НИУ «МЭИ»

Институт Радиотехники и Электроники им. В. А. Котельникова

Кафедра Радиотехнических Систем

Контрольная работа №1

по курсу «Методы оптимального приёма сигналов в аппаратуре потребителей СРНС»

Выполнила: Малафеева Д. Д.

Группа: ЭР-12м-19

Москва

2020

**Дано:**

Выборка сигналов, всего M=2048 отсчетов:



- независимые некоррелированные по времени ДБГШ с СКО 

Параметры сигналов  неизвестны, но постоянны на интервале наблюдения.

**Найти:**

 , дисперсию ошибки для полученной оценки 

**Указания:**

* В решении необходимо использовать метод максимального правдоподобия, применять итеративный алгоритм оценивания с помощью дискриминаторов.
* Неинформативные параметры (амплитуда, частота, начальная фаза) считаются информативными и тоже оцениваются:



* Вектор наблюдений:



* Отношение правдоподобия для векторных наблюдений в дискретном времени:



где- матрица дисперсий шумов наблюдений.

**Решение:**

**Теория**

1. Запишем функцию правдоподобия:



1. Теперь распишем :







Тогда :



1. Вернемся к выражению 1.1 и запишем его с учетом приведенных преобразований:



1. Далее рассчитаем все производные функции правдоподобия по составляющим вектора оцениваемых параметров:











1. Теперь найдем все вторые и смешанные производные функции правдоподобия:

* 











* 





* 











* 



1. Далее все рассчитанные производные нужно собрать в вектор (для первых производных) и матрицу (для вторых и смешанных производных) и применить алгоритм оценивания параметров сигналов с помощью дискриминаторов:



Здесь





Расчет производится с помощью метода простой итерации. Критерий окончания:



1. При условии, что ОСШ достаточно большое дисперсию ошибки оценивания параметра- можно найти по формуле:



где - это элемент матрицы Фишера, рассчитываемый по формуле:



здесь - функция вычисления мат. ожидания.

Тогда:



**Математическое моделирование**

1. Начальные данные для алгоритма оценивания параметров с помощью дискриминаторов

Расчёты проведены на языке программирования Python. Построим выборку заданных реализаций на входе приёмника:

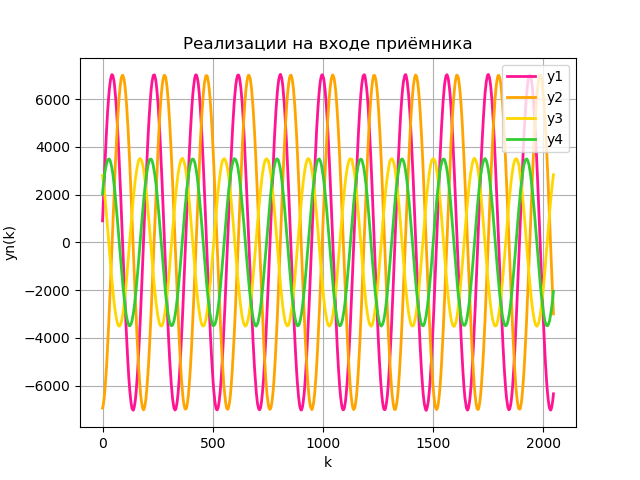


Рис. 1 Реализации на входе приёмника

Необходимо задать начальное значение вектора оцениваемых параметров:



Все параметры выбраны, вероятно, далекими от начальных значений параметров. Однако, мне кажется, это должно повлиять только на количество итераций, за которые оценка параметров достигнет заданной точности.

**Результаты**

(пока их нет)

**Листинг программы**

#!/usr/bin/env python3

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

"""

Created on Mon Oct 19 02:18:01 2020

@author: daryamalafeeva

"""

import numpy as np

import codecs

import matplotlib.pyplot as plt

import math

from numpy.linalg import inv

"""---------------------------Обработка файла-------------------------------"""

# инициализация списков

y1\_list = []

y2\_list = []

y3\_list = []

y4\_list = []

with codecs.open('Input\_Y0toT.txt', "r", encoding='utf-8', errors='ignore') as log:

for line in log:

str\_massive = line.split()

y1 = int(str\_massive[0])

y1\_list.append(y1)

y2 = int(str\_massive[1])

y2\_list.append(y2)

y3 = int(str\_massive[2])

y3\_list.append(y3)

y4 = int(str\_massive[3])

y4\_list.append(y4)

figure\_1 = plt.figure(1)

plt.plot(range(0, len(y1\_list)), y1\_list, color = 'deeppink', linewidth = 2)

plt.plot(range(0, len(y2\_list)), y2\_list, color = 'orange', linewidth = 2)

plt.plot(range(0, len(y3\_list)), y3\_list, color = 'gold', linewidth = 2)

plt.plot(range(0, len(y4\_list)), y4\_list, color = 'limegreen', linewidth = 2)

plt.xlabel('k')

plt.ylabel('yn(k)')

plt.legend(['y1','y2','y3', 'y4'])

plt.title('Реализации на входе приёмника')

plt.grid()

plt.show()

"""--------------------------Параметры моделирования------------------------"""

f\_s = 47.5\*1e6

T = 1/f\_s

M = 2048

sigma\_n = 10

"""---------------------Начальные параметры алгоритма-----------------------"""

# начальные значения параметров вектора lam\_array

A1 = 7030

A2 = 3500

f = 5\*1e5

w = 2 \* math.pi \* f

phi\_0 = math.radians(-60)

delta\_phi = math.radians(-120)

lam\_array = np.array([A1, A2, w, phi\_0, delta\_phi])

S1\_list = []

S2\_list = []

S3\_list = []

S4\_list = []

"""---------------------Алгоритм оценивания параметров----------------------"""

# начальное значение для метода простой итерацииs

delta\_phi\_old = 0.1 \* delta\_phi

for k in range(1, M, 1):

while abs(delta\_phi - delta\_phi\_old) > 1e-8:

# первые производные функции правдоподобия

d1\_dA1 = 1/(sigma\_n\*\*2) \*\

(sum(math.cos(w \* k \* T + phi\_0) \* np.array(y1\_list)+\

math.sin(w \* k \* T + phi\_0) \* np.array(y2\_list)) - M \* A1)

d1\_dA2 = 1/(sigma\_n\*\*2) \*\

(sum(math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)+\

math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)) - M \* A2)

d1\_dw = 1/(sigma\_n\*\*2) \*\

(sum(-A1 \* math.sin(w \* k \* T + phi\_0) \* k \* T \* np.array(y1\_list)+\

A1 \* math.cos(w \* k \* T + phi\_0) \* k \* T \* np.array(y2\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y3\_list)+\

A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y4\_list)))

d1\_dphi\_0 = 1/(sigma\_n\*\*2) \*\

(sum(-A1 \* math.sin(w \* k \* T + phi\_0) \* np.array(y1\_list)+\

A1 \* math.cos(w \* k \* T + phi\_0) \* np.array(y2\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)+\

A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)))

d1\_ddelta\_phi = 1/(sigma\_n\*\*2) \*\

(sum(-A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)+\

A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)))

L = np.array([d1\_dA1, d1\_dA2, d1\_dw, d1\_dphi\_0, d1\_ddelta\_phi])

# вторые и смешанные производные функции правдоподобия

d2\_dA1 = (-M)/(sigma\_n\*\*2)

d2\_dA1dA2 = 0

d2\_dA1dw = 1/(sigma\_n\*\*2) \*\

(sum(-math.sin(w \* k \* T + phi\_0) \* k \* T \* np.array(y1\_list)+\

math.cos(w \* k \* T + phi\_0) \* k \* T \* np.array(y2\_list)))

d2\_dA1dphi\_0 = 1/(sigma\_n\*\*2) \*\

(sum(-math.sin(w \* k \* T + phi\_0) \* np.array(y1\_list)+\

math.cos(w \* k \* T + phi\_0) \* np.array(y2\_list)))

d2\_dA1ddelta\_phi = 0

d2\_dA2 = (-M)/(sigma\_n\*\*2)

d2\_dA2dw = 1/(sigma\_n\*\*2) \*\

(sum(-math.sin(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y3\_list)+\

math.cos(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y4\_list)))

d2\_A2dphi\_0 = 1/(sigma\_n\*\*2) \*\

(sum(-math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)+\

math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)))

d2\_dA2ddelta\_phi = 1/(sigma\_n\*\*2) \*\

(sum(-math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)+\

math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)))

d2\_dw = 1/(sigma\_n\*\*2) \*\

(sum(-A1 \* math.cos(w \* k \* T + phi\_0) \* ((k \* T)\*\*2) \* np.array(y1\_list)-\

A1 \* math.sin(w \* k \* T + phi\_0) \* ((k \* T)\*\*2) \* np.array(y2\_list)-\

A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* ((k \* T)\*\*2) \* np.array(y3\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* ((k \* T)\*\*2) \* np.array(y4\_list)))

d2\_dwdphi\_0 = 1/(sigma\_n\*\*2) \*\

(sum(-A1 \* math.cos(w \* k \* T + phi\_0) \* k \* T \* np.array(y1\_list)-\

A1 \* math.sin(w \* k \* T + phi\_0) \* k \* T \* np.array(y2\_list)-\

A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y3\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y4\_list)))

d2\_dwddelta\_phi = 1/(sigma\_n\*\*2) \*\

(sum(-A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y3\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* k \* T \* np.array(y4\_list)))

d2\_dphi\_0 = 1/(sigma\_n\*\*2) \*\

(sum(-A1 \* math.cos(w \* k \* T + phi\_0) \* np.array(y1\_list)-\

A1 \* math.sin(w \* k \* T + phi\_0) \* np.array(y2\_list)-\

A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)))

d2\_dphi\_0ddelta\_phi = 1/(sigma\_n\*\*2) \*\

(sum(-A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)))

d2\_ddelta\_phi = 1/(sigma\_n\*\*2) \*\

(sum(-A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y3\_list)-\

A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi) \* np.array(y4\_list)))

H = np.array([[d2\_dA1, d2\_dA1dA2, d2\_dA1dw, d2\_dA1dphi\_0, d2\_dA2ddelta\_phi],

[d2\_dA1dA2, d2\_dA2, d2\_dA2dw, d2\_A2dphi\_0, d2\_dA2ddelta\_phi],

[d2\_dA1dw, d2\_dA2dw, d2\_dw, d2\_dwdphi\_0, d2\_dwddelta\_phi],

[d2\_dA1dphi\_0, d2\_A2dphi\_0, d2\_dwdphi\_0, d2\_dphi\_0, d2\_dphi\_0ddelta\_phi],

[d2\_dA2ddelta\_phi, d2\_dA2ddelta\_phi, d2\_dwddelta\_phi, d2\_dphi\_0ddelta\_phi, d2\_ddelta\_phi]])

# метод дискриминаторов

lam\_array\_old = lam\_array

lam\_array = lam\_array - np.dot(L,H)

# обновляем параметры

A1 = lam\_array[0]

A2 = lam\_array[1]

w = lam\_array[2]

pho\_0 = lam\_array[3]

delta\_phi\_old = delta\_phi

delta\_phi = lam\_array[4]

# выход из цикла while

# ошибка оценивания delta\_phi

J = -H

D\_lambda = inv(J)

D\_delta\_phi = D\_lambda[4,4]

# сигналы с оценками параметров

S1 = A1 \* math.cos(w \* k \* T + phi\_0)

S1\_list.append(S1)

S2 = A1 \* math.sin(w \* k \* T + phi\_0)

S2\_list.append(S2)

S3 = A2 \* math.cos(w \* k \* T + phi\_0 + delta\_phi)

S3\_list.append(S3)

S4 = A2 \* math.sin(w \* k \* T + phi\_0 + delta\_phi)

S4\_list.append(S4)

figure\_2 = plt.figure(2)

plt.plot(range(0, len(S1\_list)), S1\_list, color = 'deeppink', linewidth = 2)

plt.plot(range(0, len(S2\_list)), S2\_list, color = 'orange', linewidth = 2)

plt.plot(range(0, len(S3\_list)), S3\_list, color = 'gold', linewidth = 2)

plt.plot(range(0, len(S4\_list)), S4\_list, color = 'limegreen', linewidth = 2)

plt.xlabel('k')

plt.ylabel('Sn(k)')

plt.legend(['S1','S2','S3', 'S4'])

plt.title('Сигналы с оценками параметров ')

plt.grid()

plt.show()