Parabolic problem solved with finite elements

Luca Formaggia

November 10, 2024

Description

This project deals with the solution of the following differential problem

$$\begin{cases} \dfrac{\partial}{\partial t}u - \dfrac{\partial}{\partial c}(\mu \dfrac{\partial}{\partial x}u) + cu = f & x \in (0, L), t \in (0, T) \\ u(x, 0) = u^0(x) \\ \text{b.c. at } x = 0 \text{ and } x = L. \end{cases} \tag{1}$$

Here in general $\mu = \mu(x)$, $c = c(x)$ and $f = f(x, t)$. The boundary conditions may be of Dirichlet or Neumann type:
At $x = 0$:

$$\begin{cases} u(0, t) = \alpha(t) & \text{or} \\ -\mu \frac{\partial u}{\partial x}(0, t) = h_0(t), \end{cases}$$

At $x = L$:

$$\begin{cases} u(L, t) = \beta(t) & \text{or} \\ \mu \frac{\partial u}{\partial x}(L, t) = h_L(t). \end{cases}$$

Note: the different sign in the specification of the Neumann condition at the two ends is only to write them in terms of the "outgoing diffusion flux": if the problem describes heat conduction in a bar h_0 and h_L are the heat fluxes "exiting" the domain at the left and at the right, respectively.
The project consists in writing a code that solves the problem for general boundary conditions using linear finite elements in space and a time marching scheme in time (for instance an explicit Runge-Kutta and/or an implicit scheme like backward Euler or Crank-Nicolson). At each iteration, The resulting problem is governed by a tridiagonal matrix, thanks to the properties of finite element functions. You can use the Thomas algorithm to solve the linear problem.
In finite elements the problem is converted to a weak formulation. A possible (not the only one) description of the finite element problem is as follows. If V_h is the space of linear finite elements on the mesh defined by the point $0 = x_0, x_1, \ldots, x_n = L$, that is the continuous functions $v_h : (0, L) \to \mathbb{R}$ that are linear on each element $[x_1, x_{i+1}]$,, and $V_{0,h}$ the subspace formed by functions in V_h that *are zero at the Dirichlet nodes*, at each time $t \in (0, T)$ we can look for an approximation $u_h(t) \in V_h$ such that

$$\begin{cases} (\frac{d}{dt}u_h, v_h) + a(u_h, v_h) = (f(t), v_h) + n(v_h) & \forall v_h \in V_{0,h} \\ u_h(t) \text{ satisfies the Dirichlet boundary conditions at time t} \end{cases}$$

where

$$(u,v) = \int_0^L uvdx,\ a(u,v) = \int_0^L \mu\partial_x u \partial_x v dx + \int_0^L cuvdx,$$

and $n(v_h)$ depends on the Neumann boundary conditions. For instance if we want to impose $u(0,t) = \alpha(t)$ (Dirichlet on the left node) and $\mu\partial_x u(t) = h_L(t)$ (Neumann on the right node), we have

$$u_h(t) = \alpha(t), \quad m(v_h) = v_h(L)h_L(t).$$

Moreover, t=if ϕ_i, $i = 0,\ldots,n$ is a basis of V_h, we have that for this choice of boundary conditions $V_{0,h} = \text{span}(\phi_1,\ldots,\phi_n)$ and the problem (with the chosen b.c.) may be rewritten as find $u_h(x,t) = \sum_{i=0}^n u_i(t)\phi_i(x)$ such that

$$\begin{cases} \sum_{i=1}^n (\phi_i,\phi_j)\dfrac{du_i}{dt} + \sum_{i=1}^n a(\phi_i,\phi_j)u_i = (f(t),\phi_j) + \phi_j(L)h_L(t) \quad j = 1,\ldots n. \\ u_0(t) = \alpha(t), \end{cases}$$

Algebraically, if we define the matrices M and K of dimension $n+1$ with components

$$M_{ij} = (\phi_i,\phi_j) = \in_0^L \phi_i\phi_j,\ i = 1,\ldots,n,\ j = 1,\ldots,n,$$

$$K_{ij} = a(\phi_i,\phi_j) = \int_0^L \mu\partial_x\phi_i\partial_x\phi_j + \int_0^L c\phi_i\phi_j,\ i = 1,\ldots,n,\ j = 1,\ldots,n,$$

the vector

$$b_j(t) = \int_0^L f(t)\phi_j + \phi_j(L)h_L(t),\ j = 1,\ldots n$$

and we set the zeroth line of M to zero, $A_{00} = 1$ and $b_0 = \alpha(t)$ the problem becomes: at each time $t \in (0,L)$ find $\mathbf{u}(t) \in \mathbf{R}^{n+1}$ such that

$$M\frac{d}{dt}\mathbf{u}(t) + A\mathbf{u}(t) = b_j(t),$$

and initial condition $\mathbf{u}(0)(x_i) = u^0(x_i)$, $i = 0,\ldots,n$.
The problem is now an ordinary differential system of equation that may be solved with an appropriate time advancing scheme.
Thanks to the local support properties of the Lagragian finite element basis functions ϕ_i, the formulation involves the computation of integrals of the type

$$\int_{x_i}^{x_{i+1}} \mu\partial_x\phi_k\partial_x\phi_j, \quad i \le k, j \le i+1$$

or

$$\int_{x_i}^{x_{i+1}} c\phi_k\phi_j, \quad i \le k, j \le i+1$$

or

$$\int_{x_i}^{x_{i+1}} f\phi_k, \quad i \le k \le i+1$$

If the coefficient or the forcing term are supposed to be constant on each you can compute them by hand, since the ϕ_k are linear function of which you know the expression. But if you want to be more general and treat also non-constant coefficients you should use a quadrature rule. For this type of finite elements it is sufficient to adopt a two points Gauss rule see here, or Simpson rule (less efficient).
The development can proceed as follows

1. Write first the code for the stationary problem

$$-\frac{d}{dx}(\mu\frac{du}{dx}) + cu = f \quad x \in (0, L)$$

   plus boundary conditions. If you want to simplify the problem, you may assume constant coefficient for simplicity (but not constant forcing term f!). But the general form is more challenging! You may write the function that construct the elementary matrices and assemble them into the global matrix. I suggest to impose the Diriclhet boundary conditions by constructing the problem matrix ignoring them, and then modify the matrix and the right hand side appropriately. Neumann boundary conditions are easy since they correspond modification of the right-hand-side.

2. A this point, build a test problem and verify that the code works. For simple value of f and constant coefficient one may derive the exact solution to compare with.

3. The linear system is tridiagona, so one may use Thomas algorithm for its solution. A further step is implementing the parallel cycling reduction algorithm described in the thesis in bibliography.

4. One can then move to the time dependent problem. It is not difficult once you have the tools for the steady state problem.

Some technical details

The assemblyof the matrix governing the finite element problem is normally done by a loop over the elements, exploiting the local support features of Lagrangian finite element basis function: $\phi_i = 0$ in all mesh elements $[x_k, x_{k+1}]$ not containing node x_i. In the case of linear finite elements, ϕ_i is different from zero only on elements $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$.
On the generic element $[x_k, x_{k+1}]$ we can compute the "elemental contributions" to matrices and right-hand-side. For instance,

$$a_{ij} = \int_{x_k}^{x_{k+1}} \mu\partial_x\phi_i\partial_x\phi_j, \quad i \in \{k, k+1\}, j \in \{k, k+1\}$$

and

$$b_i = \int_{x_k}^{x_{k+1}} f\phi_i, \quad i \in \{k, k+1\}$$

Note that the local matrix has dimension 2×2. The global matrix is obtained by "adding" the elemental contributions.
To implement Dirichlet it is simpler to assemble first the global matrix ignoring Dirichlet boundary condition and then correct the final global matrix. For instance, if we have to implement $u(0) = \alpha$ after having assembled the global matrix A and rhs $\mathbf{b}$ ignoring it you can modify them my setting $A_{00} = 1$, $A_{01} = 1$ and $b_0 = \alpha$. This way the first equation does indeed impose $u_0 = \alpha$. If we have a Dirichlet condition on the right we operate similarly on the last line of the system of equations.

Solution of ordinary systems of equations

In the parabolic (transient) case, you have to solve a system of ordinary equations. You may choose, for instance, an explicit scheme to start with (simpler). Remember however that an implicit scheme is necessarily only conditionally stable. It means that your solution won't blow-up only if the time step is small enough. The system of equations can be written (beware of the Dirichlet boundary term though) as

$$M\frac{d}{dt}\mathbf{u}(t) + A\mathbf{u}(t) = \mathbf{b}_j(t), \tag{2}$$

You can write it in the standard "out of the book" form

$$\frac{d}{dt}\mathbf{u}(t) = \mathbf{F}(t, \mathbf{u}(t))$$

by setting $\mathbf{F}(t, \mathbf{y}) = M^{-1}(\mathbf{b}_j(t) - A\mathbf{u})$.
If we use an explicit RK4 method (with the limitation just indicated), at each time step, of length h, we compute

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{u}_n), \\
\mathbf{k}_2 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{u}_n + \frac{h}{2}\mathbf{k}_1\right), \\
\mathbf{k}_3 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{u}_n + \frac{h}{2}\mathbf{k}_2\right), \\
\mathbf{k}_4 &= \mathbf{f}\left(t_n + h, \mathbf{u}_n + h\mathbf{k}_3\right).
\end{aligned}$$

and

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

Each stage rewires to solve a linear system in the M matrix. To simplify the problem. you can use the "lumping" technique, replacing the M with L, where L is diagonal and $L_{ii} = \sum_j M_{ij}$.

1 What to do for the hansdon

You can do the steady state version and then, if you have time, try either to implement the transient case or try to implement a parallel version of the Thomas algorithm (see given biblography).

1.1 Extension for a (possible) exam project

You can move to the 3D case and/or add a transport term, trying to make a small library for finite elements, generalizing the construction of the various differential operatora at discrete level. Parallelization is also possible, using parallel linear algebra.

Bibliographic material

This is the bibliography material provided

- A chapter of the book "Calcolo Scientifico" (in Italian). Look at the chapter that describe one dimensional finite elements.
- A Master thesis on the parallel cyclic reduction algorithm for tridiagonal systems. Look only the relevant chapter, the rest is irrelevant.
- Any good introductory book on PDE and finite elements (you find dozens in the web).