Partial differential equations and optimization

Eddie Wadbro

Introduction to PDE Constrained Optimization, 2016

February 15–16, 2016

What is the use of optimization for PDEs?

- Weather forecasting:
  - Weather models (PDEs) need initial conditions at each spatial point
  - Only available data is limited to a set of local measurements at different times
  - Find through optimization the initial condition that best matches the given observations
- Parameter estimation: finding material properties, nondestructive testing
- Optimizing geometrical properties: shapes and topologies

Application Example

Engineering Design

- Which shape is the best?
- Which material composition is the right one?
- . . .

**Example**: Optimization of a cantilever beam. Use 50 % material while minimizing the compliance of the beam
Structure

Yesterday, we looked at problems where the objective function and the constraints were "direct functions" of the design variables. That is

\[ x \mapsto J \]

Here, we consider problems with the following structure

\[ x \mapsto u \mapsto J, \]

where \( x \) holds the design variables, \( u \) the state (solution of PDE), and \( J \) is the objective or constraint. There is often more intermediate steps...

- The discrete state space (number of design variables) can be of small of large dimension
- The state space is typically large (and has thousands or millions of degrees of freedom)
- The number of objectives and constraints is typically small

Nested and nonnested formulation—linear algebra example

- State equation: \( Au = Bx \)
- Objective function: \( j(x, u) = c^T u + \frac{1}{2}||x||^2 \)
- \( x \in \mathcal{A} \subset \mathbb{R}^m, u \in \mathbb{R}^n, n \) large, \( A \) is \( n \times n \), and \( B \) is \( n \times m \)
  - \( A \) is often called the set of admissible designs

Nonnested formulation

View the state equation as a constraint

\[
\min_{x \in \mathcal{A}, u \in \mathbb{R}^n} j(x, u) \\
\text{subject to } Au = Bx
\]

Nested and nonnested formulation—linear algebra example

- State equation: \( Au = Bx \)
- Objective function: \( j(x, u) = c^T u + \frac{1}{2}||x||^2 \)
- \( x \in \mathcal{A} \subset \mathbb{R}^m, u \in \mathbb{R}^n, n \) large, \( A \) is \( n \times n \), and \( B \) is \( n \times m \)
  - \( A \) is often called the set of admissible designs

Nested formulation

Define \( J(x) = j(x, u) \) where \( u \) solves \( Au = Bx \). View the state equation as an intermediate step

\[
\min_{x \in \mathcal{A}} J(x)
\]
Nested and nonnested formulation—Algorithms

The formulations suggest different algorithms

- **Non-nested** Both \( x \) and \( u \) are decision variables. The state equation is generally only be feasible at convergence. ("SAND"—Simulteneous ANalysis and Design)
- **Nested** Only \( x \) holds the decision variables. States \( u \) are feasible (that is, solves the state equation) at each iteration. ("NAND"—Nested ANalysis and Design)

- SAND-type algorithms solve the state (and adjoint, will be discussed later) equations simultaneously as the optimization problem.
- Potentially very fast (if only a small number of state solves is required).
- Algorithms need to be run to full convergence, otherwise non-meaningful results (non-feasible states)

In this mini-course, we will only consider NAND-type algorithms

Directional derivatives

- Need derivatives of the objective function and the constraints in terms of the decision variables
- Convenient to use directional derivatives in derivations

- Black graph, the function \( f \) and Red graph, the differential \( \delta f(x) \)
- \( \delta f(x; \delta x) \), the evaluation of the differential in direction \( \delta x \)
Directional derivatives

- \( f : A \rightarrow \mathbb{R} \)
- \( A \) a convex subset of \( \mathbb{R}^n \) or a function space (for example, bounded and square integrable functions)
- A design variation \( \delta x = \hat{x} - x \), where \( \hat{x}, x \in A \)
- If \( x \in A \), then (by convexity) \( x + s \delta x \) stays in \( A \) for all \( s \in [0, 1] \)

\[
\begin{align*}
\delta f(x; \delta x) &= \langle \delta f(x), \delta x \rangle \equiv \lim_{s \to 0^+} \frac{f(x + s \delta x) - f(x)}{s} \\
&= \begin{cases} 
\frac{df(x)}{dx} \delta x & \text{for } A \subset \mathbb{R}, \\
\nabla f(x)^T \delta x & \text{for } A \subset \mathbb{R}^n, \\
\int_{\Omega} Df(x) \delta x & \text{for } A \subset L^2(\Omega)
\end{cases}
\end{align*}
\]

- Analogous definition for functions with values in \( \mathbb{R}^n \) or any other normed space \( V \) (\( f : A \rightarrow V \))

Two ways to compute gradients—linear algebra example

- State equation
  \[ Au = x \]
- Objective function
  \[ J(x) = c^T u \]
- Differentiate
  \[ A \delta u = \delta x \]
  \[ \delta J = c^T \delta u \]
  \[ = c^T (A^{-1} \delta x) \]
  \[ = (cA^{-T})^T \delta x \]

Two ways to compute gradients—linear algebra example

\[
\delta J = \nabla J^T \delta x = \sum_i \frac{\partial J}{\partial x_i} \delta x_i = c^T \delta u
\]

\[
= c^T (A^{-1} \delta x) \quad \text{direct sensitivities}
\]
\[
= (A^{-T} c)^T \delta x \quad \text{adjoint sensitivities}
\]

Direct sensitivites

- Compute each component of \( \nabla J \) by choosing, successively, \( \delta x = e_i \)
  for \( i = 1, \ldots, n \) (all unit vectors)

- Computational complexity
  - The number of state equation solves grows with the number of design variables
  - No extra state solves when changing the objective function
Two ways to compute gradients—linear algebra example

\[ \delta J = \nabla J^T \delta x = \sum_i \frac{\partial J}{\partial x_i} \delta x_i = c^T \delta u \]

\[ = c^T (A^{-1} \delta x) \quad \text{direct sensitivities} \]

\[ = (A^{-T} c)^T \delta x \quad \text{adjoint sensitivities} \]

Adjoint sensitivities

- Compute all components of \( \nabla J \) at once by solving \( A^T \nabla J = c \)
- Computational complexity
  - The number of state equation solves is independent of the number of design variables
  - Grows with the number of objective functions (constraints)