Lab Assignment Bonus
The Finite Element Method (Part 2)

ACS II

Spring2020

Due May 1, 2020

Introduction

Last week we looked at the finite element method for solving elliptic ODEs. Today we will look at using it to solve one dimensional time-dependent PDEs.

1 Heat Equation

We will begin by considering the 1D heat equation on the unit interval with homogeneous Dirichlet boundary conditions,

$$\begin{aligned} u_t - u_{xx} &= f(x,t), \qquad 0 \le x \le 1,\ t \ge 0 \\ u(0,t) &= u(1,t) = 0, \\ u(x,0) &= u_0(x), \end{aligned} \tag{1}$$

where $f(x,t)$ is a given forcing function and $u_0(x)$ is a given initial condition.

1.1 Weak Formulations

By the same process we discussed last week, we can turn (1) into a corresponding weak problem:

$$\begin{aligned} &\text{Seek } u(x,t) \in H_0^1[0,1] \text{ such that for all } v(x) \in H_0^1[0,1] \text{ and } t \ge 0: \\ &\int_0^1 u_t(x,t)v(x)\ \mathrm{d}x + \int_0^1 u_x(x,t)v_x(x)\ \mathrm{d}x = \int_0^1 f(x,t)v(x)\ \mathrm{d}x. \end{aligned} \tag{2}$$

Note that while the solution $u(x,t)$ is a function of time and space, our test function v is a function of space only $\Rightarrow v_x = v'(x)$.

To turn (2) into a discrete weak formulation, we adopt the same approach as for the Sturm-Liouville equation. As in that case, we'll confine u and v to a subspace V^h of $H_0^1[0,1]$ to get a discrete weak formulation:

$$\begin{aligned} &\text{Seek } u^h(x,t) \in V^h \text{ such that for all } v^h \in V^h \text{ and } t \geq 0: \\ &\int_0^1 u_t^h(x,t)v^h(x)\ \mathrm{d}x + \int_0^1 u_x^h(x,t)v_x^h(x)\ \mathrm{d}x = \int_0^1 f(x,t)v^h(x)\ \mathrm{d}x. \end{aligned} \tag{3}$$

To turn this into a solvable system of equations, we will make the ansatz

$$u^h(x,t) = \sum_{j=1}^{N} c_j(t)\phi_j(x), \tag{4}$$

where $\{\phi_j(x)\}_{j=1}^N$ form a basis for V^h and $\{c_j(t)\}_{j=1}^N$ are the corresponding coefficients. To make this concrete, we will take V^h to be the space of all piecewise linear functions that satisfy the homogeneous Dirichlet boundary conditions. We will take $\{\phi_j(x)\}$ to be the "hat" basis functions from the last lab. Unlike the Sturm-Liouville equation, the coefficients are now time dependent. Since (3) holds for any $v^h \in V^h$ it must hold for all $\phi_i(x)$. Plugging (4) into (3) and enforcing it for all the basis functions leads to the system,

$$\int_0^1 \sum_{j=1}^{N} c_j'(t)\phi_j(x)\phi_i(x)\ \mathrm{d}x + \int_0^1 \sum_{j=1}^{N} c_j(t)\phi_j'(x)\phi_i'(x)\ \mathrm{d}s = \int_0^1 f(x,t)\phi_i(x)\ \mathrm{d}x,$$
$$i = 1, \ldots, N, \quad t \geq 0.$$

This can be rewritten as,

$$\mathcal{M}\mathbf{c}' + \mathcal{S}\mathbf{c} = \mathbf{f}(t), \tag{5}$$

where

$$\begin{aligned} &\mathcal{M}_{ij} = \int_0^1 \phi_i(x)\phi_j(x)\ \mathrm{d}x, \qquad \mathcal{S}_{ij} = \int_0^1 \phi_i'(x)\phi_j'(x)\ \mathrm{d}x, \\ &\mathbf{c}' = [c_1'(t), \ldots, c_N'(t)], \quad \mathbf{c} = [c_1(t), \ldots, c_N(t)], \qquad \mathbf{f}_i(t) = \int_0^1 f(x,t)\phi_i(x)\ \mathrm{d}x. \end{aligned} \tag{6}$$

1.2 Time Discretization

This is a system of ODEs to solve for the coefficients. To solve this, we must discretize $\mathbf{c}$ in time somehow. We will use backward (implicit) Euler. This lets us write $\mathbf{c}'$ at time step n as

$$\mathbf{c}' = \frac{\mathbf{c}^n - \mathbf{c}^{n-1}}{\Delta t},$$

where $\mathbf{c}^n = [c_1(n\Delta t), \ldots, c_N(n\Delta t)]$

Using this approximation for $\mathbf{c}'$ allows us to rewrite (5) as

$$(\mathcal{M} + \Delta t\mathcal{S})\mathbf{c}^n = \Delta t\mathbf{f}(n\Delta t) + \mathcal{M}\mathbf{c}^{n-1}, \qquad n = 1, 2, 3 \ldots. \tag{7}$$

Thus given $\mathbf{c}^0$ we can compute $\mathbf{c}^1$, then $\mathbf{c}^2$ etc, by solving a linear system at each time step. But what is $\mathbf{c}^0$? This comes from our initial condition $u_0(x)$. If we are using piecewise linear basis functions, we can get $\mathbf{c}^0$ by directly evaluating $u_0(x)$ at the discretization nodes, i.e.

$$\mathbf{c}_j^0 = u_0(x_j).$$

1.3 Algorithm

The procedure for assembling $\mathcal{M}$, $\mathcal{S}$ and $\mathbf{f}$ are identical to the last lab (take $p(x) = q(x) = 1$). The matrices $\mathcal{M}$ and $\mathcal{S}$ can be assembled at the beginning of the routine, since these do not change each time step. The right hand side $\mathbf{f}$ however has to be recomputed each time step, since in general f might depend on t. The algorithm to solve the heat equation using finite elements is outlined in Algorithm 1.

Data: number of intervals N;
functions $f(x,t)$, $u_0(x)$;
time step size Δt;
time horizon T
Result: $\{\mathbf{c}^k\},\ k = 0, \ldots, T/\Delta t$
initialize :
setup geometry;
assemble $\mathcal{M}$ and $\mathcal{S}$ according to algorithm in last lab;
$\mathbf{c}_j^0 \leftarrow u_0(x_j)$;
$n \leftarrow 1$;
while $n\Delta t \leq T$ **do**
    assemble $\mathbf{f}(n\Delta t)$ according to algorithm in last lab;
    solve $(\mathcal{M} + \Delta t\mathcal{S})\mathbf{c}^n = \Delta t\mathbf{f}(n\Delta t) + \mathcal{M}\mathbf{c}^{n-1}$ for $\mathbf{c}^n$;
    $n \leftarrow n + 1$;
end

Algorithm 1: Solving the heat equation using finite elements and backwards Euler.

1.4 Deliverable

Solve (1) using 64 elements up to $T = 8$. Use $\Delta t = 0.1$ with $u^0(x) = \sin(\pi x)$ and $f(x,t) = \sin(\pi x)(\pi^2\cos(t) - \sin(t))$. The exact solution in this case is $u(x,t) = \sin(\pi x)\cos(t)$ Plot your solution and the exact solution at $t = 0$, $t = 2$, $t = 4$ and $t = 8$.

2 Implicit-Explicit Time Stepping

When using backward Euler for the heat equation we end up with a linear system to solve at each time step. If our PDE is nonlinear, we end up with a nonlinear system to solve at each time step. This is computationally expensive. One way around this is to use implicit-explicit (IMEX) time stepping where we treat the linear terms implicitly and the nonlinear terms explicitly.

2.1 Burgers' Equation

The 1D Burgers' equation on the unit interval with homogeneous Dirichlet boundary conditions is

$$\begin{aligned} u_t - u_{xx} + uu_x &= f(x,t), \qquad 0 \le x \le 1, \quad t \ge 0, \\ u(0,t) = u(1,t) &= 0, \\ u(x,0) &= u_0(x), \end{aligned} \tag{8}$$

where again $u_0(x)$ is a given initial condition and $f(x,t)$ is a given forcing function. Following the usual steps we can convert this into a weak problem:

Seek $u(x,t) \in H_0^1[0,1]$ such that for all $v(x) \in H_0^1[0,1]$ and $t \ge 0$:

$$\int_0^1 u_t(x,t)v(x)\ \mathrm{d}x + \int_0^1 u_x(x,t)v_x(x)\ \mathrm{d}x + \int_0^1 u(x,t)u_x(x,t)v(x)\ \mathrm{d}x = \int_0^1 f(x,t)v(x)\ \mathrm{d}x,$$

and a discrete weak problem:

$$\begin{aligned} &\text{Seek } u^h(x,t) \in V^h \text{ such that for all } v^h(x) \in V^h \text{ and } t \ge 0: \\ &\int_0^1 u_t^h(x,t)v^h(x)\ \mathrm{d}x + \int_0^1 u_x^h(x,t)v_x^h(x)\ \mathrm{d}x + \int_0^1 u^h(x,t)u_x^h(x,t)v^h(x)\ \mathrm{d}x \\ &\qquad = \int_0^1 f(x,t)v^h(x)\ \mathrm{d}x. \end{aligned} \tag{9}$$

The first two terms are identical to the first two terms in the weak form of the heat equation. If we let $u^h = \sum_{j=1}^N c_j(t)\phi_j(x)$ as before and enforce (9) for all ϕ_i we end up with the system,

$$\begin{aligned} \int_0^1 \sum_{j=1}^N c_j(t)'\phi_j(x)\phi_i(x)\ \mathrm{d}x &+ \int_0^1 \sum_{j=1}^N c_j(t)\phi_j'(x)\phi_i'(x)\ \mathrm{d}x \\ &+ \int_0^1 \left(\sum_{j=1}^N c_j(t)\phi_j(x)\right)\left(\sum_{j=1}^N c_j(t)\phi_j'(x)\right)\phi_i(x)\ \mathrm{d}x \\ &= \int_0^1 f(x,t)\phi_i(x)\ \mathrm{d}x, \qquad i = 1,\ldots,N \quad t \ge 0. \end{aligned}$$

We could use backward Euler exactly as before to end up with a system of *nonlinear* systems to solve at each time step. An alternative approach is to use backward Euler for the time derivative, but treat the nonlinear term explicitly and move it to the right hand side.

Doing so leads to the sequence of *linear* problems,

$$(\mathcal{M} + \Delta t \mathcal{S})\mathbf{c}^n = \Delta t \mathbf{f}(n\Delta t) - \Delta t \mathbf{b} + \mathcal{M}\mathbf{c}^{n-1}, \qquad n = 1, 2, \ldots, \tag{10}$$

where

$$\mathbf{b}_i = \int_0^1 u^{n-1} u_x^{n-1} \phi_i(x) \, \mathrm{d}x,$$
$$u^{n-1} = \sum_{j=1}^{N} c_j((n-1)\Delta t)\phi_j(x), \qquad u_x^{n-1} = \sum_{j=1}^{N} c_j((n-1)\Delta t)\phi_j'(x).$$

2.2 Algorithm

To solve Burgers' equation we must evaluate $\mathbf{b}$. To do this requires quadrature. As with the matrix assembly we will split this integral over each element. This has the advantage that on a particular element we only need to consider basis functions that are non-zero over that element. The algorithm to compute $\mathbf{b}$ is described in Algorithm 2.

Data: number of intervals N, coefficients $\mathbf{c}^{n-1}$
Result: $\mathbf{b}$
initialize :
$\mathbf{b} = \text{zeros}(N-1, 1)$;
for *each element* I_k **do**
    get nonzero basis functions on I_k;
    get quadrature points and weights $\{x_\ell\}_{\ell=1}^{L}$ and $\{w_\ell\}_{\ell=1}^{L}$ on I_k;
    for $\ell = 1, \ldots, L$ **do**
        $u^{n-1} \leftarrow 0$;
        $u_x^{n-1} \leftarrow 0$;
        /* Compute u^{n-1} and u_x^{n-1} at quadrature point */
        for *each nonzero basis function on* I_k **do**
            $i \leftarrow$ index of basis function;
            $u^{n-1} \leftarrow u^{n-1} + \mathbf{c}_i^{n-1}\phi_i(x_\ell)$;
            $u_x^{n-1} \leftarrow u_x^{n-1} + \mathbf{c}_i^{n-1}\phi_i'(x_\ell)$;
        end
        /* update $\mathbf{b}_i$ */
        for *each nonzero basis function on* I_k **do**
            $i \leftarrow$ index of basis function;
            $\mathbf{b}_i \leftarrow \mathbf{b}_i + u^{n-1}u_x^{n-1}\phi_i(x_\ell)w\ell$;
        end
    end
end

Algorithm 2: Evaluating the nonlinear term in Burgers' equation.

The algorithm to solve Burgers' equation is similar to Algorithm 1. Every time step we must now compute $\mathbf{b}$ and $\mathbf{f}$.

Data: number of intervals N;
function $u_0(x)$;
time step size Δt;
time horizon T
Result: $\{\mathbf{c}^k\},\ k = 0, \ldots, T/\Delta t$
initialize :
setup geometry;
assemble $\mathcal{M}$ and $\mathcal{S}$ according to algorithm in last lab;
$\mathbf{c}_j^0 \leftarrow u_0(x_j)$;
$n \leftarrow 1$;
while $n\Delta t \leq T$ **do**
    assemble $\mathbf{f}(n\Delta t)$ according to algorithm in last lab;
    assemble $\mathbf{b}$ according to Algorithm 2;
    solve $(\mathcal{M} + \Delta t\mathcal{S})\mathbf{c}^n = \Delta t\mathbf{f}(n\Delta t) - \Delta t\mathbf{b} + \mathcal{M}\mathbf{c}^{n-1}$ for $\mathbf{c}^n$;
    $n \leftarrow n + 1$;
end

Algorithm 3: Solving Burgers' equation using finite elements and IMEX time stepping.

2.3 Deliverable

Solve (8) using 64 elements up to $T = 8$. Use $\Delta t = 0.1$ with $u^0(x) = \sin(\pi x)$ and $f(x,t) = e^{-t}\sin(\pi x)(\pi e^{-t}\cos(\pi x) + \pi^2 - 1)$. The exact solution in this case is $u(x,t) = e^{-t}\sin(\pi x)$. Plot your solution and the exact solution at $t = 0$, $t = 2$, $t = 4$ and $t = 8$.

2.4 Allen-Cahn Equation

The 1D Allen-Cahn equation on the unit interval with homogeneous Dirichlet boundary conditions is

$$\begin{aligned} u_t - \alpha^2 u_{xx} &= u(1-u^2), \qquad 0 \leq x \leq 1 \quad t \geq 0, \\ u(0,t) &= u(1,t) = 0, \\ u(x,0) &= u_0(x), \end{aligned} \tag{11}$$

where $\alpha > 0$ is a constant and u_0 is, as always, the given initial condition.

2.4.1 Deliverable

1. Write out the weak formulation for (11)

2. Using backward Euler IMEX, write out the linear system that must be solved at each time step

3. Solve this sequence of problems using piecewise linear basis functions. Split you domain into 512 elements and for your initial condition use random noise between -4 and 4 (but make sure u_0 satisfies the boundary conditions $u_0(0) = u_0(1) = 0$). Using a time step size of $\Delta t = 0.1$ plot your solution at $t = 20$ for the following values of $\alpha : 0.1,\ 0.01,\ 0.001$.

Submission and Grading

To get credit for this assignment, you must submit the following information to Canvas by 11:59pm, May 1, 2020

- your source code files
- Your report as a single file in pdf format, including results from your work and relevant discussion of your observations, results, and conclusions.

This information must be received by 11:59pm, May 1, 2020. Upload the required documents to Canvas. As stated in the course syllabus, late assignment submissions will be subject to a 10% point penalty per 24 hours past the due date at time of submission, to a maximum reduction of 50%, according to the formula:

*[final score] = [raw score] - min(0.5, 0.1 * [# of days past due]) * [maximum score]*