

MATH 545: INTRODUCTION TO TIME SERIES ANALYSIS

WINTER 2025

SPECTRAL ESTIMATION: SHORT AND LONG MEMORY MODELS

Model-free estimation: The spectral density function (SDF) can be estimated from time-domain samples. Let $x_{1:n}$ be the time-domain data, and let

$$\omega_k = \frac{2\pi k}{n}$$

for k equal to

$$-\left\lfloor \frac{n-1}{2} \right\rfloor, \dots, \left\lfloor \frac{n}{2} \right\rfloor$$

The set $\{\omega_k\}$ are termed the *Fourier frequencies*. For each k , define the $n \times 1$ vector e_k by

$$e_k = \frac{1}{\sqrt{n}}(e^{i\omega_k}, e^{2i\omega_k}, \dots, e^{ni\omega_k})^\top$$

so that $\bar{e}_j^\top e_k = 0$ unless $j = k$, in which case $\bar{e}_k^\top e_k = 1$.

```
n<-512
kvec<-seq(-floor((n-1)/2),n/2,by=1)
omega<-2*pi*kvec/n
emat<-matrix(0,nrow=n,ncol=length(omega))
for(k in 1:length(omega)){
  emat[,k]<-exp(1i*omega[k]*c(1:n))/sqrt(n)
}
Imat<-t(Conj(emat)) %*% emat
round(Imat[1:4,1:4],6)

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

$\{e_1, \dots, e_n\}$ form an orthonormal basis for $\mathbb{C}^n$, that is, for any n -dimensional complex number $\mathbf{x}$, we have

$$\mathbf{x} = \sum_{k=-\lfloor (n-1)/2 \rfloor}^{\lfloor n/2 \rfloor} a_k e_k$$

where

$$a_k = \bar{e}_k^\top \mathbf{x} = \frac{1}{\sqrt{n}} \sum_{t=1}^n x_t e^{-it\omega_k}$$

The vector $(a_1, \dots, a_n)^\top$ is the *discrete Fourier transform* of $\mathbf{x}$. Note that

$$x_t = \sum_{k=-\lfloor (n-1)/2 \rfloor}^{\lfloor n/2 \rfloor} a_k [\cos(\omega_k t) + i \sin(\omega_k t)]$$

is the discrete version of the spectral representation theorem.

Example: ARMA(1, 1) model with $\phi = 0.8$ and $\theta = 0.5$.

```
x<-arima.sim(n,model=list(ar=0.8,ma=0.5))
avec<-rep(0,n)
for(k in 1:length(omega)){
  avec[k]<-Conj(emat[,k]) %*% x
}
avec[1:5]

## [1] 0.2880099+0.1501901i -0.3757279+0.1611205i -0.3054674-0.1469424i
## [4] 0.1004059-0.1555172i -0.2098631-0.1110264i
```

Note: An equivalent formulation uses the re-labelled Fourier frequencies

$$\omega'_k = \frac{2\pi(k-1)}{n} \quad k = 1, \dots, n$$

and the representation using coefficients

$$b_k = \frac{1}{\sqrt{n}} \sum_{t=1}^n x_t e^{-i(t-1)\omega'_k}$$

This is the representation computed using the fast Fourier transform function `fft` in R, although that function does not introduce the $\sqrt{n}$ scaling. The re-labelled Fourier frequencies are re-indexed as

$$\omega'_k = \omega_l \quad l = \frac{n}{2} + k - 1 \quad \text{mod } n$$

for $k = 1, \dots, n$.

```
omega.new<-2*pi*c(0:(n-1))/n
bvec<-rep(0,n)
for(k in 1:length(omega)){
  bvec[k]<-sum(x*exp(-1i*c(0:(n-1))*omega.new[k]))
}
bvec[1:5]/sqrt(n)

## [1] 8.2838688+0.00000000i -3.4120874+0.62460503i -0.3277958-0.09725086i
## [4] -2.3191369-3.01699349i -3.0081092-4.75748619i

fft(x)[1:5]/sqrt(n)

## [1] 8.2838688+0.00000000i -3.4120874+0.62460503i -0.3277958-0.09725086i
## [4] -2.3191369-3.01699349i -3.0081092-4.75748619i

avec[(n/2)+c(0:4)]

## [1] 8.2838688+0.00000000i -3.4041657+0.66642956i -0.3300837-0.08917706i
## [4] -2.4286126-2.92958814i -3.2379246-4.60415467i

omega[(n/2)+c(0:4)]

## [1] 0.00000000 0.01227185 0.02454369 0.03681554 0.04908739

omega.new[1:5]

## [1] 0.00000000 0.01227185 0.02454369 0.03681554 0.04908739
```

The Periodogram: The *periodogram*, $\hat{I}_n(\lambda)$ is defined by

$$\hat{I}_n(\lambda) = \frac{1}{n} \left| \sum_{t=1}^n x_t e^{-it\lambda} \right|^2 \quad -\pi < \lambda \leq \pi.$$

Note that $\hat{I}_n(\omega_k) = |a_k|^2$, and

$$\bar{\mathbf{x}}^\top \mathbf{x} = \sum_{k=-\lfloor (n-1)/2 \rfloor}^{\lfloor n/2 \rfloor} |a_k|^2 = \sum_{k=-\lfloor (n-1)/2 \rfloor}^{\lfloor n/2 \rfloor} \hat{I}_n(\omega_k)$$

which gives a decomposition of the total variation in $\mathbf{x}$ via components at the ω_k .

```
sum(x^2)
## [1] 3231.655
sum(Mod(ave(x))^2)
## [1] 3231.655
```

Note that

$$\begin{aligned} \hat{I}_n(\omega_k) &= \left(\frac{1}{\sqrt{n}} \sum_{t=1}^n x_t e^{-it\omega_k} \right) \left(\frac{1}{\sqrt{n}} \sum_{s=1}^n x_s e^{is\omega_k} \right) \\ &= \frac{1}{n} \sum_{t=1}^n \sum_{s=1}^n x_t x_s e^{-i(t-s)\omega_k} \\ &= \frac{1}{n} \sum_{t=1}^n \sum_{s=1}^n (x_t - \bar{x})(x_s - \bar{x}) e^{-i(t-s)\omega_k} \\ &= \sum_{h=-(n-1)}^{n-1} \hat{\gamma}_n(h) e^{-ih\omega_k} \quad (h = s - t) \end{aligned}$$

as $\sum_{t=1}^n e^{-it\omega_k} = 0$ if $\omega_k \neq 0$. To see this last result note that

$$\sum_{t=1}^n e^{-it\omega_k} = \frac{e^{-i(n+1)\omega_k} - e^{-i\omega_k}}{1 - e^{-i\omega_k}}$$

by summing a finite geometric progression. In the numerator

$$\begin{aligned} e^{-i(n+1)\omega_k} - e^{-i\omega_k} &= (\cos(2\pi(n+1)k/n) - i \sin(2\pi(n+1)k/n)) - (\cos(2\pi k/n) - i \sin(2\pi k/n)) \\ &= 0. \end{aligned}$$

by the periodic nature of $\cos$ and $\sin$.

```
#For example
sum(exp(-1i*c(1:n)*omega[10])) #k=10
## [1] 0.000000000000002827816+0.000000000000008277049i
sum(exp(-1i*c(1:n)*omega[116])) #k=116
## [1] -0.000000000000002065444-0.00000000000001519188i
```

Thus $\hat{I}_n(\lambda)$ is a sample-based estimate of $2\pi f_X(\lambda)$; it is **not** consistent, however. For $0 < \lambda_1 < \dots < \lambda_m < \pi$,

$$(\hat{I}_n(\lambda_1), \dots, \hat{I}_n(\lambda_m))^T \xrightarrow{d} (V_1, \dots, V_n)^T$$

as $n \rightarrow \infty$, where

$$V_j \sim \text{Exponential}(1/(2\pi f_X(\lambda_j))).$$

That is, as $n \rightarrow \infty$

$$\mathbb{E}[\hat{I}_n(\lambda_j)] \rightarrow 2\pi f_X(\lambda_j)$$

so $\hat{I}_n(\lambda_j)$ is asymptotically unbiased, but $\hat{I}_n(\lambda_j)$ converges in distribution, not in probability or almost surely.

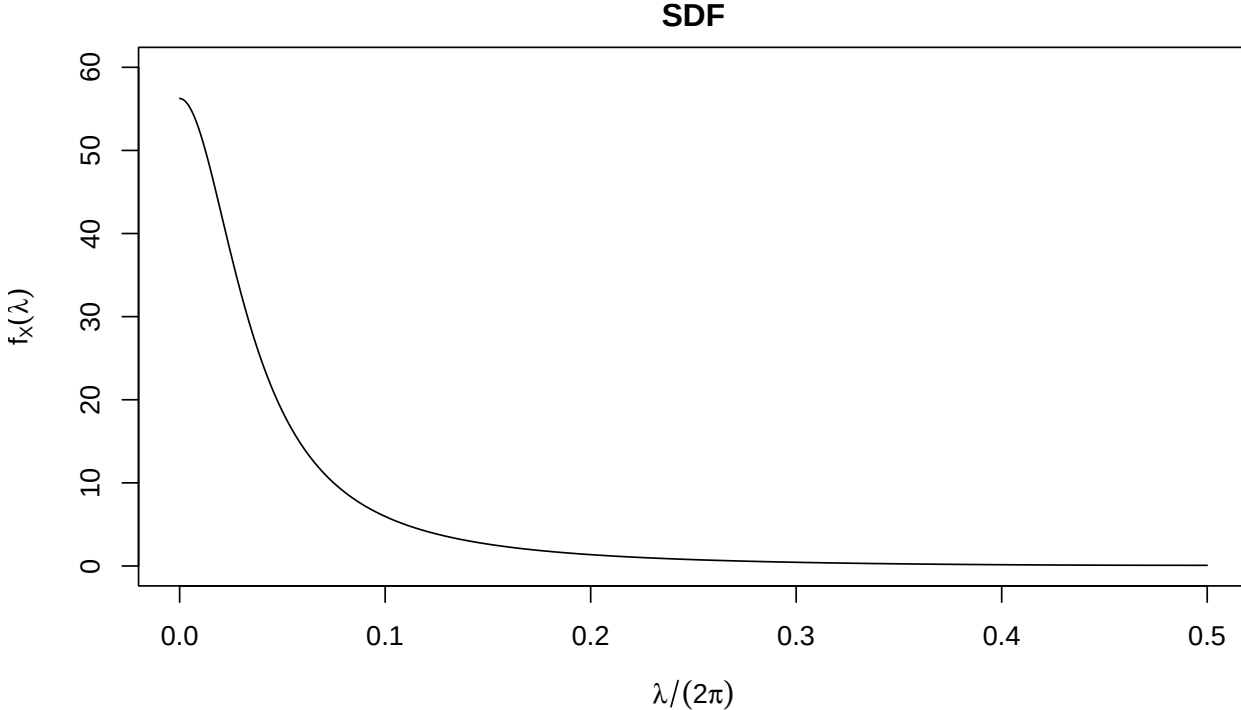
Example ARMA(1,1)

For the ARMA(1,1) model

$$(1 - 0.8B)X_t = (1 + 0.5B)Z_t$$

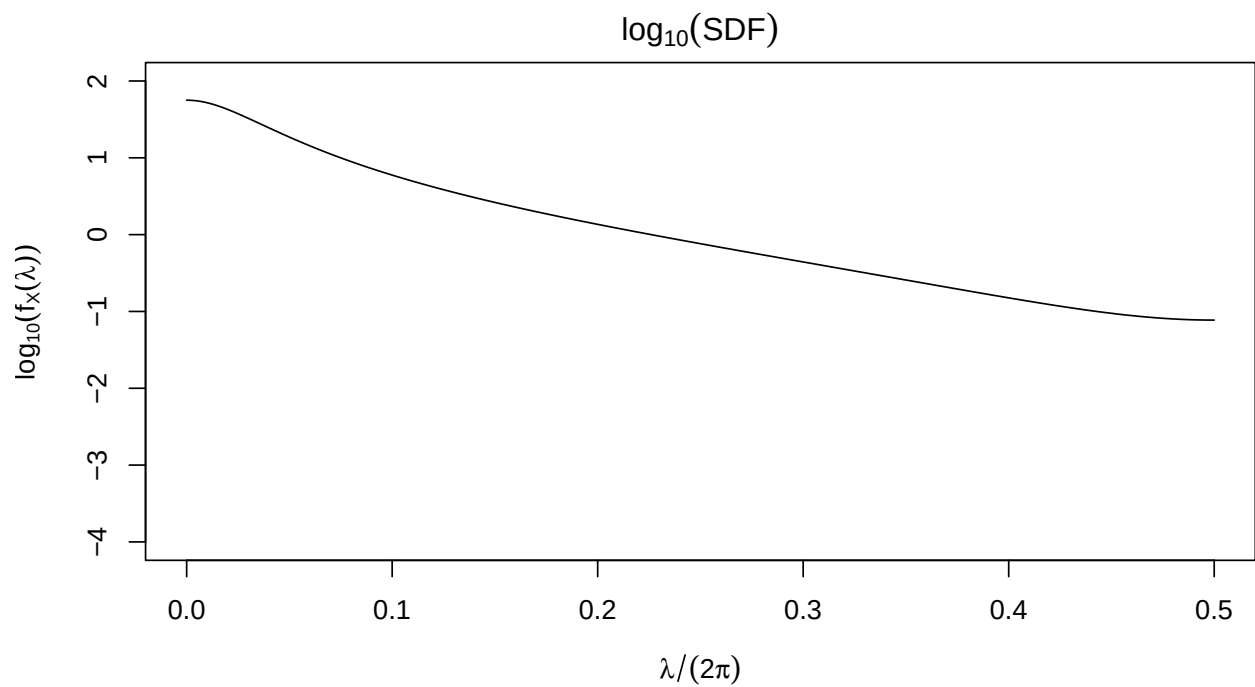
we examine the SDF by direct computation. We use `spectrum` to compute the periodogram: this uses the rescaled frequency scale, so we plot for ordinate $\lambda/2\pi$.

```
library(ltsa)
acvf.vals<-tacvfARMA(phi = 0.8, theta = -0.5, maxLag = 100)
lamvec<-seq(0,0.5,by=0.001)
sdf.vec<-lamvec*0
hvec<-0:100
for(j in 1:length(lamvec)){
  sdf.vec[j]<-sum(exp(-1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[-1])+acvf.vals[1]+
    sum(exp(1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[-1])
}
par(mar=c(4,4,2,2));plot(lamvec,sdf.vec,type='l',xlab=expression(lambda/(2*pi)),
  ylab=expression(f[X](lambda)), ylim=range(0,60),main = 'SDF')
```

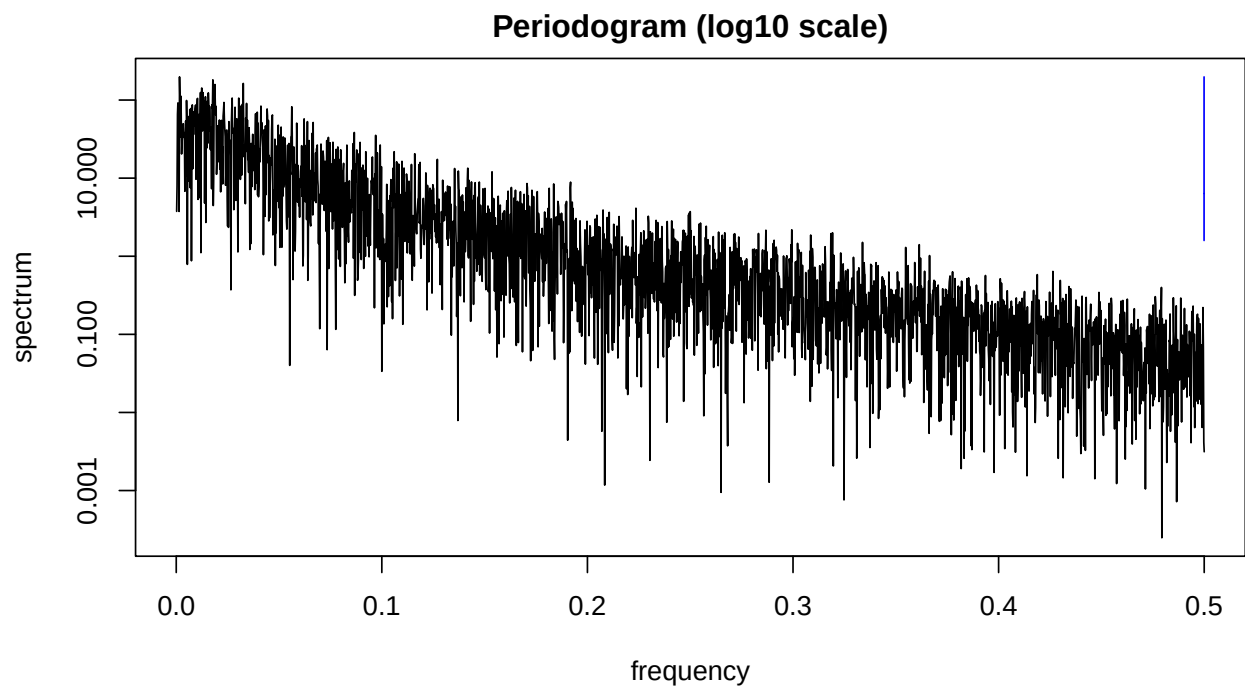


It is common to plot the periodogram on the $\log_{10}$ scale, so we re-plot the SDF on this scale also.

```
par(mar=c(4,4,2,2));plot(lamvec,log10(sdf.vec),type='l',xlab=expression(lambda/(2*pi)),
  ylab=expression(log[10](f[X](lambda))), ylim=range(-4,2),main = expression(log[10](SDF)))
```

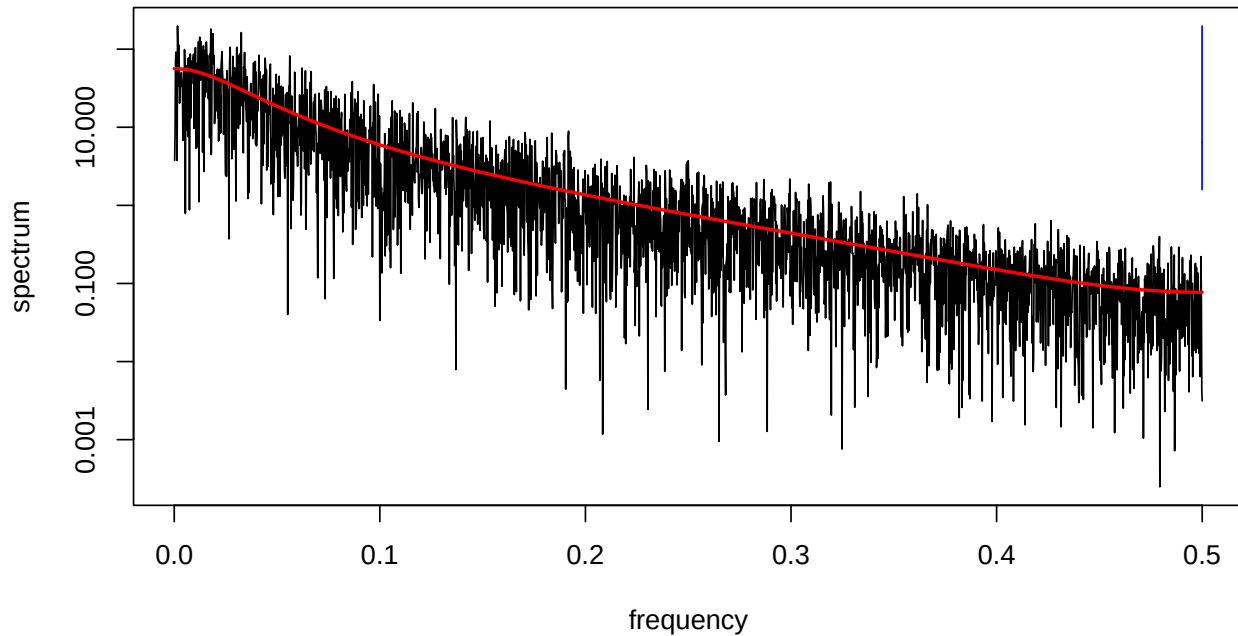


```
set.seed(3);
Xt<-arima.sim(n=5000,model=list(ar=0.8,ma=0.5))
par(mar=c(4,4,2,2));spectrum(Xt, main='Periodogram (log10 scale)', taper=0,detrend=FALSE)
```



```
par(mar=c(4,4,2,2));spectrum(Xt, main='Periodogram & True SDF (log10 scale)',
taper=0,detrend=FALSE)
lines(lamvec,sdf.vec,col='red',lwd=2)
```

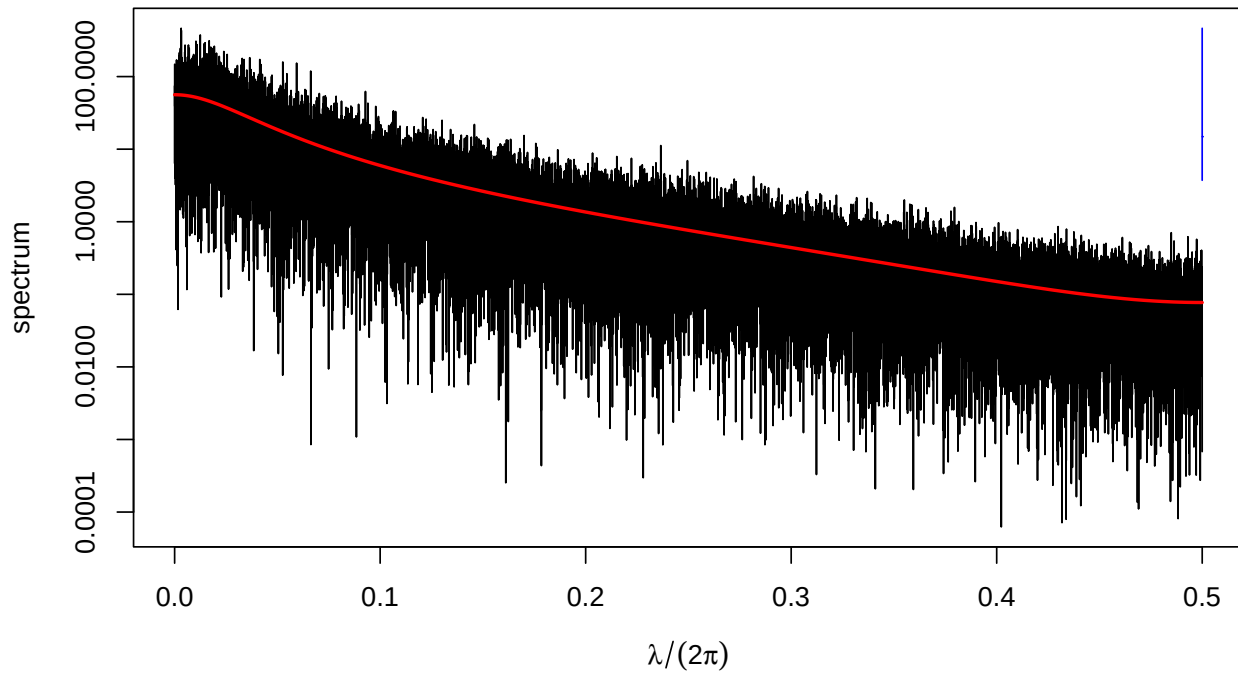
Periodogram & True SDF (log10 scale)



For a much larger sample size, the distributional result can be more clearly seen.

```
set.seed(3);
Xt<-arima.sim(n=50000,model=list(ar=0.8,ma=0.5))
par(mar=c(4,4,2,2));spectrum(Xt, main='Periodogram & True SDF (log10 scale, n=50000)',
  taper=0,detrend=FALSE,xlab=expression(lambda/(2*pi)))
lines(lamvec,sdf.vec,col='red',lwd=2)
```

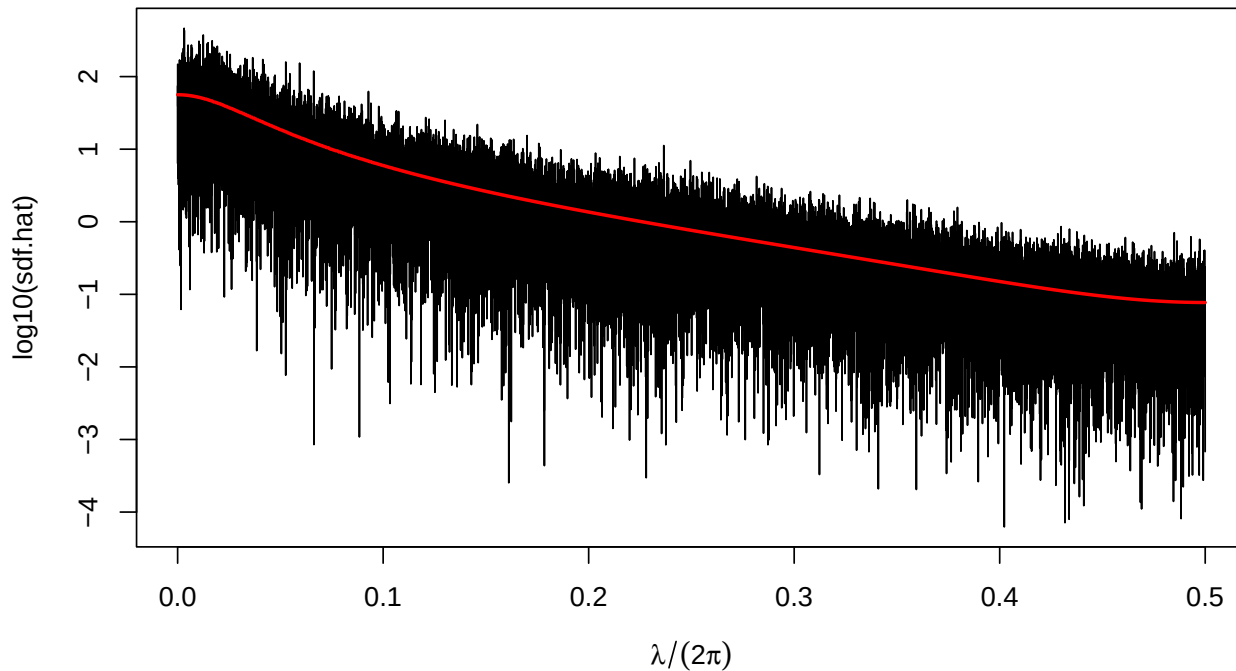
Periodogram & True SDF (log10 scale, n=50000)



```

sdf.hat<-spectrum(Xt, taper=0,detrend=FALSE,plot=FALSE)$spec
n0<-length(sdf.hat)
lamvec<-c(1:n0)/(2*n0)
sdf.vec<-lamvec*0
hvec<-0:100
for(j in 1:length(lamvec)){
  sdf.vec[j]<-sum(exp(-1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[2:length(hvec)])+acvf.vals[1]+
    sum(exp(1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[2:length(hvec)])
}
par(mar=c(4,4,2,2));plot(lamvec,log10(sdf.hat),type='l',xlab=expression(lambda/(2*pi)))
lines(lamvec,log10(sdf.vec),col='red',lwd=2)

```



If in the above notation,

$$V_j \sim \text{Exponential}(1/(2\pi f_X(\lambda_j))).$$

then

$$\frac{V_j}{2\pi f_X(\lambda_j)} \sim \text{Exponential}(1)$$

which we can verify in the sample.

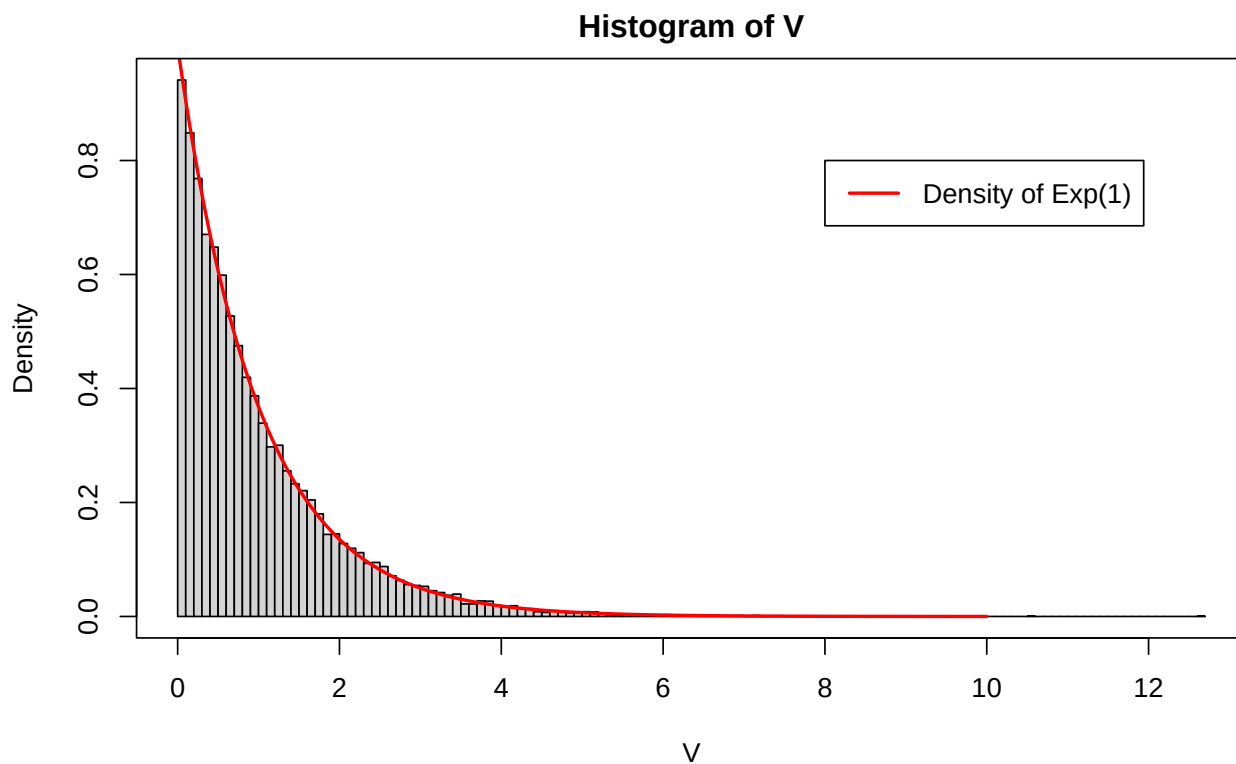
```

#Compute the transformed V values
#The factor of 2 pi is already incorporated to account for the output of spectrum.
Vvec<-as.numeric(sdf.hat/sdf.vec)
mean(Vvec); var(Vvec)

## [1] 1.012817
## [1] 1.024205

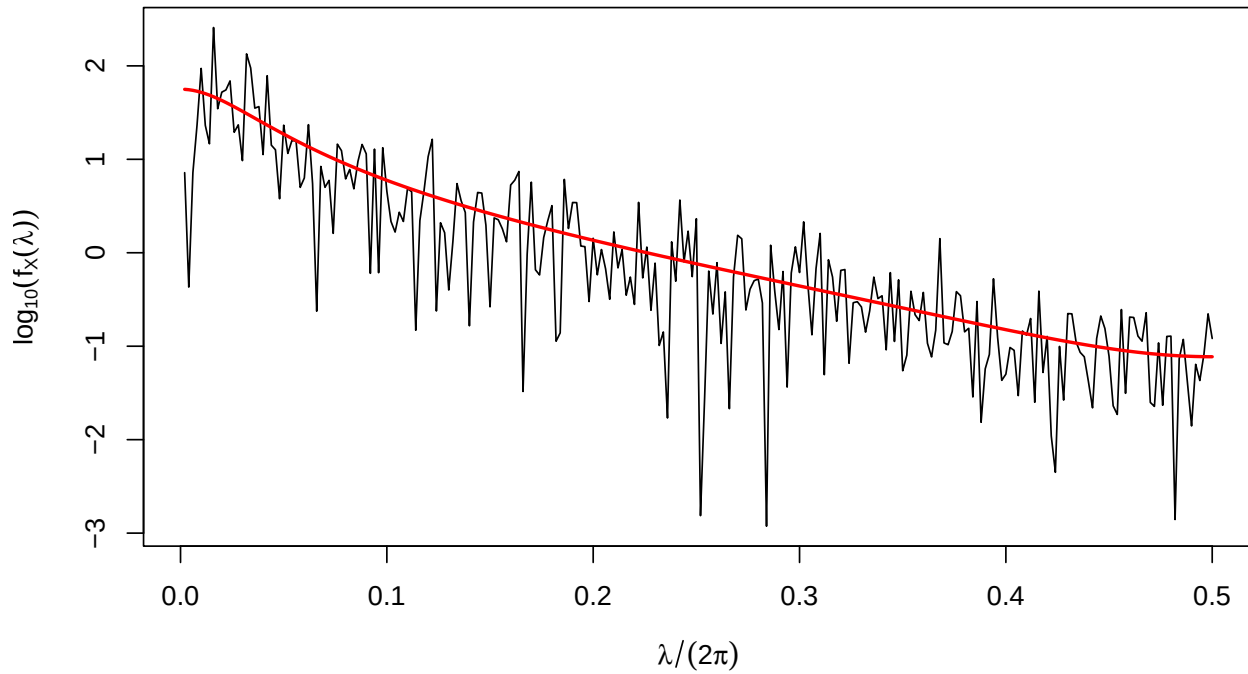
par(mar=c(4,4,2,2))
hist(Vvec,nclass=100, main='Histogram of V', xlab='V', freq=FALSE)
xv<-seq(0,10,by=0.001)
lines(xv,dexp(xv),col='red',lwd=2)
legend(8,0.8,c('Density of Exp(1)'),lwd=2,col='red')
box()

```



The distributional result is maintained reasonably well for a smaller sample size also. For $n = 500$:

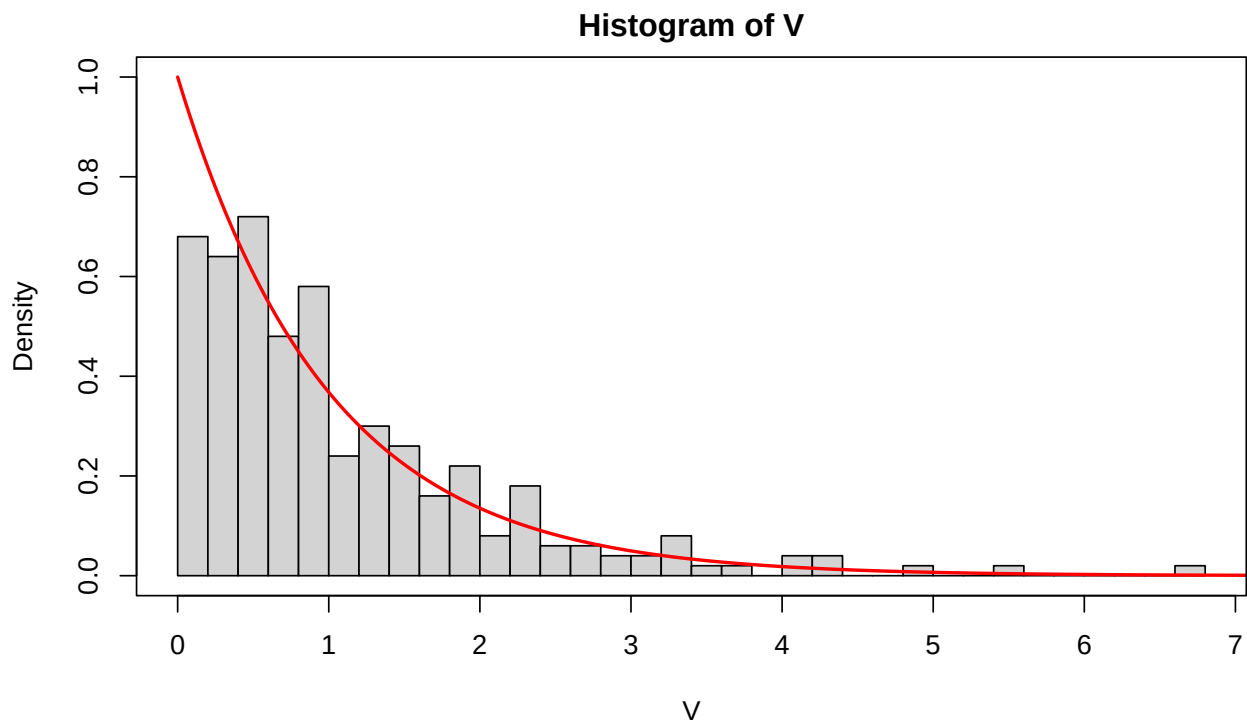
```
set.seed(3);
Xt<-arima.sim(n=500,model=list(ar=0.8,ma=0.5))
sdf.hat<-spectrum(Xt, taper=0,detrend=FALSE,plot=FALSE)$spec
n0<-length(sdf.hat)
lamvec<-c(1:n0)/(2*n0)
sdf.vec<-lamvec*0
hvec<-0:100
for(j in 1:length(lamvec)){
  sdf.vec[j]<-sum(exp(-1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[2:length(hvec)])+acvf.vals[1]+
    sum(exp(1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[2:length(hvec)])
}
par(mar=c(4,4,2,2));plot(lamvec,log10(sdf.hat),type='l',
  ylab=expression(log[10](f[X](lambda))),xlab=expression(lambda/(2*pi)))
lines(lamvec,log10(sdf.vec),col='red',lwd=2)
```



```
Vvec<-as.numeric(sdf.hat/sdf.vec); c(mean(Vvec),var(Vvec))

## [1] 1.084068 1.077051

par(mar=c(4,4,2,2)); hist(Vvec,nclass=30, main='Histogram of V',
  xlab='V', freq=FALSE,ylim=range(0,1))
xv<-seq(0,10,by=0.001); lines(xv,dexp(xv),col='red',lwd=2)
legend(8,0.8,c('Density of Exp(1)'),lwd=2,col='red');box()
```



Likelihood-based estimation: For parametric models such as ARMA models with Gaussian errors, likelihood-based inference can be carried out using a large sample approximation to the time domain likelihood known as the *Whittle likelihood*. For any λ , as $n \rightarrow \infty$

$$\hat{I}_n(\lambda) \xrightarrow{d} V \sim \text{Exponential}(1/(2\pi f_X(\lambda)))$$

If $\lambda_1, \dots, \lambda_n$ are chosen to be the Fourier frequencies, then

$$V_1, \dots, V_n$$

are approximately independent. The Whittle likelihood is then

$$\mathcal{L}(\theta) = \prod_{j=1}^n \frac{1}{2\pi f_X(\omega_j; \theta)} \exp \left\{ -\frac{\hat{I}_n(\omega_j)}{2\pi f_X(\omega_j; \theta)} \right\}.$$

This (asymptotic) likelihood can then be maximized to estimate θ .

```
log.like<-function(xp,xt){
  if(abs(xp[1])>1) return(-1e10) #Do not allow non-stationary solution

  sdf.hat<-spectrum(xt, taper=0,detrend=FALSE,plot=FALSE)$spec
  n0<-length(sdf.hat)
  lamv<-c(1:n0)/(2*n0)
  sdf.vec<-lamv*0
  acvf.vals<-tacvfARMA(phi = xp[1], theta = -xp[2], maxLag = 100)
  hvec<-0:100
  for(j in 1:length(lamv)){
    sdf.vec[j]<-sum(exp(-1i*2*pi*lamv[j]*hvec[-1])*acvf.vals[2:length(hvec)])+
      acvf.vals[1]+
      sum(exp(1i*2*pi*lamv[j]*hvec[-1])*acvf.vals[2:length(hvec)])
  }
  Vvec<-as.numeric(sdf.hat/sdf.vec)
  return(-sum(log(sdf.vec))-sum(Vvec))
}
Xt<-arima.sim(n=5000,model=list(ar=0.8,ma=0.5))
xp0<-c(0,0)
fit.arma<-optim(xp0,fn=log.like,xt=Xt,control=list(fnscale=-1),method='BFGS')
fit.arma

## $par
## [1] 0.7984667 0.5106070
##
## $value
## [1] -2524.737
##
## $counts
## function gradient
##      44      11
##
## $convergence
## [1] 0
##
## $message
## NULL
```

Thus the maximizing values are $\hat{\phi} = 0.798467$ and $\hat{\theta} = 0.510607$.

For smaller sample sizes, the method is still effective: with $n = 50$

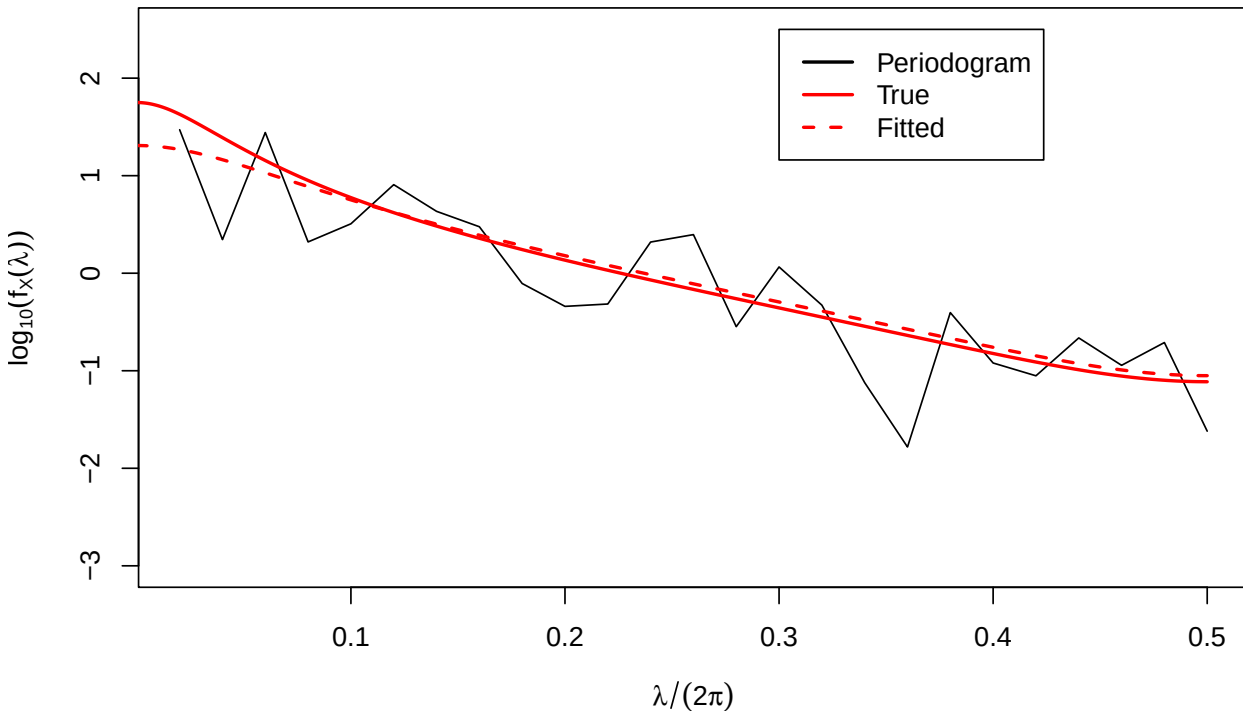
```
Xt<-arima.sim(n=50,model=list(ar=0.8,ma=0.5))
xp0<-c(0,0)
fit.arma<-optim(xp0,fn=log.likelihood,xt=Xt,control=list(fnscale=-1),method='BFGS')
fit.arma$par

## [1] 0.6671425 0.5025452

fit.arma$value

## [1] -26.08411

acvf.vals<-tacvfARMA(phi = 0.8, theta = -0.5, maxLag = 100)
acvf.vals.fit<-tacvfARMA(phi = fit.arma$par[1], theta = -fit.arma$par[2], maxLag = 100)
lamvec<-seq(0,0.5,by=0.001)
sdf.vec<-sdf.vec.fit<-lamvec*0
hvec<-0:100
for(j in 1:length(lamvec)){
  sdf.vec[j]<-sum(exp(-1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[-1])+acvf.vals[1]+
    sum(exp(1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals[-1])
  sdf.vec.fit[j]<-sum(exp(-1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals.fit[-1])+acvf.vals.fit[1]+
    sum(exp(1i*2*pi*lamvec[j]*hvec[-1])*acvf.vals.fit[-1])
}
sdf.hat<-spectrum(Xt, taper=0,detrend=FALSE,plot=FALSE)$spec
n0<-length(sdf.hat)
lamx<-c(1:n0)/(2*n0)
par(mar=c(4,4,2,2));plot(lamx,log10(sdf.hat),type='l',ylim=range(-3,2.5),
  ylab=expression(log[10](f[X](lambda))),xlab=expression(lambda/(2*pi)))
lines(lamvec,log10(sdf.vec),col='red',lwd=2)
lines(lamvec,log10(sdf.vec.fit),col='red',lwd=2,lty=2)
legend(0.3,2.5,c('Periodogram','True','Fitted'),col=c('black','red','red'),lwd=2,lty=c(1,1,2))
```



Long memory models and Fractional Integration: Let $\{X_t\}$ be a stationary process such that for some $\alpha \in (0, 1)$ and constant $c > 0$ such that

$$\lim_{h \rightarrow \infty} h^\alpha \rho_X(h) = c$$

That is, the decay rate is *polynomial* in h , and therefore relatively slow. $\{X_t\}$ is a *long memory* or *long range dependent* process.

Recall the ARIMA(p, d, q) model, for $d = 1, 2, \dots$,

$$(1 - B)^d \Phi(B) X_t = \Theta(B) Z_t$$

Now suppose that $-0.5 < d < 0.5$ is a *non-integer* value: note that if d lies in this range, then $-d$ also lies in this range. If $|z| < 1$, we have by the general binomial expansion, for $\alpha > -1$,

$$(1 - z)^\alpha = \sum_{j=0}^{\infty} \frac{\Gamma(j - \alpha)}{\Gamma(j + 1)\Gamma(-\alpha)} z^j$$

where the coefficients are given by the expansion

$$\frac{\Gamma(j - \alpha)}{\Gamma(j + 1)\Gamma(-\alpha)} = \frac{(j - 1 - \alpha)(j - 2 - \alpha) \cdots (-\alpha)}{j!} = \prod_{k=1}^j \left(\frac{k - 1 - \alpha}{k} \right)$$

We apply this formula taking $\alpha = d$ for $-0.5 < d < 0.5$.

Fractional White Noise: Let $\{Z_t\} \sim \text{WN}(0, \sigma_Z^2)$, and $W_t = (1 - B)^{-d} Z_t$, so that in the original model we have

$$\Phi(B) X_t = (1 - B)^{-d} \Theta(B) Z_t = \Theta(B) (1 - B)^{-d} Z_t = \Theta(B) W_t$$

gives an ARMA(p, q) model driven by $\{W_t\}$. $\{W_t\}$ is *fractionally integrated white noise*. In this representation, we have that for $j = 1, 2, \dots$, the linear process coefficients are

$$\psi_j = \frac{\Gamma(j + d)}{\Gamma(j + 1)\Gamma(d)}$$

$\{W_t\}$ is a stationary (linear) process. We have that

$$\gamma_W(h) = \sigma_Z^2 \sum_{j=0}^{\infty} \psi_j \psi_{j+h} = \sigma_Z^2 \sum_{j=0}^{\infty} \frac{\Gamma(j + d)}{\Gamma(j + 1)\Gamma(d)} \frac{\Gamma(j + h + d)}{\Gamma(j + h + 1)\Gamma(d)}$$

which can be shown (see Brockwell and Davis, *Time Series: Theory and Methods*, Chapter 13) to yield

$$\gamma_W(0) = \frac{\sigma_Z^2 \Gamma(1 - 2d)}{\{\Gamma(1 - d)\}^2}$$

and

$$\rho_W(h) = \frac{\Gamma(h + d)\Gamma(1 - d)}{\Gamma(h - d + 1)\Gamma(d)} = \prod_{k=1}^h \left(\frac{k - 1 + d}{k - d} \right)$$

We compute the infinite sum in the linear process representation for $d = 0.3$ by truncating at $J = 100000$ terms.

```
d<-0.3
J<-100000; jvec<-c(1:J)
psivec<-exp(lgamma(jvec+d)-lgamma(jvec+1)-lgamma(d))
psivec[1:10]; psivec[J]

## [1] 0.30000000 0.19500000 0.14950000 0.12333750 0.10607025 0.09369539
## [7] 0.08432585 0.07694734 0.07096254 0.06599517
## [1] 0.0001057062
```

```

H<-50; hvec<-c(1:H)
sigZsq<-2
gamma0<-sigZsq*gamma(1-2*d)/(gamma(1-d)^2)
gammah<-gamma(hvec+d)*gamma(1-d)/(gamma(hvec-d+1)*gamma(d))*gamma0
c(gamma0,gammah)

## [1] 2.6329121 1.1283909 0.8628872 0.7350520 0.6555869 0.5997923 0.5577016
## [8] 0.5244060 0.4971641 0.4743060 0.4547470 0.4377471 0.4227814 0.4094655
## [15] 0.3975103 0.3866937 0.3768416 0.3678154 0.3595032 0.3518133 0.3446699
## [22] 0.3380096 0.3317790 0.3259327 0.3204317 0.3152425 0.3103360 0.3056868
## [29] 0.3012726 0.2970736 0.2930726 0.2892541 0.2856042 0.2821106 0.2787621
## [36] 0.2755487 0.2724613 0.2694917 0.2666324 0.2638765 0.2612178 0.2586505
## [43] 0.2561695 0.2537698 0.2514469 0.2491968 0.2470157 0.2448999 0.2428463
## [50] 0.2408516 0.2389132

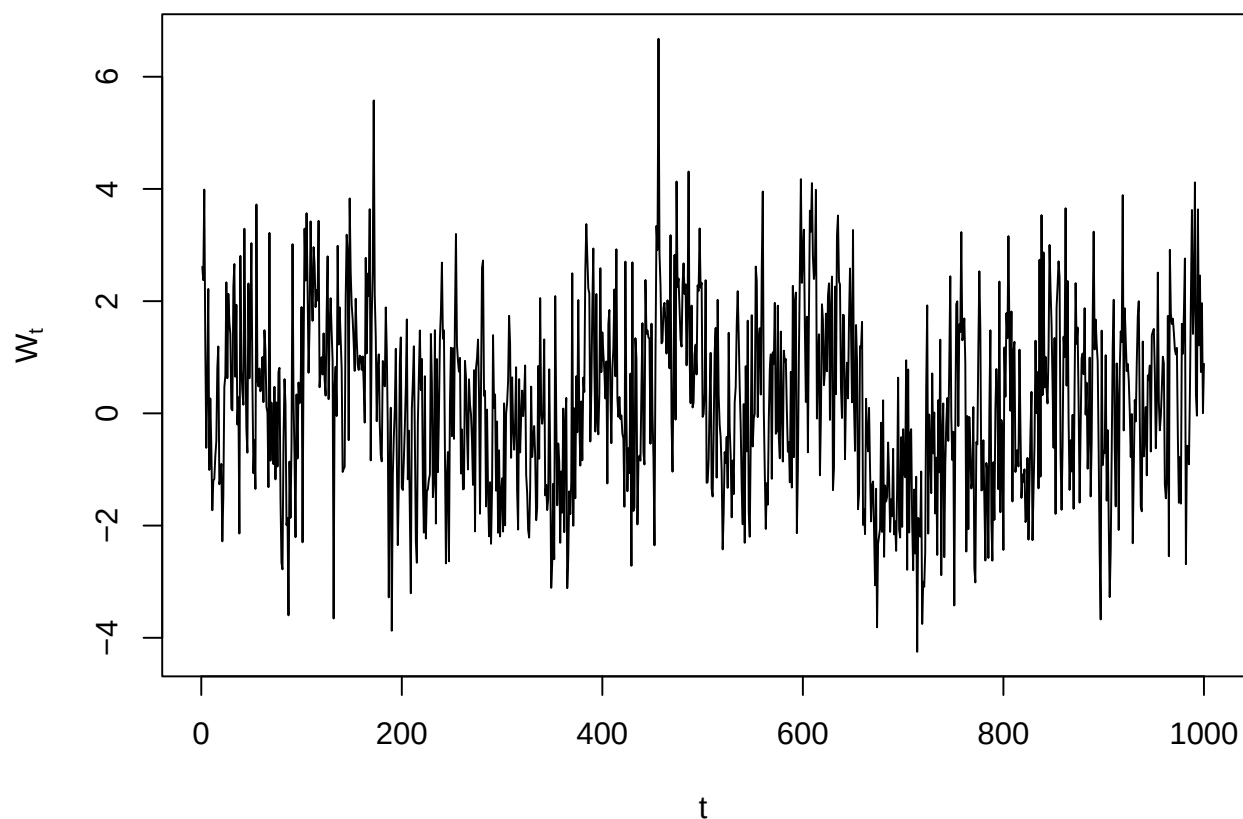
set.seed(3); n<-1000
Zt<-rnorm(n+J,0,sqrt(sigZsq))
Wt<-rep(0,n)
for(i in 1:n){
  Wt[i]<-Zt[J+i]+sum(psivec*Zt[J+i-jvec])
}
var(Wt)

## [1] 2.622585

par(mar=c(4,4,2,2))
plot(Wt,type='l',xlab='t', ylab=expression(W[t]))
title('Simulated realization of fractional white noise')

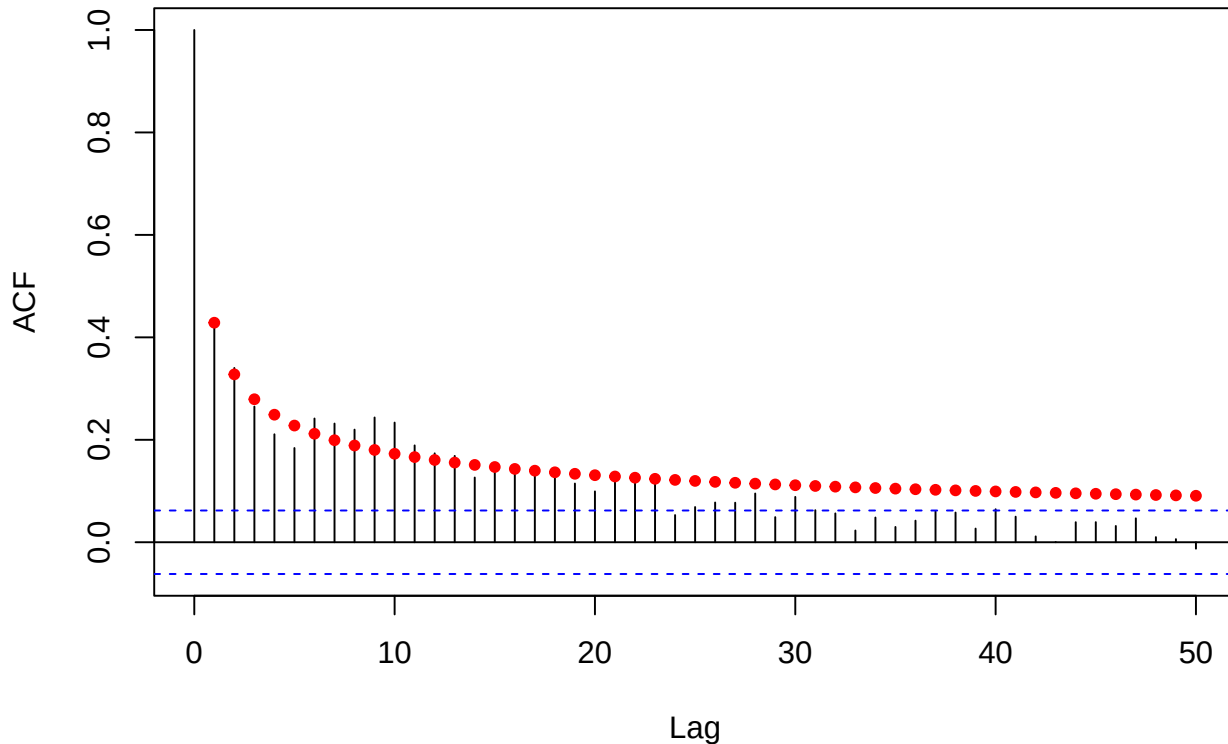
```

Simulated realization of fractional white noise



```
par(mar=c(4,4,4,2))
acf(Wt,lag.max=50,main='ACF of fractional white noise')
points(hvec,gammah/gamma0,pch=19,cex=0.7,col='red')
```

ACF of fractional white noise



The ACVF of $\{X_t\}$ can be computed using the usual approach

$$\gamma_X(h) = \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} \psi_j \psi_k \gamma_W(h+j-k)$$

where $\{\psi_j\}$ are the linear process coefficients corresponding to the ARMA process with polynomials $\Phi(B)$ and $\Theta(B)$. We have by the usual result that

$$\gamma_W(h) = \sigma_Z^2 \sum_{j=0}^{\infty} \psi_j \psi_{j+h}$$

```
gammah.lp<-hvec*0
gamma0.lp<-sigZsq*(1+sum(psivec^2))
for(h in 1:H){
  ivec<-jvec+h
  ivec<-ivec[ivec <= J]
  gammah.lp[h]<-sigZsq*(psivec[h]+sum(psivec[ivec-h]*psivec[ivec]))
}
gamma0.lp

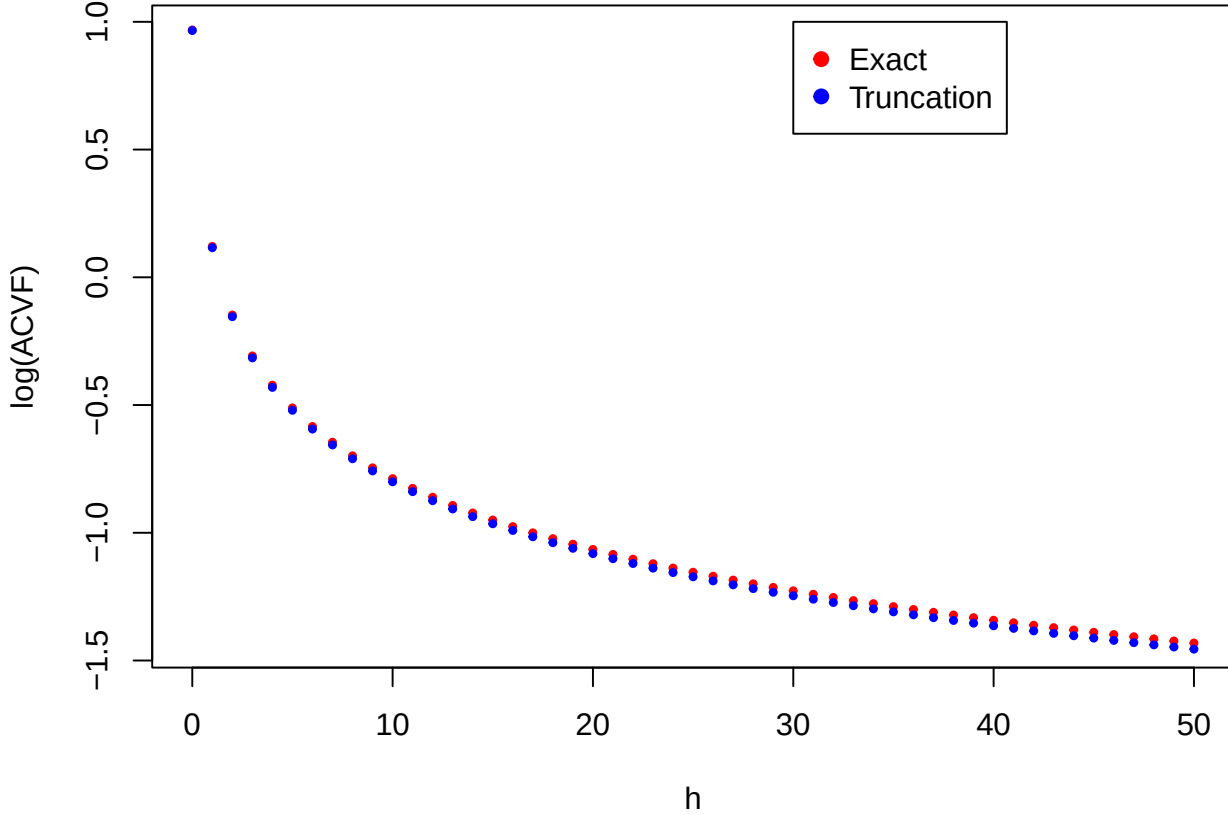
## [1] 2.627325

gammah.lp[1:10]

## [1] 1.1228040 0.8573002 0.7294651 0.6500000 0.5942054 0.5521147 0.5188190
## [8] 0.4915771 0.4687190 0.4491600
```

We can compare the computed ACVF values using the exact formula and using the truncated linear process summation. We compare on the log scale for ease of comparison.

```
par(mar=c(4,4,2,2))
plot(c(0,hvec),log(c(gamma0,gammah)),pch=19,cex=0.5,col='red',xlab='h',ylab='log(ACVF)')
points(c(0,hvec),log(c(gamma0.lp,gammah.lp)),pch=19,cex=0.5,col='blue')
legend(30,1.0,c('Exact','Truncation'),col=c('red','blue'),pch=19)
```



These results show the limitations of studying long memory processes in the time domain:

- In the linear process representation, even out to $J = 100000$ terms, the coefficients are relatively large; for example, in the calculations above

$$\psi_{100000} = 0.000106$$

- If we study the process using the ACVF and the sample ACVF, then we need an extremely long series to get accurate estimates. Even with $n = 1000$ as in the simulation, the sample-based version is not close to the true ACVF.

However, in the frequency domain, analysis is a little more straightforward. By direct calculation using the linear process representation and the general result

$$f_X(\lambda) = \frac{\sigma_Z^2}{2\pi} |\Psi(e^{-i\lambda})|^2$$

we have here that

$$f_W(\lambda) = \frac{\sigma_Z^2}{2\pi} \frac{1}{|(1 - e^{-i\lambda})^d|^2} = \frac{\sigma_Z^2}{2\pi} \frac{1}{|1 - e^{-i\lambda}|^{2d}}$$

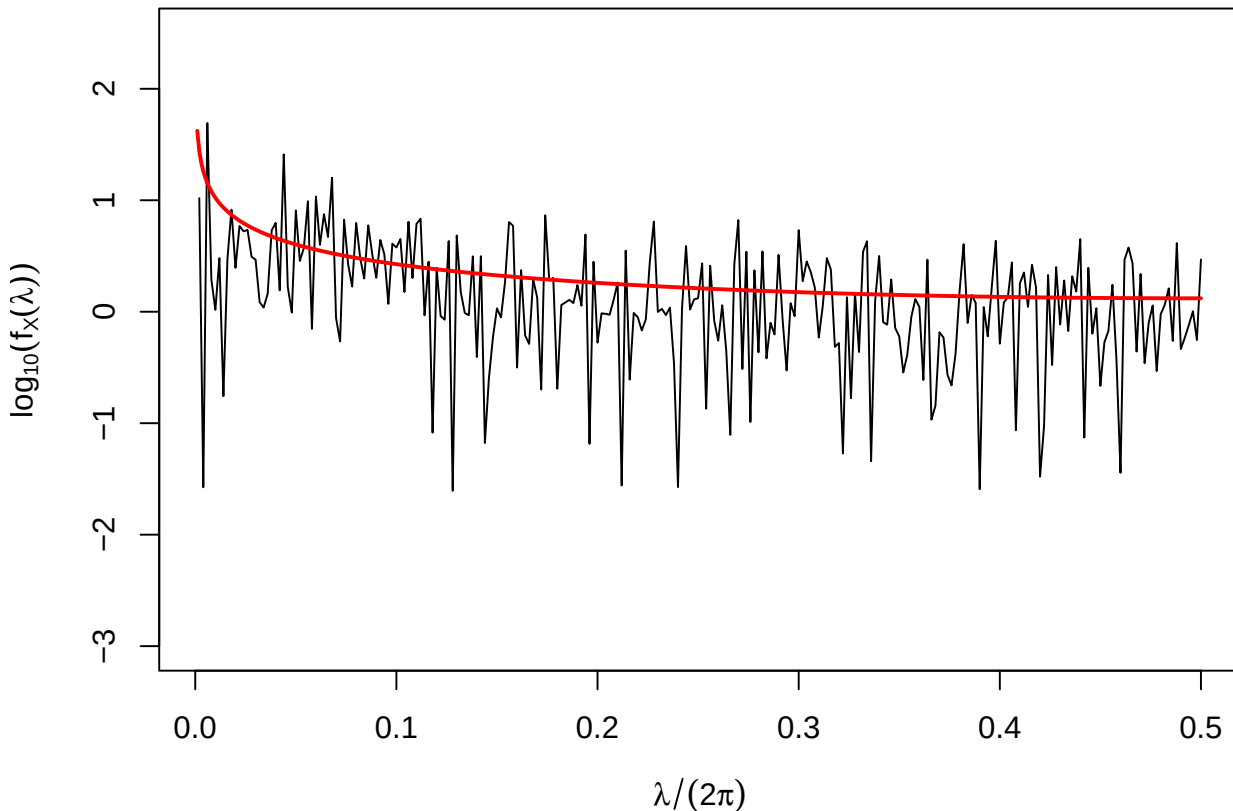
```

n<-500
Zt<-rnorm(n+J,0,sqrt(sigZsq))
Wt<-rep(0,n)
for(i in 1:n){
  Wt[i]<-Zt[J+i]+sum(psivec*Zt[J+i-jvec])
}
var(Wt)

## [1] 2.333434

sdf.hat<-spectrum(Wt, taper=0,detrend=FALSE,plot=FALSE)$spec
lamvec<-seq(0,0.5,by=0.001)
sdf.vec<-sigZsq*Mod(1-exp(-1i*2*pi*lamvec))^{-2*d}
n0<-length(sdf.hat)
lamx<-c(1:n0)/(2*n0)
par(mar=c(4,4,2,2));plot(lamx,log10(sdf.hat),type='l',ylim=range(-3,2.5),
  ylab=expression(log[10](f[X](lambda))),xlab=expression(lambda/(2*pi)))
lines(lamvec,log10(sdf.vec),col='red',lwd=2)

```



We can again estimate parameters d and σ_Z^2 using the Whittle likelihood.

```

log.like.fwn<-function(xp,xt){
  if(abs(xp[1])>0.5) return(-1e10) #Do not allow non-stationary solution
  dv<-xp[1]
  if(xp[2]<=0) return(-1e10)
  sdf.hat<-spectrum(xt, taper=0,detrend=FALSE,plot=FALSE)$spec
  n0<-length(sdf.hat)
  lamv<-c(1:n0)/(2*n0)
  sdf.vec<-xp[2]*Mod(1-exp(-1i*2*pi*lamv))^{-2*dv}
}

```

```

Vvec<-as.numeric(sdf.hat/sdf.vec)
return(-sum(log(sdf.vec))-sum(Vvec))
}
xp0<-c(0.3,2)
fit.fwn<-optim(xp0,fn=log.like.fwn,xt=Wt,control=list(fnscale=-1),method='Nelder-Mead')
fit.fwn$par

## [1] 0.3036485 1.9985952

fit.fwn$value

## [1] -421.0368

```

Hence the estimate of d is 0.303649, and the estimate of σ_Z^2 is 1.998595.

Limiting behaviour: If $\{X_t\} \sim \text{ARIMA}(p, d, q)$ with $-1/2 < d < 1/2$, then

$$f_X(\lambda) = \frac{\sigma_Z^2}{2\pi} \frac{|\Theta(e^{-i\lambda})|^2}{|\Phi(e^{-i\lambda})|^2} \frac{1}{|1 - e^{-i\lambda}|^{2d}}$$

where, if $p = q = 0$,

$$f_X(\lambda) = f_W(\lambda) = \frac{\sigma_Z^2}{2\pi} \frac{1}{|1 - e^{-i\lambda}|^{2d}}$$

In this case note that, by a Taylor-type expansion

$$|1 - e^{-i\lambda}|^{2d} = \left| 1 - \left(1 - i\lambda + \frac{\lambda^2}{2} + o(\lambda^2) \right) \right|^{2d} = |\lambda + o(\lambda)|^{2d}$$

as $\lambda \rightarrow 0$. Thus for λ near zero

$$f_X(\lambda) \approx \frac{\sigma_Z^2}{2\pi|\lambda|^{2d}}$$

and the SDF is unbounded at zero. If $\{W_t\} \sim \text{ARIMA}(0, d, 0)$, that is,

$$W_t = (1 - B)^{-d} Z_t$$

then using the binomial expansion method

- MA(∞) representation

$$W_t = \sum_{j=0}^{\infty} \psi_j Z_{t-j} \quad \psi_j = \frac{\Gamma(j+d)}{\Gamma(j+1)\Gamma(d)}$$

- AR(∞) representation

$$Z_t = \sum_{j=0}^{\infty} \pi_j W_{t-j} \quad \pi_j = \frac{\Gamma(j-d)}{\Gamma(j+1)\Gamma(-d)}$$

For large j

- MA(∞) representation

$$\psi_j \approx \frac{j^{d-1}}{\Gamma(d)}$$

- AR(∞) representation

$$\pi_j \approx \frac{j^{-d-1}}{\Gamma(-d)}$$

If $\{X_t\} \sim \text{ARIMA}(p, d, q)$, write

$$(1 - B)^d \Phi(B) X_t = \Theta(B) Z_t$$

and suppose $0 < d < 1/2$. Then for large h

$$\gamma_X(h) \approx C|h|^{2d-1} \propto \frac{1}{|h|^{1-2d}}$$

The implication is that for $h > 0$ large enough,

$$\log \gamma_X(h) \approx \beta_0 - (1 - 2d) \log h$$

so $\log \gamma_X(h)$ is approximately linear in $\log h$, with slope $2d - 1$. Similarly, for the $p = q = 0$ case

$$\frac{\hat{I}_j(\omega_j)}{2\pi f_X(\lambda)} = \frac{\hat{I}_j(\omega_j) |1 - e^{-i\lambda}|^{2d}}{\sigma_Z^2} = U_j \sim \text{Exponential}(1)$$

say. Therefore

$$\log \hat{I}_j(\omega_j) + 2d \log |1 - e^{-i\lambda}| - \log \sigma_Z^2 = \log U_j$$

and hence

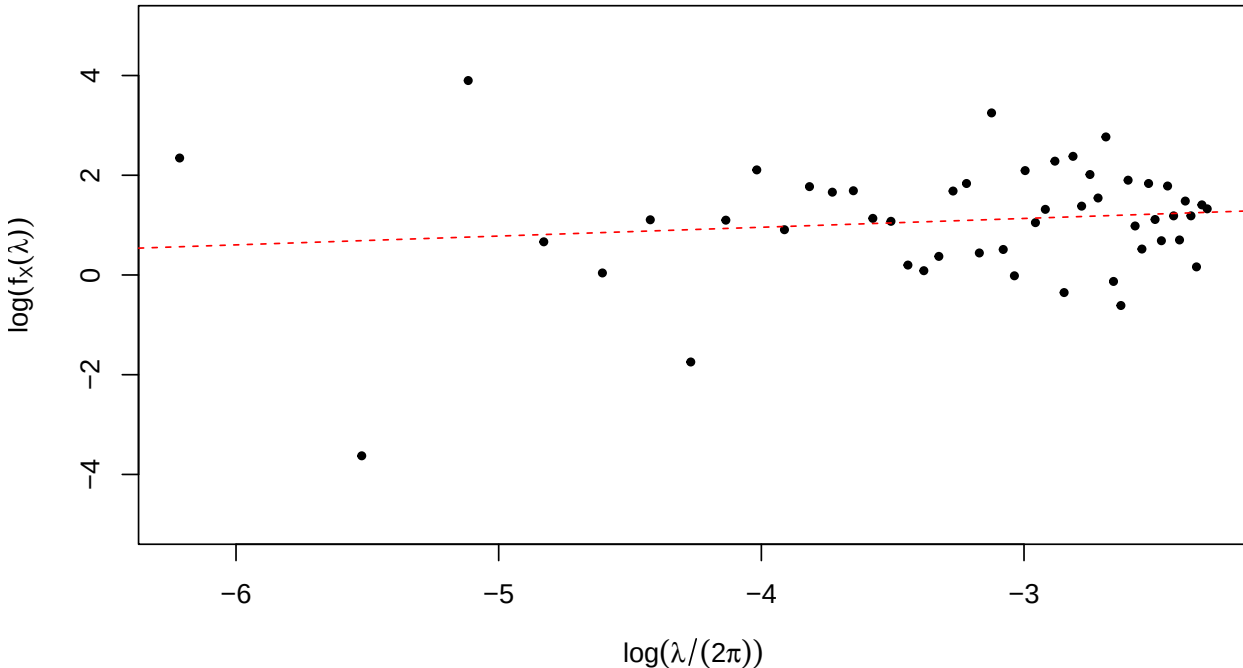
$$\log \hat{I}_j(\omega_j) = \log \sigma_Z^2 - 2d \log |1 - e^{-i\lambda}| + \log U_j$$

Making the approximation $|1 - e^{-i\lambda}| \approx |\lambda|$, we have that for λ near zero

$$\log \hat{I}_j(\omega_j) = \log \sigma_Z^2 - 2d \log |\lambda| + \log U_j$$

so plotting the log periodogram against $\log \lambda$ should yield a straight line with slope $-2d$.

```
n0<-length(sdf.hat)
lamx<-c(1:n0)/(2*n0)
par(mar=c(4,4,2,2));plot(log(lamx[1:50]),log(sdf.hat[1:50]),ylim=range(-5,5),
    pch=19,cex=0.6,ylab=expression(log(f[X](lambda))),xlab=expression(log(lambda/(2*pi))))
x<-log(lamx[1:50])
y<-log(sdf.hat[1:50])
abline(lm(y~x)$coef,col='red',lty=2)
```



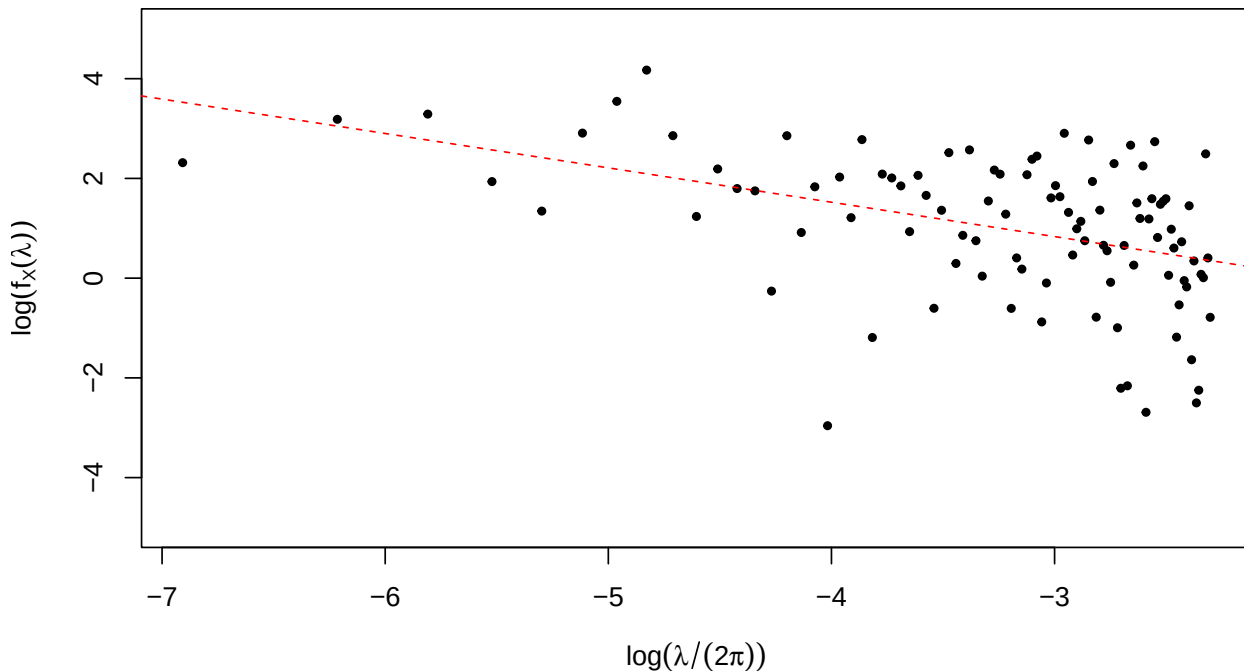
```
summary(lm(y~x))$coef

##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept)  1.6625826   0.6524351  2.5482727  0.01408073
## x            0.1764598   0.1940434  0.9093832  0.36769168
```

For a larger sample size

```
n<-1000
Zt<-rnorm(n+J,0,sqrt(sigZsq))
Wt<-rep(0,n)
for(i in 1:n){
  Wt[i]<-Zt[J+i]+sum(psivec*Zt[J+i-jvec])
}
sdf.hat<-spectrum(Wt, taper=0,detrend=FALSE,plot=FALSE)$spec
n0<-length(sdf.hat)
lamx<-c(1:n0)/(2*n0)

par(mar=c(4,4,2,2));plot(log(lamx[1:100]),log(sdf.hat[1:100]),ylim=range(-5,5),
  pch=19,cex=0.6,ylab=expression(log(f_X(lambda))),xlab=expression(log(lambda/(2*pi))))
x<-log(lamx[1:100])
y<-log(sdf.hat[1:100])
abline(lm(y~x)$coef,col='red',lty=2)
```



```
summary(lm(y~x))$coef

##              Estimate Std. Error  t value    Pr(>|t|)
## (Intercept) -1.2345691   0.4863579 -2.538396  0.01271102654
## x           -0.6892771   0.1431211 -4.816040  0.00000533306
```