Introduction

Derivative Calculation Methods
  Hyper-Dual Numbers

Supersonic Business Jet Design Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Computational Fluid Dynamics Codes
  Differentiation of the Solution of a Linear System
  Approach for Iterative Procedures

Transonic Inviscid Airfoil Shape Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Conclusions
Outline

Introduction

Derivative Calculation Methods
  Hyper-Dual Numbers

Supersonic Business Jet Design Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Computational Fluid Dynamics Codes
  Differentiation of the Solution of a Linear System
  Approach for Iterative Procedures

Transonic Inviscid Airfoil Shape Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Conclusions
Numerical optimization methods systematically vary the inputs to an objective function in order to find the maximum or minimum

- Requires many function evaluations
- Methods that use first derivative information typically converge in fewer iterations
- Using second derivatives can provide a further benefit

Tradeoff between convergence and having to compute derivatives

- Newton’s Method converges quadratically, but requires the gradient and Hessian
- Steepest Descent converges linearly, but requires only the gradient
- Quasi-Newton methods converge super-linearly, using the gradient to build an approximation to the Hessian
Need a good method for computing second derivatives

- Accurate
- Computationally Efficient
- Easy to Implement

Methods that work well for first derivatives may not have the same beneficial properties when applied to second derivatives
Introduction

Derivative Calculation Methods
  Hyper-Dual Numbers

Supersonic Business Jet Design Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Computational Fluid Dynamics Codes
  Differentiation of the Solution of a Linear System
  Approach for Iterative Procedures

Transonic Inviscid Airfoil Shape Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Conclusions
Forward-difference (FD) Approximation:

\[
\frac{\partial f(x)}{\partial x_j} = \frac{f(x + he_j) - f(x)}{h} + O(h)
\]

Central-Difference (CD) approximation:

\[
\frac{\partial f(x)}{\partial x_j} = \frac{f(x + he_j) - f(x - he_j)}{2h} + O(h^2)
\]

Subject to truncation error and subtractive cancellation error

- Truncation error is associated with the higher order terms that are ignored when forming the approximation.
- Subtractive cancellation error is a result of performing these calculations on a computer with finite precision.
Taylor series with an imaginary step:

\[ f(x + ih) = f(x) + ihf'(x) - \frac{1}{2!} h^2 f''(x) - i \frac{h^3 f'''(x)}{3!} + \ldots \]

\[ f(x + ih) = \left( f(x) - \frac{1}{2!} h^2 f''(x) + \ldots \right) + ih \left( f'(x) - \frac{1}{3!} h^2 f'''(x) + \ldots \right) \]

First-Derivative Complex-Step Approximation:

\[ \frac{\partial f(x)}{\partial x_j} = \frac{\text{Im} \left[ f(x + ihe^j) \right]}{h} + O(h^2) \]

- First derivatives are subject to truncation error but are not subject to subtractive cancellation error.

Accuracy of First-Derivative Calculations

Error in the First Derivative

\[ f(x) = \frac{e^x}{\sqrt{\sin(x)^3 + \cos(x)^3}} \]
Accuracy of Second-Derivative Calculations

Error in the Second Derivative

- Complex-Step
- Forward-Difference
- Central-Difference
- Hyper-Dual Numbers

Step Size, h

Error

10^0 10^10 10^20
10^-10
10^-20
10^-30

aerospacedesignlab
Hyper-dual numbers have one real part and three non-real parts:

\[ x = x_0 + x_1\epsilon_1 + x_2\epsilon_2 + x_3\epsilon_1\epsilon_2 \]

\[ \epsilon_1^2 = \epsilon_2^2 = 0 \]
\[ \epsilon_1 \neq \epsilon_2 \neq 0 \]
\[ \epsilon_1\epsilon_2 = \epsilon_2\epsilon_1 \neq 0 \]

Taylor series truncates exactly at second-derivative term:

\[ f(x + h_1\epsilon_1 + h_2\epsilon_2 + 0\epsilon_1\epsilon_2) = f(x) + h_1 f'(x)\epsilon_1 + h_2 f'(x)\epsilon_2 + h_1 h_2 f''(x)\epsilon_1\epsilon_2 \]

- No truncation error and no subtractive cancellation error

Fike and Alonso, AIAA 2011-886
Evaluate a function with a hyper-dual step:

\[ f(x + h_1 \epsilon_1 e_i + h_2 \epsilon_2 e_j + 0 \epsilon_1 \epsilon_2) \]

Derivative information can be found by examining the non-real parts:

\[
\frac{\partial f(x)}{\partial x_i} = \frac{\epsilon_1 \text{part} \left[ f(x + h_1 \epsilon_1 e_i + h_2 \epsilon_2 e_j + 0 \epsilon_1 \epsilon_2) \right]}{h_1}
\]

\[
\frac{\partial f(x)}{\partial x_j} = \frac{\epsilon_2 \text{part} \left[ f(x + h_1 \epsilon_1 e_i + h_2 \epsilon_2 e_j + 0 \epsilon_1 \epsilon_2) \right]}{h_2}
\]

\[
\frac{\partial^2 f(x)}{\partial x_i \partial x_j} = \frac{\epsilon_1 \epsilon_2 \text{part} \left[ f(x + h_1 \epsilon_1 e_i + h_2 \epsilon_2 e_j + 0 \epsilon_1 \epsilon_2) \right]}{h_1 h_2}
\]
To use hyper-dual numbers, every operation in an analysis code must be modified to operate on hyper-dual numbers instead of real numbers.

- Basic Arithmetic Operations: Addition, Multiplication, etc.
- Logical Comparison Operators: $\geq$, $\neq$, etc.
- Mathematical Functions: exponential, logarithm, sine, absolute value, etc.
- Input/Output Functions to write and display hyper-dual numbers

Hyper-dual numbers are implemented as a class using operator overloading in C++ and MATLAB.

- Change variable types
- Body and structure of code unaltered

Implementation available from http://adl.stanford.edu/
Hyper-Dual number operations are inherently more expensive than real number operations.

- Hyper-Dual addition: 4 real additions
- Hyper-Dual multiplication: 9 real multiplications and 5 additions
  - One HD operation up to 14 times a real operation

Forming both the gradient and Hessian of $f(x)$, for $x \in \mathbb{R}^n$, requires $n$ first-derivative calculations and $\frac{n(n+1)}{2}$ second-derivative calculations.

- Forward-Difference: $(n + 1)^2$ function evaluations
- Central-Difference: $2n(n + 2)$ function evaluations
- Hyper-Dual Numbers: $\frac{n(n+1)}{2}$ hyper-dual function evaluations
  - Approximately 7 times FD and 3.5 times CD

For some functions this can be greatly reduced.
Outline

Introduction

Derivative Calculation Methods
Hyper-Dual Numbers

Supersonic Business Jet Design Optimization
Problem Formulation
Comparison of Derivative Calculation Methods

Computational Fluid Dynamics Codes
Differentiation of the Solution of a Linear System
Approach for Iterative Procedures

Transonic Inviscid Airfoil Shape Optimization
Problem Formulation
Comparison of Derivative Calculation Methods

Conclusions
Optimization of a Supersonic Business Jet (SSBJ) design using Newton’s method

- **Objective Function** a weighted combination of aircraft range and sonic boom strength at the ground
- **33 Design Variables** describing geometry, interior structure and operating conditions of the SSBJ
- **Low-Fidelity Conceptual-Design-Level Analysis Routines**

Compare runtimes for Hyper-Dual numbers, Forward Difference, and Central Difference

Modify part of the objective function to decrease the cost of using hyper-dual numbers
Breguet Range Equation:

\[ R = M \cdot a \left( \frac{L}{D} \right) \left( \frac{1}{SFC} \right) \left( -\log \left( 1 - \frac{W_f}{W_t} \right) \right) \]

- Propulsion routine calculates engine performance and weight
- Weight routine calculates weights and structural loads
- Aerodynamics routine calculates lift and drag

Sonic Boom Procedure:
- Calculate an Aircraft Shape Factor [Carlson, NASA-TP-1122, 1978]
- Use this shape factor to create a near-field pressure signature
- Propagate signature to ground using the Waveform Parameter Method [Thomas, NASA-TN-D-6832, 1972]
Three methods used to compute gradient and Hessian

- Execution time for hyper-dual numbers is 7 times Forward-Difference time
- Execution time for hyper-dual numbers is 3.6 times Central-Difference time
  - Reasonable based on earlier discussion

Modify one routine in the sonic boom calculation procedure

- Execution time for hyper-dual numbers is 0.9 times Forward-Difference time
- Execution time for hyper-dual numbers is 0.46 times Central-Difference time
An aircraft shape factor was found during the sonic boom calculation procedure. This involved finding the location of the maximum effective area.

Maximum found using golden-section line search:
- Could have used any number of alternatives, including sweeping through at fixed intervals.
- Inner workings of the method should not affect derivatives.
This suggests a method for reducing the computational cost of using hyper-dual numbers:

- Find location of maximum value using real numbers
- Then perform one evaluation using hyper-dual numbers to calculate derivatives

For this particular situation, computational cost reduced by a factor of 8

This can be extended to general objective functions involving iterative procedures

- Converge the procedure using real numbers
- Then perform one iteration using hyper-dual numbers to calculate derivatives
Residual Equations

Drive the flux residuals to zero, \( b(q, x) = 0 \)

\[
A(x) dq(x) = b(x)
\]

Differentiating both sides with respect to the \( i^{th} \) component of \( x \) gives

\[
\frac{\partial A(x)}{\partial x_i} dq(x) + A(x) \frac{\partial dq(x)}{\partial x_i} = \frac{\partial b(x)}{\partial x_i}
\]

Differentiating this result with respect to the \( j^{th} \) component of \( x \) gives

\[
\frac{\partial^2 A(x)}{\partial x_j \partial x_i} dq(x) + \frac{\partial A(x)}{\partial x_i} \frac{\partial dq(x)}{\partial x_j} + \frac{\partial A(x)}{\partial x_j} \frac{\partial dq(x)}{\partial x_i} + A(x) \frac{\partial^2 dq(x)}{\partial x_j \partial x_i} = \frac{\partial^2 b(x)}{\partial x_j \partial x_i}
\]
This can be solved as:

\[
\begin{bmatrix}
A(x) & 0 & 0 & 0 \\
\frac{\partial A(x)}{\partial x_i} & A(x) & 0 & 0 \\
\frac{\partial A(x)}{\partial x_j} & 0 & A(x) & 0 \\
\frac{\partial^2 A(x)}{\partial x_j \partial x_i} & \frac{\partial A(x)}{\partial x_j} & \frac{\partial A(x)}{\partial x_i} & A(x)
\end{bmatrix}
\begin{bmatrix}
dq(x) \\
\frac{\partial dq(x)}{\partial x_i} \\
\frac{\partial dq(x)}{\partial x_j} \\
\frac{\partial^2 dq(x)}{\partial x_j \partial x_i}
\end{bmatrix}
= \begin{bmatrix}
b(x) \\
\frac{\partial b(x)}{\partial x_i} \\
\frac{\partial b(x)}{\partial x_j} \\
\frac{\partial^2 b(x)}{\partial x_j \partial x_i}
\end{bmatrix}
\]

Or

\[A(x) dq(x) = b(x)\]

\[A(x) \frac{\partial dq(x)}{\partial x_i} = \frac{\partial b(x)}{\partial x_i} - \frac{\partial A(x)}{\partial x_i} dq(x)\]

\[A(x) \frac{\partial dq(x)}{\partial x_j} = \frac{\partial b(x)}{\partial x_j} - \frac{\partial A(x)}{\partial x_j} dq(x)\]

\[A(x) \frac{\partial^2 dq(x)}{\partial x_j \partial x_i} = \frac{\partial^2 b(x)}{\partial x_j \partial x_i} - \frac{\partial^2 A(x)}{\partial x_j \partial x_i} dq(x) - \frac{\partial A(x)}{\partial x_i} \frac{\partial dq(x)}{\partial x_j} - \frac{\partial A(x)}{\partial x_j} \frac{\partial dq(x)}{\partial x_i}\]
For a converged solution, $dq(x) \equiv 0$. This simplifies the procedure to:

$$A(x) \frac{\partial dq(x)}{\partial x_i} = \frac{\partial b(x)}{\partial x_i}$$

$$A(x) \frac{\partial dq(x)}{\partial x_j} = \frac{\partial b(x)}{\partial x_j}$$

$$A(x) \frac{\partial^2 dq(x)}{\partial x_j \partial x_i} = \frac{\partial^2 b(x)}{\partial x_j \partial x_i} - \frac{\partial A(x)}{\partial x_i} \frac{\partial dq(x)}{\partial x_j} - \frac{\partial A(x)}{\partial x_j} \frac{\partial dq(x)}{\partial x_i}$$

If we now assume that we have converged the first derivative terms, then the second-derivative equation reduces to

$$A(x) \frac{\partial^2 dq(x)}{\partial x_j \partial x_i} = \frac{\partial^2 b(x)}{\partial x_j \partial x_i}$$
This approach is applied to the CFD code JOE

- Parallel, unstructured, 3-D, multi-physics, unsteady Reynolds-Averaged Navier-Stokes code
- Written in C++, which enables the straightforward conversion to hyper-dual numbers
- Can use PETSc to solve the linear system

Derivatives converge at same rate as flow solution

- No benefit to starting with a converged solution?
- JOE uses an approximate Jacobian
- Need the exact Jacobian
Outline

Introduction

Derivative Calculation Methods
  Hyper-Dual Numbers

Supersonic Business Jet Design Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Computational Fluid Dynamics Codes
  Differentiation of the Solution of a Linear System
  Approach for Iterative Procedures

Transonic Inviscid Airfoil Shape Optimization
  Problem Formulation
  Comparison of Derivative Calculation Methods

Conclusions
2D Euler solver

- Written in C++ using templates
- Cell-centered finite-volume discretization
- Roe’s approximate Riemann solver
- MUSCL reconstruction via the Van Albada limiter
- Last few iterations use the exact Jacobian found using the automatic differentiation tool Tapenade

Optimization performed using IPOPT

- Provide gradients and Hessians of the objective function and the constraints
- Uses BFGS to build an approximation to the Hessian if only the gradients are provided
Convergence for a NACA-0012 airfoil at $M = 0.78$ and $\alpha = 1.2^\circ$
The shape of the airfoil is parametrized using a fifth order (with rational basis functions of degree four) NURBS curve with 11 control points.

The trailing edge is fixed at \((x, y) = (1, 0)\).

Position and weight of the remaining 9 control points gives 27 design variables. Combined with the angle of attack, this results in a total of 28 design variables.
Lift Constraint: $c_l = 0.5$

Geometric Constraints:

- Location of the leading edge at $(x, y) = (0, 0)$
- Maximum curvature must be smaller than a user-prescribed value
- Maximum thickness must be larger than a user-prescribed value
- Trailing edge angle must be larger than a user-prescribed value
- Regularity constraints on the location of the control points
Inviscid drag minimization at $M = 0.78$

Baseline: NACA-0012 airfoil at $M = 0.78$ and $\alpha = 1.2^\circ$

For the baseline, the shock on the suction side is clearly visible, leading to a $c_d = 1.307 \cdot 10^{-2}$
Optimal design using different optimization software, SNOPT

- Optimal geometries are different
- Shock has completely disappeared
- Resulting drag is solely caused by the discretization error
The method for efficiently using Hyper-Dual Numbers is followed.

- The code uses templates, which allows the variable type to be changed arbitrarily
- The exact Jacobian is computed and used for the last few iterations of the flow solver
- The LU decomposition of the exact Jacobian is stored

One iteration is needed to solve for each first derivative, and one iteration is required for each second derivative.

- In general, the cost of obtaining a derivative is identical to the cost of one Newton iteration of the flow field
- For this particular case, because a direct solver is used for which the LU decomposition is stored, the derivative information is obtained for a fraction of the cost of a Newton iteration
The required second-derivative calculations were carried out using three different techniques.

- Hyper-Dual Numbers
- Central-Difference Approximation
- Complex-Step/Finite-Difference Hybrid

\[
\frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_k} = \frac{\text{Im} \left[ f(\mathbf{x} + ih_1 \mathbf{e}_j - 2h_2 \mathbf{e}_k) \right] - \text{Im} \left[ f(\mathbf{x} + ih_1 \mathbf{e}_j + 2h_2 \mathbf{e}_k) \right]}{12h_1 h_2} + 2 \left( \frac{\text{Im} \left[ f(\mathbf{x} + ih_1 \mathbf{e}_j + h_2 \mathbf{e}_k) \right] - \text{Im} \left[ f(\mathbf{x} + ih_1 \mathbf{e}_j - h_2 \mathbf{e}_k) \right]}{3h_1 h_2} \right) + O \left( h_1^2 + h_2^4 \right)
\]
The central-difference and complex-step/finite-difference hybrid require appropriate values for the step size.

Relative error and value of \( \frac{\partial^2 c_l}{\partial \alpha^2} \) as the step size is varied.
Optimal step size more sensitive for angle of attack than other design variables

Complex-Step/Finite-Difference Hybrid:
- The magnitude of the imaginary disturbance $h_1$ is typically chosen of the order $10^{-30}$ or even smaller.
- For the real valued disturbance $h_2$ the choice is more critical.
  - $h_2 = 1.0 \cdot 10^{-8}$ appears suitable for $\alpha$
  - $h_2 = 1.0 \cdot 10^{-7}$ is more suited for the other variables

Central-Difference Formula:
- $h = 1.0 \cdot 10^{-7}$ for $\alpha$
- $h = 1.0 \cdot 10^{-6}$ otherwise
Optimization is carried out using the three methods for explicitly computing the Hessian, and a Quasi-Newton method using a limited memory BFGS.

- Very similar convergence behavior
- Explicit Hessian methods coincide for first 6 iterations
- Explicit Hessian methods smoother than BFGS
Execution Time Comparison

<table>
<thead>
<tr>
<th>Method of Hessian matrix computation</th>
<th>Normalized duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-BFGS approximation</td>
<td>1.00</td>
</tr>
<tr>
<td>Hyper-Dual Numbers</td>
<td>1.37</td>
</tr>
<tr>
<td>Central-Difference approximation</td>
<td>1.18</td>
</tr>
<tr>
<td>Complex-Step/Finite-Difference Hybrid</td>
<td>1.95</td>
</tr>
</tbody>
</table>

- BFGS is the fastest, it avoids explicitly computing the Hessian.
- The finite-difference method requires nine flow solutions to compute the entries in the Hessian each of which requires three Newton iterations to be performed to obtain a converged flow solution.
- Using Hyper-Dual Numbers requires only one additional flow solution, involving two Newton iterations, for each entry of the Hessian matrix.
Outline

Introduction

Derivative Calculation Methods
   Hyper-Dual Numbers

Supersonic Business Jet Design Optimization
   Problem Formulation
   Comparison of Derivative Calculation Methods

Computational Fluid Dynamics Codes
   Differentiation of the Solution of a Linear System
   Approach for Iterative Procedures

Transonic Inviscid Airfoil Shape Optimization
   Problem Formulation
   Comparison of Derivative Calculation Methods

Conclusions
Hyper-Dual numbers can be used to compute exact gradients and Hessians

- The computational cost can be greatly reduced for some objective functions, including those involving iterative procedures.
- For iterative procedures, an efficient strategy is to converge the procedure using real numbers, and then perform one iteration using hyper-dual numbers to compute the derivatives.

Optimization of a Supersonic Business Jet Design:

- Computational cost reduced by a factor of 8
- Makes hyper-dual numbers both more accurate and less expensive to use than finite differences
Conclusions

Application of Hyper-Dual numbers to a CFD code
  • Differentiation of the solution of a linear system
  • Simplified if start with a converged solution
  • Get derivatives in one or two Newton iterations
  • Initial testing indicated no benefit
    • Need to use exact Jacobian

Inviscid Transonic Airfoil Optimization
  • 2D Euler code with the exact Jacobian
  • Accuracy of the Hessian had little impact on the convergence of the optimization
  • Cost of using Hyper-Dual numbers not unreasonable
  • Avoids searching for a good step size
Questions?
Backup Slides
JOE Results

NACA 0012, $M=0.8$, $\alpha=1^\circ$, inviscid, 1st order.
Convergence of $\partial r_1/\partial \alpha$.

- Density, Cold Start, GMRES
- 1st Deriv, Cold Start, GMRES
- 1st Deriv, Restart, Flow Updated, GMRES
- 1st Deriv, Restart, Flow Frozen, GMRES
- 1st Deriv, Restart, Flow Frozen, LU

NACA 0012, $M=0.8$, $\alpha=1^\circ$, inviscid, 1st order.
Convergence of $\partial^2 r_1/\partial \alpha \partial M$.

- Density, Cold Start, GMRES
- 2nd Deriv, Cold Start, GMRES
- 2nd Deriv, Restart, Flow Updated, GMRES
- 2nd Deriv, Restart, Flow Frozen, GMRES
- 2nd Deriv, Restart, Flow Frozen, LU

aerospacedesignlab