TMA373 Assignment 4

Edoardo Mangia
edoardom@student.chalmers.se

Bibiana Farinha
bibiana@student.chalmers.se

February 2025

Contents

1 Task 1

a)

In the this task, we will implement the FEM for the stationary convection-diffusion problem

$$\begin{aligned} -Du''(x) + \frac{1}{2}u'(x) =& 1, \quad 0 < x < \pi \\ u(0) = u(\pi) =& 0 \end{aligned} \tag{1}$$

The first step of FEM is to find the Variational Form of the problem. For that purpose, consider the space

$$H_0^1(0,\pi) = \{u \in H^1(0,\pi) \mid u(0) = u(\pi) = 0\}$$

Let $v \in H_0^1(0,\pi)$, and calculate the integral of 1 multiplied by v. Since $v \in H_0^1(0,\pi)$, v and its first derivative are square-integrable.

$$\int_0^\pi v(x)dx = \int_0^\pi \frac{1}{2}v(x)u'(x)dx - \int_0^\pi Dv(x)u''(x)dx$$

The second term of the right hand side has a second-order derivative term u''. To reduce the order, we integrate by parts, yielding:

$$\int_0^\pi v(x)dx = \int_0^\pi \frac{1}{2}v(x)u'(x)dx + \int_0^\pi Dv'(x)u'(x)dx - \Big[Dv(x)u'(x)\Big]_0^\pi$$

From the definition of $H_0^1(0,\pi)$, the values of $v(0)$ and $v(\pi)$ are known, so the expression above can be simplified further:

$$\int_0^\pi v(x)dx = \int_0^\pi \frac{1}{2}v(x)u'(x)dx + \int_0^\pi Dv'(x)u'(x)dx \tag{2}$$

The VF reads: Find $u \in H_0^1(0,\pi)$, such that 2 holds for all $v \in H_0^1(0,\pi)$.

The next step of FEM is the formulation of the Finite Elements problem. Consider the space $V_h^0 = span(\varphi_1, \varphi_2, \ldots, \varphi_m)$, where $\{\varphi_j\}_{j=1}^m$ are hat functions defined $T_h : 0 = x_0 < x_1 < \cdots < x_{m+1} = 1$.

The FE reads: Find $u_h \in V_h^0$ such that 3 and 4 hold for all $v_h \in V_h^0$.

$$\int_0^\pi v_h(x)dx = \int_0^\pi \frac{1}{2}v_h(x)u_h'(x)dx + \int_0^\pi Dv_h'(x)u_h'(x)dx \tag{3}$$

$$u_h(0) = u_h(\pi) = 0 \tag{4}$$

Finally, the last step of FEM is finding the linear system of equations. To do this, set $u_h(x) = \sum_j^m \xi_j\varphi_j(x)$, where ξ_j is unknown and $\varphi_j(x)$ is the basis, and take $v_h = \varphi_i$ for $i = 1 \ldots m$. Now we insert this in the FE problem and get:

$$\int_0^\pi \varphi_i(x)dx = \int_0^\pi \frac{1}{2}\varphi_i(x)\sum_{j=1}^m \xi_j\varphi_j'(x)dx + \int_0^\pi D\varphi_i'(x)\sum_{j=1}^m \xi_j\varphi_j'(x)dx$$

Using linearity of integration, we move the summation and constants outside:

$$\int_0^\pi \varphi_i(x)dx = \frac{1}{2}\sum_{j=1}^m \xi_j \int_0^\pi \varphi_i(x)\varphi_j'(x)dx + D\sum_{j=1}^m \xi_j \int_0^\pi \varphi_i'(x)\varphi_j'(x)dx$$

Now, we can define the convection and stiffness matrices, and the load vector b:

$$C_{ij} = \int_0^\pi \varphi_i(x)\varphi_j'(x)dx, \quad S_{ij} = \int_0^\pi \varphi_i'(x)\varphi_j'(x)dx, \quad b_i = \int_0^\pi \varphi_i(x)dx$$

Thus, the system simplifies to:

$$\sum_{j=1}^m \xi_j \left(\frac{1}{2}C_{ij} + DS_{ij}\right) = b_i, \quad \forall i = 1, \ldots, m$$

Or, in matrix form:

$$A\xi = b, \text{ where } A = DS + \frac{1}{2}C$$

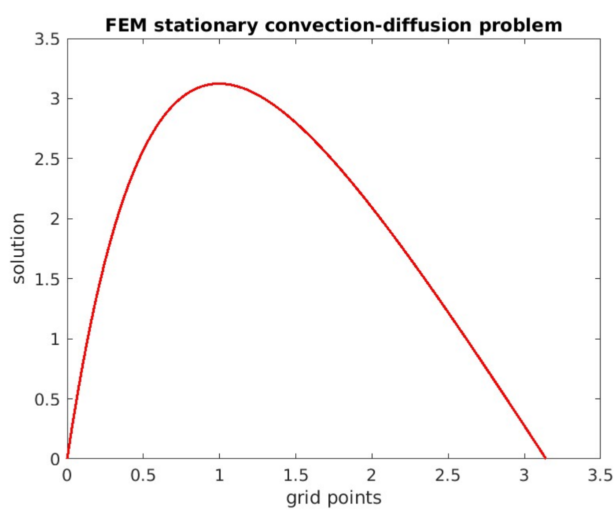
The values of the two matrices and the load vector follow:

$$S = \frac{1}{h}\begin{bmatrix} 2 & -1 & 0 & \ldots \\ -1 & 2 & -1 & \ldots \\ 0 & -1 & 2 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad C = \frac{1}{2}\begin{bmatrix} 0 & -1 & 0 & \ldots \\ 1 & 0 & -1 & \ldots \\ 0 & 1 & 0 & \ldots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad b = h\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}.$$

Implementation

```
function myfem(D, m)
h = pi/(m+1);
xv=0:h:pi;

S = zeros(m,m);
C = zeros(m,m);
b = zeros(m,1);
for j=1 : m
  S(j,j) = 2/h;
  C(j,j) = 0;
  b(j) = h;
  if j < m
      S(j+1, j) = -1/h;
      S(j, j+1) = -1/h;
      C(j+1, j) = -1/2;
      C(j, j+1) = 1/2;
  end
end

A = S*D+(1/2)*C ;
xi = A\b ;
xi =  [0; xi; 0];
```

Visualization



After implementing FEM for the stationary convection-diffusion problem, we ran the function with the arguments $D = 0.3$ and $m = 100$, the output is the visualization above.

b)

In this exercise we will study the influence that the value of D has on choosing a nice mesh size h. We will determine the exact solution, choose two values for D, one for each case in the instructions, and plot the previous FEM function along with the exact solution.

The exact solution of 1 follows:

$$u(x) = \frac{2\left(\left(e^{\pi/2D} - 1\right)x - \pi e^{x/2D} + \pi\right)}{e^{\pi/2D} - 1}$$

For case 1, we have chosen $D = 1$, and mesh sizes $h = 5$ and $h = 15$.

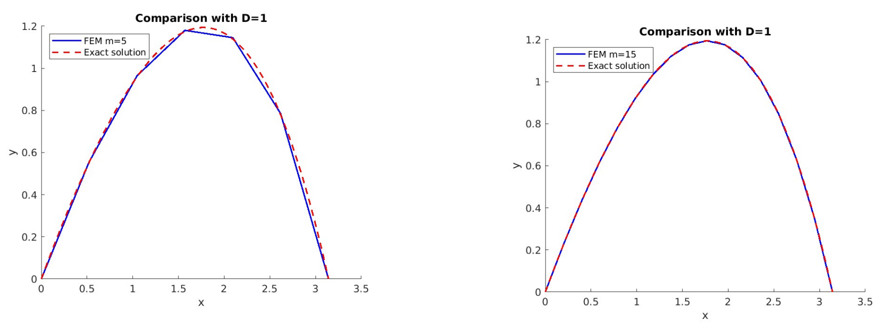
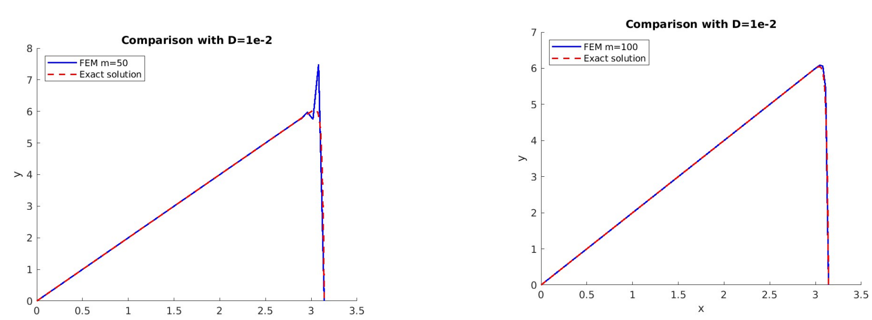


Figure 1: First figure

The figures above illustrate that, when $D \approx 1$, we don't need a lot of discritization points to have a good approximation. This is because the original function does not have any sharp angles.

For case 2, we have chosen $D = 1e-2$, and mesh sizes $h = 50$ and $h = 100$.



The figures above illustrate that, when $D << 1$, we do need a lot of discritization points to have a good approximation. As D grows smaller, the function is follows the linear function $2x$ until around the point $(3, 6)$, where it takes a sharp turn towards $y = 0$.

2 Task 2

Now, we'll try to apply cG(1) approximation (linear elements) and cG(2) approximation (with quadratic elements).

The boundary problem in consideration now is:

$$-u''(x) = f(x), \quad x \in (0, 1), \qquad u(0) = u(1) = 0.$$

We'll test the methods with the function:

$$f(x) = \pi^2 \sin(\pi x),$$

For which we know the exact solution is:

$$u(x) = \sin(\pi x).$$

a)

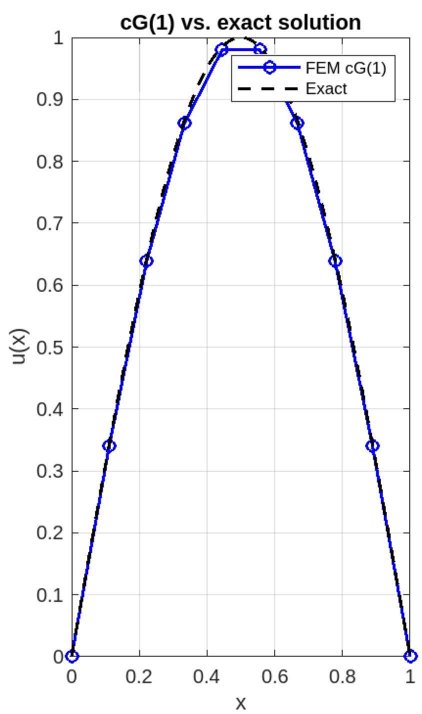
With piecewise linear elements, for the standard assembly, we'll be using the local stiffness matrix $\frac{1}{h}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ and approximating the load vector via quadrature (here, with midpoint rule).

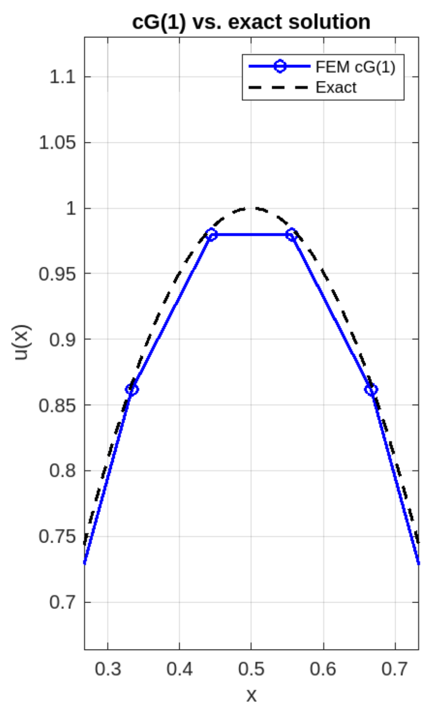
Implementation

```
a = 0; b = 1;
m = 8;
h = (b - a)/(m+1);
x_lin = linspace(a, b, m+2)';
N_lin = m;

I_lin = []; J_lin = []; S_lin = [];
b_lin = zeros(N_lin,1);

for e = 1:(m+1)
    nodes = [e, e+1];
    x_e = x_lin(nodes);
    A_loc = 1/h * [1 -1; -1 1];
    % Midpoint quadrature for the load
    xm = mean(x_e);
    phi = [(x_e(2)-xm)/h, (xm-x_e(1))/h];
    b_loc = pi^2*sin(pi*xm)*h * [phi(1); phi(2)];

    for i_local = 1:2
        I_global = nodes(i_local);
        if I_global == 1 || I_global == m+2, continue; end
        row = I_global - 1;
```

```
        b_lin(row) = b_lin(row) + b_loc(i_local);
        for j_local = 1:2
            J_global = nodes(j_local);
            if J_global == 1 || J_global == m+2, continue; end
            col = J_global - 1;
            I_lin(end+1,1) = row;
            J_lin(end+1,1) = col;
            S_lin(end+1,1) = A_loc(i_local,j_local);
        end
    end
end

A_lin = sparse(I_lin, J_lin, S_lin, N_lin, N_lin);
uh_lin = A_lin\b_lin;
u_lin = [0; uh_lin; 0];
```

Visualization





b)

With quadratic elements, each element hs 3 nodes.

We define the quadratic shape functions on the reference element $[-1, 1]$:

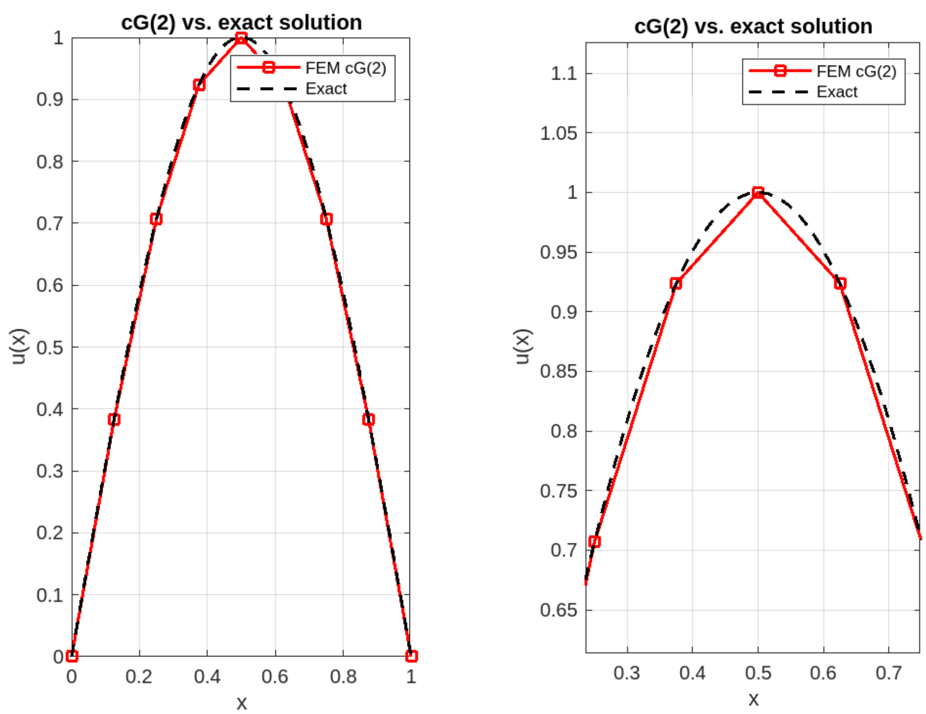
$$\phi_1(r) = \frac{r(r-1)}{2}, \quad \phi_2(r) = 1 - r^2, \quad \phi_3(r) = \frac{r(r+1)}{2}.$$

The mapping to the physical element and the use of a 3-point Gauss-Legendre quadrature allows us to compute both the local stiffness matrix and load vector.

Implementation

```
a = 0; b = 1;
m_el = 4;  % number of quadratic elements
x_quad = linspace(a, b, 2*m_el+1)';
N_quad = 2*m_el-1;  % number of interior unknowns
h_quad = (b - a)/m_el;

% 3-point Gauss-Legendre quadrature on [-1,1]
r_gauss = [-sqrt(3/5); 0; sqrt(3/5)];
w_gauss = [5/9; 8/9; 5/9];

I_quad = []; J_quad = []; S_quad = [];
b_quad = zeros(N_quad,1);

for e = 1:m_el
    nodes = [2*e-1, 2*e, 2*e+1];
    x_e = x_quad(nodes);
    A_loc = zeros(3,3);
    b_loc = zeros(3,1);
    for q = 1:length(r_gauss)
        r = r_gauss(q);
        w = w_gauss(q);
        % Quadratic shape functions and derivatives on [-1,1]
        phi = [ r*(r-1)/2; 1-r^2; r*(r+1)/2 ];
        dphi_dr = [ (2*r-1)/2; -2*r; (2*r+1)/2 ];
        dphi_dx = (2/h_quad)*dphi_dr;
        x_val = (x_e(1)+x_e(3))/2 + (h_quad/2)*r;
        A_loc = A_loc + (dphi_dx*dphi_dx')*(h_quad/2*w);
        b_loc = b_loc + phi * (pi^2*sin(pi*x_val))*(h_quad/2*w);
    end

    for i_local = 1:3
        I_global = nodes(i_local);
        if I_global == 1 || I_global == 2*m_el+1, continue; end
        row = I_global - 1;
        b_quad(row) = b_quad(row) + b_loc(i_local);
        for j_local = 1:3
            J_global = nodes(j_local);
            if J_global == 1 || J_global == 2*m_el+1, continue; end
            col = J_global - 1;
            I_quad(end+1,1) = row;
            J_quad(end+1,1) = col;
            S_quad(end+1,1) = A_loc(i_local,j_local);
        end
    end
end

A_quad = sparse(I_quad, J_quad, S_quad, N_quad, N_quad);
uh_quad = A_quad\b_quad;
u_quad = [0; uh_quad; 0];
```

Visualization




```
xx = linspace(a, b, 100);
u_exact = sin(pi*xx);

figure;
subplot(1,2,1)
plot(x_lin, u_lin, 'bo-', 'LineWidth',1.5); hold on;
plot(xx, u_exact, 'k--','LineWidth',1.5);
xlabel('x'); ylabel('u(x)');
title('cG(1) vs. Exact Solution');
legend('FEM cG(1)','Exact'); grid on;

subplot(1,2,2)
plot(x_quad, u_quad, 'rs-', 'LineWidth',1.5); hold on;
plot(xx, u_exact, 'k--','LineWidth',1.5);
xlabel('x'); ylabel('u(x)');
title('cG(2) vs. Exact Solution');
legend('FEM cG(2)','Exact'); grid on;
```

For the convergence study of the $cG(1)$ method, a loop over several mesh sizes is implemented, and the error (measured in the maximum norm) is plotted on a log-log scale to verify the expected order of convergence.

Implementation

```
a = 0; b = 1;
u_exact = @(x) sin(pi*x);
err = zeros(1,5);

for l = 1:5
    m = 2^l;          % number of interior subintervals; total nodes = m+2
    h = (b - a) / (m + 1);
    x_nodes = linspace(a, b, m+2)';  % nodes including boundaries
    N = m;

    A = sparse(N, N);
    bvec = zeros(N, 1);

    for e = 1:(m+1)
        localNodes = [e, e+1];  % indices of nodes on current element
        x_e = x_nodes(localNodes);

        % Local stiffness matrix for linear elements
        A_loc = (1/h) * [1, -1; -1, 1];

        phi1 = @(x) (x_e(2) - x) / h;
        phi2 = @(x) (x - x_e(1)) / h;

        f = @(x) pi^2 * sin(pi*x);

        % Compute local load vector b_local using exact integration
        b_local = zeros(2,1);
        b_local(1) = integral(@(x) f(x) .* phi1(x), x_e(1), x_e(2));
        b_local(2) = integral(@(x) f(x) .* phi2(x), x_e(1), x_e(2));

        % Assemble local contributions into the global system
        for i_local = 1:2
            global_i = localNodes(i_local);

            if global_i == 1 || global_i == m+2
                continue;
            end
            row = global_i - 1;  % shift index because first node is boundary
            bvec(row) = bvec(row) + b_local(i_local);
            for j_local = 1:2
                global_j = localNodes(j_local);
                if global_j == 1 || global_j == m+2
                    continue;
                end
                col = global_j - 1;
                A(row, col) = A(row, col) + A_loc(i_local, j_local);
            end
        end
    end

    % Solve the linear system for interior nodes
```

```
    U_interior = A \ bvec;
    U = [0; U_interior; 0];  % add boundary values u(0)=0 and
        u(1)=0

    xx = linspace(a, b, 10 * m);
    UU = interp1(x_nodes, U, xx, 'linear');

    uu = u_exact(xx);

    % Compute the error in the maximum norm
    err(l) = norm(UU - uu, Inf);
end

hl = (b - a) ./ ((2.^(1:5) + 1));
figure;
loglog(hl, err, 'b*-','LineWidth',1.5); hold on;
loglog(hl, hl, 'b.-', hl, hl.^2, 'r--','LineWidth',1.5);
xlabel('h'); ylabel('Error');
legend('Error','O(h)','O(h^2)','Location','southwest');
title('Convergence of cG(1) Method');
grid on;
```

Visualization