Optimization Problems

Conor Rowan

Spring 2024

1 Inverse Problems

1.1 Static, Finding Boundary Condition

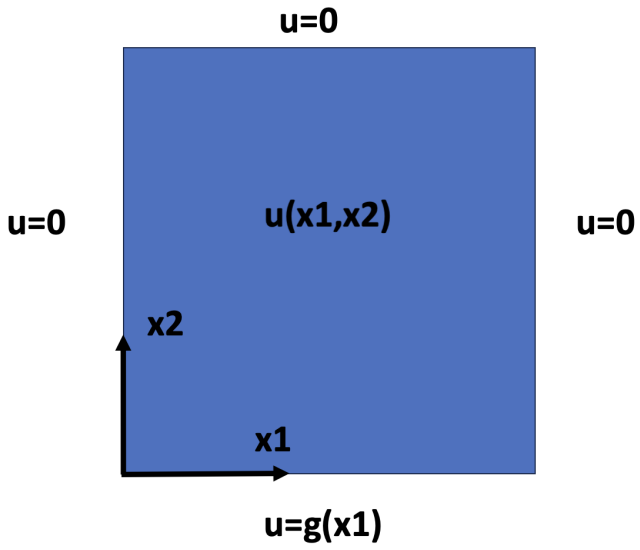

Figure 1: Set-up for heat conduction inverse problem. Three of the four boundaries are zero, and the prescribed temperature distribution is chosen to optimize some objective.

Let's work with heat conduction on a square domain. Three of the four boundaries will have zero temperature, and the fourth will have a spatially varying prescribed temperature. See Figure 1 for the setup. The goal of this problem is to find the temperature boundary condition which optimizes some objective which depends on the temperature field inside the domain. This is called an inverse problem because we are finding the boundary condition (forcing) from

the temperature field, as opposed to the temperature from the boundary condition. One common example of this is using measurements of the temperature inside a domain to determine an unknown boundary condition. As an example, this problem might arise if it is possible to accurately measure the temperature of a flow inside a pipe but not the temperature along the outside of the pipe. There are situations in which it is useful to infer the boundary condition from the solution, as one may be easier to measure than the other. A more academic problem is to simply choose some temperature $u(x_1, x_2)$ and see how closely it can be matched with an optimal choice of boundary condition. For now, we will leave the objective unspecified, noting only that it depends on the temperature inside the domain. The inverse problem can be stated as

$$\underset{\underline{\theta}}{\text{argmin}}\ z\Big(u(x_1, x_2; \underline{\theta})\Big) \quad \text{s.t.}\ \frac{\partial u}{\partial x_i \partial x_i} = 0, \quad u|_{\Gamma_1} = g(x_1; \underline{\theta}), \quad u|_{\Gamma_{i\neq 1}} = 0$$

This says that we want minimize the objective z, which depends on the temperature inside the domain. Because the temperature must satisfy the governing equation, it is a function of the parameters/optimization variables $\underline{\theta}$ which construct the one non-zero boundary condition. The boundary condition $g(x_1)$ is discretized with the parameters $\underline{\theta}$. The first step in solving this problem is to solve the governing equations for a given g numerically. There are no heat sources in the domain, and all boundary conditions are Dirichlet. The weak form of the governing equation is then

$$\int \nabla u \cdot \nabla w d\Omega = 0$$

where the test function w is zero over all the boundaries. We discretize the test function with

$$w = \sum_j w_j f_j(x_1, x_2)$$

which after plugging into the governing equation and noting that the coefficients w_j are arbitrary, we obtain the system

$$\int \nabla u \cdot \nabla f_j d\Omega = 0$$

We discretize the temperature and build in the boundary condition g. This ensures that it is automatically satisfied. Note that adding in a function which ensures that the nonzero boundary condition is satisfied must respect the other three boundary conditions, i.e. it must be zero along these three sides. There are a number of ways of accomplishing this, one of which is

$$u(x_1, x_2) = g(x_1, \underline{\theta}) \sin(\pi x_1)(1 - x_2) + \sum_i u_i f_i = \tilde{g}(x_1, x_2; \underline{\theta}) + \sum_i u_i f_i$$

Note that we have assumed the domain is square with side length 1. The sine function enforces that the temperature is zero along the two x_1 edges, and $(1 - x_2)$ ensures that the boundary condition is zero at the x_2 edge. Note that the shape functions f_i are all zero along the boundaries. The temperature discretization can be plugged into the weak form to obtain

$$u_i \int \nabla f_i \cdot \nabla f_j d\Omega = - \int \nabla \tilde{g} \cdot \nabla f_j d\Omega$$

Evaluating these integrals allows us to write this in matrix-vector form as

$$K_{ij} u_j = F_i(\underline{\theta})$$

With the discretized governing equation in hand, we can rephrase the optimization problem as

$$\underset{\underline{\theta}}{\text{argmin}} \; z\Big(u(x_1, x_2; \underline{\theta})\Big) \quad \text{s.t. } K_{ij} u_j = F_i(\underline{\theta})$$

Now we need to compute the gradient of the objective with respect to the parameters to feed into an optimization algorithm. This can be accomplished by differentiating the objective:

$$\frac{dz}{d\theta_m} = \frac{\partial z}{\partial u} \frac{\partial (\sum_i u_i f_i)}{\partial u_k} \frac{\partial u_k}{\partial \theta_m} = \frac{\partial z}{\partial u} f_k \frac{\partial u_k}{\partial \theta_m}$$

The derivative of the objective with respect to the temperature is something which can be computed analytically. Note that we have an additional step in the chain rule because of the global basis, which arises from the fact that the displacement degrees of freedom do not correspond to actual values of the displacement. The derivative of the degrees of freedom with respect to the parameters is called the sensitivity, and is computed using the discrete governing equation

$$\frac{\partial}{\partial \underline{\theta}} \Big(\underline{\underline{K}} \underline{u} = \underline{F} \Big) \implies \frac{\partial \underline{u}}{\partial \underline{\theta}} = \underline{\underline{K}}^{-1} \frac{\partial \underline{F}}{\partial \underline{\theta}}$$

Thus the sensitivity can be written as

$$\frac{dz}{d\theta_m} = \frac{\partial z}{\partial u} f_k K^{-1}_{kj} \frac{\partial F_j}{\partial \theta_m}$$

This expression is slightly misleading because it has a term f_k which is a function of the spatial coordinates. This does not make sense entering into the gradient of the objective. In reality, the objective takes a form like

$$z = \int q\Big(u(x_1, x_2; \underline{\theta}\Big) d\Omega$$

so that the gradient is the same as stated above, but inside the integral:

$$\frac{dz}{d\theta_m} = \left(\int \frac{\partial q}{\partial u} f_k d\Omega \right) K^{-1}_{kj} \frac{\partial F_j}{\partial \theta_m}$$

The derivative $\partial F_j / \partial \theta_m$ can be computed analytically, with finite differencing, or using automatic differentiation.

1.2 Static, Finding Material

In the previous problem, some objective was defined on the temperature field and a temperature boundary condition was found to optimize this objective. We can re-use much of this analysis to solve a new inverse problem, which amounts to high-tech parameter estimation. Say that we have a specified temperature boundary condition $g(x)$ with the same problem setup as before, and a known temperature field $u(x_1, x_2)$. The goal is to estimate the conductivity which produces this temperature field. Assuming isotropic heat conduction, the governing equation would be

$$u_i \int a(x_1, x_2; \underline{\theta}) \nabla f_i \cdot \nabla f_j d\Omega = - \int \nabla \tilde{g} \cdot \nabla f_j d\Omega$$

Thus, the stiffness matrix depends on the unknown parameters. The optimization problem can be stated as

$$\underset{\underline{\theta}}{\text{argmin}} \ \frac{1}{2} \int \Big(u(x_1, x_2; \underline{\theta}) - T(x_1, x_2) \Big)^2 d\Omega \quad \text{s.t. } K_{ij}(\underline{\theta}) u_j = F_i$$

where T is the known temperature field inside the domain which respects the boundary conditions. The gradient of this quantity is

$$\frac{dz}{d\theta_m} = \left(\int \Big(u(x_1, x_2; \underline{\theta}) - T(x_1, x_2) \Big) f_k d\Omega \right) \frac{\partial u_k}{\partial \theta_m}$$

The sensitivity is computed by differentiating the governing equations. In this case, the stiffness matrix depends on the parameters, not the force vector. We find that

$$\frac{\partial u_k}{\partial \theta_m} = -K^{-1}_{k\ell} \frac{\partial K_{\ell j}}{\partial \theta_m} u_j$$

The gradient of the objective is then

$$\frac{dz}{d\theta_m} = -\left(\int \Big(u(x_1, x_2; \underline{\theta}) - T(x_1, x_2) \Big) f_k d\Omega \right) K^{-1}_{k\ell} \frac{\partial K_{\ell j}}{\partial \theta_m} u_j$$

Note that there are not necessarily any constraints in these problems because the governing equations are enforced through the chain rule by noting that the solution has implicit dependence on the parameters.

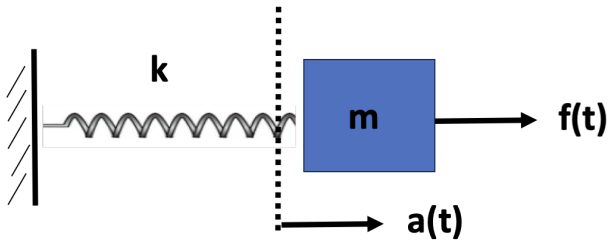

Figure 2: Mass and spring system driven by an external force.

1.3 Dynamic, Finding Forcing

An inverse problem in the context of dynamics seems to be very similar to problems in control theory. For example, we might want to find an external force $f(t)$ that gives the mass a certain trajectory $a(t)$ for given initial conditions. This is like optimally "controlling" the mass-spring system. See Figure 2 for the problem setup. The governing equation is

$$m\ddot{a} + ka(t) = f(t)$$

where $a(t)$ is the time varying position of the mass. Let's say we want to solve the control problem, where the force is chosen such that the mass follows a specified trajectory $g(t)$. The objective is then

$$z = \frac{1}{2}\int_0^T \Big(a(t) - g(t)\Big)^2 dt$$

As in the previous problems, the function to be determined by the optimization process is parameterized in terms of $\underline{\theta}$. This ensures that the solution has implicit dependence on these parameters. We want to minimize the objective while satisfying the constraint of the governing equation. Differentiating the objective, we find that

$$\frac{dz}{d\theta_m} = \int_0^T \Big(a(t;\underline{\theta}) - g(t)\Big)\frac{\partial a}{\partial \theta_m} dt$$

As before, we can compute the sensitivity by differentiating the governing equation, which in the case of dynamics is not discretized. By changing the order of differentiation, we have that

$$m\frac{\partial^2}{\partial t^2}\left(\frac{\partial a}{\partial \theta_m}\right) + k\frac{\partial a}{\partial \theta_m} = \frac{\partial f}{\partial \theta_m}$$

The sensitivities can be computed by solving a system of ODE's. What are the initial conditions on this system? Given the physical interpretation of

$\partial a/\partial\theta$ as how much the solution changes with the force parameters at a given time, the initial conditions should both be zero. This is because the initial state and velocity of the solution should not be sensitive at all to the force, given that these initial conditions are specified in the problem statement itself. So the sensitivities are computed with zero initial position and velocity.

2 Density-based Optimization with Global Basis

We want to perform topology optimization on a 2D linearly elastic structure. Before we can do this, we need to outline a method for a forward solve of the problem. Each component of the displacement will be discretized with the same set of global shape functions. Call this set of global shape functions $h_i(x_1, x_2)$. The two displacement components are discretized with

$$u_1 = \sum_{i=1}^{N} \theta_{1i} h_i(x_1, x_2), \quad u_2 = \sum_{j=1}^{N} \theta_{2j} h_j(x_1, x_2)$$

where the parameters θ_{ij} are the displacement degrees of freedom. This expression can easily be written in matrix form as

$$\begin{bmatrix} u_1(x_1,x_2) \\ u_2(x_1,x_2) \end{bmatrix} = \begin{bmatrix} \underline{h}^T(x_1,x_2) & \underline{0} \\ \underline{0} & \underline{h}^T(x_1,x_2) \end{bmatrix} \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \vdots \\ \hline \theta_{21} \\ \theta_{22} \\ \vdots \end{bmatrix}, \quad \underline{h}(x_1,x_2) = \begin{bmatrix} h_1(x_1,x_2) \\ h_2(x_1,x_2) \\ \vdots \\ h_N(x_1,x_2) \end{bmatrix}$$

$$\implies u_i = H_{ij}(x_1, x_2)\tilde{\theta}_j$$

where $\tilde{\theta}$ indicates that the $2 \times N$ matrix of parameters is reshaped to a vector. The matrix $\underline{\underline{H}}$ contains all the shape functions. Next, we can write the strain vector as a matrix of derivatives multiplying the displacement field.

$$\underline{\epsilon} = \begin{bmatrix} \partial/\partial x_1 & 0 \\ 0 & \partial/\partial x_2 \\ \partial/\partial x_2 & \partial/\partial x_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \implies \epsilon_k = B_{ki} H_{ij} \tilde{\theta}_j$$

The matrix $\underline{\underline{B}}$ maps the displacement field to the strain vector. Finally, we can write the stress as the constitutive matrix times the stain vector. This reads:

$$\sigma_\ell = D_{\ell k}\epsilon_k = D_{\ell k} B_{ki} H_{ij} \tilde{\theta}_j$$

Thus all quantities of interest can be written as linear transformations on the unknown displacement coefficients $\tilde{\theta}$. The potential energy is then

$$\Pi = \frac{1}{2}\int_A \sigma_\ell \epsilon_\ell dA - \int_\Gamma t_i u_i dS = \frac{1}{2}\int_A B_{\ell m} H_{mn} \tilde{\theta}_n D_{\ell k} B_{ki} H_{ij} \tilde{\theta}_j dA - \int_\Gamma t_i H_{ij} \tilde{\theta}_j dS$$

$$\implies \frac{1}{2}\tilde{\theta}_n \tilde{\theta}_j \left(\int_A B_{\ell m} H_{mn} D_{\ell k} B_{ki} H_{ij} dA \right) - \tilde{\theta}_j \left(\int_\Gamma t_i H_{ij} dS \right)$$

The two quantities in the parentheses are the stiffness matrix and force vector respectively:

$$K_{nj} = \left(\int_A B_{\ell m} H_{mn} D_{\ell k} B_{ki} H_{ij} dA \right), \quad F_j = \left(\int_\Gamma t_i H_{ij} dS \right)$$

The governing equation is found by computing a minimum of the potential energy through differentiation with respect to displacement degrees of freedom. Thus we obtain the usual linear system

$$K_{ij} \tilde{\theta}_j = F_i$$

We will work on square domains, and can easily use Fourier-type global shape functions to discretize the problem. Note that a square domain is not restrictive in the case of topology optimization, as the optimized design will be "carved out" of this ambient domain in some way. With density methods, the topology of the structure is defined implicitly through a density field $\rho(x_1, x_2) \in [0, 1]$ which continuously interpolates between solid (1) and void (0). Thus, we parameterize the density in some way and optimize the parameters of the density to minimize the design objective. The density scales the material properties of the solid at each point:

$$D_{ij}^{eff} = f\Big(\rho(x_1, x_2; \underline{d})\Big) D_{ij}$$

where $\underline{d}$ is the set of parameters used to discretize the density. The density simply multiplies the constitutive tensor. Note that the density does not necessarily linearly scale the stiffness. We want to choose a scaling such that the frequency of intermediate values of density, i.e. some ill-defined state between solid and void, is minimized. This can be accomplished by penalizing the volume of the structure through a constraint, and using

$$f(\rho) = \rho^n$$

for some $n > 1$. If the volume is computed as

$$V = \int_A \rho dA$$

but the stiffness scales like

$$D_{ij}^{eff} = \rho^n(x_1, x_2; \underline{d}) D_{ij}$$

it can be seen that when n is large and ρ takes an intermediate value between 0 and 1, there is more a penalty to the volume than there is a gain in stiffness. Only when ρ becomes very close to 1 do we realize the full extent of the stiffness gain. This method is especially effective at penalizing intermediate densities when the design objective is the strain energy of the structure

$$z(\underline{d}) = \frac{1}{2} K_{qj} \tilde{\theta}_q \tilde{\theta}_j = \frac{1}{2} \left(\int_A \rho^n(\underline{d}) B_{qm} H_{m\ell} D_{\ell k} B_{ki} H_{ij} dA \right) \tilde{\theta}_q \tilde{\theta}_j$$

In order to perform optimization, we need gradients of this quantity with respect to the density parameters. The stiffness matrix and displacements all depend on the density. We can simplify this process by using the Clapeyron theorem, which states that (in discrete form)

$$\frac{1}{2} K_{\ell j} \tilde{\theta}_\ell \tilde{\theta}_j = \frac{1}{2} F_j \tilde{\theta}_j$$

In other words, the stored strain energy equals the work done by quasi-static applied external forces. We thus reformulate the objective as

$$z(\underline{d}) = F_j \tilde{\theta}_j$$

where the factor of $1/2$ is omitted. This formulation avoids tedious expressions for the gradient of the objective. It is straightforward to obtain gradients of this quantity. We have

$$\frac{\partial z}{\partial d_m} = F_j \frac{\partial \tilde{\theta}_j}{\partial d_m}$$

The derivative of the displacement degrees of freedom with respect to the design variables is called the sensitivity, and can be computed by differentiating the governing equation:

$$\frac{\partial}{\partial d_s} \left[\left(\int_A \rho^n(\underline{d}) B_{qm} H_{m\ell} D_{\ell k} B_{ki} H_{ij} dA \right) \tilde{\theta}_q = F_j \right]$$

The force vector does not depend on the design variables. The stiffness matrix depends explicitly on the design, and the displacement has an implicit dependence. Using the product rule and rearranging, we obtain the sensitivity as

$$\frac{\partial \tilde{\theta}_q}{\partial d_s} = -K_{qj}^{-1}(\underline{d}) \left(\int_A n \rho^{n-1} \frac{\partial \rho}{\partial d_s}(\underline{d}) B_{pm} H_{m\ell} D_{\ell k} B_{ki} H_{ij} dA \right) \tilde{\theta}_p$$

The design optimization problem with a strain energy objective and volume constraint is stated as

$$\underset{\underline{d}}{argmin}\Big(\underline{F}\cdot\underline{\tilde{\theta}}(\underline{d})\Big) \quad \text{s.t. } g(\underline{d}) = \int_A \rho(\underline{d})dA - V_0 = 0$$

A standard optimization algorithm will require the gradient of the objective and constraint. We have shown how to compute the gradient of the objective. The gradient of the volume constraint is simply

$$\frac{\partial g}{\partial d_s} = \int_A \frac{\partial \rho}{\partial d_s} dA$$

We have alluded to the fact that a global basis is used in the discretization, and this could be any common spectral-type basis. We have not specified how the density will be discretized. Because we expect large gradients in the density field, indicating sharp delineations between solid and void, we need a discretization scheme that is capable of capturing complex and localized behavior. A natural choice is to use a deep neural network, which is defined globally over the domain and shows great expressivity in representing complex behavior. Another benefit of the neural network discretization is that the output can be passed through a function which ensures that the requirement $\rho \in [0, 1]$ is met. This means there is no need to enforce box constraints in the optimization process. Consider a neural network $D(x_1, x_2; \underline{d})$ which discretizes the density. Then the new function

$$\rho(x_1, x_2; \underline{d}) = \left(\frac{1}{2} - \varepsilon\right) \tanh\Big(D(x_1, x_2; \underline{d})\Big) + \left(\frac{1}{2} + \varepsilon\right)$$

automatically satisfies the constraints. The hyperparameter ϵ is a small number that ensures numerical stability.