Universidade Federal de Uberlândia

Lista do Módulo 9

Levy Gabriel da Silva Galvão

Uberlândia
2021

Módulo 9

Exercício 1

> Calcule os coeficientes da DFT para a sequência $x(n) = \{-1, 1, 2, -2\}$. Utilize a definição básica da DFT para realizar o cálculo.

```
dft <- function(x,N)
{
  X <- rep(0+0i, N)
  range_ <- c(0:(N-1))
  for(k in range_){X[k+1] <- sum(x[1:N] * exp(-1i*2*pi*range_*k/N))}
  return(X)
}
x <- c(-1,1,2,-2)
X <- dft(x, length(x))
print(X) # dft coefficients
```

```
## [1]  0+0i -3-3i  2+0i -3+3i
```

Exercício 2

> Calcule a matriz de rotação de fatores, W, para a sequência $x(n) = \{-1, 1, 2, -2\}$. Calcule os coeficientes da DFT e da IDFT baseado nesta matriz.

```
rotation_matrix <- function(N)
{
  range_ <- 0:(N-1)
  W <- matrix(nrow=N,ncol=N)
  for(k in range_)
  {
    for(n in range_)
    {
      W[k+1,n+1] <- exp(-1i*(2*pi/N)*k*n)
    }
  }
  return(W)
}
x <- c(-1,1,2,-2)
N <- length(x)

W <- rotation_matrix(N)
print(W) # rotation matrix
```

```
##      [,1]  [,2]  [,3]  [,4]
## [1,] 1+0i  1+0i  1+0i  1+0i
## [2,] 1+0i  0-1i -1-0i  0+1i
## [3,] 1+0i -1-0i  1+0i -1-0i
## [4,] 1+0i  0+1i -1-0i  0-1i
```

```
X <- W %*% x
print(X) # dft coefficients
```

```
##       [,1]
## [1,]  0+0i
## [2,] -3-3i
## [3,]  2+0i
## [4,] -3+3i
```

```
invW <- (1/N) * Conj(W)
print(invW) # inverse of rotation matrix
```

```
##         [,1]        [,2]     [,3]        [,4]
## [1,] 0.25+0i  0.25+0.00i  0.25+0i  0.25+0.00i
## [2,] 0.25+0i  0.00+0.25i -0.25+0i  0.00-0.25i
## [3,] 0.25+0i -0.25+0.00i  0.25-0i -0.25+0.00i
## [4,] 0.25+0i  0.00-0.25i -0.25+0i  0.00+0.25i
```

```
xx <- invW %*% X
print(xx) # original sequence by applying idft
```

```
##       [,1]
## [1,] -1+0i
## [2,]  1+0i
## [3,]  2+0i
## [4,] -2+0i
```

Exercício 3

> Assumindo que a sequência $x(n) = \{-1, 1, 2, -2\}$ foi amostrada a $fs = 33Hz$, qual a resolução em frequência da DFT?

```
x <- c(-1,1,2,-2)
N <- length(x)
fs <- 33

freq_res <- fs/N
print(freq_res) # frequency resolution if a DFT is applied in the given sequence
```

```
## [1] 8.25
```

Exercício 4

> Gere um sinal senoidal, oscilando a 20 Hz, amostrado a 500 Hz. Calcule o espectro de amplitude e de fase da DFT para o sinal. Plote os gráficos dos espectros obtidos. Dica: o intervalo entre os coeficientes da DFT é a resolução em frequência em Hz.

```
library(signal)
```

```
##
## Attaching package: 'signal'
```

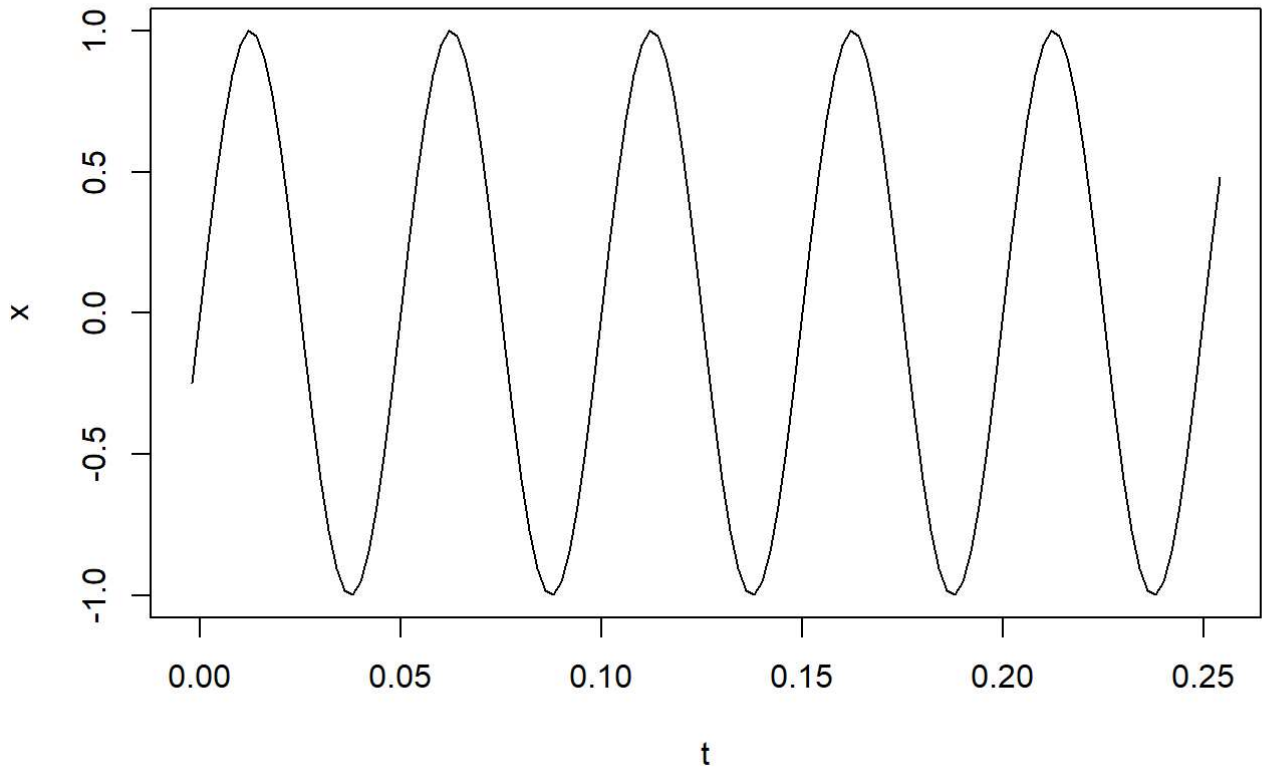
```
## The following objects are masked from 'package:stats':
##
##     filter, poly
```

```
library(REdaS)
```

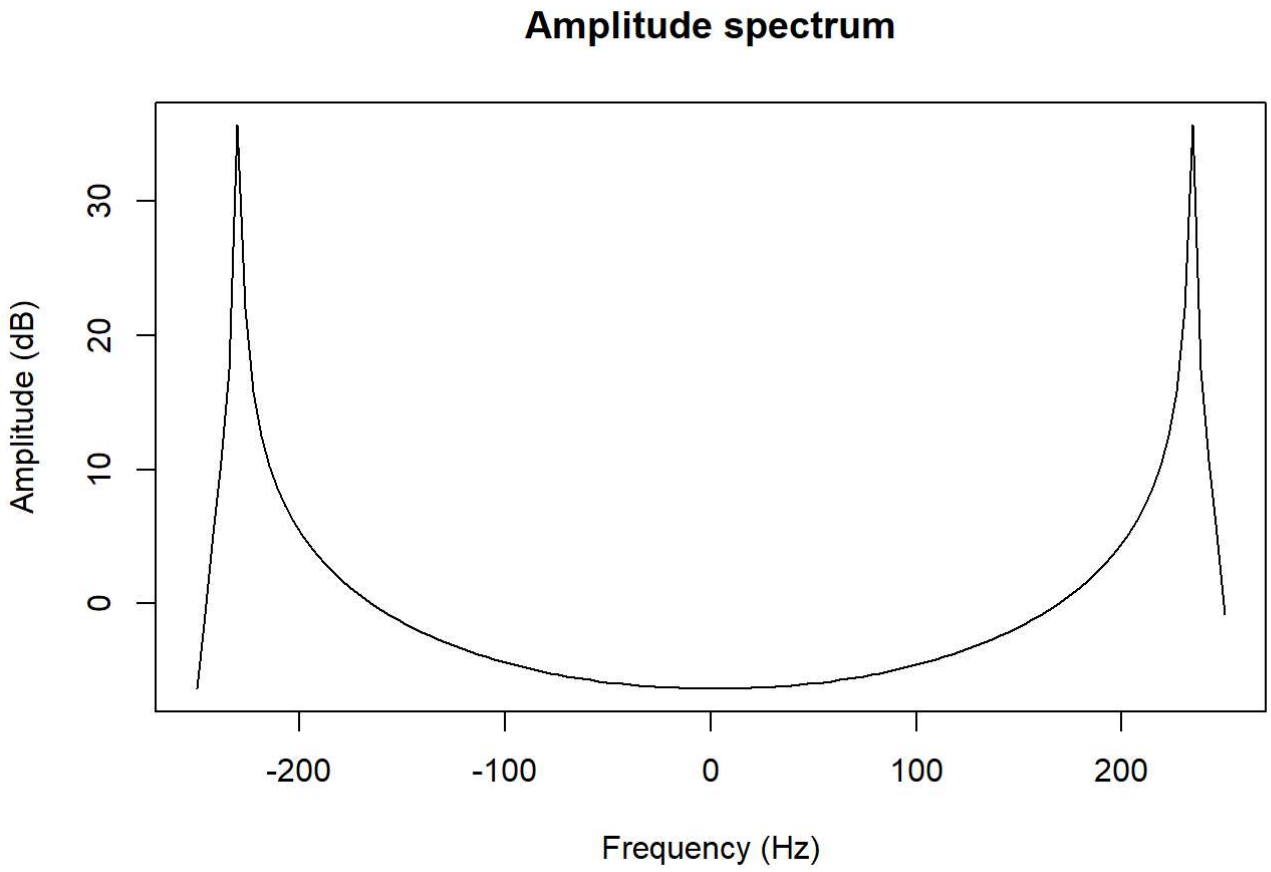
```
## Warning: package 'REdaS' was built under R version 4.1.1
```

```
## Carregando pacotes exigidos: grid
```

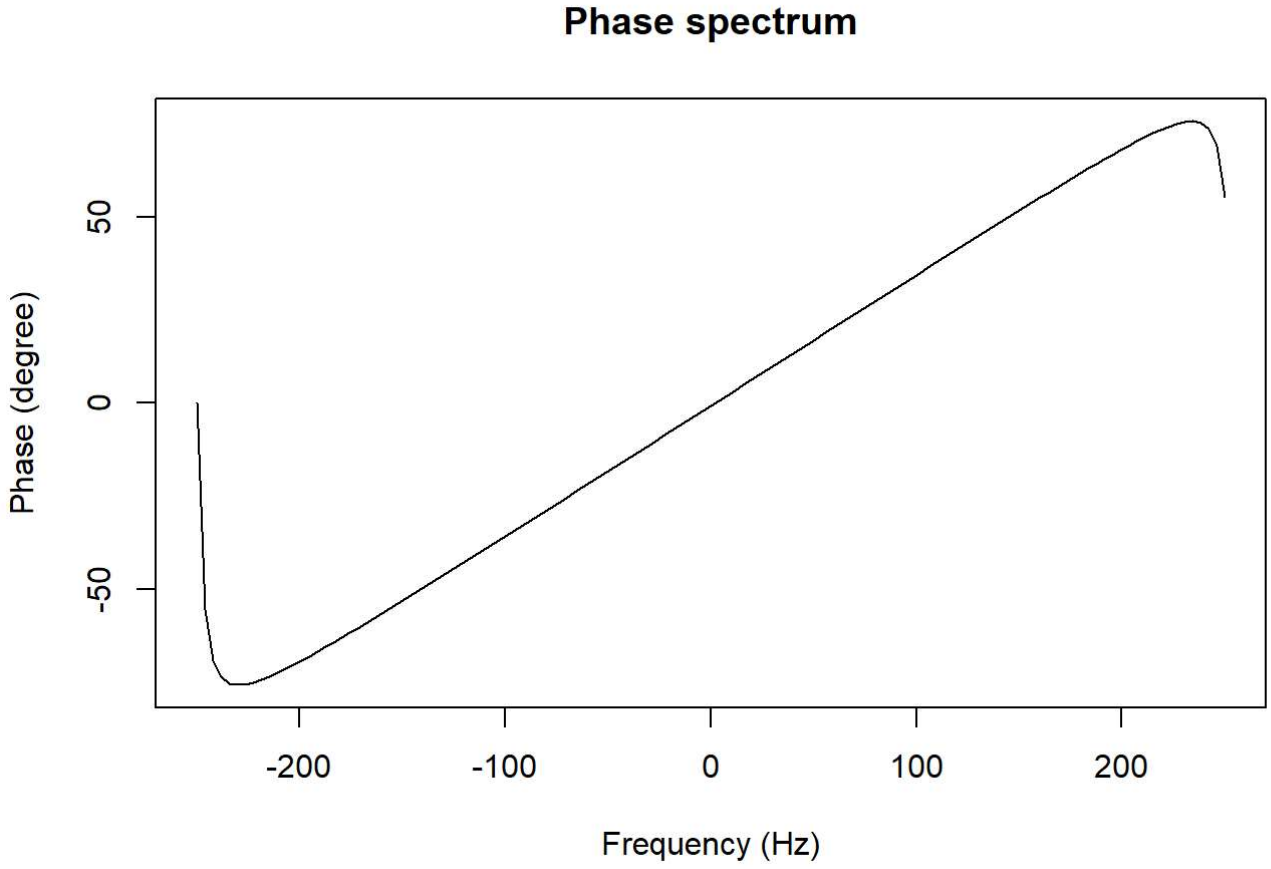
```
f <- 20 # Hz
fs <- 500 # Hz
dt <- 1/fs # s
n <- 128 # samples
t <- (0:n-1)/fs
x <- sin(2*pi*f*t)
plot(t,x,'l')
```




```
# dft spectrum
X <- dft(x,length(x))

spec_amp <- 20 * log10(sqrt(Re(X)^2 + Im(X)^2))
spec_phase <- rad2deg(atan(Im(X)/Re(X)))
freq <- seq(from=-fs/2, to=fs/2, by=fs/n)

plot(freq, spec_amp, 'l', main='Amplitude spectrum', xlab='Frequency (Hz)', ylab='Amplitude
 (dB)')
```




```
plot(freq, spec_phase, 'l', main='Phase spectrum', xlab='Frequency (Hz)', ylab='Phase (degre
e)')
```



Exercício 5

> Explique o que é ordenação bit-reversa e forneça um exemplo de aplicação da mesma sobre a sequência de caracteres `hojeodiaestabelo`. Qual a sequência resultante?

A ordenação bit-reversa se dá pelo estabelecimento de uma sequência de tamanho arbitrário na base 2 e numerados sequencialmente e, em que seguida a ordenação bit-reversa mapeia cada item em uma nova posição que é dada pela reversão binária da posição, então um item na posião 3, quem em binário é 011, será mapeado para a posição 110, que em decimal será 6. Vide abaixo o exemplo:

```
IntToBits <- function(x, nBits = 8)
{
  tail(rev(as.numeric(intToBits(x))),nBits)
}
BitsToInt <- function(x)
{
  s <- seq(length(x)-1,0,-1)
  acc <- 0
  for(i in (1:length(x))){acc = acc + x[i]*2^(s[i])}
  return(acc)
}
BitReversal <- function(bits_)
{
  bits_[seq(length(bits_),1,-1)]
}

library(stringr)
str_ <- 'hojeodiaestabelo'
list_ <- unlist(strsplit(str_, ""))
n <- log2(length(list_))
N <- length(list_)
idx <- 0:(N-1)
new_idx <- rep(NA,N)
for(i in (1:N)){new_idx[i] <- idx[i] %>% IntToBits(nBits=n) %>% BitReversal() %>% BitsToInt
()}

cat('original index:', idx, '\n')
```

```
## original index: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
```

```
cat('new index:', new_idx, '\n\n')
```

```
## new index: 0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15
```

```
cat('original string:', list_[idx+1], '\n')
```

```
## original string: h o j e o d i a e s t a b e l o
```

```
cat('new string:', list_[new_idx+1], '\n')
```

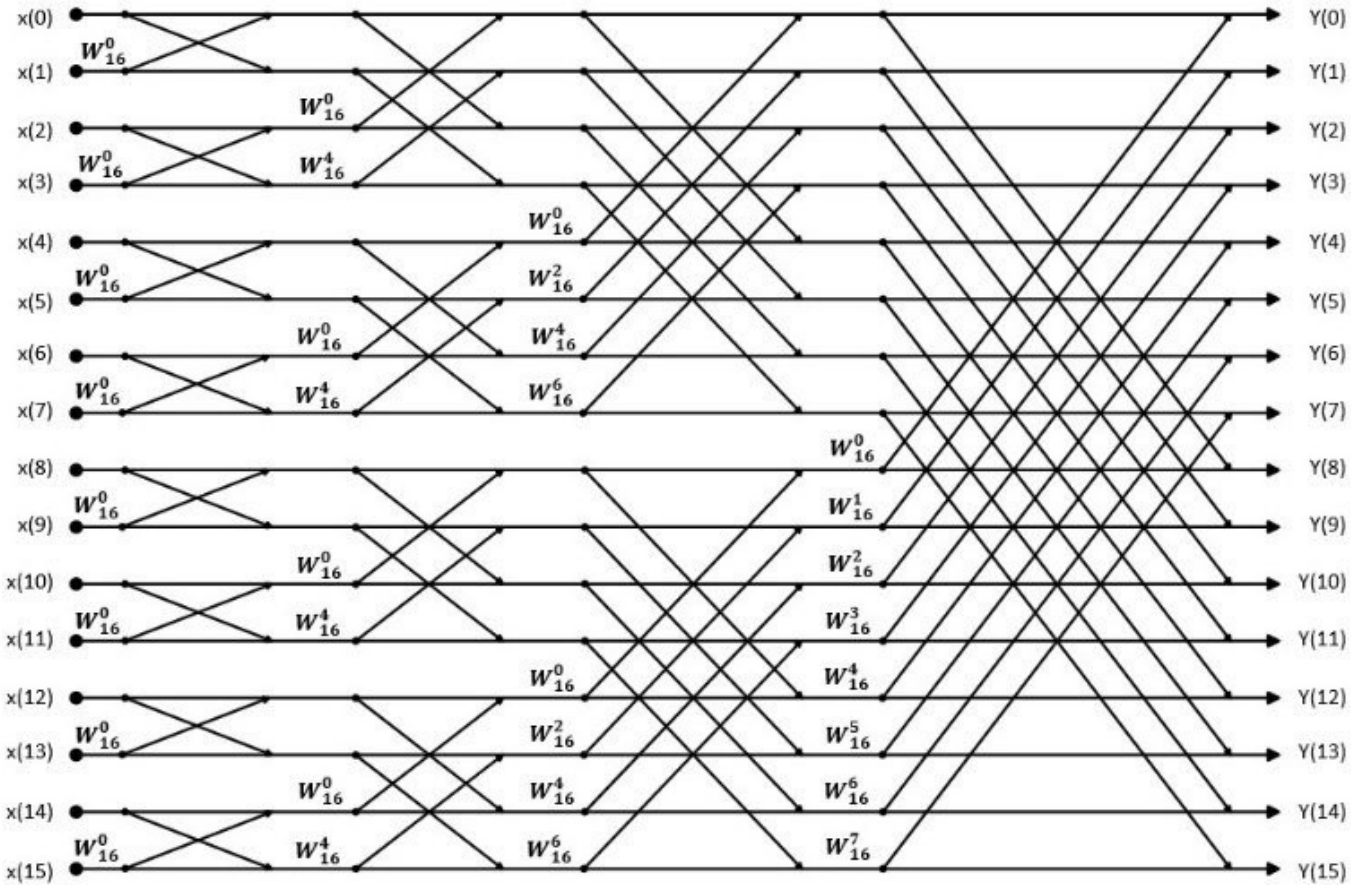
```
## new string: h e o b j t i l o s d e e a a o
```

Exercício 6

> Desenhe um diagrama de butterfly para 16 amostras. Apresente as equações de cada saída X(k). Qual o ganho em velocidade da FFT quando comparado à DFT neste exemplo?

Diagrama de borboleta de 16 amostras

Source



Este número de amostras da FFT reduz o número de interações de $N \cdot N = 16 \cdot 16 = 256$ para $Nlog_2(N) = 16log_2(16) = 64$ multiplicações.

Exercício 7

> No ambiente do R, leia o help da função fft (package: stats). Neste help existe uma implementação da DTF. Estude e comente os códigos apresentados no exemplo. Execute o exemplo, utilizando o sinal de entrada Z, e calculando os coeficientes da DFT por meio da equação geral e por meio da DFT. Utilize o trecho de código abaixo para calcular a diferença temporal entre a DFT e a FFT.

```r
x <- 1:4 # generate array

fft(x) # DFT
```

```
## [1] 10+0i -2+2i -2+0i -2-2i
```

```r
fft(fft(x), inverse = TRUE)/length(x) # IDFT
```

```
## [1] 1+0i 2+0i 3+0i 4+0i
```

```r
## Slow Discrete Fourier Transform (DFT) - e.g., for checking the formula
fft0 <- function(z, inverse=FALSE) { # implementation of DFT (slower than fft)
  n <- length(z)
  if(n == 0) return(z)
  k <- 0:(n-1)
  ff <- (if(inverse) 1 else -1) * 2*pi * 1i * k/n
  vapply(1:n, function(h) sum(z * exp(ff*(h-1))), complex(1))
}

relD <- function(x,y) 2* abs(x - y) / abs(x + y)
n <- 2^8
z <- complex(n, rnorm(n), rnorm(n))
## relative differences in the order of 4*10^{-14} :
summary(relD(fft(z), fft0(z)))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 2.569e-16 1.213e-14 2.881e-14 5.383e-14 6.271e-14 1.704e-12
```

```r
summary(relD(fft(z, inverse=TRUE), fft0(z, inverse=TRUE)))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 2.569e-16 1.330e-14 2.807e-14 5.373e-14 5.619e-14 1.288e-12
```

```r
# comparing execution time of fft vs. dft
ptm <- proc.time()
Z1 <- fft(z)
dur_fft <- proc.time() - ptm
ptm <- proc.time()
Z2 <- fft0(z)
dur_dft <- proc.time() - ptm

cat('\nduration fft:',dur_fft['elapsed'])
```

```
##
## duration fft: 0
```

```r
cat('\nduration dft:',dur_dft['elapsed'])
```

```
##
## duration dft: 0.02
```

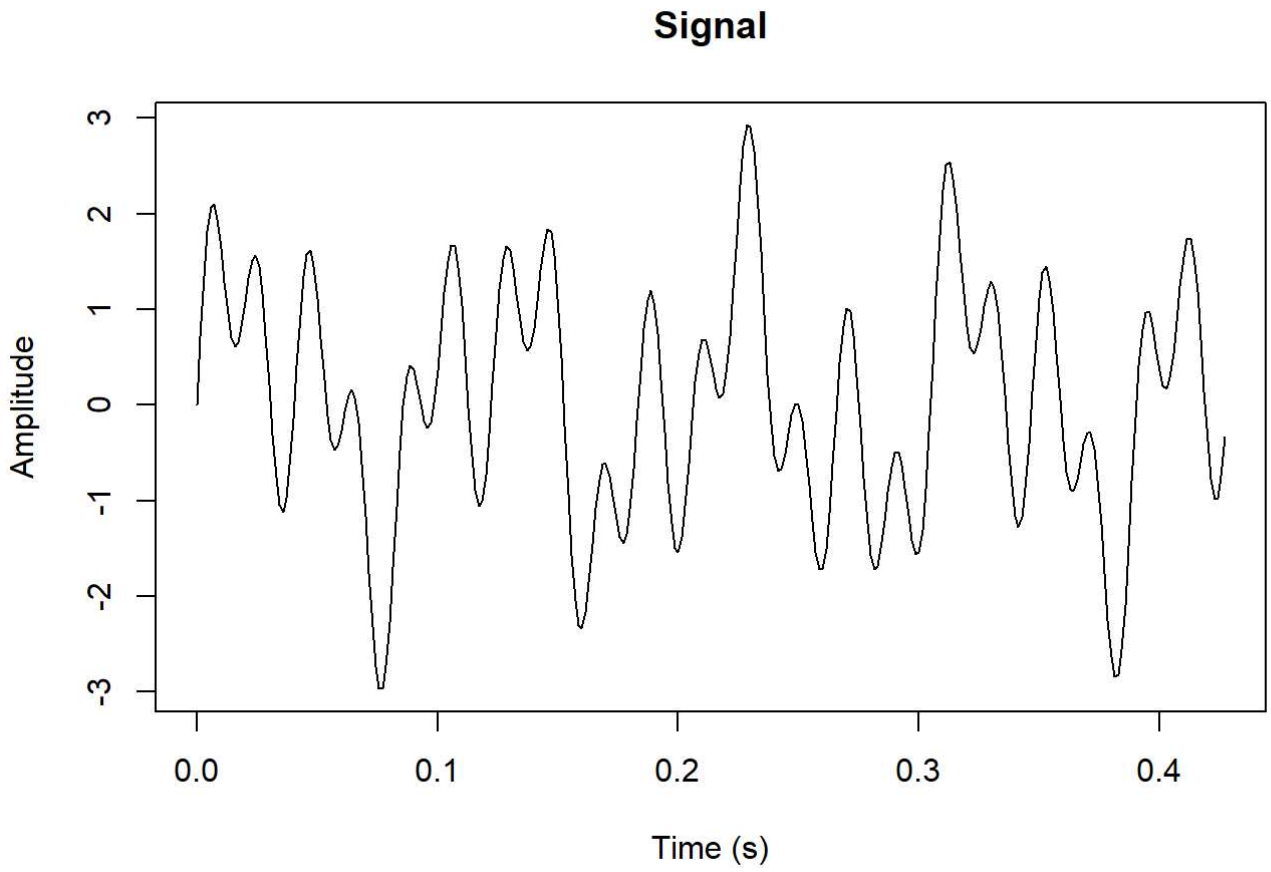
De fato foi comprovado que a a DFT executa em maior tempo que a FFT.

Exercício 8

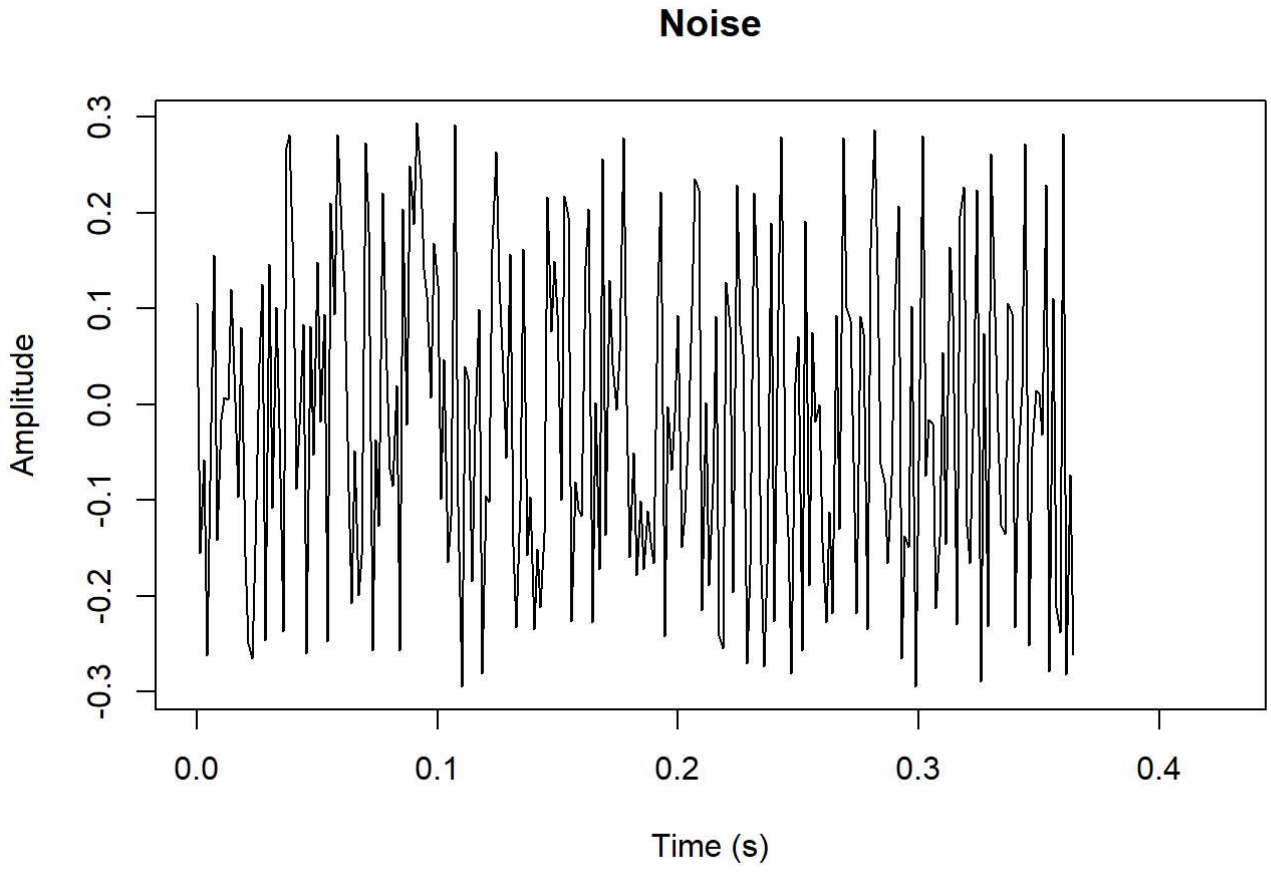
```
norm_dist <- function(length_, min_, max_){return(qnorm(runif(length_,min=pnorm(min_),max=pno
rm(max_))))}

fs <- 700 # Hz
f1 <- 10 # Hz
f2 <- 23 # Hz
f3 <- 49 # Hz

dt <- 1/fs # s
t_end <- 10 # s
t <- seq(0,t_end-dt,dt)
N <- length(t)

x <- sin(2*pi*f1*t) + sin(2*pi*f2*t) + sin(2*pi*f3*t)
max_ <- max(x)
n <- norm_dist(n, -0.1*max_, 0.1*max_)
y <- x + n
```

```
## Warning in x + n: comprimento do objeto maior não é múltiplo do comprimento do
## objeto menor
```

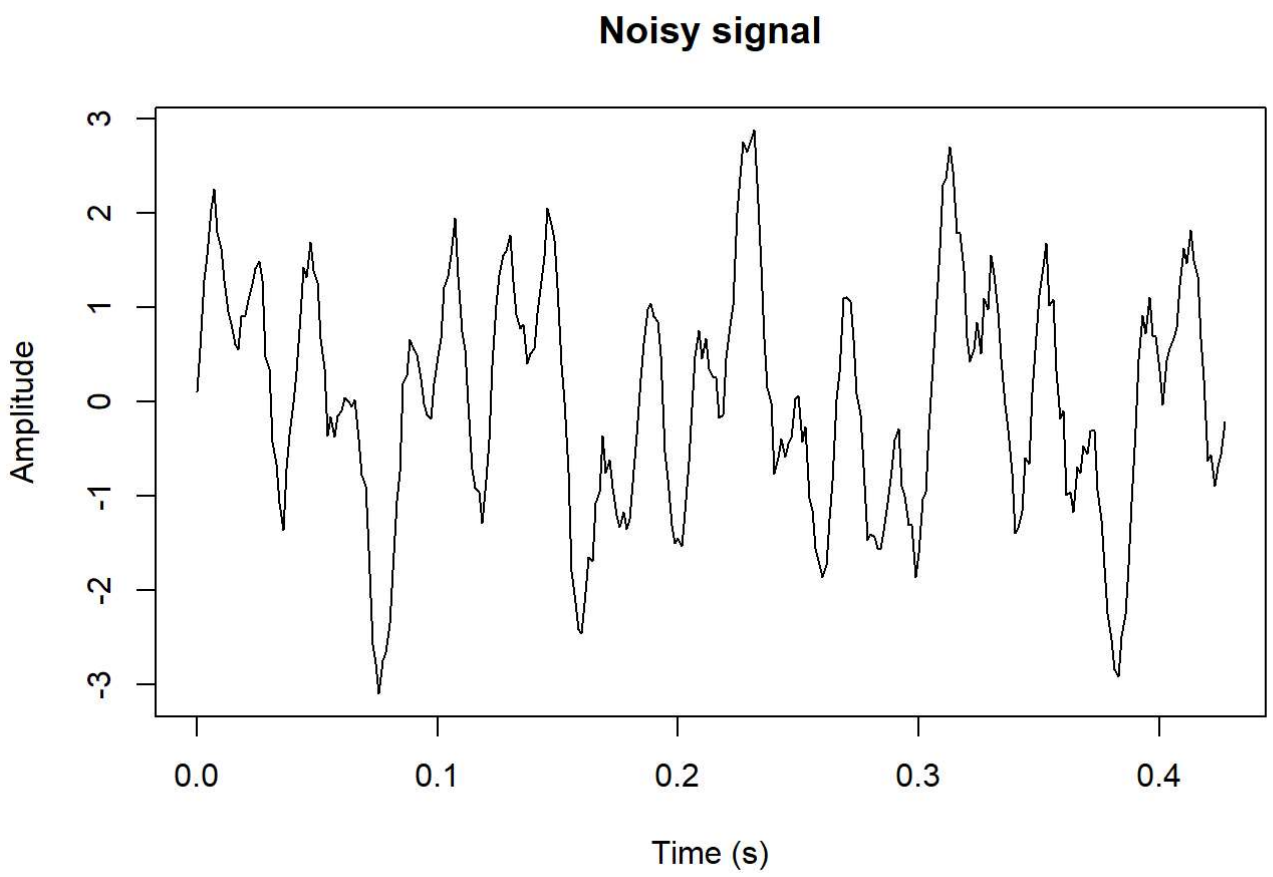
```
m = 300
plot(t[1:m],x[1:m],'l',main='Signal',xlab='Time (s)',ylab='Amplitude')
```




```
plot(t[1:m],n[1:m],'l',main='Noise',xlab='Time (s)',ylab='Amplitude')
```

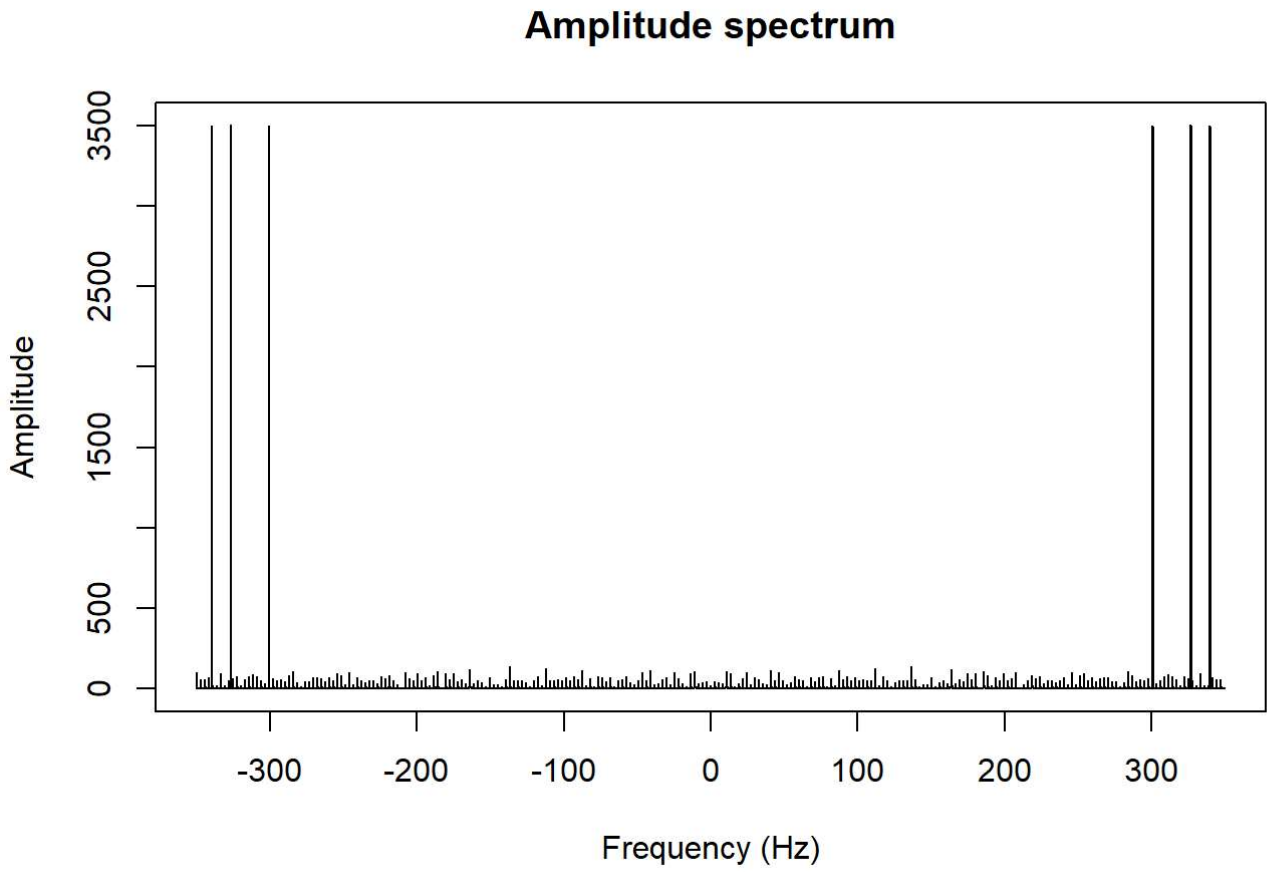


```
plot(t[1:m],y[1:m],'l',main='Noisy signal',xlab='Time (s)',ylab='Amplitude')
```



Assim o espectro em amplitude e fase são:

```
Y <- fft(y)

spec_amp <- sqrt(Re(Y)^2 + Im(Y)^2)
spec_phase <- rad2deg(atan2(Im(Y),Re(Y)))
freq <- seq(from=-fs/2, to=fs/2-fs/N, by=fs/N)

plot(freq, spec_amp, 'l', main='Amplitude spectrum', xlab='Frequency (Hz)', ylab='Amplitude')
```




```
plot(freq, spec_phase, 'l', main='Phase spectrum', xlab='Frequency (Hz)', ylab='Phase (degree)')
```