

Math 5601 Final Project

Jacob Hauck

December 4, 2023

Consider the following second-order ODE with Dirichlet boundary conditions:

$$\frac{d}{dx} \left(c(x) \frac{du(x)}{dx} \right) = f(x), \quad a \leq x \leq b, \quad (1)$$

$$u(a) = g_a, \quad u(b) = g_b. \quad (2)$$

Problem 1.

Consider the second-order ODE (1). Multiplying by $v \in H^1([a, b])$ and integrating by parts gives

$$\int_a^b f v = c(b) u'(b) v(b) - c(a) u'(a) v(a) - \int_a^b c u' v'. \quad (3)$$

(a) Suppose we have the boundary conditions

$$u'(a) = p_a, \quad u(b) = g_b. \quad (4)$$

Equation (3) still holds, and we can impose the condition $v(b) = 0$ because we already know that $u(b) = g_b$. Since $u'(a) = p_a$, equation (3) becomes

$$\int_a^b f v = -c(a) p_a v(a) - \int_a^b c u' v' \quad (5)$$

for all $v \in H^1([a, b])$ such that $v(b) = 0$, which is our weak formulation of (1) with the given boundary conditions.

(b) Suppose we have the boundary conditions

$$u'(a) = p_a, \quad u'(b) + q_b u(b) = p_b. \quad (6)$$

Equation (3) still holds. Since $u'(b) = p_b - q_b u(b)$, and $u'(a) = p_a$, we get

$$\int_a^b f v = c(b) (p_b - q_b u(b)) v(b) - c(a) p_a v(a) - \int_a^b c u' v' \quad (7)$$

for all $v \in H^1([a, b])$, which is our weak formulation of (1) with the given boundary conditions.

(c) Suppose we have the boundary conditions

$$u'(a) = p_a, \quad u'(b) = p_b. \quad (8)$$

Equation (3) still holds. Since $u'(a) = p_a$, and $u'(b) = p_b$, we get

$$\int_a^b f v = c(b) p_b v(b) - c(a) p_a v(a) - \int_a^b c u' v' \quad (9)$$

for all $v \in H^1([a, b])$, which is our weak formulation of (1) with the given boundary conditions.

We note that solutions of this formulation are not unique. Indeed, if $u \in H^1([a, b])$ satisfies (9) for all $v \in H^1([a, b])$, then so does $u + \alpha$, where $\alpha \in \mathbf{R}$ is any real number because $(u + \alpha)' = u'$ regardless of what α is, and the weak formulation depends only on u' .

Problem 2.

Consider the Poisson equation

$$\nabla \cdot (c \nabla u) = f \text{ in } D. \quad (10)$$

Using integration by parts, we have

$$\int_D f v = \int_D \nabla \cdot (c \nabla u) v = \int_{\partial D} c v \nabla u \cdot n \, dS - \int_D c \nabla u \cdot \nabla v, \quad (11)$$

where dS is the surface measure on ∂D , and $v \in H^1(\overline{D})$.

(a) Suppose that we have the boundary condition

$$u = g \text{ on } \partial D. \quad (12)$$

Equation (11) still holds. Since we know the value of u on ∂D , we can set $v = 0$ on ∂D . Then we get

$$\int_D f v = - \int_D c \nabla u \cdot \nabla v \quad (13)$$

for all $v \in H^1(\overline{D})$ such that $v = 0$ on ∂D , which is our weak formulation of (10) with the given boundary condition.

(b) Suppose that we have the boundary condition

$$\nabla u \cdot n + q u = p \text{ on } \partial D, \quad (14)$$

where n is the outward unit normal vector to ∂D , and p and q are functions on ∂D . Equation (11) still holds. Since $\nabla u \cdot n = p - q u$ on ∂D , it follows that

$$\int_D f v = \int_{\partial D} c v (p - q u) \, dS - \int_D c \nabla u \cdot \nabla v \quad (15)$$

for all $v \in H^1(\overline{D})$, which is our weak formulation of (10) with the given boundary condition.

Problem 3.

If $u \in C^2[a, b]$, then

$$\|u - I_h u\|_\infty \leq \frac{1}{8} h^2 \|u''\|_\infty, \quad (16)$$

$$\|(u - I_h u)'\|_\infty \leq \frac{1}{2} h \|u''\|_\infty. \quad (17)$$

Proof. Consider the interval $[x_i, x_{i+1}]$, where $1 \leq i \leq N$. Restricted to this interval, $I_h u$ is the degree-1 Lagrange polynomial interpolation of u on with nodes x_i and x_{i+1} . By the error formula for Lagrange polynomial approximation in the slides,

$$u(x) - I_h u(x) = \frac{f''(\xi(x))(x - x_i)(x - x_{i+1})}{2} \quad (18)$$

for some $\xi(x) \in [x_i, x_{i+1}]$. Then

$$|u(x) - I_h u(x)| \leq \|f''\|_\infty \cdot \frac{1}{2} (x - x_i)(x_{i+1} - x). \quad (19)$$

The function $g(x) = (x - x_i)(x_{i+1} - x)$ is a downward-opening parabola, so it achieves maximum halfway between its roots x_i and x_{i+1} . Therefore,

$$|u(x) - I_h u(x)| \leq \|f''\|_\infty \cdot \frac{\left(\frac{x_i + x_{i+1}}{2} - x_i\right) \left(x_{i+1} - \frac{x_i + x_{i+1}}{2}\right)}{2} \quad (20)$$

$$= \|f''\|_\infty \frac{(x_{i+1} - x_i)^2}{8} = \frac{h^2}{8} \|f''\|_\infty. \quad (21)$$

Since this holds for all $x \in [x_i, x_{i+1}]$ and all $1 \leq i \leq N$, it holds for all $x \in [a, b]$. Therefore, the inequality (16) follows.

Let $1 \leq i \leq N$, and let $x \in (x_i, x_{i+1})$. By Taylor's Theorem,

$$u(x_i) = u(x) + (x_i - x)u'(x) + \frac{1}{2}(x_i - x)^2 u''(\xi(x_i)) \quad (22)$$

$$u(x_{i+1}) = u(x) + (x_{i+1} - x)u'(x) + \frac{1}{2}(x_{i+1} - x)^2 u''(\xi(x_{i+1})) \quad (23)$$

for some $\xi(x_i), \xi(x_{i+1}) \in [x_i, x_{i+1}]$. Then

$$u(x_{i+1}) - u(x_i) = (x_{i+1} - x_i)u'(x) + \frac{1}{2}(x_{i+1} - x)^2 u''(\xi(x_{i+1})) - \frac{1}{2}(x_i - x)^2 u''(\xi(x_i)). \quad (24)$$

Since $I_h u(x) = \frac{u(x_{i+1}) - u(x_i)}{x_{i+1} - x_i}(x - x_i) + u(x_i)$ for $x \in (x_i, x_{i+1})$, it follows that $(I_h u)'(x) = \frac{u(x_{i+1}) - u(x_i)}{x_{i+1} - x_i}$ for $x \in (x_i, x_{i+1})$. Thus,

$$(u - I_h u)'(x) = u'(x) - (I_h u)'(x) = u'(x) - \frac{u(x_{i+1}) - u(x_i)}{x_{i+1} - x_i} \quad (25)$$

$$= \frac{(x_i - x)^2}{2(x_{i+1} - x_i)} u''(\xi(x_i)) - \frac{(x_{i+1} - x)^2}{2(x_{i+1} - x_i)} u''(\xi(x_{i+1})) \quad (26)$$

if $x \in (x_i, x_{i+1})$. Taking absolute values on both sides gives

$$|(u - I_h u)'(x)| \leq \frac{1}{2(x_{i+1} - x_i)} [(x_i - x)^2 |u''(\xi(x_i))| + (x_{i+1} - x)^2 |u''(\xi(x_{i+1}))|] \quad (27)$$

$$\leq \frac{1}{2h} [(x_i - x)^2 + (x_{i+1} - x)^2] \|u''\|_\infty \quad (28)$$

$$= \frac{1}{2h} g(x) \|u''\|_\infty, \quad (29)$$

where $g(x) = (x_i - x)^2 + (x_{i+1} - x)^2$. We note that $g'(x) = 4x - 2(x_{i+1} + x_i)$, so g achieves a maximum on $[x_i, x_{i+1}]$ when $g'(x) = 0$, that is, when $x = \frac{x_{i+1} + x_i}{2}$, or else when $x \in \{x_i, x_{i+1}\}$, by the Extreme Value Theorem. If $x \in \{x_i, x_{i+1}\}$, then $g(x) = h^2$, and if $x = \frac{x_i + x_{i+1}}{2}$, then $g(x) = \frac{h^2}{2}$. Therefore, the maximum of g on $[x_i, x_{i+1}]$ is h^2 , and

$$|(u - I_h u)'(x)| \leq \frac{h}{2} \|u''\|_\infty \quad (30)$$

if $x \in (x_i, x_{i+1})$. Since i was arbitrary, this inequality holds for all $x \in [a, b]$ except at the nodes $\{x_i\}$ where $I_h u$ is potentially not differentiable. The L^∞ norm $\|\cdot\|_\infty$ does not depend on the value of a function at finitely many points, so it follows that

$$\|(u - I_h u)'\|_\infty \leq \frac{1}{2} h \|f''\|_\infty, \quad (31)$$

as desired. \square

Problem 4.

Consider the weak formulation of

$$\nabla \cdot (c \nabla u) = f \text{ in } D, \quad u = g \text{ on } \partial D \quad (32)$$

derived in problem 2 (a):

$$\int_D f v = - \int_D c \nabla u \cdot \nabla v \quad (33)$$

for all $v \in H^1(\overline{D})$ such that $v = 0$ on ∂D . Suppose that we have basis functions $\{\phi_i\}_{i=1}^{N+1}$ for a finite element space U_h on \overline{D} . To approximate a solution of the weak formulation, we approximate H^1 by U_h . Thus, we want to find $u \in U_h$ such that (33) holds for all $v \in U_h$.

By the linearity of the problem and the fact that $U_h = \text{span}\{\phi_i\}$, this is equivalent to (33) being true for $v = \phi_i$, for $i = 1, \dots, N+1$. Since we want $u \in U_h$, there exist coefficients u_j such that

$$u = \sum_{j=1}^{N+1} u_j \phi_j. \quad (34)$$

Hence, we need

$$\int_D f \phi_i = - \int_D c \nabla \left(\sum_{j=1}^{N+1} u_j \phi_j \right) \cdot \nabla \phi_i \quad (35)$$

for all $i = 1, \dots, N+1$. Using the linearity of ∇ and rearranging terms, this is equivalent to

$$\sum_{j=1}^{N+1} u_j \left[- \int_D c \nabla \phi_j \cdot \nabla \phi_i \right] = \int_D f \phi_i \quad (36)$$

for all $i = 1, \dots, N+1$. If we set

$$A_{ij} = - \int_D c \nabla \phi_j \cdot \nabla \phi_i, \quad b_i = \int_D f \phi_i, \quad X_j = u_j, \quad (37)$$

then this is equivalent to the linear system $AX = b$.

Problem 5.

Let A be a nonsingular, lower-triangular matrix; that is, $i < j$ implies that $A_{ij} = 0$. Then A^{-1} is also lower-triangular.

Proof. We use induction on the size of the matrix. All 1×1 matrices are trivially lower-triangular, so the base case holds. Now suppose that the claim is true for all matrices of size $n \times n$, where $n \geq 1$.

Let A be a nonsingular, $(n+1) \times (n+1)$, lower-triangular matrix. Then every entry but the last entry of the last column of A is zero by the lower-triangular condition. That is, we can write A in block matrix form as

$$A = \begin{bmatrix} B & 0 \\ c & d \end{bmatrix}, \quad (38)$$

where B is a $n \times n$ matrix, c is a $1 \times n$ row vector, and d is a scalar. Since $A_{ij} = B_{ij}$ if $i, j \leq n$, it follows that B is also lower-triangular. Furthermore, B must be nonsingular.

Indeed, suppose for the sake of contradiction that B is singular. Then its rows $\{B_1, \dots, B_n\}$ are linearly dependent. That is, there exist $\alpha_1, \dots, \alpha_n$ not all zero such that

$$\alpha_1 B_1 + \dots + \alpha_n B_n = 0. \quad (39)$$

Let $\{A_1, \dots, A_n, A_{n+1}\}$ denote the rows of A . Then $A_i = [B_i \ 0]$ for $1 \leq i \leq n$. Hence,

$$\alpha_1 A_1 + \dots + \alpha_n A_n = 0 \quad (40)$$

as well. This implies that the rows of A are linearly dependent, which contradicts the nonsingularity of A .

Therefore, B is a nonsingular, $n \times n$, lower-triangular matrix, and the induction hypothesis implies that B^{-1} is lower-triangular.

In addition, $d \neq 0$ because $d = 0$ implies that $\det(A) = 0$ upon expansion by cofactors on the last column of A , which contradicts the nonsingularity of A .

We now observe that

$$A \begin{bmatrix} B^{-1} & 0 \\ -cB^{-1}d^{-1} & d^{-1} \end{bmatrix} = \begin{bmatrix} B & 0 \\ c & d \end{bmatrix} \begin{bmatrix} B^{-1} & 0 \\ -cB^{-1}d^{-1} & d^{-1} \end{bmatrix} = \begin{bmatrix} I_{n \times n} & 0 \\ cB^{-1} - cB^{-1}d^{-1}d & 1 \end{bmatrix} = I_{(n+1) \times (n+1)}, \quad (41)$$

so

$$A^{-1} = \begin{bmatrix} B^{-1} & 0 \\ -cB^{-1}d^{-1} & d^{-1} \end{bmatrix}. \quad (42)$$

Then A^{-1} is lower-triangular because B^{-1} is lower triangular. Hence, the inverse of any nonsingular, lower-triangular matrix is also lower-triangular by induction. \square

Problem 6.

Let

$$A = \begin{bmatrix} \kappa & \lambda \\ \lambda & \mu \end{bmatrix} \quad (43)$$

be a positive definite matrix. Then the Jacobi method for $Ax = b$ converges.

Proof. We recall from the slides that the Jacobi method is the iteration

$$x^{(k+1)} = -D^{-1}Nx^{(k)} + D^{-1}b, \quad (44)$$

where D is the diagonal of A , and N is the off-diagonal of A . This iteration converges if and only if $\rho(-D^{-1}N) < 1$. In this case,

$$-D^{-1}N = - \begin{bmatrix} 0 & \frac{\lambda}{\mu} \\ \frac{\lambda}{\kappa} & 0 \end{bmatrix}, \quad (45)$$

so any eigenvalue ρ of $-D^{-1}N$ satisfies $\rho^2 - \frac{\lambda^2}{\kappa\mu} = 0$. Therefore $|\rho| < 1$ if and only if $\lambda^2 < \kappa\mu$, or $\kappa\mu - \lambda^2 > 0$. Since $\kappa\mu - \lambda^2 = \det(A)$, and the positive definiteness of A implies that $\det(A) > 0$, it follows that $\rho(-D^{-1}N) < 1$, and the Jacobi method converges. \square

Problem 7.

(a)

(b)

Input: A symmetric, positive-definite, $n \times n$ matrix A

Input: A symmetric, positive-definite, $n \times n$ matrix M that is easy to invert (the preconditioner)

Input: A vector b of length n

Input: Initial guess $x^{(0)}$ for the solution of $Ax = b$

Input: Residual tolerance $\varepsilon > 0$

Output: Approximate solution x of $Ax = b$

// Initialization

1 $r^{(0)} \leftarrow b - Ax^{(0)};$

2 $d^{(0)} \leftarrow M^{-1}r^{(0)};$

3 $k \leftarrow 0;$

// Iteration

4 **while** $\|r^{(k)}\| \geq \varepsilon$ **do**

 // Update $x^{(k)}$

5 $\alpha^{(k)} \leftarrow \frac{(r^{(k)})^T M^{-1} r^{(k)}}{(d^{(k)})^T A d^{(k)}};$

6 $x^{(k+1)} \leftarrow x^{(k)} + \alpha^{(k)} d^{(k)};$

 // Update search direction and residual

7 $r^{(k+1)} \leftarrow r^{(k)} - \alpha^{(k)} A d^{(k)};$

8 $\beta^{(k+1)} \leftarrow \frac{(r^{(k+1)})^T M^{-1} r^{(k+1)}}{(r^{(k)})^T M^{-1} r^{(k)}};$

9 $d^{(k+1)} \leftarrow M^{-1} r^{(k+1)} + \beta^{(k+1)} d^{(k)};$

10 **end**

Problem 8.

- (a) The main implementation of the finite element method for our second-order BVP is found in `fem_dirichlet_1d.m`, and copied below. In order to avoid code duplication, this function takes the various data determining the equation as parameters, namely, a , b , g_a , g_b , f , and c , as well as the number of points in the mesh N . Most importantly, the method used for numerical integration and the solving of a linear system are passed as parameters as well.

```

1 function [v_x, v_u] = fem_dirichlet_1d(a, b, ga, gb, c, f, n, integrator, solver)
2
3 % Initialization
4 h = (b - a) / n;
5
6 v_x = linspace(a, b, n + 1);
7
8 m_A = sparse(n + 1, n + 1);
9 v_b = zeros(n + 1, 1);
10
11 % Compute integral of c(x) over each element
12 v_c_int = zeros(n, 1);
13 for j = 1:n
14     v_c_int(j) = integrator(c, v_x(j), v_x(j + 1)) / h^2;
15 end
16
17 % Assemble stiffness matrix (no BC)
18 for j = 1:n

```

```

19     m_A(j + 1, j) = v_c_int(j);
20     m_A(j, j + 1) = v_c_int(j);
21 end
22
23 for j = 2:n
24     m_A(j, j) = -v_c_int(j - 1) - v_c_int(j);
25 end
26
27 m_A(1, 1) = -v_c_int(1);
28 m_A(n + 1, n + 1) = -v_c_int(n);
29
30 % Assemble load vector (no BC)
31 for j = 2:n
32     v_b(j) = integrator(@(x) f(x) .* (x - v_x(j - 1)) / h, v_x(j - 1), v_x(j));
33     v_b(j) = v_b(j) + integrator(@(x) f(x) .* (v_x(j + 1) - x) / h, v_x(j), v_x(j + 1));
34 end
35
36 v_b(1) = integrator(@(x) f(x) .* (x - v_x(1)) / h, v_x(1), v_x(2));
37 v_b(n + 1) = integrator(@(x) f(x) .* (v_x(n + 1) - x) / h, v_x(n), v_x(n + 1));
38
39 % Enforce Dirichlet BC
40 m_A(1, :) = 0;
41 m_A(1, 1) = 1;
42 m_A(n + 1, :) = 0;
43 m_A(n + 1, n + 1) = 1;
44
45 v_b(1) = ga;
46 v_b(n + 1) = gb;
47
48 % Solve
49 v_u = solver(m_A, v_b)';

```

To apply 4-point Gaussian quadrature, we reuse `quad.m` from Homework 6 (renamed to `myquad.m`). Note that the Gauss-Legendre quadrature nodes and weights on $[-1, 1]$ are given by

$$x_1 = \dots \quad (46)$$

We implement the use of these nodes and weights in `gauss4_integrator.m`, copied below. Using the MATLAB `\` operator in the `solver` parameter is trivial, so we don't need a custom function for it.

```

1 function result = gauss4_integrator(f, a, b)
2
3 nodes = [
4     sqrt(3/7 - 2/7 * sqrt(6/5)), ...
5     -sqrt(3/7 - 2/7 * sqrt(6/5)), ...
6     sqrt(3/7 + 2/7 * sqrt(6/5)), ...
7     sqrt(3/7 + 2/7 * sqrt(6/5))
8 ];
9
10 weights = [
11     (18 + sqrt(30)) / 36, ...
12     (18 + sqrt(30)) / 36, ...
13     (18 - sqrt(30)) / 36, ...
14     (18 - sqrt(30)) / 36
15 ];

```

```
16  
17 local_nodes = nodes * ((b - a) / 2) + (a + b) / 2;  
18 local_weights = weights * ((b - a) / 2);  
19  
20 result = myquad(f, local_nodes, local_weights);
```