UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA BIOMÉDICA

Heitor Pereira Nunes Fernandes Cunha

Processamento de Sinais Biomédicos: Módulo 3

Uberlândia, MG
2025

Questão 1

O eletrocardiograma (ECG) é formado por unidades básicas do complexo PQRST (Figura 2). A duração típica do complexo PQRST é de aproximadamente 200 ms e a amplitude máxima é de aproximadamente 1 mV. Dolinský et al. (2018) descrevem uma forma relativamente simples de modelar o complexo PQRST, que pode ser realizada por meio da concatenação de modelos básicos de cada uma de suas partes. O início do complexo PQRST é dado por um potencial isoelétrico, definido na Equação 1. A onda P pode ser modelada de acordo com a Equação 2. A onda P é seguida por um potencial isoelétrico (Equação 3). A onda Q é dada pela Equação 4. A onde R é definada pela Equação 5. A onda S é modelada de acordo com a Equação 6. A transição entre as ondas S e T, denominada St, é dada pela Equação 7. A onda T é modelada conforme a Equação 8. A transição final da onda T para o nível do potencial isoelétrico é dado pela Equação 9.

Dolinský et al. (2018) definiram um conjunto de parâmetros (m) que simula o complexo PQRST em função de diferentes desordens que podem acometer o sistema cardiovascular. Esse conjunto de parâmetros é apresentado na Tabela 1.

```r
# Equacao 1
B <- function(KB){
  rep(0, KB)
}

# Equacao 2
P <- function(KP, AP) {
  k <- 0:KP
  P <- -(AP / 2) * cos((2 * pi * k + 15) / KP)
  return(P)
}

# Equacao 3
PQ <- function(KPQ){
  rep(0, KPQ)
}

# Equacao 4
Q <- function(AQ, KQ) {
  k <- 0:KQ
  Q <- (AQ * (k - 0.1 * KQ + 0.1) * (19.78 * pi / KQ)) * exp(-2 * ((6 * pi / KQ) * (k - 0.1 * KQ + 0.1))

  Q <- ifelse(Q < 0, Q, NA)
  return(Q[is.na(Q) == FALSE])
}

# Equacao 5
R <- function(AR,KR){
  k <- 0:KR
  R <- AR * sin((pi * k) / KR)
  return(R)
}

# Equacao 6
S <- function(AS, KS, KCS){
  k <- 0:(KS - KCS)
```

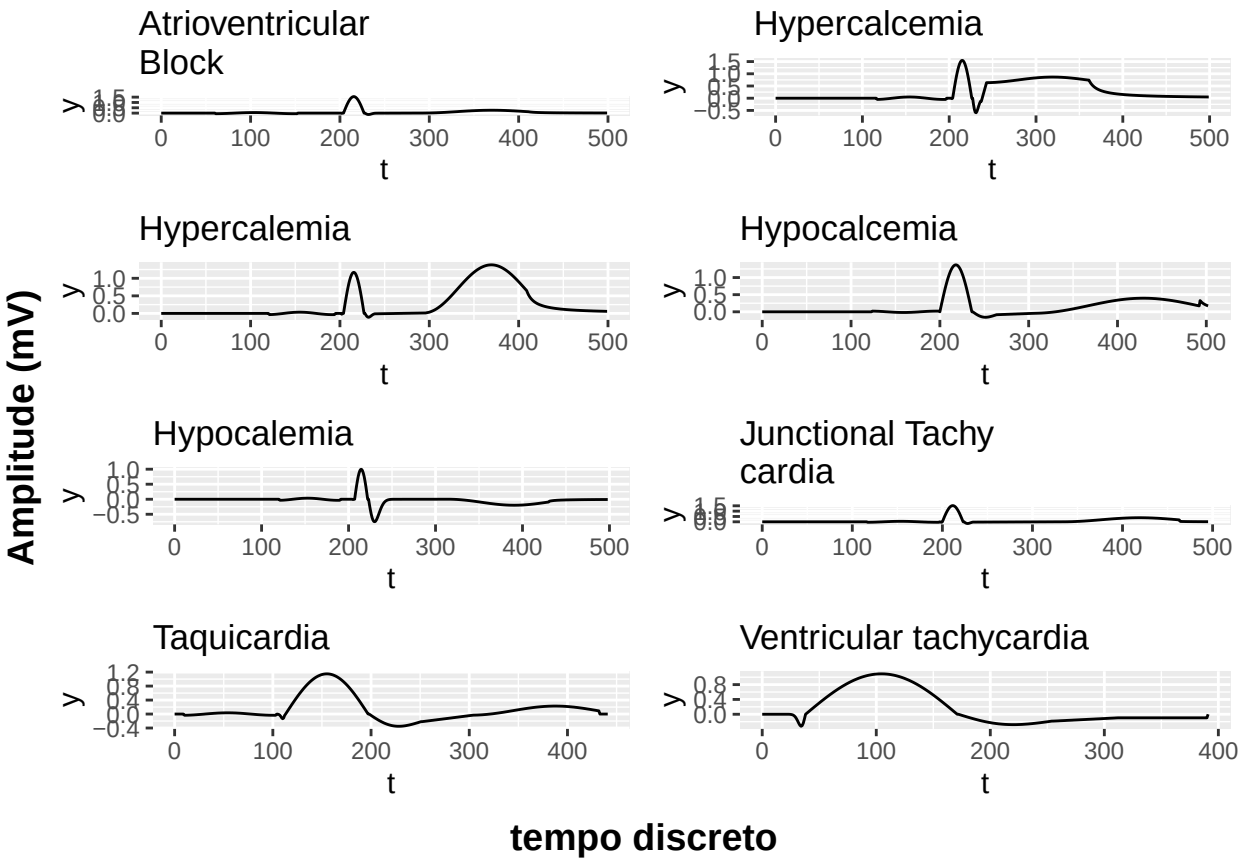
```
  S <- - AS * 0.1 * k *((19.78 * pi) / KS) * exp(-2 * (((6 * pi * 0.1 * k) / KS)^2))
  return(S)
}

# Equacao 7
ST <- function(AS, KS, KCS, KST, sm){
  k <- 0:KST
  S_KS_KCS <- S(AS, KS, KCS)
  S_s <- S_KS_KCS[length(S_KS_KCS)]
  ST <- S_s - S_s*(k/sm)
  return(ST)
}

# Equacao 8
T_ <- function(AT, KT, KST, AS, KS, KCS, sm){
  k <- 0:KT
  ST_KST <- ST(AS,KS,KCS,KST,sm)
  ST_s <- ST_KST[length(ST_KST)]
  T_ <- -AT * cos((1.48*pi*k + 15) / KT) + AT + ST_s
  return(T_)
}

# Equacao 9
I <- function(AT,KT,KST,ST,AS,KS,KCS,sm,KI,SI){
  k <- 0:KI
  T_KT <- T_(AT,KT,KST,AS,KS,KCS,sm)
  T_t <- T_KT[length(T_KT)]
  I <- T_t * (SI / (k + 10))
  return(I)
}
```

Gerar um complexo PQRST para cada uma das condições m apresentadas na Tabela 1. Cada modelo descrito pelas equações acima gera um vetor, cuja dimensão e valores resultam dos parâmetros apresentados na Tabela 1. O complexo PQRST é obtido pela concatenação desses vetores. A concatenação de vetores no R é facilmente obtida por meio da função c(), por exemplo, dado v1 <- c(1,2,3,4) e v2 <- c(5,6,7,8), a concatenação de v1 com v2 é v3 <- c(v1,v2). O resultado deve ser apresentado na forma de uma gráfico com 4 linhas e duas colunas, conforme modelo (ver Figura 3).

```
# Criando a tabela em R

dadosB <- c("tachycardia", "ventricular tachycardia", "junctional tachycardia", "atrioventricular,
            block", "hyperkalemia", "hypocalemia", "hypercalcemia", "hypocalcemia")

# Criando vetores com os dados da tabela dada e no final criando um data frame no ambiente do R

kb <- c(10,0,117,61,121,121,117,124)
ap <- c(0.070,0.000,0.090,0.106,0.070,0.070,0.100,-0.040)
kp <- c(93,23,79,91,73,69,79,75)
kpq <- c(0,0,0,48,6,13,5,0)
aq <- c(0.135,0.325,0.065,0.040,0.040,0.020,0.030,0.000)
kq <- c(85,140,25,21,21,22,20,15)
ar <- c(1.15,1.09,1.52,1.55,1.17,1.00,1.55,1.37)
```

```r
kr <- c(84,133,23,23,23,15,22,36)
as <- c(0.35,0.28,0.16,0.13,0.11,0.75,0.60,0.16)
ks <- c(114,182,15,15,15,26,14,54)
kcs <- c(61,100,5,2,4,-3,5,27)
sm <- c(61,119,96,17,26,35,1,87)
kst <- c(52,57,101,52,56,64,6,42)
at <- c(0.130,0.000,0.190,0.132,0.685,-0.100,0.115,0.220)
kt = c(127,77,126,116,112,112,116,184)
sI = c(0,0,2,9,9,7,10,19)
ki = c(8,0,31,87,89,67,138,9)

condicoes_m <- data.frame(dadosB = dadosB, KB = kb, AP = ap, KP = kp, KPQ = kpq, AQ = aq, KQ = kq, AR =
ar, KR = kr, AS = as, KS = ks, KCS = kcs, sm = sm, KST = kst, AT = at, KT = kt, sI = sI, KI =
ki)

# Feita uma lista vazia e um vetor para contagem de um laço, são guardados os vetores de cada sinal na
lista <- list()
vector <- c(1,2,3,4,5,6,7,8)

# Laço para concatenar os vetores
for (i in vector) {
  tipo <- subset(condicoes_m, dadosB == condicoes_m$dadosB[i])

  B_sinal <- B(KB = tipo$KB)
  P_sinal <- P(AP = tipo$AP, KP = tipo$KP)
  PQ_sinal <- PQ(KPQ = tipo$KPQ)
  Q_sinal <- Q(AQ = tipo$AQ, KQ = tipo$KQ)
  R_sinal <- R(AR = tipo$AR, KR = tipo$KR)
  S_sinal <- S(AS = tipo$AS, KS = tipo$KS, KCS = tipo$KCS)
  ST_sinal <- ST(AS = tipo$AS, KS = tipo$KS, KCS = tipo$KCS, KST = tipo$KST, sm = tipo$sm)
  T_sinal <- T_(AT = tipo$AT, KT = tipo$KT, KST = tipo$KST, AS = tipo$AS, KS = tipo$KS, KCS = tipo$KCS,
  I_sinal <- I(AT = tipo$AT, KT = tipo$KT, KST = tipo$KST, AS = tipo$AS, KS = tipo$KS, KCS = tipo$KCS, s

  sinal_total <- c(B_sinal,P_sinal,PQ_sinal,Q_sinal,R_sinal,S_sinal,ST_sinal, T_sinal, I_sinal)

  lista[[i]] <- sinal_total
}

# no laço sao selecionados os valores das constantes com base na condição e elas são aplicadas nas funç

library(ggplot2)
library(gridExtra)
library(grid)
library(cowplot)

# Por fim cada sinal é plotado e no fim são linkados em um painel
ta <- seq(0,length(lista[[1]]) - 1)
gg1 <- data.frame(t = ta, y = lista[[1]] )
plot1 <- ggplot(data = gg1, aes(x = t, y = y)) + geom_line() + labs(title = "Taquicardia")
#+ scale_y_continuous(limits = c(-0.5, 1.5)) + theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))
tb <- seq(0,length(lista[[2]]) - 1)
gg2 <- data.frame(t = tb, y = lista[[2]] )
plot2 <- ggplot(data = gg2, aes(x = t, y = y)) + geom_line() + labs(title = "Ventricular tachycardia")
```

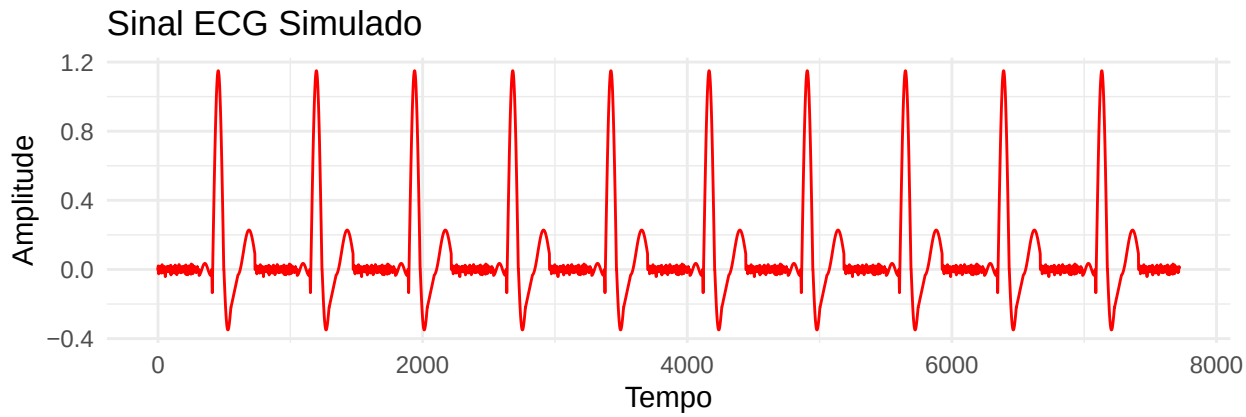
```
#+ theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))
tc <- seq(0,length(lista[[3]])-1)
gg3 <- data.frame(t = tc, y = lista[[3]] )
plot3 <- ggplot(data = gg3, aes(x = t, y = y)) + geom_line() + labs(title = "Junctional Tachy
cardia")
#+ theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))
td <- seq(0,length(lista[[4]]) - 1)
gg4 <- data.frame(t = td, y = lista[[4]] )
plot4 <- ggplot(data = gg4, aes(x = t, y = y)) + geom_line() + labs(title = "Atrioventricular
Block")
#+ theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))
te <- seq(0,length(lista[[5]]) - 1)
gg5 <- data.frame(t = te, y = lista[[5]] )
plot5 <- ggplot(data = gg5, aes(x = t, y = y)) + geom_line() + labs(title = "Hypercalemia")
#+ theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))
tf <- seq(0,length(lista[[6]]) - 1)
gg6 <- data.frame(t = tf, y = lista[[6]] )
plot6 <- ggplot(data = gg6, aes(x = t, y = y)) + geom_line() + labs(title = "Hypocalemia")
#+ theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))
tg <- seq(0, length(lista[[7]]) - 1)
gg7 <- data.frame(t = tg, y = lista[[7]] )
plot7 <- ggplot(data = gg7, aes(x = t, y = y)) + geom_line() + labs(title = "Hypercalcemia")
#+ theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))
th <- seq(0,length(lista[[8]]) - 1)
gg8 <- data.frame(t = th, y = lista[[8]] )
plot8 <- ggplot(data = gg8, aes(x = t, y = y)) + geom_line() + labs(title = "Hypocalcemia")
#+ theme(plot.margin = unit(c(0.5, 0.5, 1, 0.5), "cm"))

#mostrando o painel criado
grid.arrange(plot4, plot7, plot5, plot8, plot6, plot3, plot1, plot2, ncol = 2, nrow = 4,
             bottom = textGrob("tempo discreto", gp = gpar(fontsize = 14, fontface = "bold")),
             left = textGrob("Amplitude (mV)", rot = 90, gp = gpar(fontsize = 14, fontface = "bold")))
```

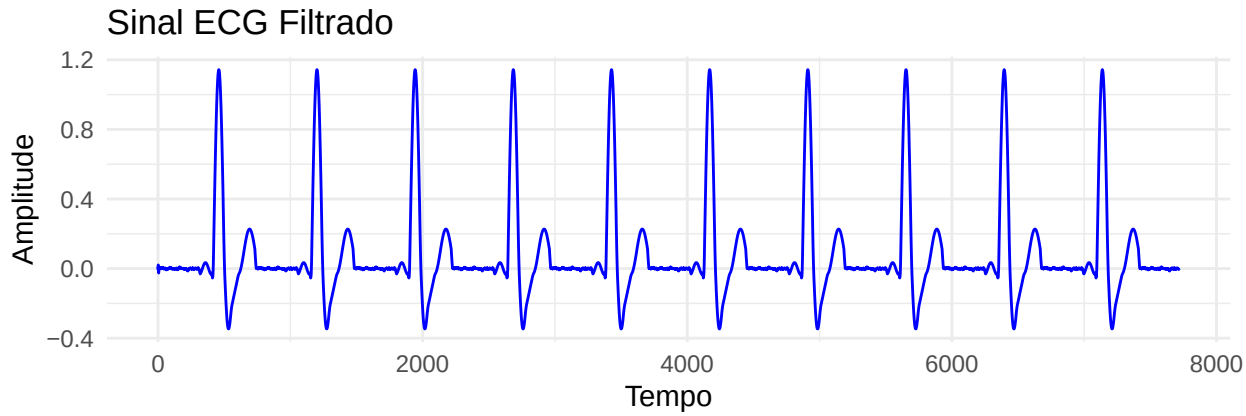


Simular um sinal ECG, simEMG, com 10 batimentos. O sinal deve ser gerado a partir da concatenação de um trecho de ruído com distribuição normal (considere 300 amostras para a geração do ruído), seguido pelo complexo PQRST (considere os parâmetros relativos à tachycardia), e seguido novamente por um trecho de ruído com distribuição normal. A amplitude do ruído deve ser 1% do valor máximo do complexo PQRST. O sinal final simulado, simECG, deve ser obtido aplicando-se a Equação 10. Como resultado final, deve-se gerar um gráfico de duas linhas e uma coluna, sendo que na primeira linha deve-se plotar o sinal simECG antes de aplicar a Equação 10 e na segunda linha deve-se plotar o sinal simECG após a aplicação da Equação 10. Um exemplo de resultado esperado para essa questão segue na Figura 4. Dica: use a função randn da biblioteca pracma para gerar o ruído.

```r
# Carregando a biblioteca necessária
library(pracma)

# Selecionando os dados da condição de taquicardia
dados_taq <- subset(condicoes_m, dadosB == condicoes_m$dadosB[1])

# Gerando os componentes do sinal ECG
sinais <- list(
  B = B(KB = dados_taq$KB),
  P = P(AP = dados_taq$AP, KP = dados_taq$KP),
  PQ = PQ(KPQ = dados_taq$KPQ),
  Q = Q(AQ = dados_taq$AQ, KQ = dados_taq$KQ),
  R = R(AR = dados_taq$AR, KR = dados_taq$KR),
  S = S(AS = dados_taq$AS, KS = dados_taq$KS, KCS = dados_taq$KCS),
```

```
  ST = ST(AS = dados_taq$AS, KS = dados_taq$KS, KCS = dados_taq$KCS, KST = dados_taq$KST, sm = dados_taq
  T_ = T_(AT = dados_taq$AT, KT = dados_taq$KT, KST = dados_taq$KST, AS = dados_taq$AS, KS = dados_taq$
  I = I(AT = dados_taq$AT, KT = dados_taq$KT, KST = dados_taq$KST, AS = dados_taq$AS, KS = dados_taq$KS
)

# Combinando os sinais
sinal_taq <- unlist(sinais)

# Criando ruído aleatório e adicionando ao sinal
set.seed(125)
ruido <- randn(n = 300, m = 1) * (max(sinal_taq) / 100)

# Gerando o sinal ECG repetitivo
simECG <- rep(c(ruido, sinal_taq), 10)
simECG <- c(simECG, ruido)

tempo <- seq_along(simECG) - 1

# Aplicando um filtro simples ao ECG
simECG_filt <- numeric(length(simECG))
for (k in seq_along(simECG)) {
  simECG_filt[k] <- if (k < 10) simECG[k] else mean(simECG[(k-9):k])
}

# Criando um dataframe para visualização
df_ECG <- data.frame(tempo, simECG, simECG_filt)

# Gerando os gráficos
library(ggplot2)
library(cowplot)

g1 <- ggplot(df_ECG, aes(tempo, simECG)) +
  geom_line(color = "red", na.rm = TRUE) +
  ggtitle('Sinal ECG Simulado') +
  theme_minimal() +
  xlab("Tempo") +
  ylab("Amplitude")

g2 <- ggplot(df_ECG, aes(tempo, simECG_filt)) +
  geom_line(color = "blue", na.rm = TRUE) +
  ggtitle('Sinal ECG Filtrado') +
  theme_minimal() +
  xlab("Tempo") +
  ylab("Amplitude")

# Exibindo os gráficos lado a lado
plot_grid(g1, g2, nrow = 2)
```





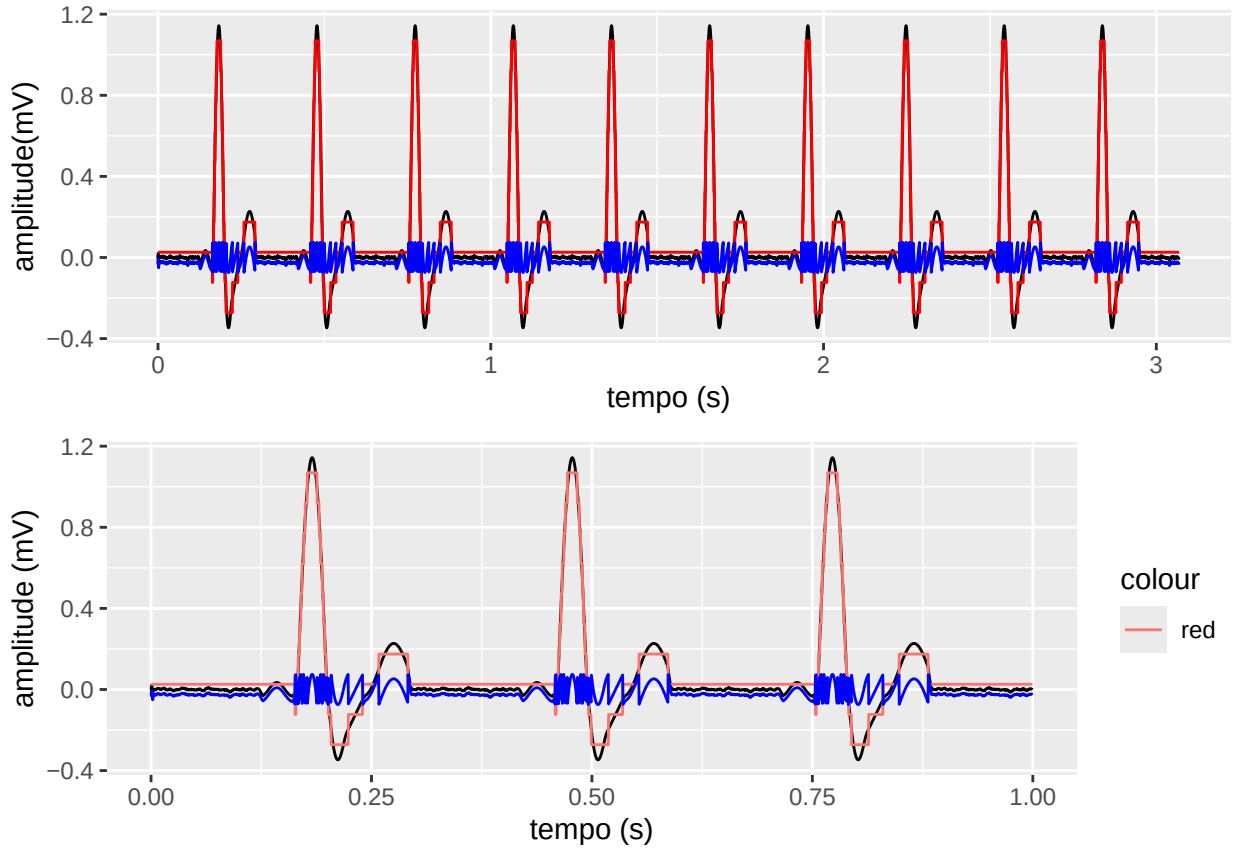
Quantizar o sinal simulado gerado no item c (simECG) em dez níveis. O gráfico do sinal quantizado deve se plotado juntamente com o gráfico do sinal simulado. O erro de quantização deve ser calculado e plotado. O resultado deve ser apresentado de acordo com o exemplo dado na Figura 5. Para a geração do vetor de tempo em segundos você deverá considerar que 503 amostras do sinal correspondem a 200 ms.

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.4     v tibble    3.2.1
## v purrr     1.0.4     v tidyr     1.3.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::combine()   masks gridExtra::combine()
## x purrr::cross()     masks pracma::cross()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x lubridate::stamp() masks cowplot::stamp()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
Vmax <- max(simECG_filt) # valor máximo
Vmin <- min(simECG_filt) # valor mínimo
k <- 10 # número de bits
```

```r
Q <- (Vmax - Vmin)/k # resolução A/D
qts <- seq(from = Vmin+(Q/2), to = Vmax-(Q/2), by = Q) # níveis disponíveis
#simECGq <- qts[findInterval(simECG, qts)]
#t <- seq(0, 0.2, length.out = 503)
#df <- data.frame(t = t, simECG = simECG, simECGq = simECGq)

ECG <- df_ECG

ECG$simECG_q <- 1 #criando uma nova coluna no data frame povoada por 1

for (i in 1:length(ECG$simECG)){
  ECG$simECG_q[i] <- qts[which.min(abs(qts-ECG$simECG_filt[i]))]
}

ECG <- ECG|> mutate(t = tempo * (0.200 / 503))
ECG <- ECG|> mutate(erro = (simECG_filt - simECG_q))

# Plotando ggplot1
ggplot1 <- ggplot(data = ECG) +
  geom_line(mapping = aes(x = t,y = simECG_filt),color = 'black') +
  geom_line(mapping = aes(x = t,y = simECG_q),color = 'red') +
  geom_line(mapping = aes(x = t,y = erro),color = 'blue') +
  labs(x = 'tempo (s)', y = 'amplitude(mV)')

sinalECG_a <- ECG |> filter(t < 1.0)

# plotando ggplot2
ggplot2 <- ggplot(data = sinalECG_a) +
  geom_line(mapping = aes(x = t, y = simECG_filt), color = 'black') +
  geom_line(mapping = aes(x = t , y = simECG_q, color = 'red')) +
  geom_line(mapping = aes(x = t, y = erro), color = 'blue') +
  labs(y = "amplitude (mV)", x = "tempo (s)")

#plotando painel
plot_grid(ggplot1, ggplot2, ncol = 1)
```




```
# Determina o tempo máximo do sinal ECG
t_1 <- max(ECG$t)

# Define a taxa de amostragem para subamostragem
fs <- 1000

# Calcula o intervalo de tempo entre amostras
dt <- 1 / fs

# Gera o vetor de tempo com a nova resolução
t_ <- seq(from = 0, to = t_1, by = dt)

# Realiza a interpolação do sinal ECG no novo vetor de tempo
ECG_subamostrado <- spline(x = ECG$t, y = ECG$simECG_filt, xout = t_)$y

# Filtra o sinal ECG no intervalo de interesse
ECG <- ECG |> filter(t >= 1, t <= 1.25)

# Cria um dataframe para armazenar o sinal subamostrado dentro do intervalo selecionado
simECG_subamostrado <- data.frame(t_, ECG_subamostrado) |> filter(t_ >= 1, t_ <= 1.25)

# Define as cores para o gráfico
cores <- c("Sinal ECG" = "red", "Sinal ECG Subamostrado" = "blue")

# Inicializa o gráfico
gg <- ggplot() +
```

```
  geom_line(data = ECG, aes(x = t, y = simECG_filt, color = "Sinal ECG")) +
  geom_point(data = ECG, aes(x = t, y = simECG_filt), color = 'black', size = 1, shape = 16) +
  geom_point(data = simECG_subamostrado, aes(x = t_, y = ECG_subamostrado, color = "Sinal ECG Subamostra
  labs(x = "amplitude (mV)", y = "tempo (s)") +
  scale_color_manual(values = cores)

# Exibe o gráfico
print(gg)
```