Dynamic Harmonic Regression

Beniamino Hadj-Amar

April 8, 2016

Observation Equation: $y_t = H_t x_t + e_t$
State Equation: $x_t = F x_{t-1} + G\eta_t$

where $e_t \sim \mathcal{N}(0, \sigma^2)$ and $\eta_t \sim \mathcal{N}(0, Q)$

Estimation of the time variable parameters

Suppose $x_{t-1}|y^{t-1} \sim \mathcal{N}(x_{t-1}|\hat{x}_{t-1}, P_{t-1})$

1 - Forward Pass Filtering Equations

Prediction

$\hat{x}_{t|t-1} = \mathbb{E}[x_t|y^{t-1}] = F\hat{x}_{t-1}$
$P_{t|t-1} = \mathbb{V}\text{ar}[x_t|y^{t-1}] = FP_{t-1}F^T + GQ_rG^T$

Correction

$\hat{x}_t = \mathbb{E}[x_t|y^t] = \hat{x}_{t|t-1} + P_{t|t-1}H_t^T\Big[1 + H_tP_{t|t-1}H_t^T\Big]^{-1}\{y_t - H_t\hat{x}_{t|t-1}\}$

$P_t = \mathbb{V}ar[x_t|y^t] = P_{t|t-1} - P_{t|t-1}H_t^T\Big[1 + H_tP_{t|t-1}H_t^T\Big]^{-1}H_tP_{t|t-1}$

Backward Pass Smoothing Equations

$\hat{x}_{t|N} = F^{-1}\Big[\hat{x}_{t+1|N} + GQ_rG^TL_t\Big]$

$L_t = \Big[I - P_{t+1}H_{t+1}^TH_{t+1}\Big]^T\Big[F^TL_{t+1} - H_{t+1}^T\{y_{t+1} - H_{t+1}\hat{x}_{t+1}\}\Big]$

$P_{t|N} = P_t + P_tF^TP_{t+1|t}^{-1}\Big[P_{t+1|N} - P_{t+1|t}\Big]P_{t+1|t}^{-1}FP_t$

where $Q_r = \dfrac{Q}{\sigma^2}$, and $P_t = \dfrac{P_t^*}{\sigma^2}$. P_t^* is the error covariance matrix associated with the state estimates.

Spectral Analysis of DHR model

IRW

Consider UC model, consisting only of an **IRW** trend plus a white noise component:

$$x_t = x_{t-1} + \nabla x_{t-1} \tag{1}$$
$$\nabla x_t = \nabla x_{t-1} + \eta_{t-1} \tag{2}$$
$$y_t = x_t + e_t \tag{3}$$

By plugging (2) in (1) we obtain:

$$\begin{aligned} x_t &= x_{t-1} + \nabla x_{t-2} + \eta_{t-1} \\ x_t &= x_{t-1} + x_{t-1} - x_{t-2} + \eta_{t-1} \\ x_t - 2x_{t-1} + x_{t-2} &= \eta_{t-1} \\ (1-L)^2 x_t &= \eta_{t-1} \\ x_t &= \frac{1}{(1-L)^2}\eta_{t-1} \end{aligned}$$

Therefore, the observational equation can be written as

$$\boxed{y_t = x_t + e_t = \frac{1}{(1-L)^2}\eta_{t-1} + e_t}$$

where L is the backward shift operator.

We take the second differences of y_t in order to capture the non stationarity of the IRW process:

$$\nabla^2 y_t = \eta_{t-1} + \nabla^2 e_t, \quad \text{where} \quad \nabla = (1-L).$$

In order to evaluate the power spectrum of the process $\nabla^2 y_t$ we need to compute its covariance function:

$$\gamma_{\nabla^2 y}(h) = \gamma_\eta(h) + \gamma_{\nabla^2 e}(h)$$

Hence,

$$\gamma_{\nabla^2 y}(h) = \begin{cases} \sigma_\eta^2 + 6\sigma^2 & \text{if } h = 0 \\ -4\sigma^2 & \text{if } |h| = 1 \\ \sigma^2 & \text{if } |h| = 2 \\ 0 & \text{if } |h| > 2 \end{cases}$$

Therefore, the **power frequency spectrum** is given by:

$$
\begin{aligned}
f_{\nabla^2 y}(\omega) &= \frac{1}{2\pi}\sum_{h\in\mathbb{Z}}\gamma_{\nabla^2 y}(h)\cdot\exp\{-i2\pi\omega h\} \\
&= \frac{1}{2\pi}\Big[\sigma_\eta^2 + 6\sigma^2 - 8\sigma^2\cos(\omega) + 2\sigma^2\cos(2\omega)\Big] \\
&= \frac{1}{2\pi}\Big[\sigma_\eta^2 + \sigma^2\{2-2\cos(\omega)\}^2\Big].
\end{aligned} \tag{4}
$$

In order to compute (4) we used the following identities:

- $2\cos(\omega) = \exp(i\omega) + \exp(-i\omega)$
- $\cos(2\omega) = \cos^2(\omega) - 1$

The pseudo-spectrum of IRW is given by

$$
f_y(\omega) = \frac{1}{2\pi}\Big[\frac{\sigma_\eta^2}{\{2-2\cos(\omega)\}^2} + \sigma^2\Big]
$$

RW

Consider UC model, with a RW trend plus a with noise component:

$$
\begin{aligned}
x_t &= x_{t-1} + \eta_{t-1} \\
y_t &= x_t + e_t
\end{aligned}
$$

Hence, the observational equation can be written as

$$
\boxed{y_t = \frac{1}{1-L}\eta_{t-1} + e_t}
$$

We can look at the first differences to study the inhernet non stationarity of the process:

$$
\nabla y_t = \eta_{t-1} + \nabla e_t
$$

The pseudo-spectrum in the case of the RW trend is given by

$$
f_y(\omega) = \frac{1}{2\pi}\Big[\frac{\sigma_\eta^2}{\{2-2\cos(\omega)\}} + \sigma^2\Big]
$$

The code to evaluate the IRW and RW pseudo-spectrum is given by:

```
# Pseudo-Spectrum Generalised Random Walk (GRW)
pseudo.spectrum.GRW <- function(omega, sigma2.eta, sigma2,
                                type = "IRW", alpha = 1) {
  if(type == "IRW") {
    alpha <- 1
  }
  else if(type == "RW") {
    alpha <- 0
  }

  spectrum <- (1/(2*pi)) * (  (sigma2.eta/((1 + (alpha^2) - (2*alpha*cos(omega)) ) *
                                           (2 - 2*cos(omega)))) + sigma2   )
  return(spectrum)
}
```

We show in 1 properties of the pseudo spectra of RW and IRW process, for different value of NVR $\frac{\sigma_\eta^2}{\sigma^2}$:

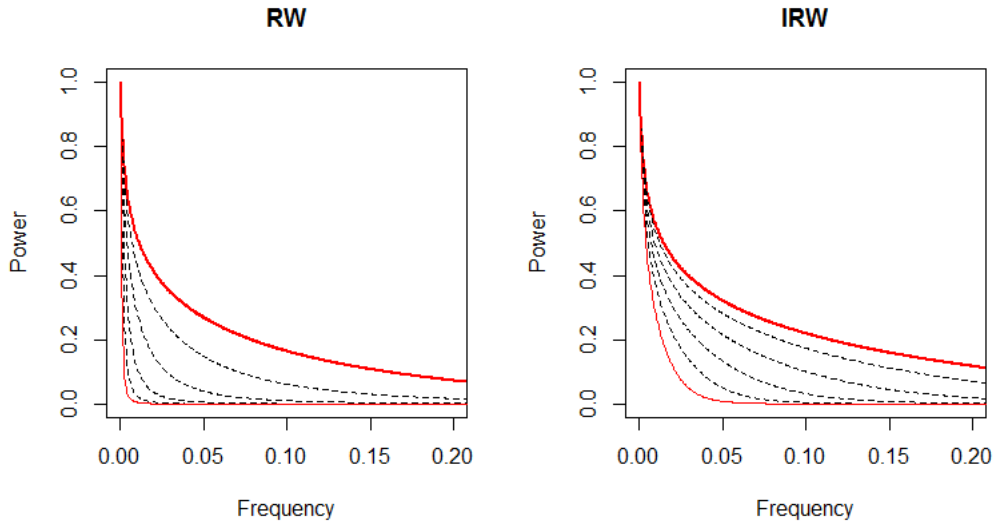

Figure 1: Pseudo-Spectrum RW and IRW: thick red line is for value of NVR of 10^{-1}; thin red line is for value of NVR of 10^{-6}. Also, we took the log of the spectrum, and normalise it to the range [0, 1]

Pseudo-spectra of the Full DHR Model

If we consider a single frequency ω_j DHR term of the form $S_t = a_t \cos(\omega_j t) + b_t \sin(\omega_j t)$, where a_t and b_t follow an IRW or RW process with equal variance parameters $\sigma^2_{\omega_j}$

The pseudo spectrum for the IRW and RW of S_t take the form:

IRW components

$$f_{\omega_j}(\omega) = \frac{1}{2\pi}\left[\frac{\sigma^2_{\omega_j}}{4\{1-\cos(\omega-\omega_j)\}^2} + \frac{\sigma^2_{\omega_j}}{4\{1-\cos(\omega+\omega_j)\}^2}\right]$$

RW components

$$f_{\omega_j}(\omega) = \frac{1}{2\pi}\left[\frac{\sigma^2_{\omega_j}}{2\{1-\cos(\omega-\omega_j)\}} + \frac{\sigma^2_{\omega_j}}{2\{1-\cos(\omega+\omega_j)\}}\right]$$

The code to compute these pseudo-spectrums is given by:

```
# Function to compute pseudo-spectrum for a single frequency DHR term

pseudo.spectrum.single <- function(omega, omega.j, sigma2, type) {

  if(type == "IRW") {
    out <- (1/(2*pi)) * ((sigma2/(4*(1 - cos(omega - omega.j))^2)) +
                           (sigma2/(4*(1 - cos(omega + omega.j))^2)))
    return(out)
  }

  else if(type == "RW") {
    out <- (1/(2*pi)) * ((sigma2/(2*(1 - cos(omega - omega.j)))) +
                           (sigma2/(2*(1 - cos(omega + omega.j)))))
    return(out)
  }
}
```

We now introduce $S(\omega, \omega_j)$ as follows (just for the IRW case, now the RW is obvious)

$$S(\omega,\omega_j) = \frac{1}{2\pi}\left[\frac{1}{4\{1-\cos(\omega-\omega_j)\}^2} + \frac{1}{4\{1-\cos(\omega+\omega_j)\}^2}\right]$$

```
# Function to compute: S(omega, omega.j)
S <- function(omega, omega.j, type) {

  if(type == "IRW") {
    out <- (1/(2*pi)) * ((1/(4*(1 - cos(omega - omega.j))^2)) +
                           (1/(4*(1 - cos(omega + omega.j))^2)))
    return(out)
  }

  else if (type == "RW") {
    out <- (1/(2*pi)) * ((1/(2*(1 - cos(omega - omega.j)))) +
                           (1/(2*(1 - cos(omega + omega.j)))))
    return(out)
  }
}
```

We show the spectrum of AR(14) fitted to the data `AirPassengers`, and the pseudo spectrum of the single RW harmonic components, and the IRW trend component in 2

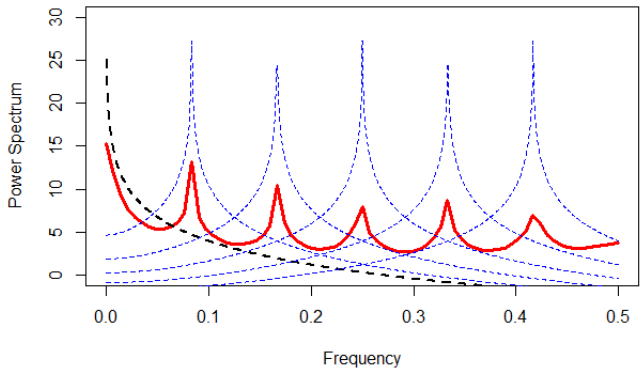

Figure 2: Red line: AR(14) fitted to the AirPassanger data. Black line: trend IRW component. Blue line: RW harmonic components. σ^2 is 1, whereas all the other $\sigma^2_{\omega_j}$ are 1/10. We plot the spectrum in log scale.

The code to realise this experiment is given as follows:

```
####### Pseudo-spectra DHR model (AirPassengers Data)
library(astsa)
data("AirPassengers")

# Variances: Obs variance, Trend variance, harmonic component variances
sigma <- c(1, 1/10, rep(1/10, 5))

# Trend frequency, Harmonic components frequencies
omega.j <- c(0, 1/12, 1/6, 1/4, 1/3, 1/(2.4))


# Finding AR(p) to approximate spectrum of AirPassanger
performance.ar.p(AirPassengers, p = 20, plot = TRUE, legend = F)

# Fitting AR(14) to AirPassangers
p <- 14
fit <- arima(AirPassengers, order = c(p, 0, 0), method="ML")
coef <- fit$coef[1:p]

# Spectrum AR(14)
ar.fit <- arma.spec(ar = coef, n.freq = length(freq) + 1,
                    var.noise = fit$sigma2)

# Plotting log(spectrum)
par(mfrow = c(1, 1))
plot(ar.fit$freq, log(ar.fit$spec), type = "l", col = "red", lwd = 3,
     xlab = "Frequency", ylab = "Power Spectrum", ylim = c(0, 30))

# Adding lines of the pseudo spectrum of IRW trend, and RW
# harmonic components.
lines(freq, log(sigma[2] * S(freq, omega.j[1], type = "IRW")/(2*pi)),
      lty = 2, lwd = 2)
for(i in 2:6) {
  lines(freq, log(sigma[i + 1] * S(freq, omega.j[i], type = "IRW")/(2*pi)),
        lty = 2, col = "blue")
  }
```

Then, the pseudo-spectrum of the full DHR model becomes:

$$\boxed{f_y(\omega, \underline{\sigma}^2) = \sum_{j=0}^{R} \left[\sigma^2_{\omega_j} S(\omega, \omega_j)\right] + \frac{\sigma^2}{2\pi}} \quad \text{where } \underline{\sigma}^2 = [\sigma^2, \sigma^2_{\omega_0}, \ldots, \sigma^2_{\omega_R}] \tag{5}$$

The code to compute the full DHR model with IRW trend, and RW harmonic components of (5)

```
###### Function to compute pseudo-spectrum full DHR (specific example)

# sigma: vector of variances
#        1st component is the observational variance
#        2nd component is the trend variance
#        3rd...Rth components are harmonic variances.

# omega.j: vector of fondamental frequency and harmonics
#          omega.j[1] must be 0 (if the trend is included)

fullDHR.spectrum <- function(omega, sigma, omega.j) {

  R <- length(omega.j) - 1

  # Trend component:
  spectrum <- sigma[2] * S(omega, omega.j[1], type = "IRW")
  print(spectrum)

  # Harmonic components:
  for(j in 1:R) {
    spectrum = spectrum + sigma[j + 2] * S(omega, omega.j[j + 1], type = "RW")
  }

  return(spectrum + (sigma[1]/(2*pi)))
}
```

Estimation in the Frequency Domain

Equation (5) can be considered as a linear combination of known spectral terms, with the unknown variance parameters appearing as regression coefficients. To estimate $\underline{\sigma^2}$ we can minimise a cost function of the form

$$E(\underline{\sigma}^2) = \sum_{i=0}^{T-1} \left[f_y(\omega_i) - \hat{f}_y(\omega_i, \underline{\sigma}^2)\right]^2$$

where $f_y(\omega)$ is the empirical spectrum, for example the spectrum otbained by fitting an AR(p) process to the data y. The ω_i are in the range 0 to 0.5, $i = 0, 1, 2, \ldots, T-1$.

An alternative error function, which is easier to minimise is the one defined on logarithmic terms:

$$E(\underline{\sigma}^2)_L = \sum_{i=0}^{T-1} \left[\log[f_y(\omega_i)] - \log[\hat{f}_y(\omega_i, \underline{\sigma}^2)]\right]^2$$

One final simplification is possible: if the empirical spectrum is defined on the estimated AR(p) spectrum, then σ^2, the estimate of the residual white noise can be used to get:

$$f_y(\omega, \underline{NVR}) = \sigma^2 \Big[\sum_{j=0}^{R} \text{NVR}_j S(\omega, \omega_j) + \frac{1}{2\pi} \Big],$$

where **NVR** is the vector with elements $NVR_j = \dfrac{\sigma^2_{\omega_j}}{\sigma^2}$, $j = 0, 1, \ldots, R$

The Complete DHR Estimation Algorithm

In the case where a cyclical component is not present, the complete DHR algorithm can be expressed in four steps.

1. AR(p) spectrum estimation

Estimate an AR(p) spectrum $f_y(\omega)$ and its associated residuals variance $\hat{\sigma}^2$, with the order p identified by reference to AIC or BIC. In this way we can also choose the R frequency ω_j that drives the data.

The code for choosing p is given by:

```
performance.ar.p <- function(residuals.data, p, plot = TRUE, legend = TRUE) {

  T <- length(residuals.data)

  AIC <- rep(0, p)
  BIC <- rep(0, p)

  for(k in 1:p) {
    fit <- ar(residuals.data, order = k, aic = FALSE)
    sigma2 <- var(fit$resid, na.rm = TRUE)
    BIC[k] <- log(sigma2) + (k*log(T)/T)
    AIC[k] <- log(sigma2) + ((T+2*k)/T)
  }

  if (plot == TRUE) {
    IC <- cbind(AIC, BIC)
    ts.plot(IC, type = "o", xlab = "p",
            ylab = "AIC and BIC", col = c("red", "blue"),
            lwd = 2)
    points(which(BIC == min(BIC)), BIC[which(BIC == min(BIC))],
           col = "black", pch = 19, lwd = 4)
    points(which(AIC == min(AIC)), AIC[which(AIC == min(AIC))],
           col = "black", pch = 19, lwd = 4)
    if (legend == TRUE) {
    legend("bottomright", c("AIC", "BIC"), lty = 1, lwd = 2,
           col = c("red", "blue"))
    }
  }

  return(list(AIC = AIC, BIC = BIC))
}
```

After having choosen p, the function to estimate the AR spectrum is given by below. This function requires as argoument the data to fit to an AR, the lag p, and the number of frequencies used. In order to avoid numerical problem, I decided to not consider frequency $\omega_i = 0$. Indeed, given T as number of frequency, this function will return the spectrum of (T-1) frequencies, the (T - 1) frequencies, and the estimated residual variance σ^2.

```r
#### Function Step 1: Estimate AR(p) spectrum of the observation process
require(astsa)
estimate.AR <- function(data, p, n.freq) {

  # Fitting AR(p) to data, getting the coefficients and the residual variance
  fit <- arima(data, order = c(p, 0, 0), method = "ML")
  coef <- fit$coef[1:p]
  sigma2 <- fit$sigma2

  # Getting spectrum AR(p)
  ar.fit <- arma.spec(ar = coef, n.freq = n.freq, var.noise = fit$sigma2)

  # Not considering frequency 0
  spectrum <- ar.fit$spec[-1]
  freq <- ar.fit$freq[-1]

  return(list(spectrum = spectrum, frequency = freq , sigma2 = sigma2))
}
```

2. Linear Least Square estimate of the NVR parameters

Let $\mathbf{Y}$ be the vector with components

$$y_i = f_y(\omega_i) - \frac{\sigma^2}{2\pi}, \quad i = 1, \ldots, T-1$$

Let $\mathbf{X}$ be the matrix, with elements:

$$x_{ij} = \sigma^2 S(\omega_i, \omega_j), \quad j = 1, \ldots, R$$

In this way $\mathbf{Y} = \mathbf{X}\cdot\mathbf{NVR} + \epsilon$, and the linear least square estimate of NVR is given by:

$$\widehat{\mathbf{NVR}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{Y}$$

The code for performing this step is given below. This function will return the NVR estimates.

```r
#### Function Step 2: Find Linear Least Square estimate of NVR parameter vector

NVR.LinearLS <- function(empirical.spectrum, sigma2,
                         frequency, omega.j, structures) {

  # Number of frequencies
  T <- length(frequency)
  # Number of harmonic components
  R <- length(omega.j) - 1

  # Preparing data for evaluating LLS NVR estimate
  Y <- empirical.spectrum - (sigma2/(2*pi))
  X <- matrix(NA, nrow = T, ncol = (R + 1))

  for(r in 1:(R + 1)) {
    X[, r] <- S(frequency, omega.j[r], type = structures[r])
  }

  X <- sigma2 * X

  # Least square estimate
  NVR.estimate <- solve(t(X) %*% X, t(X) %*% Y)

  return(NVR.estimate)
}
```

Testing Step 2

I am not sure if the results are correct, because I get different numbers from the paper. But it looks to me that the code, and the procedure is right. To see indeed if the results that I got are reasonable, I evaluated and plotted the DHR spectrum, with the estimated $\underline{\hat{\sigma}}^2$. In Figure 3 we show the true AR(14) spectrum fitted to the `AirPassangers` Data with the red line; the green line is the spectrum given by (5), with $\underline{\hat{\sigma}}^2$ estimated by LLS. The results look sensible.

```
# Testing steps of Complete DHR Estimation Algorithm

# Getting data
data.AP <- AirPassengers
# Harmonics
omega.j.AP <- c(0, 1/12, 1/6, 1/4, 1/3, 1/(2.4))
# IRW Trend, RW harmonic components
structures.AP <- c("IRW", rep("RW", 5))

# Getting spectrum and residuals variance from AR(P) fitted to AirPassengers
AP.AR14 <- estimate.AR(data = data.AP, p = 14, n.freq = length(data.AP))
freq <- AP.AR14$frequency
spec <- AP.AR14$spectrum
sigma2 <- AP.AR14$sigma2

# Estimating NVR with Step2 (LLS)
NVR.LSS.estimate <- NVR.LinearLS(empirical.spectrum = spec,
                                 sigma2 = sigma2,
                                 frequency = frequency,
                                 omega.j = omega.j.AP,
                                 structures = structures.AP)

# Sigma estimate
sigma.est <- c(sigma2, sigma2 * NVR.LSS.estimate)

# Plotting AR(14) spectrum and DHR spectrum with estimated sigma (LLS)
plot(freq, log(spec), type = "l", col = "red", lwd = 2)
lines(freq, log(fullDHR.spectrum(freq,  sigma.est, omega.j.AP)),
      lwd = 2, col = "green")
```

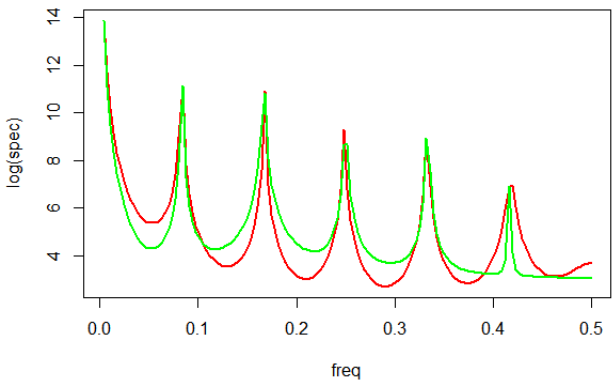

Figure 3: Red line: AR(14) spectrum for `AirPassangers`. Green line: full DHR spectrum with estimated variances from Step 2.

3. Optimisation of Logaritmich Loss

In this step the we optimise

$$E(\mathbf{NVR})_L = \sum_{i=0}^{T-1} \Big[\log[f_y(\omega_i)] - \log[\hat{f}_y(\omega_i, NVR)]\Big]^2,$$

using as initial conditions the values of $\widehat{\mathbf{NVR}}$ obtained in Step 2.

We give below the code to implement the logarithmic loss function, and we show that the estimate of this non linear function, has improved the fitting to the AR(14) spectrum.

```
#### Function Step 3:  evaluate the logarithmic error.

E <- function(NVR, freq, empirical.spectrum, omega.j, structures, sigma2) {


  # Number of frequencies
  T <- length(freq)
  # Number of harmonic components
  R <- length(omega.j) - 1

  # Matrix T * (R + 1) with components X[i, j] <- S(omega_i, omega_j)
  X <- matrix(NA, nrow = T, ncol = (R + 1))
  for(r in 1:(R + 1)) {
    X[, r] <- S(freq, omega.j[r], type = structures[r])
  }

  fitted <- sigma2 * ((X %*% NVR) + (1/(2*pi)))

  # Evaluating E(sigma)_L;
  error <- sum((log(empirical.spectrum) -
                  log(fitted))^2)

  return(error)
}
```

Testing Step 3 - Optimisation of Logaritmich Loss

We optimise the error function given above, by using `optim`.

We obtained some warnings, like `In log(fitted) :  NaNs produced`. But the results is reassuring, given that the paper doesn't seems to do better.

```
# Optimising logarithmic error: (THERE ARE WARNINGS)
optimising.NVR <- optim(NVR.LSS.estimate, E,
                        freq = freq, empirical.spectrum = spec,
                        omega.j = omega.j.AP, structures = structures.AP,
                        sigma2 = sigma2, method = "Nelder-Mead")

# Final NVR estimate
NVR.nonlinear.estimate <- optimising.NVR$par

# Final sigma^2 estimate
final.sigma.est <- c(sigma2, sigma2 * NVR.nonlinear.estimate)

# Plotting AR(14) spectrum and DHR spectrum with estimated sigma,
# by using log loss function

plot(freq, log(spec), type = "l", col = "red", lwd = 2)
lines(freq, log(fullDHR.spectrum(freq,  final.sigma.est, omega.j.AP)),
      lwd = 2, col = "green")
```

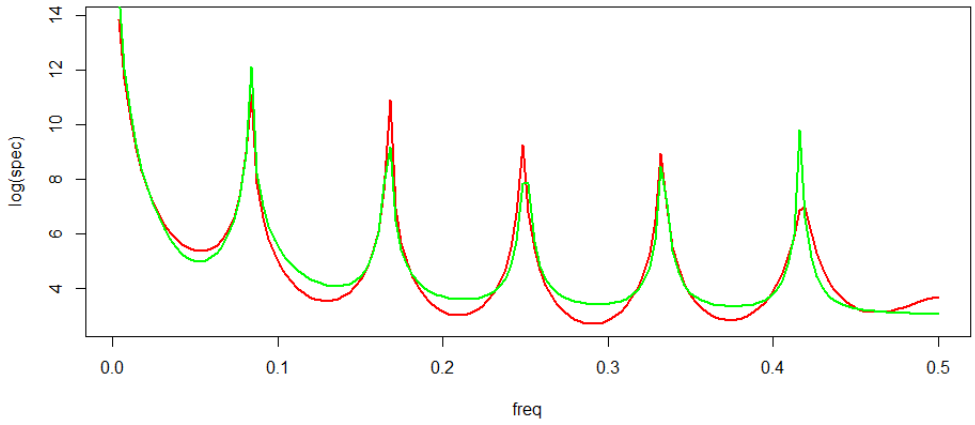

Figure 4: Red line: AR(14) spectrum for `AirPassangers`. Green line: full DHR spectrum with estimated variances from Step 3

4. Kalman Filter and Smoothing

This step consists on the filtering and the smoothing techniques described at the beginning of the paper.

Filtering

Let be R be the number of harmonic components used, and let $n = 2(R+1)$

The arguments for the Kalman Filter function are

- `y`, the observed time series

- `F`, the system matrix. It's a block diagonal n×n matrix, where each 2×2 sub matrix is of the form:

$$F_r = \begin{bmatrix} \alpha & \beta \\ 0 & \gamma \end{bmatrix}, \quad r = 0, 1, \ldots, R.$$

  For **IRW**, we have $\alpha = \beta = \gamma = 1$. For **RW**, we have $\alpha = \gamma = 1$ and $\beta = 0$.

- `H`, the observational matrix. It's a $1 \times n$ matrix. It basically describes the DHR model. It has to be a list of T matrixes, where

$$H_t = [\cos(t\omega_0), \sin(t\omega_0), \ldots, \cos(t\omega_R), \sin(t\omega_R)]$$

- `G`, the covariance matrix that control the system noise. It's a $n \times n$ matrix block diagonal matrix, with components:

$$G_r = \begin{bmatrix} \delta & 0 \\ 0 & 1 \end{bmatrix}, \quad r = 0, 1, \ldots, R.$$

  For **IRW**, we have $\delta = 0$. For **RW**, we have $\delta = 1$

- `Q`, the covariance matrix of the system noise. It's a $n \times n$ diagonal matrix, with diagonal components:

$$[\sigma^2_{\omega_0}, \sigma^2_{\omega_0}, \ldots, \sigma^2_{\omega_R}, \sigma^2_{\omega_R}]$$

- `sigma2` is σ^2, the observational variance, estimated in Step 1, from the AR(p) fitting.

The code to perform the Kalman Filter, and some auxiliary functions are given below. The function `Kalman.Filter` return an object of class `Kalman.Filter` which is a list with several elements, for which the most relevant ones are

- `states`, a list of filtered states $\mathbf{x}_t$, for $t = 1, \ldots, T$.

- `covariance.states`, a list of filtered (scaled) covariance matrixes $\dfrac{P_t}{\sigma^2}$, for $t = 1, \ldots, T$.

Other objects are returned from this function, which we'll be useful for the smoothing.

```
####### Kalman Filter:  Forward Pass Filtering #########

require(Matrix)
# Function to implement Forward Pass Filtering Equations (Kalman)

Kalman.Filter <- function(y, F, H, G, Q, sigma2) {

  # Length of the time series
  T <- length(y)

  # Number of states  [(Trend + N.Harmonics) * 2]
  n <- length(omega.j) * 2


  # Checking dimensionality
  if(!( all(dim(F) == c(n, n)) & all(dim(G) == c(n, n)) &
        all(dim(Q) == c(n, n)) & all(dim(H[[T]]) == c(1, n)))) {
    stop("Matrix dimensions are incorrect")
  }

  # States
  states <- list()
  states[[1]] <- matrix(rep(0, n), nrow = n, ncol = 1)

  # Covariances matrix
  covariance.states <- list()
  covariance.states[[1]] <- diag(n)

  # Q.r: matrix containing (scaled) system noise
  Q.r <- Q / sigma2

  # Intermediate calculation steps
  prediction.x <- list()
  prediction.P <- list()


  ### Forward Pass Filtering Equations ###

  for(t in 2:T) {
    # Prediction
    prediction.x[[t]] <- F %*% states[[t - 1]]

    prediction.P[[t]] <- F %*% covariance.states[[t - 1]] %*% t(F) +
      G %*% Q.r %*% t(G)
```

```
    # Correction
    states[[t]] <- prediction.x[[t]] + prediction.P[[t]] %*% t(H[[t]]) %*%
      solve(1 + H[[t]] %*% prediction.P[[t]] %*% t(H[[t]]),
            y[t] - H[[t]] %*% prediction.x[[t]])

    covariance.states[[t]] <- prediction.P[[t]] - prediction.P[[t]] %*% t(H[[t]]) %*%
      solve(1 + H[[t]] %*% prediction.P[[t]] %*% t(H[[t]]),
            H[[t]] %*% prediction.P[[t]])
  }

  filtered.process <- list(states = states,
                           covariance.states = covariance.states,
                           prediction.x = prediction.x,
                           prediction.P = prediction.P,
                           y = y, F = F, H = H,
                           G = G, Q = Q, sigma2 = sigma2)

  class(filtered.process) <- "Kalman.Filter"

  return(filtered.process)
}
```

```
# Function to compute the fitted value of y, given the states,
# and the model.

fitted.DHR <- function(H, states) {
  T <- length(H)
  fitted <- c()
  for(t in 1:T) {
    fitted[t] <- as.numeric(H[[t]] %*% states[[t]])
  }
  return(fitted)
}

# Function to create the time variable vector H
get.H <- function(omega.j, T) {
  H <- list()
  for(t in 1:T) {
    H[[t]] <- matrix(c(rbind(cos(omega.j * t), sin(omega.j * t))), nrow = 1)
  }
  return(H)
}
```

Testing Step 4 - Filtering

```
####### Preparing system, and observational matrixes #####
R <- 1 + 5 # Trend + Harmonic TVP's
n <- 2*R
omega.j <- c(0, 1/12, 1/6, 1/4, 1/3, 1/(2.4))

# Matrix H
H <- get.H(omega.j, T)

# Matrix F
F.IRW <- matrix(c(1, 0, 1, 1), nrow = 2, ncol = 2)
F.RW <- matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2)
F <- bdiag(c(list(F.IRW), replicate((R - 1), F.RW, simplify = FALSE)))

# Matrix G
G.IRW <- matrix(c(0, 0, 0, 1), nrow = 2, ncol = 2)
G.RW <- matrix(c(1, 0, 0, 1), nrow = 2, ncol = 2)
G <- bdiag(c(list(G.IRW), replicate((R - 1), G.RW, simplify = FALSE)))
```

```r
# Matrix Q
Q <- diag(rep(sigma2 * NVR.LSS.estimate, 1, each = 2))



#### Filtering:
filtered.process <- Kalman.Filter(y = data.AP, F = F, H = H, G = G,
                                  Q = Q, sigma2 = sigma2)

states <- filtered.process$states
covariance.states <- filtered.process$covariance.states

# Fitted values
fitted <- fitted.DHR(H, states)

# Plotting true and fitted value (after Filtering)
plot(1:t, data.AP, type = "o")
lines(1:t, fitted, type = "o", col = "red")
```

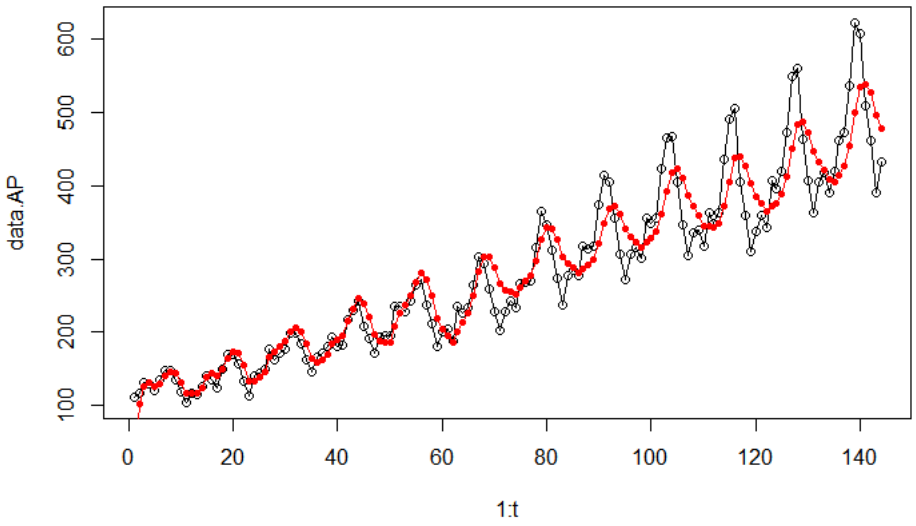

Figure 5: Filtered fitted values to the `AirPassangers` data