微分方程数值解计算实习 Lecture 4

朱荃凡

(吉林大学数学系计算唐班)

2023 年 6 月 4 日

1 问题重述

用线性元在均匀网格下求解边值问题(1.1)的数值解:

$$\begin{aligned} -y'' + \frac{\pi^2}{4}y &= \frac{\pi^2}{2}\sin\frac{\pi}{2}x,\ 0 < x < 1, \\ y(0) = 0, &\qquad y'(1) = 0. \end{aligned} \tag{1.1}$$

要求:

- 画出数值解的图像
- 对 $[0,1]$ 区间均匀剖分成 $N = 10, 20, 30, \cdots, 200$ 份, 计算数值解和真解

  $$u* = \sin\frac{\pi}{2}x$$

  的 L^2 误差和 H^1 误差, 计算其关于网格长度 $h = 1/N$ 的数值收敛阶.
- 计算系数矩阵的条件数, 并用 loglog() 函数作图表示.

2 算法设计

考虑到可读性和泛化能力, 该程序的主函数 FEM_zqf 由运行参数, 计算有限元的函数 run_main 和一些列辅助函数组成. 运行参数包括带求函数所在区间, 边值条件和网格剖分的数量和计算收敛阶时迭代次数. 辅助函数包括真解, 基函数, 右端函数 $f(x)$, 高斯积分公式等等.

重点来看 run_main() 函数, 它包含如下几个部分:

- 标准区间上高斯积分公式的选点和权重
- 生成刚度矩阵 stiffnessMatrix()
- 生成右端项 rightHands()
- 处理边值条件 boundaryMatrix()
- 解方程组 solveAF()
- 计算误差 errorEstimate()
- 作图 plotFigure()

接下来逐一解释每个部分的功能与作用.

2.1 高斯积分公式的选点和权重

由于使用有限元时已经对区间做过了一次剖分, 在每个小区间数值积分就没有必要使用复化 Simpson 公式, 故而采取高斯积分公式是不错的选择. 出于精度和计算量的考虑, 这里选取了五点高斯积分公式, 其对应的节点和权重分别是

$$\begin{aligned} positions &= [-0.9061798, -0.5384693, 0, 0.5384693, 0.9061798] \\ weights &= [0.2369269; 0.4786287; 0.5688889; 0.4786287; 0.2369269] \end{aligned} \tag{2.1}$$

想要得到任意小区间上的积分值, 首先要做一个仿射变换, 把 $[-1, 1]$ 区间上的节点变换到对应区间上的节点. 然后带入被积分函数中得到函数值. 最后再通过一个高斯积分函数得到结果.

2.2 生成刚度矩阵 stiffnessMatrix()

设网格剖分后节点 x_i 处的基函数是

$$\varphi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h}, & x_{i-1} \le x \le x_i, \\ \dfrac{x_{i+1} - x}{h}, & x_i \le x \le x_{i+1}, \\ 0, & others. \end{cases} \tag{2.2}$$

需要注意的是在区间端点处的基函数只有半支.

再设由方程(1.1)方程得到的双线性形式

$$a(u,v)=\int_0^1 (u'v'+\frac{\pi^2}{4}uv)\,\mathrm{d}x. \tag{2.3}$$

那么刚度矩阵中的元素 a_{ij} 可表示成

$$a_{ij}=\int_0^1 (\varphi_i'\varphi_j'+\frac{\pi^2}{4}\varphi_i\varphi_j)\,\mathrm{d}x. \tag{2.4}$$

但是直接生成全矩阵并不是一个很好的方式, 我们可以逐区间来看. 首先生成一个 $(N+1)\times(N+1)$ 的零矩阵 A, 然后考虑每个区间上的单元矩阵. 在区间 $I_j=[x_{j-1,x_j}]$ 上, 仅涉及到两个基函数 φ_{j-1},φ_j 和四个矩阵元素 $A_{j-1,j-1},A_{j-1,j},A_{j,j-1}$ 和 A_{jj}. 具体来说, 对矩阵元素的操作可表示为

$$\begin{aligned}
&A_{j-1,j-1}=A_{j-1,j-1}+a, \qquad A_{j,j}=A_{j,j}+a,\\
&A_{j-1,j}=A_{j-1,j}+b, \qquad A_{j,j-1}=A_{j,j-1}+b,\\
a=\int_{x_{j-1}}^{x_j}&\left[(\varphi_j')^2+\frac{\pi^2}{4}\varphi_j^2\right]\mathrm{d}x,\ b=\int_{x_{j-1}}^{x_j}\left[\varphi_j'\varphi_{j-1}'+\frac{\pi^2}{4}\varphi_j\varphi_{j-1}\right]\mathrm{d}x,
\end{aligned}$$

通过计算可以得到

$$a=\frac{1}{h}+\frac{\pi^2}{12},\quad b=-\frac{1}{h}+\frac{\pi^2}{24}$$

该数值与区间无关, 因而可以简单地使用一个 for 循环实现, 最终我们就得到了刚度矩阵 A.

2.3 生成右端项 rightHands()

生成右端项 F 时, 采取和上述矩阵类似的方法. 首先生成一个 $N+1$ 列的零向量 F, 然后在区间 $I_j=[x_{j-1},x_j]$ 上,

$$\begin{aligned}
F_{j-1}&=F_{j-1}+\int_{x_{j-1}}^{x_j} f(x)\varphi_{j-1}(x)\,\mathrm{d}x,\\
F_j&=F_j+\int_{x_{j-1}}^{x_j} f(x)\varphi_j(x)\,\mathrm{d}x.
\end{aligned} \tag{2.5}$$

此处直接采用数值积分公式即可.

2.4 处理边值条件 boundaryMatrix()

在编程课上的时候, 我和其他同学有过讨论: 由于 Dirichlet 边值条件, 所以我们已经知道了 y_0 的取值, 所以到底是要解一个 $N\times N$ 的系数矩阵, 还是解一个 $(N+1)\times(N+1)$ 的系数矩阵. 我个人的想法是解 $(N+1)\times(N+1)$ 的, 这样可以把生成矩阵和处理边值条件独立开来, 降低程序的耦合度. 对于不同的边值问题也方便修改.

2.4.1 Dirichlet 边值条件

处理 Dirichlet 边值条件 $y(0) = 0$ 总共有三种可选方法:

- 直接把矩阵的第一行元素和对应右端项清零, 然后令 $A_{00} = 1$.
- 删掉矩阵的第一行和第一列, 和右端项的第一行. 解完方程组以后再把 $y_0 = 0$ 补回去.
- 直接令 A_{00} 和为一个超级大的数 (例如说 10^{15}), 这样的话第一列其他矩阵元素对 y_0 的影响就很小, 解方程即可.

我在程序中采取了第一种方法.

2.4.2 Neumann 边值条件

处理 Neumann 边值条件只用修改右端项

$$F_j = F_j + p(b)\beta\varphi_j(b), \quad j = 0, 1, \cdots, n. \tag{2.6}$$

而事实上只修改了第 $n+1$ 个方程的右端项, 因为当 $j = 0, 1, \cdots, n-1$ 时, $\varphi_j(b) = 0$.

2.5 解方程组 solveAF()

直接解方程组 $y = A\backslash F$.

2.6 计算误差 errorEstimate()

计算 L^2 误差时先在每个区间上对真解与数值解之差的绝对值用高斯积分公式, 然后求和. 事实上我选用了 H^1 半模, 也就是真解与数值解导数之差绝对值的 L^2 模来代替 H^1 模, 这并不会影响误差的收敛速度.

如果不想用 for 循环的话, 也可以使用矩阵来储存节点坐标, 其中矩阵的每一行表示小区间上的高斯积分节点. 最后再用 sum() 函数求和. 这样做的好处是可以提升程序的运行速度.

2.7 作图 plotFigure()

画出对应图像即可.

3 程序结果

3.1 数值解图像

这里分别展示剖分数 $N = 10$ 和 $N = 50$ 时的图像, 其中橙色实线表示真解, 蓝色点表示数值解. 即使是 $N = 10$ 时, 线性有限元的计算结果依旧相当好.:

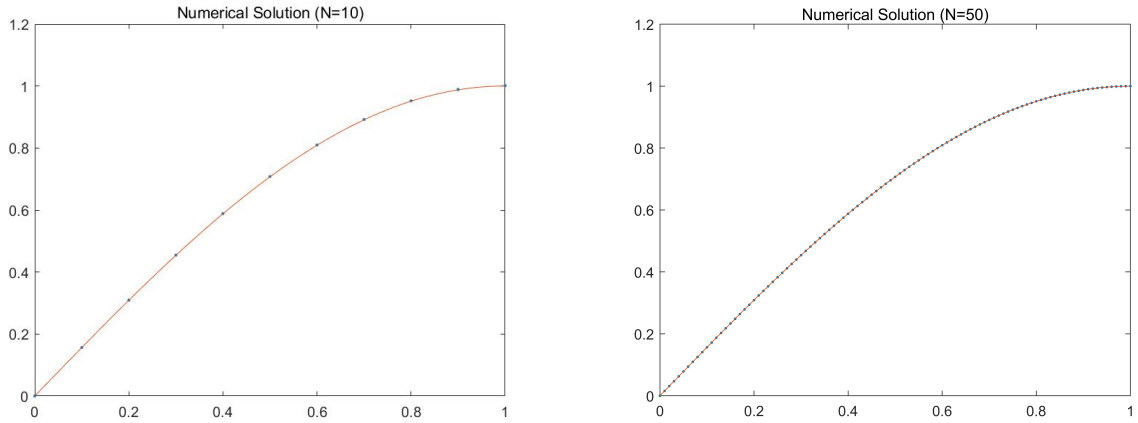

图 1: 真解和数值解

3.2 误差和收敛阶

接下来计算画出和收敛阶, 从图二图三中可以看出 $H1$ 误差的收敛速度是 $o(h)$, $L2$ 误差的收敛速度是 $o(h^2)$, 这与理论是相符合的.

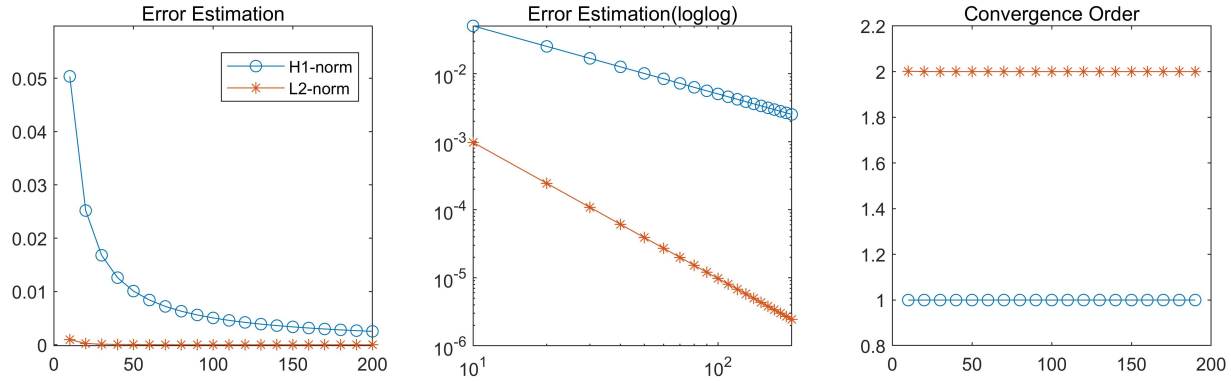

图 2: 误差和收敛阶

3.3 矩阵条件数

最后是刚度矩阵的条件数 $Cond(A)$. 假设 $Cond(A)$ 的发散速度为

$$Cond(A) = CN^{\alpha}, \tag{3.1}$$

其中 N 为剖分数. 对于不同的 N_1, N_2 有

$$\alpha = \frac{\log Cond(A_1) - \log Cond(A_2)}{\log(N_1) - \log(N_2)}. \tag{3.2}$$

于是我们画出如下的图像, 其中图一是矩阵条件数与节点数量的双对数图, 图二是发散阶.