数值计算大作业

第九组

小组成员:

莫书琪 17318086

谢应龙 19335227

张诺言 19335265

杨沅旭 19335247

A1

思路

(a) 对拉格朗日插值法中的每一个插值点，先代入x，求得插值基函数的值，再乘以该插值点对应的函数值，把所有插值点通过这样方式得到的结果累加起来即为最终结果。对牛顿插值法，先利用给定插值点建立均差表，然后根据均差表的数值进行运算。

(b) 关键是对反函数实现拉格朗日插值和牛顿插值，即f(x)为插值节点，x为插值节点的函数值。

代码

Python

```python
import numpy as np
import time

def lagrange(x):
    y_lr = 0
    for i in range(n):
        denominator = 1.0
        numerator = 1.0
        for j in range(n):
            if i == j:
                continue
            denominator *= x_sample[i] - x_sample[j]
            numerator *= x - x_sample[j]
        base = numerator / denominator
        y_lr += base * y_sample[i]
```

```python
        y_lr += base * y_sample[i]
    return y_lr

def newton(x):
    y_newton = diff[1][0]
    delta_x = x - x_sample[0]
    for i in range(1,n):
        y_newton += diff[i+1][i] * delta_x
        delta_x *= x - x_sample[i]
    return y_newton

start1 = time.time()
x_sample = np.array([0.4,0.55,0.65,0.8,0.9,1.05])
y_sample = np.array([0.41075,0.57825,0.69675,0.88811,1.02652,1.25382])
n = x_sample.shape[0]
# 均差表
diff = np.zeros((n+1,n))
for i in range(n):
    diff[0][i] = x_sample[i]
    diff[1][i] = y_sample[i]
for diff_index in range(1,n):
    for i in range(diff_index,n):
        diff[diff_index+1][i] = (diff[diff_index][i]-diff[diff_index][i-1])/(diff[0][i]-diff[0][i-diff_index])
end1 = time.time()

print("(a)")
print("拉格朗日插值法的结果：")
print("f(0.42) = {}".format(lagrange(0.42)))
print("f(0.596) = {}".format(lagrange(0.596)))
print("f(1.0) = {}".format(lagrange(1.0)))
print("牛顿插值法的结果：")
print("f(0.42) = {}".format(newton(0.42)))
print("f(0.596) = {}".format(newton(0.596)))
print("f(1.0) = {}".format(newton(1.0)))

start2 = time.time()
y_sample = np.array([0.4,0.55,0.65,0.8,0.9,1.05])
x_sample = np.array([0.41075,0.57825,0.69675,0.88811,1.02652,1.25382])
n = x_sample.shape[0]
# 均差表
diff = np.zeros((n+1,n))
for i in range(n):
    diff[0][i] = x_sample[i]
    diff[1][i] = y_sample[i]
```

```python
for diff_index in range(1,n):
    for i in range(diff_index,n):
        diff[diff_index+1][i] = (diff[diff_index][i]-diff[diff_index][i-1])/(diff[0][i]-diff[0][i-diff_index])
end2 = time.time()

print("(b)")
print("拉格朗日插值法的结果：")
print("f(z1) = 0.5, z1 = {}".format(lagrange(0.5)))
print("f(z2) = 0.75, z2 = {}".format(lagrange(0.75)))
print("f(z3) = 1.0, z3 = {}".format(lagrange(1.0)))
print("牛顿插值法的结果：")
print("f(z1) = 0.5, z1 = {}".format(newton(0.5)))
print("f(z2) = 0.75, z2 = {}".format(newton(0.75)))
print("f(z3) = 1.0, z3 = {}".format(newton(1.0)))

print("\n程序执行时间为：{}".format(end2-start2+end1-start1))
```

实验结果

国产计算平台（Python 3.7.4）

```
[root@host-11-0-0-66 ~]# python3 A1.py
(a)
拉格朗日插值法的结果：
f(0.42) = 0.4325334865920001
f(0.596) = 0.6319629080415248
f(1.0) = 1.175156953846154
牛顿插值法的结果：
f(0.42) = 0.43253348659199997
f(0.596) = 0.6319629080415247
f(1.0) = 1.1751569538461537
(b)
拉格朗日插值法的结果：
f(z1) = 0.5, z1 = 0.4810909591385529
f(z2) = 0.75, z2 = 0.6931609737072644
f(z3) = 1.0, z3 = 0.8813643472827534
牛顿插值法的结果：
f(z1) = 0.5, z1 = 0.481090959138553
f(z2) = 0.75, z2 = 0.6931609737072645
f(z3) = 1.0, z3 = 0.8813643472827535

程序执行时间(不考虑I/O)为：0.00023412704467773438

程序执行时间(考虑I/O)为：0.0010364055633544922
[root@host-11-0-0-66 ~]#
```

本地平台（Python 3.7.6）

```
In [1]: runfile('C:/Users/Administrator/Desktop/A1.py', wdir='C:/Users/Administrator/
Desktop')
(a)
拉格朗日插值法的结果:
f(0.42) = 0.4325334865920001
f(0.596) = 0.6319629080415248
f(1.0) = 1.175156953846154
牛顿插值法的结果:
f(0.42) = 0.43253348659199997
f(0.596) = 0.6319629080415247
f(1.0) = 1.1751569538461537
(b)
拉格朗日插值法的结果:
f(z1) = 0.5, z1 = 0.4810909591385529
f(z2) = 0.75, z2 = 0.6931609737072644
f(z3) = 1.0, z3 = 0.8813643472827534
牛顿插值法的结果:
f(z1) = 0.5, z1 = 0.481090959138553
f(z2) = 0.75, z2 = 0.6931609737072645
f(z3) = 1.0, z3 = 0.8813643472827535

程序执行时间(不考虑I/O)为:0.0

程序执行时间(考虑I/O)为:0.002984762191772461
```

B2

思路

(a) 计算Newton-Cotes求积公式首先需要得到Cotes公式的系数，在本次作业中我们按照课本103页公式(2.2)计算得到系数，然后根据公式(2.1)计算得到最终结果。

(b) 计算Guass-Legendre求积公式需要得到求积节点和求积系数。在本次作业中，为了减少程序运算量，我们首先根据课本61页的递推公式(2.9)列出不同n值下的勒让德多项式，然后求得对应n值下的零点作为求积节点。得到求积节点后，我们把这些求积节点作为拉格朗日插值基函数的插值节点，然后利用课本118页公式(6.6)计算求积系数。最后，我们利用课本122页公式(6.13)计算得到最终结果。

(c) 直接调用scipy.integrate.quad().

代码

Python

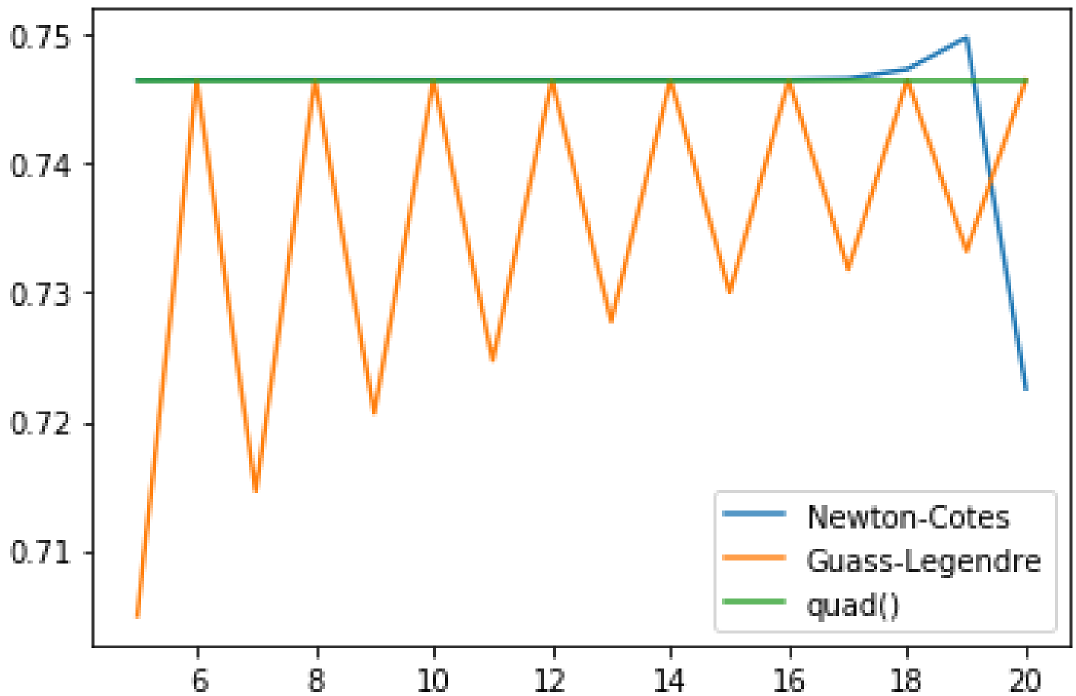
```python
import numpy as np
from scipy import integrate
import time

start = time.time()
```

```

def f(x):
    return np.cos(x) / (1 + np.sin(x)**3)

a = 0
b = 1
res_nc = []
res_gl = []

for n in range(5,21):
    h = (b - a) / n
    coefficient = np.zeros(n+1)
    for k in range(0,n+1):
        coefficient[k] = n * np.math.factorial(k) * np.math.factorial(n-k)
        if (n-k) % 2 != 0:
            coefficient[k] = -coefficient[k]
        order = []
        for j in range(n+1):
            if j != k:
                order.append(j)
        poly = np.poly1d(order, r=True, variable=["x"])
        poly_afterintegral = np.array(np.polyint(poly))
        result = 0
        for j in range(n+1):
            result += poly_afterintegral[j] * n ** (n-j+1)
        coefficient[k] = result / coefficient[k]
    ans_nc = 0
    for k in range(n+1):
        ans_nc += coefficient[k] * f(a+k*h)
    ans_nc *= b - a
    res_nc.append(ans_nc)

T = 20
L = []
L.append(np.poly1d([1], r=False, variable=["x"]))
L.append(np.poly1d([1,0], r=False, variable=["x"]))
for i in range(1,T+1):
    L.append(np.polysub((2*i+1)/(i+1)*np.polymul(L[1],L[i]),i/(i+1)*L[i-1]))

for n in range(5,21):
    Ak = []
    xk = np.roots(L[n+1])
    xk = np.sort(xk)
    for i in range(n+1):
        denominator = 1
```

```
        numerator = []
        for j in range(n+1):
            if j != i:
                denominator *= xk[i] -xk[j]
                numerator.append(xk[j])
        poly_numerator = np.poly1d(numerator, r=True, variable=["x"])
        l = poly_numerator / denominator
        l_afterintegral = np.array(np.polyint(l))
        ans = 0
        for j in range(n+2):
            if j % 2 != 0:
                continue
            else:
                ans += 2 * l_afterintegral[j]
        Ak.append(abs(ans))
    ans_gl = 0
    for i in range(n+1):
        ans_gl += Ak[i]*f((b-a)/2*xk[i]+(a+b)/2)
    ans_gl *= (b-a)/2
    res_gl.append(ans_gl)

result, err = integrate.quad(f,a,b)
n = []
quad = []
for i in range(5,21):
    n.append(i)
    quad.append(result)

end1 = time.time()

print("(a) Newton-Cotes")
for i in range(16):
    print("n = {}, the result is {}".format(i+5, res_nc[i]))
print("(b) Gauss-Legendre")
for i in range(16):
    print("n = {}, the result is {}".format(i+5, res_gl[i]))
print("(c) quad()")
print("Using quad(), the result is {}".format(result))

end2 = time.time()
print("\n程序执行时间(不考虑I/O)为：{}".format(end1-start))
print("\n程序执行时间(考虑I/O)为：{}".format(end2-start))
```

实验结果

由于国产计算平台无法安装matplotlib库，所以d问的图是在本地电脑上借助matplotlib库画的，这一部分代码在画完图后就删掉了。



国产计算平台（Python 3.7.4）

```
[root@host-11-0-0-66 ~]# python3 B2.py
(a) Newton-Cotes
n = 5, the result is 0.7464820858159198
n = 6, the result is 0.7465212011100144
n = 7, the result is 0.7465232713527835
n = 8, the result is 0.7465267538068485
n = 9, the result is 0.7465261194457199
n = 10, the result is 0.746524942944026
n = 11, the result is 0.7465250571449096
n = 12, the result is 0.7465252740859586
n = 13, the result is 0.7465252714132099
n = 14, the result is 0.7465251102703923
n = 15, the result is 0.7465256833190219
n = 16, the result is 0.7465314148792719
n = 17, the result is 0.7466131870253012
n = 18, the result is 0.7473382371608072
n = 19, the result is 0.7497959618966145
n = 20, the result is 0.7225684376292152
(b) Gauss-Legendre
n = 5, the result is 0.7049245987720739
n = 6, the result is 0.7465252232794464
n = 7, the result is 0.7145983526109826
n = 8, the result is 0.7465252358083478
n = 9, the result is 0.7206373335990033
n = 10, the result is 0.7465252359011189
n = 11, the result is 0.7247599795287643
n = 12, the result is 0.7465252359019345
n = 13, the result is 0.7277519120596448
n = 14, the result is 0.7465252359030294
n = 15, the result is 0.7300216227684372
n = 16, the result is 0.7465252358739494
n = 17, the result is 0.7318022011038251
n = 18, the result is 0.7465252357336208
n = 19, the result is 0.7332362629645627
n = 20, the result is 0.7465252353229046
(c) quad()
Using quad(), the result is 0.7465252359016145

程序执行时间(不考虑I/O)为：0.1721034049987793

程序执行时间(考虑I/O)为：0.17287826538085938
[root@host-11-0-0-66 ~]#
```

本地平台（Python 3.7.6）

```
In [5]: runfile('C:/Users/Administrator/Desktop/B2.py', wdir='C:/Users/Administrator/Desktop')
(a) Newton-Cotes
n = 5, the result is 0.7464820858159198
n = 6, the result is 0.7465212011100144
n = 7, the result is 0.7465232713527835
n = 8, the result is 0.7465267538068485
n = 9, the result is 0.7465261194457199
n = 10, the result is 0.746524942944026
n = 11, the result is 0.7465250571449096
n = 12, the result is 0.7465252740859586
n = 13, the result is 0.7465252714132098
n = 14, the result is 0.7465251102703925
n = 15, the result is 0.7465256833190219
n = 16, the result is 0.7465314148792719
n = 17, the result is 0.746613187025301
n = 18, the result is 0.7473382371608069
n = 19, the result is 0.7497959618966145
n = 20, the result is 0.7225684376292152
(b) Gauss-Legendre
n = 5, the result is 0.7049245987720748
n = 6, the result is 0.746525223279446
n = 7, the result is 0.7145983526109774
n = 8, the result is 0.7465252358083491
n = 9, the result is 0.7206373335989532
n = 10, the result is 0.7465252359011026
n = 11, the result is 0.7247599795287272
n = 12, the result is 0.7465252359020146
n = 13, the result is 0.7277519120621587
n = 14, the result is 0.7465252359024408
n = 15, the result is 0.7300216227278749
n = 16, the result is 0.7465252359202416
n = 17, the result is 0.7318022012439689
n = 18, the result is 0.7465252360234229
n = 19, the result is 0.733236260294374
n = 20, the result is 0.7465252361542485
(c) quad()
Using quad(), the result is 0.7465252359016143

程序执行时间(不考虑I/O)为:0.06096482276916504

程序执行时间(考虑I/O)为:0.06296467781066895
```

C2

思路

(a) 将矩阵A进行LU分解，首先高斯消元可以得到U，先选定主元，然后在高斯消元的过程中把每一次消元过程中的乘数，存储下来并记录该乘数的位置，就是L下三角矩阵中非主对角元素的值，再通过一个循环将主对角元素赋1即可得到L。

(b) 将矩阵A进行LU分解后，要解线性方程组Ax=b，只需要进行两步，第一步是通过解LY=b，从而求解出向量Y，第二步通过求解UX=Y从而求出解X，由于L为下三角矩阵，U为上三角矩阵，求解只需要

进行回代计算即可。

代码

Python

```python
import numpy as np
import time

start = time.time()

def LUDec(A):
    n = len(A)
    L = np.zeros(shape=(n,n))
    U = np.zeros(shape=(n,n))
    for base in range(n-1):
        for i in range(base+1,n):
            L[i,base]=A[i,base]/A[base,base]
            A[i]=A[i]-L[i,base]*A[base]
    for i in range(n):
        L[i,i]=1
    U=np.array(A)
    return L,U

def solve(L,U,b):
    rows = len(b)
    y = np.zeros(rows)
    y[0] = b[0]/L[0,0]
    for k in range(1,rows):
        y[k] = (b[k] - np.sum(L[k,:k]*y[:k]))/L[k,k]
    x = np.zeros(rows)
    k = rows-1
    x[k] = y[k]/U[k,k]
    for k in range(rows-2,-1,-1):
        x[k] = (y[k] - np.sum(x[k+1:]*U[k,k+1:]))/U[k,k]
    return x

A = np.array( [[ 20.,   2.,   3.,   0.],
               [  1.,   8.,   1.,   1.],
               [  2.,  -3.,  15.,   0.],
               [  1.,   0.,   0.,   1.]],dtype='float')
b1 = np.array([[ 1., 0., 0., 0.]],dtype='float').T
b2 = np.array([[ 0., 1., 0., 0.]],dtype='float').T
b3 = np.array([[ 0., 0., 1., 0.]],dtype='float').T
```

```python
b4 = np.array([[ 0., 0., 0., 1.]],dtype='float').T
inverse = np.linalg.inv(A)
La,Ua = LUDec(A)

end1 = time.time()

print("(a)")
print("LU分解得到的L为:")
print(La)
print("LU分解得到的U为:")
print(Ua)
print("(b)")
print("解得X1,X2,X3,X4的值分别为:")
x1 = solve(La,Ua,b1)
print("X1=",x1)
x2 = solve(La,Ua,b2)
print("X2=",x2)
x3 = solve(La,Ua,b3)
print("X3=",x3)
x4 = solve(La,Ua,b4)
print("X4=",x4)
print("矩阵A的逆为:")
print(inverse)

end2 = time.time()

print("\n程序执行时间(不考虑I/O)为：{}".format(end1-start))
print("\n程序执行时间(考虑I/O)为：{}".format(end2-start))
```

实验结果

国产计算平台（Python 3.7.4）

```
[root@host-11-0-0-66 ~]# python3 C2.py
(a)
LU分解得到的L为:
[[ 1.          0.          0.          0.        ]
 [ 0.05        1.          0.          0.        ]
 [ 0.1        -0.40506329  1.          0.        ]
 [ 0.05       -0.01265823 -0.00925536  1.        ]]
LU分解得到的U为:
[[20.          2.          3.          0.        ]
 [ 0.          7.9         0.85        1.        ]
 [ 0.          0.         15.0443038   0.40506329]
 [ 0.          0.          0.          1.01640724]]
(b)
解得X1,X2,X3,X4的值分别为:
X1= [ 0.0509106   0.00082781 -0.00662252 -0.0509106 ]
X2= [-0.01614238  0.12168874  0.02649007  0.01614238]
X3= [-0.00910596 -0.00827815  0.06622517  0.00910596]
X4= [ 0.01614238 -0.12168874 -0.02649007  0.98385762]
矩阵A的逆为:
[[ 5.09105960e-02 -1.61423841e-02 -9.10596026e-03  1.61423841e-02]
 [ 8.27814570e-04  1.21688742e-01 -8.27814570e-03 -1.21688742e-01]
 [-6.62251656e-03  2.64900662e-02  6.62251656e-02 -2.64900662e-02]
 [-5.09105960e-02  1.61423841e-02  9.10596026e-03  9.83857616e-01]]

程序执行时间(不考虑I/O)为: 0.00027370452880859375

程序执行时间(考虑I/O)为: 0.003981828689575195
[root@host-11-0-0-66 ~]#
```

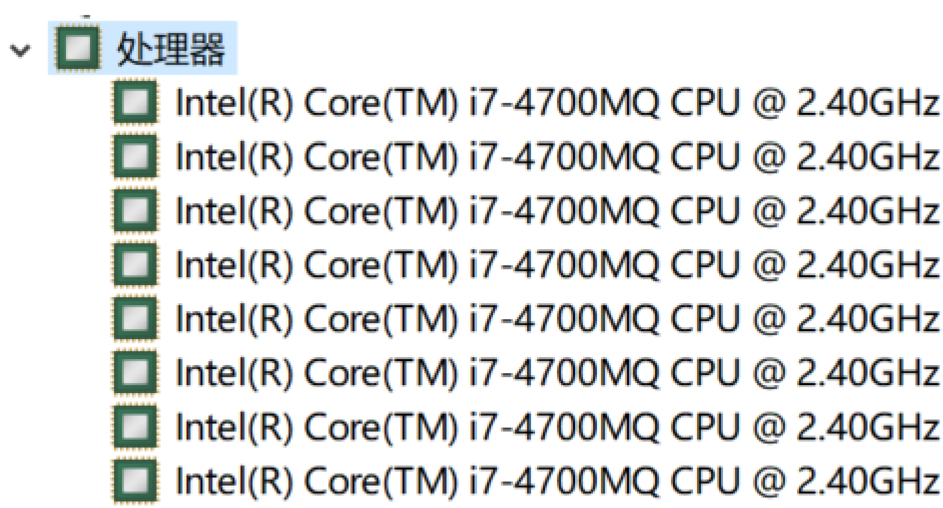
本地平台（Python 3.7.6）

```
In [8]: runfile('C:/Users/Administrator/Desktop/C2.py', wdir='C:/Users/Administrator/
Desktop')
(a)
LU分解得到的L为:
[[ 1.          0.          0.          0.        ]
 [ 0.05        1.          0.          0.        ]
 [ 0.1        -0.40506329  1.          0.        ]
 [ 0.05       -0.01265823 -0.00925536  1.        ]]
LU分解得到的U为:
[[20.          2.          3.          0.        ]
 [ 0.          7.9         0.85        1.        ]
 [ 0.          0.         15.0443038   0.40506329]
 [ 0.          0.          0.          1.01640724]]
(b)
解得X1,X2,X3,X4的值分别为:
X1= [ 0.0509106   0.00082781 -0.00662252 -0.0509106 ]
X2= [-0.01614238  0.12168874  0.02649007  0.01614238]
X3= [-0.00910596 -0.00827815  0.06622517  0.00910596]
X4= [ 0.01614238 -0.12168874 -0.02649007  0.98385762]
矩阵A的逆为:
[[ 5.09105960e-02 -1.61423841e-02 -9.10596026e-03  1.61423841e-02]
 [ 8.27814570e-04  1.21688742e-01 -8.27814570e-03 -1.21688742e-01]
 [-6.62251656e-03  2.64900662e-02  6.62251656e-02 -2.64900662e-02]
 [-5.09105960e-02  1.61423841e-02  9.10596026e-03  9.83857616e-01]]

程序执行时间(不考虑I/O)为:0.000999450683593755

程序执行时间(考虑I/O)为:0.0039975643157958984
```

实验结果分析

环境对比

本地电脑：Intel(R) Core(TM) i7-4200MQ



国产计算平台：鲲鹏920

资源概述	
资源类型	vm
名称	中山大学汪涛老师《数值计算方法》专必大作业
数量	1
开放端口	22
架构	鲲鹏920
操作系统	麒麟服务器操作系统（ARM64版）V10（RPM格式）
所需软件	python3及相关工具包
资源详情	
CPU	8
内存	16GB
磁盘	200GB
网卡数量	1
云硬盘数量	0
云硬盘大小	0

实验结果对比

对A1与C2，

- 结果精度：两个平台表现一致
- 运算耗时：不考虑I/O时间的情况下，本地平台优于国产计算平台；考虑I/O时间的情况下，国产计算平台优于本地平台

对B2，

- 结果精度
  - Newton-Cotes求积公式：结果相同
  - Gauss-Legendre求积公式：对不同的n值，两个平台的计算结果均不相同。从有效数字出现差异的位置考虑，当n=5时在第15位有效数字的地方出现了差异，当n=20时在第8位有效数字的地方出现了差异；从误差的角度考虑，当n=5时两个平台计算结果与quad()函数计算所得精确结

    果的误差相同，大概在10的-2次方数量级，当n=20时误差降低为10的-10次方数量级，且本地电脑计算所得的误差略低于国产计算平台
  - 使用quad()函数：两个平台在第16位有效数字的地方出现了差异
- 运算耗时：本地电脑的运算耗时明显低于国产计算平台，约为国产计算平台运算耗时的1/3

原因分析

- Intel(R) Core(TM) i7-4200MQ处理器的**计算性能**要优于鲲鹏920处理器
- 鲲鹏920处理器的**I/O带宽**优于Intel(R) Core(TM) i7-4200MQ