Estimation of Spectra

Jesse Perla, Thomas J. Sargent and John Stachurski

December 4, 2020

1 Contents

- Overview 2
- Periodograms 3
- Smoothing 4
- Exercises 5
- Solutions 6

2 Overview

In a previous lecture we covered some fundamental properties of covariance stationary linear stochastic processes.

One objective for that lecture was to introduce spectral densities — a standard and very useful technique for analyzing such processes.

In this lecture we turn to the problem of estimating spectral densities and other related quantities from data.

Estimates of the spectral density are computed using what is known as a periodogram — which in turn is computed via the famous fast Fourier transform.

Once the basic technique has been explained, we will apply it to the analysis of several key macroeconomic time series.

For supplementary reading, see [2] or [1].

2.1 Setup

```
In [1]: using InstantiateFromURL
        # optionally add arguments to force installation: instantiate = true,␣
 ↪precompile = true
        github_project("QuantEcon/quantecon-notebooks-julia", version = "0.8.0")
```

```
In [2]: using LinearAlgebra, Statistics
```

3 Periodograms

Recall that the spectral density f of a covariance stationary process with autocorrelation function γ can be written

$$f(\omega) = \gamma(0) + 2 \sum_{k \geq 1} \gamma(k) \cos(\omega k), \qquad \omega \in \mathbb{R}$$

Now consider the problem of estimating the spectral density of a given time series, when γ is unknown.

In particular, let $X_0, \dots, X_{n-1}$ be n consecutive observations of a single time series that is assumed to be covariance stationary.

The most common estimator of the spectral density of this process is the *periodogram* of $X_0, \dots, X_{n-1}$, which is defined as

$$I(\omega) := \frac{1}{n} \left| \sum_{t=0}^{n-1} X_t e^{it\omega} \right|^2, \qquad \omega \in \mathbb{R} \tag{1}$$

(Recall that $|z|$ denotes the modulus of complex number z)

Alternatively, $I(\omega)$ can be expressed as

$$I(\omega) = \frac{1}{n} \left\{ \left[\sum_{t=0}^{n-1} X_t \cos(\omega t) \right]^2 + \left[\sum_{t=0}^{n-1} X_t \sin(\omega t) \right]^2 \right\}$$

It is straightforward to show that the function I is even and 2π-periodic (i.e., $I(\omega) = I(-\omega)$ and $I(\omega + 2\pi) = I(\omega)$ for all $\omega \in \mathbb{R}$).

From these two results, you will be able to verify that the values of I on $[0, \pi]$ determine the values of I on all of $\mathbb{R}$.

The next section helps to explain the connection between the periodogram and the spectral density.

3.1 Interpretation

To interpret the periodogram, it is convenient to focus on its values at the *Fourier frequencies*

$$\omega_j := \frac{2\pi j}{n}, \quad j = 0, \dots, n-1$$

In what sense is $I(\omega_j)$ an estimate of $f(\omega_j)$?

The answer is straightforward, although it does involve some algebra.

With a bit of effort one can show that, for any integer $j > 0$,

$$\sum_{t=0}^{n-1} e^{it\omega_j} = \sum_{t=0}^{n-1} \exp\left\{ i 2\pi j \frac{t}{n} \right\} = 0$$

Letting $\bar{X}$ denote the sample mean $n^{-1} \sum_{t=0}^{n-1} X_t$, we then have

$$nI(\omega_j) = \left| \sum_{t=0}^{n-1} (X_t - \bar{X}) e^{it\omega_j} \right|^2 = \sum_{t=0}^{n-1} (X_t - \bar{X}) e^{it\omega_j} \sum_{r=0}^{n-1} (X_r - \bar{X}) e^{-ir\omega_j}$$

By carefully working through the sums, one can transform this to

$$nI(\omega_j) = \sum_{t=0}^{n-1} (X_t - \bar{X})^2 + 2 \sum_{k=1}^{n-1} \sum_{t=k}^{n-1} (X_t - \bar{X})(X_{t-k} - \bar{X}) \cos(\omega_j k)$$

Now let

$$\hat{\gamma}(k) := \frac{1}{n} \sum_{t=k}^{n-1} (X_t - \bar{X})(X_{t-k} - \bar{X}), \qquad k = 0, 1, \dots, n-1$$

This is the sample autocovariance function, the natural “plug-in estimator” of the autocovariance function γ.

(“Plug-in estimator” is an informal term for an estimator found by replacing expectations with sample means)

With this notation, we can now write

$$I(\omega_j) = \hat{\gamma}(0) + 2 \sum_{k=1}^{n-1} \hat{\gamma}(k) \cos(\omega_j k)$$

Recalling our expression for f given above, we see that $I(\omega_j)$ is just a sample analog of $f(\omega_j)$.

3.2 Calculation

Let’s now consider how to compute the periodogram as defined in (1).

There are already functions available that will do this for us — an example is `periodogram` in the `DSP.jl` package.

However, it is very simple to replicate their results, and this will give us a platform to make useful extensions.

The most common way to calculate the periodogram is via the discrete Fourier transform, which in turn is implemented through the fast Fourier transform algorithm.

In general, given a sequence $a_0, \dots, a_{n-1}$, the discrete Fourier transform computes the sequence

$$A_j := \sum_{t=0}^{n-1} a_t \exp\left\{ i 2\pi \frac{tj}{n} \right\}, \qquad j = 0, \dots, n-1$$

With $a_0, \dots, a_{n-1}$ stored in Julia array `a`, the function call `fft(a)` returns the values $A_0, \dots, A_{n-1}$ as a Julia array.

It follows that, when the data $X_0, \dots, X_{n-1}$ are stored in array `X`, the values $I(\omega_j)$ at the Fourier frequencies, which are given by

$$\frac{1}{n}\left|\sum_{t=0}^{n-1} X_t \exp\left\{i2\pi\frac{tj}{n}\right\}\right|^2, \qquad j = 0, \dots, n-1$$

can be computed by `abs(fft(X)).^2 / length(X)`.

Note: The Julia function `abs` acts elementwise, and correctly handles complex numbers (by computing their modulus, which is exactly what we need).

A function called `periodogram` that puts all this together can be found here.

Let's generate some data for this function using the `ARMA` type from QuantEcon.jl (see the lecture on linear processes for more details).

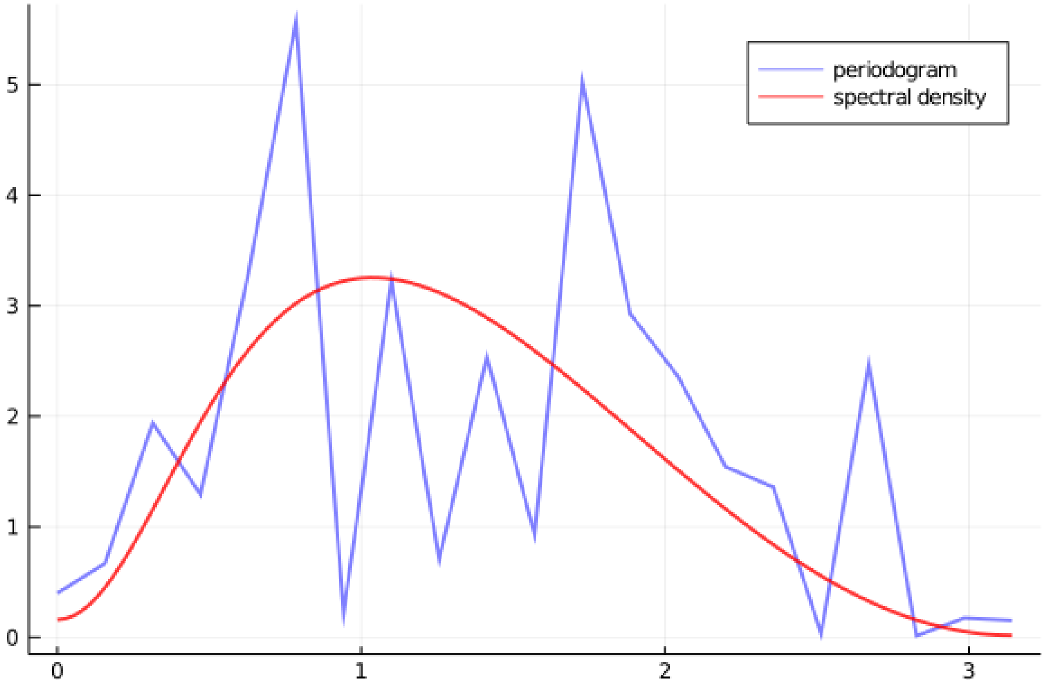
Here's a code snippet that, once the preceding code has been run, generates data from the process

$$X_t = 0.5X_{t-1} + \epsilon_t - 0.8\epsilon_{t-2} \tag{2}$$

where $\{\epsilon_t\}$ is white noise with unit variance, and compares the periodogram to the actual spectral density

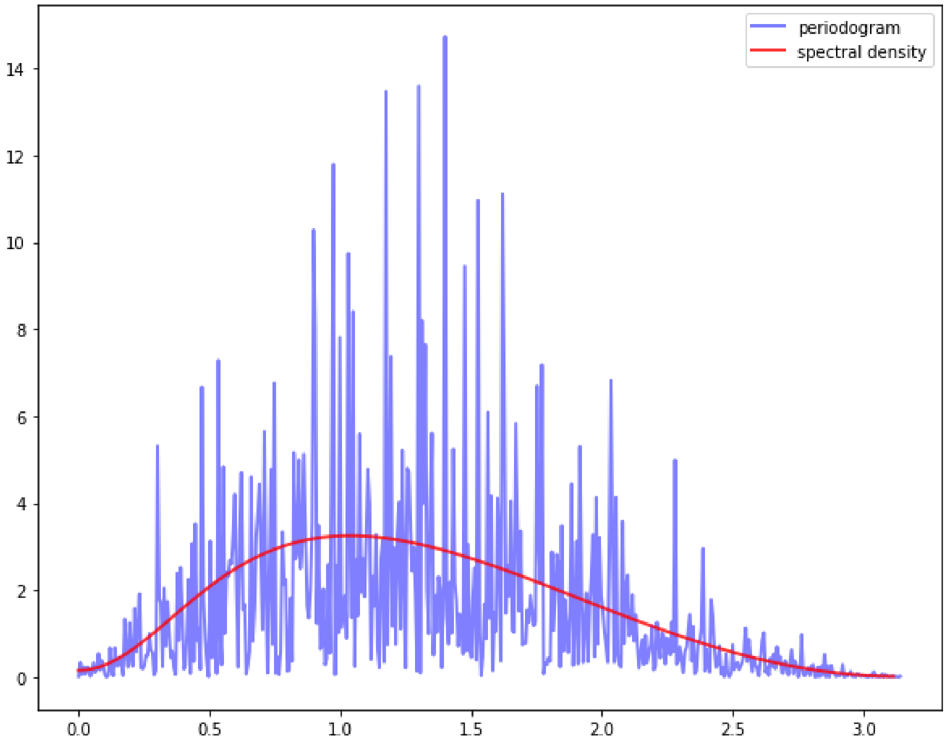
```
In [3]: using QuantEcon, Plots, Random
        gr(fmt = :png);
        Random.seed!(42) # For reproducible results.

        n = 40              # Data size
        ϕ = 0.5             # AR parameter
        θ = [0, -0.8]       # MA parameter
        σ = 1.0
        lp = ARMA(ϕ, θ, σ)
        X = simulation(lp, ts_length = n)

        x, y = periodogram(X)
        x_sd, y_sd = spectral_density(lp, two_pi=false, res=120)

        plot(x, y,linecolor="blue", linewidth=2, linealpha=0.5, lab="periodogram")
        plot!(x_sd, y_sd, linecolor="red", linewidth=2, linealpha=0.8,
  ↪lab="spectral density")
```

```
Out[3]:
```



This estimate looks rather disappointing, but the data size is only 40, so perhaps it’s not surprising that the estimate is poor.

However, if we try again with `n = 1200` the outcome is not much better



The periodogram is far too irregular relative to the underlying spectral density.

This brings us to our next topic.

4 Smoothing

There are two related issues here.

One is that, given the way the fast Fourier transform is implemented, the number of points ω at which $I(\omega)$ is estimated increases in line with the amount of data.

In other words, although we have more data, we are also using it to estimate more values.

A second issue is that densities of all types are fundamentally hard to estimate without parametric assumptions.

Typically, nonparametric estimation of densities requires some degree of smoothing.

The standard way that smoothing is applied to periodograms is by taking local averages.

In other words, the value $I(\omega_j)$ is replaced with a weighted average of the adjacent values

$$I(\omega_{j-p}), I(\omega_{j-p+1}), \dots, I(\omega_j), \dots, I(\omega_{j+p})$$

This weighted average can be written as

$$I_S(\omega_j) := \sum_{\ell=-p}^{p} w(\ell) I(\omega_{j+\ell}) \tag{3}$$

where the weights $w(-p), \dots, w(p)$ are a sequence of $2p + 1$ nonnegative values summing to one.

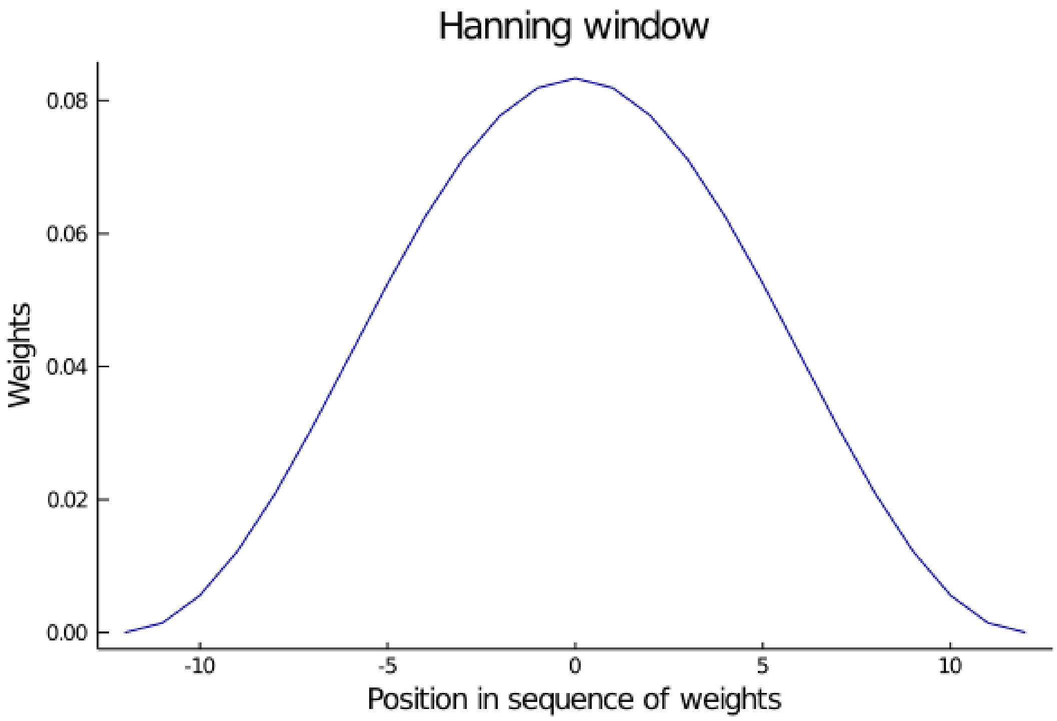
In generally, larger values of p indicate more smoothing — more on this below.

The next figure shows the kind of sequence typically used.

Note the smaller weights towards the edges and larger weights in the center, so that more distant values from $I(\omega_j)$ have less weight than closer ones in the sum (3)

In [4]:
```
function hanning_window(M)
    w = [0.5 - 0.5 * cos(2 * pi * n / (M - 1)) for n = 0:(M-1)]
    return w
end

window = hanning_window(25) / sum(hanning_window(25))
x = range(-12, 12, length = 25)
plot(x, window, color="darkblue", title="Hanning window", ylabel="Weights",
    xlabel="Position in sequence of weights", legend=false, grid=false)
```

Out[4]:



4.1 Estimation with Smoothing

Our next step is to provide code that will not only estimate the periodogram but also provide smoothing as required.

Such functions have been written in estspec.jl and are available once you've installed QuantEcon.jl.

The GitHub listing displays three functions, `smooth()`, `periodogram()`, `ar_periodogram()`. We will discuss the first two here and the third one below.

The `periodogram()` function returns a periodogram, optionally smoothed via the `smooth()` function.

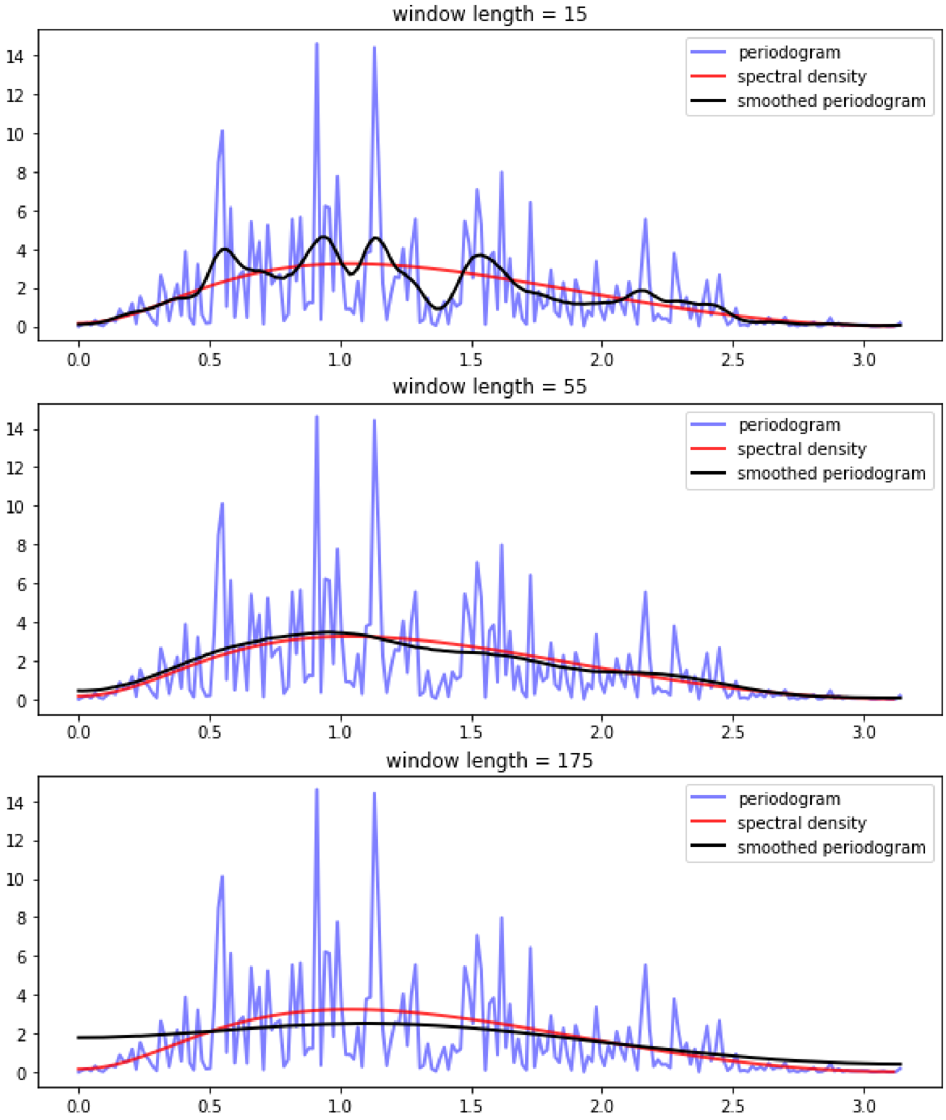
Regarding the `smooth()` function, since smoothing adds a nontrivial amount of computation, we have applied a fairly terse array-centric method based around `conv`.

Readers are left either to explore or simply to use this code according to their interests.

The next three figures each show smoothed and unsmoothed periodograms, as well as the population or "true" spectral density.

(The model is the same as before — see equation (2) — and there are 400 observations)

From top figure to bottom, the window length is varied from small to large.



In looking at the figure, we can see that for this model and data size, the window length chosen in the middle figure provides the best fit.

Relative to this value, the first window length provides insufficient smoothing, while the third gives too much smoothing.

Of course in real estimation problems the true spectral density is not visible and the choice of appropriate smoothing will have to be made based on judgement/priors or some other theory.

4.2 Pre-Filtering and Smoothing

In the code listing we showed three functions from the file `estspec.jl`.

The third function in the file (`ar_periodogram()`) adds a pre-processing step to periodogram smoothing.

First we describe the basic idea, and after that we give the code.

The essential idea is to

1. Transform the data in order to make estimation of the spectral density more efficient.
2. Compute the periodogram associated with the transformed data.
3. Reverse the effect of the transformation on the periodogram, so that it now estimates the spectral density of the original process.

Step 1 is called *pre-filtering* or *pre-whitening*, while step 3 is called *recoloring*.

The first step is called pre-whitening because the transformation is usually designed to turn the data into something closer to white noise.

Why would this be desirable in terms of spectral density estimation?

The reason is that we are smoothing our estimated periodogram based on estimated values at nearby points — recall (3).

The underlying assumption that makes this a good idea is that the true spectral density is relatively regular — the value of $I(\omega)$ is close to that of $I(\omega')$ when ω is close to ω'.

This will not be true in all cases, but it is certainly true for white noise.

For white noise, I is as regular as possible — it is a constant function.

In this case, values of $I(\omega')$ at points ω' near to ω provided the maximum possible amount of information about the value $I(\omega)$.

Another way to put this is that if I is relatively constant, then we can use a large amount of smoothing without introducing too much bias.

4.3 The AR(1) Setting

Let's examine this idea more carefully in a particular setting — where the data are assumed to be generated by an AR(1) process.

(More general ARMA settings can be handled using similar techniques to those described below)

Suppose in particular that $\{X_t\}$ is covariance stationary and AR(1), with

$$X_{t+1} = \mu + \phi X_t + \epsilon_{t+1} \tag{4}$$

where μ and $\phi \in (-1, 1)$ are unknown parameters and $\{\epsilon_t\}$ is white noise.

It follows that if we regress X_{t+1} on X_t and an intercept, the residuals will approximate white noise.

Let

- g be the spectral density of $\{\epsilon_t\}$ — a constant function, as discussed above
- I_0 be the periodogram estimated from the residuals — an estimate of g
- f be the spectral density of $\{X_t\}$ — the object we are trying to estimate

In view of an earlier result we obtained while discussing ARMA processes, f and g are related by

$$f(\omega) = \left| \frac{1}{1 - \phi e^{i\omega}} \right|^2 g(\omega) \tag{5}$$

This suggests that the recoloring step, which constructs an estimate I of f from I_0, should set

$$I(\omega) = \left| \frac{1}{1 - \hat{\phi} e^{i\omega}} \right|^2 I_0(\omega)$$

where $\hat{\phi}$ is the OLS estimate of ϕ.

The code for `ar_periodogram()` — the third function in `estspec.jl` — does exactly this. (See the code here).

The next figure shows realizations of the two kinds of smoothed periodograms

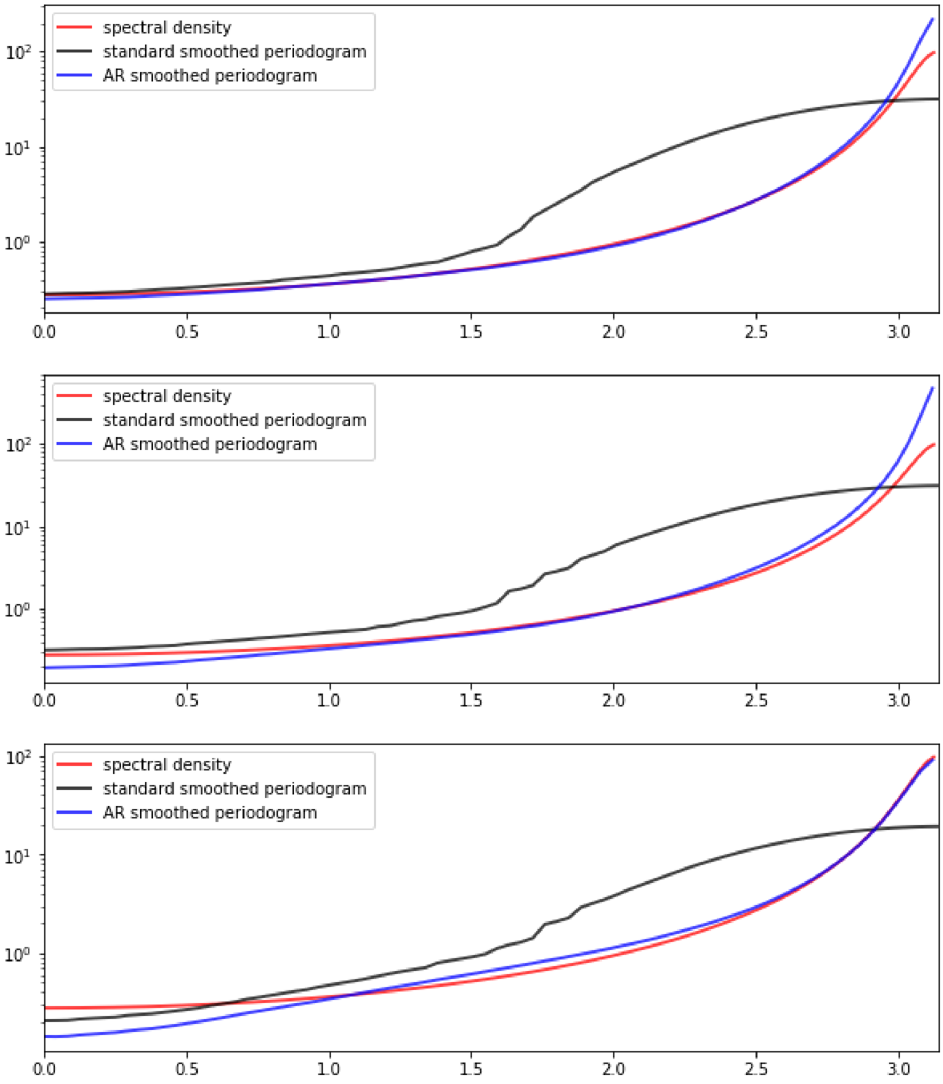
1. “standard smoothed periodogram”, the ordinary smoothed periodogram, and

2. “AR smoothed periodogram”, the pre-whitened and recolored one generated by `ar_periodogram()`

The periodograms are calculated from time series drawn from (4) with $\mu = 0$ and $\phi = -0.9$.

Each time series is of length 150.

The difference between the three subfigures is just randomness — each one uses a different draw of the time series.



In all cases, periodograms are fit with the “hamming” window and window length of 65.

Overall, the fit of the AR smoothed periodogram is much better, in the sense of being closer to the true spectral density.

5 Exercises

5.1 Exercise 1

Replicate this figure (modulo randomness).

The model is as in equation (2) and there are 400 observations.

For the smoothed periodogram, the window type is “hamming”.

5.2 Exercise 2

Replicate this figure (modulo randomness).

The model is as in equation (4), with $\mu = 0$, $\phi = -0.9$ and 150 observations in each time series.

All periodograms are fit with the “hamming” window and window length of 65.

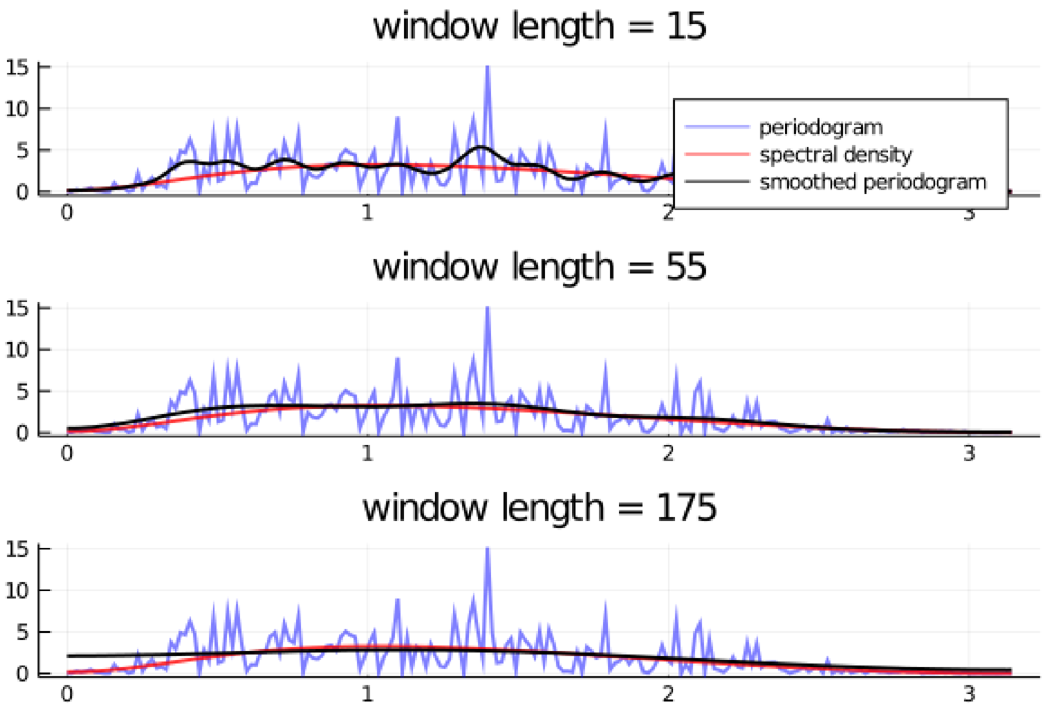
6 Solutions

6.1 Exercise 1

```
In [5]: n = 400
        ϕ = 0.5
        θ = [0, -0.8]
        σ = 1.0
        lp = ARMA(ϕ, θ, 1.0)
        X = simulation(lp, ts_length = n)

        xs = []
        x_sds = []
        x_sms = []
        ys = []
        y_sds = []
        y_sms = []
        titles = []

        for (i, wl) in enumerate([15, 55, 175])  # window lengths
            x, y = periodogram(X)
            push!(xs, x)
            push!(ys, y)

            x_sd, y_sd = spectral_density(lp, two_pi=false, res=120)
            push!(x_sds, x_sd)
            push!(y_sds, y_sd)

            x, y_smoothed = periodogram(X, "hamming", wl)
            push!(x_sms, x)
            push!(y_sms, y_smoothed)

            t = "window length = $wl"
            push!(titles, t)
        end
```

```
In [6]: plot(xs, ys, layout=(3,1), color=:blue, alpha=0.5,
            linewidth=2, label=["periodogram" "" ""])
        plot!(x_sds, y_sds, layout=(3,1), color=:red, alpha=0.8,
            linewidth=2, label=["spectral density" "" ""])
        plot!(x_sms, y_sms, layout=(3,1), color=:black,
            linewidth=2, label=["smoothed periodogram" "" ""])
        plot!(title=reshape(titles,1,length(titles)))
```

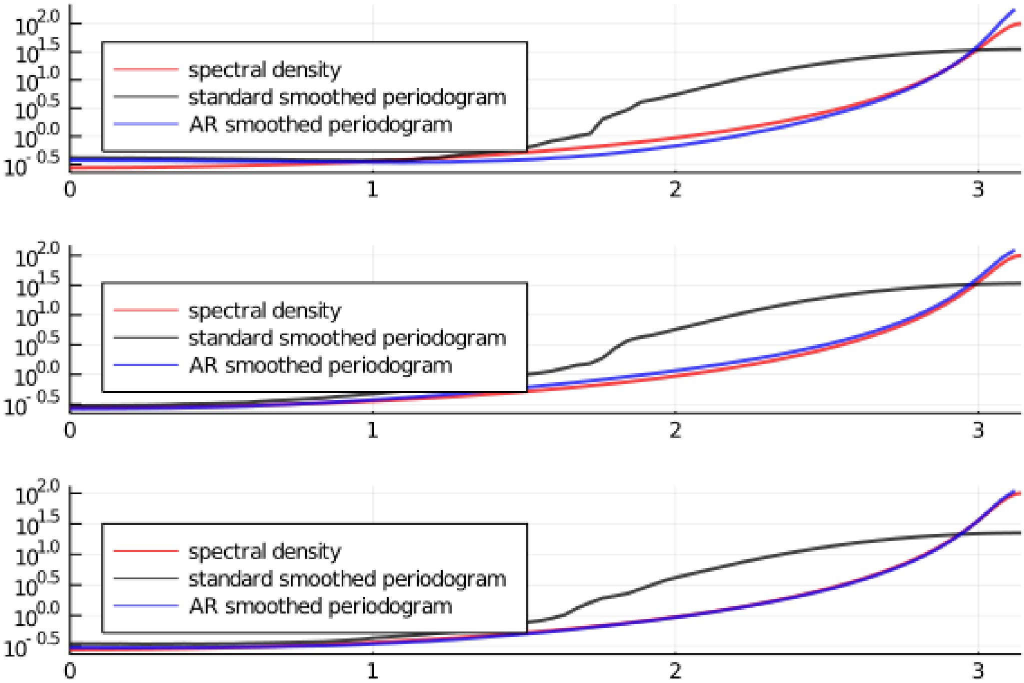
```
Out[6]:
```



6.2 Exercise 2

In [7]:
```
lp2 = ARMA(-0.9, 0.0, 1.0)
wl = 65
p = plot(layout=(3,1))

for i in 1:3
    X = simulation(lp2,ts_length=150)
    plot!(p[i],xlims = (0,pi))

    x_sd, y_sd = spectral_density(lp2,two_pi=false, res=180)
    plot!(p[i],x_sd, y_sd, linecolor=:red, linestyle=:solid,
        yscale=:log10, linewidth=2, linealpha=0.75,
        label="spectral density",legend=:topleft)

    x, y_smoothed = periodogram(X, "hamming", wl)
    plot!(p[i],x, y_smoothed, linecolor=:black, linestyle=:solid,
        yscale=:log10, linewidth=2, linealpha=0.75,
        label="standard smoothed periodogram",legend=:topleft)

    x, y_ar = ar_periodogram(X, "hamming", wl)
    plot!(p[i],x, y_ar, linecolor=:blue, linestyle=:solid,
        yscale=:log10, linewidth=2, linealpha=0.75,
        label="AR smoothed periodogram",legend=:topleft)
end
p
```

Out[7]:



References

[1] J. D. Cryer and K-S. Chan. *Time Series Analysis*. Springer, 2nd edition edition, 2008.

[2] Thomas J Sargent. *Macroeconomic Theory*. Academic Press, New York, 2nd edition, 1987.