Lecture 4: PDE-Constrained Optimization

Kevin Carlberg

Stanford University

July 31, 2009

1. Applications
2. Implementation strategy
   - Black-box NAND
   - Gradient-based NAND
     - Sensitivity analysis
   - SAND
3. Other research issues

PDE-Constrained optimization

- This lecture considers (time-independent) PDE-constrained optimization

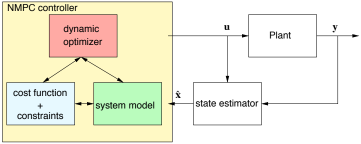
$$\begin{aligned} \underset{u\in\mathbb{R}^m, s\in\mathbb{R}^p}{\text{minimize}} \quad & f(u,s) \\ \text{subject to} \quad & c_i(u,s) = 0, \quad i = 1,\ldots,n_e \\ & d_j(u,s) \geq 0, \quad j = 1,\ldots,n_i \\ & R(u,s) = 0 \end{aligned}$$

- Time-independent PDE discretization leads to parameterized nonlinear systems of equations: $R(u,s) = 0$
- Variables split: $x = \left[u^T,\ s^T\right]^T$
- State variables: $u \in \mathbb{R}^m$ (e.g. DOF in finite element model)
- Design variables: $s \in \mathbb{R}^p$ (e.g. wing thickness)

Applications with PDE constraints

- Design optimization

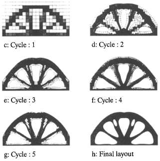
- Model predictive control Figure from R. Findeisen and F. Allgower, "An Introduction to Nonlinear Model Predictive Control," 21st Benelux Meeting on Systems and Control, 2002.



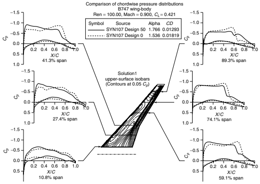
- Structural damage detection

Applications with PDE constraints

- Topology optimization (figure from K. Maute, E. Ramm, “Adaptive topology optimization,” Structural and Multidisc. Optimization, Vol. 15, No. 2, pp. 81–91, 1998)



- Aerodynamic shape optimization (figure from A. Jameson, “Aerodynamics,” Encyclopedia of Computational Mechanics, Vol. 3, pp. 325–406)



3. Redesigned Boeing 747 wing at Mach 0.90, C_p distributions.

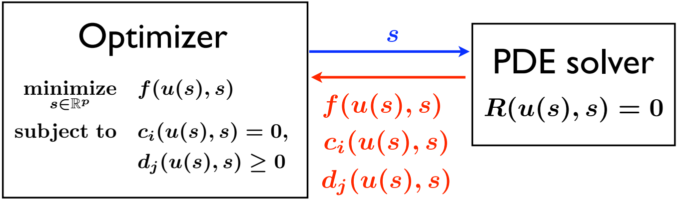
Implementation strategy

- There are two main implementation strategies:

1. *Nested Analysis and Design (NAND)*: state variables are eliminated from the optimization problem by enforcing PDE constraints to first order at each optimization iteration
   - *Black-box*: PDE solver takes in inputs and returns outputs
   - *Gradient-based*: PDE solver takes in inputs and returns outputs and output gradients
2. *Simultaneous Analysis and Design (SAND)*: PDE constraints are treated the same as any other constraint

- In order of increasing intrusiveness (and increasing efficiency): Black-box $\rightarrow$ Gradient-based NAND $\rightarrow$ SAND

Black-box NAND



- ☺ Non-invasive: can use "out-of-the-box" PDE solver and optimizer
- ☹ Since the PDE solver only returns function values, gradients are not available
- The optimizer must be:
  - a derivative-free optimization algorithm, *or*
  - a gradient-based algorithm with *finite differences*

Gradient-based NAND

Optimizer

$$\begin{aligned} \underset{s\in\mathbb{R}^p}{\text{minimize}} \quad & f(u(s),s) \\ \text{subject to} \quad & c_i(u(s),s) = 0, \\ & d_j(u(s),s) \geq 0 \end{aligned}$$

$\boldsymbol{s}$

$\boldsymbol{f(u(s),s)}$
$\boldsymbol{c_i(u(s),s)}$
$\boldsymbol{d_j(u(s),s)}$
$\boldsymbol{\nabla f(u(s),s)}$
$\boldsymbol{\nabla c_i(u(s),s)}$
$\boldsymbol{\nabla d_j(u(s),s)}$

PDE solver

$$R(u(s),s) = 0$$
+
sensitivity analysis

- ☺ Can use "out-of-the-box" gradient-based optimizer
- ☹ Somewhat invasive: must implement sensitivity analysis in PDE solver
- There are two ways to execute sensitivity analysis

Sensitivity analysis for gradient-based NAND

- Let $g_k(u(s), s)$, $k = 1, \ldots, n_i + n_e + 1$ be the optimization functions
  - $g_k = c_k$ for $k = 1, \ldots, n_e$
  - $g_k = d_{k-n_e}$ for $k = n_e + 1, \ldots n_e + n_i$
  - $g_{n_e+n_i+1} = f$
- We can differentiate $g_k(u(s), s)$ with respect to the i^{th} design variable s_i, via the chain rule

$$\frac{dg_k}{ds_i} = \frac{\partial g_k}{\partial s_i} + \frac{\partial g_k}{\partial u}\frac{du}{ds} \tag{1}$$

- Furthermore, we would like to enforce first-order consistency of the PDE: $R(u(s+\delta s), s+\delta s) = 0$ (note $R(u(s), s) = 0$)

$$R(u(s+\delta s), s+\delta s) \approx R(u(s), s) + \sum_{i=1}^{p} \frac{\partial R}{\partial u}\frac{du}{ds_i}\delta s_i + \sum_{i=1}^{p} \frac{\partial R}{\partial s_i}\delta s_i$$

$$\sum_{i=1}^{p}\left(\frac{\partial R}{\partial u}\frac{du}{ds_i} + \frac{\partial R}{\partial s_i}\right)\delta s_i = 0$$

$$\frac{du}{ds_i} = -\frac{\partial R}{\partial u}^{-1}\frac{\partial R}{\partial s_i} \tag{2}$$

  - **Jacobian**: $\frac{\partial R}{\partial u}$
- Substituting (2) into (1), we obtain

$$\frac{dg}{ds_i} = \frac{\partial g}{\partial s_i} - \frac{\partial g}{\partial u}\frac{\partial R}{\partial u}^{-1}\frac{\partial R}{\partial s_i} \tag{3}$$

Two methods for solving (3)

- **Direct sensitivity analysis**
  1. Solve $\frac{du}{ds_i} = \frac{\partial R}{\partial u}^{-1} \frac{\partial R}{\partial s_i}$ for $i = 1, \ldots, p$
  2. Cheaply compute $\frac{dg_k}{ds_i} = \frac{\partial g}{\partial s_i} - \frac{\partial g}{\partial u} \frac{du}{ds_i}$, for $k = 1, \ldots, n_e + n_i + 1$
- **Adjoint sensitivity analysis**
  1. Solve $\psi_k = \frac{\partial R}{\partial u}^{-T} \frac{\partial g}{\partial s_i}$ for $k = 1, \ldots, n_e + n_i + 1$
  2. Cheaply compute $\frac{dg_k}{ds_i} = \frac{\partial g}{\partial s_i} - \psi_k^T \frac{\partial R}{\partial s_i}$ for $i = 1, \ldots, p$
- In each case, the linear system solves (step 1) are more expensive than computing the products (step 2)

$\longrightarrow$ $p < n_e + n_i + 1$ (a few variables): direct is cheaper

$\longrightarrow$ $p > n_e + n_i + 1$ (many variables): adjoint is cheaper

SAND

Optimizer
+
PDE solver

$$\begin{aligned} \underset{u\in\mathbb{R}^m, s\in\mathbb{R}^p}{\text{minimize}} \quad & f(u,s) \\ \text{subject to} \quad & c_i(u,s) = 0, \\ & d_j(u,s) \geq 0, \\ & R(u,s) = 0 \end{aligned}$$

- The optimizer has access to the complete discretized model
- ☹ Invasive: cannot use “out-of-the-box” optimizer or PDE solver
- ☺ High efficiency: simultaneously solve the PDE and optimization problem

Other research issues for PDE-constrained optimization

- Cost reduction: expensive to repeatedly solve the PDE for NAND
- "Physics-based" globalizations: PDE solver doesn't always converge quickly in all parts of the variable space
- Jacobians $\frac{\partial R}{\partial u}$: PDE solvers use inexact Jacobians, but the optimizer needs an exact one
- Time-dependent PDE optimization: a huge number of state variables (one set for each time step) $\rightarrow$ SAND methods become infeasible