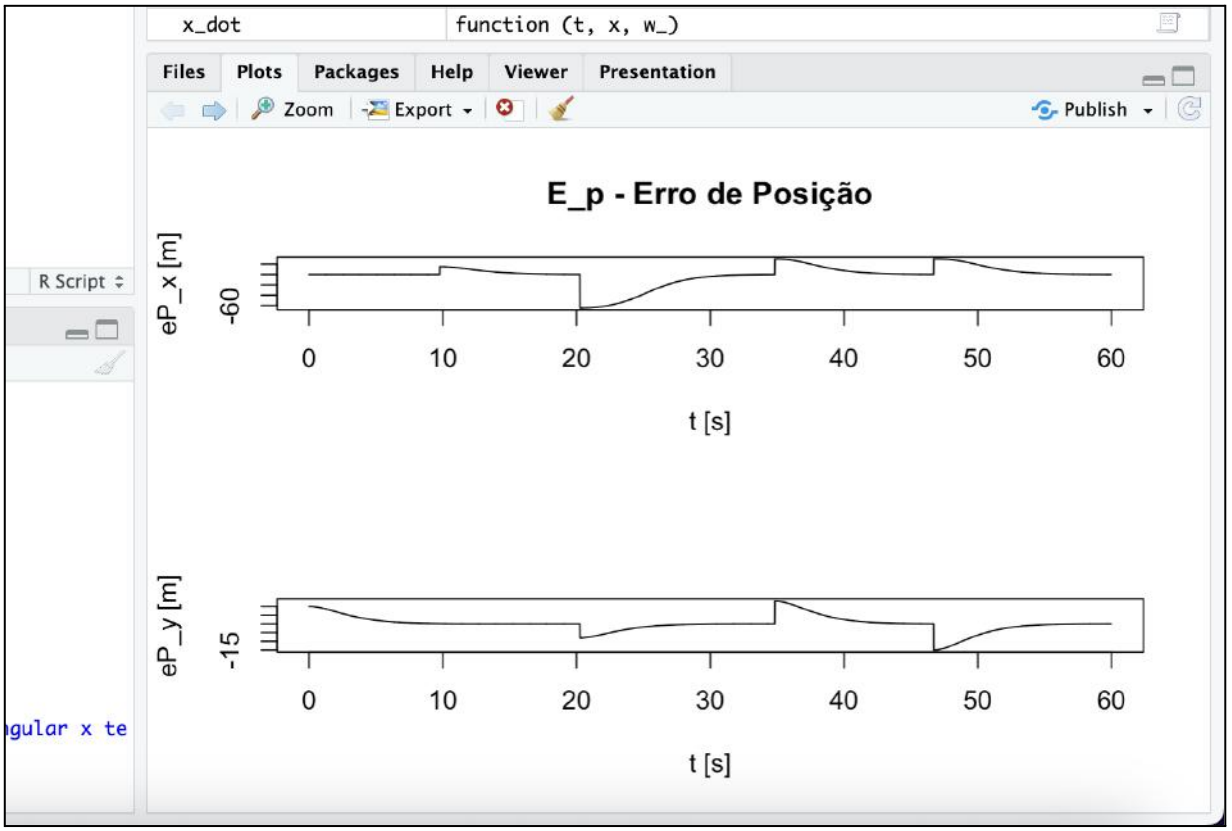
```
# Trabalho Final Sistemas De Controle
# Alunos: Gabriel Cardoso, Guilherme Gabriel, Renzo Prats, Rodrigo Henrique

library(matrixStats)

# Funcao para calcular x_dot
x_dot <- function(t, x, w_) {
 # Parametros
 w_max <- 15000 # velocidade máxima do motor
 m <- 0.25 # massa
 g <- 9.81 # aceleração da gravidade
 l <- 0.1 # tamanho
 kf <- 1.744e-08 # constante de força
 Iz <- 2e-4 # momento de inércia
 tal <- 0.005
 P <- matrix(c(0, -m * g), ncol=1)

 # Estados atuais
 w <- x[1:2]
 r <- x[3:4]
 v <- x[5:6]
 phi <- x[7]
 ome <- x[8]

 # Variáveis auxiliares

 # Forcas
 f1 <- kf * w[1]^2
 f2 <- kf * w[2]^2

 # Torque
 Tc <- l * (f1 - f2)

 # Forca de controle
 Fc_B <- matrix(c(0, (f1 + f2)), ncol = 1)

 # Matriz de atitude
 D_RB <- matrix(c(cos(phi), -sin(phi), sin(phi), cos(phi)), ncol = 2, byrow = TRUE)

 # Derivadas
 w_dot <- (-w + w_) / tal
 r_dot <- v
 v_dot <- as.vector((1 / m) * (D_RB %*% Fc_B + P))
 phi_dot <- c(ome)
 ome_dot <- c(Tc / Iz)

 xkp1 <- c(w_dot, r_dot, v_dot, phi_dot, ome_dot)
 return(xkp1)
}
```

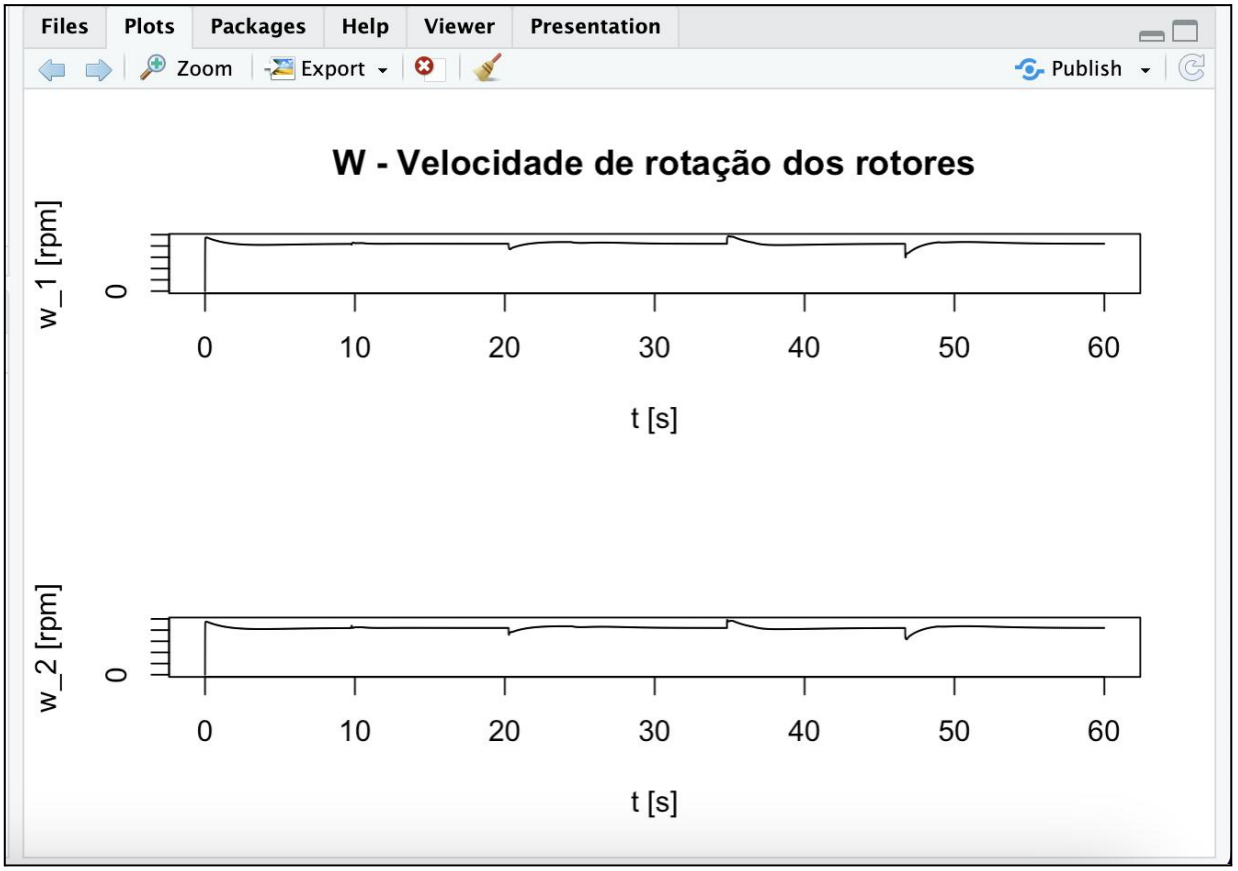
```
# Runge Kutta 4 ordem
rk4 <- function(tk, h, xk, uk) {
 k1 <- x_dot(tk, xk, uk)
 k2 <- x_dot(tk + (h/2.0), xk + (h * (k1/2.0)), uk)
 k3 <- x_dot(tk + (h/2.0), xk + (h * (k2/2.0)), uk)
 k4 <- x_dot(tk + h, xk + (h * k3), uk)

 xkp1 <- xk + (h/6.0) * (k1 + (2 * k2) + (2 * k3) + k4)
 return(xkp1)
}

# Parametros de simulacao
h <- 1e-3 # passo da simulacao de tempo continuo
Ts <- 10e-3 # intervalo de atuação do controlador
fTh <- Ts/h
maxT <- 60
tc <- seq(0, maxT - h, by = h) # k
td <- seq(0, maxT - Ts, by = Ts) # j
tam <- length(tc)
j <- 1

# Vetor de estados
x <- matrix(0, nrow = 8, ncol = tam)
x[, 1] <- c(0, 0, 0, 0, 0, 0, 0*pi/180, 0*pi/180)

# Parametros do sistema de controle
# Vetor de controle relativo a rotacao
w_ <- matrix(0, nrow = 2, ncol = length(td)) # comando de controle
Phi_ <- matrix(0, nrow = 1, ncol = length(td)) # comando de Atitude
WP_ <- matrix(0, nrow = 2, ncol = length(td)) # comando de Waypoint

# Vetor dos erros de posição
eP_ <- matrix(0, nrow = 2, ncol = length(td))
eV_ <- matrix(0, nrow = 2, ncol = length(td))
ePhi_ <- matrix(0, nrow = 1, ncol = length(td))
eOme_ <- matrix(0, nrow = 1, ncol = length(td))

ePm1 <- 1 # erro posição k-1 (passo anterior)
eVm1 <- 1 # erro atitude k-1 (passo anterior)

# Constanstes do modelo
m <- 0.25 # massa
g <- 9.81 # aceleracao da gravidade
l <- 0.1 # tamanho
kf <- 1.744e-08 # constante de forca
Iz <- 2e-4 # momento de inércia
tal <- 0.05
Fe <- c(-m * g)

# Restrições do controle
phi_max <- (15 * pi) / 180

w_max <- 15000
```

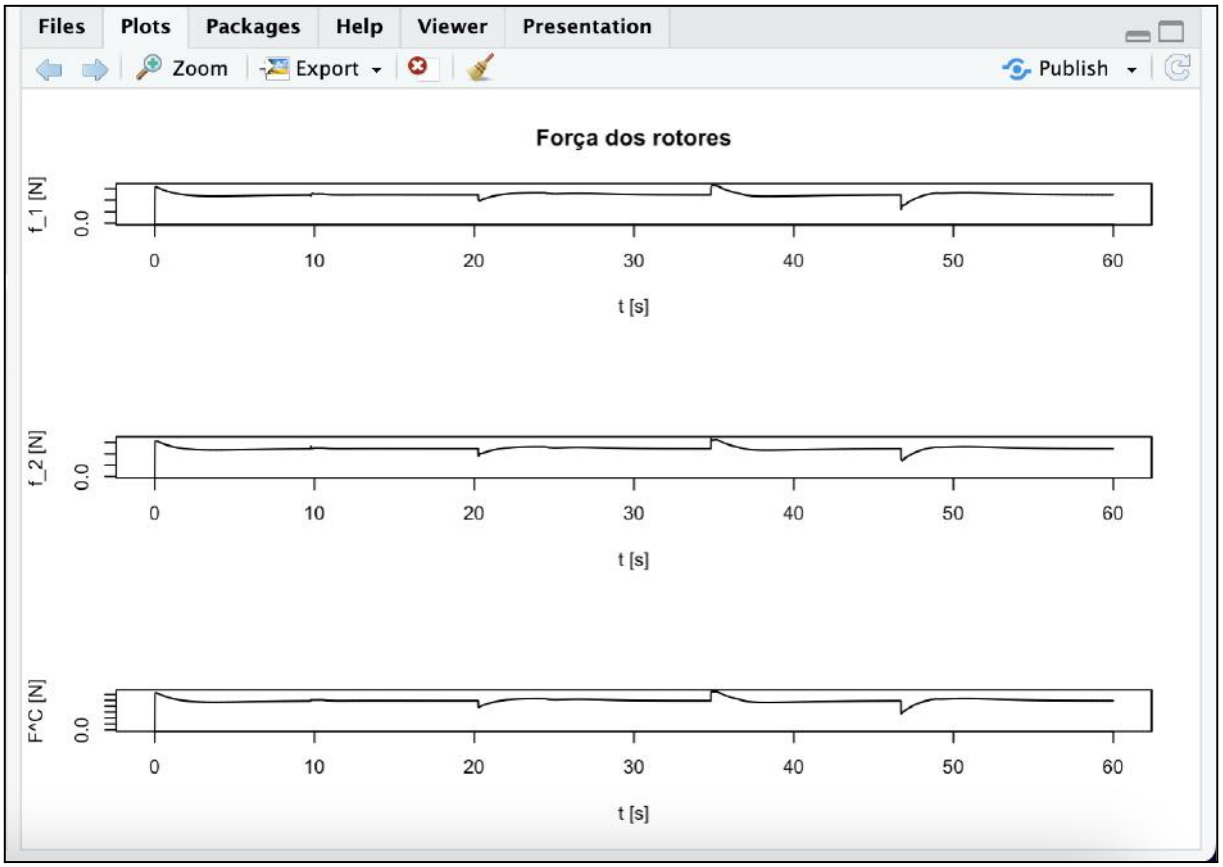
```
Fc_max <- kf * w_max^2 # Força de controle máximo
Tc_max <- l * kf * w_max^2

# Waypoints
r_ <- matrix(c(0, 15, -50, -20, 10, 10, 10, 2, 15, 0), nrow = 2, byrow = TRUE)
r_ID <- 1
r_IDN <- 5

# Simulacao
for (k in 1:(tam-1)) {
 # Sistema de controle
 if (((k-1) %% fTh) == 0) {
  # Extrai os dados do  vetor
  r_k <- x[3:4, k]
  v_k <- x[5:6, k]
  phi_k <- x[7, k]
  ome_k <- x[8, k]

  # Comando de posicao
  v_ <- c(0, 0)

  # Controle de Posicao
  kpP <- c(0.075)
  kdP <- c(0.25)
  eP <- r_[, r_ID] - r_k
  eV <- v_ - v_k
  WP_[, j] <- r_[, r_ID]
  eP_[, j] <- eP
  eV_[, j] <- eV

  eP <- as.matrix(eP)
  # Proximo waypoint
  if (norm(eP) < .1 && r_ID < r_IDN) {
   r_ID <- r_ID + 1
   cat("Bucar Waypoint: ", r_ID, "\n")
  }

  Fx <- kpP * eP[1] + kdP * eV[1]
  Fy <- kpP * eP[2] + kdP * eV[2] - Fe
  Fy <- pmax(0.2 * Fc_max, pmin(Fy, 0.8 * Fc_max))

  # Controle de Atitude
  phi_ <- atan2(-Fx, Fy)

  if (abs(phi_) > phi_max) {
   signal <- phi_ / abs(phi_)
   phi_ <- signal * phi_max

   # Limitando o angulo
   Fx <- Fy * tan(phi_)
  }

  Phi_[, j] <- phi_
```

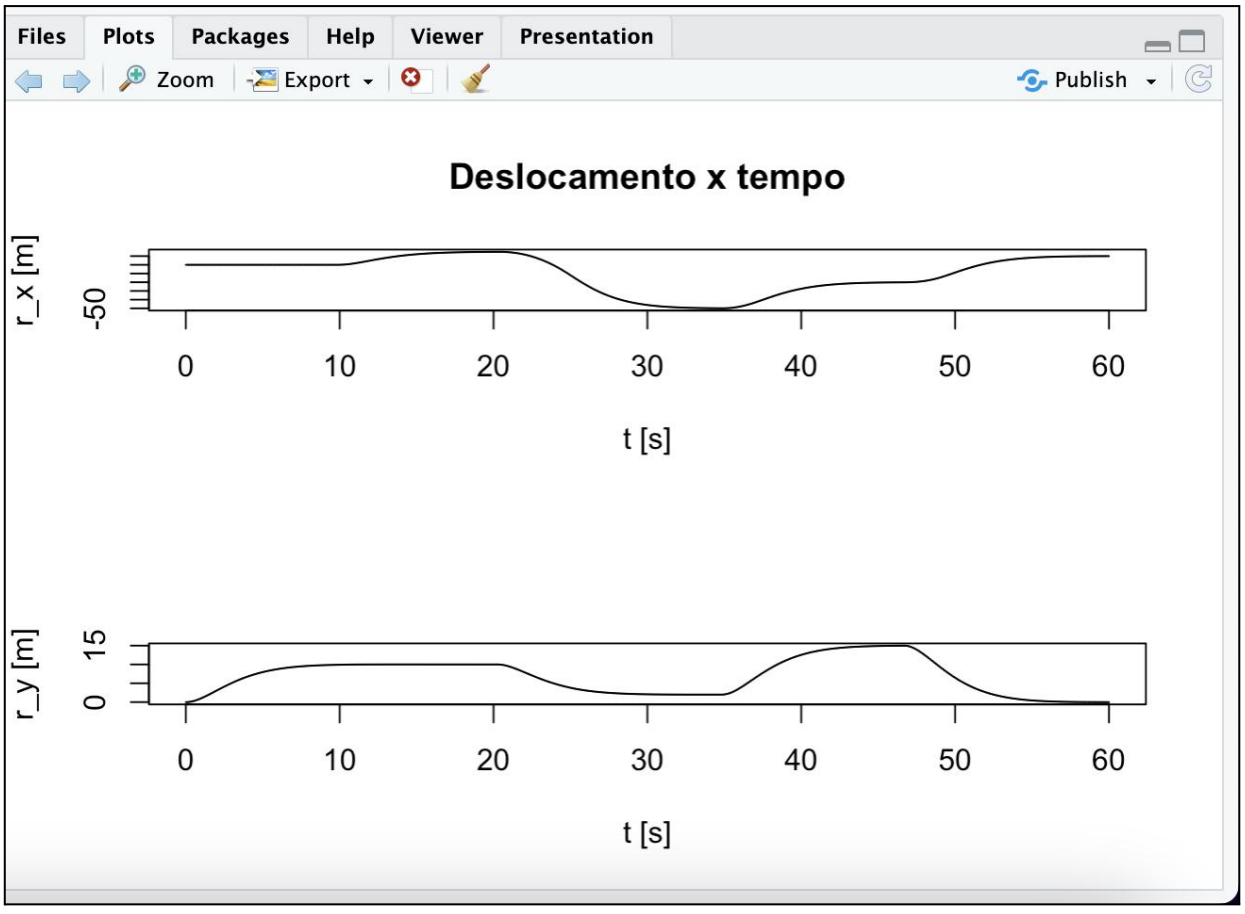
```
  Fxy <- c(Fx, Fy)
  Fc <- sqrt(sum(Fxy^2))
  f12 <- c(Fc/2.0, Fc/2.0)

  # Constantes Kp e Kd
  kpA <- c(0.75)
  kdA <- c(0.05)
  ePhi <- phi_ - phi_k
  eOme <- 0 - ome_k

  ePhi_[, j] <- ePhi
  eOme_[, j] <- eOme

  Tc <- kpA * ePhi + kdA * eOme
  Tc <- pmax(-0.4 * Tc_max, pmin(Tc, 0.4 * Tc_max))

  # Delta de forcas
  df12 <- abs(Tc) / 2.0

  # Forças f1 e f2 final f12' = f12 + deltf12
  if (Tc >= 0.0) {
   f12[1] <- f12[1] + df12
   f12[2] <- f12[2] - df12
  } else {
   f12[1] <- f12[1] - df12
   f12[2] <- f12[2] + df12
  }

  # Comando de rpm dos motores
  w1_ <- sqrt(f12[1] / kf)
  w2_ <- sqrt(f12[2] / kf)

  # Limitando o comando do motor entre 0 - 15000 rpm
  w1 <- pmax(0, pmin(w1_, w_max))
  w2 <- pmax(0, pmin(w2_, w_max))

  # Determinação do comando de entrada
  w_[, j] <- c(w1, w2)
  # Posição do sistema de tempo discreto
  j <- j + 1
 }
 # Simulacao um passo a frente
 x[, k + 1] <- rk4(tc[k], h, x[, k], w_[, j - 1])
}


# Processamento de variáveis intermediárias

# obtem a força aplicada por cada rotor
f <- matrix(0, nrow = 3, ncol = tam)

for (k in 1:tam) {
 w <- x[1:2, k]
 f[1:2, k] <- c(kf * w[1]^2, kf * w[2]^2)
```

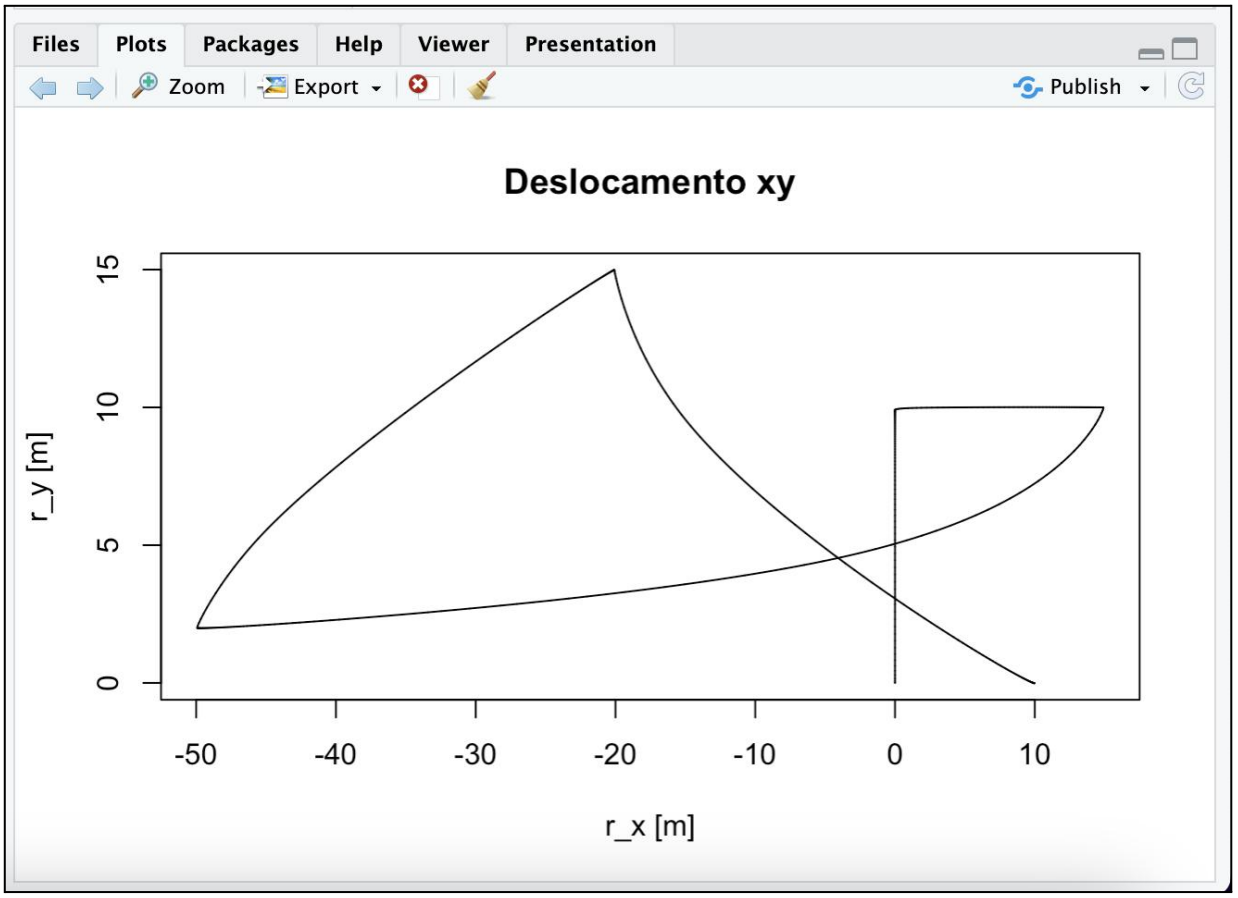
```
 f[3, k] <- f[1, k] + f[2, k] # Fc total em B
}

# Define VecXf
VecXf <- t(rbind(tc, x, f))

# Define VecControl
VecControl <- t(rbind(td, WP_, w_, eP_, eV_, ePhi_, eOme_))

# Plotando
par(mfrow=c(2,1))
plot(td, eP_[1,], type='l', ylab='eP_x [m]', xlab='t [s]', main='E_p - Erro de Posição')
plot(td, eP_[2,], type='l', ylab='eP_y [m]', xlab='t [s]')
par(mfrow=c(1,1))

par(mfrow=c(2,1))
plot(tc, x[1,], type='l', ylab='w_1 [rpm]', xlab='t [s]', main='W - Velocidade de rotação
dos rotores')
plot(tc, x[2,], type='l', ylab='w_2 [rpm]', xlab='t [s]')
par(mfrow=c(1,1))

par(mfrow = c(3, 1))
plot(tc[1:length(f[1,])], f[1,], type = 'l', ylab = 'f_1 [N]', xlab = 't [s]', main = 'Força
dos rotores')
plot(tc[1:length(f[2,])], f[2,], type = 'l', ylab = 'f_2 [N]', xlab = 't [s]')
plot(tc[1:length(f[3,])], f[3,], type = 'l', ylab = 'F^C [N]', xlab = 't [s]')
par(mfrow = c(1, 1))

par(mfrow=c(2,1))
plot(tc, x[3,], type='l', ylab='r_x [m]', xlab='t [s]', main='Deslocamento x tempo')
plot(tc, x[4,], type='l', ylab='r_y [m]', xlab='t [s]')
par(mfrow=c(1,1))

plot(x[3,], x[4,], type='l', ylab='r_y [m]', xlab='r_x [m]', main='Deslocamento xy')

par(mfrow=c(2,1))
plot(tc, x[5,], type='l', ylab='v_x [m/s]', xlab='t [s]', main='Velocidade x tempo')
plot(tc, x[6,], type='l', ylab='v_y [m/s]', xlab='t [s]')
par(mfrow=c(1,1))

plot(tc, x[7,] * 180 / pi, ylab='Attitude [deg]', xlab='t [s]', main='Attitude x tempo')

plot(tc, x[8,] * 180 / pi, ylab='Angular Velocity [deg/s]', xlab='t [s]', main='Velocidade
angular x tempo')
```

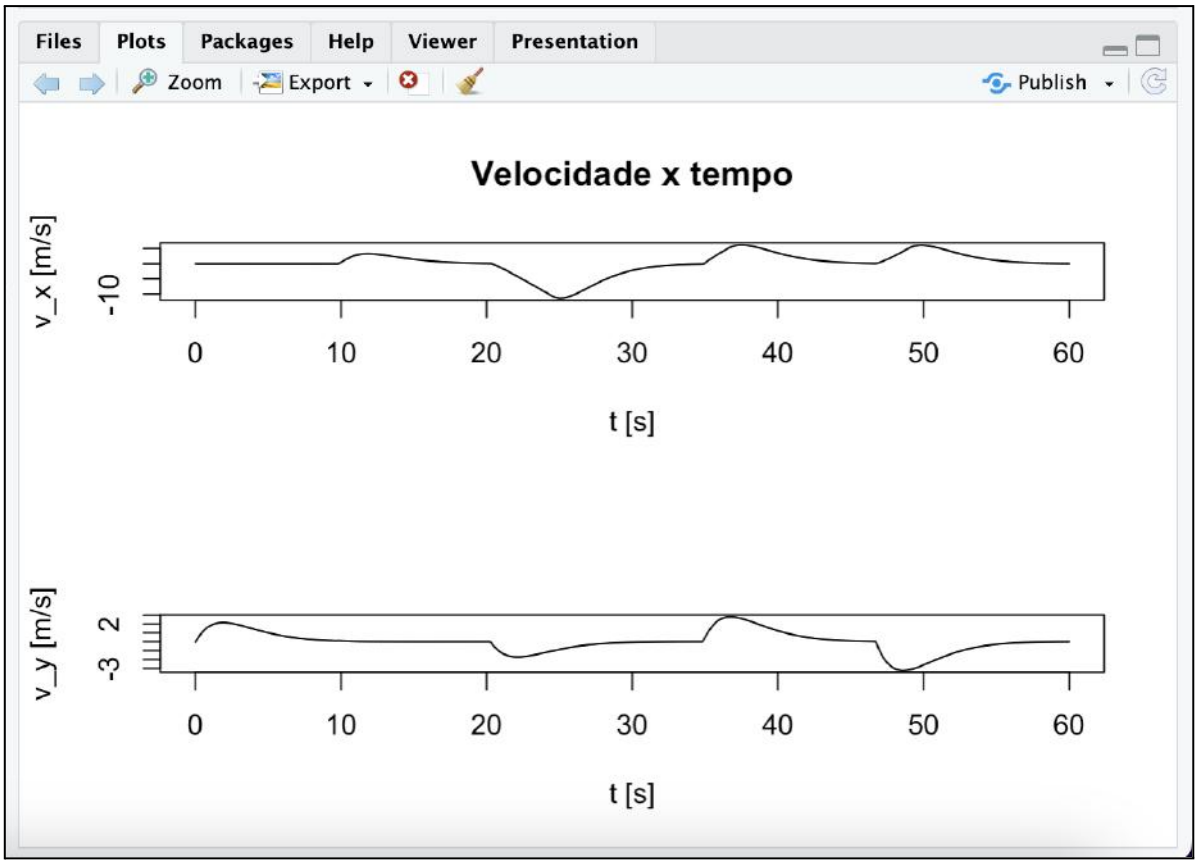

x_dot
function (t, x, w_)
Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
R Script
E_p - Erro de Posição
eP_x [m]
-60
0
10
20
30
40
50
60
t [s]
eP_y [m]
-15
0
10
20
30
40
50
60
t [s]

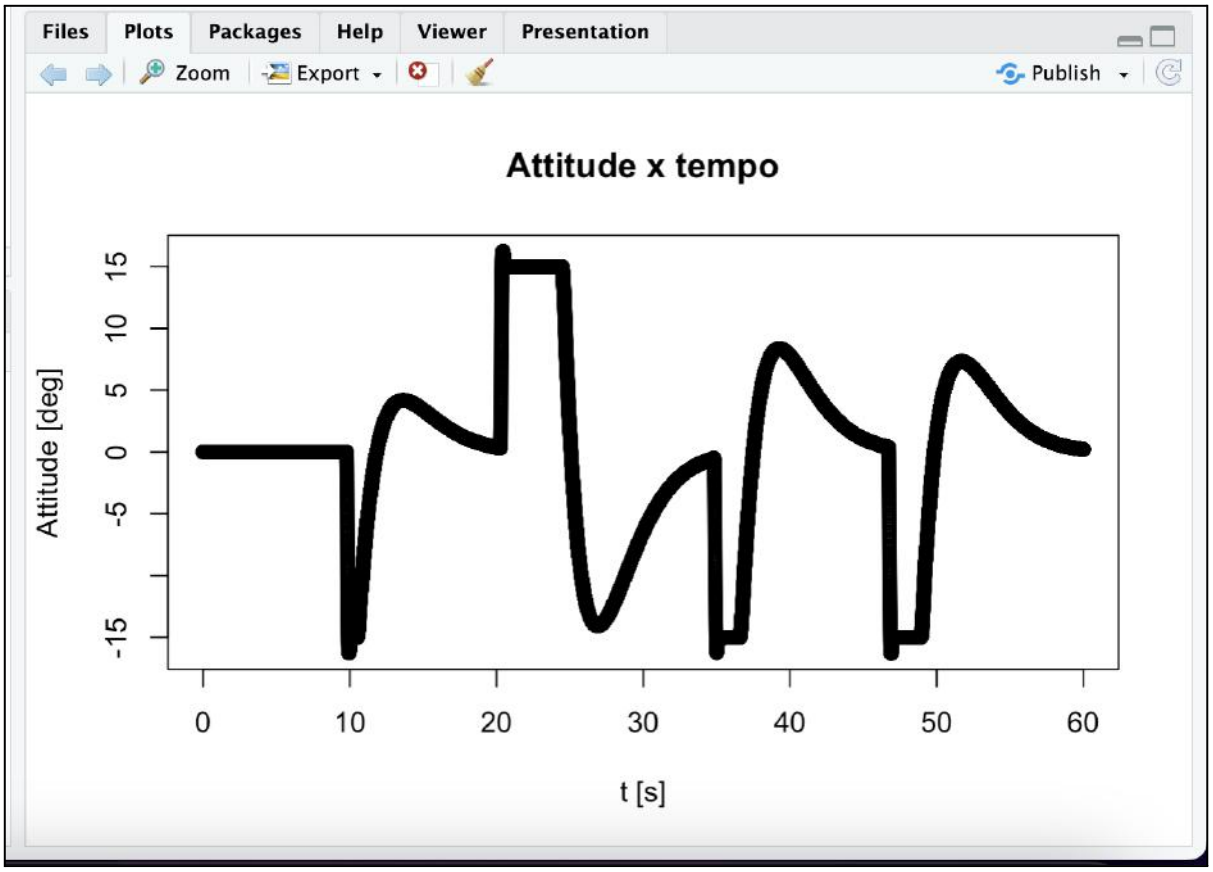

Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
W - Velocidade de rotação dos rotores
w_1 [rpm]
0
0
10
20
30
40
50
60
t [s]
w_2 [rpm]
0
0
10
20
30
40
50
60
t [s]

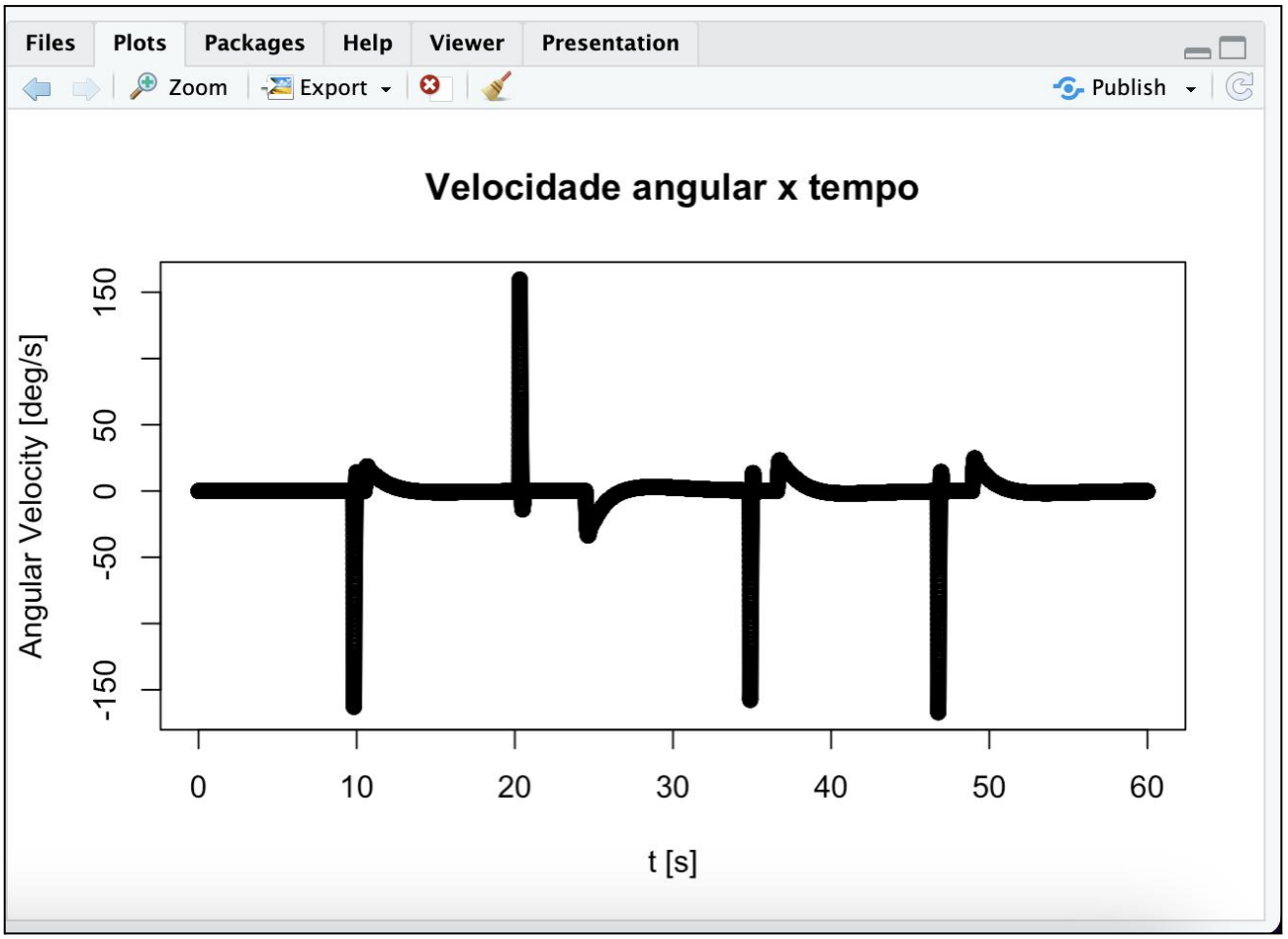

Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Força dos rotores
f_1 [N]
f_2 [N]
F^C [N]
0.0
0
10
20
30
40
50
60
t [s]


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Deslocamento x tempo
r_x [m]
-50
r_y [m]
0
15
0
10
20
30
40
50
60
t [s]


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Deslocamento xy
r_y [m]
0
5
10
15
-50
-40
-30
-20
-10
0
10
r_x [m]


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Velocidade x tempo
v_x [m/s]
-10
0
10
20
30
40
50
60
t [s]
v_y [m/s]
-3
2
0
10
20
30
40
50
60
t [s]


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Attitude x tempo
Attitude [deg]
15
10
5
0
-5
-15
0
10
20
30
40
50
60
t [s]


Files
Plots
Packages
Help
Viewer
Presentation
Zoom
Export
Publish
Velocidade angular x tempo
Angular Velocity [deg/s]
150
50
0
-50
-150
0
10
20
30
40
50
60
t [s]