Algorithm.py

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

# @Time : 2023/4/5 14:55

# @Author : DanYang

# @File : Algorithm.py

# @Software : PyCharm

import numpy as np

import pandas as pd

from sympy import symbols, Function, summation, Array, diff

from CoolProp.CoolProp import PropsSI

pd.set\_option('display.precision', 13)

R = 0.46126 # kJ/(kg.K)

Tc = 647.096 # K

pc = 22.064 # Mpa

vc = 322 # kg/m^3

test\_data\_2\_3 = [0.623\_150\_000e3, 0.165\_291\_643e2, 'T', 'P']

def test(func, odata):

pre = func(odata[2], odata[0], odata[3])

if abs(pre-odata[1]) >= 10\*\*-4:

raise ValueError(f"Wrong Function old:{odata[1]} pre:{pre}")

else:

print(f'Success old:{odata[1]} pre:{pre}')

def boundary\_between\_2\_3(known, param, goal):

ni = np.array([0.348\_051\_856\_289\_69e3, -0.116\_718\_598\_799\_75e1, 0.101\_929\_700\_393\_26e-2,

0.572\_544\_598\_627\_46e3, 0.139\_188\_397\_788\_70e2])

n = symbols('n')

if known == 'T' and goal == 'P':

fun1 = ni[0] + ni[1]\*n + ni[2]\*n\*\*2

return fun1.subs(n, param)

elif known == 'P' and goal == 'T':

fun1 = ni[3] + ((n-ni[4])/ni[2])\*\*(1/2)

return fun1.subs(n, param)

def boundary\_1(pn, Tn, goal):

T0 = 1386

p0 = 16.53

coe = pd.read\_csv('../../data/boundary\_1.csv')

p = symbols('p', positive=True)

T = symbols('T', positive=True)

j = symbols('j', integer=True)

gamma = Function('gamma')(p, T)

Ii = Array(coe['Ii'])

Ji = Array(coe['Ji'])

ni = Array(coe['ni'])

gamma = R \* T \* summation(ni[j]\*(7.1-p/p0)\*\*Ii[j]\*(T0/T-1.222)\*\*Ji[j], (j, 0, 33))

v = diff(gamma, p, 1) / 1000

h = (gamma - T \* (diff(gamma, T, 1))) \* 1000

s = -(diff(gamma, T, 1)) \* 1000

if goal == 'v':

return v.evalf(subs={p: pn, T: Tn}, n=13)

elif goal == 'h':

return h.evalf(subs={p: pn, T: Tn}, n=13)

elif goal == 's':

return s.evalf(subs={p: pn, T: Tn}, n=13)

def other(p, T, goal):

if goal == 'v':

goal = 'D'

return 1 / PropsSI(goal.upper(), 'T', T, 'P', p\*10\*\*6, 'Water')

return PropsSI(goal.upper(), 'T', T, 'P', p\*10\*\*6, 'Water')

if \_\_name\_\_ == '\_\_main\_\_':

pass

calpage.py

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

# @Time : 2023/4/12 16:05

# @Author : DanYang

# @File : calpage.py

# @Software : PyCharm

import os

import numpy as np

import sys

import pandas as pd

from PyQt5.QtWidgets import (

QApplication, QGridLayout,

QLabel, QDialog,

QHBoxLayout, QGroupBox,

QRadioButton, QTabWidget, QSizePolicy,

QTableWidget, QWidget, QPushButton, QFileDialog,

QAbstractItemView, QTableWidgetItem, QLineEdit,

QVBoxLayout, QCheckBox

)

import Algorithm

import Plot

class Qt\_cal(QDialog):

def \_\_init\_\_(self, parent=None):

super(Qt\_cal, self).\_\_init\_\_(parent=parent)

# original palette

self.originalPalette = QApplication.palette()

# label

label = QLabel("Description:\n"

"1. Calculation module is used to perform individual\n data queries\n"

"2. the plotting module is used to set the plotting\n style\n"

"3. the table module supports inputting a large amount\n of data\n"

"We wish you a pleasant experience!")

self.create\_top\_left\_group()

self.create\_top\_right\_group()

self.create\_bottom\_left\_group()

self.creat\_fileinput()

# layout

top\_layout = QHBoxLayout()

top\_layout.addWidget(label)

main\_layout = QGridLayout()

main\_layout.addLayout(top\_layout, 0, 0, 1, 2)

main\_layout.addWidget(self.top\_left\_group, 1, 0)

main\_layout.addWidget(self.top\_right\_group, 1, 1)

main\_layout.addWidget(self.bottom\_left\_group, 3, 0, 1, 2)

main\_layout.addLayout(self.file\_layout, 2, 0, 1, 2)

self.setFixedSize(700, 1000)

self.setLayout(main\_layout)

self.setWindowTitle("Wvpm")

def cal\_answer(self):

T = float(self.wid1.text())

p = float(self.wid2.text())

result = [Algorithm.other(p, T, i) for i in ['v', 'h', 's']]

answer = "v:{0:.4e}\nh:{1:.3f}\ns:{2:.3f}".format(\*result)

self.screen.setText(answer)

def create\_top\_left\_group(self):

self.top\_left\_group = QGroupBox("Calculation")

layout = QVBoxLayout()

self.wid1 = QLineEdit()

self.wid1.adjustSize()

layout.addWidget(QLabel("T(K)"))

layout.addWidget(self.wid1)

self.wid2 = QLineEdit()

self.wid2.adjustSize()

self.check\_botton = QPushButton("Calculate")

self.check\_botton.clicked.connect(self.cal\_answer)

self.screen = QLabel("")

layout.addWidget(QLabel("p(Mpa)"))

layout.addWidget(self.wid2)

layout.addWidget(self.screen)

layout.addWidget(self.check\_botton)

self.top\_left\_group.setLayout(layout)

def plot\_data(self):

ifT = self.equal\_T.isChecked()

ifp = self.equal\_p.isChecked()

if\_all = self.equal\_all.isChecked()

method = [self.plot\_style\_2.isChecked(), self.plot\_style\_1.isChecked(), self.plot\_style\_3.isChecked()]

nmethod = np.array(['classic', 'dark\_background', 'seaborn'])

nmethod = nmethod[method]

Plot.plot\_data(self.p, self.T, self.nans, self.ans, if\_plot\_p=ifp, if\_plot\_T=ifT, if\_plot\_all=if\_all, method=nmethod)

def create\_top\_right\_group(self):

self.top\_right\_group = QGroupBox("Plot")

layout = QVBoxLayout()

layout1 = QHBoxLayout()

layout2 = QVBoxLayout()

self.equal\_T = QCheckBox("Plot isothermal maps")

self.equal\_p = QCheckBox("Plot isobaric diagram")

self.equal\_all = QCheckBox("Mapping water vapor properties")

layout.addWidget(self.equal\_T)

layout.addWidget(self.equal\_p)

layout.addWidget(self.equal\_all)

self.plot\_style\_1 = QRadioButton("dark\_background")

self.plot\_style\_2 = QRadioButton("classic")

self.plot\_style\_2.setChecked(True)

self.plot\_style\_3 = QRadioButton("seaborn")

self.plot\_button = QPushButton("Plot")

self.plot\_button.clicked.connect(self.plot\_data)

layout2.addWidget(QLabel('Drawing style'))

layout1.addWidget(self.plot\_style\_2)

layout1.addWidget(self.plot\_style\_1)

layout1.addWidget(self.plot\_style\_3)

layout2.addLayout(layout1)

layout2.addWidget(self.plot\_button)

layout.addLayout(layout2)

self.top\_right\_group.setLayout(layout)

def create\_bottom\_left\_group(self):

self.bottom\_left\_group = QTabWidget()

self.bottom\_left\_group.setSizePolicy(QSizePolicy.Preferred, QSizePolicy.Ignored)

tab1 = QWidget()

self.tablewidget = QTableWidget(500, 3)

self.tablewidget.setEditTriggers(QAbstractItemView.NoEditTriggers)

tab1hbox = QHBoxLayout()

tab1hbox.addWidget(self.tablewidget)

tab1.setLayout(tab1hbox)

self.bottom\_left\_group.addTab(tab1, "Data")

def creat\_fileinput(self):

self.file\_layout = QHBoxLayout()

lab1 = QLabel("File Input: (.csv)")

a1 = QPushButton("Select File")

a1.clicked.connect(self.select\_file)

lab1.setBuddy(a1)

self.file\_layout.addWidget(lab1)

self.file\_layout.addWidget(a1)

def select\_file(self):

filename = QFileDialog.getOpenFileName(self, "Select File", os.getcwd(), "CSV Files(\*.csv);;ALL Files(\*)")

self.filename = filename[0]

if self.filename:

self.show\_in\_group3()

def show\_in\_group3(self):

df = pd.read\_csv(self.filename, encoding='gbk')

value = df.values

p = value[:, 0]

T = value[:, 1]

self.p = p

self.T = T

self.ans = df.columns[2]

self.nans = [Algorithm.other(i, j, self.ans) for i, j in zip(p, T)]

df[self.ans] = self.nans

for line in range(df.shape[0]):

for row in range(df.shape[1]):

data = QTableWidgetItem(f"{df.iat[line, row]:.5e}")

self.tablewidget.setItem(line, row, data)

self.tablewidget.setColumnCount(row + 1)

self.tablewidget.setRowCount(line + 1)

columns = list(df.columns)

columns.append('')

self.tablewidget.setHorizontalHeaderLabels(columns)

self.tablewidget.setFixedSize(self.bottom\_left\_group.size())

self.tablewidget.setAlternatingRowColors(True)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

cal = Qt\_cal()

cal.show()

sys.exit(app.exec\_())

home.py

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

# Form implementation generated from reading ui file 'home.ui'

#

# Created by: PyQt5 UI code generator 5.15.9

#

# WARNING: Any manual changes made to this file will be lost when pyuic5 is

# run again. Do not edit this file unless you know what you are doing.

import sys

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtWidgets import QApplication, QMainWindow

class Ui\_Wvpm(object):

def setupUi(self, Wvpm):

window\_size = (2500, 1500)

Wvpm.setObjectName("Wvpm")

Wvpm.setFixedSize(\*window\_size)

self.label = QtWidgets.QLabel(Wvpm)

self.label.setGeometry(QtCore.QRect(0, 0, \*window\_size))

self.label.setText("")

self.label.setPixmap(QtGui.QPixmap(":/stream/image/steam.jpg"))

self.label.setScaledContents(True)

self.label.setWordWrap(False)

self.label.setObjectName("label")

self.label\_2 = QtWidgets.QLabel(Wvpm)

self.label\_2.setGeometry(QtCore.QRect(20, 0, \*(i/10 for i in window\_size)))

font = QtGui.QFont()

font.setPointSize(15)

font.setBold(True)

font.setWeight(75)

self.label\_2.setFont(font)

self.label\_2.setScaledContents(True)

self.label\_2.setObjectName("label\_2")

self.pushButton = QtWidgets.QPushButton(Wvpm)

self.pushButton.setGeometry(QtCore.QRect(360, 1250, 300, 100))

font = QtGui.QFont()

font.setBold(True)

font.setPointSize(20)

font.setWeight(75)

self.pushButton.setFont(font)

self.pushButton.setStyleSheet("QPushButton\n"

"{\n"

" background: rgb(250, 250, 250);\n"

" border: 2px solid rgb(200, 200, 200)\n"

"}\n"

"")

icon = QtGui.QIcon()

icon.addPixmap(QtGui.QPixmap(":/stream/image/title.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

self.pushButton.setIcon(icon)

self.pushButton.setIconSize(QtCore.QSize(50, 50))

self.pushButton.setObjectName("pushButton")

self.textEdit = QtWidgets.QTextBrowser(Wvpm)

self.textEdit.setGeometry(QtCore.QRect(1700, 807, 500, 250))

font = QtGui.QFont()

font.setFamily("Roman")

font.setPointSize(25)

font.setBold(True)

font.setWeight(75)

self.textEdit.setFont(font)

self.textEdit.setAutoFillBackground(True)

self.textEdit.setStyleSheet("border: 2px solid rgb(255, 255, 255)")

self.textEdit.setObjectName("textEdit")

self.retranslateUi(Wvpm)

QtCore.QMetaObject.connectSlotsByName(Wvpm)

def retranslateUi(self, Wvpm):

\_translate = QtCore.QCoreApplication.translate

Wvpm.setWindowTitle(\_translate("Wvpm", "Wvpm"))

Icon = Wvpm.windowIcon()

Icon.addPixmap(QtGui.QPixmap(":/stream/image/title.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)

Wvpm.setWindowIcon(Icon)

self.label\_2.setText(\_translate("Form", "author: Dan"))

self.pushButton.setText(\_translate("Form", "Start"))

self.textEdit.setHtml(\_translate("Form",

"<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0//EN\" "

"\"http://www.w3.org/TR/REC-html40/strict.dtd\">\n "

"<html><head><meta name=\"qrichtext\" content=\"1\" /><style "

"type=\"text/css\">\n "

"p, li { white-space: pre-wrap; }\n"

"</style></head><body style=\" font-family:\'Roman\'; font-size:15pt; "

"font-weight:600; font-style:normal;\">\n "

"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; "

"margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\" "

"font-family:\'SimSun\'; font-size:9pt;\">Wvpm</span><span style=\" "

"font-family:\'SimSun\'; font-size:9pt; font-weight:400;\"> is a small "

"software for calculating the enthalpy, entropy, specific volume of water "

"vapor related states</span></p>\n "

"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; "

"margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\" "

"font-family:\'SimSun\'; font-size:9pt; font-weight:400; font-style:italic; "

"text-decoration: underline;\">Date of creation: 2023/4/12</span></p>\n "

"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; "

"margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><span style=\" "

"font-family:\'SimSun\'; font-size:9pt; font-weight:400;\">Reference: "

"</span><a href=\"http://www.iapws.org/relguide/IAPWS-95.html\"><span "

"style=\" font-family:\'SimSun\'; font-size:9pt; font-weight:400; "

"text-decoration: underline; color:#0000ff;\">IAPWS-95,</span></a><a "

"href=\"http://www.coolprop.org/v4/\"><span style=\" font-family:\'SimSun\'; "

"font-size:9pt; font-weight:400; text-decoration: underline; "

"color:#0000ff;\">CoolProp</span></a></p></body></html>"))

import image\_rc

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

cal = Ui\_Wvpm()

windows = QMainWindow()

cal.setupUi(windows)

windows.show()

sys.exit(app.exec\_())

Plot.py

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

# @Time : 2023/4/18 20:10

# @Author : DanYang

# @File : Plot.py

# @Software : PyCharm

import numpy as np

from mpl\_toolkits import mplot3d

import matplotlib.pyplot as plt

from matplotlib.style.core import use

from Algorithm import PropsSI

plt.rcParams['font.sans-serif'] = ['SimHei']

plt.rcParams['axes.unicode\_minus'] = False

plt.rcParams['xtick.direction'] = 'in'

plt.rcParams['ytick.direction'] = 'in'

def set\_style(style):

use(style)

def plot\_data(p, T, goal, goal\_name, if\_plot\_p=False, if\_plot\_T=False, if\_plot\_all=False, method='classic'):

set\_style(method)

fig = plt.figure()

ax = plt.axes(projection='3d')

ax.scatter3D(p, T, goal, color='#54278f', label='Raw data')

plt.title(f'p-T-{goal\_name}')

ax.set\_xlabel('p/Mpa')

ax.set\_ylabel('T/K')

if goal\_name == 'v':

an = '$m^3.kg$'

elif goal\_name == 'h':

an = '$kJ.kg^{-1}$'

elif goal\_name == 's':

an = '$kJ.kg^{-1}$'

ax.set\_zlabel(f'{goal\_name}/{an}')

if if\_plot\_all:

x, y = np.meshgrid(np.linspace(min(p), max(p), 100), np.linspace(min(T), max(T), 100))

x = np.ravel(x)

y = np.ravel(y)

z = np.array([PropsSI(goal\_name.upper(), 'P', i\*10\*\*6, 'T', j, 'Water') for i, j in zip(x, y)])

x = x.reshape((100, 100))

y = y.reshape((100, 100))

z = z.reshape((100, 100))

ax.plot\_surface(x, y, z, color='#efedf5')

plt.legend()

plt.show()

return

if if\_plot\_p:

x, y = np.meshgrid(np.linspace(min(T), max(T), 100), np.linspace(min(goal), max(goal), 100))

x = np.ravel(x)

y = np.ravel(y)

z = np.array([PropsSI('P', goal\_name.upper(), i, 'T', j, 'Water') / 1000000 for i, j in zip(y, x)])

x = x.reshape((100, 100))

y = y.reshape((100, 100))

z = z.reshape((100, 100))

ax.plot\_surface(z, x, y, color='#6baed6')

if if\_plot\_T:

x, y = np.meshgrid(np.linspace(min(p), max(p), 100), np.linspace(min(goal), max(goal), 100))

x = np.ravel(x)

y = np.ravel(y)

z = np.array([PropsSI('T', goal\_name.upper(), i, 'P', j\*10\*\*6, 'Water') for i, j in zip(y, x)])

x = x.reshape((100, 100))

y = y.reshape((100, 100))

z = z.reshape((100, 100))

ax.plot\_surface(x, z, y, color='#ef3b2c')

plt.legend()

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

plot\_data(np.linspace(3, 80, 10), np.linspace(300, 500, 10), np.linspace(300, 600, 10), 's', if\_plot\_all=True, method='seaborn')

main.py

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

# @Time : 2023/4/13 2:51

# @Author : DanYang

# @File : main.py

# @Software : PyCharm

import sys

from home import Ui\_Wvpm

from calpage import Qt\_cal

from PyQt5.QtWidgets import QApplication, QMainWindow

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ui = Ui\_Wvpm()

windows = QMainWindow()

ui.setupUi(windows)

windows.show()

cal = Qt\_cal(parent=windows)

ui.pushButton.clicked.connect(cal.show)

sys.exit(app.exec\_())