作业 4：有限元方法求解一维微分方程

谢文进

2021 年 6 月 29 日

1 作业 4：有限元方法求解一维微分方程

1.1 理论推导

对于微分方程

$$\begin{cases} -u'' = f, \quad 0 < x < 1 \\ u(0) = 0, u'(1) = 0. \end{cases}$$

现在取 $f = 1$, 求解如下方程：

$$\begin{cases} -u'' = 1, \quad 0 < x < 1 \\ u(0) = 0, u'(1) = 0, \end{cases}$$

上述方程精确解为 $u(x) = -\frac{1}{2}x^2 + x$. 现在用有限元方法进行数值求解，重要的是计算出总体刚度矩阵 A 以及载荷向量 b。

$$A = \sum_{i=1}^{N} \begin{pmatrix} & & & \\ & & & \\ \cdots & \frac{1}{h_i} & -\frac{1}{h_i} & \cdots \\ \cdots & -\frac{1}{h_i} & \frac{1}{h_i} & \cdots \\ & & & \\ & & & \end{pmatrix}_{(N+1)\times(N+1)}$$

$$
b = \sum_{i=1}^{N} \frac{h_i}{2} \int_{-1}^{1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ f(F(\xi))N_1(\xi) \\ f(F(\xi))N_2(\xi) \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{(N+1)\times 1} d\xi
$$

代入 $f(x) = 1$, 可以得到

$$
\int_{-1}^{1} f(F(\xi))N_1(\xi)d\xi = \int_{-1}^{1} \frac{1}{2}(1-\xi)d\xi = 1,
$$

$$
\int_{-1}^{1} f(F(\xi))N_2(\xi)d\xi = \int_{-1}^{1} \frac{1}{2}(1+\xi)d\xi = 1.
$$

因此

$$
b = \sum_{i=1}^{N} \frac{h_i}{2} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{(N+1)\times 1} .
$$

现在处理边界条件 $u(0) = 0, u'(1) = 0$, 由于 $u'(1) = 0$ 在证明过程中自然满足，只需处理边界条件 $u(0) = 0$. 记 $Au = b$, 即

$$
\begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0N} \\ a_{10} & a_{11} & \cdots & a_{1N} \\ \cdots & \cdots & \cdots & \cdots \\ a_{N0} & a_{N1} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_N \end{pmatrix} .
$$

由此可得

$$\begin{aligned} a_{01}u_1 + \cdots + a_{0N}u_N &= b_0 - a_{00}u_0 \\ a_{11}u_1 + \cdots + a_{1N}u_N &= b_1 - a_{10}u_0 \\ &\cdots \\ a_{N1}u_1 + \cdots + a_{NN}u_N &= b_N - a_{N0}u_0 \end{aligned}$$

加入一个方程 $1 \times u_0 + 0 \times u_1 + \cdots + 0 \times u_N = u_0$, 写成矩阵形式得到:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & a_{11} & \cdots & a_{1N} \\ \vdots & \vdots & & \vdots \\ 0 & a_{N1} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} u_0 \\ b_1 \\ \vdots \\ b_N \end{pmatrix} - \begin{pmatrix} 0 \\ a_{10} \\ \vdots \\ a_{N0} \end{pmatrix} u_0,$$

代入 $u_0 = 0$ 得到最终的方程组

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & a_{11} & \cdots & a_{1N} \\ \vdots & \vdots & & \vdots \\ 0 & a_{N1} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} 0 \\ b_1 \\ \vdots \\ b_N \end{pmatrix},$$

求解上述方程组即可得到数值解 u。

1.2 编程实现

根据上述推导，用 python 编写程序，代码如下:

```python
import numpy as np
import math
from scipy import linalg # 求解方程组需要引用的包
import matplotlib.pyplot as plt # 画图

# 求解微分方程 -u'' = 1, u(0) = 0, u'(1) = 0
# 精确解为 u(x) = -0.5 * x^2 + x
x0 = 0.0
xN = 1.0
N = 10 # 节点x0, x1, ..., xN
h = (xN - x0) / N

# 精确解 u(x) = -0.5 * x^2 + x
```

```python
def exactf(x):
    return (-1) * 0.5 * x * x + x

# f(F(\xi))N1(\xi)在[-1,1]上的积分值
def integrate_f1():
    return 1

# f(F(\xi))N2(\xi)在[-1,1]上的积分值
def integrate_f2():
    return 1


# 生成矩阵A和右端项RHS
def generate(N, f1, f2):
    A = np.zeros([N+1, N+1])
    RHS = np.zeros([N+1,1])
    for i in range(1, N+1):
        A[i,i] += 1/h
        A[i-1, i-1] += 1/h
        A[i-1,i] += -1/h
        A[i,i-1] += -1/h
        RHS[i] += (h/2) * f1
        RHS[i-1] += (h/2) * f2
    return A,RHS


# 处理边界条件 u(0) = 0
def modified_matrix(A, RHS, u0, uN):
    # 1. 先更新右端项
    # 处理边界条件u0
    for i in range(1,N):
        RHS[i] = RHS[i] - A[i,0] * u0
    RHS[0] = u0

    # # 处理边界条件uN, 当给出u(xN)的值时, 取消注释
    # for i in range(1,N):
    #     RHS[i] = RHS[i] - A[i,N] * uN
    # RHS[N] = uN

```

```python
    # 2. 更改矩阵A
    for i in range(N+1):
        A[0,i] = 0
        A[i,0] = 0
        # A[N,i] = 0 # 当给出u(xN)的值时，取消注释
        # A[i,N] = 0 # 当给出u(xN)的值时，取消注释
    A[0,0] = 1
    # A[N,N] = 1 # 当给出u(xN)的值时，取消注释
    return A,RHS

A,RHS = generate(N,integrate_f1(),integrate_f2())
A,RHS = modified_matrix(A, RHS, 0, 0)

# 数值解 求解方程组 Ax=RHS
x = linalg.solve(A, RHS)
# print(x)

# 精确解
u = np.zeros([N+1,1])
for i in range(N+1):
    u[i] = exactf(i * h)
# print(u)

# 输出误差
err = max(abs(u - x))
print(N,err)

# 画图比较精确解和数值解
t = np.arange(x0, xN + h, h)
plt.title('Result')
plt.plot(t, x, color='green', label='numerical solution')
plt.plot(t, u, color='blue', label='exact solution')
plt.legend() # show the legend

plt.xlabel('t')
plt.ylabel('u')
plt.show()
```

表 1: 不同 h 误差表

h	error
1/4	2.77555756e-16
1/8	5.55111512e-17
1/16	1.11022302e-16
1/32	8.38218384e-15
1/100	8.32667268e-15
1/200	6.29496455e-14

1.3 结果分析

当取 $N = 10$, 即 $h = \frac{1}{N} = 0.1$ 时，精确解 u 与数值解 u_1 图像如图 1 所示，可以看到两条线几乎重合，计算误差 $||u - u_1||_{\max} = 2.22044605e - 16$

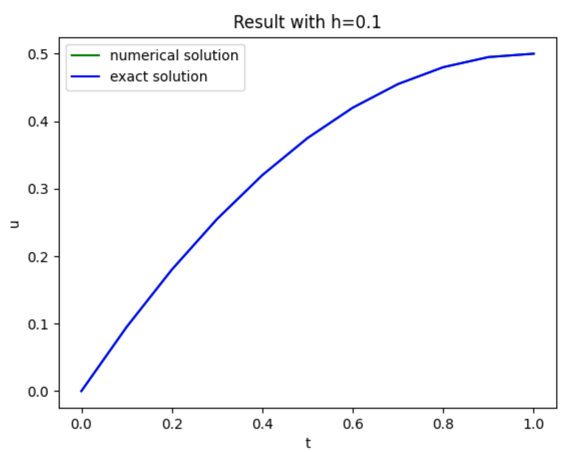

图 1: $h = 0.1$ 结果图

取不同的 h 得到的误差如表 1所示。由上述数值结果可以验证此方法及程序的正确性。