FEM4CFD Notes

Bibek Yonzan

2025

Contents

1 1D Advection Diffusion Equation

1.1 Governing Equation

The governing equation of the advection diffusion reaction is of type (1). The equation models the transport phenomena including all three advection, diffusion and reaction. In the equation (2), **F** and **G** represent the advection and diffusion coefficients, while the term **Q** represents either a reaction or source term. The advection diffusion equation is of the type

$$\frac{\partial \Phi}{\partial t} + \frac{\partial \boldsymbol{F_i}}{\partial x_i} + \frac{\partial \boldsymbol{G_i}}{\partial x_i} + \boldsymbol{Q} = 0 \tag{1}$$

$$\begin{aligned} \boldsymbol{F_i} &= \boldsymbol{F_i}(\boldsymbol{\Phi}) \\ \boldsymbol{G_i} &= \boldsymbol{G_i}\frac{\partial \boldsymbol{\Phi}}{\partial x_j} \\ \boldsymbol{Q} &= \boldsymbol{Q}(x_i, \boldsymbol{\Phi}) \end{aligned} \tag{2}$$

where in general, Φ is a basic dependent vector variable. A linear reaction between the source term and the scalar variable is referred to as the reaction term. In (2), x_i and i refer to Cartesian coordinates and the associated quantities and as a whole.
Thus, the equation in scalar terms becomes

$$\begin{aligned} \Phi &\to \phi, & \boldsymbol{Q} &\to Q(x_i, \phi) = s\phi \\ \boldsymbol{F_i} &\to F_i = a\phi, & \boldsymbol{G_i} &\to G_i = -k\frac{\partial \phi}{\partial x} \end{aligned} \tag{3}$$

$$\frac{\partial \phi}{\partial t} + \frac{\partial (a\phi)}{\partial x_i} - \frac{\partial}{\partial x_i}\left(k\frac{\partial \phi}{\partial x_i}\right) + Q = 0 \tag{4}$$

Here, U is the velocity field and ϕ is the scalar quantity being transported by this velocity. But, diffusion can also occur, and k is the diffusion coefficient.
A linear reaction term can be written associated, where c is a scalar parameter.

$$Q = c\,\phi$$

Here, a is the velocity field and ϕ is the scalar quantity being transported by this velocity. But, diffusion can also occur, and k is the diffusion coefficient. A linear reaction term can be associated, where c is a scalar parameter. The equation represented by (4) is the strong form or the differential form of the advection diffusion governing equation. For the steady state solution, the first term becomes zero, leaving only the advection, diffusion and reaction terms.

1.2 Weak Form

The use of 4 requires computation of second derivatives to solve the problem, as such a *weakened* form can be considered by solving the equation over a domain Ω using an integral, like

$$\int_\Omega w\left(a\cdot\frac{\partial\phi}{\partial x}\right)\mathrm{d}\Omega - \int_\Omega w\frac{\mathrm{d}}{\mathrm{d}x}\left(k\frac{\mathrm{d}\phi}{\mathrm{d}x}\right) + \int_\Omega wQ = 0 \tag{5}$$

where w is an arbitrary weighting function, chosen such that $w = 0$ on Dirichlet boundary condition, Γ_D. Also at the same Dirichlet boundary condition, the variable $\phi = \phi_D$. Assuming a source term, s is present, the right hand side of the above equation changes resulting in the weak form the governing equation.

$$\int_\Omega w\left(a\cdot\nabla\phi\right)\mathrm{d}\Omega - \int_\Omega w\nabla\cdot\left(k\nabla\phi\right)\mathrm{d}\Omega + \int_\Omega wQ\mathrm{d}\Omega = \int_\Omega ws\,\mathrm{d}\Omega \tag{6}$$

Noting that, $w = 0$ on Γ_D, using divergence theore, we get

$$\int_\Omega w\left(a\cdot\nabla\phi\right)\mathrm{d}\Omega - \int_\Omega \nabla w\cdot\left(k\nabla\phi\right)\mathrm{d}\Omega + \int_\Omega wQ\mathrm{d}\Omega = \int_\Omega ws\,\mathrm{d}\Omega + \int_{\Gamma_N} wh\,\mathrm{d}\Gamma \tag{7}$$

where Γ_N and h represent the Neumann boundary condition and the normal diffusive flux on the Neumann boundary condition.

1.3 Galerkin Approximation

The Galerkin approximation is a technique used to approximate numerical solutions of PDEs by replacing the infinite-dimensional spaces into finite dimensional spaces. The finite spaces are constructed using finite elements over a domain. Since spaces are finite-dimensional, the weighting function is a discrete weighting function w_h.

Using Galerkin approximation, ϕ can be written

$$\begin{aligned}\phi(x) &= N_1(x)\phi_1 + N_2(x)\phi_2 + ... + N_n(x)\phi_n \\ \phi\left(x\right) &= \sum_{i=1}^{n_{el}} N_i\left(x\right)\phi_i\end{aligned} \tag{8}$$

Here, N_i is the shape function or basis function at that node, n_{el} is the total number of elements and ϕ is the solution, also known as degrees of freedom (DOFs). For Galerkin approximation, the weighting function is equal to the shape function i.e., $w_i = N_i$. Now, equation 7 becomes:

$$\int_\Omega a\left(N_a \frac{\partial N_b}{\partial N_t}\right) \mathrm{d}\Omega + \int_\Omega k\left(\frac{\partial N_a}{\partial t}\frac{\partial N_b}{\partial t}\right) \mathrm{d}\Omega \int_\Omega Q\left(N_a \cdot N_b\right) \mathrm{d}\Omega = \int_\Omega f N_a \mathrm{d}\Omega + \int_\Gamma h N_a \mathrm{d}\Gamma \tag{9}$$

Here, in (9) f represents the source term and for problems with only Dirichlet boundary conditions, the second term on right hand vanishes, effectively giving us the weak form of the Galerkin approximatin of the ADR equation.
Solving the equation for cell Pe = 0.1, where Pe is defined by $Pe = \frac{a \cdot he}{2k}$ with 10 linear elements, with the domain length 1 and (0, 1) Dirichlet boundary conditions ont he left and the right of the domain respectively. The results of the comparison of the Galerkin solution with the analytical solution 11 without a source or a reaction term can be seen in Figure 1a.

1.3.1 Implementation

The following section explores the implementation of the Galerkin approximation for the one dimensional advection diffusion equation in MATLAB.

1. Initialization:

```
                %% 1D steady state advection diffusion
clear;
clc;
close all;

xL = 0;
xR = 1;
nelem = 10;

L = xR - xL;
he = L / nelem;

% boundary conditions
uL = 0;
uR = 1;

Pe = 0.1;
mu = 1;
c = Pe * (2 * mu) / he;
f = 0;

nGP = 2;
[gpts, gwts] = get_Gausspoints_1D(nGP);

nnode = nelem + 1;

```

```
ndof = 1;

totaldof = nnode * ndof;

node_coords = linspace(xL, xR, nnode);

elem_node_conn = [1:nelem; 2:nnode]';
elem_dof_conn = elem_node_conn;

dofs_full = 1:totaldof;
dofs_fixed = [1, totaldof];
dofs_free = setdiff(dofs_full, dofs_fixed);

% solution array
soln_full = zeros(totaldof, 1);
```

The first section of the code is dedicated for the initial boundary values, domain properties and initializing the solution arrays for calculation. he is the elemental length, while L is the length of the domain. xL and xR are the left and the right boundaries of the domain and $nelem$ is the number of elements the domain will be discretized into. $totaldof$ is the total degrees of freedom of the entire system and $soln_full$ is the final solution array i.e., $\phi(x)$. This solution utilizes Gaussian points for numerical integration over an element. For the purpose of this solution, two Gausspoints (nGP) are considered (linear element) with xi and wt being $\pm\frac{1}{\sqrt(3)}$ and 1.0, respectively.

2. Processing:

```
%% Processing
for iter = 1:9

    Kglobal_g = zeros(totaldof, totaldof);
    Fglobal_g = zeros(totaldof, 1);

    for elnum = 1:nelem
        elem_dofs = elem_dof_conn(elnum, :);
        Klocal = zeros(2, 2);
        Flocal = zeros(2, 1);

        %% Galerkin Approximation
        [Klocal, Flocal] = galerkinApproximation(c, mu, he, s,
           nGP, gpts, gwts, elem_dofs, node_coords, soln_full,
           Klocal, Flocal);

        Kglobal_g(elem_dofs, elem_dofs) = Kglobal_g(elem_dofs,
           elem_dofs) + Klocal;
        Fglobal_g(elem_dofs, 1) = Fglobal_g(elem_dofs, 1) +
           Flocal;
    end
```

```

    Fglobal_g = forceVector(Kglobal_g, Fglobal_g, iter, uL, uR,
       totaldof);

    rNorm = norm(Fglobal_g);

    if (rNorm < 1.0e-10)
        break;
    end

    Kglobal_g = stiffnessMatrix(Kglobal_g, totaldof);

    soln_incr = Kglobal_g \ Fglobal_g;
    soln_full = soln_full + soln_incr;

end
```

This solution uses an iterative solver to solve the advection diffusion equation. The solver iteratves over the number of elements and calls the function *galerkinApproximation* on each iteration to calculate the terms of the global stiffness matrix. The global stiffness matrix is a combination of the advection, diffusion and reaction contribution to the global matrix. These are:

$$K_{ad} = \frac{a}{2}\begin{bmatrix}-1 & 1\\ -1 & 1\end{bmatrix}, K_{diff} = \frac{k}{he}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}, K_{re} = \frac{s \cdot he}{6}\begin{bmatrix}2 & 1\\ 1 & 2\end{bmatrix} \tag{10}$$

```
%% Galerkin Approximation function
function [Klocal, Flocal] = galerkinApproximation(a, mu, h, s,
   nGP, gpts, gwts, elem_dofs, node_coords, soln_full, Klocal,
   Flocal)
    Klocal_g = zeros(2, 2);
    Flocal_g = zeror(2, 1);

    n1 = elem_dofs(n1);
    n2 = elem_dofs(n2);

    u1 = soln_full(n1);
    u2 = soln_full(n2);
    u = [u1 u2];
    for gp = 1:nGP
            xi = gpts(gp);
            wt = gwts(gp);
            N = [0.5 * (1 - xi), 0.5 * (1 + xi)];
            dNdxi = [-0.5, 0.5];
            Jac = h / 2;
            dNdx = dNdxi / Jac;
            du = dNdx * u';
            x = N * [x1 x2];

```

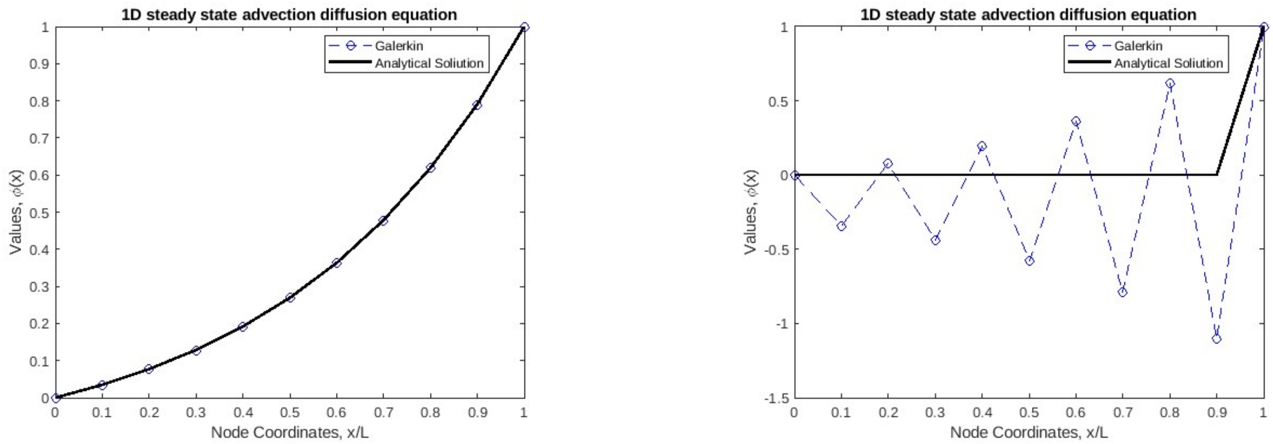
```
            % advection
            Klocal = Klocal + (a * N' * dNdx) * Jac * wt;
            % reaction
            Klocal = Klocal + (s * N' * N) * Jac * wt;
            % diffusion
            Klocal = Klocal + (mu * dNdx' * dNdx) * Jac * wt;

            % force vector
            Flocal = Flocal + N' * f * Jac * wt;
    end

            Klocal_g = Klocal_g + Klocal;
            Flocal_g = Flocal_g + Flocal;
end
```

```
%% Force vector assembly
function Fglobal = forceVector(Kglobal, Fglobal, iter, uL, uR, totaldof)

       if iter == 1
           Fglobal = Fglobal - Kglobal(:, 1) * uL;
           Fglobal = Fglobal - Kglobal(:, totaldof) * uR;
           Fglobal(1, 1) = uL;
           Fglobal(end, 1) = uR;
       else
           Fglobal(1, 1) = 0.0;
           Fglobal(end, 1) = 0.0;
       end

end
```

```
%% Stiffness Matrix Assembly
function Kglobal = stiffnessMatrix(Kglobal, totaldof)
   Kglobal(1, :) = zeros(totaldof, 1);
   Kglobal(:, 1) = zeros(totaldof, 1);
   Kglobal(1, 1) = 1.0;

   Kglobal(end, :) = zeros(totaldof, 1);
   Kglobal(:, end) = zeros(totaldof, 1);
   Kglobal(end, end) = 1.0;
end
```

It can be seen the Galerkin approximation for the one dimensional advection diffusion equation closely follows the analytical solution. However, at higher Peclet numbes, or due to a finer mesh, spurious oscillations are introduced, as seen in Figure 1b.

$$\phi(x) = \frac{e^{axk} - 1}{e^{Pe} - 1} \tag{11}$$



(a) At Pe = 0.1, L = 1, number of elements = 10, (0, 1) Dirichlet BCs

(b) At Pe = 10, L = 1, number of elements = 10, (0, 1) Dirichlet BCs

Figure 1: Galerkin Approximation solution