FEM Homework Report

Yue Wu*

September 7, 2022

1 Introduction

Make a program to solve the two-point boundary value problem:

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

Use an equidistant mesh and a piecewise linear polynomial space V_h as the finite element space. Use $f(x) = -2\cos x + (x-1)\sin x$, $u(x) = (x-1)\sin x$ to test your program, and compute the following errors:

$$\|u - u_h\|_{L^2[0,1]}, \quad \|u - u_h\|_{H^1[0,1]} \tag{2}$$

2 Method

We use the following equidistant mesh:

$$x_j = jh,\ j = 0, 1, \cdots, N+1, \quad h = \frac{1}{N+1}, \quad I_j = [x_j, x_{j+1}] \tag{3}$$

and we use the following finite element space:

$$V_h = \left\{ v \in C[0,1] \middle| v|_{I_j} \in \mathcal{P}^1(I_j), j = 1, \cdots, N+1,\ v(0) = v(1) = 0 \right\} \tag{4}$$

The Galerkin approximation to the problem using approximation space V_h is given by the following:

$$\begin{aligned} &\text{find } u_h \in V_h, \text{ such that:} \\ &\forall v_h \in V_h,\ \left(u_h', v_h'\right) = (f, v_h) \end{aligned} \tag{5}$$

For every function $v_h \in V_h$, we use the nodal representation as the following:

$$v_h(x) = \sum_{j=1}^{N} v_h(x_j)\phi_j(x) \tag{6}$$

*School of the Gifted Young, University of Science and Technology of China, Hefei, Anhui, China. E-mail: pilotjohnwu@mail.ustc.edu.cn

then the discrete Galerkin system becomes:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \tag{7}$$

where:

$$\boldsymbol{A} = \left(\left(\phi_i', \phi_j'\right)\right)_{N\times N} \tag{8}$$

$$\boldsymbol{u} = \left(u_h(x_i)\right)_{N\times 1} \tag{9}$$

$$\boldsymbol{f} = \left((f, \phi_i)\right)_{N\times 1} \tag{10}$$

To approximate integrals of non-polynomial functions **in computing the right-hand-side term $\boldsymbol{f}$**, we apply the 2-point Gauss-Legendre quadrature rule **in both cases**.

To approximate integrals of non-polynomial functions **in computing the errors**, we apply the 2-point Gauss-Legendre quadrature rule and the 2-point Gauss-Lobatto quadrature rule **respectively**.

Here, we give the 2-point Gauss-Legendre quadrature rule and the 2-point Gauss-Lobatto quadrature rule:

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}}) \tag{11}$$

$$\int_{-1}^{1} f(x)\mathrm{d}x \approx f(-1) + f(1) \tag{12}$$

3 Results

We use $N = 10, 20, 40, 80$ DOFs for computation. By running the scripts in `main.m`, the outputs from the `matlab` program we coded are given as following:

N	L^2 error	order	H^1 error	order
10	1.2857e-03	-	4.9020e-02	-
20	3.5276e-04	2.0000	2.5666e-02	1.0007
40	9.2545e-05	2.0000	1.3144e-02	1.0002
80	2.3711e-05	2.0000	6.6532e-03	1.0000

Table 1: Table of error and order under different norms using Gauss-Legendre rule

N	L^2 error	order	H^1 error	order
10	2.3122e-08	-	8.4844e-02	-
20	1.7400e-09	4.0006	4.4446e-02	0.9999
40	1.1974e-10	4.0002	2.2766e-02	1.0000
80	7.8678e-12	3.9986	1.1523e-02	1.0000

Table 2: Table of error and order under different norms using Gauss-Lobatto rule

4 Discussion

From this experiement, we verified the theoretical 2-nd order convergence under the L^2 norm and 1-st order convergence under the H^1 norm numerically. In the algorithm, we used the 2-point Gauss-Legendre quadrature rule to approximate integration of non-polynomial functions in the right-hand-side. This does not cause the convergence order to degenerate since the quadrature is 4-th order accurate.

There is an interesting fact that from the error table resulting from different quadrature rules, we observe significantly different convergence rates in the L^2 norm. This is a typical phenomenon of the superconvergence theory of finite element methods: the numerical solution approximates the mathematical solution better at some specific points (superconvergent point) than anywhere else. In fact, the 4-th order L^2 convergence here verifies the superconvergence at cell boundaries instead of the true L^2 convergence.