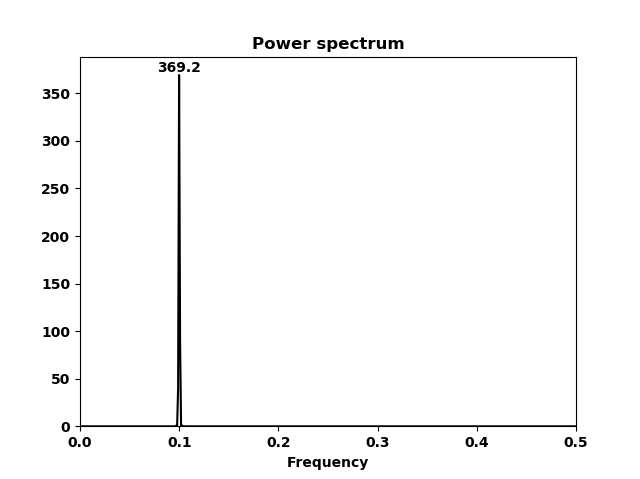
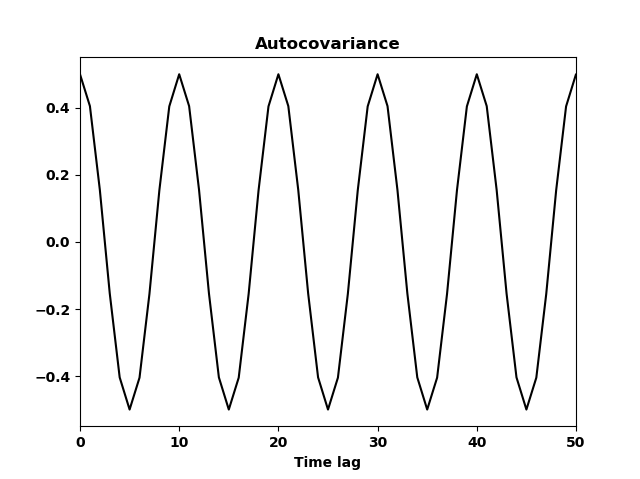
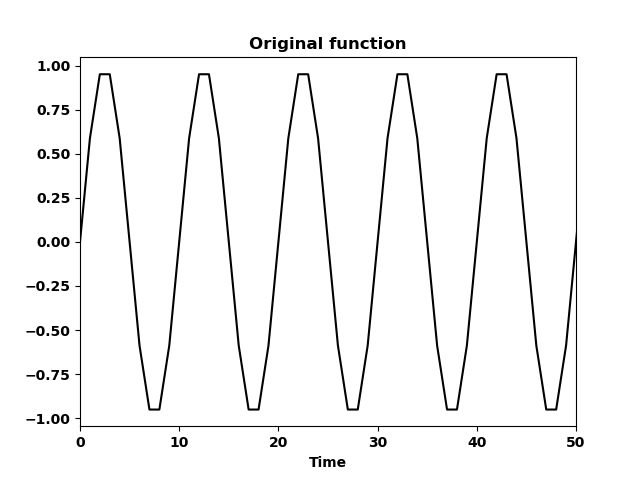
**객관적자료분석 HW#5**

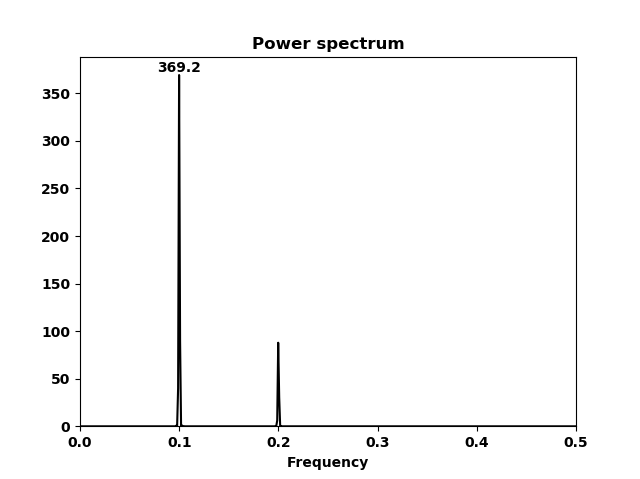
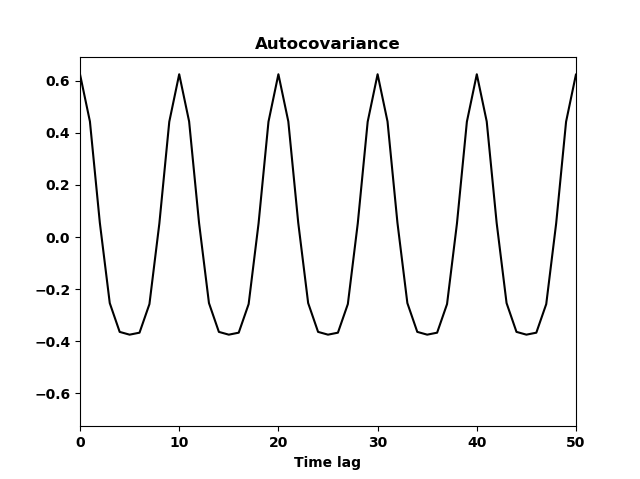
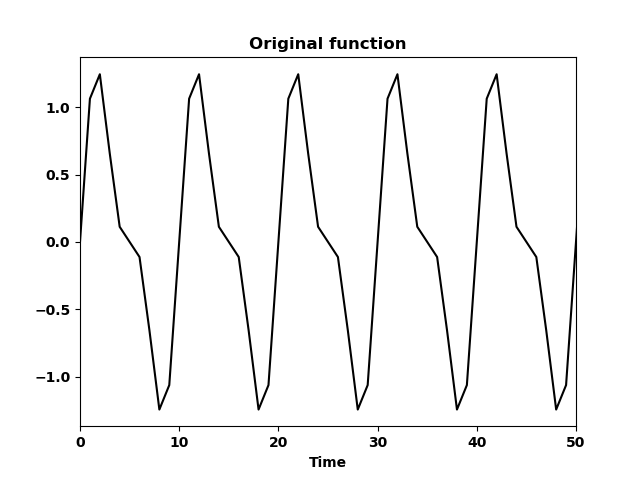
2017-29008 백관구

**1.**



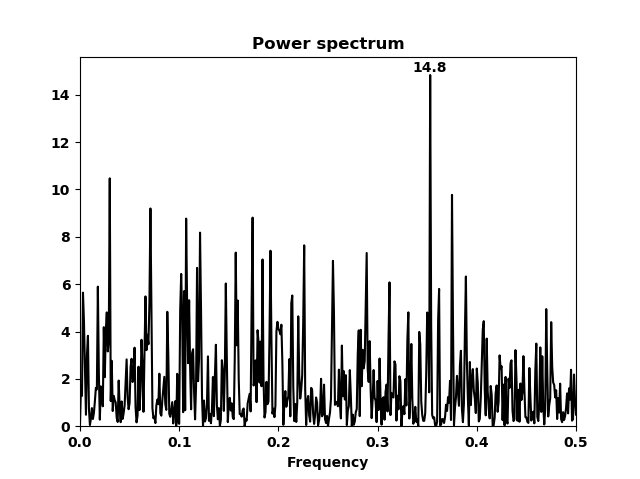
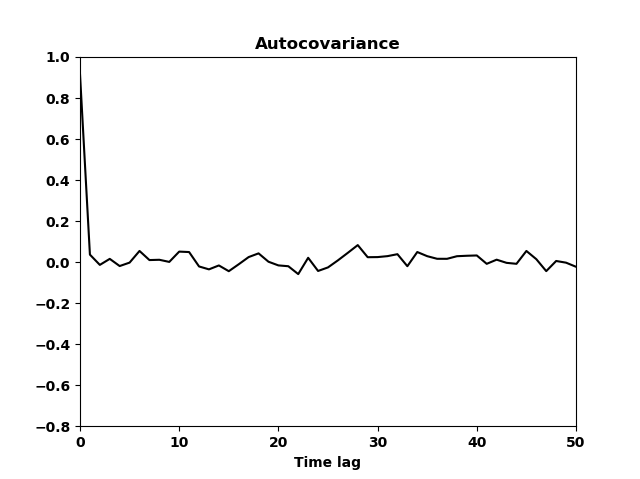
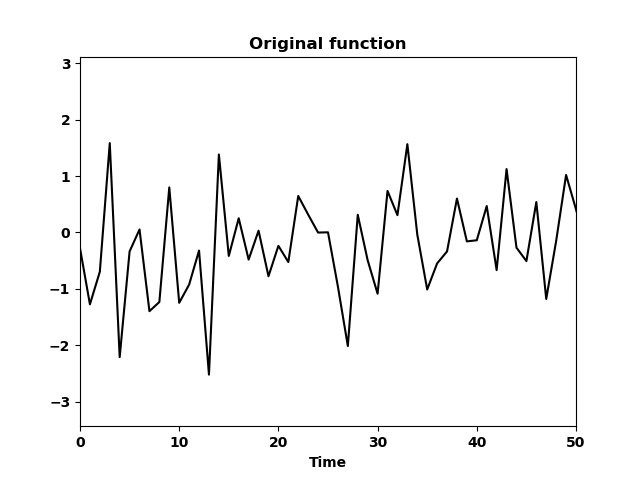
주기가 10인 주기함수의 자기공분산도 주기가 10인 형태로 나타난다. 파워 스펙트럼을 그려보면 주기 10에 대응되는 주파수 0.1에서 파워가 감지된다.

**2.**



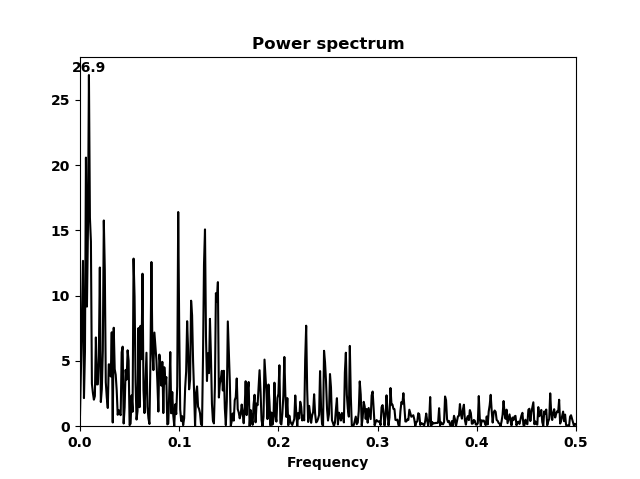
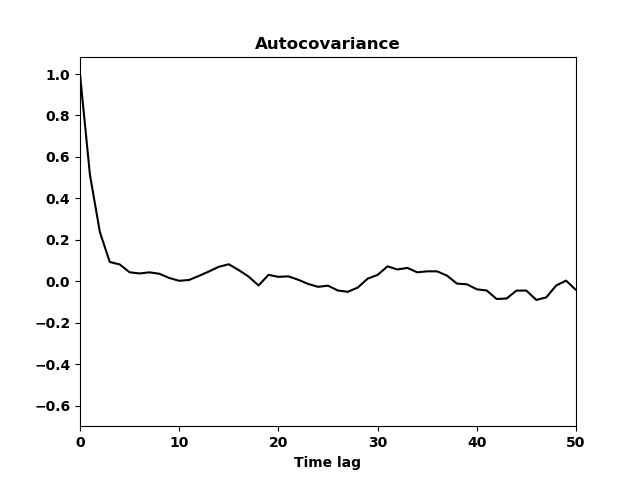
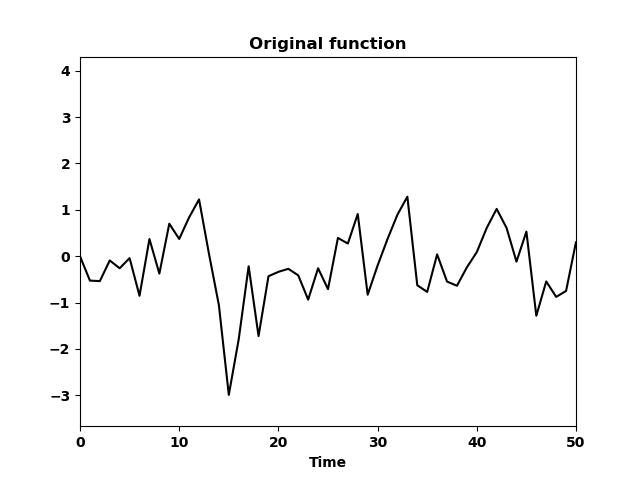
주기가 10, 진폭이 1인 주기함수와 주기가 5, 진폭이 0.5인 주기함수의 자기공분산도 주기가 10인 형태로 나타난다. 파워 스펙트럼을 그려보면 주기 10에 대응되는 주파수 0.1과 주기 5에 대응되는 주파수 0.2에서 스펙트럼이 구분되어 나타난다. 이 때 스펙트럼의 크기는 진폭의 제곱인 4배 차이가 난다.

**3.**



정규분포를 따르는 무작위 수(white noise)는 시간에 따른 경향성이 없다. 자기공분산은 time lag가 0인 경우에만 높은 자기공분산 값을 가지고, 나머지에 대해서는 0에 가까운 공분산 값을 갖는다. 파워 스펙트럼을 그려보면 특정 주파수로 설명할 수 없을만큼 여러 주파수에 파워가 분산되어 있는 것을 확인할 수 있다.

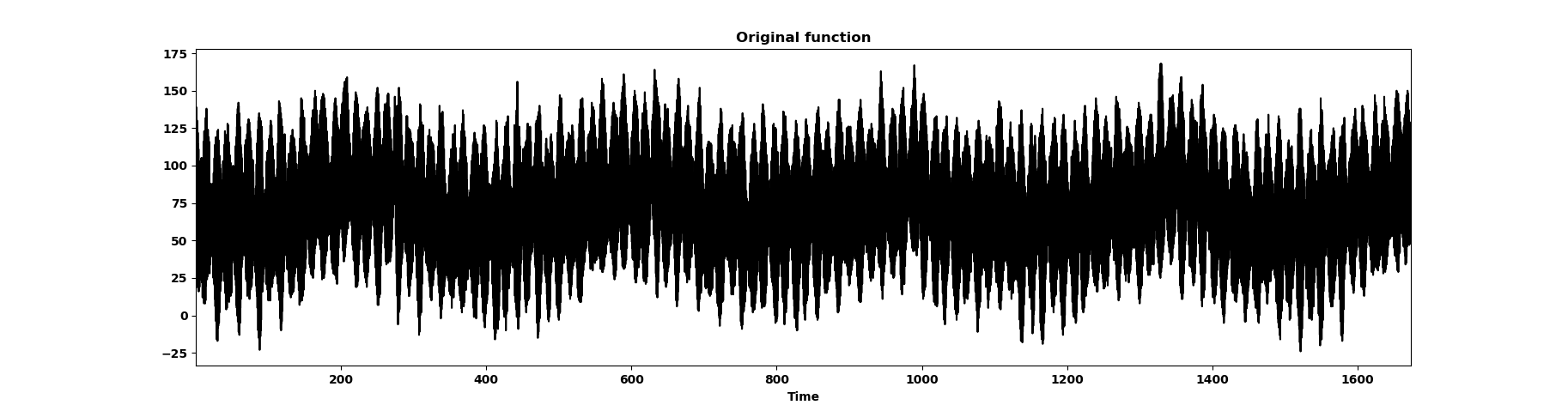
**4.**



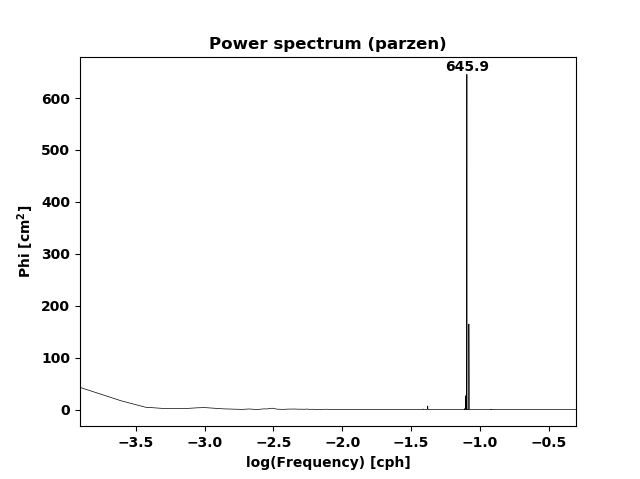
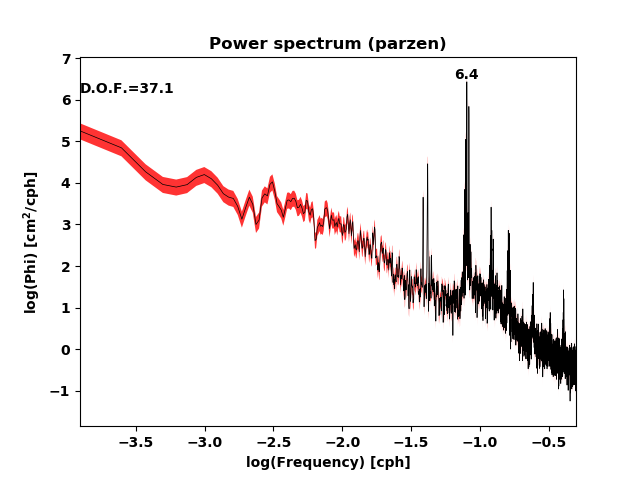
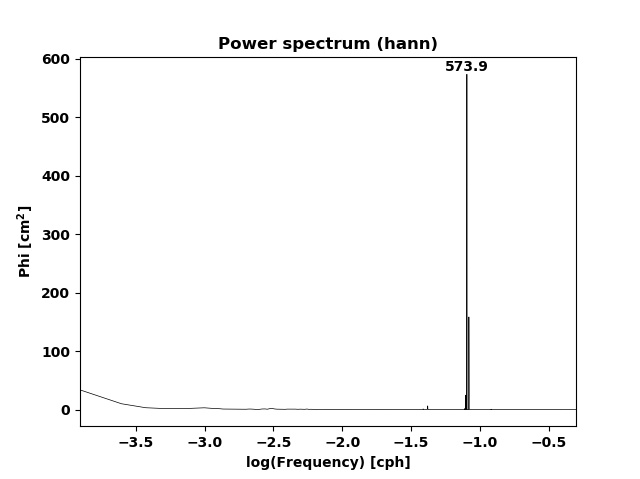
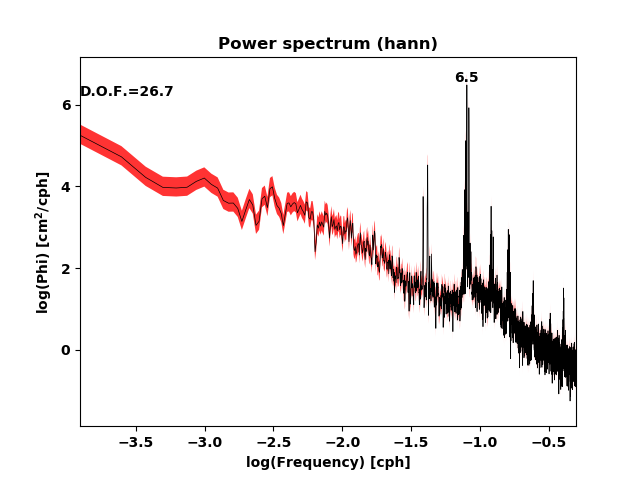
a=0.5인 red noise를 이용하였기 때문에 time lag가 1일 때 자기공분산이 time lag가 0일 때의 공분산의 절반 값을 갖는다. 파워 스펙트럼을 그려보면 작은 주파수에서 큰 주파수로 갈수록 점차 감소하여 0에 근사하는 형태를 보인다. 즉 red noise는 주파수가 작을 때 설명되는 정도가 가장 크다.

**5.** 부산 1 시간 간격 해수면 높이(cm) 자료

문제에서 주어진 자료는 아래 그림과 같은 형태로 나타난다.

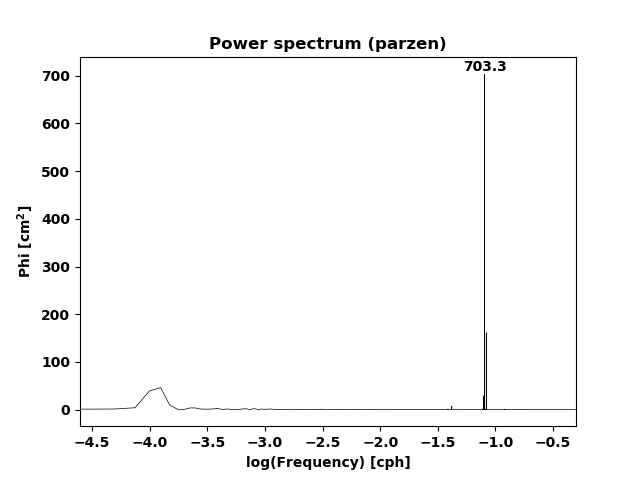
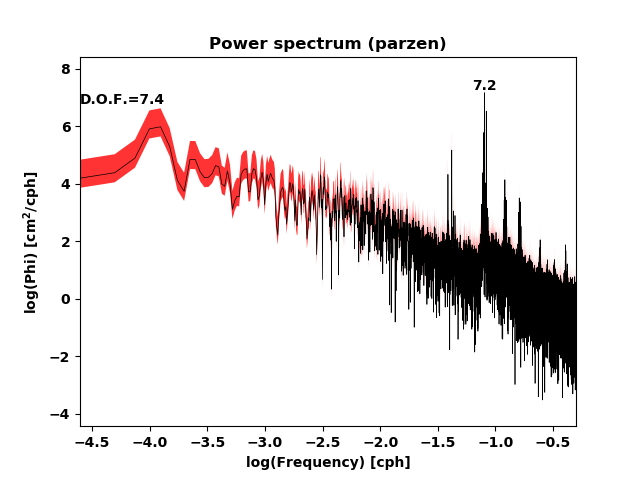
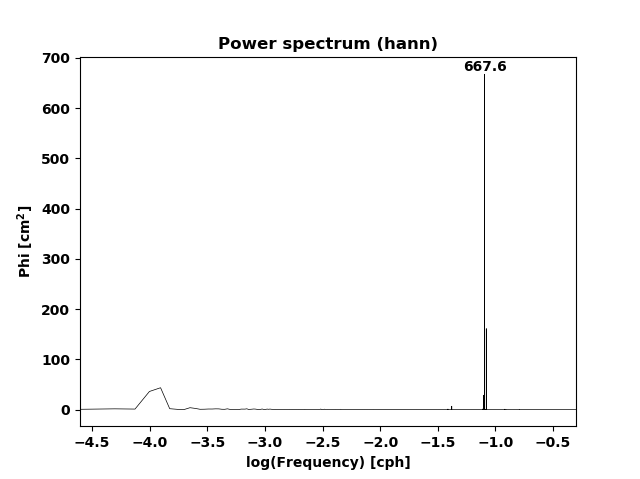
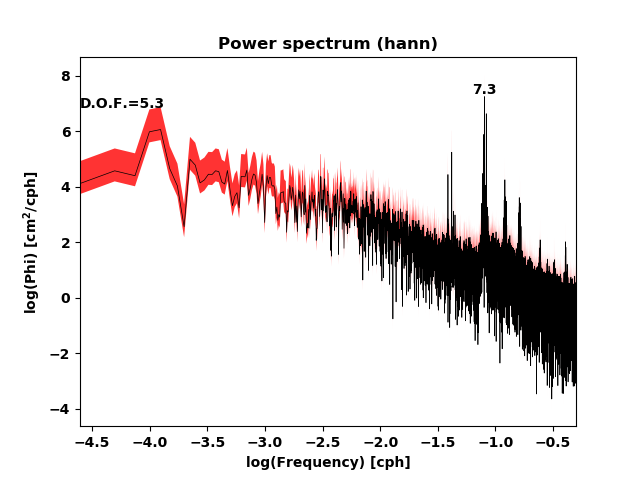


**1) M=N/10, delta t=1 hour**



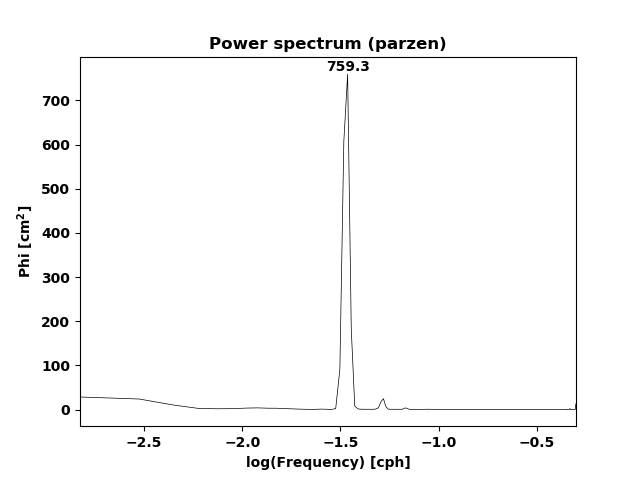
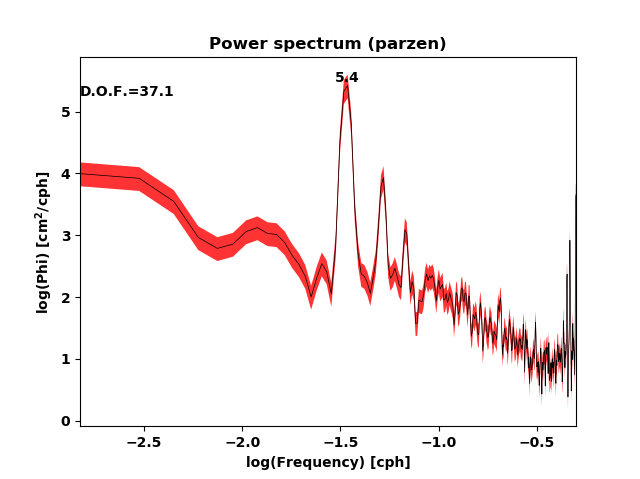
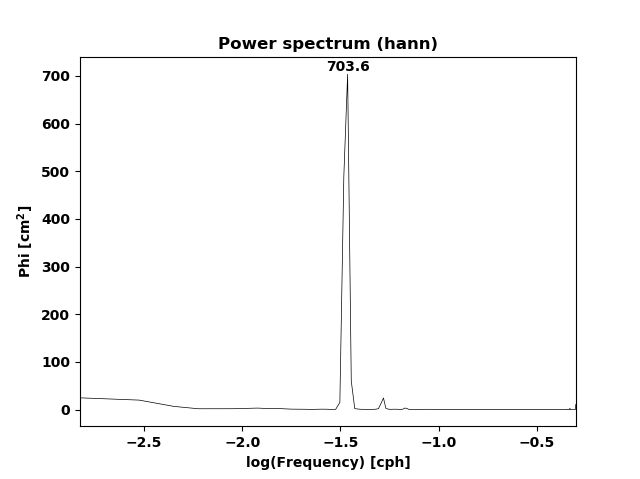
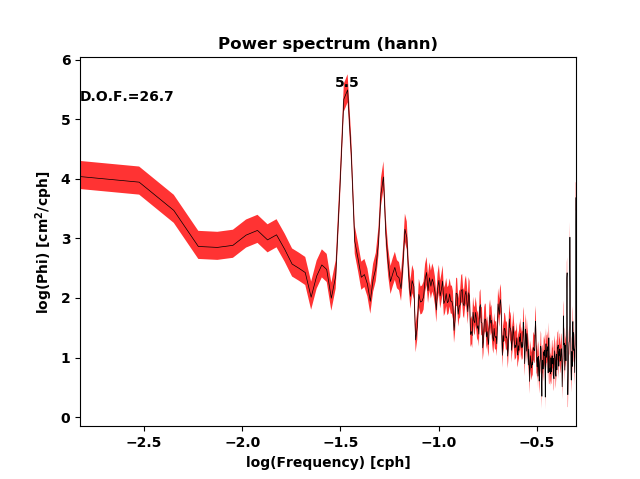
파워 스펙트럼 분석에서 진동수가 약 0.1보다 조금 작으므로 주기가 약 12 시간인 반일주기에서 강한 스펙트럼과 진동수가 약 0.04, 주기가 약 25시간의 일주기에서 두 번째로 강한 스펙트럼이 나타난다. Hanning과 parzen window를 각각 적용하였을 때 자유도는 26.7과 37.1로 계산되었다. 빨간 음영은 95% 신뢰구간을 나타낸다.

**2) M=N/2, delta t=1 hour**



전체 자료를 이용해 파워 스펙트럼을 분석하였을 때 결과로, 자유도가 hanning과 parzen window에서 각각 5.3과 7.4로 위의 문제보다 많이 낮아 신뢰구간의 폭이 넓어진다.

**3) M=N/10, delta t=12 hours**



12 시간 간격으로 샘플링하였을 때 1 시간 간격으로 샘플링했을 때보다 큰 진동수, 즉 단주기에서 강한 스펙트럼이 나타난다. 이 경우 파워 스펙트럼에서 잡을 수 있는 가장 짧은 주기는 24 시간인데 반일주기 현상이 24 시간보다 짧은 주기의 현상이므로 aliasing이 발생해 반일주기의 스펙트럼이 장주기의 스펙트럼으로 넘어온 것을 확인할 수 있다.

# -\*- coding: utf-8 -\*-

"""

Created on Thu Apr 25 12:49:52 2019

@author: WHITE

"""

#%% IMPORT MODULES

import numpy as np

import matplotlib as mpl

mpl.rc("font", weight = "bold")

import matplotlib.pyplot as plt

from statsmodels.tsa.stattools import acovf

from scipy.signal import periodogram

#%% FUNCTIONS

def FUNC1(i, t):

return np.sin(2 \* np.pi \* i \* t / 10)

def FUNC2(i, t):

return np.sin(2 \* np.pi \* i \* t / 10) + np.sin(2 \* np.pi \* i \* t / 5) / 2

def WHITE(n):

return np.random.normal(0, 1, n)

def RED(n, a):

result = np.zeros(n)

error = np.random.normal(0, 1, n)

for ni in range(n - 1):

result[ni + 1] = a \* result[ni] + ((1 - a \*\* 2) \*\* 0.5) \* error[ni + 1]

return result

#%% DRAW FUNCTION

def DRAWFUNC(time, x, tlim):

fig, sub = plt.subplots()

sub.set\_title("Original function", fontdict = {"weight": "bold"})

sub.plot(time, x, ls = "-", c = "k")

sub.set\_xlabel("Time", fontdict = {"weight": "bold"})

sub.set\_xlim(min(time), tlim)

def DRAWAUTOCOV(x, llim):

ac = acovf(x, unbiased = True, demean = True)

lag = np.arange(len(ac))

fig, sub = plt.subplots()

sub.set\_title("Autocovariance", fontdict = {"weight": "bold"})

sub.plot(lag, ac, ls = "-", c = "k")

sub.set\_xlabel("Time lag", fontdict = {"weight": "bold"})

sub.set\_xlim(min(lag), llim)

def DRAWPS(x, sampling\_frequency, window, separate, scaling, xscale, yscale, xunit, yunit):

for ss in range(separate):

xlen = int(len(x) / separate)

if ss == 0:

freq, ps = periodogram(x[xlen \* ss : xlen \* (ss + 1)], fs = sampling\_frequency, window = window, detrend = "linear", scaling = scaling)

else:

\_, ps\_ss = periodogram(x[xlen \* ss : xlen \* (ss + 1)], fs = sampling\_frequency, window = window, detrend = "linear", scaling = scaling)

ps = ps + ps\_ss

ps = ps / separate

fig, sub = plt.subplots()

sub.set\_title("Power spectrum (" + window + ")", fontdict = {"weight": "bold"})

freq = freq[1 :]

ps = ps[1 :]

xlabel = "Frequency"

if xscale == "log":

freq = np.log10(freq)

xlabel = "log(" + xlabel + ")"

xlabel = xlabel + " [" + str(xunit) + "]"

ylabel = "Phi"

if yscale == "log":

ps = np.log10(ps)

ylabel = "log(" + ylabel + ")"

if xscale == "log" and yscale == "log":

nm = separate \* 2

if window == "hann":

dof = 8 \* nm / 3

elif window == "parzen":

dof = 3.7086 \* nm

print("D.O.F:", dof)

chi1 = input("Chi square 0.025:")

chi2 = input("Chi square 0.975:")

conf1 = np.log10(dof / float(chi1))

conf2 = np.log10(dof / float(chi2))

sub.fill\_between(freq, ps + conf1, ps + conf2, facecolor = "r", alpha = 0.8, interpolate = True)

sub.text(min(freq), max(ps), "D.O.F.=%.1f" % dof, ha = "left", va = "top")

if scaling == "density":

yunit = yunit + "/" + xunit

ylabel = ylabel + " [" + str(yunit) + "]"

sub.plot(freq, ps, ls = "-", lw = 0.5, c = "k")

sub.set\_xlabel(xlabel, fontdict = {"weight": "bold"})

sub.set\_ylabel(ylabel, fontdict = {"weight": "bold"})

sub.set\_xlim(min(freq), max(freq))

first\_idx = np.argmax(ps)

sub.text(freq[first\_idx], ps[first\_idx], "%.1f" %(ps[first\_idx]), ha = "center", va = "bottom")

#%% PROBLEM#1

i = np.arange(1000 + 1)

t = 1

x = FUNC1(i = i, t = t)

DRAWFUNC(time = i, x = x, tlim = 50)

DRAWAUTOCOV(x = x, llim = 50)

DRAWPS(x = x, flim = 0.5, sampling\_frequency = 1, window = "bartlett", nfft = None, scaling = "density")

#%% PROBLEM#2

i = np.arange(1000 + 1)

t = 1

x = FUNC2(i = i, t = t)

DRAWFUNC(time = i, x = x, tlim = 50)

DRAWAUTOCOV(x = x, llim = 50)

DRAWPS(x = x, flim = 0.5, sampling\_frequency = 1, window = "bartlett", nfft = None, scaling = "density")

#%% PROBLEM#3

i = np.arange(1000)

x = WHITE(1000)

DRAWFUNC(time = i, x = x, tlim = 50)

DRAWAUTOCOV(x = x, llim = 50)

DRAWPS(x = x, flim = 0.5, sampling\_frequency = 1, window = "bartlett", nfft = None, scaling = "density")

#%% PROBLEM#4

i = np.arange(1000)

x = RED(n = 1000, a = 0.5)

DRAWFUNC(time = i, x = x, tlim = 50)

DRAWAUTOCOV(x = x, llim = 50)

DRAWPS(x = x, flim = 0.5, sampling\_frequency = 1, window = "bartlett", nfft = None, scaling = "density")

#%% PROBLEM#5

data = np.loadtxt("C:/Users/WHITE/Google 드라이브/SNU/2019.1/객관적자료분석/HW5/pus9802.dat")

juli = data[:, 0]

tide = data[:, 1]

DRAWFUNC(time = juli, x = tide, tlim = max(juli))

#%% PROBLEM#5-1

separate = 5

DRAWPS(x = tide, sampling\_frequency = 1, window = "hann",\

separate = separate, scaling = "density", xscale = "log", yscale = "log", xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide, sampling\_frequency = 1, window = "hann",\

separate = separate, scaling = "spectrum", xscale = "log", yscale = None, xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide, sampling\_frequency = 1, window = "parzen",\

separate = separate, scaling = "density", xscale = "log", yscale = "log", xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide, sampling\_frequency = 1, window = "parzen",\

separate = separate, scaling = "spectrum", xscale = "log", yscale = None, xunit = "cph", yunit = r"cm$^2$")

#%% PROBLEM#5-2

separate = 1

DRAWPS(x = tide, sampling\_frequency = 1, window = "hann",\

separate = separate, scaling = "density", xscale = "log", yscale = "log", xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide, sampling\_frequency = 1, window = "hann",\

separate = separate, scaling = "spectrum", xscale = "log", yscale = None, xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide, sampling\_frequency = 1, window = "parzen",\

separate = separate, scaling = "density", xscale = "log", yscale = "log", xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide, sampling\_frequency = 1, window = "parzen",\

separate = separate, scaling = "spectrum", xscale = "log", yscale = None, xunit = "cph", yunit = r"cm$^2$")

#%% PROBLEM#5-3

separate = 5

DRAWPS(x = tide[::12], sampling\_frequency = 1, window = "hann",\

separate = separate, scaling = "density", xscale = "log", yscale = "log", xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide[::12], sampling\_frequency = 1, window = "hann",\

separate = separate, scaling = "spectrum", xscale = "log", yscale = None, xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide[::12], sampling\_frequency = 1, window = "parzen",\

separate = separate, scaling = "density", xscale = "log", yscale = "log", xunit = "cph", yunit = r"cm$^2$")

DRAWPS(x = tide[::12], sampling\_frequency = 1, window = "parzen",\

separate = separate, scaling = "spectrum", xscale = "log", yscale = None, xunit = "cph", yunit = r"cm$^2$")