Introduction and Implementation for Finite Element Methods

Chapter 4: Finite elements for 2D second order parabolic and hyperbolic equations

Xiaoming He
Department of Mathematics & Statistics
Missouri University of Science & Technology
Email: hex@mst.edu
Homepage: https://web.mst.edu/~hex/

Outline

Outline

Target problem

- Consider the 2D second order parabolic equation

$$\begin{aligned} &u_t - \nabla \cdot (c \nabla u) = f, \quad \text{in } \Omega \times [0, T], \\ &u = g, \quad \text{on } \partial\Omega \times [0, T], \\ &u = u_0, \quad \text{at } t = 0 \text{ and in } \Omega. \end{aligned}$$

where Ω is a 2D domain, $[0, T]$ is the time interval, $f(x, y, t)$ and $c(x, y, t)$ are given functions on $\Omega \times [0, T]$, $g(x, y, t)$ is a given function on $\partial\Omega \times [0, T]$, $u_0(x, y)$ is given function on Ω at $t = 0$, and $u(x, y, t)$ is the unknown function.

Weak formulation

- First, multiply a function $v(x,y)$ on both sides of the original equation,

$$
\begin{aligned}
& u_t - \nabla\cdot(c\nabla u) = f \quad \text{in } \Omega \\
\Rightarrow \quad & u_t v - \nabla\cdot(c\nabla u)v = fv \quad \text{in } \Omega \\
\Rightarrow \quad & \int_\Omega u_t v \; dxdy - \int_\Omega \nabla\cdot(c\nabla u)v \; dxdy = \int_\Omega fv \; dxdy.
\end{aligned}
$$

- $u(x,y,t)$ is called a trail function and $v(x,y)$ is called a test function.

Weak formulation

- Second, using Green's formula (divergence theory, integration by parts in multi-dimension)

$$\int_\Omega \nabla \cdot (c\nabla u)v \; dxdy = \int_{\partial\Omega} (c\nabla u \cdot \vec{n}) \, v \; ds - \int_\Omega c\nabla u \cdot \nabla v \; dxdy,$$

we obtain

$$\begin{aligned} & \int_\Omega u_t v \; dxdy + \int_\Omega c\nabla u \cdot \nabla v \; dxdy - \int_{\partial\Omega} (c\nabla u \cdot \vec{n}) \, v \; ds \\ = \; & \int_\Omega fv \; dxdy. \end{aligned}$$

Weak formulation

- Since the solution on the domain boundary $\partial\Omega$ are given by $u(x, y, t) = g(x, y, t)$, then we can choose the test function $v(x, y)$ such that $v = 0$ on $\partial\Omega$.
- Hence
$$\int_\Omega u_t v \; dxdy + \int_\Omega c\nabla u \cdot \nabla v \; dxdy = \int_\Omega fv \; dxdy.$$
- What spaces should u and v belong to? Sobolev spaces! (See Chapter 3)
- Define
$$H^1(0, T; H^1(\Omega)) = \{v(t, \cdot), \; \frac{\partial v}{\partial t}(t, \cdot) \in H^1(\Omega), \; \forall t \in [0, T]\}.$$

Weak formulation

- Weak formulation: find $u \in H^1(0, T; H^1(\Omega))$ such that

$$\int_\Omega u_t v \; dxdy + \int_\Omega c\nabla u \cdot \nabla v \; dxdy = \int_\Omega fv \; dxdy.$$

  for any $v \in H_0^1(\Omega)$.
- Let $a(u, v) = \int_\Omega c\nabla u \cdot \nabla v dxdy$ and $(f, v) = \int_\Omega fv dxdy$.
- Weak formulation: find $u \in H^1(0, T; H^1(\Omega))$ such that

$$(u_t, v) + a(u, v) = (f, v)$$

  for any $v \in H_0^1(\Omega)$.

Outline

Galerkin formulation

- Consider a finite element space $U_h \subset H^1(\Omega)$. Define U_{h0} to be the space which consists of the functions of U_h with value 0 on the Dirichlet boundary.
- Then the Galerkin formulation is to find $u_h \in H^1(0, T; U_h)$ such that

$$
\begin{aligned}
& (u_{h_t}, v_h) + a(u_h, v_h) = (f, v_h) \\
\Leftrightarrow \quad & \int_\Omega u_{h_t} v_h \, dxdy + \int_\Omega c \nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy
\end{aligned}
$$

  for any $v_h \in U_{h0}$.
- Basic idea of Galerkin formulation: use finite dimensional space to approximate infinite dimensional space.
- Here $U_h = span\{\phi_j\}_{j=1}^{N_b}$ is chosen to be a finite element space where $\{\phi_j\}_{j=1}^{N_b}$ are the global finite element basis functions.

Galerkin formulation

- For an easier implementation, we consider the Galerkin formulation (without considering the Dirichlet boundary condition, which will be handled later): find $u_h \in H^1(0, T; U_h)$ such that

$$
\begin{aligned}
& (u_{h_t}, v_h) + a(u_h, v_h) = (f, v_h) \\
\Leftrightarrow \quad & \int_\Omega u_{h_t} v_h \, dxdy + \int_\Omega c \nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy
\end{aligned}
$$

for any $v_h \in U_h$.

Discretization formulation

Recall the following definitions from Chapter 2:

- N: number of mesh elements.
- N_m: number of mesh nodes.
- E_n $(n = 1, \cdots, N)$: mesh elements.
- Z_k $(k = 1, \cdots, N_m)$: mesh nodes.
- N_l: number of local mesh nodes in a mesh element.
- P:information matrix consisting of the coordinates of all mesh nodes.
- T: information matrix consisting of the global node indices of the mesh nodes of all the mesh elements.

Discretization formulation

- We only consider the nodal basis functions (Lagrange type) in this course.
- N_{lb}: number of local finite element nodes (=number of local finite element basis functions) in a mesh element.
- N_b: number of the finite element nodes (= the number of unknowns = the total number of the finite element basis functions).
- X_j $(j = 1, \cdots, N_b)$: finite element nodes.
- P_b: information matrix consisting of the coordinates of all finite element nodes.
- T_b: information matrix consisting of the global node indices of the finite element nodes of all the mesh elements.

Discretization formulation

- Since $u_h \in H^1(0, T; U_h)$ and $U_h = span\{\phi_j\}_{j=1}^{N_b}$, then

$$u_h(x, y, t) = \sum_{j=1}^{N_b} u_j(t)\phi_j(x, y)$$

for some coefficients $u_j(t)$ $(j = 1, \cdots, N_b)$.

- If we can set up a linear algebraic system for

$$u_j(t) \ (j = 1, \cdots, N_b)$$

and solve it, then we can obtain the finite element solution u_h.

Discretization formulation

- Therefore, we choose $v_h = \phi_i \ (i = 1, \cdots, N_b)$. Then

$$\int_\Omega \left(\sum_{j=1}^{N_b} u_j(t)\phi_j \right)_t \phi_i \ dxdy + \int_\Omega c\nabla \left(\sum_{j=1}^{N_b} u_j(t)\phi_j \right) \cdot \nabla\phi_i \ dxdy$$
$$= \int_\Omega f\phi_i \ dxdy, \ \ i = 1, \cdots, N_b$$
$$\Rightarrow \sum_{j=1}^{N_b} u_j'(t) \left[\int_\Omega \phi_j\phi_i \ dxdy \right] + \sum_{j=1}^{N_b} u_j(t) \left[\int_\Omega c\nabla\phi_j \cdot \nabla\phi_i \ dxdy \right]$$
$$= \int_\Omega f\phi_i \ dxdy, \ \ i = 1, \cdots, N_b.$$

- Here the basis functions $\phi_i \ (i = 1, \cdots, N_b)$ depend on (x, y) only. But the given functions c and f may depend on t and (x, y).

Matrix formulation

- Define the stiffness matrix

$$A(t) = [a_{ij}]_{i,j=1}^{N_b} = \left[\int_{\Omega} c\nabla\phi_j \cdot \nabla\phi_i \; dxdy\right]_{i,j=1}^{N_b}.$$

- Define the mass matrix

$$M = [m_{ij}]_{i,j=1}^{N_b} = \left[\int_{\Omega} \phi_j\phi_i \; dxdy\right]_{i,j=1}^{N_b}.$$

- Define the load vector

$$\vec{b}(t) = [b_i]_{i=1}^{N_b} = \left[\int_{\Omega} f\phi_i \; dxdy\right]_{i=1}^{N_b}.$$

- Define the unknown vector

$$\vec{X}(t) = [u_j(t)]_{j=1}^{N_b}.$$

- Then we obtain the system

$$M\vec{X}'(t) + A(t)\vec{X}(t) = \vec{b}(t).$$

Matrix formulation

- At a given time t, the assembly of the stiffness matrix $A(t)$ and the load vector $\vec{b}(t)$ is the same as that of the A and b in Chapter 3. But the given time t needs to be incorporated into the code.
- In some simulation, the functions c in the given parabolic equation may not depend on t. In this case, the stiffness matrix $A(t)$ is actually independent of t, hence can be generated before the time marching in exactly the same way as the A in Chapter 3.
- Similarly, the functions f in the given parabolic equation may not depend on t in some simulation. In this case, the load vector $\vec{b}(t)$ is actually independent of t, hence can be generated before the time marching in exactly the same way as the $\vec{b}$ in Chapter 3.

Assembly of the mass matrix

- Any observation for the mass matrix

$$M = [m_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega \phi_j \phi_i \ dxdy\right]_{i,j=1}^{N_b} ?$$

- Following the same procedure for A from

$$\int_\Omega c\nabla\phi_j \cdot \nabla\phi_i \ dxdy$$

to

$$\int_{E_n} c\nabla\psi_{n\alpha} \cdot \nabla\psi_{n\beta} \ dxdy$$

in Chapter 3, we can also get

$$\int_{E_n} \psi_{n\alpha}\psi_{n\beta} \ dxdy \ \left(\text{from } \int_\Omega \phi_j\phi_i \ dxdy\right).$$

- Just use Algorithm I-3 with $r = s = p = q = 0$ and $c = 1$!

Assembly of a time-independent matrix

Recall Algorithm I-3 from Chapter 3:

- Initialize the matrix: $A = sparse(N_b, N_b)$;
- Compute the integrals and assemble them into A:

$FOR\ n = 1, \cdots, N$:

  $FOR\ \alpha = 1, \cdots, N_{lb}$:

    $FOR\ \beta = 1, \cdots, N_{lb}$:

      Compute $r = \int_{E_n} c \frac{\partial^{r+s}\psi_{n\alpha}}{\partial x^r \partial y^s} \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\ dxdy$;

      Add r to $A(T_b(\beta, n), T_b(\alpha, n))$.

    END

  END

END

Assembly of a time-dependent matrix

Algorithm I-5:

- Specify a value for the time t based on the input time;
- Initialize the matrix: $A = sparse(N_b, N_b)$;
- Compute the integrals and assemble them into A:

$FOR\ n = 1, \cdots, N$:

$\quad FOR\ \alpha = 1, \cdots, N_{lb}$:

$\quad\quad FOR\ \beta = 1, \cdots, N_{lb}$:

$\quad\quad\quad$ Compute $r = \int_{E_n} c(t) \frac{\partial^{r+s}\psi_{n\alpha}}{\partial x^r \partial y^s} \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\, dxdy$;

$\quad\quad\quad$ Add r to $A(T_b(\beta, n), T_b(\alpha, n))$.

$\quad\quad END$

$\quad END$

END

Assembly of the stiffness matrix

- First, we call Algorithm I-5 with $r = p = 1$, $s = q = 0$, and $c(x, y, t)$ to obtain $A1(t)$.
- Second, we call Algorithm I-5 with $r = p = 0$, $s = q = 1$, and $c(x, y, t)$ to obtain $A2(t)$.
- Then the stiffness matrix $A(t) = A1(t) + A2(t)$.
- If c does not depend on t, then this part is exactly the same as the assembly of the stiffness matrix with Algorithm I-3 in Chapter 3.

Assembly of a time-independent vector

Recall Algorithm II-3 from Chapter 3:

- Initialize the matrix: $b = sparse(N_b, 1)$;
- Compute the integrals and assemble them into b:

$FOR\ n = 1, \cdots, N$:

$\quad FOR\ \beta = 1, \cdots, N_{lb}$:

$\quad\quad$ Compute $r = \int_{E_n} f \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\, dxdy$;

$\quad\quad b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;

$\quad END$

END

Assembly of a time-dependent vector

Algorithm II-5:

- Specify a value for the time t based on the input time;
- Initialize the vector: $b = sparse(N_b, 1)$;
- Compute the integrals and assemble them into b:

$FOR\ n = 1, \cdots, N$:

$\quad FOR\ \beta = 1, \cdots, N_{lb}$:

$\quad\quad$ Compute $r = \int_{E_n} f(t) \frac{\partial^{p+q}\psi_{n\beta}}{\partial x^p \partial y^q}\ dxdy$;

$\quad\quad b(T_b(\beta, n), 1) = b(T_b(\beta, n), 1) + r$;

$\quad END$

END

Assembly of the load vector

- We call Algorithm II-5 with $p = q = 0$ and $f(x, y, t)$ to obtain $b(t)$.
- If f does not depend on t, then this part is exactly the same as the assembly of the load vector with Algorithm II-3 in Chapter 3.

Time-dependent Dirichlet boundary condition

Since Algorithm III Chapter 3 is time-independent, it is not suitable for the time-dependent Dirichlet boundary condition in this chapter. Therefore, we will use the following Algorithm III-2:

- Specify a value for the time t based on the input time;
- Deal with the Dirichlet boundary conditions:

  FOR $k = 1, \cdots, nbn$:

  If $boundarynodes(1, k)$ shows Dirichlet condition, then

  $i = boundarynodes(2, k)$;

  $\bar{A}(i, :) = 0$;

  $\bar{A}(i, i) = 1$;

  $\bar{b}(i) = g(P_b(:, i), t)$;

  ENDIF

  END

Outline

Observation

- Any observation for the system

$$M\vec{X}'(t) + A(t)\vec{X}(t) = \vec{b}(t)?$$

- System of ordinary differential equations (ODEs)!
- How to solve it?
- Finite difference (FD) method!

Review of finite difference method for a first order ODE

Basic idea:

- Consider the IVP
$$y'(t) = f(t, y(t)) \ (a \le t \le b), \ \ y(a) = g_a$$
given the initial value g_a.
- Assume that we have a uniform partition of $[a, b]$ into J elements with mesh size h.
- The mesh nodes are $t_j = a + jh, \ \ j = 0, 1, \cdots, J$.
- Assume y_j is the numerical solution of $y(t_j)$.
- Then the initial condition implies: $y_0 = y(a) = g_a$.
- A straightforward discretization of $f(t, y(t))$ at t_j is $f(t_j, y_j)$.
- How about the discretization of $y'(t)$ at t_j?
- Taylor's expansion!

Review of finite difference method for a first order ODE

Theorem

Suppose that $f(x)$ is a $(n+1)^{th}$ differentiable function on $[a,b]$ and $x_0 \in [a,b]$. Then for any $x \in [a,b]$, we have the following Taylor's expansion of $f(x)$ at x_0:

$$f(x) = P_n(x) + R_n(x),$$

where

$$\begin{aligned} P_n(x) &= \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x_0)(x-x_0)^k \\ &= f(x_0) + f'(x_0)(x-x_0) + \frac{1}{2!} f''(x_0)(x-x_0)^2 + \cdots \\ &\quad + \frac{1}{n!} f^{(n)}(x_0)(x-x_0)^n, \end{aligned}$$

Review of finite difference method for a first order ODE

Theorem (Continued)

$$R_n = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x-x_0)^{n+1}$$
$$\text{for some } \xi \in [x_0, x] \quad \text{(Lagrange form of the remainder)}\ ,$$

or

$$R_n = \frac{1}{n!}\int_{x_0}^{x} f^{(n+1)}(s)(x-s)^n \, ds$$
$$\text{(Integral form of the remainder)}\ .$$

Review of finite difference method for a first order ODE

- Pick $n = 3$ in the Taylor's expansion:

$$\begin{aligned} f(x) &= f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 \\ &\quad + \frac{1}{6}f'''(x_0)(x - x_0)^3 + \frac{1}{24}f^{(4)}(\xi)(x - x_0)^4. \end{aligned}$$

- Replace x by $x + h$ and x_0 by x:

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{1}{6}f'''(x)h^3 + O\left(h^4\right).$$

- We first consider the discretization of the first derivative $f'(x)$. Then

$$\begin{aligned} f'(x) &= \frac{f(x+h) - f(x)}{h} - \frac{1}{2}f''(x)h - \frac{1}{6}f'''(x)h^2 - O\left(h^3\right) \\ &= \frac{f(x+h) - f(x)}{h} + O(h). \end{aligned}$$

Review of finite difference method for a first order ODE

- Assume that we have a uniform partition of $[a, b]$ into J elements with mesh size h.
- The mesh nodes are $t_j = a + jh, \ j = 0, 1, \cdots, J$.
- Then

$$
\begin{aligned}
f'(t_j) &= \frac{f(t_j + h) - f(t_j)}{h} + O(h) \\
&= \frac{f(t_{j+1}) - f(t_j)}{h} + O(h) \\
&\approx \frac{f_{j+1} - f_j}{h}, \ j = 0, 1, \cdots, J - 1.
\end{aligned}
$$

Here f_j is the approximation of $f(t_j)$. This is called forward difference.

Review of finite difference method for a first order ODE

- Recall the Taylor's expansion with $n = 3$:

$$
\begin{aligned}
f(x) &= f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(x_0)(x - x_0)^2 \\
&\quad + \frac{1}{6} f'''(x_0)(x - x_0)^3 + \frac{1}{24} f^{(4)}(\xi)(x - x_0)^4.
\end{aligned}
$$

- Replace x by $x - h$ and x_0 by x:

$$
f(x-h) = f(x) - f'(x)h + \frac{1}{2} f''(x)h^2 - \frac{1}{6} f'''(x)h^3 + O\left(h^4\right).
$$

- Then

$$
\begin{aligned}
f'(x) &= \frac{f(x) - f(x-h)}{h} + \frac{1}{2} f''(x)h - \frac{1}{6} f'''(x)h^2 + O\left(h^3\right) \\
&= \frac{f(x) - f(x-h)}{h} + O(h).
\end{aligned}
$$

Review of finite difference method for a first order ODE

- Consider the same partition as above.
- Then

$$
\begin{aligned}
f'(t_j) &= \frac{f(t_j) - f(t_j - h)}{h} + O(h) \\
&= \frac{f(t_j) - f(t_{j-1})}{h} + O(h) \\
&\approx \frac{f_j - f_{j-1}}{h}, \; j = 1, \cdots, J.
\end{aligned}
$$

This is called backward difference.

Review of finite difference method for a first order ODE

- Observation: Both of the forward and backward difference schemes are of first order.
- Is it possible to construct a higher order difference scheme for $f'(x_j)$? Yes!
- Recall

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \frac{1}{6}f'''(x)h^3 + O\left(h^4\right),$$

$$f(x-h) = f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 - \frac{1}{6}f'''(x)h^3 + O\left(h^4\right).$$

- Subtract the second equation from the first one:

$$f(x+h) - f(x-h) = 2f'(x)h + \frac{1}{3}f'''(x)h^3 + O\left(h^4\right).$$

Review of finite difference method for a first order ODE

- Then

$$\begin{aligned} f'(x) &= \frac{f(x+h)-f(x-h)}{2h} - \frac{1}{12}f'''(x)h^2 + O\left(h^3\right) \\ &= \frac{f(x+h)-f(x-h)}{2h} + O\left(h^2\right). \end{aligned}$$

This is second order!

- Hence

$$\begin{aligned} f'(t_j) &= \frac{f(t_j+h)-f(t_j-h)}{2h} + O\left(h^2\right) \\ &= \frac{f(t_{j+1})-f(t_{j-1})}{2h} + O\left(h^2\right) \\ &\approx \frac{f_{j+1}-f_{j-1}}{2h}, \; j = 1, \cdots, J-1. \end{aligned}$$

This is called centered difference.

Review of finite difference method for a first order ODE

Hence we obtain the following difference schemes:

- Forward difference for $y'(t_j) \approx \frac{y_{j+1} - y_j}{h}$.
- Backward difference for $y'(t_j) \approx \frac{y_j - y_{j-1}}{h}$.
- Centered difference for $y'(t_j) \approx \frac{y_{j+1} - y_{j-1}}{2h}$.

Review of finite difference method for a first order ODE

- Forward Euler scheme:

$$
\begin{aligned}
& y'(t) = f(t, y(t)) \\
\Rightarrow\quad & y'(t_j) = f(t_j, y(t_j)),\ j = 0, \cdots, J-1 \\
\Rightarrow\quad & \frac{y(t_{j+1}) - y(t_j)}{h} + O(h) = f(t_j, y(t_j)),\ j = 0, \cdots, J-1 \\
\Rightarrow\quad & \frac{y_{j+1} - y_j}{h} = f(t_j, y_j),\ j = 0, \cdots, J-1 \\
\Rightarrow\quad & {\color{red} y_{j+1} = y_j + h \cdot f(t_j, y_j),\ j = 0, \cdots, J-1}, \\
& y_0 = y(a) = g_a.
\end{aligned}
$$

Review of finite difference method for a first order ODE

- Backward Euler scheme:

$$
\begin{aligned}
& y'(t) = f(t, y(t)) \\
\Rightarrow\quad & y'(t_j) = f(t_j, y(t_j)),\ j = 1, \cdots, J \\
\Rightarrow\quad & \frac{y(t_j) - y(t_{j-1})}{h} + O(h) = f(t_j, y(t_j)),\ j = 1, \cdots, J \\
\Rightarrow\quad & \frac{y_j - y_{j-1}}{h} = f(t_j, y_j),\ j = 1, \cdots, J \\
\Rightarrow\quad & \frac{y_{j+1} - y_j}{h} = f(t_{j+1}, y_{j+1}),\ j = 0, \cdots, J-1 \\
\Rightarrow\quad & y_{j+1} = y_j + h \cdot f(t_{j+1}, y_{j+1}),\ j = 0, \cdots, J-1, \\
& y_0 = y(a).
\end{aligned}
$$

Review of finite difference method for a first order ODE

- Trapezoidal scheme(Crank-Nicolson scheme if it's applied to PDE):

$$\frac{y_{j+1}-y_j}{h}=\frac{f(t_{j+1},y_{j+1})+f(t_j,y_j)}{2};$$

- Two-step backward differentiation:

$$\frac{3y_{j+1}-4y_j+y_{j-1}}{2h}=f(t_{j+1},y_{j+1});$$

- Three-step backward differentiation:

$$\frac{11y_{j+1}-18y_j+9y_{j-1}-2y_{j-2}}{6h}=f(t_{j+1},y_{j+1}).$$

Review of finite difference method for a first order ODE

- Actually, the forward Euler scheme, backward Euler scheme, and Crank-Nicolson scheme can be rewritten into a more general $\theta-$scheme:

$$\frac{y_{j+1}-y_j}{h}=\theta f(t_{j+1},y_{j+1})+(1-\theta)f(t_j,y_j);$$

- $\theta=0$: forward Euler scheme;
- $\theta=1$: backward Euler scheme;
- $\theta=\frac{1}{2}$: Crank-Nicolson scheme.

Temporal discretization for the ODE system

- Now let's consider the system of ODEs:

$$M\vec{X}'(t) + A(t)\vec{X}(t) = \vec{b}(t).$$

- Assume that we have a uniform partition of $[0, T]$ into M_m elements with mesh size $\triangle t$.
- The mesh nodes are $t_m = m\triangle t,\ \ m = 0, 1, \cdots, M_m$.
- Assume $\vec{X}^m$ is the numerical solution of $\vec{X}(t_m)$.
- Then the corresponding $\theta-$scheme is

$$M\frac{\vec{X}^{m+1} - \vec{X}^m}{\triangle t} + \theta A(t_{m+1})\vec{X}^{m+1} + (1-\theta)A(t_m)\vec{X}^m$$

$$= \ \ \theta\vec{b}(t_{m+1}) + (1-\theta)\vec{b}(t_m),\ \ m = 0, \cdots, M_m - 1.$$

Temporal discretization for the ODE system

- Then

$$
\begin{aligned}
& M\frac{\vec{X}^{m+1}-\vec{X}^{m}}{\triangle t}+\theta A(t_{m+1})\vec{X}^{m+1}+(1-\theta)A(t_m)\vec{X}^{m} \\
&= \theta\vec{b}(t_{m+1})+(1-\theta)\vec{b}(t_m) \\
\Rightarrow\quad & \left[\frac{M}{\triangle t}+\theta A(t_{m+1})\right]\vec{X}^{m+1} \\
&= \theta\vec{b}(t_{m+1})+(1-\theta)\vec{b}(t_m)+\frac{M}{\triangle t}\vec{X}^{m}-(1-\theta)A(t_m)\vec{X}^{m}.
\end{aligned}
$$

- Iteration scheme 1:

$$\bar{A}^{m+1}\vec{X}^{m+1}=\bar{\vec{b}}^{m+1},\ \ m=0,\cdots,M_m-1,$$

where

$$
\begin{aligned}
\bar{A}^{m+1} &= \frac{M}{\triangle t}+\theta A(t_{m+1}), \\
\bar{\vec{b}}^{m+1} &= \theta\vec{b}(t_{m+1})+(1-\theta)\vec{b}(t_m)+\frac{M}{\triangle t}\vec{X}^{m}-(1-\theta)A(t_m)\vec{X}^{m}.
\end{aligned}
$$

Temporal discretization for the ODE system

Algorithm A:

- Generate the mesh information matrices P and T.
- Assemble the mass matrix M by using Algorithm I-3.
- Generate the initial vector $\vec{X}^0$.
- Iterate in time:

  FOR $m = 0, \cdots, M_m - 1$:

  $t_{m+1} = (m+1)\triangle t$;

  $t_m = m\triangle t$;

  Assemble the stiffness matrices $A(t_{m+1})$ and $A(t_m)$ by using Algorithm I-5 at $t = t_{m+1}$ and $t = t_m$;

  Assemble the load vectors $\vec{b}(t_{m+1})$ and $\vec{b}(t_m)$ by using Algorithm II-5 at $t = t_{m+1}$ and $t = t_m$;

  Deal with Dirichlet boundary conditions by using Algorithm III-2 for $\bar{A}^{m+1}$ and $\bar{\vec{b}}^{m+1}$ at $t = t_{m+1}$;

  Solve iteration scheme 1 for $\vec{X}^{m+1}$.

  END

Temporal discretization for the ODE system

Remark

The matrix A, vector $\vec{b}$ and boundary conditions could be independent of the time. In this case, they can be handled before the loop for the time iteration starts, which can dramatically save the computational cost.

Temporal discretization for the ODE system

- If the function c is independent of the time t, then the stiffness matrix A is independent of time t. Then

$$M\frac{\vec{X}^{m+1}-\vec{X}^{m}}{\triangle t}+\theta A\vec{X}^{m+1}+(1-\theta)A\vec{X}^{m}=\theta\vec{b}(t_{m+1})+(1-\theta)\vec{b}(t_m)$$

$$\Rightarrow \quad \left(\frac{M}{\triangle t}+\theta A\right)\vec{X}^{m+1}=\theta\vec{b}(t_{m+1})+(1-\theta)\vec{b}(t_m)+\frac{M}{\triangle t}\vec{X}^{m}-(1-\theta)A\vec{X}^{m}.$$

- Iteration scheme 2:

$$\bar{A}\vec{X}^{m+1}=\vec{\bar{b}}^{m+1},\quad m=0,\cdots,M_m-1,$$

where

$$\bar{A}=\frac{M}{\triangle t}+\theta A,$$

$$\vec{\bar{b}}^{m+1}=\theta\vec{b}(t_{m+1})+(1-\theta)\vec{b}(t_m)+\left[\frac{M}{\triangle t}-(1-\theta)A\right]\vec{X}^{m}.$$

Temporal discretization for the ODE system

Algorithm B:

- Generate the mesh information matrices P and T.
- Assemble the mass matrix M by using Algorithm I-3.
- Assemble the stiffness matrix A by using Algorithm I-3.
- Generate the initial vector $\vec{X}^0$.
- Iterate in time:

  FOR $m = 0, \cdots, M_m - 1$:

  $t_{m+1} = (m+1)\triangle t$;

  $t_m = m\triangle t$;

  Assemble the load vectors $\vec{b}(t_{m+1})$ and $\vec{b}(t_m)$ by using Algorithm II-5 at $t = t_{m+1}$ and $t = t_m$;

  Deal with Dirichlet boundary conditions by using Algorithm III-2 for $\bar{A}$ and $\vec{\bar{b}}^{m+1}$ at $t = t_{m+1}$;

  Solve iteration scheme 2 for $\vec{X}^{m+1}$.

  END

Temporal discretization for the ODE system

- Define $\vec{X}^{m+\theta} = \theta\vec{X}^{m+1} + (1-\theta)\vec{X}^{m}$.
- Then $\vec{X}^{m+1} - \vec{X}^{m} = \frac{\vec{X}^{m+\theta}-\vec{X}^{m}}{\theta}$ if $\theta \neq 0$.
- Hence

$$
\begin{aligned}
& M\frac{\vec{X}^{m+1}-\vec{X}^{m}}{\triangle t} + \theta A\vec{X}^{m+1} + (1-\theta)A\vec{X}^{m} = \theta\vec{b}(t_{m+1}) + (1-\theta)\vec{b}(t_m) \\
\Rightarrow\quad & M\frac{\vec{X}^{m+1}-\vec{X}^{m}}{\triangle t} + A\left[\theta\vec{X}^{m+1} + (1-\theta)\vec{X}^{m}\right] = \theta\vec{b}(t_{m+1}) + (1-\theta)\vec{b}(t_m) \\
\Rightarrow\quad & M\frac{\vec{X}^{m+\theta}-\vec{X}^{m}}{\theta\triangle t} + A\vec{X}^{m+\theta} = \theta\vec{b}(t_{m+1}) + (1-\theta)\vec{b}(t_m) \\
\Rightarrow\quad & \left(\frac{M}{\theta\triangle t} + A\right)\vec{X}^{m+\theta} = \theta\vec{b}(t_{m+1}) + (1-\theta)\vec{b}(t_m) + \frac{M\vec{X}^{m}}{\theta\triangle t}.
\end{aligned}
$$

Temporal discretization for the ODE system

- Iteration scheme 3:

$$\bar{A}^{\theta}\vec{X}^{m+\theta} = \vec{\bar{b}}^{m+\theta}, \quad m = 0, \cdots, M_m - 1,$$

where

$$\bar{A}^{\theta} = \frac{M}{\theta \triangle t} + A,$$

$$\vec{\bar{b}}^{m+\theta} = \theta\vec{b}(t_{m+1}) + (1-\theta)\vec{b}(t_m) + \frac{M}{\theta \triangle t}\vec{X}^{m}.$$

- Since $\vec{X}^{m+\theta} = \theta\vec{X}^{m+1} + (1-\theta)\vec{X}^{m}$, then

$$\vec{X}^{m+1} = \frac{\vec{X}^{m+\theta} - \vec{X}^{m}}{\theta} + \vec{X}^{m}.$$

Temporal discretization for the ODE system

Algorithm C:

- Generate the mesh information matrices P and T.
- Assemble the mass matrix M by using Algorithm I-3.
- Assemble the stiffness matrix A by using Algorithm I-3.
- Generate the initial vector $\vec{X}^0$.
- Iterate in time:

  FOR $m = 0, \cdots, M_m - 1$:

  $t_{m+1} = (m+1)\triangle t$;

  $t_m = m\triangle t$;

  Assemble the load vectors $\vec{b}(t_{m+1})$ and $\vec{b}(t_m)$ by using Algorithm II-5 at $t = t_{m+1}$ and $t = t_m$;

  Deal with Dirichlet boundary conditions by using Algorithm III-2 for $\bar{A}^\theta$ and $\bar{\vec{b}}^{m+\theta}$ at $t = t_{m+\theta}$;

  Solve iteration scheme 3 for $\vec{X}^{m+1}$.

  END

Numerical example

- Example 1: Use the finite element method to solve the following equation for $u(x, y, t)$ on the domain $\Omega = [0, 2] \times [0, 1]$:

$$
\begin{aligned}
u_t - \nabla \cdot (2\nabla u) &= -3e^{x+y+t}, \quad \text{in } \Omega \times [0, 1], \\
u &= e^{x+y}, \quad \text{at } t = 0 \text{ and in } \Omega, \\
u &= e^{y+t} \quad \text{on } x = 0, \\
u &= e^{2+y+t} \quad \text{on } x = 2, \\
u &= e^{x+t} \quad \text{on } y = 0, \\
u &= e^{x+1+t} \quad \text{on } y = 1.
\end{aligned}
$$

- The analytic solution of this problem is $u = e^{x+y+t}$, which can be used to compute the error of the numerical solution.

Numerical example

- Let's code for the linear and quadratic finite element method of the 2D second order parabolic equation together!
- We will use Algorithm B.
- Open your Matlab!

Numerical example

h	$\|u-u_h\|_\infty$	$\|u-u_h\|_0$	$\|u-u_h\|_1$
$1/4$	3.7039×10^{-1}	1.9449×10^{-1}	2.5875×10^{0}
$1/8$	9.8704×10^{-2}	5.0853×10^{-2}	1.2865×10^{0}
$1/16$	2.5483×10^{-2}	1.2871×10^{-2}	6.4214×10^{-1}
$1/32$	6.4745×10^{-3}	3.2279×10^{-3}	3.2092×10^{-1}
$1/64$	1.6318×10^{-3}	8.0763×10^{-4}	1.6044×10^{-1}

Table: Case 1: The numerical errors at $t=1$ for linear finite element and backward Euler scheme ($\theta = 1$) with $\triangle t = 4h^2$.

- Any Observation?

Numerical example

- Second order convergence $O(h^2)$ in L^2/L^∞ norm and first order convergence $O(h)$ in H^1 semi-norm.
- The backward Euler scheme has first order accuracy for temporal discretization.
- The linear finite element has second order accuracy in L^2/L^∞ norm and first order accuracy in H^1 semi-norm for spatial discretization.
- Hence the accuracy order is expected to be $O(\triangle t + h^2)$ in L^2/L^∞ norm and $O(\triangle t + h)$ in H^1 norm, which match the above observation since $\triangle t = 4h^2$ in case 1.

Numerical example

- Case 2: The numerical errors at $t = 1$ for quadratic finite element and backward Euler scheme ($\theta = 1$) with $\triangle t = 8h^3$.
- In your final exam project, you should observe third order convergence $O(h^3)$ in L^2/L^∞ norm and second order convergence $O(h^2)$ in H^1 semi-norm.
- The backward Euler scheme has second order accuracy for temporal discretization.
- The quadratic finite element has third order accuracy in L^2/L^∞ norm and second order accuracy in H^1 semi-norm for spatial discretization.
- Hence the accuracy order is expected to be $O(\triangle t + h^3)$ in L^2/L^∞ norm and $O(\triangle t + h^2)$ in H^1 norm, which match the above observation since $\triangle t = 8h^3$ in case 2.

Numerical example

- However, you will also observe much more cost in time for this case too since $\triangle t = 8h^3$ is much smaller than that of the previous cases.
- When the mesh becomes finer and finer or the problem becomes 3D, the situation is even worse.
- This is why we need temporal discretization with higher order accuracy and efficient methods to solve linear systems.

Numerical example

h	$\|u-u_h\|_\infty$	$\|u-u_h\|_0$	$\|u-u_h\|_1$
$1/4$	3.7039×10^{-1}	1.4423×10^{-1}	2.5748×10^{0}
$1/8$	9.8704×10^{-2}	3.5921×10^{-2}	1.2845×10^{0}
$1/16$	2.5483×10^{-2}	8.9715×10^{-3}	6.4187×10^{-1}
$1/32$	6.4745×10^{-3}	2.2423×10^{-3}	3.2089×10^{-1}
$1/64$	1.6318×10^{-3}	5.6055×10^{-4}	1.6044×10^{-1}

Table: Case 3: The numerical errors at $t=1$ for linear finite element and Crank-Nicolson scheme ($\theta=\frac{1}{2}$) with $\triangle t=h$.

- Any Observation?

Numerical example

- Second order convergence $O(h^2)$ in L^2/L^∞ norm and first order convergence $O(h)$ in H^1 semi-norm.
- The Crank-Nicolson scheme has second order accuracy for temporal discretization.
- The linear finite element has second order accuracy in L^2/L^∞ norm and first order accuracy in H^1 semi-norm for spatial discretization.
- Hence the accuracy order is expected to be $O(\triangle t^2 + h^2)$ in L^2/L^∞ norm and $O(\triangle t^2 + h)$ in H^1 norm, which match the above observation since $\triangle t = h$ in case 3.

Numerical example

h	$\triangle t$	$\|u-u_h\|_\infty$	$\|u-u_h\|_0$	$\lvert u-u_h\rvert_1$
$1/4$	$1/8$	6.1549×10^{-3}	2.2830×10^{-3}	8.3065×10^{-2}
$1/8$	$1/23$	8.1024×10^{-4}	2.8702×10^{-4}	2.0725×10^{-2}
$1/16$	$1/64$	1.0403×10^{-4}	3.6236×10^{-5}	5.1789×10^{-3}
$1/32$	$1/181$	1.3179×10^{-5}	4.5451×10^{-6}	1.2946×10^{-3}
$1/64$	$1/512$	1.6587×10^{-6}	5.6913×10^{-7}	3.2363×10^{-4}

Table: Case 4: The numerical errors at $t=1$ for quadratic finite element and Crank-Nicolson scheme ($\theta=\frac{1}{2}$) with $\triangle t^2\approx h^3$.

- Any Observation?

Numerical example

- Third order convergence $O(h^3)$ in L^2/L^∞ norm and second order convergence $O(h^2)$ in H^1 semi-norm.
- The Crank-Nicolson scheme has second order accuracy for temporal discretization.
- The quadratic finite element has third order accuracy in L^2/L^∞ norm and second order accuracy in H^1 semi-norm for spatial discretization.
- Hence the accuracy order is expected to be $O(\triangle t^2 + h^3)$ in L^2/L^∞ norm and $O(\triangle t^2 + h^2)$ in H^1 norm, which match the above observation since $\triangle t^2 \approx h^3$ in case 4.

Outline

Efficient methods

- Forward Euler: cheap at each time iteration step, but conditionally stable, which means that $\triangle t$ must be smaller enough.
- Multi-step methods for temporal discretization: two-step backward differentiation, three-step backward differentiation, Runge-Kutta method......
- Efficient solvers for linear systems: multi-grid, PCG, GMRES......

Mixed boundary conditions

- The treatment of the Neumann/Robin boundary conditions is similar to that of Chapter 3.
- If the functions in the Neumann/Robin boundary conditions are independent of time, then the same subroutines from Chapter 3 can be used before the time iteration starts.
- If the functions in the Neumann/Robin boundary conditions depend on time, then the same algorithms as those in Chapter 3 can be used at each time iteration step. But the time needs to be specified in these algorithms.

Mixed boundary conditions

- Consider

$$
\begin{aligned}
& u_t - \nabla \cdot (c \nabla u) = f \quad \text{in } \Omega \times [0, T], \\
& \nabla u \cdot \vec{n} = p \quad \text{on } \Gamma_N \times [0, T], \\
& \nabla u \cdot \vec{n} + ru = q \quad \text{on } \Gamma_R \times [0, T], \\
& u = g \quad \text{on } \Gamma_D \times [0, T], \\
& u = u_0, \quad \text{at } t = 0 \text{ and in } \Omega
\end{aligned}
$$

  where $\Gamma_N, \ \Gamma_R \subset \partial\Omega$ and $\Gamma_D = \partial\Omega / (\Gamma_N \cup \Gamma_R)$.

- Recall

$$
\begin{aligned}
& \int_\Omega u_t v \ dxdy + \int_\Omega c \nabla u \cdot \nabla v \ dxdy - \int_{\partial\Omega} (c \nabla u \cdot \vec{n}) \, v \ ds \\
= \ & \int_\Omega fv \ dxdy.
\end{aligned}
$$

Mixed boundary conditions

- Since the solution on $\Gamma_D = \partial\Omega/(\Gamma_N \cup \Gamma_R)$ is given by $u = g$, then we can choose the test function v such that $v = 0$ on $\partial\Omega/(\Gamma_N \cup \Gamma_R)$.
- Hence, similar to the treatment of the mixed boundary condition in Chapter 3, the weak formulation is to find $u \in H^1(0, T; H^1(\Omega))$ such that

$$\int_\Omega u_t v \, dxdy + \int_\Omega c\nabla u \cdot \nabla v \, dxdy + \int_{\Gamma_R} cruv \, ds$$
$$= \int_\Omega fv \, dxdy + \int_{\Gamma_N} cpv \, ds + \int_{\Gamma_R} cqv \, ds$$

for any $v \in H^1_{0D}(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \Gamma_D\}$.

- Code? Combine all of the subroutines for Dirichlet/Neumann/Robin boundary conditions.

Non-isotropic second order parabolic equation with mixed boundary conditions

- Consider

$$
\begin{aligned}
& u_t - \nabla \cdot (c \nabla u) = f \quad \text{in } \Omega \times [0, T], \\
& c \nabla u \cdot \vec{n} = p \quad \text{on } \Gamma_N \times [0, T], \\
& c \nabla u \cdot \vec{n} + ru = q \quad \text{on } \Gamma_R \times [0, T], \\
& u = g \quad \text{on } \Gamma_D \times [0, T], \\
& u = u_0, \quad \text{at } t = 0 \text{ and in } \Omega
\end{aligned}
$$

  where $\Gamma_N, \ \Gamma_R \subset \partial\Omega$, $\Gamma_D = \partial\Omega / (\Gamma_N \cup \Gamma_R)$, and

$$
c = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.
$$

- The treatment of the non-isotropic equation is similar to that of Chapter 3.

Another format of full discretization

- Recall the Galerkin formulation of the semi-discretization (without considering the Dirichlet boundary condition, which will be handled later): find $u_h \in H^1(0, T; U_h)$ such that

$$(u_{h_t}, v_h) + a(u_h, v_h) = (f, v_h)$$

$$\Leftrightarrow \quad \int_\Omega u_{h_t} v_h \, dxdy + \int_\Omega c \nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy$$

  for any $v_h \in U_h$.

- Instead of obtaining the matrix formulation from this semi-discretization and proposing the full discretization based on the matrix formulation, we can first present the full discretization based on this semi-discretization and then obtain the matrix formulation for the full discretization.

Another format of full discretization

- Assume that we have a uniform partition of $[0, T]$ into M_m elements with mesh size $\triangle t$.
- The mesh nodes are $t_m = m\triangle t,\ \ m = 0, 1, \cdots, M_m$.
- Let u_h^m denote the numerical solution at t_m.
- Then we consider the full discretization (without considering the Dirichlet boundary condition, which will be handled later): for $m = 0, \cdots, M_m - 1$, find $u_h^{m+1} \in U_h$ such that

$$
\begin{aligned}
& \left(\frac{u_h^{m+1} - u_h^m}{\triangle t}, v_h\right) + \theta a(u_h^{m+1}, v_h) + (1-\theta)a(u_h^m, v_h) \\
= \quad & \theta(f(t_{m+1}), v_h) + (1-\theta)(f(t_m), v_h),
\end{aligned}
$$

for any $v_h \in U_h$.

Another format of full discretization

- That is, for $m = 0, \cdots, M_m - 1$, find $u_h^{m+1} \in U_h$ such that

$$
\begin{aligned}
& \int_\Omega \frac{u_h^{m+1} - u_h^m}{\triangle t} v_h \, dxdy \\
& + \theta \int_\Omega c \nabla u_h^{m+1} \cdot \nabla v_h \, dxdy + (1-\theta) \int_\Omega c \nabla u_h^m \cdot \nabla v_h \, dxdy \\
= \; & \theta \int_\Omega f(t_{m+1}) v_h \, dxdy + (1-\theta) \int_\Omega f(t_m) v_h \, dxdy,
\end{aligned}
$$

for any $v_h \in U_h$.

Another format of full discretization

- Hence, for $m = 0, \cdots, M_m - 1$, find $u_h^{m+1} \in U_h$ such that

$$
\begin{aligned}
& \int_\Omega \frac{u_h^{m+1}}{\triangle t} v_h \, dxdy + \theta \int_\Omega c(t_{m+1}) \nabla u_h^{m+1} \cdot \nabla v_h \, dxdy \\
= \quad & \theta \int_\Omega f(t_{m+1}) v_h \, dxdy + (1-\theta) \int_\Omega f(t_m) v_h \, dxdy \\
& + \int_\Omega \frac{u_h^m}{\triangle t} v_h \, dxdy - (1-\theta) \int_\Omega c(t_m) \nabla u_h^m \cdot \nabla v_h \, dxdy
\end{aligned}
$$

for any $v_h \in U_h$.

Another format of full discretization

- Since $u_h^{m+1} \in U_h$ and $U_h = span\{\phi_j\}_{j=1}^{N_b}$, then

$$u_h^{m+1}(x,y) = \sum_{j=1}^{N_b} u_j^{m+1}\phi_j(x,y)$$

  for some coefficients u_j^{m+1} $(j = 1, \cdots, N_b)$.

- If we can set up a linear algebraic system for

$$u_j^{m+1} \ (j = 1, \cdots, N_b)$$

  and solve it, then we can obtain the finite element solution u_h^{m+1}.

Another format of full discretization

- Therefore, we choose $v_h = \phi_i$ $(i = 1, \cdots, N_b)$. Then

$$
\begin{aligned}
& \int_\Omega \frac{\sum_{j=1}^{N_b} u_j^{m+1}\phi_j}{\triangle t}\phi_i \ dxdy \\
& +\theta \int_\Omega c(t_{m+1})\nabla \left(\sum_{j=1}^{N_b} u_j^{m+1}\phi_j\right) \cdot \nabla\phi_i \ dxdy \\
= \quad & \theta \int_\Omega f(t_{m+1})\phi_i \ dxdy + (1-\theta)\int_\Omega f(t_m)\phi_i \ dxdy \\
& + \int_\Omega \frac{\sum_{j=1}^{N_b} u_j^{m}\phi_j}{\triangle t}\phi_i \ dxdy \\
& -(1-\theta)\int_\Omega c(t_m)\nabla \left(\sum_{j=1}^{N_b} u_j^{m}\phi_j\right) \cdot \nabla\phi_i \ dxdy
\end{aligned}
$$

Another format of full discretization

- Hence

$$
\begin{aligned}
& \sum_{j=1}^{N_b} u_j^{m+1} \frac{1}{\triangle t} \left(\int_\Omega \phi_j \phi_i \ dxdy \right) \\
& + \theta \sum_{j=1}^{N_b} u_j^{m+1} \left(\int_\Omega c(t_{m+1}) \nabla \phi_j \cdot \nabla \phi_i \ dxdy \right) \\
= \quad & \theta \int_\Omega f(t_{m+1}) \phi_i \ dxdy + (1-\theta) \int_\Omega f(t_m) \phi_i \ dxdy \\
& + \sum_{j=1}^{N_b} u_j^{m} \frac{1}{\triangle t} \left(\int_\Omega \phi_j \phi_i \ dxdy \right) \\
& -(1-\theta) \sum_{j=1}^{N_b} u_j^{m} \left(\int_\Omega c(t_m) \nabla \phi_j \cdot \nabla \phi_i \ dxdy \right)
\end{aligned}
$$

Another format of full discretization

- Define the stiffness matrix

$$A(t) = [a_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega c\nabla\phi_j \cdot \nabla\phi_i \ dxdy\right]_{i,j=1}^{N_b}.$$

- Define the mass matrix

$$M = [m_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega \phi_j\phi_i \ dxdy\right]_{i,j=1}^{N_b}.$$

- Define the load vector

$$\vec{b}(t) = [b_i]_{i=1}^{N_b} = \left[\int_\Omega f\phi_i \ dxdy\right]_{i=1}^{N_b}.$$

- Define the unknown vector

$$\vec{X}^{m+1} = [u_j^{m+1}]_{j=1}^{N_b}.$$

- Then we obtain the same system as in the last section:

$$\left[\frac{M}{\triangle t} + \theta A(t_{m+1})\right]\vec{X}^{m+1} = \theta\vec{b}(t_{m+1}) + (1-\theta)\vec{b}(t_m) + \frac{M}{\triangle t}\vec{X}^m - (1-\theta)A(t_m)\vec{X}^m$$

Hence the rest of the derivation and the pseudo code are the same as in the last section.

Outline

1. Weak formulation
2. Semi-discretization
3. Full discretization
4. More Discussion
5. Second order hyperbolic equation

Weak formulation

- Consider

$$u_{tt} - \nabla \cdot (c\nabla u) = f \quad \text{in } \Omega \times [0, T],$$
$$u = g \quad \text{on } \partial\Omega \times [0, T],$$
$$u = u_0, \quad \frac{\partial u}{\partial t} = u_{00} \quad \text{at } t = 0 \text{ and in } \Omega.$$

- Similar to the second order parabolic equation, one can obtain

$$\int_\Omega u_{tt} v \; dxdy + \int_\Omega c\nabla u \cdot \nabla v \; dxdy - \int_{\partial\Omega} (c\nabla u \cdot \vec{n}) \, v \; ds$$
$$= \int_\Omega fv \; dxdy.$$

Weak formulation

- Since the solution on the domain boundary $\partial\Omega$ are given by $u(x, y, t) = g(x, y, t)$, then we can choose the test function $v(x, y)$ such that $v = 0$ on $\partial\Omega$.
- Hence

$$\int_\Omega u_{tt} v \; dxdy + \int_\Omega c \nabla u \cdot \nabla v \; dxdy = \int_\Omega fv \; dxdy.$$

- What spaces should u and v belong to? Sobolev spaces! (See Chapter 3)
- Define

$$H^2(0, T; H^1(\Omega)) = \{v(t, \cdot), \; \frac{\partial v}{\partial t}(t, \cdot), \; \frac{\partial^2 v}{\partial t^2}(t, \cdot) \in H^1(\Omega), \; \forall t \in [0, T]\}.$$

Weak formulation

- Weak formulation for the second order hyperbolic equation: find $u \in H^2(0, T; H^1(\Omega))$ such that

$$\int_\Omega u_{tt} v \ dxdy + \int_\Omega c\nabla u \cdot \nabla v \ dxdy = \int_\Omega fv \ dxdy.$$

for any $v \in H_0^1(\Omega)$.

- Let $a(u, v) = \int_\Omega c\nabla u \cdot \nabla v dxdy$ and $(f, v) = \int_\Omega f v dxdy$.

- Weak formulation: find $u \in H^2(0, T; H^1(\Omega))$ such that

$$(u_{tt}, v) + a(u, v) = (f, v)$$

for any $v \in H_0^1(\Omega)$.

Galerkin formulation

- Assume there is a finite dimensional subspace $U_h \subset H^1(\Omega)$. Define U_{h0} to be the space which consists of the functions of U_h with value 0 on the Dirichlet boundary.
- Then the Galerkin formulation is to find $u_h \in H^2(0, T; U_h)$ such that

$$
\begin{aligned}
& (u_{h_{tt}}, v_h) + a(u_h, v_h) = (f, v_h) \\
\Leftrightarrow \quad & \int_\Omega u_{h_{tt}} v_h \, dxdy + \int_\Omega c \nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy
\end{aligned}
$$

for any $v_h \in U_{h0}$.
- Basic idea of Galerkin formulation: use finite dimensional space to approximate infinite dimensional space.
- Here $U_h = span\{\phi_j\}_{j=1}^{N_b}$ is chosen to be a finite element space where $\{\phi_j\}_{j=1}^{N_b}$ are the global finite element basis functions.

Galerkin formulation

- For an easier implementation, we use the following Galerkin formulation (without considering the Dirichlet boundary condition, which will be handled later): find $u_h \in H^2(0, T; U_h)$ such that

$$\begin{aligned} & (u_{h_{tt}}, v_h) + a(u_h, v_h) = (f, v_h) \\ \Leftrightarrow \quad & \int_\Omega u_{h_{tt}} v_h \, dxdy + \int_\Omega c \nabla u_h \cdot \nabla v_h \, dxdy = \int_\Omega f v_h \, dxdy \end{aligned}$$

for any $v_h \in U_h$.

Discretization formulation

- Since $u_h \in H^2(0, T; U_h)$ and $U_h = span\{\phi_j\}_{j=1}^{N_b}$, then

$$u_h(x, y, t) = \sum_{j=1}^{N_b} u_j(t)\phi_j(x, y)$$

for some coefficients $u_j(t)$ $(j = 1, \cdots, N_b)$.

- If we can set up a linear algebraic system for

$$u_j(t) \ (j = 1, \cdots, N_b)$$

and solve it, then we can obtain the finite element solution u_h.

Discretization formulation

- Therefore, we choose $v_h = \phi_i$ $(i = 1, \cdots, N_b)$. Then

$$\int_\Omega \left(\sum_{j=1}^{N_b} u_j(t)\phi_j\right)_{tt} \phi_i \; dxdy + \int_\Omega c\nabla\left(\sum_{j=1}^{N_b} u_j(t)\phi_j\right) \cdot \nabla\phi_i \; dxdy$$
$$= \int_\Omega f\phi_i \; dxdy, \;\; i = 1, \cdots, N_b$$
$$\Rightarrow \;\; \sum_{j=1}^{N_b} u_j''(t)\left[\int_\Omega \phi_j\phi_i \; dxdy\right] + \sum_{j=1}^{N_b} u_j(t)\left[\int_\Omega c\nabla\phi_j \cdot \nabla\phi_i \; dxdy\right]$$
$$= \int_\Omega f\phi_i \; dxdy, \;\; i = 1, \cdots, N_b.$$

- Here the basis functions ϕ_i $(i = 1, \cdots, N_b)$ depend on (x, y) only. But the given functions c and f may depend on t and (x, y).

Matrix formulation

- Define the stiffness matrix

$$A(t) = [a_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega c \nabla \phi_j \cdot \nabla \phi_i \ dxdy \right]_{i,j=1}^{N_b}.$$

- Define the mass matrix

$$M = [m_{ij}]_{i,j=1}^{N_b} = \left[\int_\Omega \phi_j \phi_i \ dxdy \right]_{i,j=1}^{N_b}.$$

- Define the load vector

$$\vec{b}(t) = [b_i]_{i=1}^{N_b} = \left[\int_\Omega f \phi_i \ dxdy \right]_{i=1}^{N_b}.$$

- Define the unknown vector

$$\vec{X}(t) = [u_j(t)]_{j=1}^{N_b}.$$

- Then we obtain the system

$$M\vec{X}''(t) + A(t)\vec{X}(t) = \vec{b}(t).$$

Temporal discretization for the ODE system

- Consider the centered finite difference scheme the system of ODEs:

$$M\vec{X}''(t) + A\vec{X}(t) = \vec{b}(t).$$

- Assume that we have a uniform partition of $[0, T]$ into M_m elements with mesh size $\triangle t$.
- The mesh nodes are $t_m = m\triangle t,\ m = 0, 1, \cdots, M_m$.
- Assume $\vec{X}^m$ is the numerical solution of $\vec{X}(t_m)$.
- Then the centered finite difference scheme is

$$M\frac{\vec{X}^{m+1} - 2\vec{X}^m + \vec{X}^{m-1}}{\triangle t^2} + A\frac{\vec{X}^{m+1} + 2\vec{X}^m + \vec{X}^{m-1}}{4}$$
$$= \vec{b}(t_m),\ m = 1, \cdots, M_m - 1.$$

Temporal discretization for the ODE system

- Iteration scheme 2:

$$\bar{A}\vec{X}^{m+1} = \vec{\bar{b}}^{m+1}, \ m = 1, \cdots, M_m - 1,$$

where

$$\bar{A} = \frac{M}{\triangle t^2} + \frac{A}{4},$$

$$\vec{\bar{b}}^{m+1} = \vec{b}(t_m) + \left[\frac{2M}{\triangle t^2} - \frac{A}{2}\right]\vec{X}^m - \left[\frac{M}{\triangle t^2} + \frac{A}{4}\right]\vec{X}^{m-1}.$$

Temporal discretization for the ODE system

Algorithm B:

- Generate the mesh information matrices P and T.
- Assemble the mass matrix M by using Algorithm I-3.
- Assemble the stiffness matrix A by using Algorithm I-3.
- Generate the initial vector $\vec{X}^0$ and $\vec{X}^1$ based on the initial conditions.
- Iterate in time:

  $FOR\ m = 1, \cdots, M_m - 1$:

  $t_m = m\triangle t$;

  Assemble the load vectors $\vec{b}(t_m)$ by using Algorithm II-5 at $t = t_m$;

  Deal with Dirichlet boundary conditions by using Algorithm III-2 for $\bar{A}$ and $\vec{\bar{b}}^{m+1}$ at $t = t_{m+1}$;

  Solve iteration scheme 2 for $\vec{X}^{m+1}$.

  END

Mixed boundary conditions for second order hyperbolic equations

- Consider

$$
\begin{aligned}
& u_{tt} - \nabla \cdot (c \nabla u) = f \quad \text{in } \Omega \times [0, T], \\
& \nabla u \cdot \vec{n} = p \quad \text{on } \Gamma_N \times [0, T], \\
& \nabla u \cdot \vec{n} + ru = q \quad \text{on } \Gamma_R \times [0, T], \\
& u = g \quad \text{on } \Gamma_D \times [0, T], \\
& u = u_0, \quad \frac{\partial u}{\partial t} = u_{00}, \quad \text{at } t = 0 \text{ and in } \Omega.
\end{aligned}
$$

- Recall

$$
\begin{aligned}
& \int_\Omega u_{tt} v \, dxdy + \int_\Omega c \nabla u \cdot \nabla v \, dxdy - \int_{\partial\Omega} (c \nabla u \cdot \vec{n}) \, v \, ds \\
= \; & \int_\Omega f v \, dxdy.
\end{aligned}
$$

Mixed boundary conditions for second order hyperbolic equations

- Since the solution on $\Gamma_D = \partial\Omega/(\Gamma_N \cup \Gamma_R)$ is given by $u = g$, then we can choose the test function v such that $v = 0$ on $\partial\Omega/(\Gamma_N \cup \Gamma_R)$.
- Hence, similar to the treatment of the mixed boundary condition in Chapter 3, the weak formulatio is to find $u \in H^2(0, T; H^1(\Omega))$ such that

$$\int_\Omega u_{tt} v \ dxdy + \int_\Omega c\nabla u \cdot \nabla v \ dxdy + \int_{\Gamma_R} cruv \ ds$$
$$= \int_\Omega fv \ dxdy + \int_{\Gamma_N} cpv \ ds + \int_{\Gamma_R} cqv \ ds$$

for any $v \in H^1_{0D}(\Omega) = \{v \in H^1(\Omega) : v = 0 \text{ on } \Gamma_D\}$.
- Code? Combine all of the subroutines for Dirichlet/Neumann/Robin boundary conditions.

Non-isotropic second order hyperbolic equation with mixed boundary conditions

- Consider

$$
\begin{aligned}
& u_{tt} - \nabla \cdot (c\nabla u) = f \quad \text{in } \Omega \times [0, T], \\
& c\nabla u \cdot \vec{n} = p \quad \text{on } \Gamma_N \times [0, T], \\
& c\nabla u \cdot \vec{n} + ru = q \quad \text{on } \Gamma_R \times [0, T], \\
& u = g \quad \text{on } \Gamma_D \times [0, T], \\
& u = u_0, \quad \frac{\partial u}{\partial t} = u_{00}, \quad \text{at } t = 0 \text{ and in } \Omega.
\end{aligned}
$$

where $\Gamma_N, \ \Gamma_R \subset \partial\Omega, \ \Gamma_D = \partial\Omega / (\Gamma_N \cup \Gamma_R)$, and

$$
c = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.
$$

- The treatment of the non-isotropic equation is similar to that of Chapter 3.