Time Series: A First Course with Bootstrap Starter

Contents

Lesson 7-1: Herglotz Theorem

We further study the relationship between autocovariance and spectral density.

Example 7.1.2. Absolutely Summable Autocovariance.

- Suppose an ACVF $\{\gamma(k)\}$ is absolutely summable, so that the spectral density f exists.
- Define its anti-derivative, for $\lambda \in [-\pi, \pi]$:

$$F(\lambda) = \int_{-\pi}^{\lambda} f(\omega) d\omega.$$

- So $F(-\pi) = 0$ and $F(\pi) = 2\pi\gamma(0)$.
- F is a bounded, non-decreasing, absolutely-continuous function. It is called the *spectral distribution*.
- With $dF(\lambda) = f(\lambda)d\lambda$, we can write

$$\gamma(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda k} dF(\lambda).$$

- This can be interpreted as a "Stieltjes integral" when F is not differentiable.

Example 7.1.4. Stochastic Cosine.

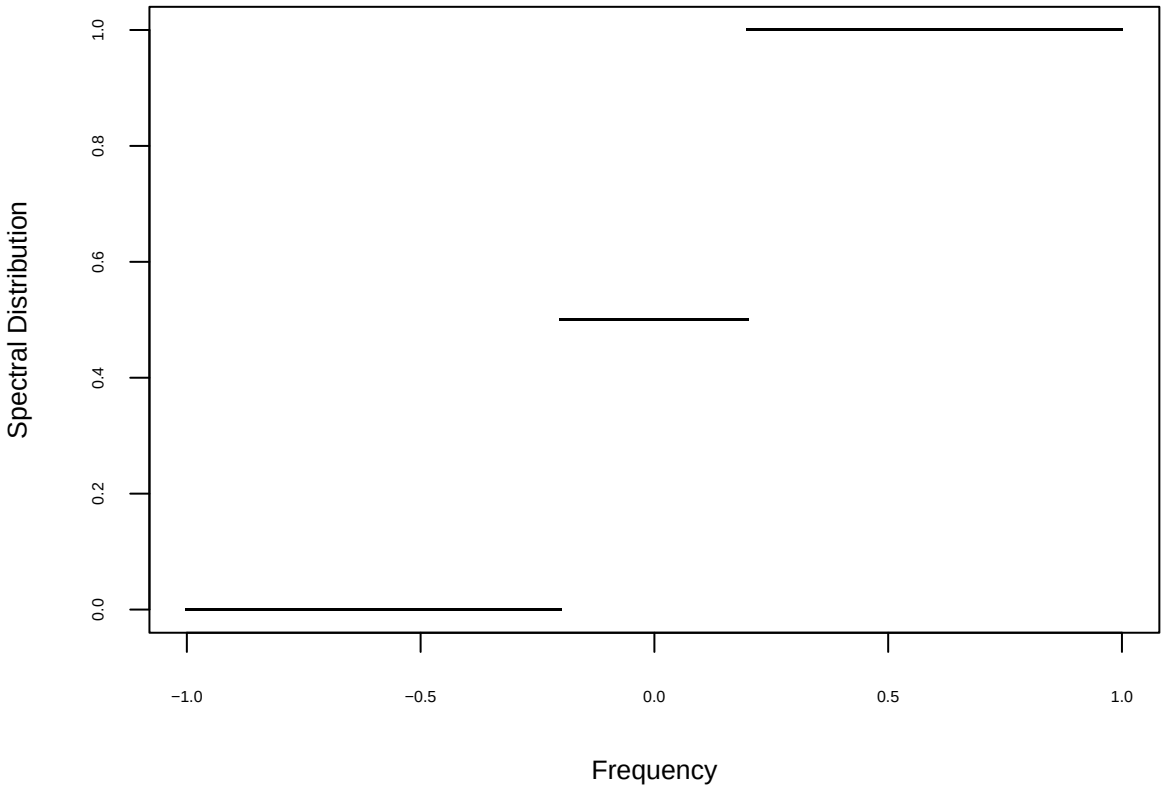
- Consider $X_t = A\cos(\vartheta t + \Phi)$, where Φ is uniformly distributed on $[0, 2\pi]$, and A has mean zero and variance σ^2. Also $\vartheta \in [0, \pi]$.
- The ACVF is $\gamma(k) = \sigma^2 \cos(\vartheta k)/2$, which is not summable, so the spectral density does not exist.
- However, we can choose F such that the above relation still holds:

$$F(\lambda) = \frac{\sigma^2 \pi}{2} \left(1_{\{\lambda \in [-\vartheta, \pi]\}} + 1_{\{\lambda \in [\vartheta, \pi]\}}\right).$$

```
theta <- pi/5
sig2 <- 1/pi
grid <- 1000
lambda <- pi*rep(0,grid+1)/grid
lambda <- c(rev(lambda),lambda[-1])
j <- grid*theta/pi
F.distr <- 0*cos(0*lambda)
F.distr[(grid+1-j):(grid+j)] <- .5*pi*sig2
F.distr[(grid+1+j):(2*grid+1)] <- pi*sig2
par(mar=c(4,4,2,2)+0.1,cex.lab=.8)
plot(ts(F.distr,start=-1,frequency=grid),xlab="Frequency",
    ylab="Spectral Distribution",yaxt="n",xaxt="n",type="points",pch=".")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): Le type de graphique
## 'points' sera tronqué au premier caractère
```

```
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
```



Definition 7.1.7.

- A function $F(\lambda)$ for $\lambda \in [-\pi, \pi]$ that is bounded, non-decreasing, and right continuous, and satisfies $F^-(\lambda) = F(\pi) - F(-\lambda)$ is called a **spectral distribution function.**
- $F^-(\lambda)$ is the left-hand limit.

Theorem 7.1.8. Herglotz Theorem

A sequence $\{\gamma(k)\}$ is an ACVF (non-negative definite with even symmetry) if and only if it can be written as

$$\gamma(k) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda k} dF(\lambda)$$

for some spectral distibution function F.

Fact 7.1.9. Spectral Decomposition

- We can decompose a spectral distribution function F into a singular portion F_s and a absolutely continuous portion F_c:
$$F = F_s + F_c.$$
- The absolutely continuous portion is differentiable almost everywhere, with derivative called the spectral density.
- The singular part corresponds to a step function.

Exercise 7.5. Spectral Distribution of an ARMA Process.

- We write code to compute the spectral distribution F of an ARMA process.
- We begin with the code for an ARMA spectral density.

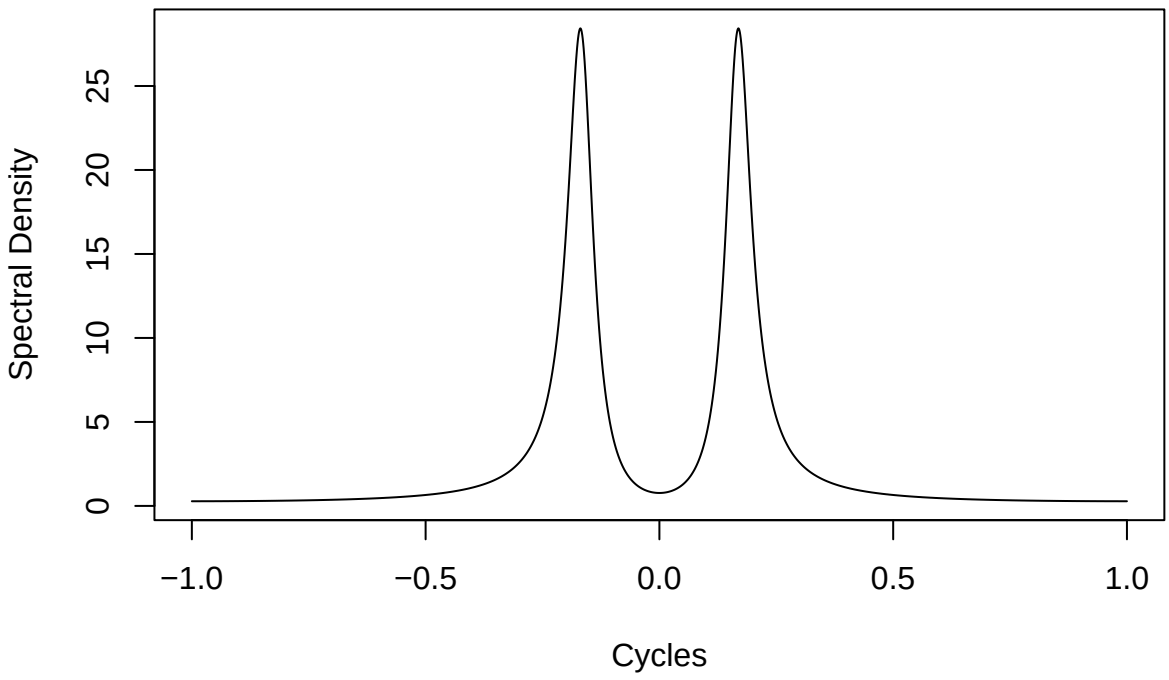
```
armapq.spec <- function(ar.coef,ma.coef,sigma,mesh)
{
    p <- length(ar.coef)
    q <- length(ma.coef)
    lambda <- pi*seq(0,mesh)/mesh
    spec.ar <- rep(1,mesh+1)
    if(p > 0)
    {
        for(k in 1:p)
        {
            spec.ar <- spec.ar - ar.coef[k]*exp(-1i*lambda*k)
        }
    }
    spec.ma <- rep(1,mesh+1)
    if(q > 0)
    {
        for(k in 1:q)
        {
            spec.ma <- spec.ma + ma.coef[k]*exp(-1i*lambda*k)
        }
    }
    spec <- sigma^2*Mod(spec.ma)^2/Mod(spec.ar)^2
    return(spec)
}
```

- Then we get the spectral distribution by numerical integration, via trapezoidal rule.

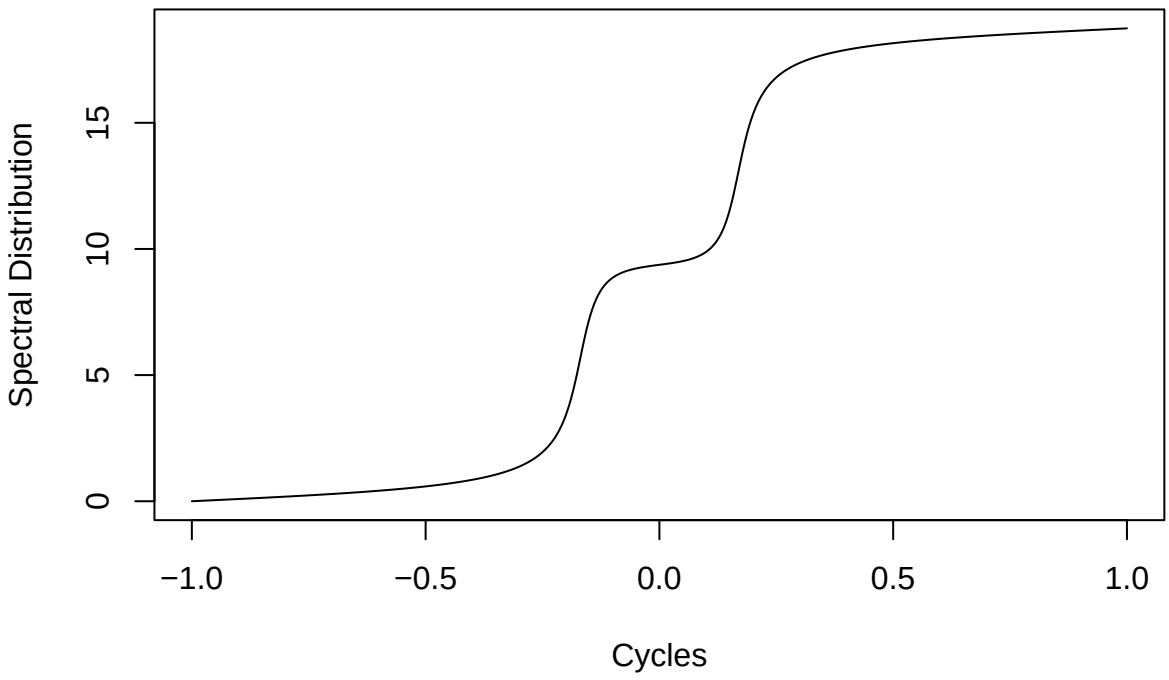
```
armapq.distr <- function(ar.coef,ma.coef,sigma,mesh)
{

    f.spec <- armapq.spec(ar.coef,ma.coef,sigma,mesh)
    f.spec <- c(rev(f.spec),f.spec[-1])
    f.distr <- rep(0,2*mesh+1)
    for(i in 1:(2*mesh))
    {
        f.distr[i+1] <- f.distr[i] + (f.spec[i]+f.spec[i+1])/2
    }
    f.distr <- pi*f.distr/mesh
    return(f.distr)
}
```

- Then we apply this in the case of the cycle ARMA(2,1). We plot the spectral density and the spectral distribution.

```
mesh <- 1000
rho <- .9
omega <- pi/6
ar.coef <- c(2*rho*cos(omega),-1*rho^2)
ma.coef <- -1*rho*cos(omega)
spec <- armapq.spec(ar.coef,ma.coef,1,mesh)
plot(ts(c(rev(spec),spec[-1]),start=-1,frequency=mesh),xlab="Cycles",ylab="Spectral Density",main="")
```




```
spec <- armapq.distr(ar.coef,ma.coef,1,mesh)
plot(ts(spec,start=-1,frequency=mesh),xlab="Cycles",ylab="Spectral Distribution",main="")
```



Lesson 7-2: Discrete Fourier Transform

We introduce the data analysis tool of Discrete Fourier Transform.

Definition 7.2.3.

Given a sample $X_1, \ldots, X_n$, the **Discrete Fourier Transform** (DFT) at the Fourier frequency $\lambda_l = 2\pi l/n$ (for $[n/2] - n + 1 \leq l \leq [n/2]$) is

$$\widetilde{X}(\lambda_l) = n^{-1/2} \sum_{t=1}^{n} X_t e^{-i\lambda_l t}.$$

Definition 7.2.4.

- The **periodogram** $I(\lambda)$ is a non-negative function of $\lambda \in [-\pi, \pi]$, constructed from the sample $X_1, \ldots, X_n$:

$$I(\lambda) = n^{-1} \left| \sum_{t=1}^{n} (X_t - \overline{X}) e^{-i\lambda t} \right|^2.$$

- So $I(0) = 0$. Sometimes we consider an "uncentered" periodogram, where there is no centering by the sample mean. We denote this by $\widetilde{I}(\lambda)$, and

$$\widetilde{I}(\lambda_l) = |\widetilde{X}(\lambda_l)|^2.$$

- The periodogram is an empirical version of the spectral density, and shares certain properties. Higher values correspond to cyclical effects in the data.

Proposition 7.2.7.

- Let the vector of DFTs be denoted $\underline{\widetilde{X}}$, which has components $\widetilde{X}(\lambda_l)$.
- This is a linear function of the sample vector $\underline{X}$:

$$\underline{\widetilde{X}} = Q^* \, \underline{X},$$

  where Q was defined in Definition 6.4.3.
- Since Q is unitary, $Q^{-1} = Q^*$, so we can recover the data from the DFT vector via

$$\underline{X} = Q \, \underline{\widetilde{X}}.$$

Exercise 7.14. DFT of an AR(1).

- We simulate an AR(1).

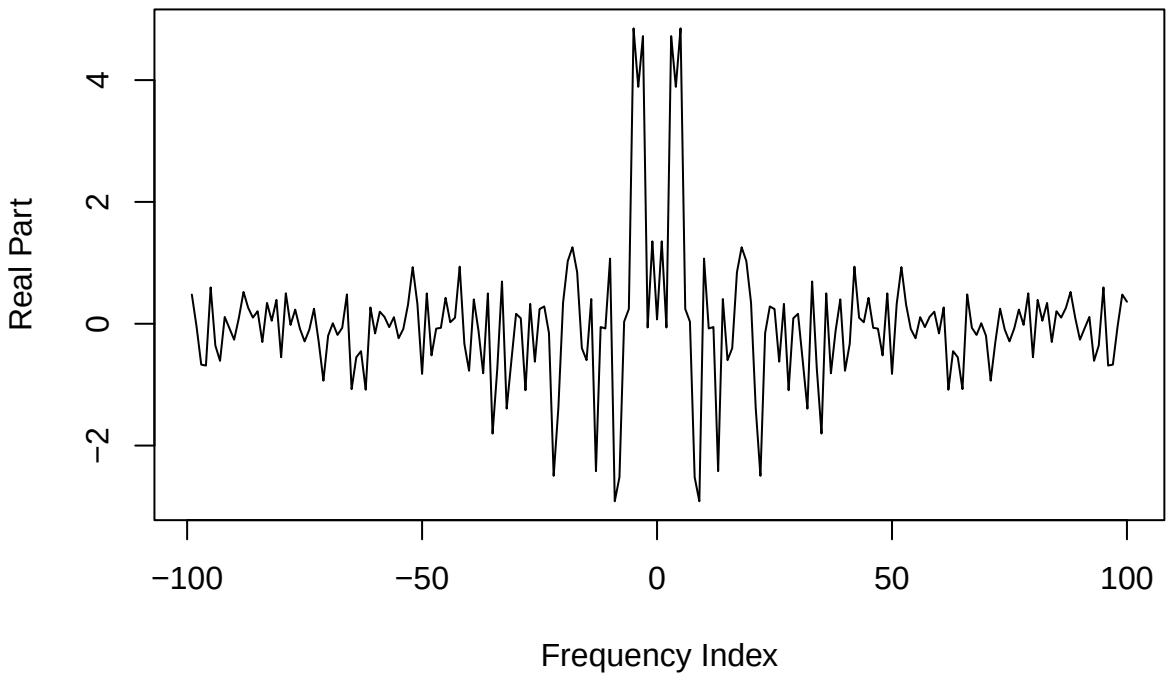
```
arp.sim <- function(n,burn,ar.coefs,innovar)
{
    p <- length(ar.coefs)
    z <- rnorm(n+burn+p,sd=sqrt(innovar))
    x <- z[1:p]
    for(t in (p+1):(p+n+burn))
    {
        next.x <- sum(ar.coefs*x[(t-1):(t-p)]) + z[t]
        x <- c(x,next.x)
    }
    x <- x[(p+burn+1):(p+burn+n)]
    return(x)
}

phi <- .8
innovar <- 1
n <- 200
x.sim <- arp.sim(n,500,phi,innovar)
```

- We compute the DFT. We begin with code to compute Q.

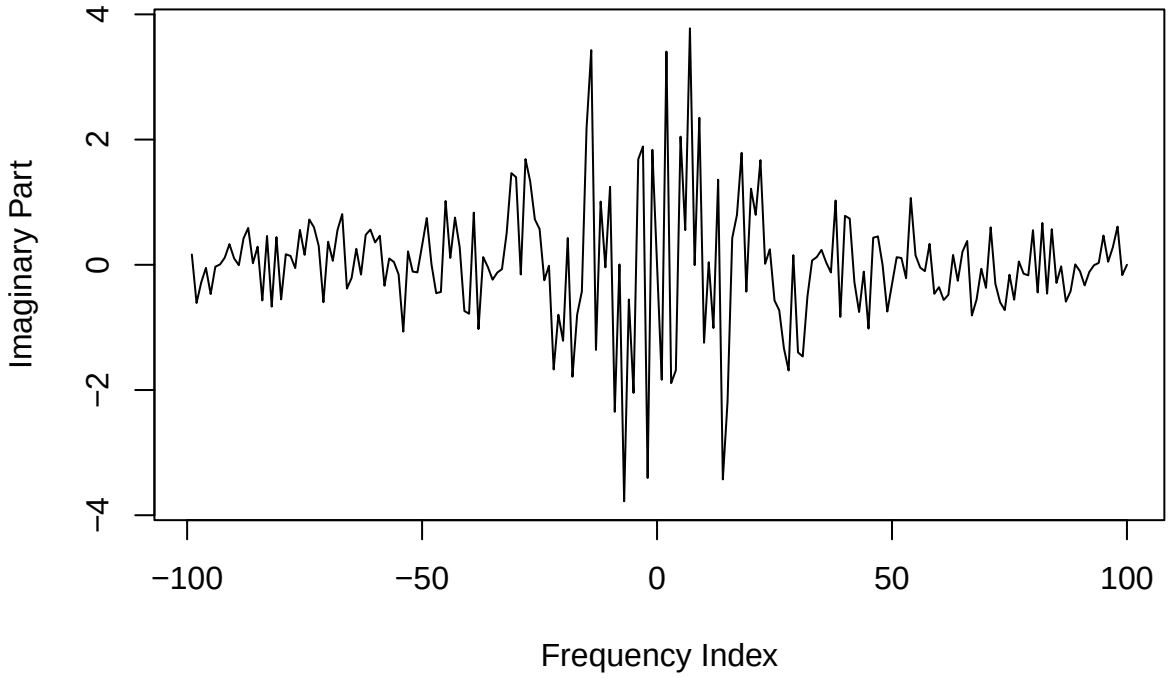
```
get.qmat <- function(mesh)
{
    mesh2 <- floor(mesh/2)
    inds <- seq(mesh2-mesh+1,mesh2)
    Q.mat <- exp(1i*2*pi*mesh^{-1}*t(t(seq(1,mesh)) %x% inds))*mesh^{-1/2}
    return(Q.mat)
}
Q.mat <- get.qmat(n)
```

- Then we use Proposition 7.2.7 to get the DFT vector. We plot the real part, imaginary part, and the modulus.

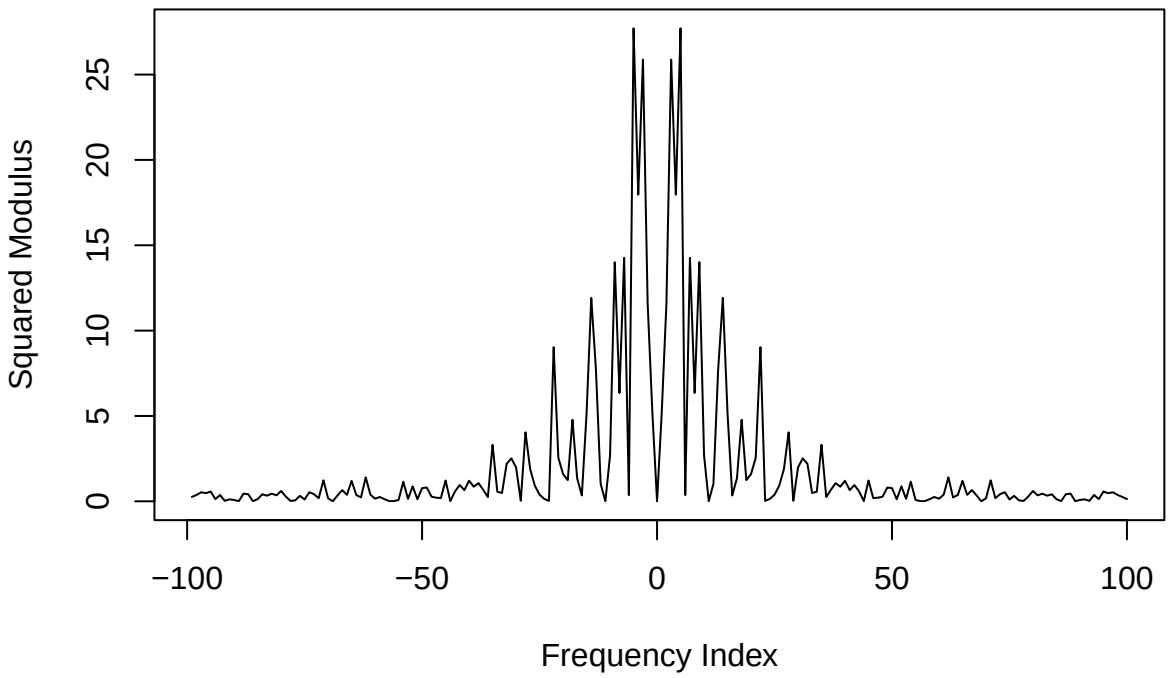
```
x.dft <- Conj(t(Q.mat)) %*% x.sim
plot(ts(Re(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Real Part")
```




```
plot(ts(Im(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Imaginary Part")
```




```
plot(ts(Mod(x.dft)^2,start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Squared Modulus")
```



Corollary 7.2.9. Decorrelation Property of the DFT.

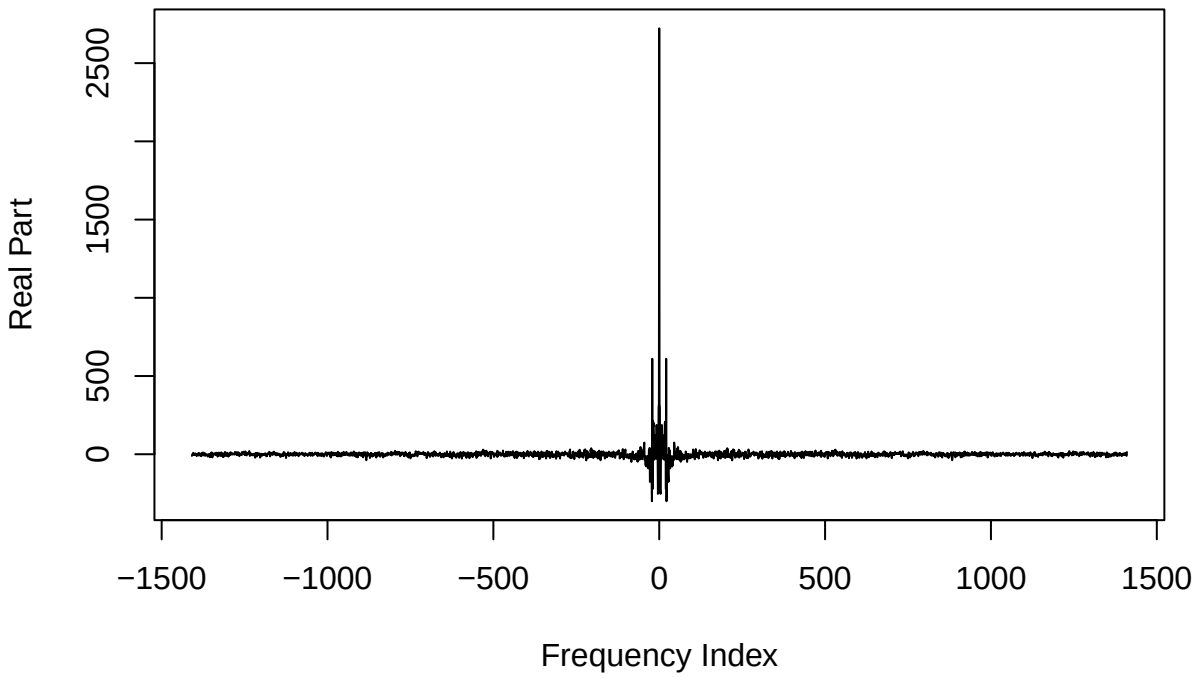
- Let $X_1, \ldots, X_n$ be a sample from a mean zero, covariance stationary time series with absolutely summable ACVF and spectral density f. Then $\widetilde{\underline{X}}$ has approximate covariance matrix $\text{diag}\{f(\lambda_{[n/2]-n+k})\}$.
- This follows from Theorem 6.4.5:

$$\text{Cov}[\widetilde{\underline{X}}] = Q^* \, \text{Cov}[\underline{X}] \, Q \approx \Lambda.$$

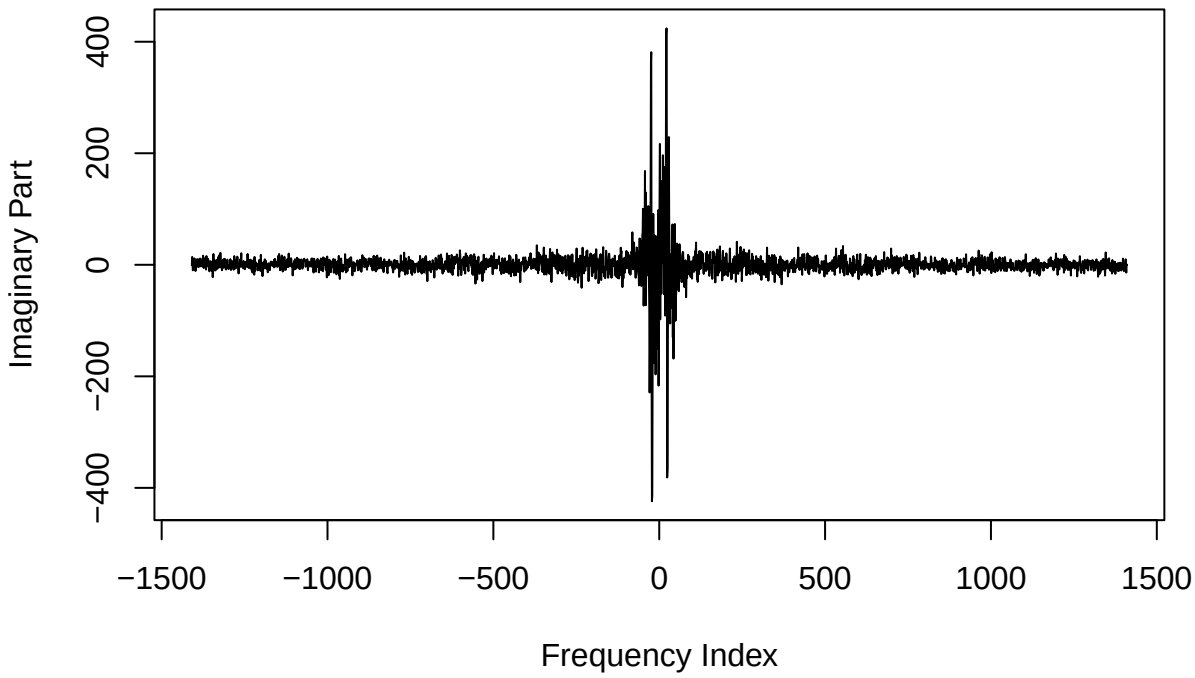
Exercise 7.18. DFT of Wolfer Sunspots.

- We compute the DFT of the Wolfer sunspot data, and plot.

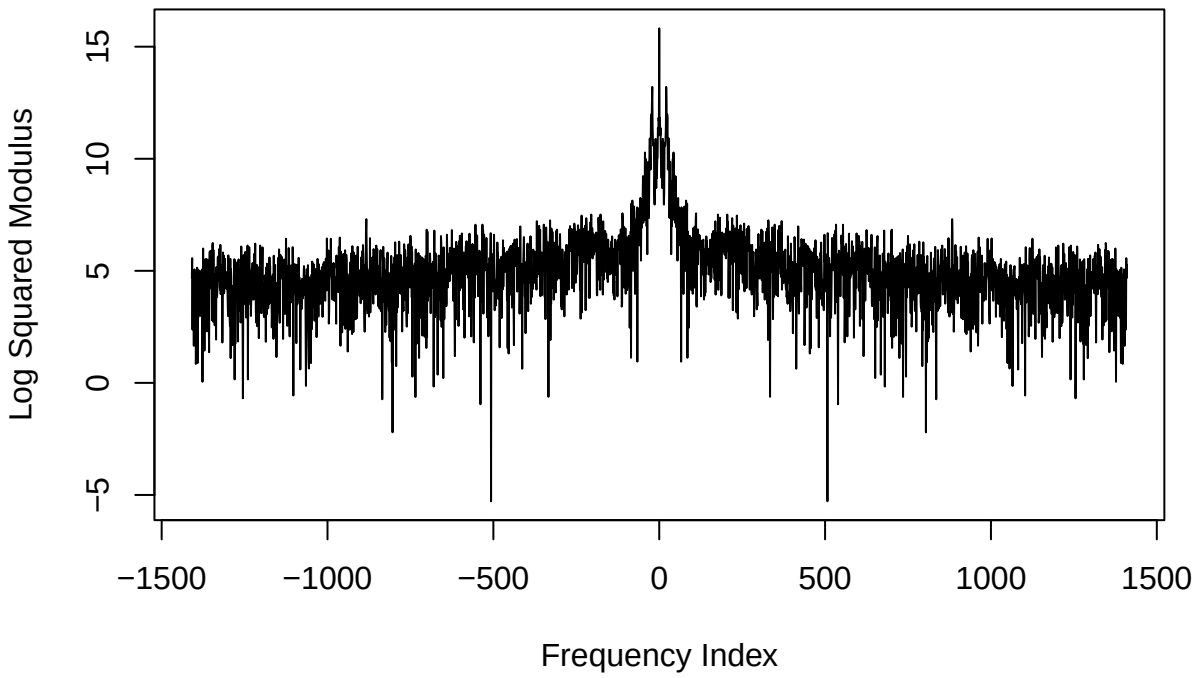
```
wolfer <- read.table("wolfer.dat")
wolfer <- ts(wolfer,start=1749,frequency=12)
n <- length(wolfer)
Q.mat <- get.qmat(n)
x.dft <- Conj(t(Q.mat)) %*% wolfer
plot(ts(Re(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Real Part")
```




```
plot(ts(Im(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Imaginary Part")
```




```
plot(ts(log(Mod(x.dft)^2),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Log Squared Modulus")
```

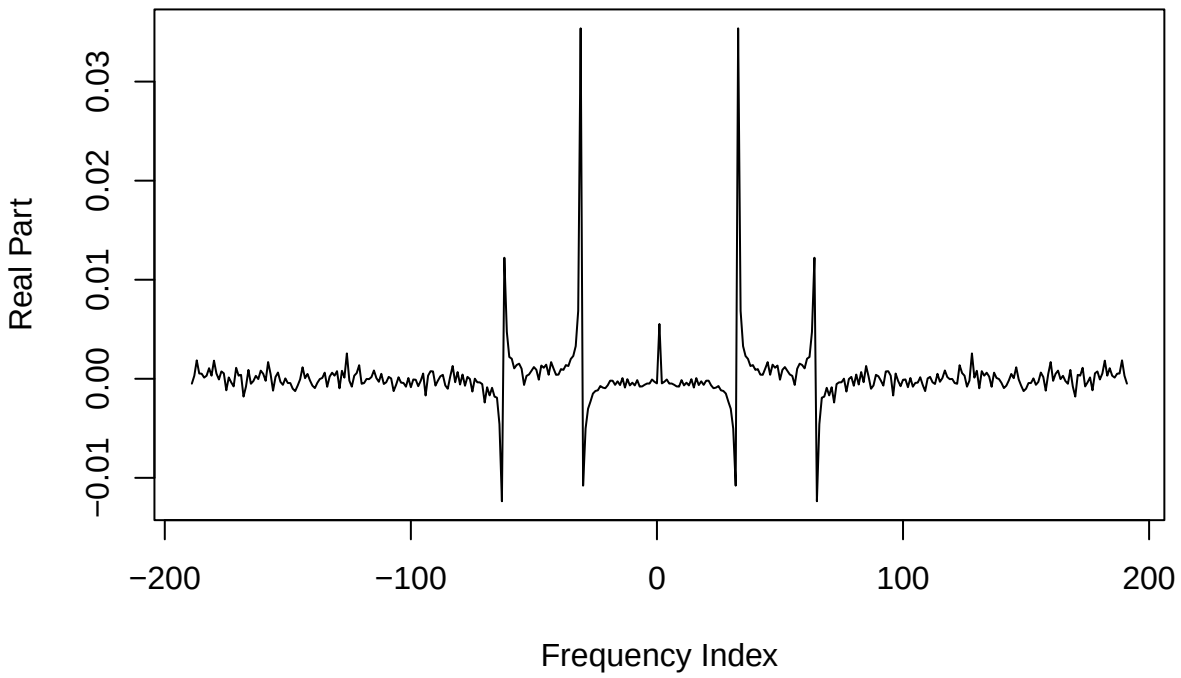


- We see higher values of the squared modulus (the uncentered periodogram) near frequency zero.

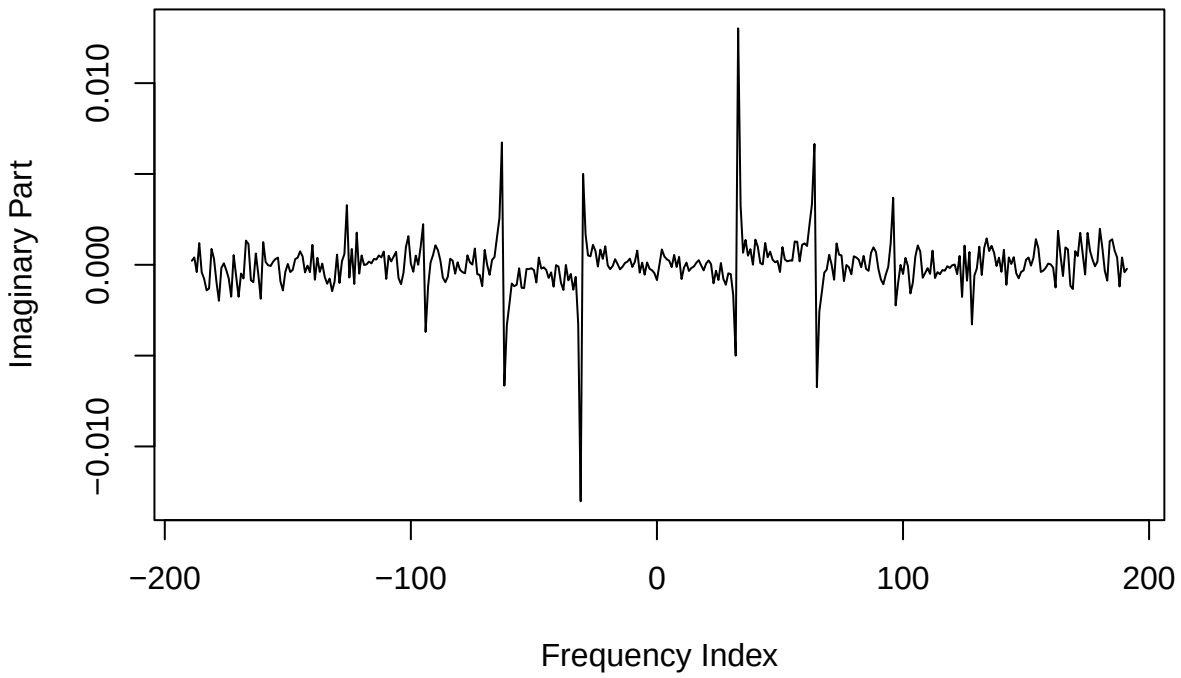
Exercise 7.20. DFT of Mauna Loa Growth Rate.

- We compute the DFT of the Mauna Loa growth rate, and plot.

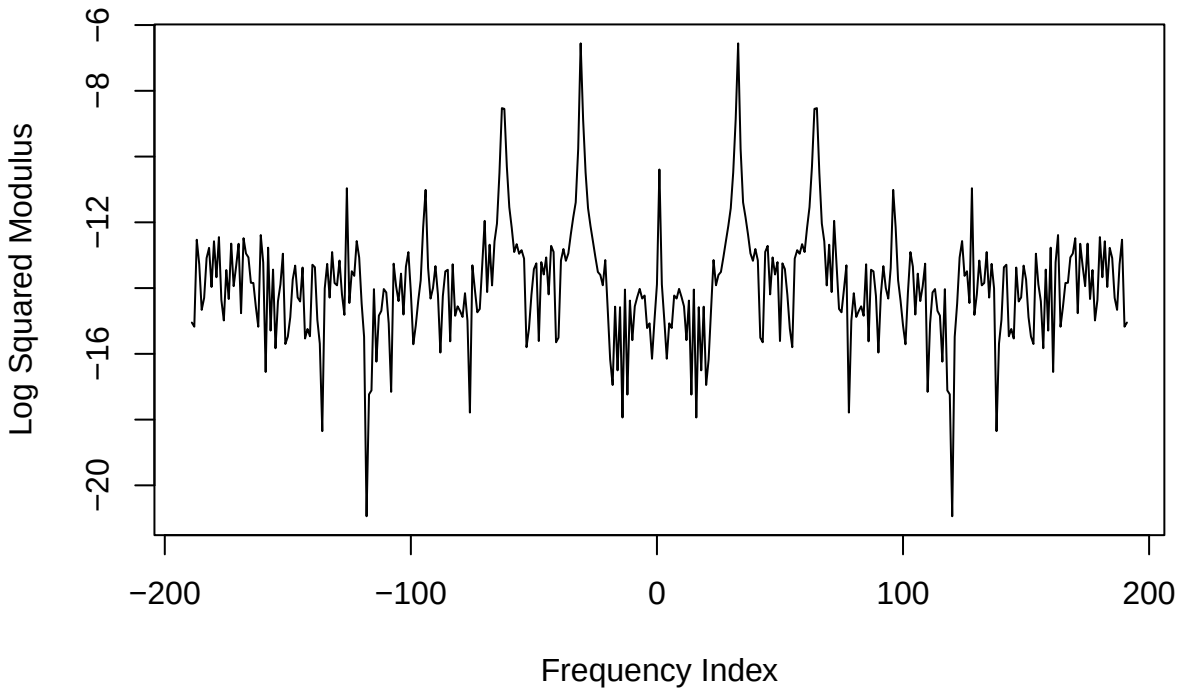
```
mau <- read.table("mauna.dat",header=TRUE,sep="")
mau <- ts(mau,start=1958,frequency=12)
mau.gr <- diff(log(mau))
n <- length(mau.gr)
Q.mat <- get.qmat(n)
x.dft <- Conj(t(Q.mat)) %*% mau.gr
plot(ts(Re(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Real Part")
```




```
plot(ts(Im(x.dft),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Imaginary Part")
```




```
plot(ts(log(Mod(x.dft)^2),start=-floor(n/2)+1,frequency=1),
    xlab="Frequency Index",ylab="Log Squared Modulus")
```



- We see higher values of the uncentered periodogram in the shape of peaks, at four non-zero frequencies. These correspond to cyclical seasonal effects.

Lesson 7-3: Spectral Representation

We discuss a representation of a stationary time series as a sum of stochastic cosines.

Definition 7.3.1.

- For a given spectral distribution F, a **spectral increment process** Z is a complex-valued continuous-time stochastic process defined on the interval $[-\pi, \pi]$, which has mean zero with *orthogonal increments*: for $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$, the random variables $Z(\lambda_2) - Z(\lambda_1)$ and $Z(\lambda_4) - Z(\lambda_3)$ are orthogonal. Also,

$$\mathrm{Var}[Z(\lambda_2) - Z(\lambda_1)] = \frac{1}{2\pi}\left(F(\lambda_2) - F(\lambda_1)\right).$$

- The variance of a complex random variable is the expectation of its squared modulus.
- We abbreviate the above expression by

$$\mathrm{Var}[dZ(\lambda)] = \frac{1}{2\pi} dF(\lambda).$$

Paradigm 7.3.4. Time Series Defined as a Stochastic Integral

- We can define a time series via a stochastic integral as follows:

$$X_t = \int_{-\pi}^{\pi} e^{i\lambda t} dZ(\lambda).$$

- This resembles a sum of stochastic cosines, where $dZ(\lambda)$ is the amplitude for a sinusoid $e^{i\lambda t}$.
- Then $\{X_t\}$ has mean zero and autocovariance

$$\text{Cov}[X_{t+h}, X_t] = \int_{-\pi}^{\pi}\int_{-\pi}^{\pi} e^{i\lambda(t+h)}e^{-i\omega t}\text{Cov}[dZ(\lambda), dZ(\omega)] = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i\lambda h}dF(\lambda).$$

- So $\{X_t\}$ is weakly stationary with ACVF $\gamma(h)$ given by above formula.
- Conversely: any mean zero weakly stationary time series $\{X_t\}$ with spectral distribution function F can be represented by the above stochastic integral!

Corollary 7.3.8.

- Suppose $\{X_t\}$ is a mean zero weakly stationary time series with spectral representation, and let $Y_t = \psi(B)X_t$. Then

$$Y_t = \sum_j \psi_j X_{t-j} = \sum_j \psi_j \int_{-\pi}^{\pi} e^{i\lambda(t-j)}dZ(\lambda) = \int_{-\pi}^{\pi}\sum_j \psi_j e^{-i\lambda j}e^{i\lambda t}dZ(\lambda) = \int_{-\pi}^{\pi}\psi(e^{-i\lambda})e^{i\lambda t}dZ(\lambda).$$

- So $\{Y_t\}$ also has a spectral representation, but its increment process is $\psi(e^{-i\lambda})dZ(\lambda)$, where $\psi(e^{-i\lambda})$ is the filter frequency response function. We can use this result to understand the impact of a filter in the frequency domain.
- The ACVF is

$$\text{Cov}[Y_{t+h}, Y_t] = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i\lambda h}|\psi(e^{-i\lambda})|^2 dF(\lambda).$$

Example 7.3.10. Time Shift

- Consider $\psi(B) = B^k$, a shift by k time units. Then $\psi(e^{-i\lambda}) = e^{-i\lambda k}$ and $Y_t = \int_{-\pi}^{\pi} e^{i\lambda(t-k)}dZ(\lambda)$.
- This is the same as $B^k X_t = X_{t-k}$.

Definition 7.3.11.

- We can decompose the frequency response function into the **gain** function and the **phase delay** function, each with an interpretation from the spectral representation.
- We use the polar decomposition of a complex number: for any λ,

$$\psi(e^{-i\lambda}) = |\psi(e^{-i\lambda})|\ \exp\{i\text{Arg}\psi(e^{-i\lambda})\}.$$

- The magnitude $|\psi(e^{-i\lambda})|$ is called the **gain** function of the filter. It is computed by taking the square root of sum of squares of real and imaginary parts.
- The angular portion $\text{Arg}\psi(e^{-i\lambda})$ is called the **phase** function of the filter. It is computed by taking the arc tangent of the ratio of imaginary to real parts.
- When the phase function is differentiable with respect to λ, we define the **phase delay** via

$$\Upsilon(\lambda) = \frac{-\text{Arg}\psi(e^{-i\lambda})}{\lambda}$$

for $\lambda \neq 0$, and the limit of such for $\lambda = 0$.
- The phase delay may be discontinuous in λ.
- The gain is an even function; both phase and phase delay are odd. So we usually just plot them over $[0, \pi]$ instead of $[-\pi, \pi]$.

Fact 7.3.12. Action of Phase Delay.

- From Corollary 7.3.8,

$$Y_t = \int_{-\pi}^{\pi} e^{i(\lambda t + \text{Arg}_{\psi(e^{-i\lambda})})}|\psi(e^{-i\lambda})|dZ(\lambda) = \int_{-\pi}^{\pi} e^{i\lambda(t-\Upsilon(\lambda))}|\psi(e^{-i\lambda})|dZ(\lambda).$$

- So at frequency λ, time t is delayed by $\Upsilon(\lambda)$ time units.
- Also, the gain modifies the autocovariances.

Example 7.3.13. Simple Moving Average Filters Cause Delay

- Consider the simple moving average filter $\psi(B) = (1 + B + B^2)/3$, which gives the average of the past and present 3 observations.
- We directly compute the frequency response function:

$$\psi(e^{-i\lambda}) = \frac{1 + e^{-i\lambda} + e^{-i2\lambda}}{3} = e^{-i\lambda}\,\frac{e^{i\lambda} + 1 + e^{-i\lambda}}{3} = e^{-i\lambda}\,\frac{1 + 2\cos(\lambda)}{3}.$$

- The gain function is

$$\frac{1}{3}|1 + 2\cos(\lambda)|.$$

- Note that $1 + 2\cos(\lambda)$ is non-negative for $\lambda \in [0, 2\pi/3]$, so the phase function equals $-\lambda$ over that set. Otherwise, we need a -1 factor which leads to a phase function equal to $-\lambda - \pi$.
- The phase delay function is then

$$\Upsilon(\lambda) = \begin{cases} 1 & \text{if } \lambda \in [0, 2\pi/3] \\ 1 + \pi/\lambda & \text{else.} \end{cases}$$

- Since this function is positive, the filter always provides a delay.
- The gain function attenuates higher frequencies, due to the cosine shape, so the filter is a "low-pass."

Example 7.3.14. Differencing Causes an Advance

- Consider the differencing filter $\psi(B) = 1 - B$.
- The frequency response function is

$$\psi(e^{-i\lambda}) = 1 - e^{-i\lambda} = 1 - \cos(\lambda) - i\sin(\lambda).$$

- The squared gain function is $2 - 2\cos(\lambda)$.
- The phase function is $(\pi - \lambda)/2$.
- Away from $\lambda = 0$, the phase delay is $\Upsilon(\lambda) = .5(1 - \pi/\lambda)$. This is negative for $\lambda \in (0, \pi]$, so differencing causes an advance.
- The gain function attenuates lower frequencies, so the filter is a "high-pass."

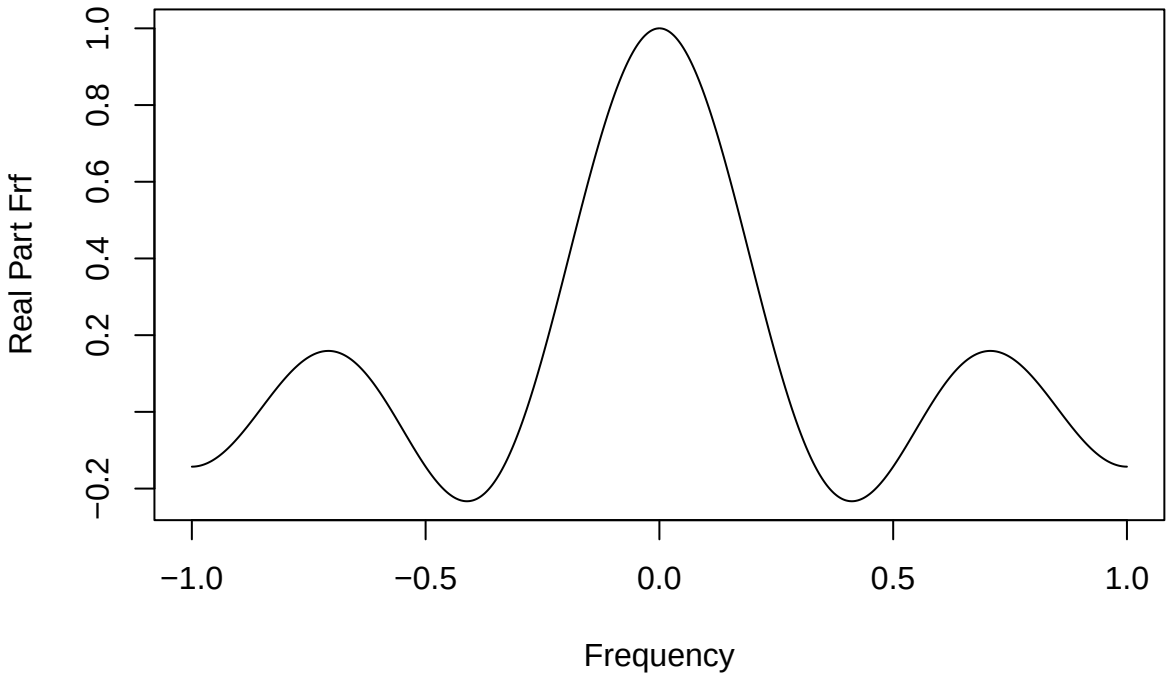
Exercise 7.29. Phase and Gain for Simple Moving Average.

- Consider the simple moving average of order p: $\psi(B) = (2p + 1)^{-1} \sum_{j=-p}^{p} B^j$.
- It can be shown that

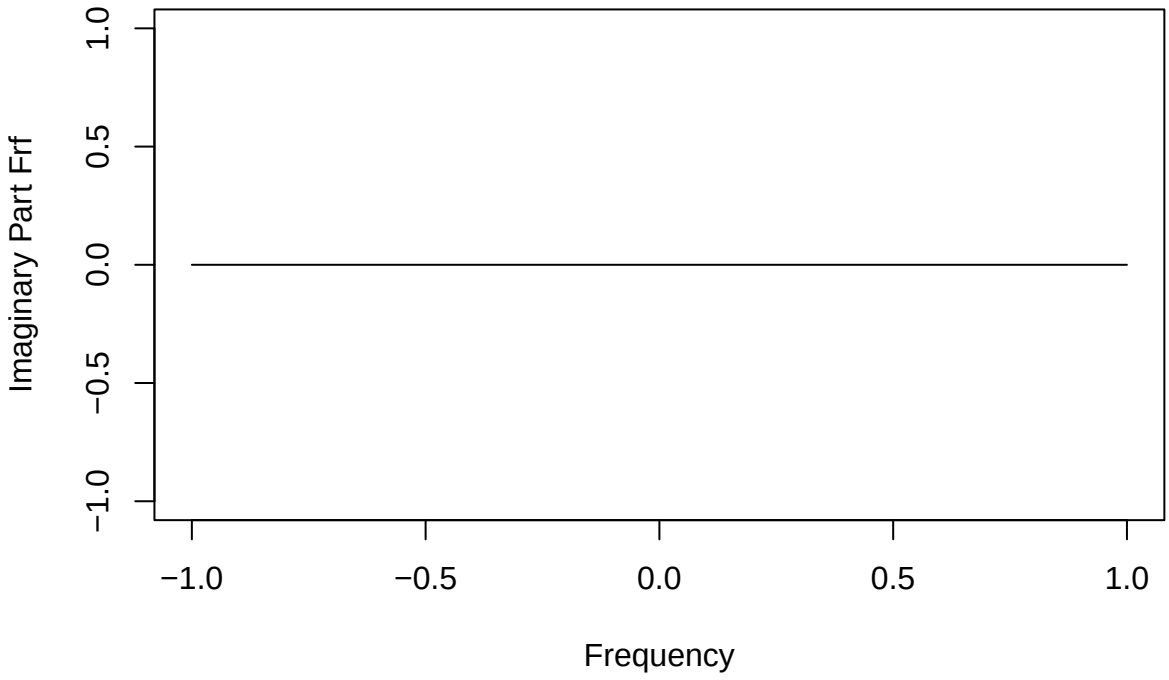
$$\psi(e^{-i\lambda}) = \frac{\sin(\lambda(p + 1/2))}{(2p + 1)\sin(\lambda/2)}$$

- We use this formula to encode and display the gain and phase functions, as well as the real and imaginary parts of the frequency response function.

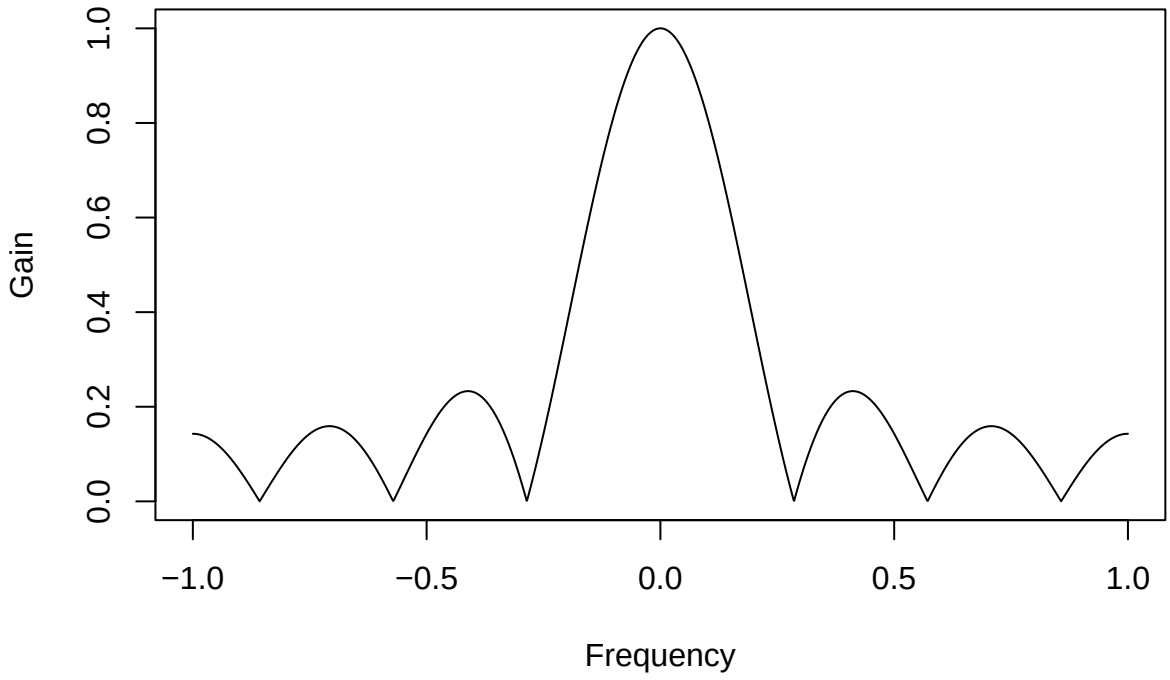
```
lambda <- pi*seq(-1000,1000)/1000
p <- 3
simplema.frf <- sin((p+1/2)*lambda)/((2*p+1)*sin(lambda/2))
simplema.gain <- Mod(simplema.frf)
simplema.phase <- atan(Im(simplema.frf)/Re(simplema.frf))
simplema.delay <- -1*simplema.phase/lambda
plot(ts(Re(simplema.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Real Part Frf")
```



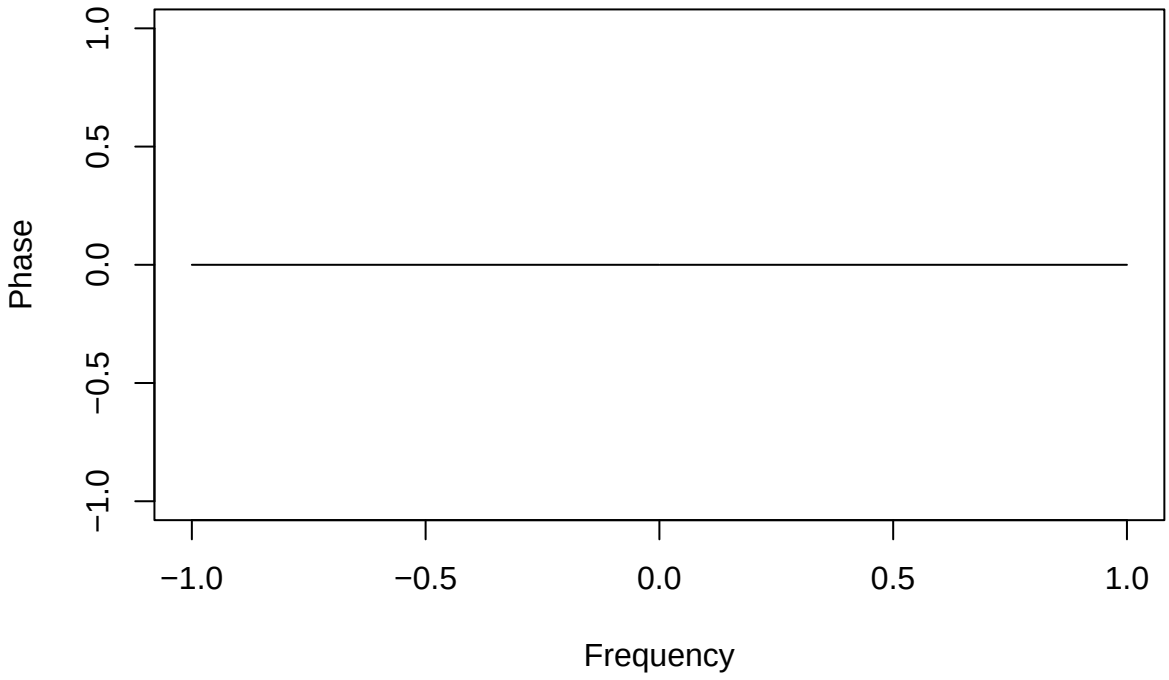

```
plot(ts(Im(simplema.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Imaginary Part Frf")
```



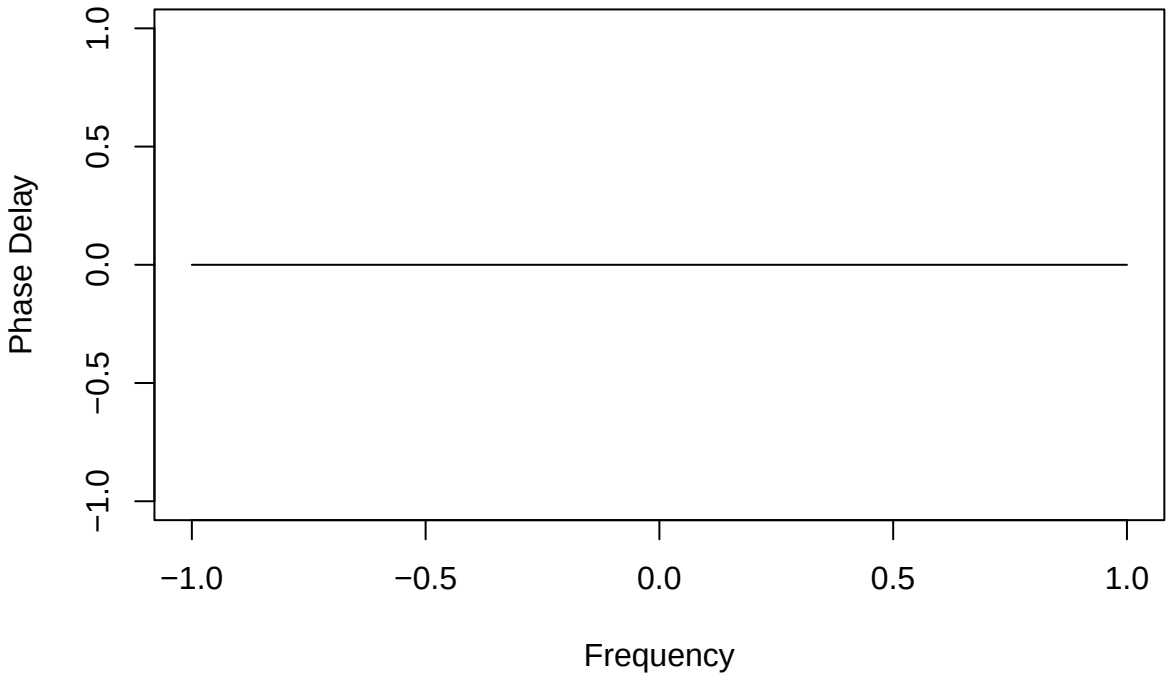

```
plot(ts(simplema.gain,start=-1,frequency=1000),
    xlab="Frequency",ylab="Gain")
```




```
plot(ts(simplema.phase,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase")
```



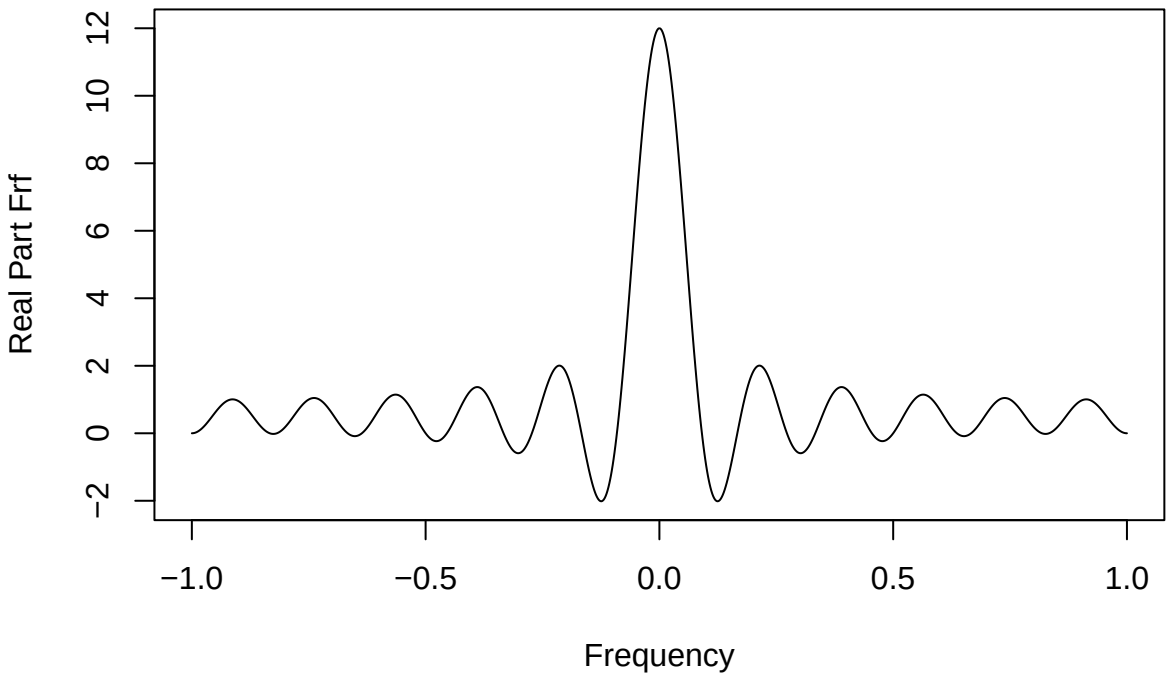

```
plot(ts(simplema.delay,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase Delay")
```



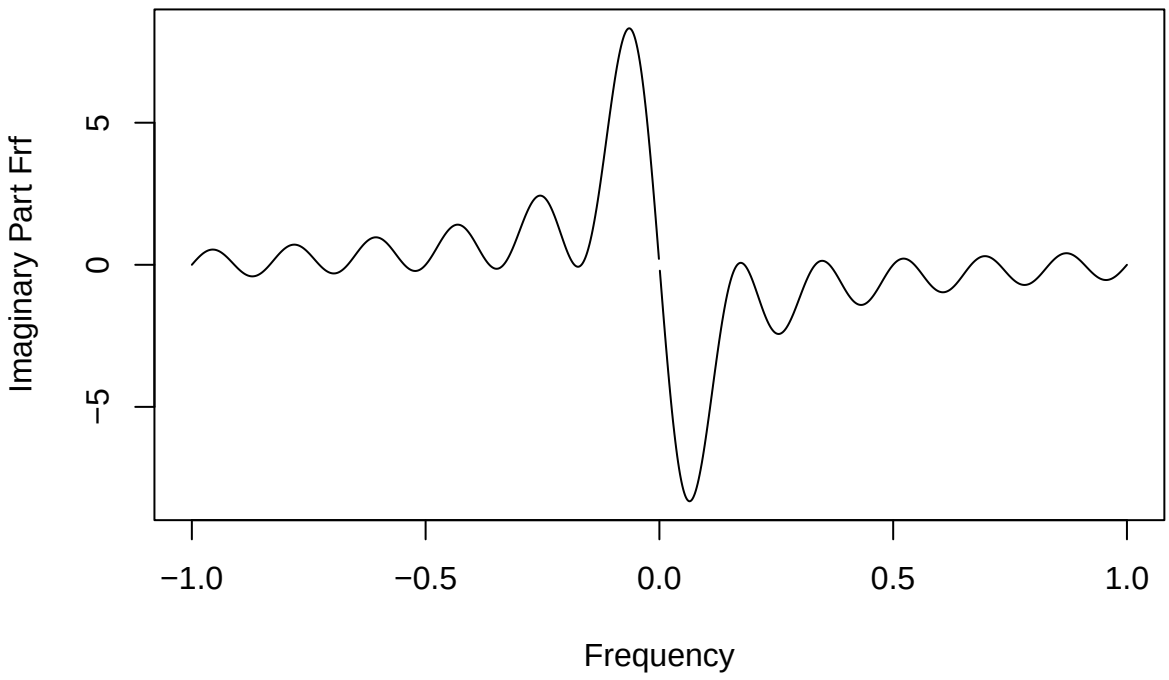
Exercise 7.30. Phase and Gain for Seasonal Aggregation Filter.

- Consider the seasonal aggregation filter: $\psi(B) = \sum_{j=0}^{s-1} B^j$.
- We encode and display the gain and phase functions, as well as the real and imaginary parts of the frequency response function.

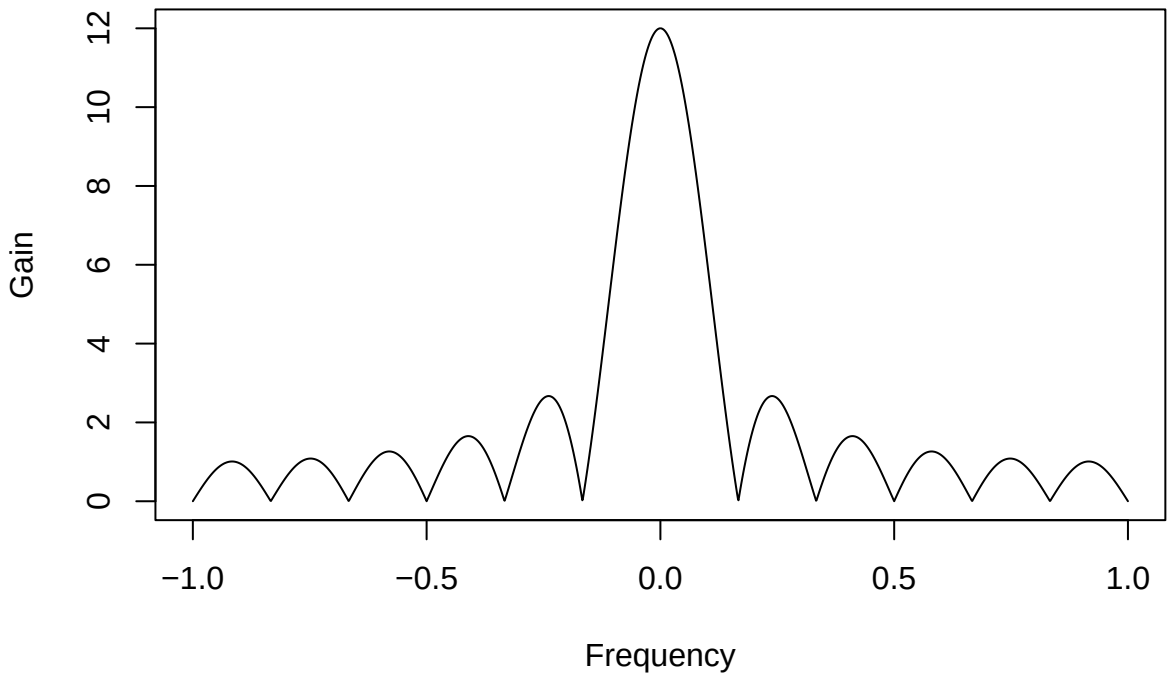
```
lambda <- pi*seq(-1000,1000)/1000
s <- 12
seasagg.frf <- exp(-1i*lambda*(s-1)/2)*sin((s/2)*lambda)/sin(lambda/2)
seasagg.gain <- Mod(seasagg.frf)
seasagg.phase <- atan(Im(seasagg.frf)/Re(seasagg.frf))
seasagg.delay <- -1*seasagg.phase/lambda
seasagg.delay[1001] <- NA
plot(ts(Re(seasagg.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Real Part Frf")
```



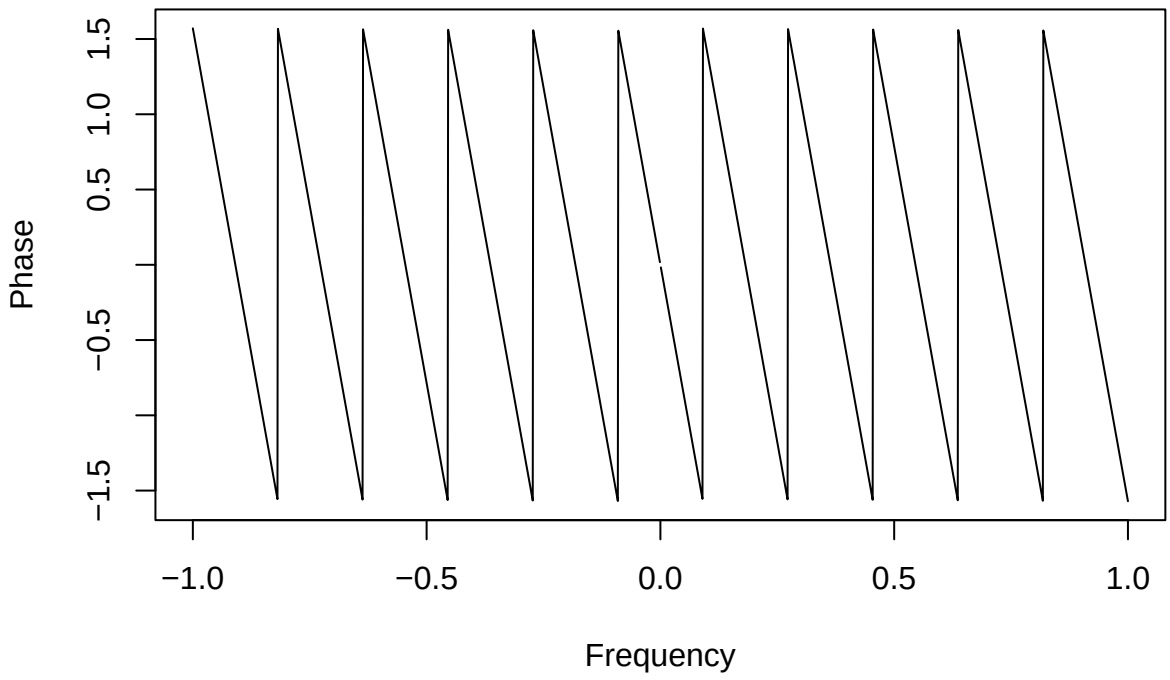

```
plot(ts(Im(seasagg.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Imaginary Part Frf")
```



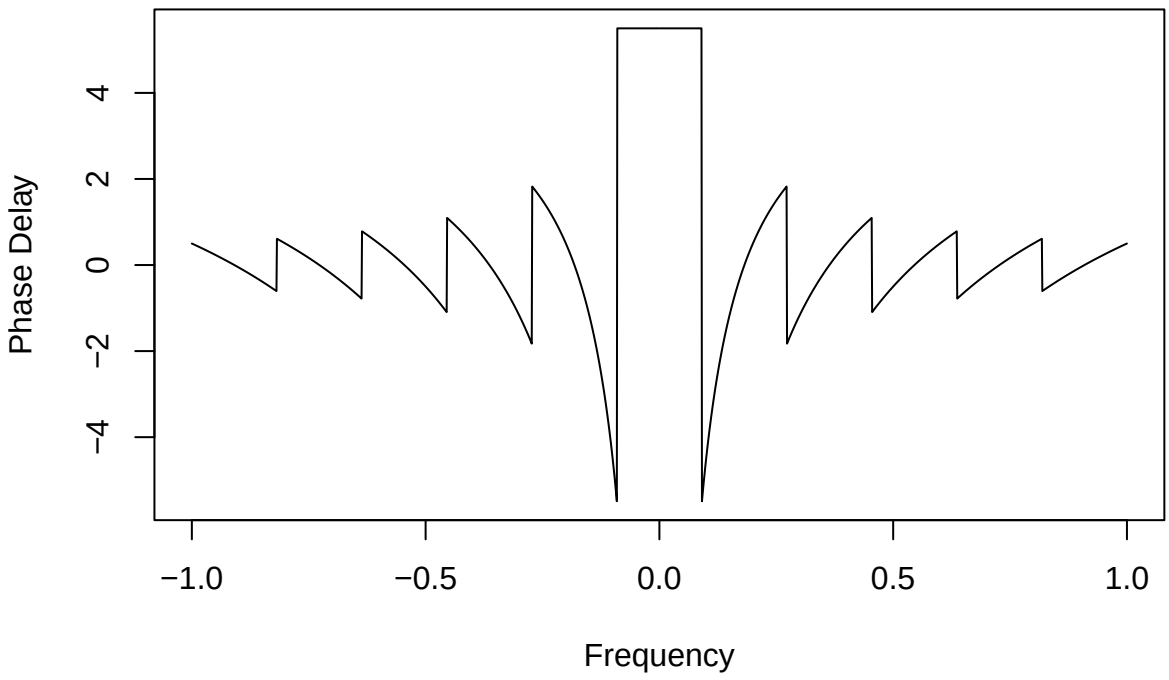

```
plot(ts(seasagg.gain,start=-1,frequency=1000),
    xlab="Frequency",ylab="Gain")
```




```
plot(ts(seasagg.phase,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase")
```




```
plot(ts(seasagg.delay,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase Delay")
```



Lesson 7-4: Optimal Filtering

We consider applications of the spectral representation to prediction problems.

7.4.3. Optimal h-Step-Ahead Forecasting

- We seek an expression for $\widehat{X}_{t+h} = P_{\overline{\text{sp}}\{X_s, s \leq t\}}[X_{t+h}]$.
- The case of $h = 1$ for an MA(1) was considered in Example 6.3.3.
- We suppose that $\{X_t\}$ is stationary and invertible, with absolutely summable ACVF.
- So the optimal linear predictor is some "causal" filter $\psi(B)$, where $\psi_j = 0$ for $j < 0$, and $\widehat{X}_{t+h} = \psi(B)X_t$.
- By the spectral representation,

$$\widehat{X}_{t+h} = \int_{-\pi}^{\pi} e^{i\lambda t}\psi(e^{-i\lambda})dZ(\lambda) \quad \text{and} \quad X_{t+h} = \int_{-\pi}^{\pi} e^{i\lambda(t+h)}dZ(\lambda).$$

- Using the normal equations, we have the condition that for all $j \geq 0$,

$$0 = \text{Cov}[X_{t+h} - \widehat{X}_{t+h}, X_{t-j}] = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i\lambda j}[e^{i\lambda h} - \psi(e^{-i\lambda})]\, dF(\lambda).$$

- Since the ACVF is absolutely summable, the spectral density exists, and $dF(\lambda) = f(\lambda)d\lambda$.
- By Theorem 6.1.16, we have $f(\lambda) = \sigma^2|\theta(e^{-i\lambda})|^2$ for some $\theta(z) = \sum_{k\geq 0}\theta_k z^k$ (with $\theta_0 = 1$). Also $\theta(e^{-i\lambda})$ is non-zero by the invertibility assumption.
- Guess the following solution:

$$\psi(e^{-i\lambda}) = \sum_{k\geq 0}\theta_{k+h}e^{-i\lambda k}/\theta(e^{-i\lambda}).$$

- Check the normal equations:

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i\lambda j}[e^{i\lambda h} - \psi(e^{-i\lambda})]\, f(\lambda)d\lambda \tag{1}$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i\lambda j}\left[e^{i\lambda h} - \sum_{k\geq h}\theta_k e^{-i\lambda(k-h)}/\theta(e^{-i\lambda})\right] f(\lambda)d\lambda \tag{2}$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i\lambda(h+j)}\left[\theta(e^{-i\lambda}) - \sum_{k\geq h}\theta_k e^{-i\lambda k}\right] \sigma^2\theta(e^{i\lambda})d\lambda \tag{3}$$

$$= \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{i\lambda(h+j)}\sum_{k<h}\theta_k e^{-i\lambda k}\,\sigma^2\theta(e^{i\lambda})d\lambda. \tag{4}$$

  The integrand is a linear combination of strictly positive power of $e^{i\lambda}$, each of which integrates to zero! So the normal equations hold.

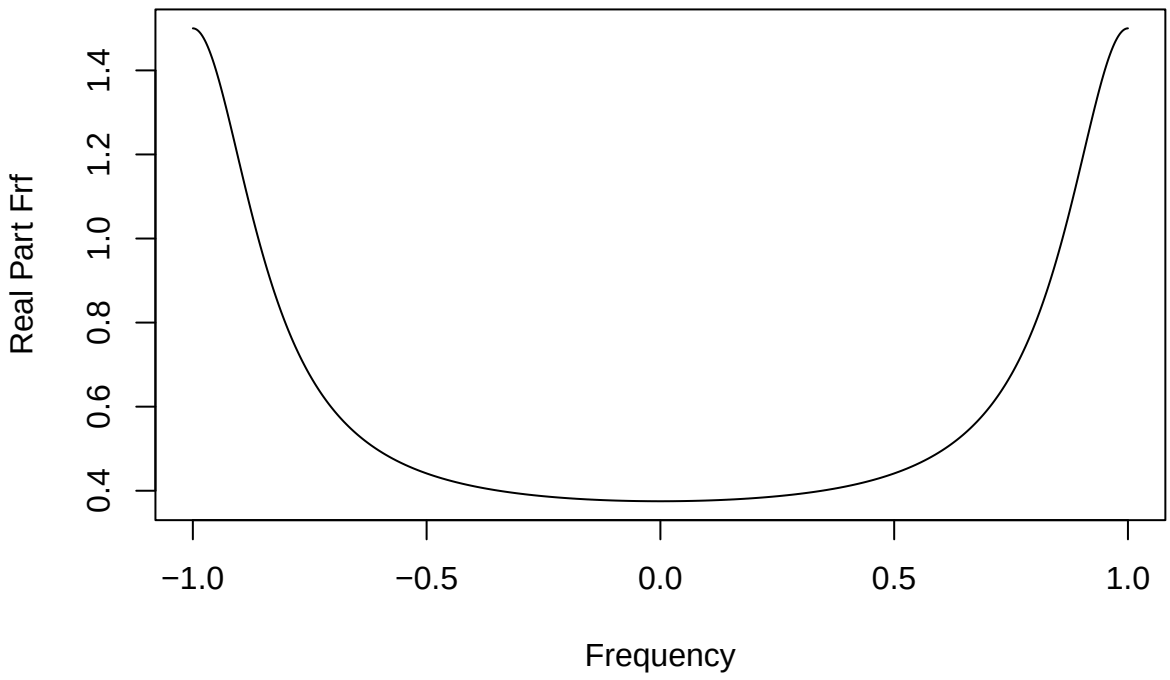
Example of an MA(1)

- We specialize the above result to an MA(1), where $\theta(z) = 1 + \theta_1 z$:

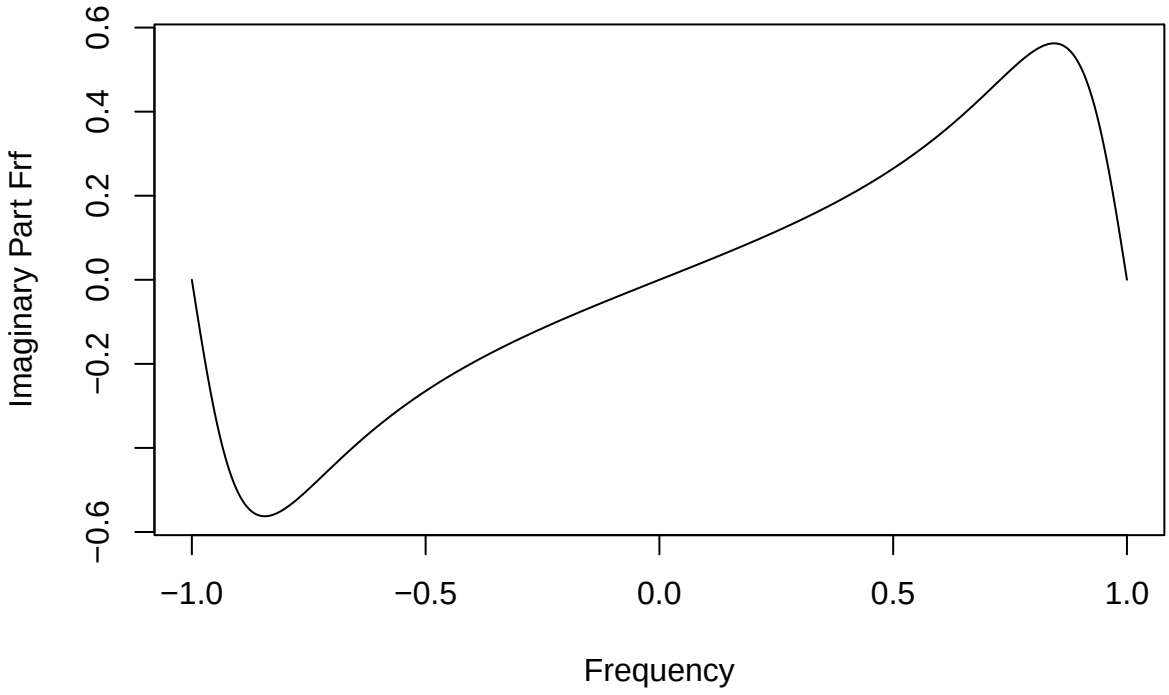
$$\psi(e^{-i\lambda}) = \sum_{k\geq 0}\theta_{k+h}e^{-i\lambda k}/\theta(e^{-i\lambda}) = 1_{\{h=1\}}\theta_1/(1+\theta_1 e^{-i\lambda}).$$

- So if $h > 1$, the forecast is zero!

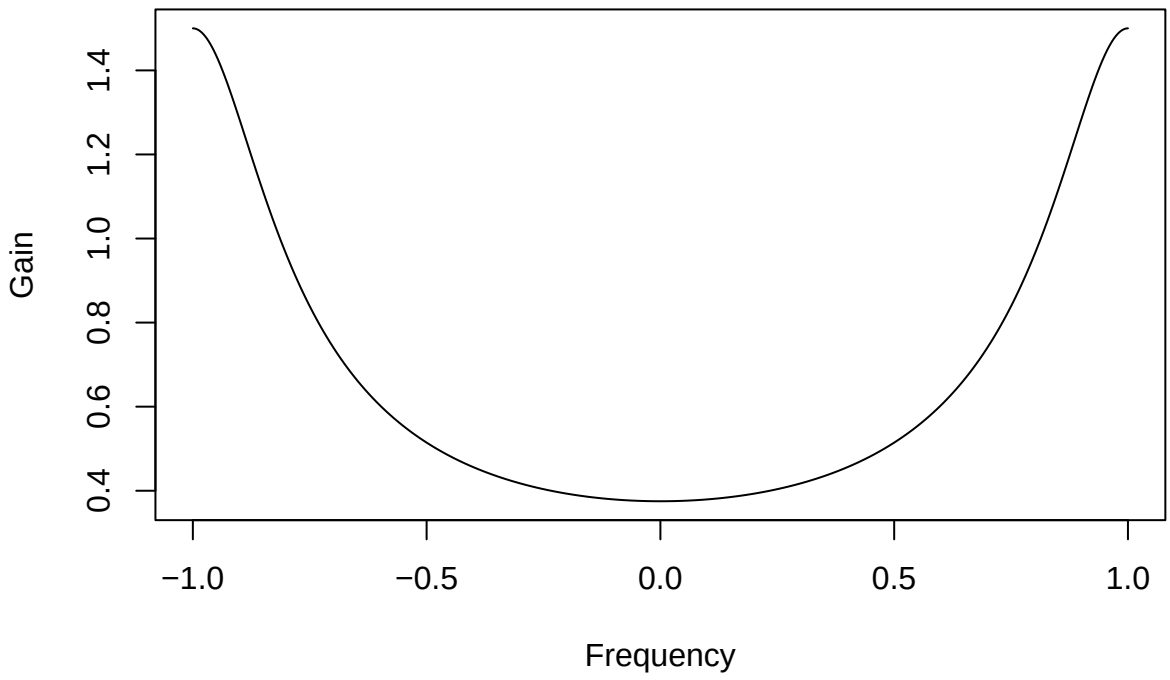
```
lambda <- pi*seq(-1000,1000)/1000
theta1 <- .6
forecast.frf <- theta1/(1 + theta1*exp(-1i*lambda))
forecast.gain <- Mod(forecast.frf)
forecast.phase <- atan(Im(forecast.frf)/Re(forecast.frf))
forecast.delay <- -1*forecast.phase/lambda
forecast.delay[1001] <- NA
plot(ts(Re(forecast.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Real Part Frf")
```



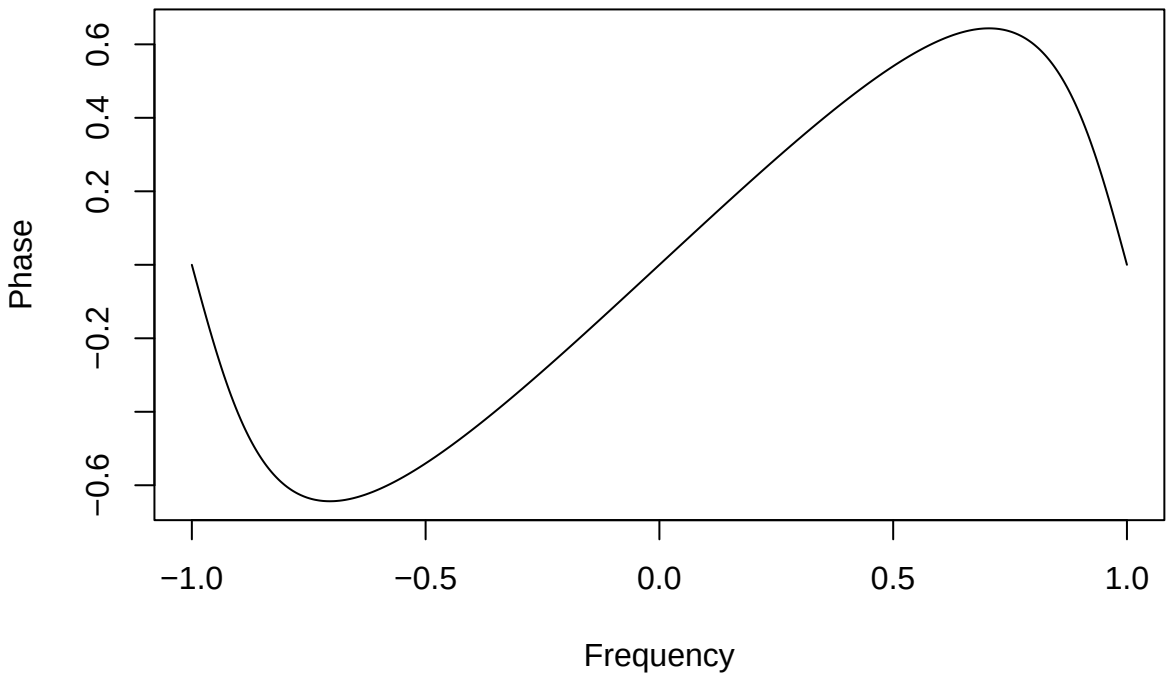

```
plot(ts(Im(forecast.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Imaginary Part Frf")
```



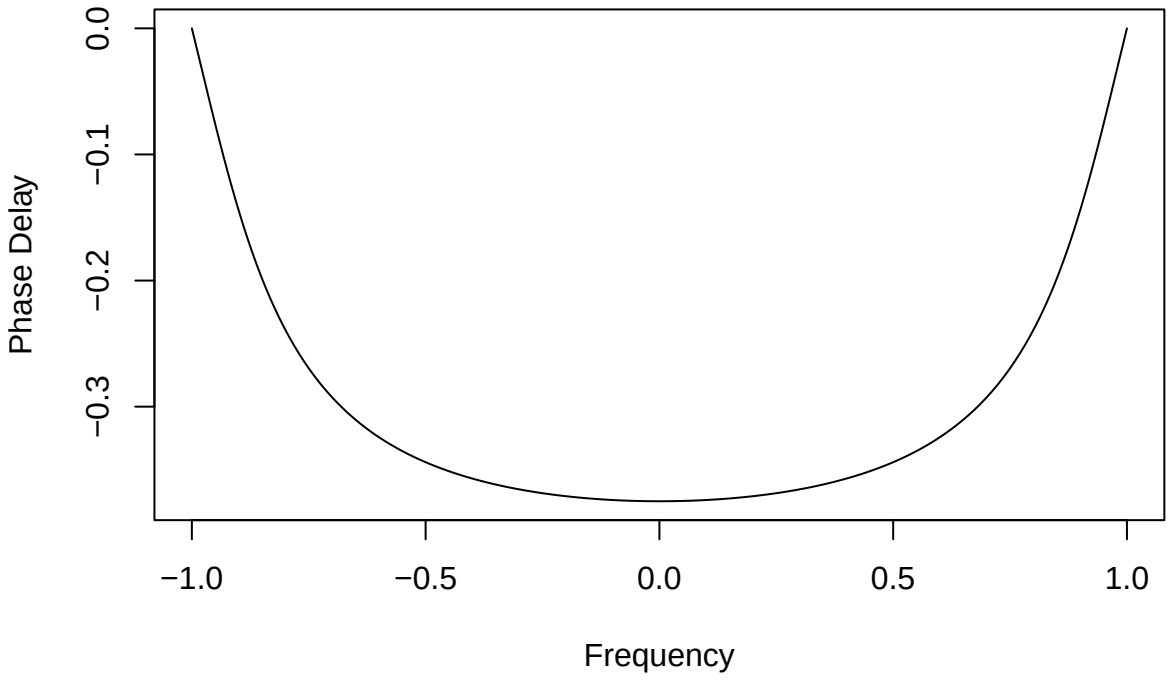

```
plot(ts(forecast.gain,start=-1,frequency=1000),
    xlab="Frequency",ylab="Gain")
```




```
plot(ts(forecast.phase,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase")
```




```
plot(ts(forecast.delay,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase Delay")
```



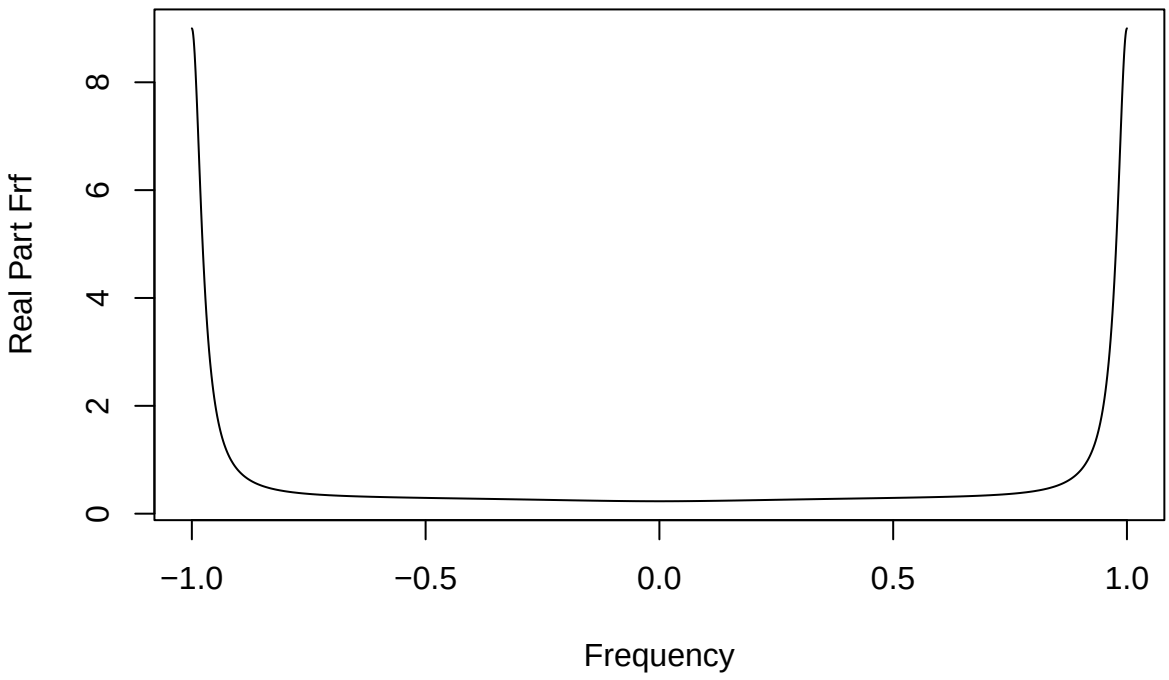
Example of an MA(2)

- We specialize the above result to an MA(2), where $\theta(z) = 1 + \theta_1 z + \theta_2 z^2$:

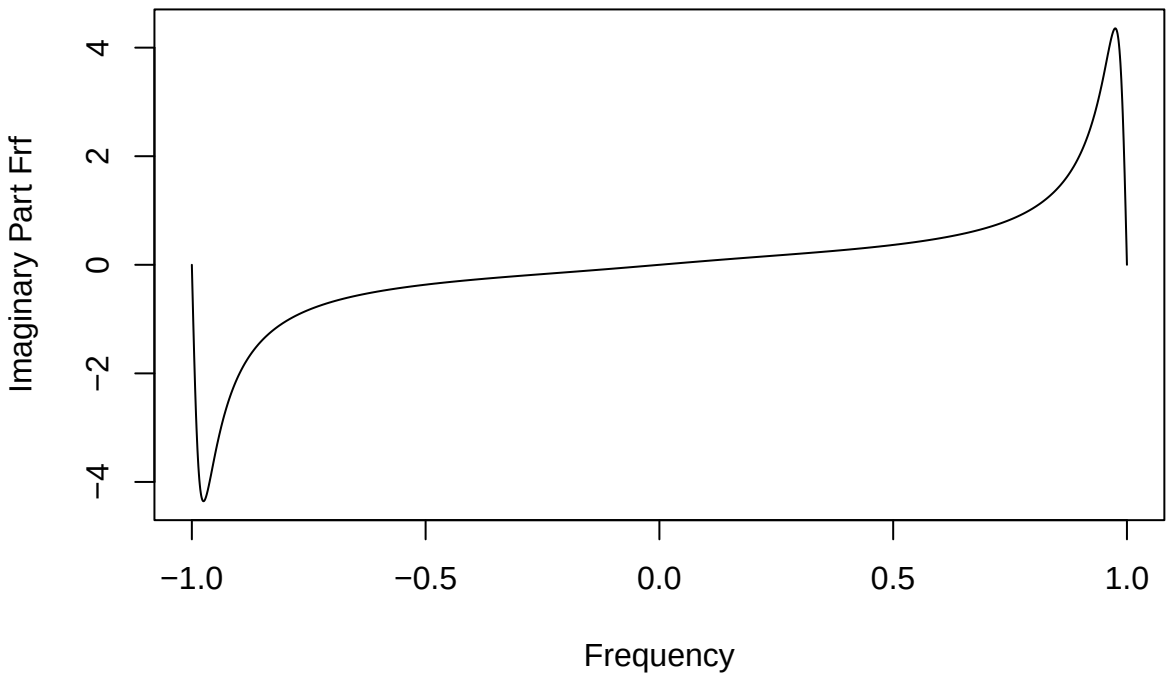
$$\psi(e^{-i\lambda}) = \sum_{k\geq 0} \theta_{k+h} e^{-i\lambda k} / \theta(e^{-i\lambda}) = 1_{\{h\leq 2\}}(\theta_h + \theta_{h+1} e^{-i\lambda})/(1 + \theta_1 e^{-i\lambda} + \theta_2 e^{-i2\lambda}).$$

- So if $h > 2$, the forecast is zero!
- Plots for the $h = 1$ case:

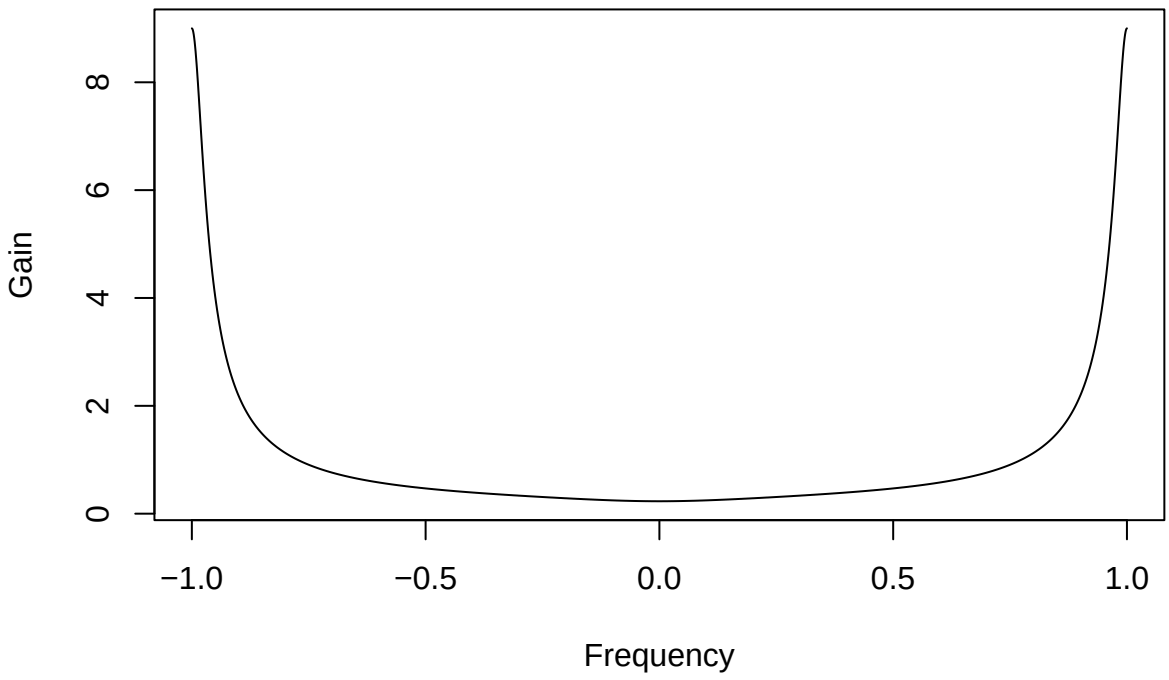
```
lambda <- pi*seq(-1000,1000)/1000
theta1 <- .6
theta2 <- -.3
h <- 1
forecast.frf <- (theta1 + theta2*exp(-1i*lambda))/(1 + theta1*exp(-1i*lambda) + theta2*exp(-1i*2*lambda
forecast.gain <- Mod(forecast.frf)
forecast.phase <- atan(Im(forecast.frf)/Re(forecast.frf))
forecast.delay <- -1*forecast.phase/lambda
forecast.delay[1001] <- NA
plot(ts(Re(forecast.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Real Part Frf")
```



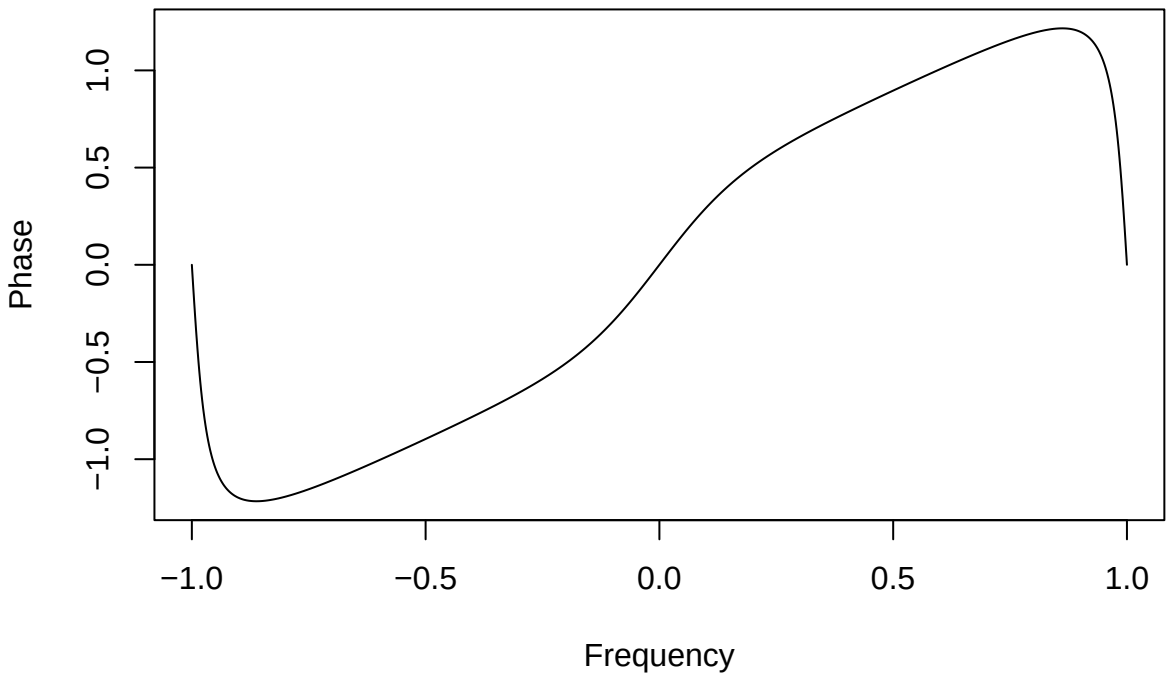

```
plot(ts(Im(forecast.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Imaginary Part Frf")
```



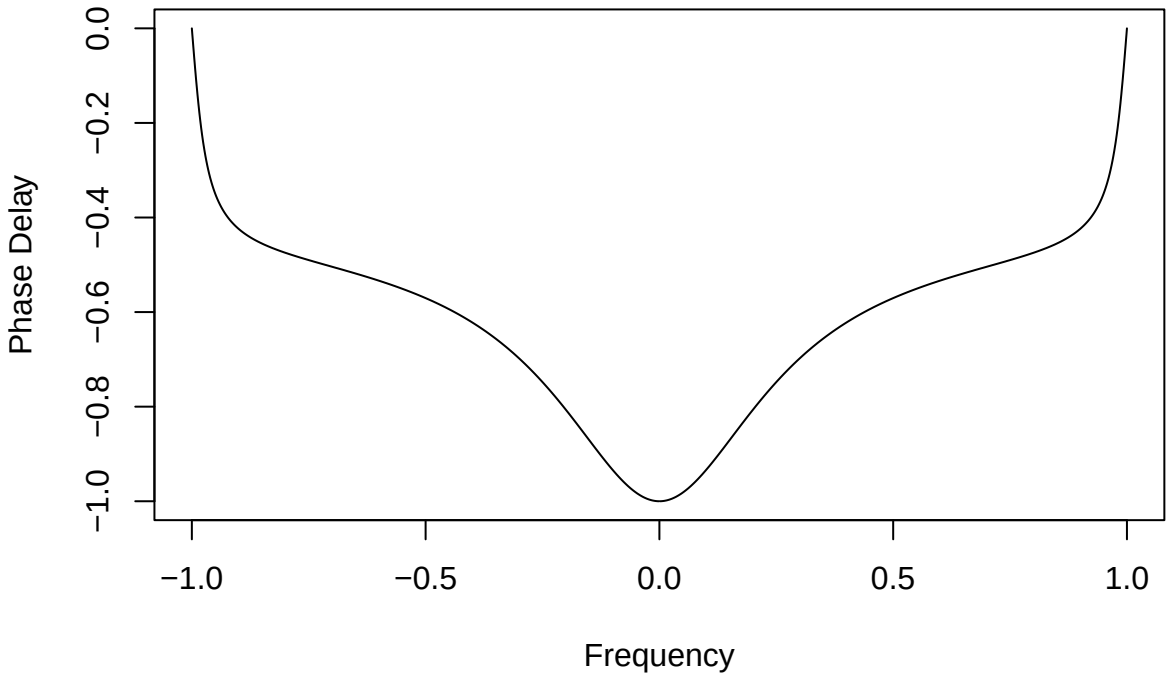

```
plot(ts(forecast.gain,start=-1,frequency=1000),
    xlab="Frequency",ylab="Gain")
```




```
plot(ts(forecast.phase,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase")
```

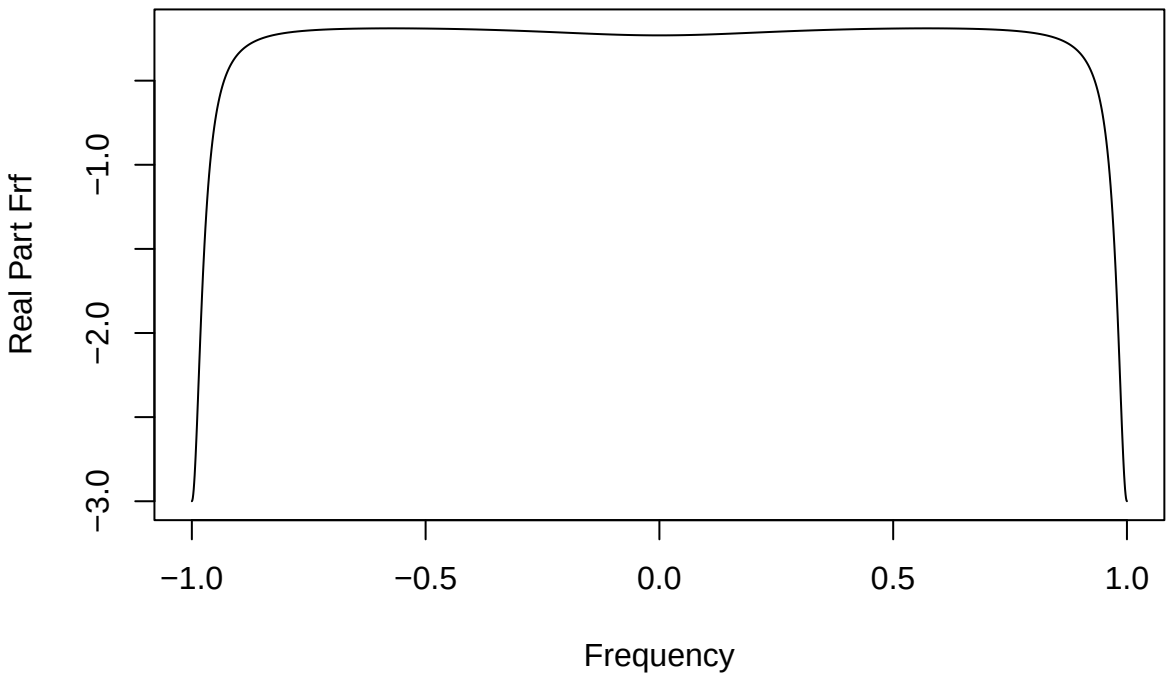



```
plot(ts(forecast.delay,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase Delay")
```

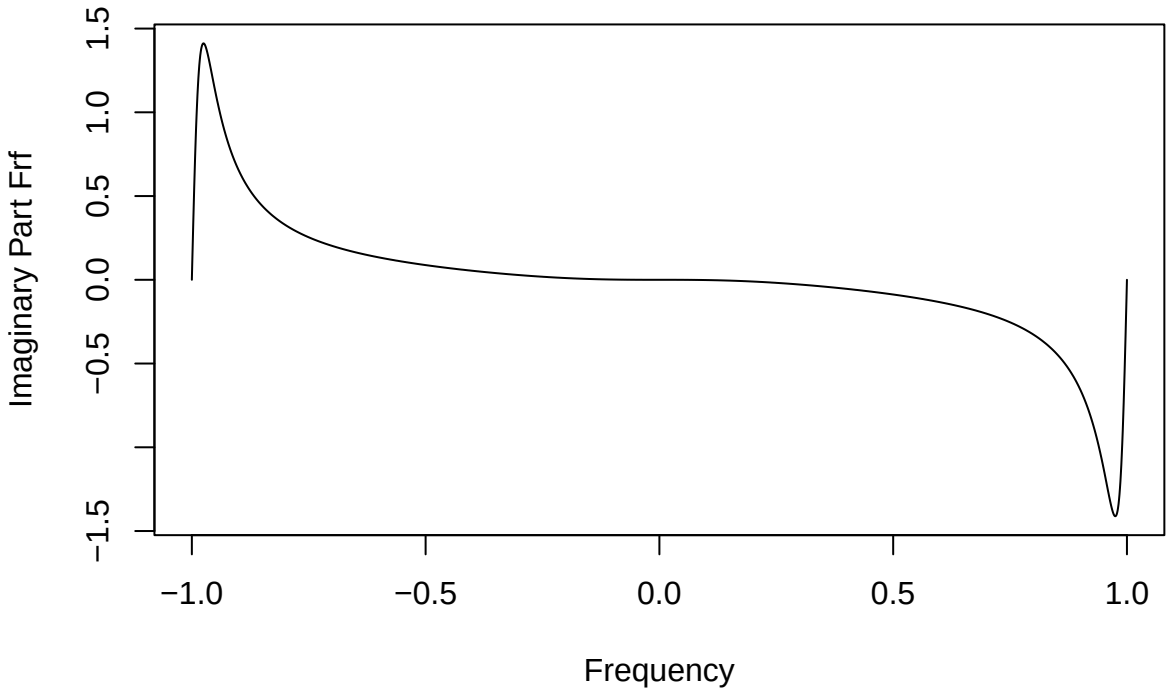


Plots for the $h = 2$ case:

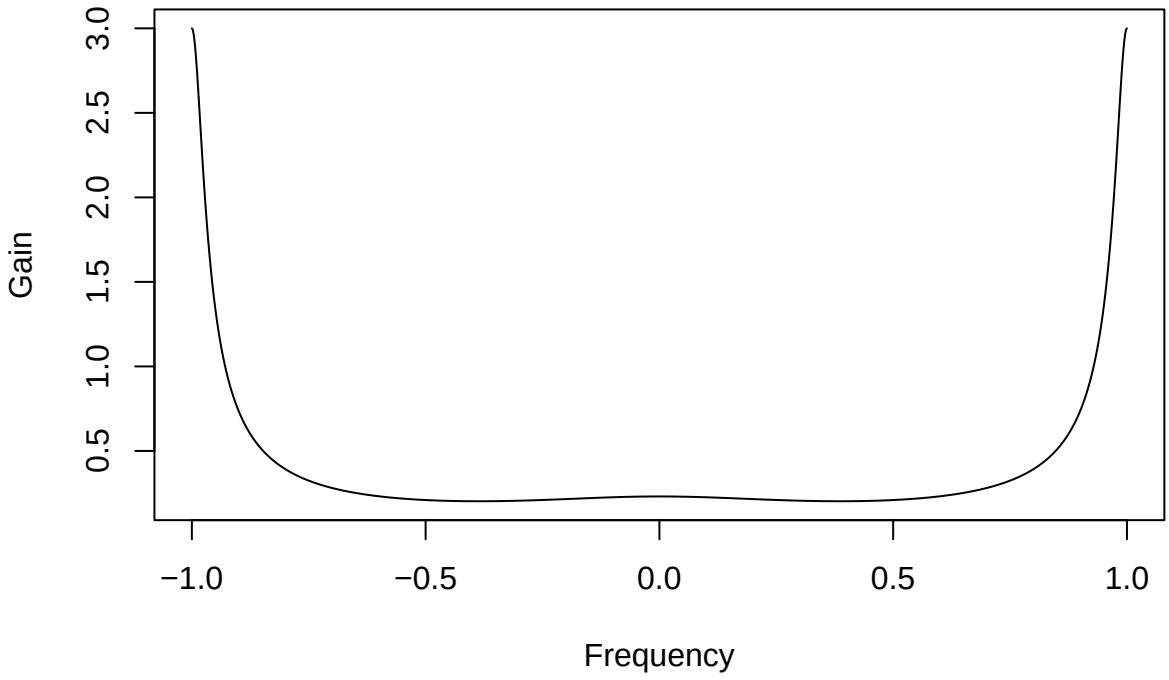
```
lambda <- pi*seq(-1000,1000)/1000
theta1 <- .6
theta2 <- -.3
h <- 2
forecast.frf <- theta2/(1 + theta1*exp(-1i*lambda) + theta2*exp(-1i*2*lambda))
forecast.gain <- Mod(forecast.frf)
forecast.phase <- atan(Im(forecast.frf)/Re(forecast.frf))
forecast.delay <- -1*forecast.phase/lambda
forecast.delay[1001] <- NA
plot(ts(Re(forecast.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Real Part Frf")
```



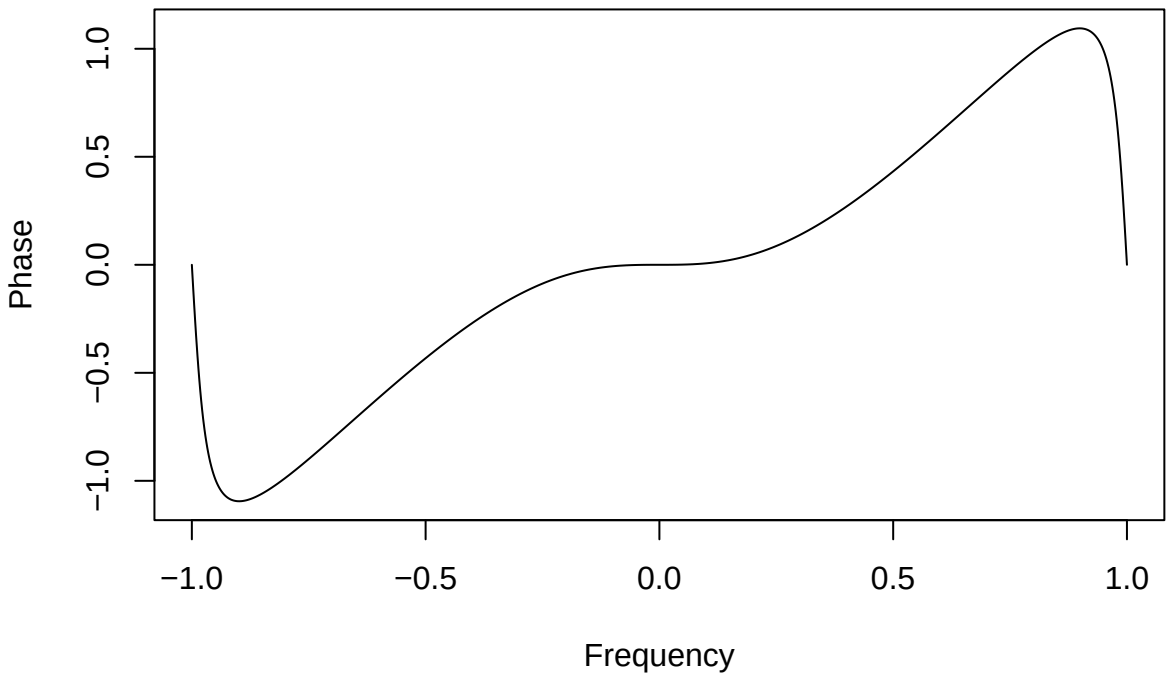

```
plot(ts(Im(forecast.frf),start=-1,frequency=1000),
    xlab="Frequency",ylab="Imaginary Part Frf")
```



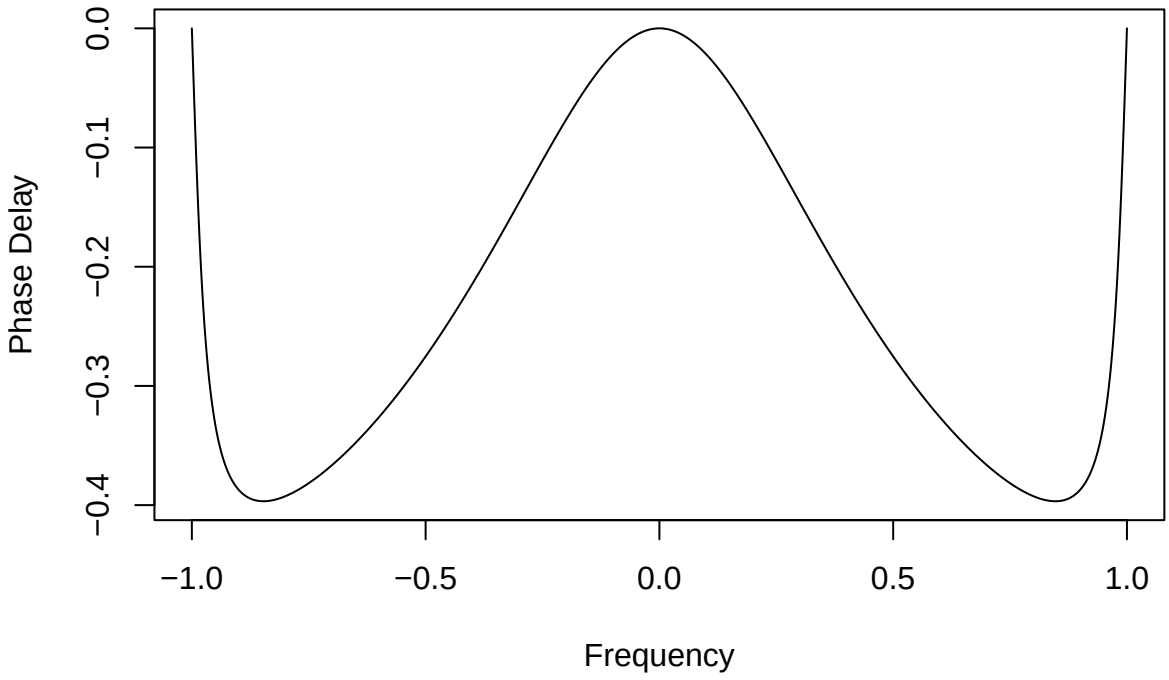

```
plot(ts(forecast.gain,start=-1,frequency=1000),
    xlab="Frequency",ylab="Gain")
```




```
plot(ts(forecast.phase,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase")
```




```
plot(ts(forecast.delay,start=-1,frequency=1000),
    xlab="Frequency",ylab="Phase Delay")
```



Lesson 7-5: Kolmogorov's Formula

- We discuss a formula for the one-step ahead prediction MSE.

Definition 7.5.2.

- For a process $\{X_t\}$, the **innovation** at time t is the prediction error given the infinite past:

$$\epsilon_t^{(\infty)} = X_t - P_{\overline{sp}\{X_s, s<t\}}[X_t].$$

- The one-step ahead prediction error based on the past n observations is

$$\epsilon_t^{(n)} = X_t - P_{\overline{sp}\{X_s, t-n\leq s<t\}}[X_t].$$

- It must be true that $\text{Var}[\epsilon_t^{(n)}] \geq \text{Var}[\epsilon_t^{(\infty)}]$.

Theorem 7.5.3.

If $\{X_t\}$ is a causal invertible ARMA with white noise inputs $\{Z_t\}$ of variance σ^2, then Z_t is the innovation at time t, and as $n \to \infty$

$$\text{Var}[\epsilon_t^{(n)}] \to \text{Var}[\epsilon_t^{(\infty)}] = \sigma^2.$$

Example 7.5.4. MA(1) Prediction Variance

- Consider 1-step ahead prediction of an invertible MA(1) process with parameter θ, based on a finite past. Invertibiilty implies $|\theta| < 1$.

- From Example 4.6.4,

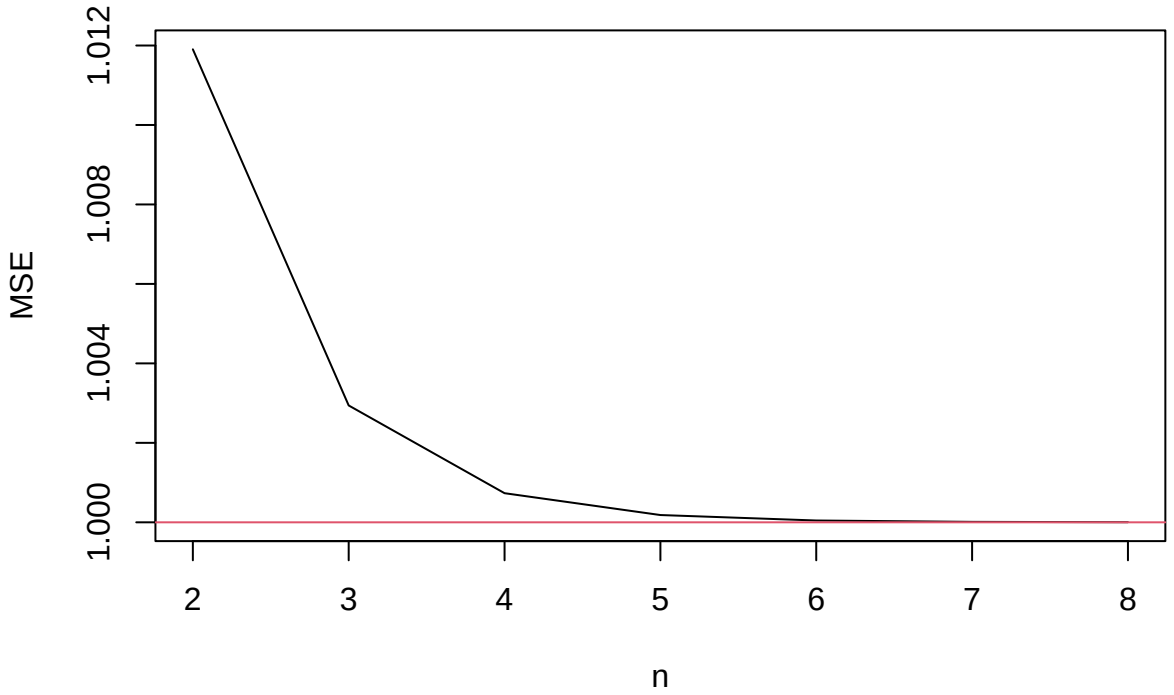
$$\text{Var}[\epsilon_t^{(n)}] = \gamma(0) - \underline{\gamma}_n' \, \Gamma_n^{-1} \, \underline{\gamma}_n = \sigma^2(1+\theta^2) - \sigma^2\theta^2 \frac{1+\theta^2+\cdots+\theta^{2n-2}}{1+\theta^2+\cdots+\theta^{2n}} = \sigma^2 \frac{1+\theta^2+\cdots+\theta^{2n+2}}{1+\theta^2+\cdots+\theta^{2n}},$$

  which is greater than σ^2 but tends to it (since $|\theta| < 1$) as $n \to \infty$.

Exercise 7.42. Prediction Error Variance Computation.

- We numerically demonstrate the formulas of Example 7.5.4, for $2 \leq n \leq 8$.
- Consider $\sigma = 1$ and $\theta = .5$:

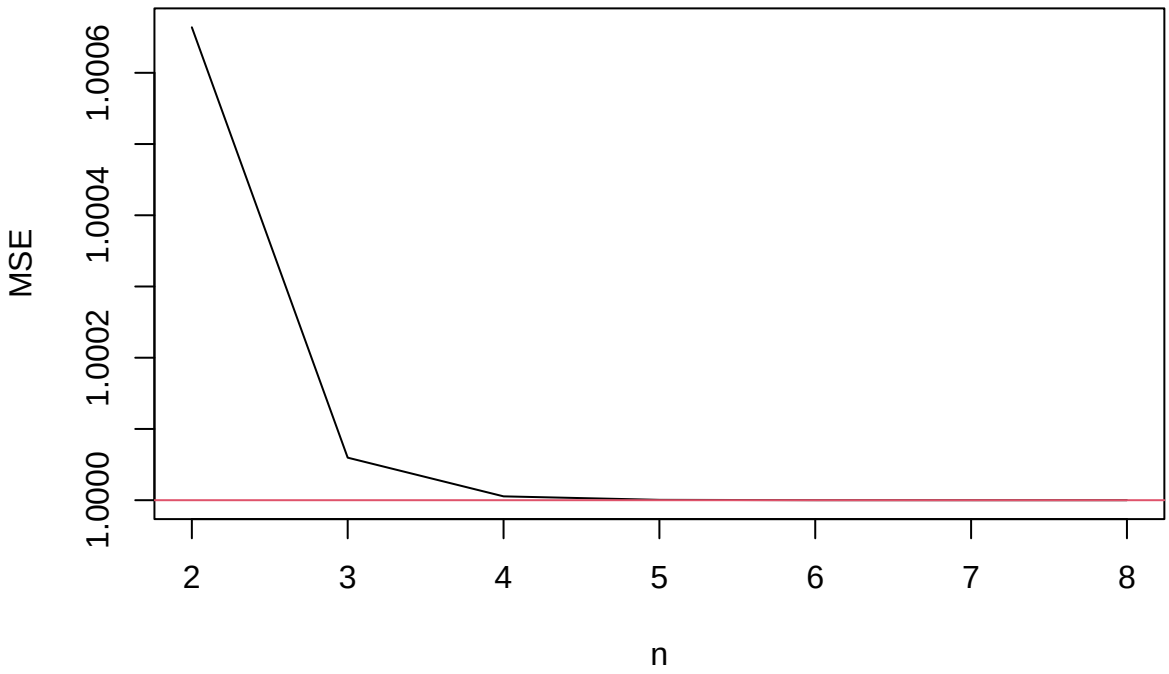
```
theta <- .5
var.eps <- NULL
for(n in 2:8)
{
    gamma <- c(1+theta^2,theta,rep(0,n-1))
    var.eps <- c(var.eps,gamma[1] - t(gamma[-1]) %*% solve(toeplitz(gamma[-length(gamma)])) %*% gamma[-
}
plot(ts(var.eps,start=2),ylab="MSE",xlab="n")
abline(h=1,col=2)
```



- Consider $\sigma = 1$ and $\theta = -.3$:

```
theta <- -.3
var.eps <- NULL
for(n in 2:8)
{
    gamma <- c(1+theta^2,theta,rep(0,n-1))
```

```
    var.eps <- c(var.eps,gamma[1] - t(gamma[-1]) %*% solve(toeplitz(gamma[-length(gamma)])) %*% gamma[-1])
}
plot(ts(var.eps,start=2),ylab="MSE",xlab="n")
abline(h=1,col=2)
```



Theorem 7.5.6. Kolmogorov's Formula

Let $\{X_t\}$ be a zero-mean stationary process with absolutely summable ACVF and spectral density f that is positive. Then the innovation variance is

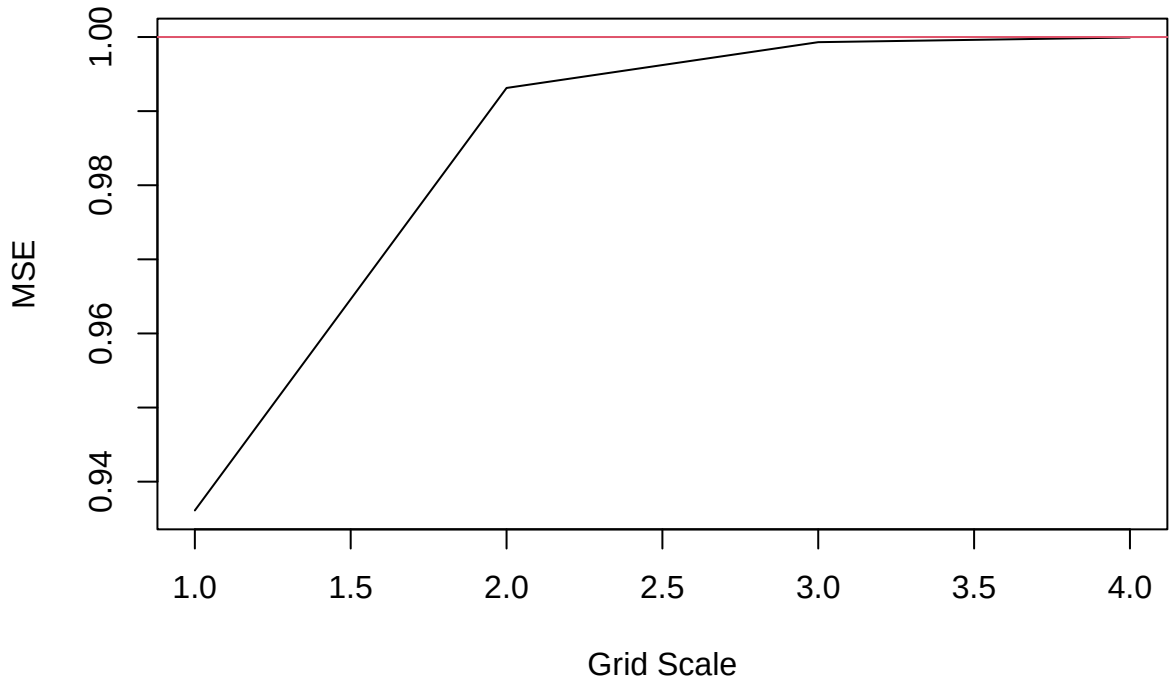
$$\text{Var}[\epsilon_t^{(\infty)}] = \exp\left\{\frac{1}{2\pi}\int_{-\pi}^{\pi} \log f(\lambda)\, d\lambda\right\}.$$

MA(1) Example

- Consider an MA(1) with $\theta = .5$ and $\sigma = 1$. We numerically compute Kolmogorov's formula, and should obtain 1 (as a verification).

```
theta <- .5
kolg <- NULL
grids <- 10^seq(1,4)
for(grid in grids)
{
  lambda <- pi*seq(-grid,grid)/grid
  spec.ma <- Mod(1 + theta*exp(-1i*lambda))^2
  kolg <- c(kolg,exp(mean(log(spec.ma))))
}
```

```
plot(ts(kolg,start=1),ylab="MSE",xlab="Grid Scale")
abline(h=1,col=2)
```



Lesson 7-6: Wold Decomposition

- We examine a decomposition of a stationary time series due to Wold.
- We also take a look at spectral factorization.

Definition 7.6.1.

- A covariance stationary time series with innovation variance equal zero is called **predictable**, since it can be forecasted without error.
- Any stochastic sinusoid is predictable: it is perfectly periodic!

Theorem 7.6.4. Wold Decomposition.

A weakly stationary mean zero time series $\{X_t\}$ that is not predictable can be decomposed into a sum of processes $\{U_t\}$ and $\{Y_t\}$ such that:

- $X_t = U_t + Y_t$.
- $\{U_t\}$ and $\{Y_t\}$ are uncorrelated with one another.
- $\{U_t\}$ is predictable.
- $\{Y_t\}$ has a MA(∞) representation $Y_t = \psi(B)Z_t$, with $Z_t \sim \text{WN}(0, \sigma^2)$.
- $\psi(z) = \sum_{j \geq 0} \psi_j z^j$ is causal with first coefficient 1 ($\psi(0) = 1$).
- Z_t is the innovation at time t of $\{Y_t\}$ and $\{X_t\}$
- σ^2 is the prediction error variance

- $\{\psi_j\}$ are called the **Wold** coefficients

Theorem D.2.7. Spectral Factorization

- Suppose $g(\lambda) = \sum_{k=0}^{q} g_k \cos(\lambda k)$ is a degree q cosine polynomial that is positive. Then there exists a degree q polynomial $\theta(z)$ with unit leading coefficient ($1 = \theta(0)$) and constant $\kappa > 0$ such that

$$g(\lambda) = \kappa |\theta(e^{-i\lambda})|^2.$$

- This can be applied to the spectral density: suppose we know the autocovariances γ_k of an MA(q) process, but not the MA coefficients. Then the spectral density is a positive cosine polynomial of degree q, and we can find $\theta(z)$, the corresponding MA polynomial.
- For large q, these MA coefficients approximate the Wold coefficients.

Exercise 7.44. Spectral Factorization of an MA(2) via Root-Finding.

- Use root-finding to do spectral factorization for an MA(2) process.
- First write code:

```
polymul <- function(a,b)
{
    bb <- c(b,rep(0,length(a)-1))
    B <- toeplitz(bb)
    B[lower.tri(B)] <- 0
    aa <- rev(c(a,rep(0,length(b)-1)))
    prod <- B %*% matrix(aa,length(aa),1)
    return(rev(prod[,1]))
}
specfact.ma2 <- function(gamma)
{
    my.roots <- polyroot(c(rev(gamma),gamma[-1]))
    ind.inv <- which(Mod(my.roots)>1)
    theta.pol <- polymul(c(1,-1/my.roots[ind.inv[1]]),c(1,-1/my.roots[ind.inv[2]]))
    theta.pol <- Re(theta.pol)
    kappa <- Re(gamma[3]*my.roots[ind.inv[1]]*my.roots[ind.inv[2]])
    return(list(theta.pol,kappa))
}
```

- Test this out on an MA(2) with $\gamma_0 = 1$, $\gamma_1 = .6$, and $\gamma_2 = .2$.

```
gamma <- c(1,.6,.2)
out <- specfact.ma2(gamma)
theta.pol <- out[[1]]
kappa <- out[[2]]
print(c(theta.pol))
```

```
## [1] 1.0000000 0.7565112 0.3372030
```

```
print(kappa)
```

```
## [1] 0.5931145
```

```
## check
print(polymul(theta.pol,rev(theta.pol))*kappa)
```

```
## [1] 0.2 0.6 1.0 0.6 0.2
```

Lesson 7-7: Cepstrum

- The AR and MA (and ARMA) processes are dense in the class of covariance stationary processes, in the sense that an arbitrary continuous spectral density can be approximated by an MA(q), AR(p), or ARMA(p,q) spectral density for sufficiently large p and q.
- Another class of spectral density is given by the "cepstrum".

Paradigm 7.7.5. The Cepstrum

- We find the Fourier transform of the log of a positive spectral density:

$$\log f(\lambda) = \sum_{k=-\infty}^{\infty} \tau_k e^{-i\lambda k}.$$

- The Fourier coefficients $\{\tau_k\}$ are called **cepstral** coefficients.
- This representation of $\log f$ is called the **cepstrum**; the word "cepstrum" is an anagram of "spectrum".
- $\log f(\lambda)$ can be negative.
- The cepstral coefficients are recovered via Fourier inversion:

$$\tau_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{i\lambda k} \log f(\lambda)\, d\lambda.$$

- Since f is even, $\tau_{-k} = \tau_k$.
- Kolmogorov: $\tau_0 = \log \sigma^2$.

Proposition 7.7.7. Cepstral Representation of Wold Coefficients

- We can relate the cepstral coefficients to the Wold coefficients, since

$$f(\lambda) = \exp\left\{\sum_{k=-\infty}^{\infty} \tau_k e^{-i\lambda k}\right\} = e^{\tau_0} \cdot \exp\left\{\sum_{k>0} \tau_k e^{-i\lambda k}\right\} \cdot \exp\left\{\sum_{k>0} \tau_k e^{i\lambda k}\right\},$$

  which suggests

$$\exp\left\{\sum_{k>0} \tau_k z^k\right\} = \sum_{j\geq 0} \psi_j z^j,$$

  where $\{\psi_j\}$ are Wold coefficients.
- This relationship provides a recursive formula for Wold in terms of cepstral:

$$\psi_j = \frac{1}{j} \sum_{k=1}^{j} k\, \tau_k\, \psi_{j-k}$$

  for $j \geq 1$, which is initialized with $\psi_0 = 1$.

Definition 7.7.9.

- The **exponential process** or order m, or EXP(m), is a causal time series $\{X_t\}$ with spectral density f given by

$$f(\lambda) = \exp\left\{\sum_{|k|\leq m} \tau_k e^{-i\lambda k}\right\}.$$

- All EXP processes are invertible!
- This defines a class of models, just like AR and MA models.
- The EXP class is also dense.

Exercise 7.46. Moving Average Representation of Cepstral Process

- We encode the cepstral to Wold mapping, and apply to an EXP(2) process.
- Our function takes a sequence $\tau_1, \ldots, \tau_m$ as inputs, and generates $\psi_1, \ldots, \psi_q$ for given q.

```
ceps2wold <- function(ceps,q)
{
    m <- length(ceps)
    if(q > m) { ceps <- c(ceps,rep(0,q-m)) }
    wold <- 1
    wolds <- wold
    for(j in 1:q)
    {
        wold <- sum(seq(1,j)*ceps[1:j]*wolds[j:1])/j
        wolds <- c(wolds,wold)
    }
    return(wolds)
}
```

- We apply this with $\tau_1 = .5$ and $\tau_2 = -.2$.

```
print(round(ceps2wold(c(.5,-.2),20),digits=4))
```

```
##  [1]  1.0000  0.5000 -0.0750 -0.0792 -0.0024  0.0061  0.0007 -0.0003 -0.0001
## [10]  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
## [19]  0.0000  0.0000  0.0000
```

Exercise 7.52. Spectral Factorization of an MA(q) via Cepstrum

- We can use the cepstrum to do a spectral factorization, as an alternative method to the root-finding approach of Exercise 7.44.
- We take as inputs the autocovariances, numerically compute the cepstral coefficients, and then apply Proposition 7.7.7 to get the Wold coefficients.

```
specceps.maq <- function(gamma,mesh)
{
    lambda <- pi*seq(0,mesh)/mesh
    q <- length(gamma)-1
    f.spec <- gamma[1]*cos(0*lambda)
    for(h in 1:q)
    {
        f.spec <- f.spec + 2*gamma[h+1]*cos(h*lambda)
    }
    theta.pol <- 1
    kappa <- 1
    if(min(f.spec)>0)
    {
        tau <- mean(log(f.spec))
        taus <- tau
        for(h in 1:q)
        {
            tau <- mean(log(f.spec)*cos(h*lambda))
            taus <- c(taus,tau)
        }
        theta.pol <- ceps2wold(taus[-1],q)
        kappa <- exp(taus[1])
    }
```

```
    return(list(theta.pol,kappa))
}
```

- This function computes cepstral coefficients.
- We test on the MA(5) with $\theta(z) = 1 + .8z + .6z^2 + .3z^3 + .5z^4 + .2z^5$ with $\sigma^2 = 8$, and show that the MA coefficients are recovered.

```
polymul <- function(a,b)
{
    bb <- c(b,rep(0,length(a)-1))
    B <- toeplitz(bb)
    B[lower.tri(B)] <- 0
    aa <- rev(c(a,rep(0,length(b)-1)))
    prod <- B %*% matrix(aa,length(aa),1)
    return(rev(prod[,1]))
}
theta.pol <- c(1,.8,.6,.3,.5,.2)
sig2 <- 8
gamma <- polymul(theta.pol,rev(theta.pol))*sig2
gamma <- gamma[6:11]
print(gamma)
```

```
## [1] 19.04 13.68  9.60  6.56  5.28  1.60
```

```
out <- specceps.maq(gamma,10000)
theta.pol <- out[[1]]
kappa <- out[[2]]
print(theta.pol)
```

```
## [1] 1.0000000 0.8000647 0.6003317 0.3004084 0.5005758 0.2005801
```

```
print(kappa)
```

```
## [1] 8.0008
```

- We can repeat with a larger mesh size, to get a closer approximation.