4750 HW 3

Duncan Wilkie

10 October 2022

Problem 1. *Consider a square wave of frequency f_0: $x(t) = \operatorname{sgn}\sin(2\pi f_0 t)$. Calculate its Fourier transform and Fourier series. Consider a discrete sampling of the function with sampling frequencies $f_0/2$, $f_0/5$, and $f_0/1000$, and calculate the discrete Fourier transform in each case.*

Solution. In the convention used in class, the Fourier transform of $f(x)$ is

$$\hat{f}(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

The Fourier series of a periodic function $g(x)$ with period T is

$$g(x) = \sum_{n=-N}^{N} c_n e^{-inx/T}$$

where

$$c_n = \frac{1}{T}\int_{-T/2}^{T/2} g(t)e^{int/T}dt$$

Our $x(t)$ has period $1/f_0$, so the Fourier series coefficients are

$$c_n = f_0 \int_{-1/2f_0}^{1/2f_0} \operatorname{sgn}\sin(2\pi f_0 t)e^{inf_0 t}dt$$

On $(0, 1/2f_0)$, the argument to sgn ranges from 0 to 1, and on $(-1/2f_0, 0)$, it ranges from -1 to 0. Accordingly, on the two ranges, the value of $\operatorname{sgn}\sin$ is always $+1$ and always -1, respectively, so we can break the integral up as

$$c_n = f_0 \int_0^{1/2f_0} e^{inf_0 t}dt - f_0 \int_{-1/2f_0}^{0} e^{inf_0 t}dt = \frac{f_0}{inf_0}\left(e^{inf_0 t}\Big|_0^{1/2f_0} - e^{inf_0 t}\Big|_{-1/2f_0}^{0}\right)$$
$$= \frac{-i}{n}\left(e^{in/2} - e^{-in/2}\right) = \frac{-i}{n}2i\sin(n/2) = \frac{2}{n}\sin(n/2)$$

The Fourier transform is, on grounds of just knowing what it means, something like $\delta(\omega - 2\pi f_0)$, but let's do it mathematically. Using the Fourier series for x,

$$\tilde{x}(\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt = \int_{-\infty}^{\infty}\sum_{n=-\infty}^{\infty}\frac{2}{n}\sin(n/2)\sin(2\pi f_0 t)e^{i\omega t}dt$$

$$= \sum_{n=-\infty}^{\infty} \frac{2}{n} \sin(n/2) \int_{-\infty}^{\infty} \sin(2\pi f_0 t) e^{i\omega t} dt$$

We want to show that $\sin(2\pi f_0 t) = \delta(\omega - 2\pi f_0)$; the delta distribution is defined by its action as a linear functional. Noting that

$$\tilde{\delta}(\omega) = \int_{-\infty}^{\infty} \delta(t-a) e^{-i\omega t} dt = e^{-i\omega a} \Rightarrow \delta(t-a) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega a} e^{i\omega t} d\omega = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\omega(t-a)} d\omega,$$

we can write

$$\int_{-\infty}^{\infty} \sin(2\pi f_0 t) e^{i\omega t} = \int_{-\infty}^{\infty} \frac{e^{2\pi i f_0 t} - e^{-2\pi i f_0 t}}{2i} e^{i\omega t} dt$$

$$= \frac{1}{2i} \int_{-\infty}^{\infty} e^{it(\omega + 2\pi f_0)} dt - \frac{1}{2i} \int_{-\infty}^{\infty} e^{it(\omega - 2\pi f_0)} dt = \pi i [\delta(\omega - 2\pi f_0) - \delta(\omega + 2\pi f_0)]$$

Accordingly,

$$\tilde{x}(\omega) = 2\pi i [\delta(\omega - 2\pi f_0) - \delta(\omega + 2\pi f_0)] \sum_{n=-\infty}^{\infty} \frac{2}{n} \sin(n/2)$$

Taking $n = 0$ to have the value of the functional limit, 1, the sum appears to converge empirically, in fact to 2π:

```scheme
;; Scheme Lisp
(define (seq n)
  (if (= n 0)
      1
      (* (/ 2 n)
         (sin (/ n 2)))))

(define (sum seq start stop)
  (apply + (map seq (iota (- stop start) start))))

(sum seq -1000000 1000000)
;; => 6.283193014966864
```

Symbolic tools do have an exact value for this, clearly by way of the complex plane; it'd be interesting to work out an exact solution, but I'm satisfied knowing my formula above is very likely well-defined. □

Problem 2. *Find the paper from the first detection of gravitational waves, GW150914_065416, and download 4096 seconds of data sampled at 4kHz for LIGO Livingston and Hanford detectors L1 and H1 (files are 170 MB each). Using Matlab programs posted in Moodle for class (or if you prefer, adapting those to Python), calculate and plot the power spectral density of each time series using 2, 8 and 32 averages. Describe the differences between L1 and H1 PSDs, and the differences when using a different number of averages.*

Solution. I implemented the power spectral density estimate in Haskell for fun.

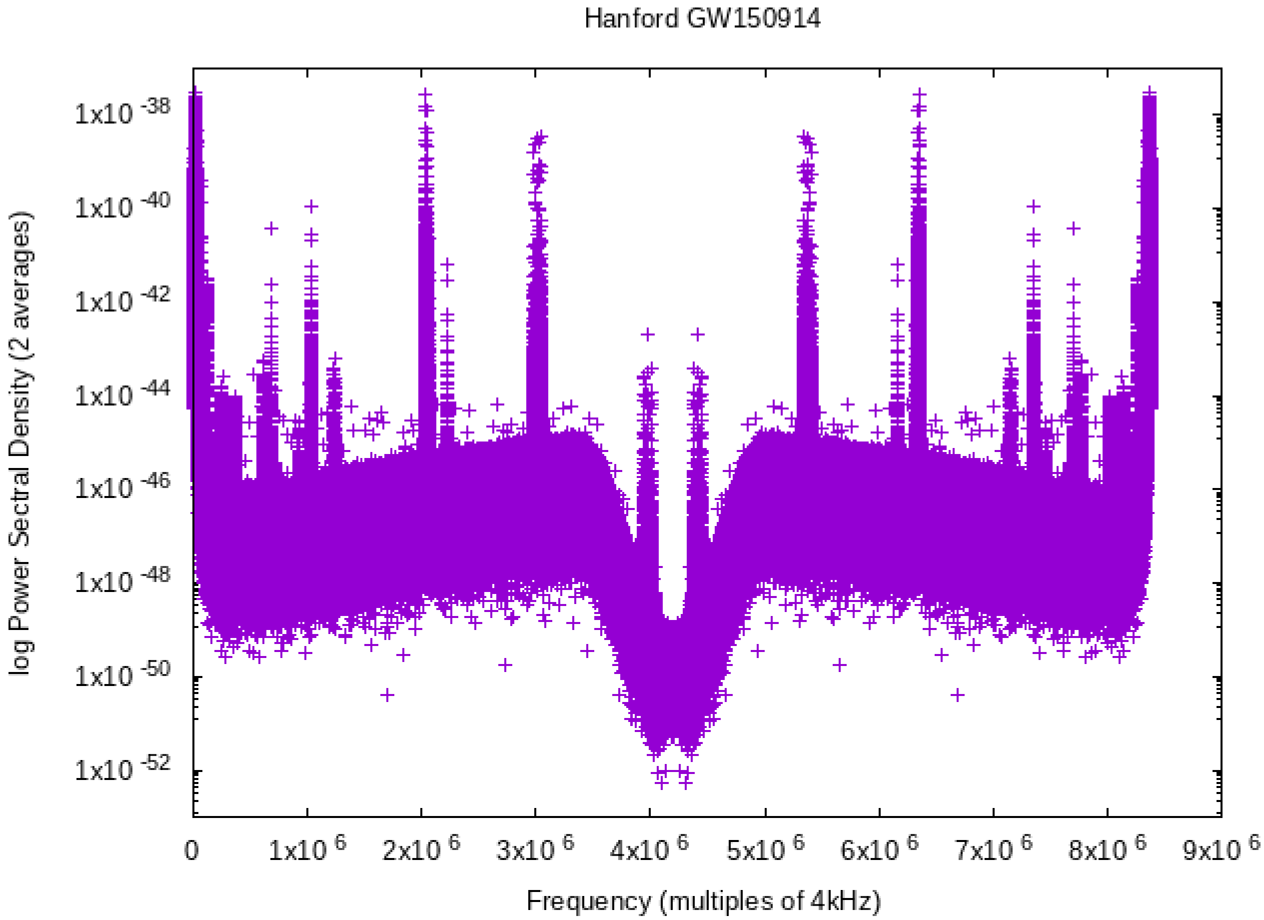
```haskell
-- GHC Haskell

{-# LANGUAGE BangPatterns #-}

import Prelude hiding (zip, take, (++), replicate, length, drop, map, foldl', sum)
import qualified Prelude as P (drop, take, (++))
import Data.Maybe (fromJust)
import Data.Complex ( Complex(..), magnitude, realPart)
import Data.Vector
import Statistics.Transform (fft)
import qualified Data.ByteString as BS
import Data.ByteString.Lex.Fractional (readSigned, readExponential)



rpad :: Vector e -> e -> Int -> Vector e
rpad xs with to = take to (xs ++ replicate to with)

chunksOf :: Int -> Vector e -> e -> Vector (Vector e)
chunksOf size xs padWith = let elts = length xs
                               fullChunks = elts `div` size
                               partChunks = ceiling (fromIntegral elts / fromIntegral size
                                                     - fromIntegral fullChunks)
                               in generate (fullChunks + partChunks)
                                  (\i -> if i /= fullChunks + partChunks
                                    then slice (i * size) (size) xs
                                    else if partChunks == 1
                                         then rpad (slice (fullChunks * size)
                                                    (elts - fullChunks * size) xs)
                                              padWith size
                                         else slice (i * size) (size) xs)


-- Fast/Discrete Fourier Transform: $\tilde{x}_T(f_k) = \sum_{j=-N/2}^{N/2} x_j e^{-2\pi ikj/N} \Delta t$
-- where $x_j(t)$ is the input time series with uniform time spacing $\Delta t$,
-- $T = N\Delta t$ is the total measurement time, $N$ is the number of points in the time series,
-- and $f_k = \frac{k}{T}$, where $0 \leq k \leq N-1$.
-- This algorithm is $O(n^2)$, and accordingly, is impossible to run on the input data.
badFft :: Vector (Complex Double) -> Double -> Vector (Complex Double)
```

```haskell
badFft ts delta_t = let n = length ts
                        bound = n `div` 2
                    in generate n (\k -> sum
                                    (generate n
                                     (\j  -> (ts ! j) * (delta_t :+ 0)
                                             * exp (0 :+ (-1) * 2 * pi
                                                    * fromIntegral k * fromIntegral (j - bound)
                                                    / fromIntegral n))))




-- Estimate power spectral density of time series, the "energy" carried by given frequencies:
-- $S_n(f_k) = \frac{1}{MN\Delta t} \sum_{i=0}^{M} |\tilde{x}_i(f_k)|^2$
-- where $M$ is the number of samples desired, and $\tilde{x}_i(f_k)$ is interpreted as
-- the FFT of the $i$th sub-series of length $T/M$ at value $f_k$.
-- Higher $M$ yields a better estimate, at the cost of frequency resolution.
psdEstimate :: Vector (Complex Double) -> Double -> Int -> Vector Double
psdEstimate ts delta_t samples = let coeff = 1 / (fromIntegral samples * fromIntegral n * delta_t)
                                     n = length ts
                                     sublength =  n `div` samples
                                     ffts = map (\chunk -> map ((*) $ delta_t :+ 0) (fft chunk))
                                       $ chunksOf sublength ts 0
                                   in generate sublength
                                      (\k -> coeff
                                        * sum (generate samples
                                               (\i -> (^2) . magnitude $ (ffts ! i) ! k)))



-- monad to read gravitational wave time series from file; they use '^#' as a comment
readData :: Int -> BS.ByteString -> (Vector (Complex Double))
readData n file = map (\x -> (fst $ fromJust (readSigned readExponential x)) :+ 0 )
                  $ fromList (P.take n (BS.split 10 file))



main :: IO ()
main = do h1str <- BS.readFile "H-H1_GWOSC_4KHZ_R1-1126257415-4096.txt"
--          l1str <- BS.readFile "L-L1_GWOSC_4KHZ_R1-1126257415-4096.txt"
          let h1 = readData 16777216 h1str
          let h1s = psdEstimate h1 (1 / 4e3) 32
          print h1s
          -- you can use "tail" to view the progress
          -- pipe stdout into tr , '\n' and redirect to file
```

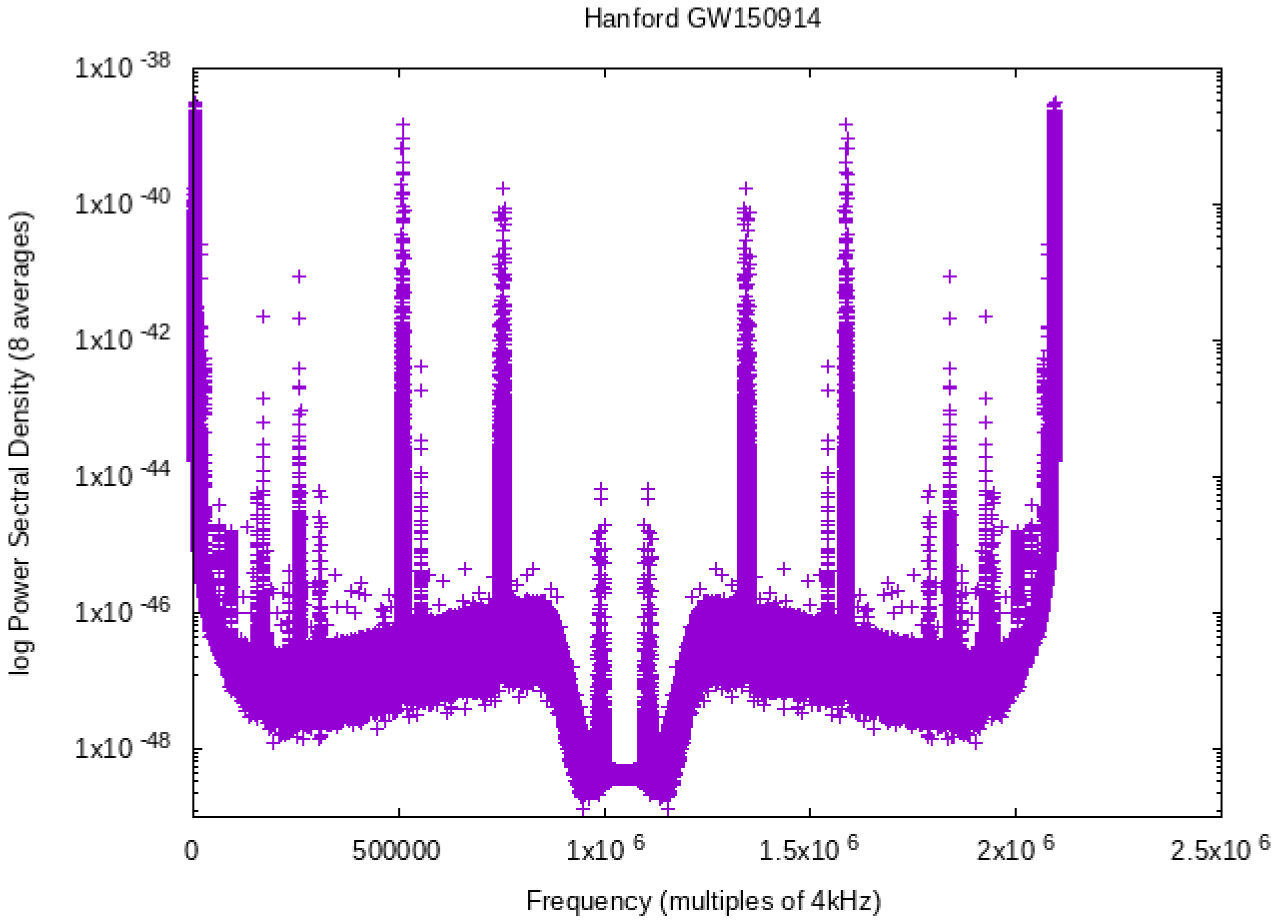
The output can be fed into gnuplot. For the Livingston data, a very high-power low-frequency data point was skewing the observation, so it was suppressed to generate a readable plot.

```
set terminal png font ",10"
set xlabel "Frequency (multiples of 4kHz)"
set key noautotitle
set logscale y


set title "Hanford GW150914"
set ylabel "log Power Sectral Density (2 averages)"
set output "h1s2.png"
plot "h1s2.txt" using 1

set title "Hanford GW150914"
set ylabel "log Power Sectral Density (8 averages)"
set output "h1s8.png"
plot "h1s8.txt" using 1

set title "Hanford GW150914"
set ylabel "log Power Sectral Density (32 averages)"
set output "h1s32.png"
plot "h1s32.txt" using 1

set title "Livingston GW150914"
set ylabel "log Power Sectral Density (2 averages)"
set output "l1s2.png"
plot "l1s2.txt" using 1

set title "Livingston GW150914"
set ylabel "log Power Sectral Density (8 averages)"
set output "l1s8.png"
plot "l1s8.txt" using 1

set title "Livingston GW150914"
set ylabel "log Power Sectral Density (32 averages)"
set output "l1s32.png"
plot "l1s32.txt" using 1
```

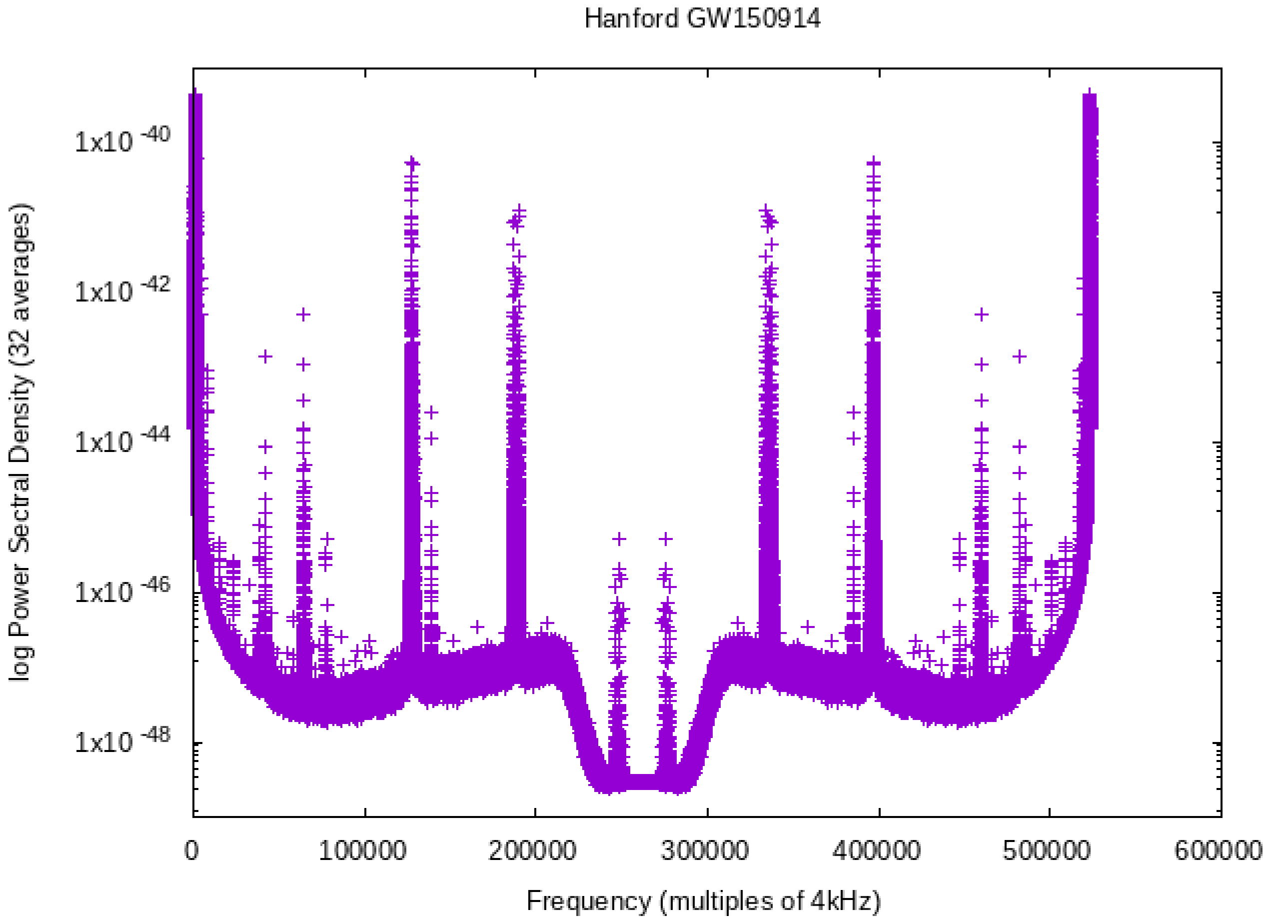
Resulting in

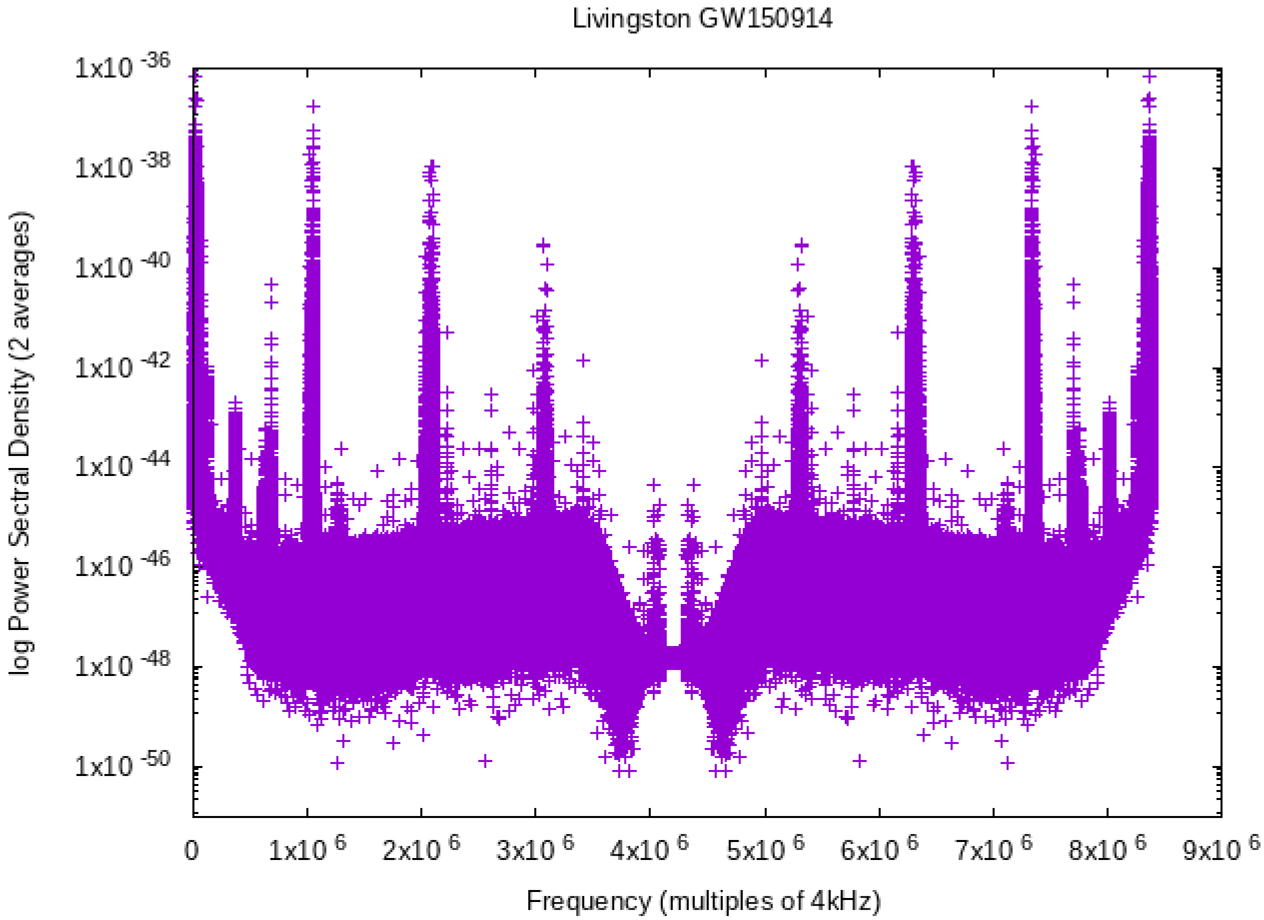


Hanford GW150914
log Power Sectral Density (2 averages)
1x10 -38
1x10 -40
1x10 -42
1x10 -44
1x10 -46
1x10 -48
1x10 -50
1x10 -52
0
1x10 6
2x10 6
3x10 6
4x10 6
5x10 6
6x10 6
7x10 6
8x10 6
9x10 6
Frequency (multiples of 4kHz)

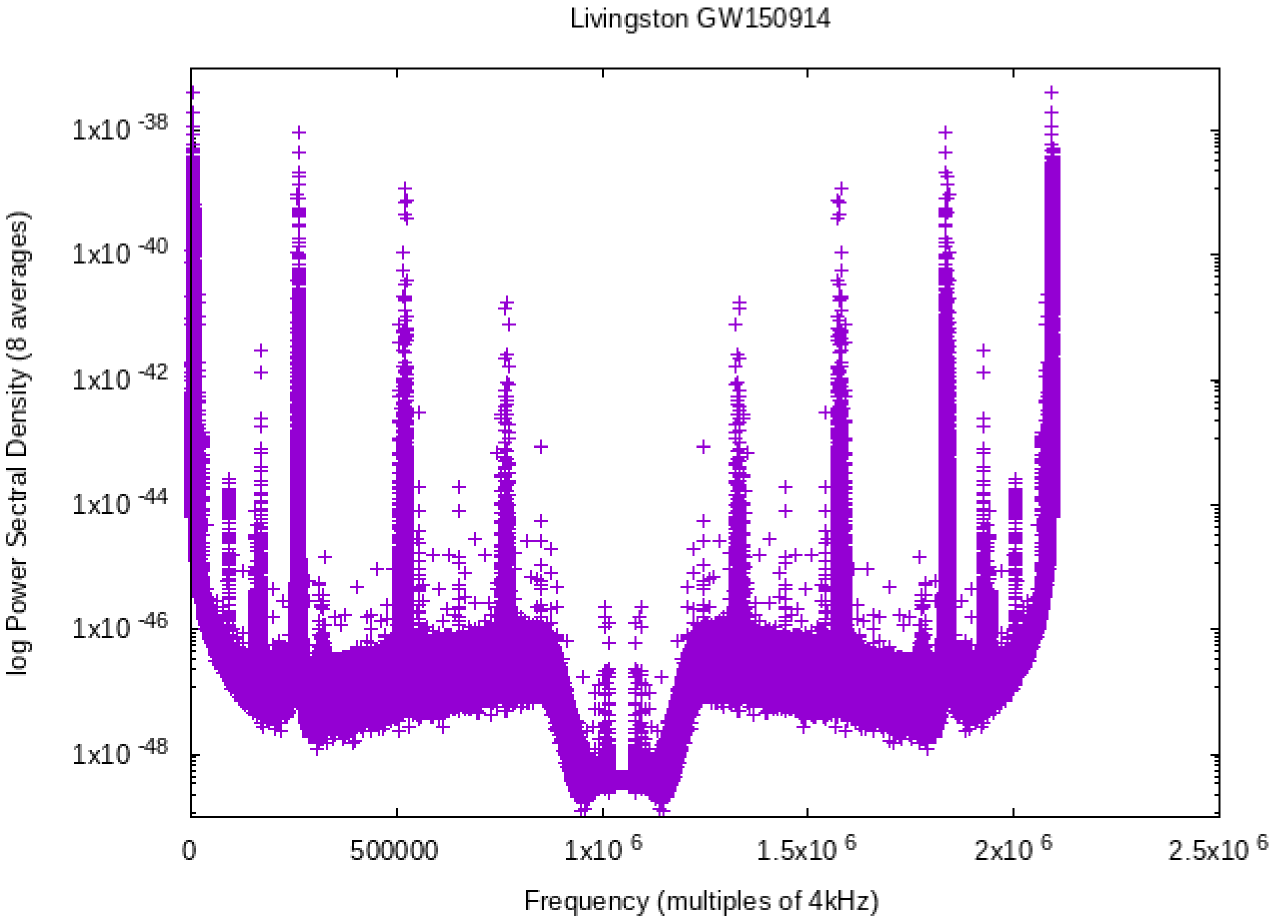


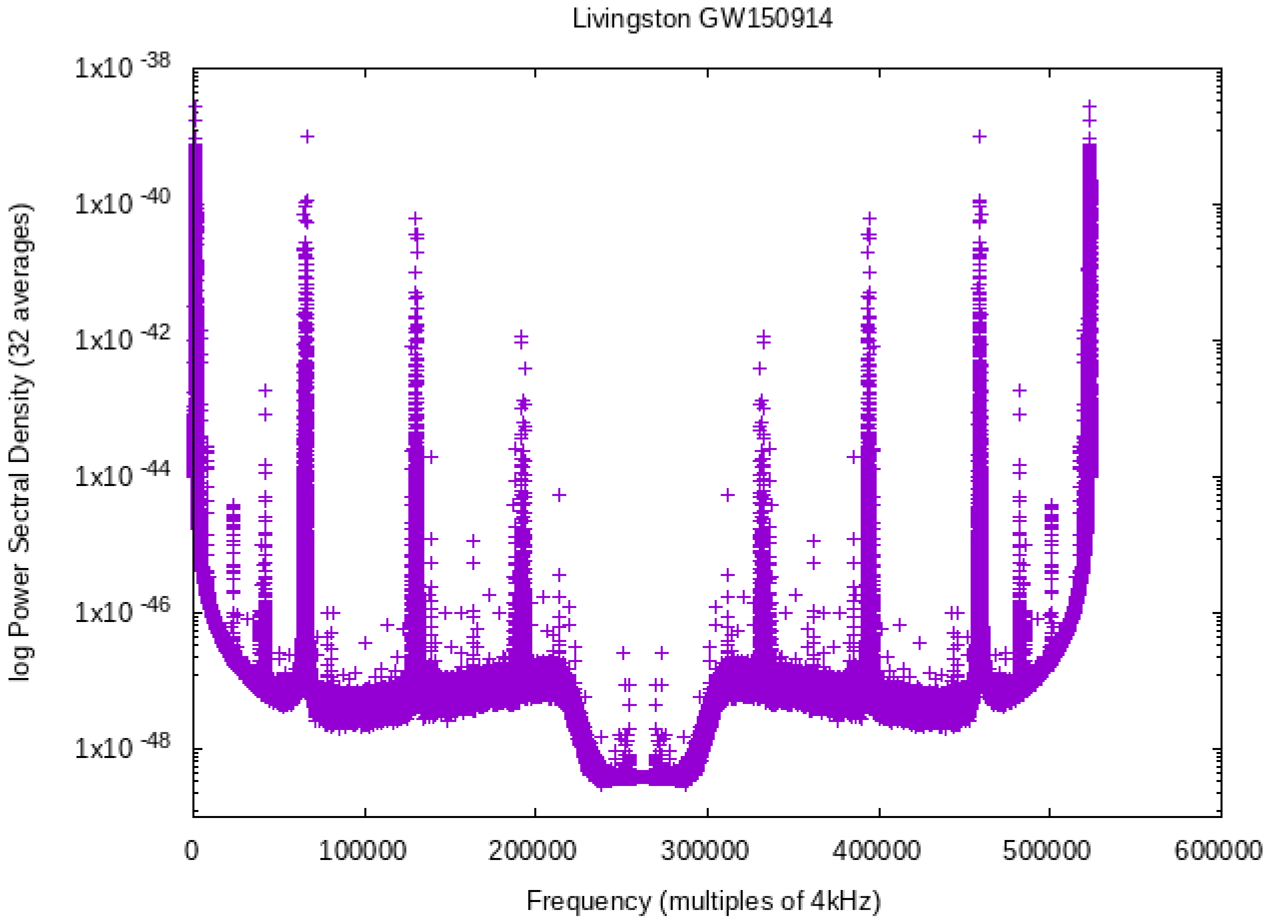
Hanford GW150914
log Power Sectral Density (8 averages)
1x10 -38
1x10 -40
1x10 -42
1x10 -44
1x10 -46
1x10 -48
0
500000
1x10 6
1.5x10 6
2x10 6
2.5x10 6
Frequency (multiples of 4kHz)


Hanford GW150914
log Power Sectral Density (32 averages)
1x10 -40
1x10 -42
1x10 -44
1x10 -46
1x10 -48
0
100000
200000
300000
400000
500000
600000
Frequency (multiples of 4kHz)



Livingston GW150914
log Power Sectral Density (2 averages)
1x10 -36
1x10 -38
1x10 -40
1x10 -42
1x10 -44
1x10 -46
1x10 -48
1x10 -50
0
1x10 6
2x10 6
3x10 6
4x10 6
5x10 6
6x10 6
7x10 6
8x10 6
9x10 6
Frequency (multiples of 4kHz)


Livingston GW150914
log Power Sectral Density (8 averages)
1x10 -38
1x10 -40
1x10 -42
1x10 -44
1x10 -46
1x10 -48
0
500000
1x10 6
1.5x10 6
2x10 6
2.5x10 6
Frequency (multiples of 4kHz)



The Livingston data appears to have a much lower actual spectral density for a given frequency, but the contour is the same; fewer averages appears to "smear out" the spectrum, consistent with the uncertainty principle for Fourier series. □

Problem 3. *Using the Fluctuation-dissipation theorem, we proved that the power spectral density of thermal fluctuations of a harmonic oscillator with viscous damping coefficient γ is given by*

$$x^2(\omega) = \frac{4k_BT\gamma}{k}\frac{1}{\left(1-(\omega/\omega_0)^2\right)^2+\gamma^2\omega^2}$$

Integrate this expression over frequency to calculate the average of $x^2(t)$ (mean square, or x^2_{rms}), and write your result as the expression of the equipartition theorem.

Solution. We have

$$\int_{-\infty}^{\infty} x^2(\omega)d\omega = \frac{4k_BT\gamma}{k}\int_{-\infty}^{\infty}\frac{d\omega}{\left(1-(\omega/\omega_0)^2\right)^2+\gamma^2\omega^2}$$

$$= \frac{4k_BT\gamma}{k}\int_{-\infty}^{\infty}\frac{\omega_0^4 d\omega}{\omega_0^4-(\omega_0^4\gamma^2-2\omega_0^2)\omega^2+\omega^4}$$

The partial fraction decomposition of the integrand isn't hard:

$$\frac{1}{a-bx+x^2}=\frac{1}{(x+r_1)(x+r_2)}=\frac{A}{x+r_1}+\frac{B}{x+r_2}\Leftrightarrow 1=A(x+r_2)+B(x+r_1)\Rightarrow A+B=0, r_2-r_1=\frac{1}{A}$$

The roots of this polynomial are given by

$$\omega^2=\frac{(\omega_0^4\gamma^2-2\omega_0^2)^2\pm\sqrt{(\omega_0^4\gamma^2-2\omega_0^2)^2-4\omega_0^4}}{2}$$

and their difference is

$$r_2-r_1=\sqrt{(2\omega_0^2-\omega_0^4\gamma^2)^2-4\omega_0^4}=\omega_0^3\sqrt{\gamma^2\omega_0^2-4\gamma}$$

Accordingly, the integral is

$$= \frac{4k_BT\gamma\omega_0^4}{k\omega_0^3\sqrt{\gamma^2\omega_0^2-4\gamma}}\left(\int_{-\infty}^{\infty}\frac{d\omega}{\omega^2+r_1}-\int_{-\infty}^{\infty}\frac{d\omega}{\omega^2+r_2}\right)$$

$$= \frac{4k_BT\gamma\omega_0^4}{k\omega_0^3\sqrt{\gamma^2\omega_0^2-4\gamma}}\left(\tan^{-1}\left(\frac{\omega}{r_1}\right)\Big|_{-\infty}^{\infty}-\tan^{-1}\left(\frac{\omega}{r_2}\right)\Big|_{-\infty}^{\infty}\right)$$

$$= \frac{4k_BT\gamma\omega_0^4}{k\omega_0^3\sqrt{\gamma^2\omega_0^2-4\gamma}}\left(\frac{\pi\,\mathrm{sgn}(r_1)}{2\sqrt{r_1}}+\frac{\pi\,\mathrm{sgn}(r_1)}{2\sqrt{r_1}}-\frac{\pi\,\mathrm{sgn}(r_2)}{2\sqrt{r_2}}-\frac{\pi\,\mathrm{sgn}(r_2)}{2\sqrt{r_2}}\right)$$

$$= \frac{4\pi k_BT\gamma\omega_0}{k\sqrt{\gamma^2\omega_0^2-4\gamma}}(\mathrm{sgn}(r_1)-\mathrm{sgn}(r_2))$$

In any case of the signs, the magnitude of the integral is either zero or

$$= \frac{4\pi k_BT\gamma\omega_0}{k\sqrt{\gamma^2\omega_0^2-4\gamma}}$$

□

Problem 4. *Consider a simple pendulum with its suspension point excited by seismic noise. Assume the resonance pendulum frequency is 0.7 Hz with* $Q=2$ *(this is achieved using active damping), and the seismic noise is the amplitude spectral density measured at a LIGO Livingston seismometer on September 17, 2:00 UTC, available in Moodle for this homework. Using Matlab or your favorite program, plot the amplitude spectral density of the pendulum displacement, and calculate (numerically) the the rms motion between 0.01 Hz and 100 Hz endplm*

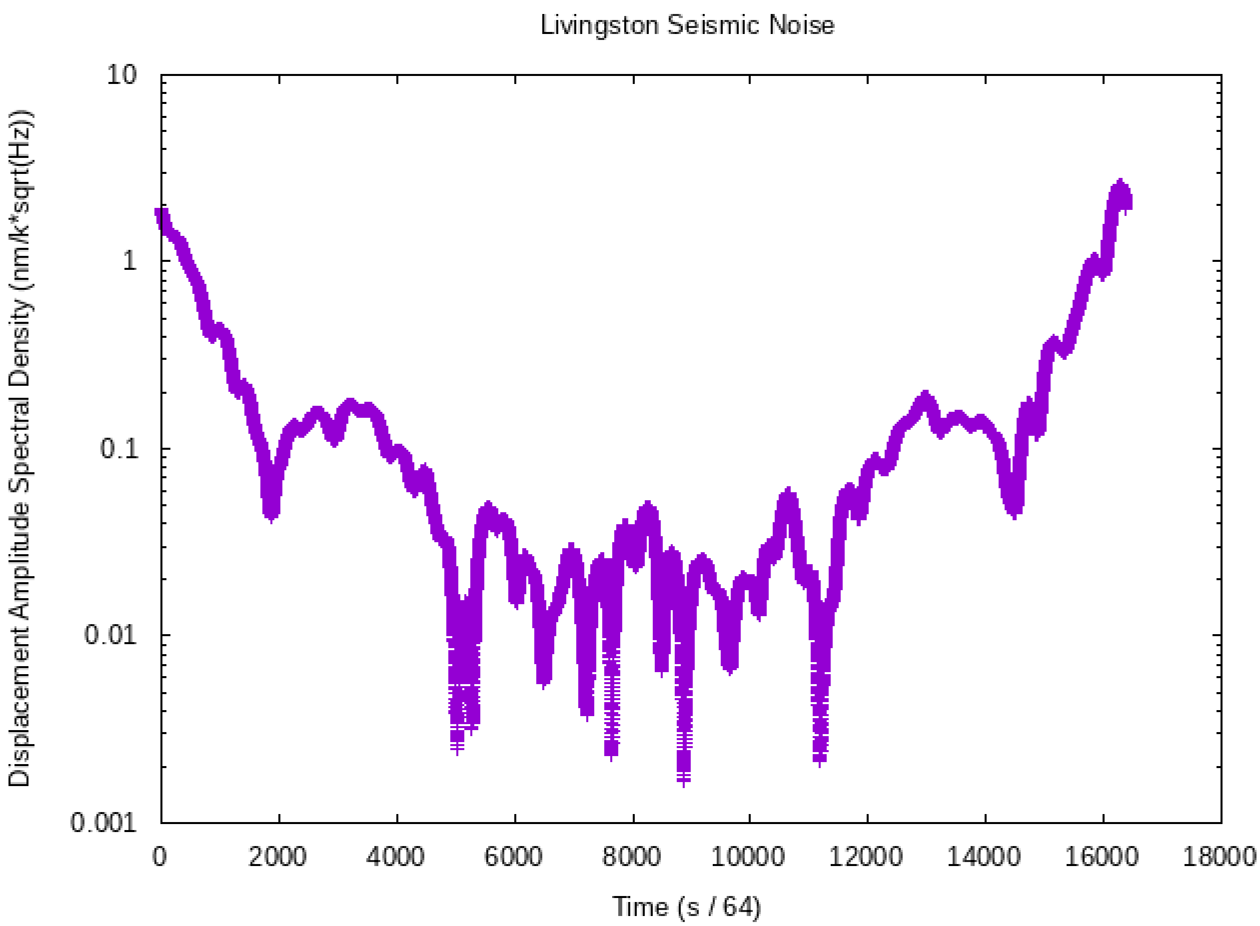
Solution. The amplitude spectral density, the square root of the power spectral density, is a proxy for the Fourier transform. Accordingly, the amplitude spectral density of displacement, the inverse FT of the product of the FTs of the forcing signal and response signal, can be computed as

```haskell
-- GHC Haskell

import Prelude hiding (zip, unzip, filter, take, (++), replicate, length, drop, map, foldl', sum)
import qualified Prelude as P (drop, take, (++), map)
import Data.Maybe (fromJust)
import Data.Complex ( Complex(..), magnitude, realPart)
import Data.Vector
import Statistics.Transform (fft, ifft)
import qualified Data.ByteString as BS
import Data.ByteString.Lex.Fractional (readSigned, readExponential)



-- monad to read gravitational wave time series from file; they use '^#' as a comment
readData :: Int -> BS.ByteString -> Vector (Double, Complex Double)
readData n file = fromList $ P.map (\l -> let wds = (BS.split 32 l)
                                     in (fst (fromJust $ readExponential (wds !! 0)),
                                          fst (fromJust $ readSigned readExponential (wds !! 1))
                                          :+ 0))
                  (P.take n (BS.split 10 file))


-- This is the driving force per unit 1/k, since k was not given
h :: Complex Double -> Complex Double -> Complex Double -> Complex Double
h w w0 q = 1 / (1 - (w / w0)^2 - (0 :+ 1) * (1 / q) * (w / w0))


main :: IO ()
main = do datastr <- BS.readFile "seisETMYY_220917_02UTC.txt"
          let dat = readData 16384 datastr
          let displ = map magnitude (ifft $ map (\tup -> (h (fst tup :+ 0) 0.7 2) * (snd tup)) dat)
          let rms = (sum $ map snd $ filter (\(w, _) -> w >= 0.1 && w <= 100)
                      $ zip (map fst dat) displ) / fromIntegral (length displ)

          print displ           -- pipe stdout into tr , '\n' and redirect to file

          print rms             -- returns ~0.0909m
```

Graphing this with gnuplot,

```
set terminal png font ",10"

set title "Livingston Seismic Noise"
set xlabel "Time (s / 64)"
set ylabel "Displacement Amplitude Spectral Density (nm/k*sqrt(Hz))"
set key noautotitle
set logscale y

set output "seismic.png"
plot "seismic.txt" using 1
```

results in



□