Projects Scie-Computing 2019-20

Leszek Marcinkowski

May 23, 2020

In case of errors, typos please inform me.

1 Introduction

I present two problems modeling the same situation - stationary i.e. modeling the situation after the temperature stabilize and evolutionary - time dependent.

The easiest is 1D stationary with finite difference discretization. And this one is really recommended unless somebody is more ambitious or interested in doing more. Naturally, if one picks more complicated version I will be more lenient in the evaluating process - but anyway any not finished project will be not be graded high good (by many points) so any version you choose try to make it working.

2 Optimal control problem

A room $(0, a_1) \times (0, a_2)$ with the height $H > 0$ is heated by setting a temperature on a part pf the wall. We want to know how to control the temperature in a part of the room by changing the temperature of the heater. The temperature is controlled by a given PDE. We have to set the boundary values such that the temperature in the room is as close as possible to a given function.

We have two cases - stationary and time-dependent. The simplest is 1D version where we assume that we heat one wall and control the temperature along the room (so the temperature equals $u(x)$ where x is the distance from the wall).

We can use different discretizations for given models: I propose to use standard finite difference method (FDM) (the domain is an interval, rectangle (or a cuboid in 3D - but I stick to 1 or 2D)). So we can use equidistant mesh in each direction.

2.1 Stationary case

Let $\Omega = \Pi_{k=1}^{d}(0, a_k)^d$ $d = 1, 2$, we heat $\Gamma_1 = \{0\}$ in 1D or

$$\Gamma_1 = \{(0, s) : 0 \leq s \leq a_2\},$$

we assume that u satisfies:

$$-Lu - S(u) = -\sum_{k=1}^{d} \frac{\partial}{\partial x_k} \mathrm{A}_k(t,x) \frac{\partial}{\partial x_k} u(x) - S(u) = f(x) \qquad x \in \Omega \tag{1}$$

$$\lambda \frac{\partial}{\partial n} u(s) + u(s) = g(s) \qquad x \in \Gamma_1 \tag{2}$$

$$\frac{\partial}{\partial n} u(s) = 0 \quad x \in \Gamma_2 = \partial\Omega \setminus \Gamma_1 \tag{3}$$

Let simplify the model and set $\mathrm{A}_k(t,x) = \alpha > 0$ a positive constant, λ is also a positive constant, $S(u)$ a given function (we should be able to use any univariate function, u - the solution depends on the function g defined over Γ_1, we assume $g_{min} \le g(s) \le g_{max}$. We want to compute g_0 such that

$$F(g_0) = \min_g F(g) \quad F(g) = \int_D |u(x) - \mathrm{H}(x)|^2 dx$$

where H is a given function defined over $D = \Pi_{k=1}^{d}[b_k, a_k]$, here $0 \le b_k < a_k$, further for simplicity may assume that $a_k = m * b_k$ for some positive integer m - it is a technical assumption simplifying computations in the case FDM discretization on a uniform mesh.

Summing up - we define the function $F(g) = \int_D |u(x) - \mathrm{H}(x)|^2 dx$ with u solving the BVP and we would like to find g_0 minimizing this function.

Possible discretizations:

- FDM - the derivative in the Robin bnd condition may be approximated by the simplest 2 point finite difference (forward or backward) - that's the simplest and recommended project
- linear continuous finite element method (FEM)
- quadratic continuous finite element method (FEM) (more difficult)
- another method e.g. spectral collocation method or a finite volume method (only if you are really interested in those methods or for some reason know them well or have to implement one of them anyway)

Note that $g \in [g_{min}, g_{max}]$ is just a scalar so our minimizing problem is also in 1D over a closed bounded interval, but in 2D we minimize over a set of bounded (by g_{min} from below or g_{max} from above) continuous functions defined over Γ_1... In practice we have to discretize this set too, the simplest straightforward approach in the case of FDM discretization is to take all the set of discrete

$$\{g_h : \Gamma_{1,h} \to R : g_{min} \le g_h \le g_{max}\}.$$

We also have to discretize the function F, i.e. use some quadrature rule. In the FDM case we could use a prolongation: $p_h : U_h \to C(\overline{\Omega})$, where U_h is the FDM discrete mesh space in our FDM discretization, e.g. we can interpolate u_h the FDM solution at the mesh points getting a bilinear function, or the simplest just approximate the integral of the rhs of F by a quadrature which uses only the values of H, u at the mesh points of the FDM method, e.g. we can use a trapezoidal quadrature rule (once in 1D or twice in 2D): in 1d we might define:

$$F_h(g) = \sum_{k=l+1}^{N-1} h|\mathrm{H}(x_k) - u_h(x_k)|^2 + 0.5h|\mathrm{H}(x_l) - u_h(x_l)|^2 + 0.5h|\mathrm{H}(x_N) - u_h(x_N)|^2$$

where $x_k = h * k$ with $h = a_k/N$, we assume that $b_1 = x_l = l * h$. Here u_h is a discrete FDM solution of the discrete problem defined over the mesh: $\overline{\Omega}_h = \{x_k\}_{k=0}^{N}$.

2.2 Non-stationary case

We heat the same part of the boundary, cf. Section 2.1 and that $u(t,x)$ satisfies: equation:

$$\frac{\partial}{\partial t}u - \sum_{k=1}\frac{\partial}{\partial x_k}\mathrm{A}_k(t,x)\frac{\partial}{\partial x_k}u(t,x) - S(u) = 0 \qquad x \in \Omega \quad t \in (0, T_{Max}) \tag{4}$$

$$\lambda\frac{\partial}{\partial n}u(t,s) + u(t,s) = g(t,s) \qquad x \in \Gamma_1 \quad t \in (0, T_{Max}) \tag{5}$$

$$\frac{\partial}{\partial n}u(t,s) = 0 \quad x \in \Gamma_2 = \partial\Omega \setminus \Gamma_1 \quad t \in (0, T_{Max}) \tag{6}$$

S, a_k are the same as in Section 2.1

u is controlled by $g^*(t,x)$ defined for $x \in D$ s.t. $g_{min} \le g^* \le g_{max}$ and

$$F(g^*) = \min_g F(g) \quad F(g) = \int_0^{T_{max}}\int_D |u(t,x) - \mathrm{H}(t,x)|^2 dx\, dt$$

where H is a given function defined on $(0, T_{max}) \times D$ for $D = \Pi_{k=1}^d [b_k, a_k]$, $0 < b_k < a_k$ $d = 1, 2$. We will further simplify the model in 2D i.e. we take

$$g(t,(0,s)) = \begin{cases} g_0(t) & s \in [0, 0.25 * a_2] \\ g_0(t)(1 - 0.5\frac{s - 0.25*a_2}{0.75*a_2}) & s \in [0.25 * a_2, a_2] \end{cases} \tag{7}$$

Thus really we are looking for $g_0 : (0, T_{Max}) \to R$ which minimizes F. In 1D we also have that $g(t,0) = g_0(t)$ so we have to find analogous g_0.

We introduce a space discretization (FDM or FEM) obtaining a IVP which we solve

- using any octave solver e.g. **lsode**() (default)
- using implicit Euler or Crank-Nicholson scheme

In case of using **lsode**() we have to define $g_0(t)$ for any time $t \le T_{Max}$ even if we set discrete times since the octave solver may need to compute the vector field also at auxiliary times. The most reasonable is to take g_0 as a linear spline defined by the values at our discrete times (e.g. equidistant points on $[0, T_{MAX}]$. So our unknowns will be the values of g_0 at discrete times. The other option is to look for g_0 in a polynomial space of not too large degree or to take a linear spline but with fewer nodes e.g .we can take every discrete time point which differ by a constant time span e.g. 3 seconds, as a node defining a linear spline in t. Paradoxically, it is reasonable as the device requires some time to switch from e.g. heating to cooling.

2.2.1 Tests

Please test your code for $a_1 = 2, a_2 = 4$, $b_1 = 1, b_2 = 2$, $\mathrm{A}_k = \alpha \equiv 1$, $S(u) \equiv 0$ with $f = 2$ or $S(u) = -atan(u)/\pi$ z $f = 0$, $\lambda = 1$, $\mathrm{H} \equiv 20$. In 1D case take $d = 1$ $a_1 = 2, b_1 = 1$.

One can show me the code - how it works and write a 2-3 pages long report, describing the discretization and used tools (e.g. what optimization or ODE solver were used etc). You should also test parts of the solver e.g. if the solver for the respective PDE works for some known solutions etc. I can ask that during the exam.