计算物理大作业

姚铭星

1700011321

1 第一题

1.1 思路

2 第二题：高斯求积

、

2.1 问题描述

2.2 思路

具有如下形式的积分问题可以用高斯求积来解决：

$$\int_a^b w(x)f(x)dx$$

解决高斯求积问题的关键是找到一组在该权函数下的正交基。在定义内积

$$(f,g)=\int_a^b w(x)f(x)g(x)dx$$

的前提下，权重以及正交基满足下列关系:

$$\begin{aligned}
p_{-1}(x) &= 0 \\
p_0(x) &= 1 \\
p_{j+1}(x) &= (x-a_j)p_j(x)-b_jp_{j-1}(x)
\end{aligned}$$

其中：

$$a_j = \frac{(xp_j, p_j)}{(p_j, p_j)}$$
$$b_j = \frac{(p_j, p_j)}{(p_{j-1}, p_{j-1})}$$

设 x_μ 是正交多项式 $p_N(x)$ 的第 μ 个根，则系数满足：

$$w_j = \frac{(p_{N-1}(x), p_{N-1}(x))}{p_{N-1}(x_j)p'_N(x_j)}$$

3 第三题：放射衰变问题

3.1 问题描述

考虑 A 和 B 两类原子核随时间的放射衰变问题，t 时刻，其布居数分别为 $N_A(t)$ 和 $N_B(t)$。假定 A 类核衰变为 B 类核，B 类核可以继续衰变，满足以下微分方程组：

$$\frac{dN_A}{dt} = -\frac{N_A}{\tau_A}$$
$$\frac{dN_B}{dt} = \frac{N_A}{\tau_A} - \frac{N_B}{\tau_B}$$

其中,τ_A 和 τ_B 是时间衰变常数，在给定初始条件 $t_i = 0$ 时 $N_A(t_i) = N_B(t_i) = 1$ 下，回答下面三个问题：

1. 给出问题的解析解
2. 使用合适的算法数值求解上述耦合方程
3. 在给定 $\tau_A = 1s$，分别讨论 $\tau_B = 0.1s$,$1s$,$10s$, 三种情况下的短期和长期衰变行为。选取 $\tau_B = 10s$ 这种情况，讨论数值算法的误差，展示取不同步长 $\Delta t = 0.2s$,$0.1s$,$0.05s$ 时与解析结果的比较

3.2 解析结果

解析解可以直接给出：

$$\begin{gathered} N_A(t) = e^{-t/\tau_A} \\ N_B(t) = \frac{\tau_B}{\tau_A - \tau_B} e^{-t/\tau_A} + \frac{\tau_A - 2\tau_B}{\tau_B - \tau_B} e^{-t/\tau_B} \end{gathered} \tag{1}$$

‘

3.3 思路

由于方程形式很简单，可以直接用有限差分的形式来代替微分, 从而得到递推关系：

$$N_A((i+1)\Delta t) - N_A(i\Delta t) = -N_A(i\Delta t)\cdot\frac{\Delta t}{\tau_A}$$
$$N_B((i+1)\Delta t) - N_B(i\Delta t) = -N_B(i\Delta t)\cdot\frac{\Delta t}{\tau_B} + N_A(i\Delta t)\cdot\frac{\Delta t}{\tau_A}$$

经过 $N = t_{max}/\Delta t$ 步迭代就能得到答案。

3.4 实现

```
def decay(ta,tb,dt,tmax=1):
'''
衰变函数
返回<class='tuple'> (NA,NB)
'''
        n = 0
        NA=[1]
        NB=[1]
        while n*dt < tmax :
                NA.append(NA[n]*(1-dt/ta))
                NB.append(NB[n]*(1-dt/tb)+NA[n]*(dt/ta))
                n+=1
        return (NA,NB)
```

3.5 结果讨论

3.5.1 短期衰变

在固定 $\tau_A = 1s$ 后，可以认为，当时间 t_{max} 小于任何一个半衰期，就可以认为是短期衰变，这里取固定 $t_max = 0.1s$

计算结果如下图: 不难看出，当 $t\tau_B = 0.1s < \tau_A$ 时，二者短程呈现下降趋势。当 $\tau_B = 1 = \tau_A$ 时，N_B 短程内几乎不发生变化。而当 $\tau_B = 10s > \tau_A$ 时，N_B 开始呈现上升趋势，这与式 1是一致的。

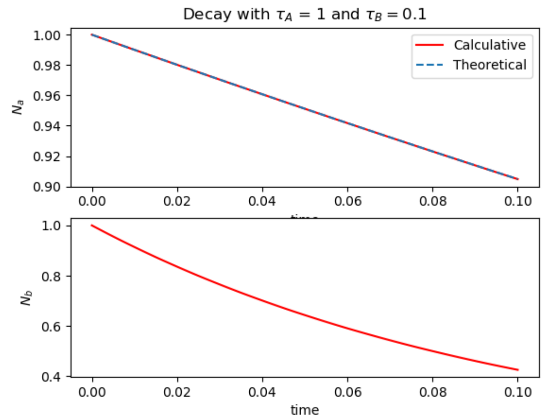

图 1: 短程结果

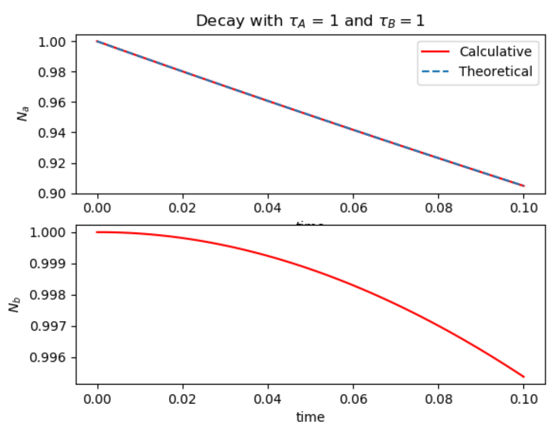

图 2: 短程结果

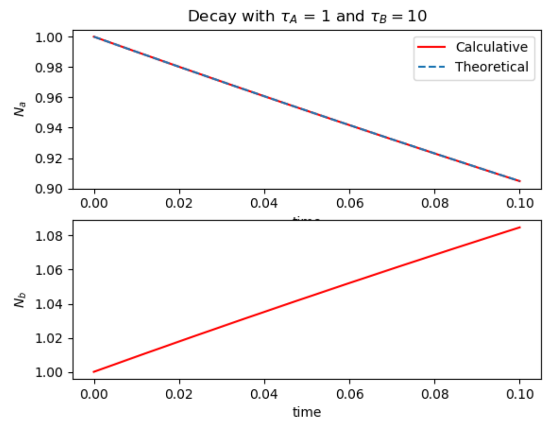

图 3: 短程结果

3.5.2 长期衰变

同样地，我们取 $t_{max} = 100s > max(\tau_A, \tau_B)$，以研究长程情况

计算结果如下图： 也可以看出，去除最开始的那段短程过程有些许不同之外，在最后都会趋于零。

3.5.3 误差

对于每一次计算，可以利用泰勒展开估计误差

$$N((i+1)\Delta t) = N(i\Delta t) + N'(i\Delta t) \cdot \Delta t + \frac{1}{2}N^{(2)}(\xi)\Delta t^2$$

在忽略机器精度的情况下，绝对误差可以估计为一个上界：

$$\Delta(t) \leq \frac{1}{2}\frac{\Delta t \cdot t}{\tau^2}$$

等号当且仅当在 $t = 0$ 时成立。

下面是实际的情况（绝对误差随时间的变化）, 可以看出，在尺度上而言，的确是与 Δt 的取值成正比

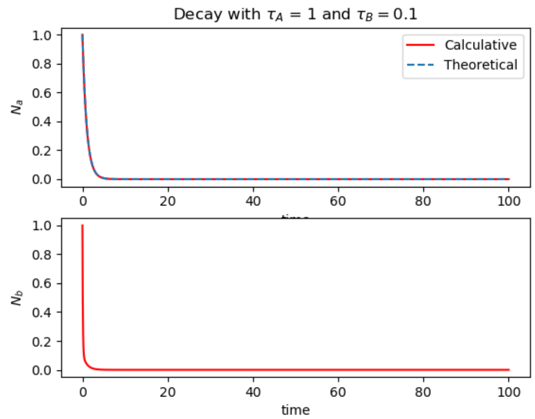

图 4: 长期衰变

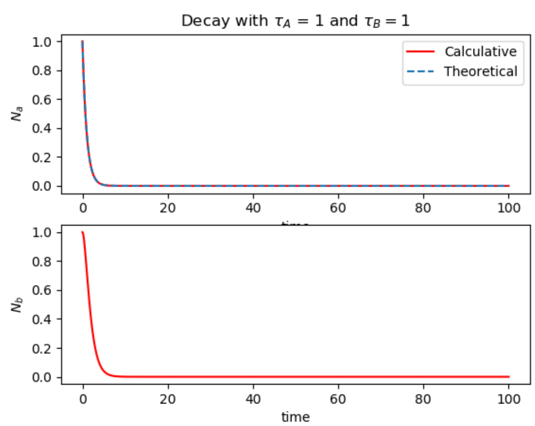

图 5: 长期衰变

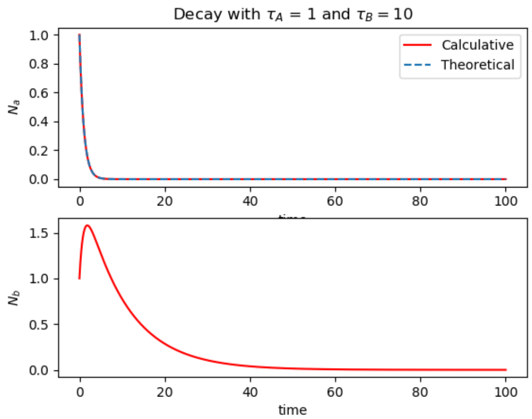

图 6: 长期衰变

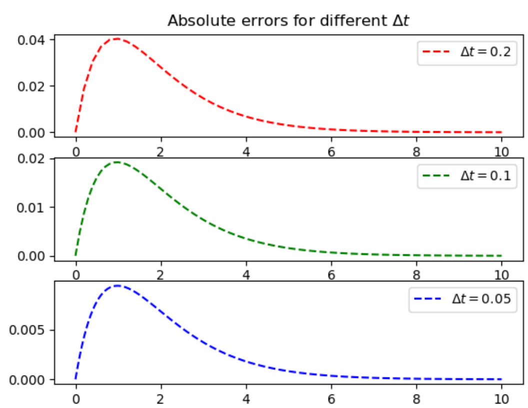

图 7: 误差随时间的变化

4 第四题：三次样条差值

4.1 问题描述

对于飞机机翼的轮廓每隔 $0.1m$ 的值，利用自然边界条件，使用三次样条插值估计任意一点的 y 值，需要与原始数据点一同绘制出飞机的轮廓曲线。

x	0	3	5	7	9	11	12	13	14	15
y	0	1.2	1.7	2.0	2.1	2.0	1.8	1.2	1.0	1.6

4.2 三次样条插值

三次样条差值也是多项式插值的一种。相比于拉格朗日插值和牛顿插值等多项式插值，三次样条方法拥有更加高阶的导数连续性。具体来说，三次样条方法不仅要求在插值点上有连续的函数值，而且还要求有连续的一阶以及二阶导数值。即因满足条件：

$$\left\{\begin{array}{l} S\left(x_i-0\right)=S\left(x_i+0\right) \\ S^{\prime}\left(x_i-0\right)=S^{\prime}\left(x_i+0\right), \quad i=1,2,\cdots,n-1 \\ S^{\prime\prime}\left(x_i-0\right)=S^{\prime\prime}\left(x_i+0\right) \end{array}\right.$$

在每一个区间 $x\in\left[x_k,x_{k+1}\right]$ 内部使用三次函数差值。

类似于 Hermite 插值法，使用分段 Hermite 多项式，此时零阶和一阶条件自动满足：

$$H_3^{(k)}(x)=y_k\alpha_0^{(k)}(x)+y_{k+1}\alpha_1^{(k)}(x)+y_k^{\prime}\beta_0^{(k)}(x)+y_{k+1}^{\prime}\beta_1^{(k)}(x)$$

取用各节点的一阶导数为未知数 $S^{\prime}\left(x_i\right)=m_i$，利用二阶导数条件：

$$\mu_i m_{i+1}+2m_i+\lambda_i m_{i-1}=e_i,\quad i=1,2,\cdots,n-1$$

其中 $\mu_j=\frac{h_{j-1}}{h_{j-1}+h_j},\quad \lambda_j=\frac{h_j}{h_{j-1}+h_j}, e_i=3\left(\lambda_i f\left[x_{i-1},x_i\right]+\mu_i f\left[x_i,x_{i+1}\right]\right)$

并且使用自然边界条件：

$$S^{\prime\prime}(0)=S^{\prime\prime}(x_{n-1})=0$$

这样问题变为求解三对角矩阵方程解问题，可以利用追赶法快速求解。

4.3 实现

```
def firstorder(x,y):
        '''
        返回插值中n个点的一阶导数m_i
        采用三转角方程求解一阶导数
        '''
        # 建立矩阵
        mu=[h(i-1)/(h(i)+h(i-1)) for i in range(1,n)]
        lamb=[h(i)/(h(i)+h(i-1)) for i in range(1,n)]
        # 建立待解向量
        b = [3*d(0)]+[3*(lamb[i-1]*d(i-1)+mu[i-1]*d(i))\# 写不下换行
                for i in range(1,n)]+[3*d(n-1)]
        mu = [1] + mu
        lamb = lamb +[1]
        dia = [2 for _ in range(n+1)]
        # 开始追赶法求解三对角矩阵
        for i in range(1,n+1):
                lamb[i-1] = lamb[i-1]/dia[i-1]
                dia[i] = dia[i] - mu[i-1]* lamb[i-1]
                b[i] -= b[i-1]*lamb[i-1]
                m = [-1 for _ in range(n+1)]
        m[n] = b[n]/dia[n]
        for i in range(n-1,-1,-1):
                m[i] = (b[i] - mu[i]*m[i+1])/dia[i]
        return m
```

4.4 结果分析

可以看出，函数的光滑性是很好的。