Computers and Fluids 168 (2018) 265–284

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Benchmark solutions

Approximate Hessian for accelerated convergence of aerodynamic shape optimization problems in an adjoint-based framework

Doug Shi-Dong[1,*], Siva Nadarajah[2]

Department of Mechanical Engineering, McGill University, Montreal, Quebec, Canada

ARTICLE INFO

Article history:
Received 19 July 2017
Revised 6 March 2018
Accepted 13 April 2018
Available online 14 April 2018

Keywords:
Hessian
Aerodynamic shape optimization
Adjoint

ABSTRACT

The current work explores the use of an approximate Hessian to accelerate the convergence of an adjoint-based aerodynamic shape optimization framework. Exact analytical formulations of the direct-direct, adjoint-direct, adjoint-adjoint, and direct-adjoint Hessian approaches are presented and the equivalence between the adjoint-adjoint and direct-adjoint formulations is demonstrated. An approximation of the Hessian is obtained from the analytical formulation by partially solving first-order sensitivities to reduce computational time, while neglecting second-order sensitivities to ease implementation. Error bounds on the resulting approximation are presented for the first-order sensitivities through perturbation analysis. The proposed method is first assessed using an inverse pressure problem for a quasi-one-dimensional Euler flow. Additionally, three-dimensional inviscid transonic test cases are used to demonstrate the effectiveness of the method.

1. Introduction

The objective of this work is to develop the necessary tools to design environmentally and economically friendly aircraft. Since the use of computational fluid dynamics in conjunction with numerical optimization has been a dominant design method in recent years, improvement of flow solvers and optimization algorithms lead to faster and more robust design tools. The current state-of-the-art framework for aerodynamic shape optimization (ASO) uses a sequential quadratic programming (SQP) approach with adjoint-based gradients.

The adjoint approach developed by Pironneau [1] and extended by Jameson [2] provides the optimizer with the gradients of the objective function with respect to the design variables. The computational cost of the adjoint is independent of the number of design variables, making it superior to finite differences or the direct method in ASO. In effect, gradient-based algorithms such as gradient descent use the sensitivities to march towards a descent direction.

Multiple techniques have been devised to accelerate the optimization problem convergence, most of which involve a Hessian formulation. Gradient descent can alternatively be interpreted as Newton's method with an identity matrix as its Hessian approximation. The most simplistic change to the gradient descent algorithm is to scale the design variables such that the search direction is not disproportionate.

The resulting scaling can be represented as a Hessian with scaled diagonal entries that seeks to imitate curvature information. Quasi-Newton methods such as BFGS and symmetric rank-one (SR1) approximate the Hessian at each design cycle from the change in design variables and gradients. Analytical approximate Hessians that do not require second-order flow sensitivities have been formulated for inverse design problems due to their quadratic nature [3,4]. The idea of gradient smoothing using Sobolev gradients [2] has extended to more complex approximate Hessians via shape calculus and Fourier analysis [5,6]. With the advent of automatic differentiation (AD), exact second-order sensitivities [7–9] have been employed in ASO problems. However, the high cost of computing the numerically exact discrete Hessian has lead to truncated-Newton methods [10,11] that use conjugate-gradient or Newton–Krylov methods to approximate the search direction with matrix-vector products of the Hessian. Due to their low computational and implementation cost, identity-initialized quasi-Newton methods are still the workhorse of most ASO frameworks such as SNOPT [12], NLPQLP [13], and IPOPT [14] and will be referred as the conventional approach.

The goal of the BFGS algorithm is to update the Hessian matrix at every design cycle, such that it properly estimates the design space curvature. In other words, it retrieves some second-order in-

* Corresponding author.
E-mail addresses: doug.shi-dong@mail.mcgill.ca (D. Shi-Dong), siva.nadarajah@mcgill.ca (S. Nadarajah).
[1] Graduate Student.
[2] Assistant Professor.

https://doi.org/10.1016/j.compfluid.2018.04.019
0045-7930/

formation at every step. During the final iterations of the optimization, it hopes to perform Newton steps such that a local minimum is found superlinearly. However, for a quadratic design space, the BFGS algorithm requires as many steps as the number of design variables to retrieve the exact Hessian [15]. A non-linear design space will require even more design cycles to retrieve a good approximation. Ideally, Newton steps are taken from the initial design to the optimal design to converge quadratically. Unfortunately, the prohibitive cost of evaluating the Hessian disfavor the use of Newton's method.

The work of Papadimitriou and Giannakoglou [8,16–19] explores the use of an exactly-initialized BFGS algorithm in an ASO framework. Although the Hessian is only evaluated once, the initial cost is still too large to be effective. It is especially risky when the initial design lies in a non-convex region, where the Hessian must be modified. Furthermore, the initial time required to compute the Hessian is a large investment since the optimizer spends most of its computational time without stepping towards an optimum.

The current work aims to drastically reduce the initial cost by evaluating an approximate Hessian. The same initialized-BFGS framework as Papadimitriou and Giannakoglou [8] is used, but with an approximate initial Hessian. A flowchart of the optimization framework is shown in Fig. 1. The approximation is recovered by partially solving flow sensitivities, which are the most computationally intensive terms. The error incurred by the partial convergence of the flow sensitivities on the Hessian eigenvalues is investigated. Given an existing aerodynamic optimization framework, the proposed method can be easily implemented by further approximating the Hessian by discarding second-order flow sensitivities.

Four analytical formulations are presented in the first sections based on previous works [7,8]. Among those, the equivalence between the adjoint-adjoint and direct-adjoint formulations is demonstrated. Subsequently, parts of the Hessian are approximated in order to alleviate computational and implementation cost. It is followed by a mathematical bound on the incurred error through perturbation analysis. Finally, the proposed methods are tested using a quasi-one-dimensional Euler flow inverse pressure problem and three-dimensional inviscid transonic optimization test cases.

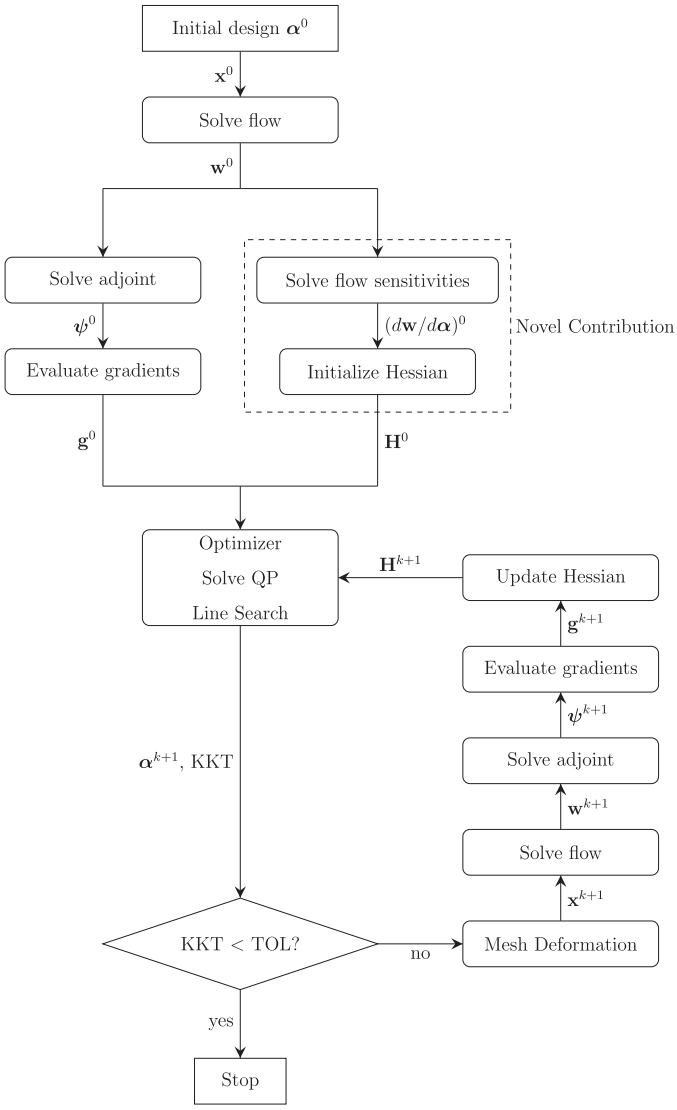

Fig. 1. Optimization framework flowchart.

2. First-order sensitivities

The gradient of an objective function can be computed from either direct differentiation or the adjoint method. Although the adjoint method is used to compute the gradient, the direct method will be required for the evaluation of the Hessian.

2.1. Flow sensitivities

The objective function $I = I(\mathbf{w}, \mathbf{x})$ is a function of the flow state variables $\mathbf{w} = \mathbf{w}(\mathbf{x})$ and the geometry $\mathbf{x} = \mathbf{x}(\boldsymbol{\alpha})$, which in turn is parametrized through the control points $\boldsymbol{\alpha}$. The number of design variables $\boldsymbol{\alpha}$ is defined by $N_{\boldsymbol{\alpha}}$. The state variables are implicitly defined through the steady-state solution of the Navier–Stokes equations, where the residual of a converged solution is zero.

$$\mathbf{R} = \mathbf{R}(\mathbf{w}, \mathbf{x}) = \mathbf{0}. \tag{1}$$

The gradient of the cost function and the flow residual with respect to control points are defined through the chain rule.

$$\frac{dI(\mathbf{w}, \mathbf{x})}{d\boldsymbol{\alpha}} = \mathbf{w}\frac{d\mathbf{w}}{d\boldsymbol{\alpha}} + \mathbf{x}\frac{d\mathbf{x}}{d\boldsymbol{\alpha}}, \tag{2}$$

$$\frac{d\mathbf{R}(\mathbf{w}, \mathbf{x})}{d\boldsymbol{\alpha}} = \mathbf{w}\frac{d\mathbf{w}}{d\boldsymbol{\alpha}} + \mathbf{x}\frac{d\mathbf{x}}{d\boldsymbol{\alpha}} = \mathbf{0}. \tag{3}$$

2.2. Direct differentiation

The derivative of the state variables with respect to the design variables $d\mathbf{w}/d\boldsymbol{\alpha}$ can be solved by using Eq. (3). The linear system is then solved for $N_{\boldsymbol{\alpha}}$ right-hand sides.

$$\frac{\partial \mathbf{R}}{\partial \mathbf{w}}\frac{d\mathbf{w}}{d\boldsymbol{\alpha}} = -\frac{\partial \mathbf{R}}{\partial \mathbf{x}}\frac{d\mathbf{x}}{d\boldsymbol{\alpha}}. \tag{4}$$

Unfortunately, the number of required solutions increases linearly with the number of design variables. As a result, the ASO community has been using the adjoint method to evaluate the functional derivatives.

An alternative to solving the linear system is to use the finite-difference method. This method is attractive when the above linear system is stiff and the flow solver converges well. Furthermore, frameworks that solve the flow and the adjoint explicitly will not need to implement the implicit linear system solver. Since the finite-difference perturbations are extremely small, the converged flow solution is a very good initialization for the slightly perturbed design and should converge quickly.