Multidisciplinary System Design Optimization (MSDO)

Approximation Methods
Lecture 19
16 April 2004

Karen Willcox

Today's Topics

• Design variable linking
• Reduced-Basis Methods
• Response Surface Approximations
• Kriging
• Variable-Fidelity Models

Why Approximation Methods?

We have seen throughout the course the constant trade-off between computational cost and fidelity.

Approximation methods provide a way to get high-fidelity model information throughout the optimization without the computational expense.

Recall that the analysis or simcode must be invoked each time the optimizer selects a new design vector to try.

Typically, hundreds (thousands) of design vectors will be analyzed throughout an optimization run.

Can use Approximation Models (Surrogate Models) for objective functions and constraints.

If approximate models are inexpensive to evaluate, can analyze many more design vector options without worrying about computational resources.

Concept first introduced in structural optimization by Barthelemy and Haftka, 1993.
Approximation Methods Overview

- **Design variable linking**
  - Reduced-basis methods
    - reduce the number of design variables in the optimization code
    - simcode analysis full order

- **Response surface methods**
  - Kriging
    - same number of design variables
    - simcode analysis simplified

- **Variable-fidelity methods**
  - combine high-fidelity and approximation models

Design Variable Linking

- Not all design variables may be independent
- For example, symmetry may exist in the problem

Define:
\[
y = Tx + y^C
\]

Optimizer uses \(y\), but provides \(x\) to simcode for analysis

Reduced-Basis Methods

Consider \(r\) feasible design vectors: \(x^1, x^2, ..., x^r\)

We could consider the desired design to be a linear combination of these basis vectors:

\[
x^* = \sum_{i=1}^{r} \alpha_i x^i + x^C
\]

We can now optimize \(J(x)\) by finding the optimal values for the coefficients \(\alpha_i\).

- dimension \(n\) \(\rightarrow\) dimension \(r\)
- do one full-order evaluation of resulting answer
- approach is efficient if \(r << n\)
- will give the true optimum only if \(x^*\) lies in the span of \(\{x^i\}\)
- basis vectors could be
  - previous designs
  - solutions over a particular range (DoE)
  - derived in some other way (POD)
Example using a reduced-basis approach (van der Plaats Fig 7-2): airfoil design for a unique application.

- many airfoil shapes with known performance are available
- design variables are \((x,y)\) coordinates at chordwise locations \((n \sim 100)\)
- use four basis airfoil shapes (low-speed airfoils) which contain the \(n\) geometry points
- plus two basis shapes which allow trailing edge thickness to vary
- \(r=6\) \((r<<n)\)
- optimize for high speed, maximum lift with a constraint on drag

Vanderplaats, G. N. *Numerical Optimization Techniques for Engineering Design*. Vanderplaats R&D, 1999. Figure 7-2.

Also known as principal components analysis, Kahunen-Loève decomposition, singular value decomposition

\[ x^* = \sum_{i=1}^{r} \alpha_i \varphi_i \]

- The \(r\) basis vectors \(\varphi_i\) are orthogonal
- They are computed from \(M\) empirical solutions \(\{x^1, x^2, \ldots, x^M\}\)
- They are optimal in the sense that they minimize the error between the original and the projected data

\[ \max_{\varphi} \frac{\langle(x, \varphi) \rangle^2}{\|\varphi\|^2} \]

These optimal basis functions can be calculated by:

1. Evaluating the correlation matrix:

\[ R_{ij} = x^iT x^j \]

2. Solving the \(M\timesM\) eigenvalue problem:

\[ R v_i = \lambda_i v_i \]

3. Constructing the basis vectors:

\[ \varphi_j = \sum_{i=1}^{M} v_j^i x^i \]

Use components of the \(j\)th eigenvector to calculate the \(j\)th POD basis vector. The \(j\)th eigenvalue tells us how important is the \(j\)th basis vector.
 Approximation Methods Overview

- **Design variable linking**
  - Reduced-basis methods
    - reduce the number of design variables in the optimization code
    - simcode analysis full order

- **Response surface methods**
  - Kriging
    - same number of design variables
    - simcode analysis simplified

- **Variable-fidelity methods**
  - combine high-fidelity and approximation models

Response Surface Methodology

- Keep the same number of design variables, but simplify the simcode analysis
- Create approximating functions to objective and constraints
- Optimize using the approximations
- Update approximations using current optimal solution guess and repeat
- Response surfaces are smooth even if design space is noisy
- Polynomial-based modeling technique
- Provide compact, explicit functional relationship between response and design variables
- Least squares is computationally inexpensive and easy to implement

Local Approximations

Most common are Taylor series expansions:

\[
J(x) = J(x^0) + \left[ \nabla J(x^0) \right]^T \delta x + \frac{1}{2} \delta x^T H(x^0) \delta x + \cdots
\]

\[
\delta x = x - x^0
\]

- Could use the first two terms ⇒ linear approximation
- Solve, reanalyze and repeat = sequential linear programming
- Could also include quadratic term:

\[
J(x) = J(x^0) + \left[ \nabla J(x^0) \right]^T \delta x + \frac{1}{2} \delta x^T H(x^0) \delta x + \cdots
\]

update requires: 1 function evaluation \( n \) function evaluations \( n(n+1)/2 \) function evaluations

- It is expensive to update the gradient vector and Hessian matrix
- One approach:
  - perform several approximation cycles updating only the constant term
  - then update the linear term and repeat
  - finally, update the Hessian only when no other progress can be made
- If the design space is highly nonlinear, there is no guarantee that this approach will work
Response Surface Methodology

- Another approach: use what information is available to create the approximation
- Use this approximation to make a small move in the design variables
- Analyze the result precisely ⇒ new function evaluation
- Use the new function evaluation to improve the approximation to the design space

⇒ **Fit a response surface**
- Can use a quadratic or higher order surface
- Might choose to use only some of the function evaluations (e.g. those in a local neighborhood)

**RSM**

Assume we have evaluated the baseline plus \( q \) designs:

\[ \mathbf{x}^0, \mathbf{x}^1, \ldots, \mathbf{x}^q \rightarrow J^0, J^1, \ldots J^q \]

We could write \( q \) equations of the form \((*)\) using

\[ (\mathbf{x}^1 - \mathbf{x}^0, J^1 - J^0), (\mathbf{x}^2 - \mathbf{x}^0, J^2 - J^0), \ldots, (\mathbf{x}^q - \mathbf{x}^0, J^q - J^0) \]

There is a total of \( N = n + n(n+1)/2 \) unknowns:

\[ \frac{\partial J}{\partial x_1}, \frac{\partial J}{\partial x_2}, \ldots, \frac{\partial J}{\partial x_n}, H_{11}, H_{12}, \ldots, H_{nn} \]

**RSM**

\[ e.g. \text{ define } \Delta J = J(\mathbf{x}) - J(\mathbf{x}^0) \]

quadratic approximation:

\[ \Delta J = \nabla J^T \delta \mathbf{x} + \delta \mathbf{x}^T \mathbf{H} \delta \mathbf{x} \]

\[ = \frac{\partial J}{\partial x_1} \delta x_1 + \frac{\partial J}{\partial x_2} \delta x_2 + \cdots + \frac{\partial J}{\partial x_n} \delta x_n \]

\[ + \frac{1}{2} \left( H_{11} \delta x_1^2 + H_{22} \delta x_2^2 + \cdots + H_{nn} \delta x_n^2 \right) \]

\[ + H_{12} \delta x_1 \delta x_2 + H_{13} \delta x_1 \delta x_3 + \cdots + H_{1n} \delta x_1 \delta x_n \]

\[ + H_{23} \delta x_2 \delta x_3 + \cdots + H_{n-1,n} \delta x_{n-1} \delta x_n \]

(all derivatives and entries of \( \mathbf{H} \) are evaluated at \( \mathbf{x}^0 \))

**RSM**

- \( q \) equations, \( N \) unknowns
- If \( q < N \), only some coefficients can be calculated
- If \( q > N \), use weighted least squares
- Weight designs closer to current \( \mathbf{x}^0 \) more heavily
- In general, use \( q \geq n + 1 \) initial designs so an initial linear approximation can be provided
- If have baseline plus \( n \) designs, can fit a linear approximation in each direction (i.e. a hyperplane)
- If have more solutions, can fit a quadratic or higher order surface
• In other words, fit objective function with a polynomial
  • e.g. quadratic approximation:
    \[
    J(x) = a_0 + \sum_i b_i x_i + \sum_{i,j<i} c_{ij} x_i x_j
    \]
  • Update model by including a new function evaluation then doing least squares fit to compute the new coefficients

Estimation problem:

\[
J \approx Xc
\]

\[
J = \begin{bmatrix} J^1 & J^2 & \cdots & J^M \end{bmatrix}^T
\]

\[
c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_p \end{bmatrix}
\]

\[
x = \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^M \end{bmatrix}
\]

least squares solution:

\[
c = (X^T X)^{-1} X^T J
\]

RSM tends to create a global model of the design space (especially if all points weighted equally in the LS)

• May be of limited accuracy when multiple extrema exist (especially quadratic polynomial models)

RSM tends to create a global model of the design space

• May be of limited accuracy when multiple extrema exist

Kriging:

Unknown function to be modeled expressed as:

\[
J(x) = f(x) + Z(x)
\]

• known function
• Gaussian random function
  • zero mean
  • variance \( \sigma^2 \)
• “global” model for design space
• “localized” deviation from global model

\[
\text{cov}[Z(x^i), Z(x^j)] = \sigma^2 R[x^i, x^j]
\]

\[\text{covariance matrix of } Z(x) \]
\[\text{correlation matrix} \]
\[\text{correlation function}\]
**Approximation Methods Overview**

- **Design variable linking**
  - Reduced-basis methods
    - reduce the number of design variables in the optimization code
    - simcode analysis full order

- **Response surface methods**
  - Kriging
    - same number of design variables
    - simcode analysis simplified

- **Variable-fidelity methods**
  - combine high-fidelity and approximation models

---

**Variable Fidelity Models**

**Initialization**

- High-fidelity model
  - recourse to detailed model

- Approximation model
  - optimization on a simplified model

From: Fig. 1, Alexandrov et al.

- Use information from high-fidelity model to check approximate designs
- Optimize using approximate model
- Recalibrate using high-fidelity model

---

**Questions:**

- What do we do when the design derived from the low-fidelity optimization does not provide an improvement in the true objective?
- How can we use information about the predictive capability of the approximation model to decide when to go back to the high-fidelity model?
- How do we decide when to update the approximate model?

---

When optimization with approximation model is unsuccessful, there are two possible approaches:

1. Improve the fidelity of the approximate model
2. Do "less" optimization

One option: use a **trust region** approach
Trust Region Approach

Classic approach:
- Regulate the length of the steps taken by the iterative optimization algorithm
- Regulate based on quality of current approximation model

\[ J(x^k + s) \approx q^k(x^k + s) = J(x^k) + \left[ \nabla J(x^k) \right]^T s + \frac{1}{2} s^T B^k s \]

where \( s \) is the prospective step in the design variables, \( \Delta x \), and \( B^k \) is a model of the Hessian matrix at \( x^k \).

Trust Region Update

- Trust radius, \( \delta^k \), is updated adaptively
- Update depends on predictive quality of quadratic model:
  - if model did a good job of predicting \( J \), or if there was more improvement than predicted, then increase \( \delta^k \)
  - if model did a bad job of predicting \( J \) (\( J \) increased or decrease was much lower than predicted), then decrease \( \delta^k \)
  - if model did an acceptable job of predicting \( J \), then do not change \( \delta^k \)
Classic Trust Region Algorithm

For $k=0,1,...$ until convergence do {

Find an approximate solution $s_k$ to the subproblem:

$$\begin{aligned}
\min_{s} & \quad q^k(x^k + s) \\
\text{s.t.} & \quad \|s\| \leq \delta^k
\end{aligned}$$

Compare the actual and predicted decrease:

$$r = \frac{J(x^k) - J(x^k + s)}{J(x^k) - q(x^k + s)}$$

Update $x^k$ and $\delta^k$

From Fig. 2, Alexandrov et al.

β-Correlation Method

- Basic idea is to take low-fidelity approximate model, $J_a$, and correct it by scaling
- Define the scale factor:

$$\beta = \frac{J(x)}{J_a(x)}$$

Given the current design $x^k$, build a first order model of $\beta$ about $x^k$:

$$\beta^k(x) = \beta(x^k) + \nabla \beta(x^k)(x - x^k)$$

Optimize with approximate model and use local model of $\beta$ to scale result:

$$J(x) \approx \beta^k J_a(x)$$

Variable Complexity Model

- iSight uses a Variable Complexity Model (VCM)
- Use two simcodes for the same physical process:
  1. more accurate, longer running (exact) $J(x)$
  2. less accurate, shorter running (approx.) $J_a(x)$
- Compute a multiplicative or additive correction factor, $\sigma$:

$$\begin{aligned}
\sigma &= \frac{J(x^0)}{J_a(x^0)} \\
J(x) &= \sigma J_a(x)
\end{aligned}$$

or

$$\begin{aligned}
\sigma &= J(x^0) - J_a(x^0) \\
J(x) &= \sigma + J_a(x)
\end{aligned}$$

VCM

Optimize with approximate model, update with simcode:

1. Run both models, calculate $\sigma^0$
2. Optimize using approximate model and $\sigma^0$
3. Update correction factor using simcode and repeat
Lecture Summary

- Approximation methods are one way to capture high-fidelity responses without the computational cost
- Design variable linking if physical problem allows
- Reduced-basis methods use low-order representation of the design vector
  - use previous designs, DoE or POD
- Response surface methodology
  - use polynomial models
  - weighted least squares
- Kriging
  - interpolation models
  - global + local behavior
- Variable-fidelity models
  - Trust region approach
  - $\beta$-correlation

References


