1. Finite Element Discretization

1.1. Discretizing the Domain

Discretize the domain $[0, L]$ into E finite elements with nodes x_i, where $i = 1, 2, \ldots, n$ and n is the total number of nodes.

1.2. Approximating the Solution and Test Functions

Approximate the solution $u(x,t)$ and the test function $v(x)$ using finite element basis functions $N_i(x)$:

$$u(x,t) \approx \sum_{j=1}^{n} U_j(t) N_j(x), \quad v(x) = N_i(x) \tag{1}$$

Here, $U_j(t)$ are the time-dependent coefficients (nodal values) of the solution, and $N_j(x)$ are the shape functions associated with each node.

1.3. Substituting into the Weak Form

Substitute these approximations into the weak form and simplify:

$$\sum_{j=1}^{n} \frac{dU_j(t)}{dt} \int_0^L N_j(x) N_i(x) dx + \sum_{j=1}^{n} U_j(t) \int_0^L U(x,t) \frac{\partial N_j(x)}{\partial x} N_i(x) dx + \nu \sum_{j=1}^{n} U_j(t) \int_0^L \frac{\partial N_j(x)}{\partial x} \frac{\partial N_i(x)}{\partial x} dx = \int_0^L f(x,t) N_i(x) dx \tag{2}$$

1.4. Defining the Matrices

This can be rewritten in matrix form:

- **Mass Matrix M:**

$$\mathbf{M}_{ij} = \int_0^L N_i(x) N_j(x) dx \tag{3}$$

- **Convection Term C(U):**

$$\mathbf{C}_{ij}(\mathbf{U}) = \int_0^L U(x,t) \frac{\partial N_j(x)}{\partial x} N_i(x) dx \tag{4}$$

- **Diffusion Matrix K:**

$$\mathbf{K}_{ij} = \nu \int_0^L \frac{\partial N_i(x)}{\partial x} \frac{\partial N_j(x)}{\partial x} dx \tag{5}$$

- **Load Vector F:**

$$\mathbf{F}_i(t) = \int_0^L f(x,t) N_i(x) dx \tag{6}$$

1.5. Discretized System of Equations

The overall system of equations is:

$$\mathbf{M} \frac{d\mathbf{U}(t)}{dt} + \mathbf{C}(\mathbf{U})\mathbf{U} + \mathbf{K}\mathbf{U} = \mathbf{F}(t) \tag{7}$$

where:

- $\mathbf{U}(t)$ is the vector of unknowns at the nodes.

1.6. Time Discretization

To solve this system over time, you apply a time discretization method such as the implicit Euler scheme:

$$\mathbf{M} \frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} + \mathbf{C}(\mathbf{U}^{n+1})\mathbf{U}^{n+1} + \mathbf{K}\mathbf{U}^{n+1} = \mathbf{F}^{n+1} \tag{8}$$

This forms a nonlinear system at each time step, which can be solved iteratively.

1.7. Solving the Nonlinear System

Once the time discretization is applied, we obtain a nonlinear system of equations at each time step, which can be written as:

$$\mathbf{A}(\mathbf{U}^{n+1})\mathbf{U}^{n+1} = \mathbf{b}^{n+1}$$

where:

$$\mathbf{A}(\mathbf{U}^{n+1}) = \mathbf{M} + \Delta t\mathbf{C}(\mathbf{U}^{n+1}) + \Delta t\mathbf{K}$$

$$\mathbf{b}^{n+1} = \mathbf{M}\mathbf{U}^{n} + \Delta t\mathbf{F}^{n+1}$$

1.7.1. Picard Iteration (Fixed-Point Iteration)

To solve this nonlinear system, we can use the Picard iteration method, which is a fixed-point iterative scheme. The steps are as follows:

1. **Initial Guess**: Start with an initial guess $\mathbf{U}_0^{n+1}$, which is usually taken as the solution from the previous time step, $\mathbf{U}^n$.
2. **Iterative Update**: Update the solution using the current approximation:

$$\mathbf{A}(\mathbf{U}_k^{n+1})\mathbf{U}_{k+1}^{n+1} = \mathbf{b}^{n+1}$$

3. **Convergence Check**: Compute the error between successive iterations:

$$\text{error}_U = \frac{\|\mathbf{U}_{k+1}^{n+1} - \mathbf{U}_k^{n+1}\|}{\|\mathbf{U}_{k+1}^{n+1}\|}$$

4. **Stopping Criterion**: The iteration continues until the error falls below a specified tolerance, or a maximum number of iterations is reached.

This method iteratively solves the linearized system until convergence, gradually refining the approximation for $\mathbf{U}^{n+1}$.

Note: Newton-Raphson Method (Alternative Approach)

An alternative approach to solving the nonlinear system is the Newton-Raphson method, which involves computing the Jacobian matrix and performing a linearization around the current guess at each iteration. This method typically converges faster but requires the computation and inversion of the Jacobian, which can be computationally expensive.

1.7.2. Boundary Conditions

Boundary conditions are crucial in ensuring that the numerical solution adheres to the physical constraints of the problem.

Dirichlet Boundary Condition at $x = 0$. For the 1D Burgers' equation, the Dirichlet boundary condition is:

$$u(0, t) = \mu_1, \quad \forall t \in [0, T]$$

To enforce this in the FEM system:

- **Modify the System Matrix A**:

$$\mathbf{A}[0, :] = 0 \quad \text{and} \quad \mathbf{A}_{00} = 1$$

  This sets the first row of $\mathbf{A}$ to ensure the solution at $x = 0$ is fixed.

- **Adjust the Right-Hand Side Vector b**:

$$\mathbf{b}[0] = \mu_1$$

  This forces the solution at the boundary node to be μ_1.

Initial Condition. The initial condition is:

$$u(x, 0) = 1, \quad \forall x \in [0, L]$$

This initializes the solution vector $\mathbf{U}(0)$ at time $t = 0$.

Summary.

- **Matrix Modification**: Set $\mathbf{A}[0,:]$ to 0 and $\mathbf{A}_{00} = 1$ to enforce Dirichlet conditions.
- **Vector Adjustment**: Set $\mathbf{b}[0] = \mu_1$ to match the boundary condition.
- **Initial Condition**: Set $\mathbf{U}(0)$ based on the initial state of the system.

This ensures that the boundary and initial conditions are properly incorporated into the numerical solution.