

# Adjoint-Based Approaches to Uncertainty Quantification

discussing the thesis of Qiqi Wang

Roy H. Stogner

The University of Texas at Austin

May 21, 2009

# Outline

- 1 Outline
- 2 Definitions and Notation
- 3 Sensitivity-based Uncertainty Quantification
  - Quantification of Margins and Risk
  - Gradient-based Interpolation
- 4 Adjoint-based Sensitivities

# Definitions and Notation

## Variables

$\vec{\xi} \in \mathbb{R}^n$ : parameter set

$\vec{u}(\vec{x}, t; \vec{\xi}) \in H(\Omega; [t_0, t_f])$ : parameter-dependent PDE solution

$J(\vec{v})$ : Quantity of interest functional

## Primal/Forward Problem

$$\mathcal{R}(\vec{u}(\vec{\xi}), \vec{v}; \vec{\xi}) \equiv 0 \quad \forall \vec{v}$$

## Uncertainty Quantification Problems

$$P_C \equiv P(J(\vec{u}(\vec{\xi})) > J_C)$$

$$\tilde{J}(\vec{\xi}) \approx J(\vec{u}(\vec{\xi}))$$

# Quantification of Margins and Risk

## Forward UQ Problem

$$P_C \equiv P(J(\vec{u}(\vec{\xi})) > J_C)$$

I.e., what is the (small!) probability of failure?

## Challenges

- Basic Monte Carlo:

$$P_C \approx P_N \equiv \frac{1}{N} \sum_{i=1}^N I(J(\vec{\xi}_i) > J_C)$$

- Relative error:

$$\hat{e}_{MC} = \frac{\sqrt{\text{Var}[P_N]}}{P_C} = \sqrt{\frac{1 - P_C}{NP_C}}$$

- Monte Carlo costs are  $N = \mathcal{O}\left(\frac{1}{P_C}\right)$

# Surrogate Accelerated Uncertainty Quantification

## Surrogate Response Functions

$$J_S(\vec{\xi}) \approx J(\vec{\xi})$$

Linear example:

$$J_L(\vec{\xi}) \equiv J(\vec{\xi}_0) + \nabla J(\vec{\xi}) \cdot (\vec{\xi} - \vec{\xi}_0)$$

## Advantages

- $J_L$  can be obtained from one forward and one adjoint solve
- $J_S(\vec{\xi}_i)$  can be evaluated much more often than  $J(\vec{\xi}_i)$  for the same cost
- UQ problems applied to the surrogate function are cheap to solve
- UQ problems applied to the real quantity of interest can be approximated by or accelerated using the surrogate function

# Control-Variate Accelerated Monte Carlo

## Control Variate Risk Estimation

$$\begin{aligned}
 P_C &= P(J_S > J_C) + P(J > J_C) - P(J_S > J_C) \\
 &\approx P_N^{CV} \equiv P(J_S > J_C) + \frac{1}{N} \sum_{i=1}^N \left( I(J(\vec{\xi}_i) > J_C) - I(J_S(\vec{\xi}_i) - J_C) \right)
 \end{aligned}$$

Still an unbiased estimator, but with a lower variance:

$$\begin{aligned}
 \text{Var} [P_N^{CV}] &= \frac{1}{N} \text{Var} [I(J > J_C) - I(J_S - J_C)] \\
 &= \frac{1}{N} \left( P(J < J_C < J_S \text{ or } J > J_C > J_S) - \right. \\
 &\quad \left. (P(J_S > J_C) - P(J > J_C))^2 \right)
 \end{aligned}$$

# Surrogate Functions

## Multi-sample Surrogate Functions

Lots of prior work:

- Polynomial Chaos, other Bases
- Collocation
- Least-Squares Fit
- Quadrature-based Projections

# Gradient-based Interpolation

## Qiqi Wang's Method

- From the  $J$  samples at parameters  $\vec{\xi}_i$ , estimate the “magnitude”  $\beta$  and “roughness”  $\gamma$  satisfying  $\beta\gamma^k = \sqrt{\text{Var} [J^{(k)}(\vec{\xi})]}$
- Treat  $J^{(k)}$  as a random variable, mean 0, stddev  $\beta\gamma^k$
- Write surrogate function at parameter  $z$  as

$$\tilde{J}(z) \equiv \sum_{i=1}^n a_i J(\vec{\xi}_i) + \sum_{j=1}^m \vec{b}_j \cdot \nabla J(\vec{\xi}_j)$$

- Calculate  $a_i, \vec{b}_i$  to minimize expected interpolation residual (minimize  $[ab]A[ab]^T$ )

# Other Sensitivity Derivative Calculations

## Finding Sensitivity Derivatives

$$\nabla J \equiv \frac{\partial J}{\partial \vec{\xi}}$$

- Finite Difference Methods:

$$\nabla_k J \approx \frac{J(\vec{\xi}_0 + h e_k) - J(\vec{\xi}_0)}{h}$$

- “Full Sensitivity” (“Tangent Linear Method”?):

$$\frac{\partial \mathcal{R}}{\partial \vec{u}} \frac{\partial \vec{u}}{\partial \xi_k} = - \frac{\partial \mathcal{R}}{\partial \xi_k}$$

$$\frac{dJ}{d\xi_k} = \frac{\partial J}{\partial \xi_k} + \frac{\partial J}{\partial \vec{u}} \frac{\partial \vec{u}}{\partial \xi_k}$$

# Other Parameter Sensitivity Methods

## Using Sensitivity Derivatives

- Perturbation Method (Taylor series)

$$J_S(\vec{\xi}) \equiv J(\vec{\xi}_0) + (\vec{\xi} - \vec{\xi}_0) \cdot \nabla J(\vec{\xi}) + \dots$$

- Hybrid Dimensionality Reduction (ignore insensitive parameters)

# Adjoint-based Sensitivity Calculation

## Primal Problem

$$\begin{aligned} \mathcal{R}(\vec{u}(\vec{\xi}), \vec{v}; \vec{\xi}) &\equiv 0 \quad \forall \vec{v} \\ \frac{d\mathcal{R}}{d\vec{\xi}} &= 0 \\ \frac{\partial \mathcal{R}}{\partial \vec{\xi}} + \frac{\partial \mathcal{R}}{\partial \vec{u}} \frac{\partial \vec{u}}{\partial \vec{\xi}} &= 0 \end{aligned}$$

## Adjoint Problem

$$\begin{aligned} \nabla J &\equiv \frac{dJ(\vec{u}; \vec{\xi})}{d\vec{\xi}} \\ \frac{\partial \mathcal{R}(\vec{u}, \phi(\vec{u}, \vec{\xi}); \vec{\xi})}{\partial \vec{u}} &\equiv \frac{\partial J(\vec{u}; \vec{\xi})}{\partial \vec{u}} \end{aligned}$$

## Costs

- Forward solution expensive: May be highly nonlinear
- Steady adjoint solutions efficient: Just one linear solve
- Forward implementation simple: requires residual
- Adjoint implementation complicated: requires full Jacobian

# Adjoint-based Sensitivity Calculation

## Adjoint-based Sensitivity

$$\begin{aligned}
 \nabla_k J &= \frac{\partial J}{\partial \xi_k} + \frac{\partial J}{\partial \vec{u}} \frac{\partial \vec{u}}{\partial \xi_k} \\
 &= \frac{\partial J}{\partial \xi_k} + \frac{\partial \mathcal{R}(\vec{u}, \phi(\vec{u}, \vec{\xi}))}{\partial \vec{u}} \frac{\partial \vec{u}}{\partial \xi_k} \\
 &= \frac{\partial J}{\partial \xi_k} - \frac{\partial \mathcal{R}(\vec{u}, \phi(\vec{u}, \vec{\xi}))}{\partial \xi_k}
 \end{aligned}$$

## Costs

- One adjoint solve per quantity of interest
- Only one inner product per uncertain parameter

# Adjoint of Transient Problems

## Continuous Problem

$$\left(\frac{\partial \vec{u}}{\partial t}\right)^* = \left(-\frac{\partial \vec{u}}{\partial t}\right)$$

## Discrete Problem

- Each timestep's residual depends on  $s$  preceding solutions
- Forward Problem Jacobian: block banded lower triangular
- Adjoint Problem Matrix: block banded upper triangular

## Costs

- Unsteady problem adjoint is solved “backwards” in time!
- Recreating each timestep's Jacobian requires that timestep's solution

# Checkpointing

CPU time is too limited to recalculate forward timesteps  $\mathcal{O}\left(\frac{1}{\Delta t^2}\right)$  times.

## Saving Intermediate Solutions

- I/O is too limited to save/load  $\mathcal{O}\left(\frac{1}{\Delta t}\right)$  timesteps to disk
- RAM is far too limited to keep  $\mathcal{O}\left(\frac{1}{\Delta t}\right)$  timesteps in memory

## Saving Checkpoint Solutions

- Store up to  $\delta$  checkpoints, allow up to  $\tau$  recalculations of each timestep
- Sufficient to take  $\left(\frac{\delta+\tau}{\tau}\right)$  timesteps
- Logarithmic complexity growth
- Dynamic checkpointing avoids preselecting  $\delta, \tau$
- See Figure 3.2

# Monte Carlo Adjoint Solver

## Monte Carlo Linear Solver

- Monte Carlo iterates begin at timestep 0
- Iterates step forward in time along with primal problem
- And Then A Miracle Occurs