Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

(ФГАОУ ВО СПбПУ)

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

**ЛАБОРАТОРНАЯ РАБОТА №9**

**"функции antiShift и antiSpike"**

по дисциплине «Методы обработки экспериментальных данных»

Выполнил   
студент гр.5140203/30102 М. С. Антоненко

Преподаватель И. Н. Белых

Санкт-Петербург 2023

**Задание на лабораторную работу:**

В классе PROCESSING реализовать две функции редактирования данных:

1) функцию procData=antiShift(data, N, …) для обнаружения и удаления смещения в данных data путем нахождения среднего значения (центра рассеивания) и вычитания его из всех значений данных data. Рекомендуется использовать любые смоделированные данные, смещенные не фрагментарно, а по всей длине выборки в интервале [0, N].

2) В классе PROCESSING реализовать функцию procData=antiSpike(data, N, R, …) для обнаружения и удаления неправдоподобных значений за пределами задаваемого диапазона R в аддитивных моделях данных data:

а) noise() + spikes();

б) harm() + spikes().

Для подавления неправдоподобных значений 𝑥𝑘, выходящих за пределы диапазона R рекомендуется использовать простейший 3- х точечный фильтр линейной интерполяции.

**Алгоритм программы**

|  |  |
| --- | --- |
| № | proccessing.py |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | import numpy as np  import math  class Proccessing:  @staticmethod  def antiShift(data):  out\_data = []  avg = sum(data) / len(data)  for i in range(len(data)):  out\_data.append(data[i] - avg)  return out\_data  @staticmethod  def antiSpike(data):  out\_data = data  for i in range(1, len(out\_data)-1):  if math.fabs(out\_data[i]\*5) > math.fabs(out\_data[i+1]):  out\_data[i] = (out\_data[i-1]+out\_data[i+1])/2  return out\_data |
|  | main.py |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | import time  import matplotlib.pyplot as plt  from classes.model import Model  from classes.proccessing import Proccessing  plt.rcParams["figure.figsize"] = [20, 7.5]  plt.rcParams["figure.autolayout"] = True  def main():  new\_model = Model()  N = 10 \*\* 3 # Число точек  A0 = 5 # Амплитуда  f0 = 50 # Частота гармонического процесса  dt = 0.001 #Постоянная времени  harm = new\_model.harm(N, A0, f0, dt)  noise = new\_model.noise(N, 10)  harm = new\_model.shift(harm, 100, 0, N)  noise = new\_model.shift(noise, 100, 0, N)  harm = new\_model.spikes(harm, N, 5, 50, time.time())  noise = new\_model.spikes(noise, N, 5, 50, time.time())  harm\_source = Proccessing.antiShift(harm)  noise\_source = Proccessing.antiShift(noise)  harm\_source = Proccessing.antiSpike(harm\_source)  noise\_source = Proccessing.antiSpike(noise\_source)  fig, ax = plt.subplots(nrows=2, ncols=2)  fig.suptitle("Задание 9", fontsize=15)  ax[0, 0].plot(harm)  ax[1, 0].plot(noise)  ax[0, 1].plot(harm\_source)  ax[1, 1].plot(noise\_source)  ax[0, 0].set\_title("harm")  ax[1, 0].set\_title("noise")  ax[0, 1].set\_title("harm\_after")  ax[1, 1].set\_title("noise\_after")  plt.show()  main() |

**Полученные результаты**

Изображение выглядит как линия, снимок экрана, текст, График

Автоматически созданное описание

Рисунок 1 ‒ Результат работы программы 09\_antiShift.py

В ходе выполнения программы создаются массивы данных, соответствующие гармоническому и случайному процессу (строки 19-20), функция shift и spike создают сдвиг и случайные выбросы, согласно входным параметрам (смещение 100, 5 случайных выбросов) (строки 21-24), далее с помощью функций antiShift и antiSpike убираются смещение и случайные выбросы (строки 25-28). Построение графика полученных значений представлен строками 30‒40 листинга.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

(ФГАОУ ВО СПбПУ)

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

**ЛАБОРАТОРНАЯ РАБОТА №10**

**"функции antiTrendLinear, antiTrendNonLinear и antiNoise"**

по дисциплине «Методы обработки экспериментальных данных»

Выполнил   
студент гр.5140203/30102 М. С. Антоненко

Преподаватель И. Н. Белых

Санкт-Петербург 2023

**Задание на лабораторную работу:**

1) В классе PROCESSING реализовать функцию

procData = antiTrendLinear (data, N, …) для удаления линейного тренда путем вычисления первой производной данных data аддитивной модели линейного тренда и гармонического процесса и отобразить результат на графике.

2) В классе PROCESSING реализовать функцию procData = antiTrendNonLinear (data, N, W, …) для выделения и удаления нелинейного тренда методом скользящего среднего из данных аддитивной модели data={𝑥𝑘} нелинейного тренда и полигармонического процесса и отобразить результат на графике

Для подавления неправдоподобных значений 𝑥𝑘, выходящих за пределы диапазона R рекомендуется использовать простейший 3- х точечный фильтр линейной интерполяции.

3) В классе PROCESSING реализовать функцию procData=antiNoise(data, N, M, …) для подавления случайного шума методом накопления - путем поэлементного сложения и осреднения M реализаций случайного шума 𝑑𝑎𝑡𝑎𝑚 = 𝑥(𝑡)𝑚 длины N, сгенерированных с использованием реализованной ранее функции noise (data, N, R,…) и отобразить графики осредненных реализаций для M=1,10,100,10000 и для каждого M вычислить значения стандартного отклонения σ𝑀 осредненной реализации; случай для M=1 означает одну исходную реализацию шума.

4) Эмпирическим путем определить аналитическую зависимость изменения значения σ𝑀, и для наглядности отобразив график этой зависимости от M с инкрементом 10: M=1,10, 20,…,100,…,1000,…

5) Повторить п.1 для реализаций 𝑥(𝑡)𝑚 в виде аддитивной модели случайного шума noise (…) и гармонического процесса harm (…) с отображением графиков.

**Алгоритм программы**

|  |  |
| --- | --- |
| № | proccessing.py |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | @staticmethod  def antiTrendLinear(data, N):  out\_data = []  for i in range(N - 1):  out\_data.append(data[i + 1] - data[i])  return out\_data  @staticmethod  def antiTrendNonLinear(data, N, W):  out\_data = []  for i in range(N - W):  x\_n = 0  for k in range(W):  x\_n += data[i + k]  x\_n = x\_n / W  out\_data.append(x\_n)  return out\_data  @staticmethod  def antiNoise(data, N, M):  out\_data = []  for i in range(N):  m = 0  for j in range(M):  m += data[j]  m = m / M  out\_data.append(m)  return out\_data |
|  | main.py |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | import matplotlib.pyplot as plt  from classes.model import Model  from classes.proccessing import Proccessing  plt.rcParams["figure.figsize"] = [20, 7.5]  plt.rcParams["figure.autolayout"] = True  def main():  new\_model = Model()  N = 10 \*\* 3 # число точек  A0 = 100 # амплитуда первой частоты  f0 = 50 # первая частота  A1 = 25 # амплитуда второй частоты  f1 = 130 # вторая частоота  A2 = 30 # амплитуда второй частоты  f2 = 15 # вторая частота  dt = 0.001 # постоянная времени  linear = new\_model.trend\_linear(N, 3, 0)  harm = new\_model.harm(N, A0, f0, dt)  non\_linear = new\_model.exp(N, 0.006, 2)  polyharm = new\_model.polyHarm(N, A0, f0, A1, f1, A2, f2, dt)  noise = new\_model.noise(N, 10)  addSignal\_1 = new\_model.addModel(linear, harm, N)  addSignal\_2 = new\_model.addModel(non\_linear, polyharm, N)  addSignal\_3 = Proccessing.antiTrendLinear(addSignal\_1, N)  addSignal\_4 = Proccessing.antiTrendNonLinear(addSignal\_2, N, 100)  non\_noise = Proccessing.antiNoise(noise, N, 10)  fig, ax = plt.subplots(nrows=3, ncols=2)  fig.suptitle("Задание 10", fontsize=15)  ax[0, 0].plot(addSignal\_1)  ax[1, 0].plot(addSignal\_2)  ax[2, 0].plot(noise)  ax[0, 1].plot(addSignal\_3)  ax[1, 1].plot(addSignal\_4)  ax[2, 1].plot(non\_noise)  ax[0, 0].set\_title("harm")  ax[1, 0].set\_title("noise")  ax[0, 1].set\_title("harm\_after")  ax[1, 1].set\_title("noise\_after")  plt.show()  main() |

**Полученные результаты**

Изображение выглядит как снимок экрана, текст, линия, График

Автоматически созданное описание

Рисунок 2 ‒ Результат работы программы 10\_antiTrend.py

В результате работы программы происходит обработка входных данных, а именно удаляется линейный тренд из аддитивной модели (первая строка графиков), выделяется нелинейный тренд (вторая строка графиков) и подавляется шум (3 строка графиков). В первом случае использовалась функция antiTrendLinear, поскольку результирующий сигнал возрастает линейно, то проходя по массиву данных и декрементируя каждое последующее значение предыдущим, можно получить исходные данные. Аналогичный результат может быть достигнут дифференцированием исходной функции, поскольку она представлена суммой линейной и нелинейной составляющей. Во втором случае методом скользящего среднего исходная функция сглаживается. В третьем случае методом накопления сглаживается шум.

На рисунке 3 приведены графики выделения нелинейного тренда для трех значений длины окна W: 10, 100 и 500. Видно, что с увеличением окна, становится более гладким, однако растет и трудоемкость операций за счет увеличения итераций цикла.

На рисунке 4 представлены результаты сглаживания шума при трех значениях M: 1, 10, 100. Видно, что с увеличением числа реализаций случайного шума полученное значение исходных приближается к реальному значению, так в текущей реализации белого шума [-10; 10] на 1000 точек, среднее значение для M = 1, 10 и 100 равно -7, 0,6 и -0,15 соответственно.

Изображение выглядит как текст, линия, снимок экрана, График

Автоматически созданное описание

Рисунок 3 ‒ Результаты с разными длинами окон на графиках

Изображение выглядит как снимок экрана, текст, линия, График

Автоматически созданное описание

Рисунок 4 ‒ Графики осредненных реализаций для различных M

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

(ФГАОУ ВО СПбПУ)

Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

**ЛАБОРАТОРНАЯ РАБОТА №11**

**"функция convolModel и cardiogram "**

по дисциплине «Методы обработки экспериментальных данных»

Выполнил   
студент гр.5140203/30102 М. С. Антоненко

Преподаватель И. Н. Белых

Санкт-Петербург 2023

**Задание на лабораторную работу:**

1) В классе Model реализовать функцию дискретной свертки y = convolModel(x, N, h, M,…) заданных функций 𝑥(𝑡) = {𝑥𝑘} и ℎ(𝑡) = {ℎ𝑘}

2) На основе понятий линейных систем реализовать первое приближение модели кардиограммы длительностью 4 сек с помощью функции свертки convolModel() импульсной реакции модели сердечной мышцы и управляющей функции ритма, для этого:

а) импульсную реакцию модели сердечной мышцы реализовать с помощью функции мультипликативной модели h(t)=multModel(h1, h2, M,…) гармонического процесса harm() и нисходящего экспоненциального тренда trend()

б) управляющую функцию ритма 𝑥(𝑡) задать в виде четырех импульсов минимальной длительности с амплитудами 1 ± 0.1, следующих через равные интервалы времени, т.е. в виде массива данных длины N=1000, состоящего из нулей за исключением 4-х значений на позициях кратных 200, что при шаге dt=0.005 [сек] равно 1 сек.

в) Отобразить графики трех функций в разных окнах.

3) Используя функцию x(t)=spikes() и функцию h(t) из п.2а, а также функцию свертки из п.1 смоделировать и отобразить «патологическую» кардиограмму.

**Алгоритм программы**

|  |  |
| --- | --- |
| № | model.py |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | def convolModel(self, x, h, N, M):  out\_data = []  for i in range(N):  y = 0  for j in range(M):  y += x[i - j] \* h[j]  out\_data.append(y)  return out\_data  def cardiogram(self, N, f, dt, a):  harm = self.harm(N, 1, f, dt)  exp\_trend = self.exp(N, -a \* dt, 1)  h = self.multModel(harm, exp\_trend, N)  h\_n = []  max\_h = max(h)  for i in range(len(h)):  h\_n.append(h[i] / max\_h \* 120)  return h\_n  def rhythm(self, N, M, R, Rs):  x\_t = [0 for \_ in range(N)]  for i in range(N):  if i % M == 0 and i != 0:  x\_t[i] = random.random() \* 2 \* Rs + (R - Rs)  return x\_t |
|  | main.py |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | import matplotlib.pyplot as plt  from classes.model import Model  plt.rcParams["figure.figsize"] = [20, 7.5]  plt.rcParams["figure.autolayout"] = True  def main():  new\_model = Model()  N = 10 \*\* 3 # число точек  M = 200 #  a = 30 # степень экспаненты  f = 7 # частота гармонического процесса  R = 1 # амплитуда импульсов  Rs = 0.1 # разброс пиков  dt = 0.005 # шаг  h = new\_model.cardiogram(N, f, dt, a)  x = new\_model.rhythm(N, M, R, Rs)  convolution = new\_model.convolModel(x, h, N, M)  fig, ax = plt.subplots(nrows=3, ncols=1)  fig.suptitle("Задание 11", fontsize=15)  ax[0].plot(h)  ax[1].plot(x)  ax[2].plot(convolution)  plt.show()  main() |

**Полученные результаты**

На рисунках 5 и 6 приведены результаты работы программы в виде 3х графиков. Верхний график представляет собой импульсную реакцию модели сердечной мышцы реализовать с помощью функции мультипликативной модели гармонического процесса и нисходящего экспоненциального тренда. Второй график является управляющую функцию ритма 𝑥(𝑡) задать в виде четырех импульсов минимальной длительности с амплитудами 1 ± 0.1, следующих через равные интервалы времени. Нижний график есть результат свертки этих двух функций и является аналогом кардиограммы. На рисунке 6 изображен аналогичный результат, только для патологической кардиограммы, для чего, вместо равностоящих импульсов, генерировались псевдослучайные пики амплитудой 1 с помощью функции spikes().

Изображение выглядит как текст, диаграмма, линия, Параллельный

Автоматически созданное описание

Рисунок 5 ‒ Графики функций импульсной реакции, равностоящих пиков и кардиограммы

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 6 ‒ Патологическая кардиограмма