# 数值计算作业源代码

S322517454 张云涛

哈尔滨工程大学 烟台研究院

# 作业1插值

import numpy as np

from scipy import interpolate

# 导入数据

years = np.arange(1960, 2021, 10)

populations = np.array([180671, 205052, 227225, 249623, 282162, 309327, 329484])

def \_poly\_newton\_coefficient(x, y):

m = len(x)

x = np.copy(x)

a = np.copy(y)

for k in range(1, m):

a[k:m] = (a[k:m] - a[k-1])/(x[k:m] - x[k - 1])

return a

def newton\_polynomial(x\_data, y\_data, x):

'''Newton插值'''

a = \_poly\_newton\_coefficient(x\_data, y\_data)

n = len(x\_data) - 1 # Degree of polynomial

p = a[n]

for k in range(1, n + 1):

p = a[n - k] + (x - x\_data[n - k])\*p

return p

# Lagrange插值

f\_l = interpolate.lagrange(years, populations)

# 三次样条插值

f\_sp = interpolate.CubicSpline(years, populations)

test\_years = [1950, 2005, 2030]

p\_1950 = 151326

p\_2005 = 295516

print("Lagrange插值结果")

for year in test\_years:

print("\t\t{}年: {}".format(year, f\_l(year)))

er\_l1 = (f\_l(test\_years[0])-p\_1950)/p\_1950

er\_l2 = (f\_l(test\_years[1])-p\_2005)/p\_2005

print("\t{}年相对误差: {}".format(test\_years[0], er\_l1))

print("\t{}年相对误差: {}".format(test\_years[1], er\_l2))

print("Newton插值结果")

for year in test\_years:

print("\t\t{}年: {}".format(year, newton\_polynomial(years, populations, year)))

er\_n1 = (newton\_polynomial(years, populations, test\_years[0])-p\_1950)/p\_1950

er\_n2 = (newton\_polynomial(years, populations, test\_years[1])-p\_2005)/p\_2005

print("\t{}年相对误差: {}".format(test\_years[0], er\_n1))

print("\t{}年相对误差: {}".format(test\_years[1], er\_n2))

print("三次样条插值结果")

for year in test\_years:

print("\t\t{}年: {}".format(year, f\_sp(year)))

er\_sp1 = (f\_sp(test\_years[0])-p\_1950)/p\_1950

er\_sp2 = (f\_sp(test\_years[0])-p\_2005)/p\_2005

print("\t{}年相对误差: {}".format(test\_years[0], er\_sp1))

print("\t{}年相对误差: {}".format(test\_years[1], er\_sp2))

# **作业2拟合**

import warnings

warnings.filterwarnings("ignore",category=DeprecationWarning)

import math

from symbol import parameters

from typing\_extensions import dataclass\_transform

from unittest import result

import pandas

import pandas as pd

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

import matplotlib.image as mpimg

from scipy import optimize

df = pd.read\_excel("../相关资源/数据/题目2拟合.xlsx", sheet\_name="Sheet1")

data\_w = df["w"]

data\_r = df["R"]

def f\_ab(params, x, y):

a, b = params

residual = y - (b + a\*x)

return residual

def f\_abc(params, x, y):

a, b, c = params

residual = y - (b + a\*x + c\*x\*\*2)

return residual

data\_lnw, data\_lnr = [], []

for i in range(len(data\_w)):

data\_lnw.append(math.log(data\_w[i]))

data\_lnr.append(math.log(data\_r[i]))

data\_lnw = np.array(data\_lnw)

data\_lnr = np.array(data\_lnr)

# 拟合lnR = lnb + alnw，确定a，b

params\_ab = [0, 0]

result1 = optimize.leastsq(f\_ab, params\_ab, (data\_lnw, data\_lnr))

a1, lnb1 = result1[0]

b1 = math.exp(lnb1)

# 计算平方误差

err\_sq1 = 0

for i in range(len(data\_lnr)):

err\_sq1 += (data\_r[i]-math.exp(lnb1+a1\*data\_lnw[i]))\*\*2

# err\_sq1 += (data\_lnr[i]-(lnb1+a1\*data\_lnw[i]))\*\*2

print("lnR = lnb + alnw")

print("参 数: a={},b={}".format(a1, b1))

print("拟合方程: lnR = ln({}) + ({})lnw".format(b1, a1))

print("平方误差: {}\n".format(err\_sq1))

# 拟合lnR = lnb + alnw + c(lnw)^2，确定a，b，c

params\_abc = [0, 0, 0]

result = optimize.leastsq(f\_abc, params\_abc, (data\_lnw, data\_lnr))

a2, lnb2, c2 = result[0]

b2 = math.exp(lnb2)

# 计算平方误差

err\_sq2 = 0

for i in range(len(data\_lnr)):

err\_sq2 += (data\_r[i]-math.exp(lnb2+a2\*data\_lnw[i]+c2\*data\_lnw[i]\*\*2))\*\*2

# err\_sq2 += (data\_lnr[i]-(lnb1+a1\*data\_lnw[i]))\*\*2

print("lnR = lnb + alnw + c(lnw)^2")

print("参 数: a={},b={},c={}".format(a2, b2, c2))

print("拟合方程: lnR = ln({}) + ({})lnw + ({})(lnw)^2".format(b2, a2, c2))

print("平方误差: {}\n".format(err\_sq2))

# **作业3非线形方程求根与数值积分综合**

import math

import numpy as np

from scipy.integrate import romberg

err = math.inf

err\_limits = 1e-10

print("设置误差限为{}".format(err\_limits))

print("Romberg求积")

xi = 0.5

round = 0

while err>err\_limits:

# 被积函数

g = lambda x: 1/math.sqrt(2\*math.pi)\*math.exp(-x\*\*2/2)

# Romberg求积

fri = romberg(g, 0, xi, show=False)

f = fri - 0.45

# Newton迭代中的f'(x)

df = 1/math.sqrt(2\*math.pi)\*math.exp(-xi\*\*2/2)

# Newton迭代

xii = xi - f/df

# 计算两次求得得根之间的误差

err = abs(xii - xi)

xi = xii

round += 1

print("第{}次Newton迭代后，Romberg积分值为{}，\t根为{}".format(round, fri, xi))

print("\nGauss-Lgendre求积")

xi = 0.5

round = 0

err = math.inf

while err>err\_limits:

# 被积函数

gl = lambda x: xi\*math.exp(-xi\*xi\*(x+1)\*\*2/8)/(math.sqrt(2\*math.pi)\*2)

# Gauss-Legendre求积

gli = 5/9\*gl(-math.sqrt(3/5))+8/9\*gl(0)+5/9\*gl(math.sqrt(3/5))

f = gli - 0.45

# Newton迭代中的f'(x)

df = 1/math.sqrt(2\*math.pi)\*math.exp(-xi\*\*2/2)

# Newton迭代

xii = xi - f/df

# 计算两次求得得根之间的误差

err = abs(xii - xi)

xi = xii

round += 1

print("第{}次Newton迭代后，Gauss-Legendre积分值为{}，\t根为{}".format(round, gli, xi))

# **作业4常微分方程初值问题**

import numpy as np

from scipy import integrate

import matplotlib.pyplot as plt

k = 6.22e-19

n1 = 2e3

n2 = 2e3

n3 = 3e3

def f(t,x):

return k\*(n1-x/2)\*\*2\*(n2-x/2)\*\*2\*(n3-3\*x/4)\*\*3

result=integrate.solve\_ivp(f,(0,0.2),[0], method='RK45', dense\_output=True)

print(result)

print("解得在0.2s后将形成{}单位的氢氧化钾".format(result.sol(result.t)[0][-1]))

# t=np.linspace(0,0.2,101)

# plt.plot(t,result.sol(t)[0],label='numerical solution')

plt.plot(result.t,result.sol(result.t)[0],label='numerical solution')

plt.grid()

plt.legend()

plt.xlabel('t')

plt.ylabel('y(t)')

plt.show()

# **作业5思政案例-插值算法在扫描软件中的应用**

因为是公司项目，整个项目无法完全开源，此处粘贴部分相关代码。

//扫描相关

int LineManager::ScanThread() {

float r = 1.0;

//启动时清空存放的结果

while(!scan\_res\_.empty())scan\_res\_.pop();

//启动编码器

if (scan\_flag\_ == true) {

encoder\_->StartRot();

}

while (scan\_flag\_ == true) {

auto \_frame\_f = cam\_group\_->Capture();

//判断是否存在相机连接断开

if(\_frame\_f.size() < 10){

controller\_->error1\_camerr();

return 1;

}else{

//

float \_y\_pos = encoder\_->GetPosition();

if (scan\_res\_.size() > 0){

auto \_frame0 = scan\_res\_.back();

auto \_frame1 = std::make\_pair(\_y\_pos, \_frame\_f);

auto \_res = tool::InterpolateYXZ(1, \_frame0, \_frame1);

for(size\_t i=1; i<\_res.size(); i++) {

scan\_res\_.push(\_res[i]);

}

}else{ // 第一帧

scan\_res\_.push(std::make\_pair(\_y\_pos, \_frame\_f));

}

while (scan\_res\_.size() > q\_scan\_size\_) {

scan\_res\_.pop();

cout << "POP" << endl;

}

int scan\_duration\_ = 5;

if (scan\_flag\_){

std::this\_thread::sleep\_for(std::chrono::milliseconds(scan\_duration\_));

}else{

encoder\_->StopRot();

}

cout << endl;

}

}

return 0;

}

//插值相关

inline std::vector<std::pair<float, float> > InterpolateXZ(float inter\_val, std::vector<std::pair<float, float> > src\_data) {

std::vector<std::pair<float, float> > inter\_cap = {src\_data[0]}; // 先插入第一点

std::vector<double> \_frame\_x, \_frame\_z;

\_frame\_x.clear();

\_frame\_z.clear();

auto last\_x = src\_data[0].first;

for(size\_t i=1; i<src\_data.size(); i++) {

if (src\_data[i].first > last\_x) {

\_frame\_x.push\_back(src\_data[i].first);

\_frame\_z.push\_back(src\_data[i].second);

last\_x = \_frame\_x.back();

}

}

// 三次样条插值

tk::spline s;

s.set\_points(\_frame\_x, \_frame\_z);

float p\_0 = \_frame\_x[0];

double p\_end = \_frame\_x[\_frame\_x.size()-1];

for(auto p=p\_0+inter\_val; p<p\_end; p+=inter\_val) {

float p\_z = s(p);

auto \_point = std::make\_pair(p, p\_z);

inter\_cap.push\_back(\_point);

}

return inter\_cap;

}

typedef std::vector<std::pair<float, float> > VectorXZ;

inline std::vector<std::pair<float, VectorXZ> > InterpolateYXZ(float interval, std::pair<float, VectorXZ> frame0, std::pair<float, VectorXZ> frame1) {

std::vector<std::pair<float, VectorXZ> > inter\_res;

float \_inter\_y = frame0.first;

while (\_inter\_y <= frame1.first) {

float a = \_inter\_y; // 返回第一帧

VectorXZ \_frame;

for(size\_t i=0; i<frame1.second.size(); i++) {

auto \_point = (frame1.second)[i];

auto \_prev\_point = (frame0.second)[i];

// float b = \_point.second;

float b = (\_point.second-\_prev\_point.second)/(frame1.first-frame0.first)\*(a-frame0.first) + \_prev\_point.second;

\_frame.push\_back(std::make\_pair(\_point.first, b)); // x-z

}

\_inter\_y += interval;

inter\_res.emplace\_back(a, \_frame); // y-(x-z)

}

return inter\_res;

}

张云涛 S322517454

2022年11月1日星期二