Error Estimation and Mesh Adaptation using Output Adjoints

Krzysztof J. Fidkowski, University of Michigan

38th Advanced CFD Lecture Series
von Karman Institute, Rhode-St-Genèse, Belgium

September 14-17, 2015
Outline

1. Introduction
2. Discretization
3. The Adjoint
4. Output Error Estimation
5. Adaptation
6. Mesh Optimization
7. References
Introduction

Complex CFD simulations are made possible by
- Increasing computational power
- Improvements in numerical algorithms

New liability: ensuring accuracy of computations
- Management by expert practitioners is not feasible for increasingly-complex flow fields
- Reliance on best-practice guidelines is an open-loop solution: numerical error is unchecked for novel configurations
- Output calculations are not yet sufficiently robust, even on relatively standard simulations
Errors in simulations come from various sources

- **Experimental data**
  - Observation errors
    - Noise, calibration...

- **Reality**
  - Modeling errors

- **Validation**
  - Governing equations
    - Discretization errors
  - System of equations
    - Convergence errors
  - CFD solution/data

- **Calibration**
  - Comparison of CFD and experiment
    - Compounding of errors

- **Verification**
Improving CFD robustness

**Error estimation**
- Error estimates on outputs of interest are necessary for confidence in CFD results
- Mathematical theory exists for obtaining such estimates
- Recent works demonstrate the success of this theory for aerospace applications

**Mesh adaptation**
- Error estimation alone is not enough
- Engineering accuracy for complex aerospace simulations demands mesh adaptation to control numerical error
- Automated adaptation improves robustness by closing the loop in CFD analysis
A typical output-adaptive result

Initial mesh

Adapted mesh

adaptive iterations

±\epsilon (error est.)

output
cost (degrees of freedom)
raw output
corrected output
exact output value
Why not just adapt “obvious” regions?

Fishtail shock in $M_\infty = 0.95$ inviscid flow over a NACA 0012 airfoil

Error Estimation and Mesh Adaptation using Output Adjoints
Mesh adaptation

- Adaptation can isolate singularities with small elements
- In many high-order methods, local $p$-enrichment is possible
- Combination of both can yield a powerful method for efficiently improving accuracy

![Graph showing log(error) vs. log(dof) for different mesh adaptation methods: uniform h, adaptive h, adaptive hp, one adapt iter, two adapt iters.](image)
Outline

1. Introduction
2. Discretization
3. The Adjoint
4. Output Error Estimation
5. Adaptation
6. Mesh Optimization
7. References
Conservation equations

\[ \mathbf{r}(\mathbf{u}) \equiv \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{H}(\mathbf{u}, \nabla \mathbf{u}) = 0 \]

- \( \mathbf{u} \in \mathbb{R}^s \) is the state vector
- \( \mathbf{H} \in [\mathbb{R}^s]^d \) is the total flux with spatial components

\[ H_i = F_i(\mathbf{u}) + G_i(\mathbf{u}, \nabla \mathbf{u}) \]

- Inviscid flux
- Viscous flux

- \( 1 \leq i \leq d = \text{spatial dimension} \)
- The viscous flux is linear in the state gradient

\[ G_i(\mathbf{u}, \nabla \mathbf{u}) = -K_{ij}(\mathbf{u}) \partial_j \mathbf{u} \]
Polynomials of order $p_e$ on each element:

$$u_h(\bar{x}) \approx \sum_{e=1}^{N_e} \sum_{n=1}^{N_{pe}} U_{en} \phi_{en}(\bar{x})$$

- $N_e$ = # of elements
- $N_{pe}$ = # of basis functions on element $e$
- $\phi_{en}(\bar{x})$ = $n^{th}$ basis function of order $p_e$ on $e$
- $p_e$ = approximation order on element $e$
- $U_{en}$ = vector of $s$ coefficients on $n^{th}$ basis function on element $e$
Discontinuous basis functions

Continuous Galerkin (CG)

Discontinuous Galerkin (DG)

- DG approximation space: no inter-element continuity,

\[ u \in \mathcal{V}_h = [\mathcal{V}_h]^s, \quad \mathcal{V}_h = \{ u \in L^2(\Omega) : u|_{\Omega_e} \in \mathcal{P}^p(\Omega_e) \ \forall \ \Omega_e \in T_h \} \]

- Equations: multiply by test functions and integrate by parts

\[ R_h(u_h, v_h) \equiv \int_{\Omega} v_h^T r(u_h) d\Omega = 0, \ \forall v_h \in \mathcal{V}_h \]

- Elements coupled together through upwind flux functions
Discrete system

- Discrete residual on element $e$ for $n^{th}$ test function,

$$R_{en} \equiv \{ \mathcal{R}_h(u_h, \phi_{en} e_r) \}_{r=1...s} \in \mathbb{R}^s$$

- We lump all residuals and states into single vectors (size $N$),

$$R(U) = 0$$

$$U = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ U_{e1} \\ U_{e2} \\ \vdots \\ U_{en} \\ \vdots \\ U_{eN_{pe}} \end{bmatrix} \quad \text{element } e$$

$$\begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \quad \text{basis fcn } n$$

$$\begin{bmatrix} U_1 \\ U_2 \\ \vdots \\ U_s \end{bmatrix} \quad \text{state approx. coefficients for element } e \text{ and basis function } n$$

$$\text{numbers needed to describe } s \text{ order } p \text{ polynomials inside element } e$$
Outline

1 Introduction
2 Discretization
3 The Adjoint
4 Output Error Estimation
5 Adaptation
6 Mesh Optimization
7 References
Local sensitivities

- Suppose $N_\mu$ parameters affect our PDE, but we only have one scalar output, $J(U)$:

\[
\begin{align*}
\mu \in \mathbb{R}^{N_\mu} &\rightarrow \mathbf{R}(\mathbf{U}, \mu) = 0 \\
N \text{ equations} \rightarrow \mathbf{U} \in \mathbb{R}^N &\rightarrow J(U) \\
\text{state} \rightarrow \text{output (scalar)}
\end{align*}
\]

- We are interested in how $J$ changes with $\mu$,

\[
\frac{dJ}{d\mu} \in \mathbb{R}^{1 \times N_\mu} = N_\mu \text{ sensitivities}
\]

- Brute force approach: perturb each entry in $\mu$ individually, re-solve the PDE, and measure the perturbation in the output

This is inefficient for large $N_\mu$
The discrete adjoint

- We can efficiently compute sensitivities using a discrete adjoint vector, $\Psi \in \mathbb{R}^N$,

$$\frac{dJ}{d\mu} = \Psi^T \frac{\partial R}{\partial \mu}$$

- Each entry in $\Psi$ is the sensitivity of $J$ to residual source perturbations in the corresponding entry in $R$

$$\delta J = \Psi^T \delta R$$
The discrete adjoint equation

- Consider a small perturbation $\delta R$ to the residual.
- The resulting (linearized) state perturbation, $\delta U$ satisfies
  \[
  \frac{\partial R}{\partial U} \delta U + \delta R = 0
  \]

- Also linearizing the output we have,
  \[
  \delta J = \frac{\partial J}{\partial U} \delta U = \Psi^T \delta R = -\Psi^T \frac{\partial R}{\partial U} \delta U
  \]

- Requiring the above to hold for arbitrary perturbations yields the linear discrete adjoint equation
  \[
  \left( \frac{\partial R}{\partial U} \right)^T \Psi + \left( \frac{\partial J}{\partial U} \right)^T = 0
  \]
Consider flow over an airfoil:

Output
\( J = \text{Lift} \)

State: \( U_e = [\rho, \rho \vec{v}, \rho E]_e \)

Inputs
\( \mu = \begin{bmatrix} \alpha \\ M_\infty \\ Re \end{bmatrix} \)

Residual: \( R_e = \int_{\partial \Omega_e} \vec{F} \cdot \vec{n} \)
Output sensitivity to residuals: the adjoint

The lift adjoint $\Psi$ is the sensitivity of lift to residual sources.

We have a solution $U$ when $R = 0$

$\Psi_e$ element $e$

Lift $= J(U)$

state $U$
Output sensitivity to residuals: the adjoint

The lift adjoint $\Psi$ is the sensitivity of lift to residual sources.

We have a solution $U$ when $R = 0$

What if we add a residual source, $\delta R_e$?

Lift $= J(U)$
Output sensitivity to residuals: the adjoint

The lift adjoint $\Psi$ is the sensitivity of lift to residual sources.

We have a solution $U$ when $R = 0$

What if we add a residual source, $\delta R$?

$Lift = J(U) + \delta J$

$U + \delta U$

resolving for the state ...

$\Psi_e \delta R = \Psi^T_e \delta R$
The lift adjoint $\Psi$ is the sensitivity of lift to residual sources.

We have a solution $U$ when $\mathbf{R} = 0$

What if we add a residual source, $\delta \mathbf{R}_e$?

What if we add a residual source, $\delta \mathbf{R}_e$?

resolving for the state ...
Sample steady adjoint solution

M = 1.5 flow

output = pressure integral

diamond airfoil

green = zero adjoint

(showing y−mom component)
Another steady adjoint solution

RAE 2822, $M_\infty = 0.5$, $Re = 10^5$, $\alpha = 1^\circ$

- $x$-momentum primal state
- cons. of $x$-mom drag adjoint

- Adjoint shares similar qualitative features with primal
- Note wake “reversal” in adjoint solution
- The discrete adjoint solution approximates the continuous adjoint when the discretization is adjoint consistent
The discrete adjoint, $\Psi$, is obtained by solving a linear system. This system involves linearizations about the primal solution, $U$, which is generally obtained first. When the full Jacobian matrix, $\frac{\partial R}{\partial U}$, and an associated linear solver are available, the transpose linear solve is straightforward. When the Jacobian matrix is not stored, the discrete adjoint solve is more involved: all operations in the primal solve must be linearized, transposed, and applied in reverse order. In unsteady discretizations, the adjoint must be marched backward in time from the final to the initial state.
Outline

1. Introduction
2. Discretization
3. The Adjoint
4. Output Error Estimation
5. Adaptation
6. Mesh Optimization
7. References
Output error estimation

We want: \( \delta J = J_H(U_H) - J(U) \)

This is the difference between \( J \) computed with the discrete system solution, \( U_H \), and \( J \) computed with the exact solution, \( U \)

We’ll settle for: \( \delta J = J_H(U_H) - J_h(U_h) \)

This is the difference in \( J \) relative to a finer discretization (\( h \))

coarse space: \( \rightarrow \mathbf{R}_H(U_H) = 0 \rightarrow \underbrace{\mathbf{U}_H}_{\text{\( N_H \) equations}} \rightarrow \underbrace{J_H(U_H)}_{\text{output (scalar)}} \)

fine space: \( \rightarrow \mathbf{R}_h(U_h) = 0 \rightarrow \underbrace{\mathbf{U}_h}_{\text{\( N_h \) equations}} \rightarrow \underbrace{J_h(U_h)}_{\text{output (scalar)}} \)
Fine-space injection

- The fine space can arise from $h$ or $p$ refinement
- Define an injection of the coarse state into the fine space

$U_H^H$ will generally not satisfy the fine-space equations,

$$R_h(U_h^H) \neq 0$$
A finer space (e.g. order enrichment) can uncover residuals in a converged solution.

Example: NACA 0012 at $\alpha = 2^\circ$ in $Re = 5000, M_\infty = 0.5$ flow

Coarse space state, $U_H$

Coarse space residual, $R_H(U_H)$

$p_H = 1$

Zero as expected
Fine-space residuals

- A finer space (e.g. order enrichment) can uncover residuals in a converged solution
- Example: NACA 0012 at $\alpha = 2^\circ$ in $Re = 5000$, $M_\infty = 0.5$ flow

Injected state, $U^H_h$

Fine space residual, $R_h(U^H_h)$

$p_h = 2$

Nonzero: new info
The adjoint-weighted residual

- $U^H_h$ solves a *perturbed* fine-space problem
  
  find $U'_h$ such that: $\underbrace{R_h(U'_h) - R_h(U^H_h)}_{\delta R_h} = 0$ \implies \text{answer: } U'_h = U^H_h

- The fine-space adjoint, $\Psi_h$, then tells us to expect an output perturbation of
  
  $J_h(U^H_h) - J_h(U_h) = \underbrace{\Psi^T_h \delta R_h = -\Psi^T_h R_h(U^H_h)}_{\approx \delta J}


- This equation assumes small perturbations (e.g. if nonlinear)
- In summary, we have an *adjoint-weighted residual*:

$$\delta J \approx -\Psi^T_h R_h(U^H_h)$$
Adjoint-weighted residual example

Fine space residual, $\mathbf{R}_h(U^H_h)$

Error indicator, $\epsilon_e = \left| \Psi^T_{h,e} \mathbf{R}_{h,e}(U^H_h) \right|$

Output error: $\delta J \approx -\Psi^T_{h} \mathbf{R}_h(U^H_h)$

Idea: adapt where $\epsilon_e$ is high, to reduce the residual there
Two more definitions

**Corrected output**

\[ J_H^{\text{corrected}} = J_H - \delta J \]

- Should converge faster than \( J_H \)
- Remaining error = error left in corrected output

**Error effectivity**

\[ \eta_H = \frac{J_H(U_H) - J_h(U_h)}{J_H(U_H) - J} \]

- \( J = \) exact output
- We want \( \eta_H \) close to 1
- Effectivity is affected by choice of fine space
Drag error in viscous flow over an airfoil

Mach contours

Error Estimation and Mesh Adaptation using Output Adjoint

Drag coefficient error

- error in output
- error in corrected output

Error effectivity

- relative to exact error
- relative to fine-space error

Ideal = 1
Approximations

How do we calculate $\Psi_h = \text{the adjoint on the fine space}$?

Options:

1. Solve for $U_h$ and then $\Psi_h$ – expensive! Potentially still useful to drive adaptation. [14: Solín and Demkowicz, 2004]

2. Solve for the coarse space adjoint, $\Psi_H$, and:
   
   1. Reconstruct $\Psi_H$ on the fine space using a higher-accuracy stencil. Smoothness assumption on adjoint.
   
   2. Initialize $\Psi_h$ with $\Psi_H$ and take a few iterative solution (smoothing) steps on the fine space.
      [2: Barter and Darmofal, 2008] [11: Oliver and Darmofal, 2008]
Corrections and remainders

- Define $\Psi_h^H \equiv I_h^H \Psi_H$ = injection of $\Psi_H$ into $h$.
- Define the adjoint perturbation, $\delta \Psi_h \equiv \Psi_h^H - \Psi_h$
- Rewrite the adjoint-weighted residual as:

$$\delta J = - (\Psi_h^H)^T R_h (U_h^H) + (\delta \Psi_h)^T R_h (U_h^H) + O(\delta U_h, \delta \Psi_h)^2$$

- The computable correction is tempting to use directly, but:
  - It does not incorporate fine-space information $\Rightarrow$ it performs poorly as an adaptive indicator
  - It is zero for FEM with Galerkin orthogonality
- For nonlinear problems the “error in the estimate” can be reduced to third-order via [13: Rannacher, 2001]:

$$\delta J \approx - (\Psi_h^H)^T R_h (U_h^H) + \frac{1}{2} (\delta \Psi_h)^T R_h (U_h^H) + \frac{1}{2} (\delta U_h)^T R_h^\nu (\Psi_h^H)$$
Error estimation summary

1. Solve the coarse-discretization forward and adjoint problems: $U_H$ and $\Psi_H$

2. Pick a fine discretization “$h$” (mesh refinement or order enrichment)

3. Calculate or approximate $\Psi_h = \text{adjoint on the fine space}$

4. Project $U_H$ onto the fine discretization and calculate the residual $R_h(U^H_h)$

5. Weight the fine-space residual with the fine-space adjoint to obtain the output error error estimate

6. The computed output error $\delta J$ is an estimate of the true error, not a bound
Mesh adaptation

Initial coarse mesh & error tolerance

Flow and adjoint solution

Output error estimate

Error localization

Mesh adaptation

Tolerance met?

Done
Error localization

- Recall that the adjoint-weighted residual expression for the output error involves a sum over elements ($e$)

$$J_H(U_H) - J_h(U_h) \approx -\Psi^T_h R_h(U_H^H) = - \sum_e \Psi^T_{he} R_{he}(U_H^H)$$

- The absolute-value of each element’s contribution to the error is the **error indicator** on that element

$$\epsilon_e \equiv |\Psi^T_{he} R_{he}(U_H^H)|$$

*Right*: plot of error indicator for a viscous DG simulation, $p_H = 1$, $p_h = 2$
# Output-based mesh adaptation

## Motivating ideas
- The error indicator \( (e_e) \) identifies elements with large adjoint-weighted residuals
- Locally refining a mesh reduces local residuals
- So we can reduce the output error by refining those elements that have a high \( e_e \)

## Adaptation choices
- Local refinement versus global re-meshing
- Which/how many elements should be targeted?
- Isotropic versus anisotropic refinement
- \( h, p, \) or \( hp \) mechanics
- Should coarsening be allowed?
Local mesh modification

- Modify the mesh incrementally (mesh generation is hard)
- Often more robust than global re-meshing
- With node movement, can be flexible for unstructured meshes
- Hanging nodes easily supported in DG

Unstructured local mesh operators

Hanging-node refinement
Example: inviscid flow over an airfoil

NACA 0012, Euler, $M_\infty = 0.5$, $\alpha = 2^\circ$

- Mach number contours
- Drag adjoint ($x$–momentum)

- Output $J = \text{drag}$ (expect $\sim 0$)
- Compare hanging-node adaptation to uniform refinement
- Look at approximation orders $p = 1$ and $p = 2$
Inviscid flow: hanging-node adaptation

- Isotropic hanging-node refinement
- Fine space = $p + 1$
- Fixed fraction $f^{\text{adapt}} = 5\%$
- 20 adaptive iterations
- No coarsening
- Use adjoint-weighted residual $\delta J$ as correction
- “Exact” output from a $p = 3$ fine-mesh solve
- *Right*: $p = 2$ first adaptation
Inviscid flow: $p = 2$ mesh sequence

![Graph showing drag coefficient error vs number of elements](image)

Number of elements

Drag coefficient error

Actual error

Error Estimation and Mesh Adaptation using Output Adjoints

Adaptation
Inviscid flow: $p = 2$ mesh sequence

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: $p = 2$ mesh sequence
Inviscid flow: $p = 2$ mesh sequence
Inviscid flow: $p = 2$ mesh sequence

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: $p = 2$ mesh sequence

Error Estimation and Mesh Adaptation using Output Adjoints

Adaptation
Inviscid flow: $p = 2$ mesh sequence
Inviscid flow: $p = 2$ mesh sequence
Inviscid flow: $p = 2$ mesh sequence

![Graph showing drag coefficient error vs number of elements]

- **Actual error**
- **Estimated error**

Error Estimation and Mesh Adaptation using Output Adjoints

Adaptation
Inviscid flow: $p = 2$ mesh sequence

Error Estimation and Mesh Adaptation using Output Adjoint Adaptation
Inviscid flow: $p = 2$ mesh sequence

![Graph showing drag coefficient error vs number of elements]

- Actual error
- Estimated error

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: $p = 2$ mesh sequence

![Graph showing drag coefficient error vs. number of elements]

- **Actual error**
- **Estimated error**

![Mesh adaptation diagram]
Inviscid flow: $p = 2$ mesh sequence

![Graph showing the drag coefficient error vs. number of elements. The graph compares actual error and estimated error, with both error estimates decreasing as the number of elements increases.](image-url)

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: $p = 2$ mesh sequence
Inviscid flow: $p = 2$ mesh sequence

<table>
<thead>
<tr>
<th>Number of elements</th>
<th>Drag coefficient error Actual error</th>
<th>Estimated error</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^2$</td>
<td>$10^{-6}$</td>
<td></td>
</tr>
<tr>
<td>$10^3$</td>
<td>$10^{-5}$</td>
<td></td>
</tr>
<tr>
<td>$10^4$</td>
<td>$10^{-4}$</td>
<td></td>
</tr>
<tr>
<td>$10^5$</td>
<td>$10^{-3}$</td>
<td></td>
</tr>
<tr>
<td>$10^6$</td>
<td>$10^{-2}$</td>
<td></td>
</tr>
</tbody>
</table>

![Graph showing drag coefficient error vs. number of elements](image)

- Blue circles represent the actual error.
- Red squares represent the estimated error.

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: $p = 2$ mesh sequence

![Graph showing drag coefficient error vs. number of elements]

- **Actual error**
- **Estimated error**

![Flow field diagram]
Inviscid flow: $p = 2$ mesh sequence

![Graph showing drag coefficient error vs. number of elements]

- Actual error
- Estimated error
- Uniform refinement

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: $p = 2$ mesh sequence

![Graph showing drag coefficient error vs. number of elements with markers for uniform and adaptive refinement.](image)

Uniform ref 1
Adapt ref 10

Number of elements

Drag coefficient error

- Actual error
- Estimated error
- Uniform refinement

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: adapted/uniform comparison

Farfield region

Uniform refinement 1

Adapt refinement 10

672 elements

659 elements
Inviscid flow: adapted/uniform comparison

Near-field region

Uniform refinement 1

Adapt refinement 10

672 elements

659 elements
Inviscid flow: adapted/uniform comparison

Leading edge

Uniform refinement 1

Adapt refinement 10

672 elements

659 elements
Inviscid flow: adapted/uniform comparison

Trailing edge

**Uniform refinement 1**
- 672 elements

**Adapt refinement 10**
- 659 elements
Inviscid flow: drag convergence (exact $\psi_h$)

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: drag convergence (exact $\psi_h$)

Rate summary:
$2p + 1$ output-adapted
$2p + 2$ corrected

Drag coefficient error

2.5
3
4
5
6
(degrees of freedom)$^{-1/2}$

Error Estimation and Mesh Adaptation using Output Adjoints
Inviscid flow: drag convergence (approx $\psi_h$)

Five block-Jacobi smoothing iterations on fine space
Inviscid flow: drag convergence (approx $\psi_h$)

Drag coefficient error

Adapt, $p=1$
Adapt, $p=1$ (corrected)
Adapt, $p=2$
Adapt, $p=2$ (corrected)
Uniform, $p=1$
Uniform, $p=2$

Five block-Jacobi smoothing iterations on fine space

Adaptation, error estimates mostly unaffected
Inviscid flow: timing comparison (exact $\psi_h$)
Inviscid flow: timing comparison (approx $\psi_h$)
Inviscid flow: timing percentages

Exact fine-space adjoint solve

Approximate fine-space adjoint solve
Incorporating anisotropy with hanging nodes

- Crucial for high-Reynolds number simulations, esp. in 3D
- Create anisotropy by cutting in only one direction
- Solve local sub-problems to determine impact of anisotropy directly on the output error

Discrete choices

Choice 1

Choice 2

Choice 3 = both
Choosing the right cut [6: Ceze + Fidkowski, 2012]

- On an element, pick the cut \( i \) with the highest merit
- For each cut \( i \) define,

\[
\text{merit}(i) = \frac{\text{benefit}(i)}{\text{cost}(i)}
\]

**benefit(\( i \))**
- error addressed by cut \( i \)
- estimated using adjoint-weighted residual:

\[
\text{benefit}(i) = \sum_{\kappa_h \in \kappa_H} |\Psi_h^T R_h(U^H_h)|_{\kappa_h}
\]

\( h \) denotes the error estimation fine space, e.g. \( p + 1 \)

**cost(\( i \))**
- degrees of freedom (may be too simple)
- number of nonzeros in Jacobian (\( \sim \) cost of linear solve)
Transonic RANS airfoil

- NACA 0012
- $M = 0.8$, $\alpha = 1.25^\circ$
- $Re = 10^5$
- 10% fixed fraction
- $q = 3$ curved geometry
- $p = 2$ solution approximation
- RANS-SA model
- Adapt on drag, lift
- Discrete-choice $h$ cut optimization vs. isotropic

Initial mesh (1740 elements)

Mach number contours
Transonic RANS airfoil: output convergence

![Graph showing drag coefficient vs. degrees of freedom for different adjoint methods and uniform refinement.](image)

- Drag adjoint (isotropic)
- Lift adjoint (isotropic)
- Drag adjoint (anisotropic)
- Lift adjoint (anisotropic)
- Uniform refinement

The graph illustrates the convergence of the drag coefficient with increasing degrees of freedom for various adjoint methods and a uniform refinement strategy.
Transonic RANS airfoil: output convergence

![Graph showing lift coefficient vs. degrees of freedom for different adjoint methods and refinement strategies.]

- **Drag adjoint (isotropic)**
- **Lift adjoint (isotropic)**
- **Drag adjoint (anisotropic)**
- **Lift adjoint (anisotropic)**
- **Uniform refinement**

Error Estimation and Mesh Adaptation using Output Adjoints
Transonic RANS airfoil: drag-adapted meshes

- Isotropic
- Adapt iter 6
- 8736 elems

- Anisotropic
- Adapt iter 10
- 4816 elems
Transonic RANS airfoil: drag-adapted meshes

- **Isotropic**
  - Adapt iter 6
  - 8736 elems

- **Anisotropic**
  - Adapt iter 10
  - 4816 elems
Transonic RANS airfoil: drag-adapted meshes

- Isotropic
- Adapt iter 6
- 8736 elems

- Anisotropic
- Adapt iter 10
- 4816 elems
Transonic RANS flow over a wing

DPW III wing-alone case: $M_\infty = 0.76$, $Re = 5 \times 10^6$

- Initial mesh: cubic hex elements generated by agglomeration of linear multiblock meshes (first element $y^+ \approx 1$)

- Artificial viscosity shock capturing

- Spalart-Allmaras turbulence model with negative $\tilde{\nu}$ modification [1: Allmaras et al, 2012]

- Drag-adaptive simulation using $hp$ discrete choice algorithm [7: Ceze + Fidkowski, 2013]
DPW wing: adapted meshes

Original mesh, with $c_p$ contours

7th drag-adapted mesh

Mach/mesh using DOF cost

Mach/mesh using non-zero entries cost
DPW wing: comparison to uniform refinement

Dashed lines indicate outputs corrected with error estimate.

Error Estimation and Mesh Adaptation using Output Adjoint Adaptation
DPW wing: comparison to uniform refinement

Error Estimation and Mesh Adaptation using Output Adjoints

Dashed lines indicate outputs corrected with error estimate.
Outline

1. Introduction
2. Discretization
3. The Adjoint
4. Output Error Estimation
5. Adaptation
6. Mesh Optimization
7. References
Mesh adaptation using a metric field

- Unstructured meshes offer more geometric and adaptive flexibility over structured ones.
- Resolution information: size and shape of an element.
- This can be encoded in a metric field [5: Borouchaki, 1995] [12: Pennec, 2006] over the domain.
- We are interested in an adaptive method where the mesh is regenerated at each iteration using the current mesh and information from the solution.
- Key ingredients:
  1. Metric-conforming mesh generator
  2. Solution-based metric specification.
A Riemannian metric field

\[ M(\vec{x}) \in \mathbb{R}^{d \times d} \]

A symmetric positive definite (SPD) tensor field that provides a “yardstick” for measuring distances in different directions

Metric distance between \( \vec{x} \) and \( \vec{x} + \delta\vec{x} \): 
\[ \delta \ell = \sqrt{\delta\vec{x}^T M \delta\vec{x}} \]

- Eigenvectors of \( M \) are principal stretching directions
- Eigenvalues: \( \lambda_i = 1/h_i^2 \); \( h_i \) is the “stretching magnitude”: the distance along the eigenvector for unit metric measure
Mesh-conforming mesh generation

**Idea**

Make mesh in which each edge has the same metric length

Metric distance from $A$ to $B$: $\ell_{AB} = \int_{A}^{B} d\ell = \int_{A}^{B} \sqrt{d\vec{x}^T \mathcal{M} d\vec{x}}$

- e.g. BAMG = Bi-dimensional Anisotropic Mesh Generator
[5: Borouchaki, 1995]
- Input: background mesh and desired metric at nodes
- Output: metric-conforming mesh
Affine-invariant metric modification

Need a systematic way to alter $\mathcal{M}$: must keep SPD

- $\mathcal{M}_0 =$ current metric
- $\mathcal{M} =$ new metric $= \mathcal{M}_0^{\frac{1}{2}} \exp(S) \mathcal{M}_0^{\frac{1}{2}}$
- $S \in \mathbb{R}^{d \times d} =$ metric step matrix (symmetric)

Example ($\mathcal{M}_0 = \mathcal{I}$):

$$S = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$

Positive values on diagonal produce a contraction
Affine-invariant metric modification

Need a systematic way to alter $M$: must keep SPD

- $M_0 =$ current metric
- $M =$ new metric $= \sqrt{M_0} \exp(S) \sqrt{M_0}$
- $S \in \mathbb{R}^{d\times d} =$ metric step matrix (symmetric)

Example ($M_0 = I$):

$$S = \begin{bmatrix} 0 & 0.5 \\ 0.5 & 0 \end{bmatrix}$$

Positive values off diagonal produce negative shear
Affine-invariant metric modification

Need a systematic way to alter $\mathcal{M}$: must keep SPD

- $\mathcal{M}_0 =$ current metric
- $\mathcal{M} =$ new metric $= \mathcal{M}_0^{\frac{1}{2}} \exp(S) \mathcal{M}_0^{\frac{1}{2}}$
- $S \in \mathbb{R}^{d \times d} =$ metric step matrix (symmetric)

Example ($\mathcal{M}_0 = \mathcal{I}$):

$$S = \begin{bmatrix} 0 & -0.5 \\ -0.5 & 0 \end{bmatrix}$$

Negative values off diagonal produce positive shear
Affine-invariant metric modification

Need a systematic way to alter $\mathcal{M}$: must keep SPD

- $\mathcal{M}_0 = \text{current metric}$
- $\mathcal{M} = \text{new metric} = \mathcal{M}_0^{\frac{1}{2}} \exp(S) \mathcal{M}_0^{\frac{1}{2}}$
- $S \in \mathbb{R}^{d \times d} = \text{metric step matrix (symmetric)}$

Example ($\mathcal{M}_0 = I$):

$$S = \begin{bmatrix} 0.5 & -0.5 \\ -0.5 & 0 \end{bmatrix}$$

Combination of positive shear and contraction in $x$
Affine-invariant metric modification

Need a systematic way to alter $\mathcal{M}$: must keep SPD

- $\mathcal{M}_0 = $ current metric
- $\mathcal{M} = $ new metric = $\mathcal{M}_0^{\frac{1}{2}} \exp(S) \mathcal{M}_0^{\frac{1}{2}}$
- $S \in \mathbb{R}^{d \times d} = $ metric step matrix (symmetric)

Example ($\mathcal{M}_0 = \mathcal{I}$):

$$S = \begin{bmatrix} 5 & -0.5 \\ -0.5 & 0 \end{bmatrix}$$

Combination of positive shear and a lot of contraction in $x$
Affine-invariant metric modification

Need a systematic way to alter $\mathcal{M}$: must keep SPD

- $\mathcal{M}_0 =$ current metric
- $\mathcal{M} =$ new metric = \[ \mathcal{M}_0^{\frac{1}{2}} \exp(S) \mathcal{M}_0^{\frac{1}{2}} \]
- $S \in \mathbb{R}^{d \times d} =$ metric step matrix (symmetric)

Example ($\mathcal{M}_0 = \mathcal{I}$):

\[ S = \begin{bmatrix} 0 & -0.5 \\ -0.5 & 5 \end{bmatrix} \]

Combination of positive shear and a lot of contraction in $y$
Mesh-implied metric

- We need to “back out” a metric given a mesh, since we will be prescribing changes to the metric.
- Calculate elemental metric, $\mathcal{M}_e$, by enforcing that each edge length is of unit measure under the metric.

\[ \Delta \vec{x}_{AB}^T \mathcal{M}_e \Delta \vec{x}_{AB} = 1 \]
\[ \Delta \vec{x}_{BC}^T \mathcal{M}_e \Delta \vec{x}_{BC} = 1 \]
\[ \Delta \vec{x}_{CA}^T \mathcal{M}_e \Delta \vec{x}_{CA} = 1 \]

- This gives 3 equations for the three independent unknowns in the symmetric matrix representation of $\mathcal{M}_e$.
- Note: the elemental metric can be mapped to nodes via an affine-invariant average [12: Pennec et al, 2006]
A mesh optimization algorithm [16: Yano, 2012]

- **Given**: current mesh, primal and adjoint solutions
- **Determine**: metric step matrix, $S_v$, at each mesh vertex, $v$, that produces a mesh with the smallest output error at a fixed solution cost

**Key ingredients**

1. Error convergence model: $S_v \rightarrow$ output error
2. Cost model: $S_v \rightarrow$ solution cost
3. Iterative algorithm that equidistributes the marginal error-to-cost ratio

- Expect multiple iterations of optimization until error “bottoms out” at a fixed cost; can then increase allowable cost to further reduce error
Error convergence model

- $E_{e0} =$ current output error indicator on element $e$ (from AWR)
- $S_e =$ proposed metric step matrix on element $e$
- Model for error after metric modification with $S_e$:
  \[
  E_e = E_{e0} \exp \left[ \text{tr}(R_eS_e) \right]
  \]
- $R_e =$ error convergence rate tensor (identified by sampling)
- Note, this is a generalization to anisotropic shape changes of the more familiar isotropic model,
  \[
  E_e = E_{e0} \left( \frac{h}{h_0} \right)^r = E_{e0} \exp \left[ r \log (h/h_0) \right]
  \]
- Sum over elements to get the total error on the mesh,
  \[
  E = \sum_e E_e
  \]
Cost model

\[
\text{cost} = \text{degrees of freedom (do} f\text{) in solution approximation}
\]

- Assume \( p = \) approximation order = same for all elements
- \( C_{e0} = \) current cost on element \( e \), e.g. \( (p + 1)(p + 2)/2 \)
- New cost after application of step matrix \( S_e \),

\[
C_e = C_{e0} \exp \left[ \frac{1}{2} \text{tr}(S_e) \right] \frac{\text{Area}_0}{\text{Area}}
\]

- Note, the cost is just scaled by \( \frac{\text{Area}_0}{\text{Area}} = \) # new elements occupying the original area of element \( e \)
- Sum over elements to get the total cost on the mesh,

\[
C = \sum_e C_e
\]
From elements to vertices

\[ S_e = \frac{1}{|V_e|} \sum_{v \in V_e} S_v \]

\( V_e = \text{set of vertices adjacent to element } e \)

\( E_v = \text{set of elements adjacent to vertex } v \)

### Error

\[
\mathcal{E} = \sum_e \mathcal{E}_e \\
\frac{\partial \mathcal{E}}{\partial S_v} = \sum_{e \in E_v} \left[ \frac{\partial \mathcal{E}_e}{\partial S_e} \frac{\partial S_e}{\partial S_v} \right] \mathcal{E}_e R_e \frac{1}{|V_e|}
\]

### Cost

\[
\mathcal{C} = \sum_e \mathcal{C}_e \\
\frac{\partial \mathcal{C}}{\partial S_v} = \sum_{e \in E_v} \left[ \frac{\partial \mathcal{C}_e}{\partial S_e} \frac{\partial S_e}{\partial S_v} \right] \mathcal{C}_e R_e \frac{1}{|V_e|}
\]
Optimization algorithm

Given:

- $\mathcal{M}_0 = \text{mesh-implied metric (elements } \rightarrow \text{ nodes)}$
- $\mathcal{E}_{e0} = \text{error indicators on elements (AWR)}$
- $R_e = \text{error rate tensor (sampling)}$

Calculate:

$\mathcal{S}_v = \text{step matrix at vertices that minimizes error at a fixed cost}$

- We separate $\mathcal{S}_v$ into size (trace) and shape (trace-free) contributions,

$$\mathcal{S}_v = s_v \mathcal{I} + \tilde{\mathcal{S}}_v$$

- Derivatives of the error with respect to $s_v$ and $\tilde{\mathcal{S}}_v$ are

$$\frac{\partial \mathcal{E}}{\partial s_v} = \text{tr} \left( \frac{\partial \mathcal{E}}{\partial \mathcal{S}_v} \right), \quad \frac{\partial \mathcal{E}}{\partial \tilde{\mathcal{S}}_v} = \frac{\partial \mathcal{E}}{\partial \mathcal{S}_v} - \frac{\partial \mathcal{E}}{\partial s_v} \mathcal{I}$$
Optimization algorithm (continued)

Use an iterative approach [16: Yano, 2012]

- Initialize $S_v = 0, \forall v$, set $\delta s = s_{\text{max}} / n_{\text{step}} = 2 \log 2 / 20$
- Loop $i = 1 : n_{\text{step}}$,
  1. $S_v \rightarrow S_e \rightarrow \frac{\partial \varepsilon_e}{\partial S_e}, \frac{\partial C_e}{\partial S_e} \rightarrow$ linearizations w.r.t $s_v$ and $\tilde{S}_v$
  2. Define $\lambda_v = \frac{\partial \varepsilon / \partial s_v}{\partial C / \partial s_v}$ = marginal error to marginal cost ratio of mesh refinement
     - Refine 30% of vertices with the largest $|\lambda_v|$: $S_v = S_v + \delta s I$
     - Coarsen 30% of the vertices with the smallest $|\lambda_v|$: $S_v = S_v - \delta s I$
  3. Update the trace-free part of $S_v$, $S_v = S_v + \delta s (\frac{\partial \varepsilon / \partial \tilde{S}_v}{\partial \varepsilon / \partial s_v})$
  4. Rescale $S_v \rightarrow S_v + \beta I$, to meet total cost constraint via $\beta = \frac{2}{d} \log \frac{C_{\text{target}}}{C}$, where $C_{\text{target}}$ is the target cost
Error sampling to obtain $R_e$

- An a-posteriori data-driven approach
- Idea: cut an element in different ways, measure change in error indicator, and fit model via least-squares regression

Determine entries of $R_e$ (symmetric) that minimize misfit between the model and observed errors for each refinement option $i$

$$\text{misfit} = \sum_i \left[ \log \frac{\mathcal{E}_{ei}}{\mathcal{E}_{e0}} - \text{tr}(R_e S_{ei}) \right]^2$$

$\mathcal{E}_{ei} = \text{output error for refinement option } i$

$S_{ei} = \text{average metric step matrix for refinement option } i$
Estimating errors after refinement

- Error left after refining via option $i$, 
  \[ \mathcal{E}_{ei} = \mathcal{E}_{e0} - \Delta \mathcal{E}_{ei} \]

- $\Delta \mathcal{E}_{ei}$ = error between coarse solution and refinement option $i$
  \[ \Delta \mathcal{E}_{ei} \equiv | \mathcal{R}_h^{p+1}(u^p_h, \tilde{\psi}_{hi})| \]

- $\tilde{\psi}_{hi}$ = adjoint after refining via option $i$, obtained by projecting the fine-space adjoint $\psi_{h}^{p+1}$

Note, can use pre-calculated projection matrices for each refinement option
Combining adaptation and optimization

1. Start with a coarse mesh at a certain cost = $dof$

2. Run multiple (~10) mesh optimization iterations at fixed cost
   - Each iteration requires primal and adjoint solves
   - Solves are quick since starting from good initial guesses
   - Error will drop, then stagnate/oscillate
   - Use results from final run or average of last few runs

3. Increase $dof$ cost by a prescribed factor if need more accuracy and can afford more cost; return to step 2
Example: NACA 0012 in inviscid flow

Euler equations, $M_\infty = 0.5$, $\alpha = 2^\circ$, $\gamma = 1.4$, output = drag

Mach number contours
Example: NACA 0012 in inviscid flow

Euler equations, $M_\infty = 0.5$, $\alpha = 2^\circ$, $\gamma = 1.4$, output = drag

Initial mesh: 356 triangles, farfield @2000c
\( p = 2, \) 15 optimization iterations at each \( \text{dof} \)
NACA 0012 in inviscid flow: output convergence

Compare to uniform refinement at different orders $p$

![Graph showing drag coefficient error for different $p$ values]
NACA 0012 in inviscid flow: optimized meshes

\[ p = 1, \text{dof} = 2000 \]

\[ p = 1, \text{dof} = 4000 \]

\[ p = 1, \text{dof} = 8000 \]

\[ p = 1, \text{dof} = 16000 \]
NACA 0012 in inviscid flow: optimized meshes

$p = 2, \text{dof} = 2000$

$p = 2, \text{dof} = 4000$

$p = 2, \text{dof} = 8000$

$p = 2, \text{dof} = 16000$
NACA 0012 in inviscid flow: optimized meshes

$p = 3, \text{dof} = 2000$

$p = 3, \text{dof} = 4000$

$p = 3, \text{dof} = 8000$

$p = 3, \text{dof} = 16000$
Example: RAE 2822 in transonic flow

RANS-SA, $M_\infty = 0.73$, $\alpha = 2.79^\circ$, $Re = 6.5M$, output = drag

Mach number contours
Example: RAE 2822 in transonic flow

RANS-SA, $M_\infty = 0.73, \alpha = 2.79^\circ, Re = 6.5M$, output = drag

Initial mesh: 758 triangles, farfield @2000c
RAE 2822 in transonic flow: sample run

$p = 2$, 15 optimization iterations at each $\text{dof}$
RAE 2822 in transonic flow: output convergence

Compare to uniform refinement at different orders $p$

![Graph showing drag coefficient error vs. $1/\sqrt{\text{dof}}$ for optimized and uniform refinement at different orders $p$. The graph has logarithmic scales on the y-axis for drag coefficient error ranging from $10^{-7}$ to $10^{-1}$, and on the x-axis for $1/\sqrt{\text{dof}}$ ranging from $10^{-3}$ to $10^{-2}$. Legend includes symbols for optimized and uniform refinement at $p=1$, $p=2$, and $p=3$. The optimized refinement shows better convergence for all orders compared to uniform refinement.]
RAE 2822 in transonic flow: optimized meshes

\[ p = 1, \, \text{dof} = 5000 \]

\[ p = 1, \, \text{dof} = 10000 \]

\[ p = 1, \, \text{dof} = 20000 \]

\[ p = 1, \, \text{dof} = 40000 \]
RAE 2822 in transonic flow: optimized meshes

$p = 2, \text{dof} = 5000$

$p = 2, \text{dof} = 10000$

$p = 2, \text{dof} = 20000$

$p = 2, \text{dof} = 40000$
RAE 2822 in transonic flow: optimized meshes

$p = 3, \text{dof} = 5000$

$p = 3, \text{dof} = 10000$

$p = 3, \text{dof} = 20000$

$p = 3, \text{dof} = 40000$
Summary

- We can quantify numerical error and adapt the mesh to reduce it at its source.
- This requires an adjoint and fine-space calculations.
- Exact fine-space adjoints yield effective error estimates that we can use as corrections.
- Approximate fine-space adjoints (e.g. via Jacobi smoothing) are still good for adaptation.
- Mesh anisotropy is critical for efficiently resolving boundary layers.
- Hanging-node anisotropy through single cuts is limited by structure of original mesh.
- Unstructured metric-based mesh regeneration offers an opportunity to globally optimize meshes for accurate output prediction at minimal cost.
Outline

1. Introduction
2. Discretization
3. The Adjoint
4. Output Error Estimation
5. Adaptation
6. Mesh Optimization
7. References
Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. 
Seventh International Conference on Computational Fluid Dynamics (ICCFD7) 1902, 2012.

*Shock Capturing with PDE-Based Artificial Viscosity for an Adaptive Higher–Order Discontinuous Galerkin Finite Element Method.* 

A posteriori error estimates for higher order Godunov finite volume methods on unstructured meshes. 
Hermes Penton.

A feed-back approach to error control in finite element methods: Basic analysis and examples. 

Mailleur bidimensionnel de Delaunay gouverné par une carte de métriques. Partie I: Algorithmes. 

An anisotropic hp-adaptation framework for functional prediction. 

Drag prediction using adaptive discontinuous finite elements. 

An adaptive simplex cut-cell method for discontinuous Galerkin discretizations of the Navier-Stokes equations. 
Adjoint equations in CFD: duality, boundary conditions and solution behavior.

An a Posteriori Error Control Framework for Adaptive Precision Optimization Using Discontinuous Galerkin Finite Element Method.


A riemannian framework for tensor computing.


Goal-oriented hp-adaptivity for elliptic problems.

Grid adaptation for functional outputs: application to two-dimensional inviscid flows.