**Структура проекта**

* ssaCore.py – модуль реализующий SSA алгоритм;
* pars.py – модуль реализующий парсер файла с данными;
* main.py – главный модуль.

**Описание классов**

Класс **SSA –** реализуетSSA алгоритм.

Методы:

* \_\_init\_\_(F, L) – инициализация класса. F: ndarray – данные для обработки, L: int – длинна окна
* ts(Xi) – усреднение побочных диагоналей и перевод матрицы в временный ряд. Xi: ndarray – матрица для перевода.
* getContributions() – получить вклад элементарных матриц в ряд
* decompose() - Создание траекторной матрицы и проведение SVD
* reconstruct(comp) – восстановить ряд. comp: ndarray – компоненты по которым производится восстановление, значение по умолчанию: [1, 2, 3, …, 9, 10].
* getFilt() – возвращает восстановленный ряд
* getTrend() – возвращает восстановленный тренд
* getPeriod() – возвращает восстановленный период
* forecast(steps) – прогноз. steps: int – количество шагов для прогноза.

Класс **Parser** – реализует парсер файла с данными.

Методы:

* read(fileName) – прочитать файл. fileName: string – путь к файлу с данными. Возвращает количество строк в файле.
* getDate() – возвращает массив с датой
* getData(self, name) – получить необходимые данные. name: string – название поля с данными, возможные значения: "X (A)", "Y (A)", "UTC-UT1 (A)", "X (B)", "Y (B)", "UTC-UT1 (B) ". Возвращает соответствующие данные

**Приложение**

**Исходный код приложения**

**ssaCore.py**

import numpy as np

import scipy.stats as stats

import scipy.linalg as linalg

class SSA():

def \_\_init\_\_(self, F, L):

self.F = F

self.N = len(F)

self.L = L

self.decompose()

self.reconstruct()

def ts(self, Xi):

"""

Усредняем побочные диоганали элементарной матрицы

и переводим в временной ряд

"""

Xrev = Xi[::-1]

return np.array([Xrev.diagonal(i).mean() for i in range(-Xi.shape[0]+1, Xi.shape[1])])

def getContributions(self):

lambdas = np.power(self.s,2)

norm = np.linalg.norm(self.X)

cont = [(lambdas[i]/(norm\*\*2)).round(4) for i in range(len(self.s))]

cont = {i:cont[i] for i in range(len(cont)) if cont[i]>0}

return cont

def decompose(self):

"""

Создание траекторной матрицы и SVD

"""

self.K = self.N - self.L + 1

self.X = np.column\_stack([self.F[i:i+self.L] for i in range(0,self.K)])

self.d = np.linalg.matrix\_rank(self.X)

self.U, self.s, self.V = np.linalg.svd(self.X)

self.V = self.V.T

self.sContributions = self.getContributions()

self.r = len(self.sContributions)

self.orthonormalBase = {i:self.U[:,i] for i in range(self.r)}

def reconstruct(self, comps = np.arange(10)):

Xs = np.array([self.ts(self.s[i] \* \

np.outer(self.U[:,i], self.V[:,i])) for i in comps])

self.tsRec = np.zeros(len(Xs[0]))

for i in range(len(Xs)):

self.tsRec += Xs[i]

def getFilt(self):

return self.tsRec

def getTrend(self):

self.reconstruct([0])

return self.tsRec

def getPeriod(self):

self.reconstruct(np.arange(1,10))

return self.tsRec

def forecast(self, steps):

self.verticalityCoeff = 0

self.R = np.zeros(self.orthonormalBase[0].shape)[:-1]

for Pi in self.orthonormalBase.values():

pi = np.ravel(Pi)[-1]

self.verticalityCoeff += pi\*\*2

self.R += pi\*Pi[:-1]

self.R = np.matrix(self.R/(1-self.verticalityCoeff))

self.tsForecast = self.tsRec

for i in range(self.N + steps):

if i >= self.N:

Z = np.array([self.tsForecast[j] for j in range(i-self.L, i-1)])

x = self.R @ Z

self.tsForecast = np.append(self.tsForecast, x)

return self.tsForecast

**pars.py**

from datetime import datetime

class Parser():

def read(self, fileName):

"""

Считать файл

"""

self.year = []

self.month = []

self.day = []

self.date = []

self.MJD = []

self.IERSpolar = []

self.IERSutc = []

self.aPMx = []

self.aPMy = []

self.aUTC = []

self.bPMx = []

self.bPMy = []

self.bUTC = []

fileLen = 0

f = open(fileName, "r")

for line in f:

# Считать mjd

mjd = float(line[7:15].replace(" ", ""))

self.MJD.append(mjd)

# Год, месяц, день

yr = line[0:2].replace(" ", "")

if len(yr) == 1:

yr = "0" + yr

if mjd < 51544.0:

yr = "19" + yr

self.year.append(int(yr))

else:

yr = "20" + yr

self.year.append(int(yr))

mon = line[2:4].replace(" ", "")

self.month.append(int(mon))

d = line[4:6].replace(" ", "")

self.day.append(int(d))

# Сформировать дату

dt = datetime.strptime(yr+"/"+mon+"/"+d, "%Y/%m/%d")

self.date.append(dt)

self.IERSpolar = line[17].replace(" ", "")

self.IERSutc.append(line[57].replace(" ", ""))

# A: X, Y, UTC

x = line[18:27].replace(" ", "")

try:

if x[0] == "-":

self.aPMx.append(float("-"+"0"+x[1:]))

else:

self.aPMx.append(float("0"+x))

except:

self.aPMx.append(self.aPMx[-1])

y = line[37:46].replace(" ", "")

try:

if y[0] == "-":

self.aPMy.append(float("-"+"0"+y[1:]))

else:

self.aPMy.append(float("0"+y))

except:

self.aPMy.append(self.aPMy[-1])

u = line[58:68].replace(" ", "")

try:

if u[0] == "-":

if u[1] == "0":

self.aUTC.append(float(u))

else:

self.aUTC.append(float("-"+"0"+y[1:]))

else:

if u[0] == "0":

self.aUTC.append(float(u))

else:

self.aUTC.append(float("0"+u))

except:

self.aUTC.append(self.aUTC[-1])

# B: X, Y, UTC

x = line[134:144].replace(" ", "")

try:

if x[0] == "-":

self.bPMx.append(float("-"+"0"+x[1:]))

else:

self.bPMx.append(float("0"+x))

except:

self.bPMx.append(self.bPMx[-1])

y = line[144:154].replace(" ", "")

try:

if y[0] == "-":

self.bPMy.append(float("-"+"0"+y[1:]))

else:

self.bPMy.append(float("0"+y))

except:

self.bPMy.append(self.bPMy[-1])

u = line[154:165].replace(" ", "")

try:

if u[0] == "-":

if u[1] == "0":

self.bUTC.append(float(u))

else:

self.bUTC.append(float("-"+"0"+y[1:]))

else:

if u[0] == "0":

self.bUTC.append(float(u))

else:

self.bUTC.append(float("0"+u))

except:

self.bUTC.append(self.bUTC[-1])

fileLen += 1

return fileLen

def getDate(self):

return self.date

def getData(self, name):

"""

Получить необходимые данные

Возвращает соответствующие данные

"""

if name == "X (A)":

return self.aPMx

if name == "X (B)":

return self.bPMx

if name == "Y (A)":

return self.aPMy

if name == "Y (B)":

return self.bPMy

if name == "UTC-UT1 (A)":

return self.aUTC

if name == "UTC-UT1 (B)":

return self.bUTC

**main.py**

import tkinter as tk

from tkinter import ttk

import tkinter.filedialog

import numpy as np

import math

import matplotlib

import matplotlib.pyplot as plt

import datetime

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk

from matplotlib.figure import Figure

matplotlib.use("TkAgg")

import pars

import ssaCore

date = []

mainData = []

data = []

forc = []

filt = []

trend = []

period = []

par = pars.Parser()

filename = ""

length = 0

mode = -1

plt.rcParams["date.autoformatter.hour"] = "%Y-%m-%d"

def drawPlot():

global data, forc, filt, mode, trend, period, date, mainData

if mode == -1:

return

begin, end = getSize()

if mode == 0:

fig.clf()

ax = fig.add\_subplot(111)

if cvar1.get():

ax.plot(date[begin:end], mainData[begin:end], linestyle='-', marker='o')

else:

ax.plot(date[begin:end], mainData[begin:end])

ax.set\_title("Начальные данные")

fig.tight\_layout()

canvas.draw()

elif mode == 1:

fig.clf()

ax1 = fig.add\_subplot(221)

if cvar1.get():

ax1.plot(date[begin:end], mainData[begin:end], linestyle='-', marker='o')

else:

ax1.plot(date[begin:end], mainData[begin:end])

ax1.set\_title("Начальные данные")

ax3 = fig.add\_subplot(222)

newDate = date[begin:end]

for i in range(len(forc) - len(filt)):

newDate.append(newDate[-1]+datetime.timedelta(days=1))

t = newDate[end-begin-1:]

b = len(forc)-len(t)

if cvar1.get():

ax3.plot(t, forc[b:], color = "red", label = "Предсказанные", linestyle='-', marker='o')

ax3.plot(date[begin:end], filt, label = "Отфильтрованные", linestyle='-', marker='o')

else:

ax3.plot(t, forc[b:], color = "red", label = "Предсказанные")

ax3.plot(date[begin:end], filt, label = "Отфильтрованные")

ax3.set\_title("Предсказанные данные")

ax3.legend()

ax4 = fig.add\_subplot(223)

if cvar1.get():

ax4.plot(date[begin:end], mainData[begin:end], label = "Оригинальные", linestyle='-', marker='o')

ax4.plot(date[begin:end], trend, color = "red", label = "Тренд", linestyle='-', marker='o')

else:

ax4.plot(date[begin:end], mainData[begin:end], label = "Оригинальные")

ax4.plot(date[begin:end], trend, color = "red", label = "Тренд")

ax4.set\_title("Тренд")

ax4.legend()

ax4 = fig.add\_subplot(224)

if cvar1.get():

ax4.plot(date[begin:end], mainData[begin:end], label = "Оригинальные", linestyle='-', marker='o')

ax4.plot(date[begin:end], period, color = "red", label = "Период", linestyle='-', marker='o')

else:

ax4.plot(date[begin:end], mainData[begin:end], label = "Оригинальные")

ax4.plot(date[begin:end], period, color = "red", label = "Период")

ax4.set\_title("Период")

ax4.legend()

canvas.draw()

elif mode == 2:

fig.clf()

ax = fig.add\_subplot(111)

newDate = date[begin:end]

for i in range(len(forc) - len(filt)):

newDate.append(newDate[-1]+datetime.timedelta(days=1))

t = newDate[end-begin-1:]

t2 = date[begin:end+len(t)-1]

b = len(forc)-len(t)

if cvar1.get():

ax.plot(t, forc[b:], color = "red", label = "Предсказанные", linestyle='-', marker='o')

ax.plot(date[begin:end], filt, label = "Отфильтрованные", linestyle='-', marker='o')

ax.plot(t2, mainData[begin:end+len(t)-1], color = "black", label = "Фактические", linestyle='-', marker='o')

else:

ax.plot(t, forc[b:], color = "red", label = "Предсказанные")

ax.plot(date[begin:end], filt, label = "Отфильтрованные")

ax.plot(t2, mainData[begin:end+len(t)-1], color = "black", label = "Фактические")

ax.set\_title("Прогноз-факт")

ax.legend()

canvas.draw()

def getSize():

try:

begin = int(beginT.get())

except:

beginT.delete(0,tk.END)

beginT.insert(0,"0")

return -1, -1

try:

end = int(endT.get())

except:

endT.delete(0,tk.END)

endT.insert(0,str(length))

return -1, -1

if begin >= end:

beginT.delete(0,tk.END)

beginT.insert(0,"0")

endT.delete(0,tk.END)

endT.insert(0,str(length))

return -1, -1

if end < 1 or end > length:

endT.delete(0,tk.END)

endT.insert(0,str(length))

return -1, -1

if begin < 0:

beginT.delete(0,tk.END)

beginT.insert(0,"0")

return -1, -1

return begin, end

def uploadFile(event=None):

"""

Загрузить файл

"""

global filename, length, date

filename = tkinter.filedialog.askopenfilename()

cvar2.set(0)

try:

length = par.read(filename)

labelText.set(filename.split("/")[-1] + " Длина: " + str(length))

lenT.delete(0,tk.END)

lenT.insert(0,math.floor(length/4))

beginT.delete(0,tk.END)

beginT.insert(0,"0")

endT.delete(0,tk.END)

endT.insert(0,str(length))

date = par.getDate()

showData("\_")

except:

labelText.set("Неверный файл")

def showData(\_=""):

"""

Показать данные

"""

global mainData, data, forc, mode

cvar2.set(0)

data = par.getData(combBox.get())

mainData = data

forc = []

begin, end = getSize()

if begin == -1:

return

mode = 0

drawPlot()

def forecast():

"""

Предсказать и вывести

"""

global length, data, filt, forc, mode, trend, period

cvar2.set(0)

begin, end = getSize()

if begin == -1:

return

try:

steps = int(forecastT.get())

L = int(lenT.get())

if L > (end - begin)/2 or L < 1:

lenT.delete(0,tk.END)

lenT.insert(0,math.floor((end - begin)/2))

return

except:

return

data = par.getData(combBox.get())

ssa = ssaCore.SSA(data[begin:end], L)

filt = ssa.getFilt()

forc = ssa.forecast(steps)

trend = ssa.getTrend()

period = ssa.getPeriod()

mode = 1

drawPlot()

data = data[begin:end]

def save():

"""

Сохранить текущие данные

"""

global data, forc

f = tkinter.filedialog.asksaveasfile(mode='w', defaultextension=".txt")

if f is None:

return

mjd = par.MJD

while len(mjd) < len(data)+len(forc):

mjd.append(mjd[-1]+1)

pred = []

while len(pred) < len(data):

pred.append("D")

while len(pred) < len(forc):

pred.append("P")

text = ""

if len(forc) == 0:

l = len(data)

else:

l = len(forc)

data = forc

for i in range(l):

text += str(mjd[i]) + " " + str(data[i]) + " " + str(pred[i]) + "\n"

f.write(text)

f.close()

def fact():

global mainData, length, data, filt, forc, mode

if cvar2.get():

try:

begin, end = getSize()

if begin == -1:

return

absolute = 0

relative = 0

j = 0

for i in range(len(filt), len(forc)):

a = abs(mainData[end+j]-forc[i])

absolute += a

relative += abs(a / mainData[end+j])

j+=1

errorText.set("Абсолютная средняя ошибка: " + str(round(absolute/j,2)) + \

"\n Относительная средняя ошибка: " +str(round(relative/j,1)\*100) +"%")

mode = 2

drawPlot()

except:

errorText.set("Нет данных для сравнения")

cvar2.set(0)

else:

mode = 1

drawPlot()

root = tk.Tk()

root.minsize(900, 500)

root.title("SSA")

frame = tk.Frame()

frame.pack()

labelText = tk.StringVar(root)

labelText.set("Выберите файл с данными")

messageBox = tk.Label(frame, textvariable=labelText, background = "gray85")

messageBox.pack(side=tk.LEFT, padx=(10), pady=(5))

fileChoose = tk.Button(frame, text='Выбрать файл', command=uploadFile)

fileChoose.pack(side=tk.LEFT, padx=(10), pady=(5))

saveData = tk.Button(frame, text='Сохранить данные', command=save)

saveData.pack(side=tk.LEFT, padx=(10), pady=(5))

frame = tk.Frame()

frame.pack()

messageBox1 = tk.Label(frame, text = "Использовать:", background = "gray85")

messageBox1.pack(side=tk.LEFT, padx=(10, 5), pady=(5))

combBox = ttk.Combobox(frame, width=12, values=["X (A)", "Y (A)", "UTC-UT1 (A)", "X (B)", "Y (B)", "UTC-UT1 (B)"])

combBox.pack(side=tk.LEFT, padx=(0, 10), pady=(5))

combBox.current(0)

combBox.bind('<<ComboboxSelected>>', showData)

beginL = tk.Label(frame, text="От:", background = "gray85")

beginL.pack(side=tk.LEFT, padx=(10, 5), pady=(5))

beginT = tk.Entry(frame, width=5)

beginT.pack(side=tk.LEFT, padx=(0, 10), pady=(5))

endL = tk.Label(frame, text="До:", background = "gray85")

endL.pack(side=tk.LEFT, padx=(10, 5), pady=(5))

endT = tk.Entry(frame, width=5)

endT.pack(side=tk.LEFT, padx=(0, 10), pady=(5))

calcLn = tk.Button(frame, text='Показать данные', command=showData)

calcLn.pack(side=tk.LEFT, padx=(10), pady=(5))

frame = tk.Frame()

frame.pack()

enterL = tk.Label(frame, text="Длина окна:", background = "gray85")

enterL.pack(side=tk.LEFT, padx=(10, 5), pady=(5))

lenT = tk.Entry(frame, width=5)

lenT.pack(side=tk.LEFT, padx=(0, 10), pady=(5))

forcastB = tk.Button(frame, text='Предсказать на', command=forecast)

forcastB.pack(side=tk.LEFT, padx=(10, 5), pady=(5))

forecastT = tk.Entry(frame, width=5)

forecastT.pack(side=tk.LEFT, padx=(0, 10), pady=(5))

cvar1 = tk.BooleanVar()

cvar1.set(0)

c1 = tk.Checkbutton(frame, text="Отображать маркеры", variable=cvar1, onvalue=1, offvalue=0, command=drawPlot)

c1.pack(side=tk.LEFT, padx=(10), pady=(5))

frame = tk.Frame()

frame.pack()

errorText = tk.StringVar(root)

errorText.set("Абсолютная средняя ошибка:\n Относительная средняя ошибка:")

errorL = tk.Label(frame, textvariable=errorText, background = "gray85")

errorL.pack(side=tk.LEFT, padx=(10), pady=(5))

cvar2 = tk.BooleanVar()

cvar2.set(0)

c2 = tk.Checkbutton(frame, text="Прогноз-факт", variable=cvar2, onvalue=1, offvalue=0, command=fact)

c2.pack(side=tk.LEFT, padx=(10), pady=(5))

fig = Figure(figsize=(4, 3), dpi=100)

fig.tight\_layout()

canvas = FigureCanvasTkAgg(fig, master=root)

canvas.get\_tk\_widget().pack(side=tk.LEFT, fill=tkinter.BOTH, expand=1)

toolbar = NavigationToolbar2Tk(canvas, root)

toolbar.update()

canvas.get\_tk\_widget().pack(side=tk.TOP, fill=tkinter.BOTH, expand=1)

root.mainloop()