Текст программы «Программа для моделирования процесса ионно-плазменного нанесения оксинитридных тонких пленок при одновременном распыление мишеней двух химических элементов».

Основной модуль:

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

# %% Настройка питона

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import scipy.constants as cnst

import matplotlib.pylab as mpl

from scipy import interpolate

import target\_setup as ts

import Model

import matplotlib.colors as colors

mpl.rcParams['pdf.fonttype'] = 42

mpl.rcParams['font.size'] = 10

mpl.rcParams['xtick.labelsize'] = 12

mpl.rcParams['ytick.labelsize'] = 12

plt.style.use('seaborn-whitegrid')

mpl.rc('font', size=13)

mpl.rcParams['font.family'] = "serif"

mpl.rcParams['axes.linewidth'] = 2

# %% Переменные

A\_Si = 0.004 # Площадь кремниевой мишени (м2)

A\_Mo = 0.004 # Площадь молибденовой мишени (м2)

# Подверженная воздействию потока частиц площадь поверхности камеры Si и Si3N4 (м2)

A\_chamber\_Si = 0.3

# Подверженная воздействию потока частиц площадь поверхности камеры Mo и MoN (м2)

A\_chamber\_Mo = 0.3

S\_Si = 0.9 # Коэффициент распыления Si для E(Ar+) = 400 эВ

S\_Si3N4 = 0.3 # Коэффициент распыления SiN для E(Ar+) = 400 эВ

S\_Mo = 0.9 # Коэффициент распыления Mo для E(Ar+) = 400 эВ

S\_MoN = 0.3 # Коэффициент распыления MoN для E(Ar+) = 400 эВ

S\_O2 = 1200/1000 # Скорость перекачки (м3/с)

alpha0\_Si = 1 # Коэффициент задерживания молекулы кислорода на непрореагировавшей поверхности Si

# Коэффициент задерживания молекулы кислорода на прореагировавшей поверхности Si3N4

alpha0\_Si3N4 = 0.01

# Коэффициент задерживания молекулы кислорода на прореагировавшей поверхности Mo

alpha0\_Mo = 0.1

# Коэффициент задерживания молекулы кислорода на прореагировавшей поверхности MoN

alpha0\_MoN = 0.01

# Плотность потока ионов аргона, вызывающих распыление с поверхности мишени Si (А\*м^-2)

J\_Si = 165

# Плотность потока ионов аргона, вызывающих их распыление с поверхности мишени Mo (А\*м^-2)

J\_Mo = 85

T = 273 # Температура в К

# %% Определение пределов моделирования

df = pd.DataFrame([])

df["P\_O2"] = np.arange(start=0.0, stop=0.003 \* 133.322368, step=0.00001)

# %% Интерполяция силы тока из экспериментальных данных по Si

x\_Si = np.array([0, 0.0001, 0.0003, 0.0006, 0.001, 0.0015,

0.002, 0.0025, 0.003]) \* 133.322368

y\_Si = np.array([0.66, 0.69, 0.72, 0.79, 0.95, 1.0, 1.05, 1.1, 1.14]) / A\_Si

spl = interpolate.UnivariateSpline(x\_Si, y\_Si, k=2)

df["J\_Si"] = spl(df.P\_O2)

# %% Интерполяция силы тока из экспериментальных данных по Mo

x\_Mo = np.array([0.0001, 0.0003, 0.0006, 0.001, 0.003]) \* 133.322368

y\_Mo = np.array([0.27, 0.27, 0.27, 0.26, 0.25]) / A\_Mo

linear\_model = np.polyfit(x\_Mo, y\_Mo, 1)

linear\_model\_fn = np.poly1d(linear\_model)

df["J\_Mo"] = linear\_model\_fn(df.P\_O2)

# %% Интерполяция изменения эффективности распыленя мишеней

input\_df = pd.read\_excel(".\Data\Spray\_rate.xls")

input\_df["P\_O2"] = input\_df["P"] \* 133.322368

input\_df["P"] \*= 1000

Si\_interpolation = interpolate.PchipInterpolator(

input\_df["P\_O2"], input\_df["Si"])

Mo\_interpolation = interpolate.interp1d(

input\_df["P\_O2"], input\_df["Mo"], kind="quadratic")

df["k\_S\_Si"] = Si\_interpolation(df["P\_O2"])

df["k\_S\_Mo"] = Mo\_interpolation(df["P\_O2"])

df["k\_S\_Si\_orig"] = Si\_interpolation(df["P\_O2"])

df["k\_S\_Mo\_orig"] = Mo\_interpolation(df["P\_O2"])

input\_df["Si"] /= input\_df["Si"][0]

input\_df["Mo"] /= input\_df["Mo"][0]

df["k\_S\_Mo"] /= df["k\_S\_Mo"][0]

df["k\_S\_Si"] /= df["k\_S\_Si"][0]

# %% Инициализация мишеней

Si = ts.target(k=3, n=4/3, t = 2, A=A\_Si, A\_chamber=A\_chamber\_Si,

S=S\_Si \* df.k\_S\_Si, S\_compound=S\_Si3N4 \* df.k\_S\_Si,

alpha0=0.1, alpha0\_compound=0.00001, #Азот к Si на мишени

alpha0\_c=0.005, alpha0\_compound\_c=0.00002, #Азот к Si на поверхности

alpha0\_O2 = 0.15, alpha0\_O2\_compound=0.13, #Кислород к Si

J=J\_Si, Ro=3210, M = 140.2833/1000)

Mo = ts.target(k=1, n=1, t = 2, A=A\_Mo, A\_chamber=A\_chamber\_Mo,

S=S\_Mo \* df.k\_S\_Si, S\_compound=S\_MoN \* df.k\_S\_Si,

alpha0=0.6, alpha0\_compound=0.00001, #Азот к молибдену на мишени

alpha0\_c=0.01, alpha0\_compound\_c=0.000002, #Азот к молибдену на поверхности

alpha0\_O2 = 0.01, alpha0\_O2\_compound=0.09, #Кислород к молибдену

J=J\_Mo, Ro=9300, M = 109.95/1000)

Si\_Mo = Model.model(target\_1=Si, target\_2=Mo, T=273, S = 1200 \* 1E-3,

moleclar\_mass=28/1000/cnst.Avogadro)

# %% Расчёт характеристической функции и потока

Sl = pd.DataFrame()

Sl["P\_O2"] = df.P\_O2

df["dq/dP"] = Si\_Mo.dq\_dp(df.P\_O2)

df["S\_l"] = (Si\_Mo.S - df["dq/dP"])\*1000

df["q\_O2"] = Si\_Mo.q\_of\_F\_t\_c(df.P\_O2)/np.sqrt(cnst.pi)

df["flow\_rate"] = df.q\_O2 \* 1E3

# %% Расчёт коэфициента распыления

df["F"] = Si\_Mo.F\_of\_P(df.P\_O2)

df["R\_Si"] = Si.R\_of\_F(Si.N\_target\_of\_F(df.F), in\_nm\_s=True)

df["R\_Mo"] = Mo.R\_of\_F(Mo.N\_target\_of\_F(df.F), in\_nm\_s=True)

# %% Вычисление количества веществ в образце

df["N\_Si"]= Si.N\_target\_of\_F(df.F)

df["N\_Mo"] = Mo.N\_target\_of\_F(df.F)

df["N\_O2"] = Si\_Mo.N\_O2\_of\_P(df.P\_O2, P\_residual=7e-5\*133)

n\_N = pd.DataFrame()

n\_N["a\_1"], n\_N["a\_2"], n\_N["b\_1"], n\_N["b\_2"], n\_N["c\_1"], n\_N["c\_2"], df["N\_N"] = Si\_Mo.N\_gase\_of\_P(

df.P\_O2)

n\_N["sum\_1"] = n\_N["a\_1"] + n\_N["b\_1"] + n\_N["c\_1"]

n\_N["sum\_2"] = n\_N["a\_2"] + n\_N["b\_2"] + n\_N["c\_2"]

# %% Расчёт соотношения элементов

df["N"] = df.N\_Si + df.N\_Mo + df.N\_O2 + df.N\_N

df["C\_Si"] = df.N\_Si / df.N \* 100

df["C\_Mo"] = df.N\_Mo / df.N \* 100

df["C\_N"] = df.N\_N / df.N \* 100

df["C\_O2"] = df.N\_O2 / df.N \* 100

df["P\_O2\_torr"] = df.P\_O2 \* 0.0075 \* 1000

df["P\_O2\_from\_q"] = df.q\_O2/Si\_Mo.S \* 0.0075 \* 1000

# %% Зарузка эксперементаьных значений концентрации

exp\_C = pd.read\_excel("./Data/Atomic\_content.xlsx")

exp\_R = pd.read\_excel("./Data/Spray\_rate\_original.xlsx")

п

# %% Настройи отображения

colors\_list = list(colors.TABLEAU\_COLORS.values())

#%% Вывод тока на кремниевой мишени

plt.plot(x\_Si, y\_Si)

plt.plot(df.P\_O2, df.J\_Si)

plt.legend(["Experiment Si", "Model"])

plt.show()

#%% Вывод тока на кремниевой мишени

plt.plot(x\_Mo, y\_Mo)

plt.plot(df.P\_O2, df.J\_Mo)

plt.legend(["Experiment Mo", "Model"])

plt.show()

#%% Вывод скорости осаждения кремния и молибдена

plt.scatter(exp\_R["P"], y=exp\_R.Si, c=colors\_list[0])

plt.scatter(exp\_R["P"], y=exp\_R.Mo, c=colors\_list[1])

df.R\_Si /= 600

df.R\_Mo /= 42

plt.plot(df["P\_O2\_torr"], df["R\_Si"], label="Si", c=colors\_list[0])

plt.plot(df["P\_O2\_torr"], df["R\_Mo"], label="Mo", c=colors\_list[1])

plt.title("Скорость осаждения Si и Mo")

plt.xlabel("Давление, мТорр")

plt.ylabel("Скорость распыления, нм/c")

plt.legend()

plt.show()

#%% Вывод коэфициентов для коэфициента распыления

plt.scatter(exp\_R["P"], y=exp\_R.Si, c=colors\_list[0])

plt.scatter(exp\_R["P"], y=exp\_R.Mo, c=colors\_list[1])

plt.plot(df["P\_O2\_torr"], df.k\_S\_Si\_orig, label="Si", c=colors\_list[0])

plt.plot(df["P\_O2\_torr"], df.k\_S\_Mo\_orig, label="Mo", c=colors\_list[1])

plt.title("Скорость осаждения Si и Mo")

plt.xlabel("Давление, мТорр")

plt.ylabel("Скорость распыления, нм/c")

plt.legend()

plt.show()

#%% Вывод эксериментальных значений скорости распыления

plt.scatter(exp\_R["P"], y=exp\_R.Si, c="black", marker = '\*')

plt.scatter(exp\_R["P"], y=exp\_R.Mo, c="black", marker = 'o')

plt.plot(df["P\_O2\_torr"], df.k\_S\_Si\_orig, label="Si", c="black", linestyle = "-",)

plt.plot(df["P\_O2\_torr"], df.k\_S\_Mo\_orig, label="Mo", c="black", linestyle = "--",)

plt.title("Скорость осаждения Si и Mo")

plt.xlabel("Давление, мТорр")

plt.ylabel("Скорость распыления, нм/c")

plt.legend()

plt.savefig("./image/ЧБ Скорость осаждения Si и Mo.tiff", format = "tiff", dpi = 300)

plt.show()

#%% Вывод расчитаных зменение интенсивности распыления Si

plt.scatter(input\_df["P"], input\_df["Si"],

label="Experiment", c=colors\_list[5])

plt.plot(df["P\_O2\_torr"], df["k\_S\_Si"], label="Interpolate", c=colors\_list[0])

plt.legend()

plt.title("Изменение интенсивности распыления Si")

plt.xlabel("Давление, мТорр")

plt.ylabel("Эффективность распыления Si")

plt.show()

#%% Вывод расчитаных зменение интенсивности распыления Mo

plt.scatter(input\_df["P"], input\_df["Mo"],

label="Experiment", c=colors\_list[5])

plt.plot(df["P\_O2\_torr"], df["k\_S\_Mo"], label="Interpolate", c=colors\_list[1])

plt.legend()

plt.title("Изменение интенсивности распыления Mo")

plt.xlabel("Давление, мТорр")

plt.ylabel("Эффективность распыления Mo")

plt.show()

#%% Вывод харатеристичской функции и потока азота в зависимости от давения азота

df[["P\_O2", "S\_l"]].plot(x = "P\_O2", legend = None, ylabel = r"Characteristic function $S\_L$ $(L\*s^{-1})$", xlabel = "Oxygen partial pressure (Pa)")

df.plot(x = "flow\_rate", y = "P\_O2", legend = None, ylabel = r"$P\_{O\_2}$ (Pa)", xlabel = "Oxygen flow rate (sccm)")

#%% Вывод количество осаденных атосмов разного сорта

df[["P\_O2\_torr", "N\_Si", "N\_N", "N\_O2", "N\_Mo"]].plot(x="P\_O2\_torr")

plt.show()

#%% Цветная версия вывода расёта содержания элементов

plt.plot(df["P\_O2\_torr"], df.C\_Si, label="Si", c=colors\_list[0])

plt.plot(df["P\_O2\_torr"], df.C\_Mo, label="Mo", c=colors\_list[1])

plt.plot(df["P\_O2\_torr"], df.C\_N, label="N", c=colors\_list[2])

plt.plot(df["P\_O2\_torr"], df.C\_O2, label="O2", c=colors\_list[3])

plt.scatter(exp\_C["мТорр"], y=exp\_C.Si, c=colors\_list[0])

plt.scatter(exp\_C["мТорр"], y=exp\_C.Mo, c=colors\_list[1])

plt.scatter(exp\_C["мТорр"], y=exp\_C.N, c=colors\_list[2])

plt.scatter(exp\_C["мТорр"], y=exp\_C.O, c=colors\_list[3])

plt.xlabel("Давление, мТорр")

plt.ylabel("Содержание элемента, Ат %")

plt.legend(bbox\_to\_anchor=(0., 1.02, 1., .102), loc='lower left',

ncols=4, mode="expand", borderaxespad=0.)

plt.show()

#%% ЧБ версия вывода расёта содержания элементов

plt.plot(df["P\_O2\_torr"], df.C\_Si, label="Si", c="black", linestyle = "-")

plt.scatter(exp\_C["мТорр"], y=exp\_C.Si, label="Si", c="black", marker = '\*')

plt.plot(df["P\_O2\_torr"], df.C\_Mo, label="Mo", c="black", linestyle = "--")

plt.scatter(exp\_C["мТорр"], y=exp\_C.Mo, label="Mo", c="black", marker = 'o')

plt.plot(df["P\_O2\_torr"], df.C\_N, label="N", c="black", linestyle = "-.")

plt.scatter(exp\_C["мТорр"], y=exp\_C.N, label="N", c="black", marker = 'v')

plt.plot(df["P\_O2\_torr"], df.C\_O2, label="O2", c="black", linestyle = ":")

plt.scatter(exp\_C["мТорр"], y=exp\_C.O, label="O2", c="black", marker = 'h')

plt.xlabel("Давление, мТорр")

plt.ylabel("Содержание элемента, Ат %")

plt.legend(bbox\_to\_anchor=(0., 1.02, 1., .102), loc='lower left',

ncols=4, mode="expand", borderaxespad=0.)

plt.savefig("./image/ЧБ Расчёт содержания элекментов.tiff", format = "tiff", dpi = 300)

plt.show()

#%% Экспериментальное содержание элекментов ЧБ

plt.plot(exp\_C["мТорр"], exp\_C.Si, label="Si", c="black", marker = '\*', linestyle = "-")

plt.plot(exp\_C["мТорр"], exp\_C.Mo, label="Mo", c="black", marker = 'o', linestyle = "--")

plt.plot(exp\_C["мТорр"], exp\_C.N, label="N", c="black", marker = 'v', linestyle = "-.")

plt.plot(exp\_C["мТорр"], exp\_C.O, label="O2", c="black", marker = 'h', linestyle = ":")

plt.xlabel("Давление, мТорр")

plt.ylabel("Содержание элемента, Ат %")

plt.legend(bbox\_to\_anchor=(0., 1.02, 1., .102), loc='lower left',

ncols=4, mode="expand", borderaxespad=0.)

plt.savefig("./image/ЧБ Экспериментальное содержание элекментов.tiff", format = "tiff", dpi = 300)

plt.show()

#%% Перерасчёт с разными значениями остаточного авления кислорода

s = ["10 ", "30 ", "70 "]

for i, P\_O2 in enumerate([1e-5\*133, 3e-5\*133, 7e-5 \* 133]):

df["N\_Si"]= Si.N\_target\_of\_F(df.F)

df["N\_Mo"] = Mo.N\_target\_of\_F(df.F)

df["N\_O2"] = Si\_Mo.N\_O2\_of\_P(df.P\_O2, P\_residual=P\_O2)

n\_N = pd.DataFrame()

n\_N["a\_1"], n\_N["a\_2"], n\_N["b\_1"], n\_N["b\_2"], n\_N["c\_1"], n\_N["c\_2"], df["N\_N"] = Si\_Mo.N\_gase\_of\_P(

df.P\_O2)

n\_N["sum\_1"] = n\_N["a\_1"] + n\_N["b\_1"] + n\_N["c\_1"]

n\_N["sum\_2"] = n\_N["a\_2"] + n\_N["b\_2"] + n\_N["c\_2"]

df["N"] = df.N\_Si + df.N\_Mo + df.N\_O2 + df.N\_N

df["C\_Si"] = df.N\_Si / df.N \* 100

df["C\_Mo"] = df.N\_Mo / df.N \* 100

df["C\_N"] = df.N\_N / df.N \* 100

df["C\_O2"] = df.N\_O2 / df.N \* 100

df["P\_O2\_torr"] = df.P\_O2 \* 0.0075 \* 1000

df["P\_O2\_from\_q"] = df.q\_O2/Si\_Mo.S \* 0.0075 \* 1000

if i == 0:

plt.plot(df["P\_O2\_torr"], df.C\_Si, label="Si", alpha = 0.3, c="black", linestyle = "-")

plt.plot(df["P\_O2\_torr"], df.C\_Mo, label="Mo", alpha = 0.3, c="black", linestyle = "--")

plt.plot(df["P\_O2\_torr"], df.C\_N, label="N", alpha = 0.3, c="black", linestyle = "-.")

plt.plot(df["P\_O2\_torr"], df.C\_O2, label="O2", c="black", linestyle = ":")

else:

plt.plot(df["P\_O2\_torr"], df.C\_Si, alpha = 0.3, c="black", linestyle = "-")

plt.plot(df["P\_O2\_torr"], df.C\_Mo, alpha = 0.3, c="black", linestyle = "--")

plt.plot(df["P\_O2\_torr"], df.C\_N, alpha = 0.3, c="black", linestyle = "-.")

plt.plot(df["P\_O2\_torr"], df.C\_O2, c="black", linestyle = ":")

plt.text(3.03, df.C\_Si[39996] - 0.7, s[i], size = 7)

plt.text(3.03, df.C\_Mo[39996] - 0.7, s[i], size = 7)

plt.text(3.03, df.C\_N[39996] - 0.7, s[i], size = 7)

plt.xlabel("Давление, мТорр")

plt.ylabel("Содержание элемента, Ат %")

plt.legend(bbox\_to\_anchor=(0., 1.02, 1., .102), loc='lower left',

ncols=4, mode="expand", borderaxespad=0.)

plt.savefig("./image/ЧБ Экспериментальное содержание элекментов", format = "tiff", dpi = 300)

plt.text(2, 3, s[0] +" мкТорр")

plt.text(0.7, 19, s[1] + " мкТорр")

plt.text(1, 35, s[2] + " мкТорр")

plt.xlim(-0.01, 3.3)

plt.ylim(-0.01, 60)

plt.savefig("./image/ЧБ Разные остаточные давления кислорода.tiff", format = "tiff", dpi = 300)

plt.show()

#%% Перерасчёт с разными значениями тока кремниевой мишени

for i, J\_Si in enumerate([50, 120, 200]):

Si.J = J\_Si

df["N\_Si"]= Si.N\_target\_of\_F(df.F)

df["N\_Mo"] = Mo.N\_target\_of\_F(df.F)

df["N\_O2"] = Si\_Mo.N\_O2\_of\_P(df.P\_O2, P\_residual=0.933e-2)

df["N\_N"] = Si\_Mo.N\_gase\_of\_P(df.P\_O2)[-1]

df["N"] = df.N\_Si + df.N\_Mo + df.N\_O2 + df.N\_N

df["C\_Si"] = df.N\_Si / df.N \* 100

df["C\_Mo"] = df.N\_Mo / df.N \* 100

df["C\_N"] = df.N\_N / df.N \* 100

df["C\_O2"] = df.N\_O2 / df.N \* 100

df["P\_O2\_torr"] = df.P\_O2 \* 0.0075 \* 1000

if i == 0:

plt.plot(df["P\_O2\_torr"], df.C\_Si, label="Si", c="black", linestyle = "-")

plt.plot(df["P\_O2\_torr"], df.C\_Mo, label="Mo", alpha = 0.3, c="black", linestyle = "--")

plt.plot(df["P\_O2\_torr"], df.C\_N, label="N", alpha = 0.3, c="black", linestyle = "-.")

plt.plot(df["P\_O2\_torr"], df.C\_O2, label="O2", alpha = 0.3, c="black", linestyle = ":")

else:

plt.plot(df["P\_O2\_torr"], df.C\_Si, c="black", linestyle = "-")

plt.plot(df["P\_O2\_torr"], df.C\_Mo, alpha = 0.3, c="black", linestyle = "--")

plt.plot(df["P\_O2\_torr"], df.C\_N, alpha = 0.3, c="black", linestyle = "-.")

plt.plot(df["P\_O2\_torr"], df.C\_O2, alpha = 0.3, c="black", linestyle = ":")

plt.text(3.03, df.C\_Mo[39996] - 0.7, str(J\_Si), size = 7)

plt.text(3.03, df.C\_O2[39996] - 0.7, str(J\_Si), size = 7)

plt.text(3.03, df.C\_N[39996] - 0.7, str(J\_Si), size = 7)

plt.xlabel("Давление, мТорр")

plt.ylabel("Содержание элемента, Ат %")

plt.legend(bbox\_to\_anchor=(0., 1.02, 1., .102), loc='lower left',

ncols=4, mode="expand", borderaxespad=0.)

plt.savefig("./image/ЧБ Экспериментальное содержание элекментов", format = "tiff", dpi = 300)

plt.text(2, 6, r"$J\_{Si}$ = 50 $A/м^{2}$")

plt.text(2.1, 29, r"$J\_{Si}$ = 120 $A/м^{2}$")

plt.text(2.1, 38, r"$J\_{Si}$ = 200 $A/м^{2}$")

plt.xlim(-0.01, 3.3)

plt.ylim(-0.01, 60)

plt.savefig("./image/ЧБ Разные токи Si.tiff", format = "tiff", dpi = 300)

plt.show()

#%% Перерасчёт с разными значениями тока молибденой мишени

Si.J = 165

for i, J\_Mo in enumerate([50, 120, 200]):

Mo.J = J\_Mo

df["N\_Si"]= Si.N\_target\_of\_F(df.F)

df["N\_Mo"] = Mo.N\_target\_of\_F(df.F)

df["N\_O2"] = Si\_Mo.N\_O2\_of\_P(df.P\_O2, P\_residual=0.933e-2)

df["N\_N"] = Si\_Mo.N\_gase\_of\_P(df.P\_O2)[-1]

df["N"] = df.N\_Si + df.N\_Mo + df.N\_O2 + df.N\_N

df["C\_Si"] = df.N\_Si / df.N \* 100

df["C\_Mo"] = df.N\_Mo / df.N \* 100

df["C\_N"] = df.N\_N / df.N \* 100

df["C\_O2"] = df.N\_O2 / df.N \* 100

df["P\_O2\_torr"] = df.P\_O2 \* 0.0075 \* 1000

if i == 0:

plt.plot(df["P\_O2\_torr"], df.C\_Si, label="Si", alpha = 0.3, c="black", linestyle = "-")

plt.plot(df["P\_O2\_torr"], df.C\_Mo, label="Mo", c="black", linestyle = "--")

plt.plot(df["P\_O2\_torr"], df.C\_N, label="N", alpha = 0.3, c="black", linestyle = "-.")

plt.plot(df["P\_O2\_torr"], df.C\_O2, label="O2", alpha = 0.3, c="black", linestyle = ":")

else:

plt.plot(df["P\_O2\_torr"], df.C\_Si, alpha = 0.3, c="black", linestyle = "-")

plt.plot(df["P\_O2\_torr"], df.C\_Mo, c="black", linestyle = "--")

plt.plot(df["P\_O2\_torr"], df.C\_N, alpha = 0.3, c="black", linestyle = "-.")

plt.plot(df["P\_O2\_torr"], df.C\_O2, alpha = 0.3, c="black", linestyle = ":")

plt.text(3.03, df.C\_Si[39996] - 0.7, str(J\_Mo), size = 7)

plt.text(3.03, df.C\_O2[39996] - 0.7, str(J\_Mo), size = 7)

plt.text(3.03, df.C\_N[39996] - 0.7, str(J\_Mo), size = 7)

plt.xlabel("Давление, мТорр")

plt.ylabel("Содержание элемента, Ат %")

plt.legend(bbox\_to\_anchor=(0., 1.02, 1., .102), loc='lower left',

ncols=4, mode="expand", borderaxespad=0.)

plt.savefig("./image/ЧБ Экспериментальное содержание элекментов", format = "tiff", dpi = 300)

plt.text(0.5, 3, r"$J\_{Mo}$ = 50 $A/м^{2}$")

plt.text(0.6, 9.5, r"$J\_{Mo}$ = 120 $A/м^{2}$")

plt.text(0.8, 16, r"$J\_{Mo}$ = 200 $A/м^{2}$")

plt.xlim(-0.01, 3.3)

plt.ylim(-0.01, 60)

plt.savefig("./image/ЧБ Разные токи Mo.tiff", format = "tiff", dpi = 300)

plt.show()

Модуль target\_setup

Описывает объект мишени и функции рассчитываемые для мишени.

import math

import scipy.constants as cnst

class target:

def \_\_init\_\_(self, k, n, t, A, A\_chamber, S, S\_compound, alpha0,

alpha0\_compound, alpha0\_c, alpha0\_compound\_c, alpha0\_O2, alpha0\_O2\_compound, J, Ro, M):

"""

Инициалиирует мешени, камеру вокруг нее и молекулярный поток направленый в нее

Parameters

----------

k : TYPE

Количество атомов мишени в соединение (в Cr2O3 - 2).

n : TYPE

Количество атомов окислителя в соединение на один атом мишени (в Cr2O3 - 1.5).

A : TYPE

Площадь мишени (м2).

A\_chamber : TYPE

Подверженность поверхности камеры воздействию потока частиц (м2).

S : TYPE

Коэффициент распыления материала мишени.

S\_compoud : TYPE

Коэффициент распыления прореагировавшего материала.

alpha0 : TYPE

Коэффициент задерживания молекулы окислителя на непрореагировавшей поверхности мишени.

alpha0\_c: TYPE

alpha0 для поверхности осаждения.

alpha0\_compound : TYPE

Коэффициент задерживания молекулы окислителя на прореагировавшей поверхности мишени.

alpha0\_compound\_с : TYPE

alpha0\_compound для поверхности осаждения.

J : TYPE

Плотность потока ионов аргона распыляющих мешень (молекул/(м2\*с)).

Ro : Type

Плотность вещества мишени (кг/м^3).

M : Type

Молярная масса вещества мишени (кг/м).

Returns

-------

None.

"""

self.k = k

self.n = n

self.t = t

self.A = A

self.A\_chamber = A\_chamber

self.S = S

self.S\_compound = S\_compound

self.alpha0 = alpha0

self.alpha0\_c = alpha0\_c

self.alpha0\_compound = alpha0\_compound

self.alpha0\_compound\_c = alpha0\_compound\_c

self.alpha0\_O2 = alpha0\_O2

self.alpha0\_O2\_compound = alpha0\_O2\_compound

self.J = J

self.Ro = Ro

self.M = M

def t\_of\_F (self, F):

"""

Возвращает t = (1 - theta\_t)

Parameters

----------

F : TYPE

Молекулярный поток реактивного газа (молекул/(м2\*с)).

Returns

-------

t : TYPE

t = (1 - theta\_t)

"""

kn = self.k / self.n

Je = self.J / cnst.elementary\_charge

a = kn \* self.alpha0 \* F

b = Je \* self.S\_compound

t = (a/b + 1)\*\*-1

return t

def c\_of\_F (self, F):

"""

Возвращает c = (1 - theta\_c).

Parameters

----------

F : TYPE

Молекулярный поток реактивного газа (молекул/(м2\*с)).

Returns

-------

c : TYPE

c = (1 - theta\_c).

"""

kn = self.k / self.n

eF = cnst.elementary\_charge \* F

A\_t\_c = self.A / self.A\_chamber

alpha0 = self.alpha0\_c

a = (self.S/self.S\_compound) \* (A\_t\_c)

b1 = (kn \* alpha0 \* (eF / (self.J \* self.S\_compound)))\*\*2

b2 = eF / (self.J \* self.S\_compound) \* (kn \* alpha0 + A\_t\_c \* kn \* alpha0)

b3 = self.S/self.S\_compound \* A\_t\_c

c = a / (b1 + b2 + b3)

return c

def theta\_t\_of\_F(self, F):

"""

Возвращает долю прореагировавшей поверхности мишени из коэффициент задерживания молекулы окислителя на поверхности мишени.

Parameters

----------

F : TYPE

Молекулярный поток реактивного газа (молекул/(м2\*с)) .

Returns

-------

tetha : TYPE

Доля прореагировавшей поверхности мишени.

"""

k = self.k

n = self.n

alpha0\_target = self.alpha0

alpha0\_compound = self.alpha0\_compound

J = self.J

S\_compound = self.S\_compound

kn = k/n

a = kn \* alpha0\_target \* F

b = (kn \* F \* (alpha0\_target - alpha0\_compound)) + (J / cnst.elementary\_charge \* S\_compound)

tetha\_t = a / b

return tetha\_t

def theta\_c\_of\_F(self, F):

kn = self.k / self.n

Je = self.J / cnst.elementary\_charge

A\_tc = self.A/ self.A\_chamber

a = (kn \* self.alpha0 \* F) + (Je \* self.S\_compound \* self.theta\_t\_of\_F(F) \* A\_tc)

b = (kn \* self.alpha0 \* F) + (Je \* self.S\_compound \* self.theta\_t\_of\_F(F) \* A\_tc)

c = -1 \* (kn \* self.alpha0\_compound \* F) + (Je \* self.S \* (1 - self.theta\_t\_of\_F(F)) \* A\_tc)

tetha\_c = a / (b + c)

return tetha\_c

def R\_of\_F (self, F, in\_nm\_s = False):

"""

Возвращает распылённый поток из мишени

Parameters

----------

F : TYPE

Молекулярный поток реактивного газа (молекул/(м2\*с)) .

Returns

-------

R : TYPE

Распылённый из мишени поток (Атомы) / (ионы врезавшиеся в мишень \* секунды \* метр^2).

"""

S\_compoud = self.S\_compound

tetha\_t = self.theta\_t\_of\_F(F)

S = self.S

t = self.t\_of\_F(F)

Je = self.J / cnst.elementary\_charge

R = Je \* (S\_compoud \* tetha\_t + S \* t)

if in\_nm\_s:

R = (R / cnst.Avogadro \* self.M) / self.Ro \* 1e9

return R

return R

def N\_target\_of\_F (self, F):

"""

Возвращает колличество осевщего вещества мишении.

Parameters

----------

F : TYPE

Молекулярный поток реактивного газа (молекул/(м2\*с)) .

Returns

-------

N : TYPE

Колличество осевщего вещества мишении.

"""

Je = self.J / cnst.elementary\_charge

S\_compoud = self.S\_compound

tetha\_t = 1 - self.t\_of\_F(F)

S = self.S

A\_t = self.A

A\_c = self.A\_chamber

Je = self.J / cnst.elementary\_charge

N = (Je \* S\_compoud \* tetha\_t \* (self.k + self.t) + Je \* S \* (1-tetha\_t) ) \* A\_t/A\_c

return N

if \_\_name\_\_ == "\_\_main\_\_":

Ti = target(k = 2, n = 2, A = 0.031, A\_chamber = 0.2,

S = 0.4, S\_compound = 0.015,

alpha0 = 1.0, alpha0\_compound = 0.01, J = 60)

print(Ti.t\_of\_F(10))

print(Ti.c\_of\_F(10))

Модуль Model

Описывает модель осаждения на двух ранее определённых мишенях.

import math

import scipy.constants as cnst

import target\_setup as ts

class model:

def \_\_init\_\_(self, target\_1: ts.target , target\_2: ts.target, T: float,

moleclar\_mass: float, S: float, K1: float = 3.7e-21):

"""

Parameters

----------

target\_1 : ts.target

Первая мишень.

target\_2 : ts.target

Вторая мишень.

T : float

Абсолютная температура реакционного газа (K).

moleclar\_mass : float

Молекулярная масса реактивного газа (кг).

S : float

Скорость накачки подаваемого газа (м3/c).

alpha0\_O2 : float

Коэффициент задерживания молекулы кислорода.

K1 : float, optional

Коэффициент пересчёта. The default is 3.7e-21.

Returns

-------

None.

"""

self.target\_1 = target\_1

self.target\_2 = target\_2

self.T = T

self.moleclar\_mass = moleclar\_mass

self.S = S

self.K1 = K1

def q\_of\_P (P: float, S: float):

"""

Возвращает скорость потка газа, зная давление и скорость накачки.

Parameters

----------

P : float

Давление подаваемого газа (Па).

S : float

Скорость накачки подаваемого газа (м3/c).

Returns

-------

q : float

Скорость потока газа (Па\*м3/с).

"""

q = P\*S

return q

def F\_of\_P (self, P: float):

"""

Возвращает молекулярнй поток реагирующего газа.

Parameters

----------

P : float

Давление подаваемого газа (Па).

T : float

Абсолютная температура реакционного газа (K).

M : float

Молекулярная масса реактивного газа (кг).

Returns

-------

F : float

Молекулярный поток реактивного газа (молекул/(м2\*с))

"""

F = P/math.sqrt(2 \* cnst.pi \* cnst.Boltzmann \* self.T \* self.moleclar\_mass)

return F

def K\_calc (self):

"""

Возвращает К, нужный для характеристической функции

Parameters

----------

T : float

Абсолютная температура реакционного газа (K).

M : float

Молекулярная масса реактивного газа (кг).

K1 : float, optional

Коэфициент пересчета. The default is 3.7e-21.

Returns

-------

K : float

DESCRIPTION.

"""

K = math.sqrt(2 \* cnst.pi \* cnst.Boltzmann \* self.T \* self.moleclar\_mass)/self.K1

return K

def q\_of\_F\_t\_c (self, P):

"""

Вычисляет поток кислорада (Pa\*m^3/s)

Parameters

----------

P : TYPE

Давление подаваемого газа (Па).

Returns

-------

Поток кислорада (Pa\*m^3/s).

"""

F = self.F\_of\_P(P)

S\_a = self.S \* self.K\_calc()

a1 = self.target\_1.t\_of\_F(F) \* self.target\_1.alpha0 \* self.target\_1.A

a2 = self.target\_2.t\_of\_F(F) \* self.target\_2.alpha0 \* self.target\_2.A

a3 = self.target\_1.c\_of\_F(F) \* self.target\_1.alpha0 \* self.target\_1.A\_chamber

a4 = self.target\_2.c\_of\_F(F) \* self.target\_2.alpha0 \* self.target\_2.A\_chamber

q\_O2 = self.K1 \* F \* (a1 + a2 + a3 + a4 + S\_a)

return q\_O2

def dq\_dp (self, P):

"""

Возвращает производную потока от давления.

Parameters

----------

P : TYPE

Давление (Па).

Returns

-------

dq\_dp : TYPE

Производную потока от давления.

"""

F = self.F\_of\_P(P)

t\_1 = self.target\_1.t\_of\_F(F)

t\_2 = self.target\_2.t\_of\_F(F)

c\_1 = self.target\_1.c\_of\_F(F)

c\_2 = self.target\_2.c\_of\_F(F)

a1 = self.target\_1.alpha0 \* self.target\_1.A\_chamber \* c\_1\*\*2 \* ( (self.target\_1.S\_compound/self.target\_1.S) \* (self.target\_1.A\_chamber/self.target\_1.A) \* ((1 - t\_1)/t\_1)\*\*2 - 1)

a2 = self.target\_2.alpha0 \* self.target\_2.A\_chamber \* c\_2\*\*2 \* ( (self.target\_2.S\_compound/self.target\_2.S) \* (self.target\_2.A\_chamber/self.target\_2.A) \* ((1 - t\_2)/t\_2)\*\*2 - 1)

a3 = self.target\_1.alpha0 \* self.target\_1.A \* t\_1\*\*2

a4 = self.target\_2.alpha0 \* self.target\_1.A \* t\_2\*\*2

dq\_dp = self.S - (1/self.K\_calc()) \* (a1 + a2 - a3 - a4)

return dq\_dp

def qt\_of\_P(self, P, target: ts.target):

"""

Часть потока идущая на мишень.

Parameters

----------

P : TYPE

Давление (Па).

target : ts.target

Мишень.

Returns

-------

qt : TYPE

Поток на мишень.

"""

F = self.F\_of\_P(P)

t = target.t\_of\_F(F)

qt = F \* t \* target.alpha0 \* target.A

return qt

def qc\_of\_P(self, P, target: ts.target):

"""

Часть потока идущая на камеру.

Parameters

----------

P : TYPE

Давление (Па).

target : ts.target

Мишень.

Returns

-------

qt : TYPE

Поток на камеру.

"""

F = self.F\_of\_P(P)

c = target.c\_of\_F(F)

qc = F \* c \* target.alpha0 \* target.A\_chamber

return qc

def qS\_of\_P(self, P):

"""

Часть потока идущая в систему откачки.

Parameters

----------

P : TYPE

Давление (Па).

Returns

-------

qt : TYPE

Поток на камеру.

"""

S\_a = self.S \* self.K\_calc()

qS = self.F\_of\_P(P)\*S\_a

return qS

def N\_gase\_of\_P(self, P):

"""

Возвращает колличество реакционного газа в образце.

Parameters

----------

P : TYPE

Давление.

Returns

-------

Колличество реакционного газа в образце.

"""

t\_1 = self.target\_1

t\_2 = self.target\_2

je\_1 = t\_1.J/cnst.elementary\_charge

je\_2 = t\_2.J/cnst.elementary\_charge

S\_1 = t\_1.S\_compound

S\_2 = t\_2.S\_compound

F = self.F\_of\_P(P)

tetha\_t\_1 = 1 - t\_1.t\_of\_F(F)

tetha\_t\_2 = 1 - t\_2.t\_of\_F(F)

tetha\_c\_1 = 1 - t\_1.c\_of\_F(F)

tetha\_c\_2 = 1 - t\_2.c\_of\_F(F)

At\_Ac\_1 = t\_1.A/t\_1.A\_chamber

At\_Ac\_2 = t\_2.A/t\_2.A\_chamber

a\_1 = je\_1 \* S\_1 \* tetha\_t\_1 \* At\_Ac\_1 #Азот из Si3N4 с мишени

a\_2 = je\_2 \* S\_2 \* tetha\_t\_2 \* At\_Ac\_2 #Азот из MoN с мишени

b\_1 = t\_1.alpha0\_c \* F \* (1 - tetha\_c\_1) #Азот захваченный Si в полёте

b\_2 = t\_2.alpha0\_c \* F \* (1 - tetha\_c\_2) #Азот захваченный Mo в полёте

c\_1 = t\_1.alpha0\_compound\_c \* F \* tetha\_c\_1 #Азот захваченный Si3N4 в полёте

c\_2 = t\_2.alpha0\_compound\_c \* F \* tetha\_c\_2 #Азот захваченный MoN в полёте

N = a\_1 + a\_2 + b\_1 + b\_2 + c\_1 + c\_2

#N = N/2

return (a\_1, a\_2, b\_1, b\_2, c\_1, c\_2, N)

def N\_O2\_of\_P (self, P, P\_residual = 0.733e-3, D\_0 = 20.95):

"""

Возвращает количество кислорода в образце.

Parameters

----------

P : TYPE

Давление.

Returns

-------

Колличество реакционного газа в образце.

"""

t\_1 = self.target\_1

t\_2 = self.target\_2

F = self.F\_of\_P(P)

tetha\_t\_1 = 1 - t\_1.t\_of\_F(F)

tetha\_t\_2 = 1 - t\_2.t\_of\_F(F)

tetha\_c\_1 = 1 - t\_1.c\_of\_F(F)

tetha\_c\_2 = 1 - t\_2.c\_of\_F(F)

F = D\_0/100 \* (P\_residual) / math.sqrt(2 \* cnst.pi \* cnst.Boltzmann \* self.T \* 32/1000/cnst.Avogadro)

F = F \* (1 - t\_1.alpha0\_O2 \* (1 - tetha\_t\_1))

O\_Si\_1 = F \* ( t\_1.alpha0\_O2 \* (1 - tetha\_c\_1))

O\_Si\_2 = F \* t\_1.alpha0\_O2\_compound \* tetha\_c\_1

O\_Mo\_1 = F \* (t\_2.alpha0\_O2 \* (1 - tetha\_c\_2))

O\_Mo\_2 = t\_2.alpha0\_O2\_compound \* F \* tetha\_c\_2

N = (O\_Si\_1 + O\_Si\_2 + O\_Mo\_1 + O\_Mo\_2)

return N