有限元方法代码作业 1 报告

SA23001071 杨哲睿

2023 年 9 月 15 日

1 Introduction

编写程序求解两点边值问题

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

选取等距网格剖分，有限元空间选取分段线性多项式空间 V_h。其中，选取了 $f(x) = -(2\cos x - (x-1)\sin x)$，已知其解为 $u(x) = (x-1)\sin x$。完成求解后，使用 L^2 范数和 H^1 范数计算误差并讨论结果。

2 Method

定义双线性型 $a(u,v) = \int_0^1 u'v'\mathrm{d}x$，内积 $(f,g) = \int_0^1 f\cdot g\mathrm{d}x$。那么问题 (1) 的变分问题是

找 $u \in V = \{v \in C[0,1], v(0) = v(1) = 0\}$，使得 $a(u,v) = (f,v)$，$\forall v \in V$ 都成立.

实验中采用等距网格划分，节点数为 N，节点处的值为 u_i，$h = u_{i+1} - u_i$。选取基函数为 ϕ_i，并选取其张成的有限维线性空间进行求解，基函数定义如下：

$$\phi_i = \begin{cases} \frac{x - x_{j-1}}{x_j - x_{j-1}} & x_{j-1} \le x < x_j \\ \frac{x_{j+1} - x}{x_{j+1} - x_j} & x_j \le x < x_{j+1} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

对于 a，计算

$$a(\phi_i, \phi_j) = \begin{cases} -\frac{1}{h}, & i = j+1 \text{ 或} j = i+1 \\ \frac{2}{h}, & i = j \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

因此，刚度矩阵

$$A = \frac{1}{h} \begin{pmatrix} 2 & -1 & \cdots & 0 & 0 \\ -1 & 2 & \cdots & 0 & 0 \\ 0 & \vdots & & \vdots & 0 \\ 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & \cdots & -1 & 2 \end{pmatrix} \tag{4}$$

计算 (f, ϕ_j) 得：

$$\begin{aligned} f_j &= \int_0^1 f \phi_j \mathrm{d}x \\ &= \frac{4(hj-1)\sin^2(h/2)\sin(hj)}{h} - 2\sin(h)\cos(hj) \end{aligned} \tag{5}$$

因此为求解节点上的值, 只需要求解出 $AU = F$，其中：

$$U = \begin{pmatrix} u_1 \\ \vdots \\ v_N \end{pmatrix}, \quad F = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix} \tag{6}$$

在估计 L^2 和 H^1 误差进行时，使用梯形公式来进行近似。

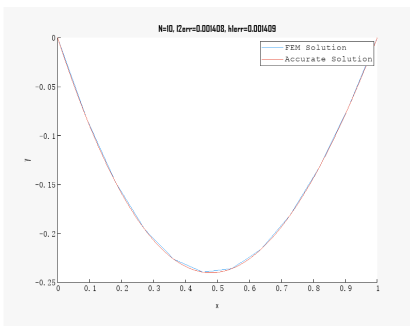
3 Results

实验得到的结果如下：

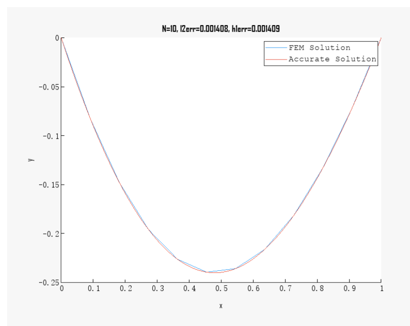
表 1: 误差分析

n	L^2 error	order	H^1 error	order
10	1.41E-03	–	1.43E-03	–
20	3.86E-04	1.87	3.88E-04	1.88
40	1.01E-04	1.93	1.07E-04	1.85
80	2.6E-05	1.95	4.2E-05	1.35

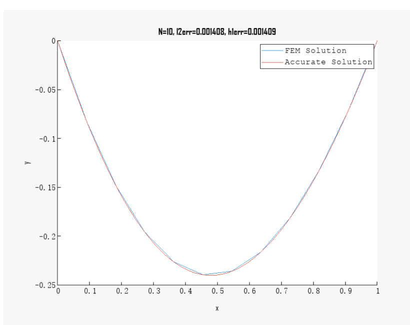
观察到，该方法对 L^2 误差具有二阶精度，对 H^1 误差具有一阶精度. 有限元解和真解对比如下图所示：



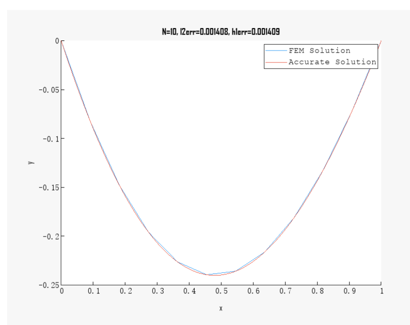
(a) $N = 10$



(b) $N = 20$



(c) $N = 40$



(d) $N = 80$

图 1: 使用北太天元绘制的数值解和真解图像

4 Discussion

本节着重进行误差分析，令 $v \in V_h$ 为一分片线性函数，满足：

$$v(x_i) = u(x_i) \quad \forall i = 1, 2, \cdots, N \tag{7}$$

其中 u 是两点边值问题的古典解。令 $e = v - u$, 那么

考虑区间 $[a, b]$ 上的 C^1 函数 f，满足 $f(a) = f(b) = 0$，那么 $|f| < \frac{b-a}{2} \max |f'|$，积分

$$\int_a^b |f(x)| \mathrm{d}x < \frac{(b-a)^2}{2} \max |f'| \tag{8}$$

对 e, e' 反复利用该结论以及微分中值定理可知，对于 e, e' 有如下的最大误差估计：

$$\begin{cases} \sup_x |e'| \leq C_1 h \sup_x |u''| \\ \sup_x |e| \leq C_2 h^2 \sup_x |u''| \end{cases} \tag{9}$$

那么

$$\begin{cases} ||u - v||_2 = \leq C_1 \cdot h^2 \sup_x |u''| \\ ||u' - v'||_2 = \leq C_2 \cdot h \sup_x |u''| \end{cases} \tag{10}$$

代入 L^2 误差和 H 误差的表达式中可知，其在 L^2 意义下有二阶精度，H^1 意义下有一阶精度。

5 Computer Code

5.1 main.m

```
clear;
% Grid size, ...
% compute the accurate data.
N_sample = 16384;
x_sample = linspace(0, 1, N_sample);
u_accurate = (x_sample - 1) .* sin(x_sample);

N = 80;
h = 1.0 / (N+1);
```

```
x = linspace(0, 1, N + 2);
% Stiffness matrix
A = compute_a(N - 2, h);

% Righthand side
fi = zeros(1, N);
for i = 1 : N
    fi(i) = compute_fi(h, i);
end

% Solve the linear equation.
u_inner = fi / A;
u = zeros(1, N + 2);
% setup the result
u(2:N+1) = u_inner;
% Trapezoidal Rule for Integration
u_sample = interp1(x, u, x_sample);
% L2 norm
l2err = sqrt(trapz(1 / N_sample, ((u_sample - u_accurate) .^ 2)'));
% H2 norm, first compute the
u_diff_accurate = diff(u_accurate);
u_diff_sample   = diff(u_sample);
u_diff_err_sqr = trapz(1 / N_sample, ...
    ((u_diff_accurate - u_diff_sample) .^ 2 / h / h)');
h1err = sqrt(l2err ^ 2 + u_diff_err_sqr);

% Plotting
plot(x_sample, u_sample)
hold on
plot(x_sample, u_accurate)
xlabel("x")
ylabel("y")
legend("FEM Solution", "Accurate Solution")
title(sprintf("N=%d, l2err=%f, h1err=%f", N, l2err, h1err))
```

5.2 compute_a.m

```
function [spmt] = compute_a(N, h)
```

```
    % Fill the sparse matrix.
    A = eye(N + 2) * 2;
    A(1:N+1, 2:N+2) = A(1:N+1, 2:N+2) - eye(N+1);
    A(2:N+2, 1:N+1) = A(2:N+2, 1:N+1) - eye(N+1);
    spmt = 1/h * A;
end
```

5.3 compute_fi.m

```
% The value =
%       Integrate[ f[x] * phi_n [x], {x, n h - h, n h + h} ]
% The formula is derived and simplified by Mathematica. See Derivatives.nb
% for more details.

function [value] = compute_fi(h, n)
  value = -2*cos(h*n)*sin(h) + (4*(-1 + h*n)*sin(h/2)^2*sin(h*n))/h;
end
```