Large Scale Nonlinear Optimization, pp. 1-000
Editors
©2005 Kluwer Academic Publishers

Projected Hessians for Preconditioning in One-Step One-Shot Design Optimization[1]

Andreas Griewank (griewank@math.hu-berlin.de)
Institut für Mathematik Fakultät Math.-Nat. II
Humboldt-Universität Rudower Chaussee 25, Adlershof
Post: Unter den Linden 6, 10099 Berlin

Abstract

One-shot optimization aims at attaining feasibility and optimality simultaneously, especially on problems where even the linearized constraint equations cannot be resolved economically. Here we consider a scenario where forming and factoring the active Jacobian is out of the question, as is for example the case when the constraints represent some discretization of the Navier Stokes equation. Assuming that the 'user' provides us with a linearly converging solver that gradually restores feasibility after each change in the design variables, we derive a corresponding adjoint iteration and attach an optimization (sub)step.

The key question addressed is how the approximate reduced gradient generated by the adjoint iteration should be preconditioned in order to achieve overall convergence at a reasonable speed. An eigenvalue analysis yields necessary conditions on the preconditioning matrix, which are typically not satisfied by the familiar reduced Hessian. Some other projection of the Lagrangian Hessian appears more promising and is found to work very satisfactorily on a nonlinear test problem.

The analyzed approach is *one-step* in that the normal, dual and design variables are always updated simultaneously on the basis of one function evaluation and its adjoint. Multi-step variants are promising but remain to be investigated.

Keywords: Jacobian-free optimization, Adjoint equation, Algorithmic differentiation, R-linear Convergence, Fixed Point Iteration

[1]Supported by the DFG Research Center MATHEON
"Mathematics for Key Technologies" in Berlin

1 Introduction

Design optimization problems are distinguished from general NLPs by the fact that the set of variables is split a priori into a state vector $y \in Y$ and a set of design or control variables $u \in U$. For simplicity we will assume that not only Y but also U and thus their Cartesian product $X = Y \times U$ are Hilbert spaces. With $c : X \to Y$ the state residual and $f : X \to \mathbb{R}$ the objective we obtain the equality constrained optimization problem

$$\mathbf{Min} f(y, u) \quad \text{s.t.} \quad c(y, u) = 0 \in Y$$

While the solution space Y has often a natural scalar product, the design space U may be rather heterogeneous, representing for example material properties and/or shapes in various parameterizations [14]. We will therefore strive to avoid any dependences on the inner product in U.

For notational convenience we will assume that Y and U have finite dimensions $n = \dim(Y)$ and $m = dim(U)$ with $n \gg m$, typically.. Then elements $y \in Y$ and $u \in U$ may be identified with their coordinate vectors in $\mathbb{R}^n$ and $\mathbb{R}^m$ with respect to suitable Hilbert bases. This convention allows us to write duals as transposed vectors and inner products as the usual scalar products in Euclidean space. Also we will be able to compute determinants and characteristic polynomials to determine eigenvalues rather than performing spectral theory in Hilbert spaces.

A key assumption throughout the paper is that at all arguments $x = (y, u)$ of interest

$$\det(c_y(y, u)) \neq 0 \quad \text{where} \quad c_y \equiv \partial c(y, u) / \partial y \tag{1}$$

Here and elsewhere subscripting by real vectors indicates partial differentiation, whereas integer subscripts are iteration counters. Throughout we will denote a locally uniquee solution of $c(y, u) = 0$ for fixed u as $y_* = y_*(u)$ and also annotate other quantities evaluated at such a feasible point by subscript $*$.

Related Concepts

The scenario sketched above is similar to the one described by Biros and Ghattas in [2] under the heading 'PDE Constrained Optimization'. They also consider a situation, where there are 'only' equality constraints, which makes the problem look simple from a superficial point of view. However, the key assumption is that the state constraint system is so complex that it is impossible to directly evaluate and manipulate the active constraint Jacobian in any way. In particular it is considered very troublesome to compute and represent the generally dense matrix of tangential directions because that requires matrix factorizations or a large number of iterative equation solves. Instead one wishes to get by with just a few approximate solves for the state equation and its adjoint per outer iteration.

In another excellent paper [13] Heinkenschloss and Vincente develop a framework, which is even notationally very close to the one used here. They rightly emphasize the importance of inner products and the corresponding norms for the definition of gradients and second derivatives as well as iterative equation solvers of the Krylov and quasi-Newton type. These aspects may sometimes be ignored with impunity on smaller problems but become absolutely crucial on large scale NLPs that arise form the discretization of design or control problems involving differential equations. Ideally, the overall optimization algorithm should be set up such that it behaves mesh-invariant in that it generates approximately the same iterates and function values for all sufficiently fine meshes [5].

Complexity Growth

Especially when partial differential equations are involved, the process of simulation, i.e. the computation of a feasible state y given values of the design or control parameters u may take a substantial amount of computational resources. It is then very important that the transition to optimization causes neither the operations count nor the number of data transfers and total memory required to grow by more than a moderate factor. Otherwise one has to perform the optimization calculation on cruder models, or rely on shear intuition altogether. Ideally, the factor limiting the growth in complexity should be a problem independent constant. However, even theoretically, one has to allow for a logarithmic dependence on the problem size in order to limit the storage requirement [7] for gradient calculations on evolutionary, or instationary, problems. Even though for actual runtimes the characteristics of the computing platform and many other issues are important, we will keep in mind that the complexity of each outer optimization step as well as their total number required to achieve an acceptable solution accuracy does not exceed the corresponding costs for the underlying simulation by more than a moderate factor.

2 (R)SQP variants on the structured KKT System

Whenever the regularity assumption (1) is satisfied in the vicinity of the feasible set $c^{-1}(0) = \{x \in X : c(x) = 0\}$, the subspace $Y \equiv Y \times \{0\} \subset X$ can be used as a subspace of normal directions. The reason is that it must then be transversal to the nullspace spanned by the row vectors of the matrix

$$[Z^T, I_m] \quad \text{where} \quad Z \;=\; Z(y,u) \;\equiv -c_y(y,u)^{-1} c_u(y,u)$$

Here I_m denotes the identity matrix of order m. We may therefore use the adjective *normal* to qualify all terms and actions that are concerned with solving the state space equations without changing the design vector u. In contrast we will use the adjective *adjoint* for all terms and actions related to adjoint equation associated with

$c(y,u) = 0$, namely

$$0 \;=\; \bar{c}(y,\bar{y},u) \;\equiv\; \bar{y}\, c_y(y,u) + \mu\, f_y(y,u) \quad \text{with} \quad \bar{y}^T \in \mathbb{R}^n \tag{2}$$

Here $\mu > 0$ is a constant weighting of the objective function, which will be handy later on as relative scale of normal and adjoint errors. It is well known that the solution vector $\bar{y}$ may be interpreted as the gradient of the weighted objective function value $\mu\, f$ with respect to perturbations of the right hand side of the state equation $c(y,u) = 0$. We will view all barred quantities and all partial derivatives as row vectors, since they really belong to the dual spaces of Y or U. At feasible points $(y(u),u) \in c^{-1}(0)$ the total derivative, or reduced gradient, of f with respect to u is given by the adjoint vector

$$\bar{u}(y,\bar{y},u) \;\equiv\; \bar{y}\, c_u(y,u) + \mu\, f_u(y,u) \quad \text{with} \quad \bar{u}^T \in \mathbb{R}^m$$

We note that by their definition the adjoint vectors $\bar{y}$ and $\bar{u}$ like all Fréchet derivatives are heavily dependent on the inner products in Y and U. We will assume that for Y the inner product has been given in a 'natural' way and strive to eliminate the influence of the inner product in U, which we view as more or less arbitrary. In other words we are looking for invariance with respect to linear transformations on the design space U. Whenever $\bar{c}(y,\bar{y},u) = 0$ we will speak of *adjoint feasibility*. It should be noted that due to the given variable partitioning this is not quite the same as the concept of *dual feasible* familiar from linear optimization, in particular.

Adding the optimality condition $\bar{u} = 0$ to the state equation $c(y,0) = 0$ and its adjoint (2) we get the coupled system

$$\begin{bmatrix} c(y,u) \\ \bar{c}(y,\bar{y},u) \\ \bar{u}(y,\bar{y},u) \end{bmatrix} \;=\; 0 \tag{3}$$

This is nothing but the KKT system for an equality constrained optimization problem with the a priori partitioned variable vector $x = (y,u)$. The notational association of the Lagrange Multiplier vector $\bar{y}$ with the state variables y is meaningful because (2) has the exact solution $\bar{y} = \mu\, f_y(y,u)\; c_y(y,u)^{-1}$. In other words $\bar{c}(y,\bar{y},u) = 0$ defines the Lagrange multiplier estimates such that the partial gradient of the Lagrangian

$$L(y,\bar{y},u) \;\equiv\; \mu\, f(y,u) + \bar{y}\, c(y,u) \tag{4}$$

with respect to the state vector y is always zero. Computationally this makes especially sense when the state space Jacobian $c_y(y,u)$ is not only nonsingular but also has sufficient structure to facilitate the economical computation of Newton steps $\Delta y = -c_y(y,u)^{-1} c(y,u)$. Whether this is done by factorization or using iterative solvers, the exact or approximate computation of $\bar{y}$ by the same scheme applied to the transposed system will then only require a similar amount of computing time and

very little extra coding. This applies in particular when $c(y,u)$ is selfadjoint with respect to y, i.e. the partial Jacobian c_y is symmetric. However, in view of nonelliptic problems we will later in this paper be mainly concerned with the case, where solving even the linearized state or adjoint equation for fixed u with any degree of accuracy is already very expensive.

Inexact Subspace Iteration

The structure of the KKT system (3) with the state equation Jacobian c_y being nonsingular suggests a block solution approach consisting of three substeps in each major iteration or cycle. First compute a normal step

$$y \;=\; y + \Delta y \quad \text{with} \quad \Delta y = -c_y(y,u)^{-1} c(y,u) \tag{5}$$

towards primal feasibility (Here a+= b denotes the incremental assignment $a \leftarrow a+b$). Then, at the new y point evaluate

$$\bar{y} \;=\; \mu\, f_y(y,u)\, c_y(y,u)^{-1} \quad \text{and} \quad \bar{u} = \bar{y}\, c_u(y,u) + \mu\, f_u(y,u). \tag{6}$$

Finally, given a symmetric positive matrix $H \in \mathbb{R}^{n\times n}$ perform an optimization step

$$u \;=\; u + \Delta u \quad \text{with} \quad \Delta u \equiv\; -H^{-1}\bar{u}^T \tag{7}$$

In other words we perform a cycle of three steps on the primal state, adjoint state, and design vector, to enhance feasibility, adjoint feasibility, and optimality, respectively.

On nonlinear problems one needs conceptually infinitely many such cycles, and it does not matter for the asymptotic behavior with which of the three steps one begins. Hence we may characterize this approach schematically as

$$\cdots \rightarrow \textbf{normal} \rightarrow \textbf{adjoint} \rightarrow \textbf{optim} \rightarrow \cdots$$

An (R)SQP interpretation

When the Lagrange multipliers are really recomputed from scratch as suggested by (6) one may merge the second step with the third and obtains a scheme similar to the Reduced SQP Algorithm 1 detailed in [12] by Hazra and Schulz. The only difference is that they avoid the evaluation of the constraints at two different arguments in each cycle by replacing the right hand side for the computation of Δy with the approximation $c(y,\tilde{u}) + c_u(y,\tilde{u})(u-\tilde{u}) \;\approx\; c(y,u)$. Here $\tilde{u} = u - \Delta u$ denotes the old design at which c and its derivatives were already evaluated in the previous cycle. Either variant yields locally 2-step, or more precisely in our context 2-cycle Q-superlinear convergence, if the preconditioner H equals at least asymptotically the reduced Hessian

$$H(1) \;=\; L_{uu} + Z^T L_{yu} + L_{uy} Z + Z^T L_{yy} Z \tag{8}$$

Throughout the paper we will use this notation and refer to $H(1)$ as the *reduced* rather than the projected Hessian, since there will be other projections of interest.

R-superlinear convergence has also been established for more classical RSQP, i.e. reduced SQP, methods where the last step is properly tangential. One should notice that this is not the case for our optimization step (7), which leaves the state variables uncorrected and relies on the subsequent normal step (5) to straighten them out. Thus we obtain in some way a method of alternating direction, which like other nonlinear block Gauss-Jordan variants is generally prone to yield only linear convergence. Proper RSQP methods do not need to apply the Schulz correction mentioned above and can evaluate all function and derivative evaluations at a single argument per cycle. There is, however, a variant due to Coleman and Conn [3] that expends a second residual evaluation after the normal step in order to achieve 1-cyle Q-superlinear convergence. In contrast we require a second residual evaluation after the optimization step or need to approximate its value by the Schulz correction. This still does not give us 1-cycle Q-superlinear convergence unless we perform two normal corrections in a row thus applying the scheme

$$\cdots \rightarrow \textbf{normal} \rightarrow \textbf{adjoint} \rightarrow \textbf{optim} \rightarrow \textbf{normal} \rightarrow \cdots$$

For other RSQP variants and a very detailed convergence analysis see the seminal paper by Biegler, Nocedal and Schmid [1].

3 Pseudo-Newton Solvers and Piggy-backing

The discussion above tacitly assumed that it is possible to approximate normal and adjoint Newtons steps with some degree of accuracy at a reasonable cost. Especially on discretizations of nonelliptic PDEs this assumption is rather optimistic and one may have to make do with a possibly rather slowly convergent fixed point iteration of the form

$$y_{k+1} \;=\; G(y_k, u) \quad \text{with} \quad G : X = Y \times U \rightarrow Y \tag{9}$$

Naturally, we assume that the *iteration function* G is stationary only at feasible points, i.e.

$$y = G(y, u) \quad \Longleftrightarrow \quad c(y, u) = 0$$

This holds for example when $c(y, u) = P(y, u)[G(y, u) - y]$ with $P(x) = I_m$ or some other family of nonsingular matrices, e.g. $P(y, u) = c_y(y, u)$ when G is defined by Newton's methods.

Linear Convergence Rates

In aerodynamics it is not unusual that thousands of iterations are needed to obtain a high accuracy solution of the state equation. We will assume throughout that G is

contractive such that for some constant ρ in the induced matrix norm $\|\cdot\|$

$$\|G_y(y,u)\| \;\leq\; \rho < 1 \implies \|G(y,u) - G(z,u)\| \leq \rho \, \|y - z\| \tag{10}$$

The implication follows from a mean value theorem on any convex subdomain of Y. Another consequence is that no eigenvalue of $G_y(y,u)$ can be in modulus greater than ρ at any point (y,u).

Now it follows from Banach's fixed point theorem that the normal iteration (9) converges for any given u to the unique fixed point $y_* = y_*(u)$ solving $c(y(u),u) = 0$. Moreover, the rate of convergence is linear with the Q-factor

$$\rho_0 \;\equiv\; Q\big\{y_k - y_*\big\}_{k\in\mathbb{N}} \;\equiv\; \limsup_k \, \|y_{k+1} - y_*\| / \, \|y_k - y_*\| \;\leq\; \rho \; .$$

The norm independent R-factor may be smaller [15] and is given by

$$\rho_* \;\equiv\; R\big\{y_k - y_*\big\}_{k\in\mathbb{N}} \;\equiv\; \limsup_k \sqrt[k]{\|y_k - y_*\|} \;\leq\; \rho_0 \;\leq\; \rho$$

When even ρ_* is quite close to 1.0 it makes little sense to refer to (9) as a Newton-like iteration. In his thesis [4] Courty has considered such slowly convergent solvers for Navier Stokes variants and called them more appropriately *pseudo-Newton* iterations.

Our key assumption is that the 'user' provides (only) such a pseudo-Newton solver, which may have been developed over a long period of time by various people. Presumably, they were experts at exploiting the particular structure of the state equation, so one cannot easily accelerated or otherwise improve the given fixed point solver. Now the optimizer faces the task of converting this given simulation tool into an optimization tool without performing major surgery on the code. We have coined the term *piggy-backing* for this hopefully convenient transition.

The Adjoint Iteration

Assuming that the 'user' also provides a code for the objective function $f(y,u)$ one can apply automatic, or algorithmic differentiation in the reverse mode to evaluate the associated adjoints

$$\bar{y}_{k+1} \quad = \quad \bar{G}(y_k, \bar{y}_k, u) \;\equiv\; \bar{y}_k \; G_y(y_k,u) + \mu \, f_y(y_k,u) \tag{11}$$

and

$$\bar{u}_{k+1} \quad \equiv \quad \bar{y}_k \; G_u(y_k,u) + \mu f_u(y_k,u) \tag{12}$$

It is in the nature of the reverse mode that both $\bar{y}_{k+1}$ and $\bar{u}_{k+1}$ are obtained simultaneously and with a basic operations count not exceeding 4 times that for evaluating G and f. While this complexity result from [7] takes into account memory moves as operations, it does rely on the ability to temporarily store the intermediate results

of all arithmetic operations and special function evaluations. Alternatively, one may employ checkpointing techniques, which allow various trade-offs between the increase in the temporal and the spatial complexity for the adjoint (11) relative to the normal (9) iteration. In particular one can achieve an only logarithmic growth in both complexity components. For more details see again [7] and [8].

In the current context it is only important to note that performing the adjoint step (11,12) is not much more expensive than computing the normal step (9) and that, except in linear cases, the former cannot be executed without the latter. As a cautionary note we add that the mechanical application of an AD tool in reverse mode to l successive steps (9) will lead to a memory requirement that grows proportional to l. This memory growth in proportion to the number of iterations can and should be avoided whenever there is convergence to a stationary solution, in other words if we have a pseudo-time evolution.

Notice that in (11) the Lagrange multipliers $\bar{y}_k$ have a true iteration history, rather than being recalculated from scratch at each primal iterate. The latter is usually done in NLP solvers that have access to the active constraint Jacobian. Naturally, the resulting approximate reduced gradient $\bar{u}_{k+1}$ will be as good or bad as the current multiplier estimate $\bar{y}_k$. For $c(y,u) = P(y,u)\,[G(y,u) - y]$ we have at feasible points (y,u) the identity $c_y(y,u) \;=\; P(y,u)\,[G_y(y,u) - I_n]$. Then it is easy to see that $\bar{y}_* \;=\; \bar{y}_*(u)$ is exactly a fixed point of (11) at $(y,u) \in c^{-1}(0)$ if $\bar{y}_* P(y_*,u)^{-1}$ is a solution of the original adjoint equation (2).

It is well understood that if one first solves the state equation $c(y,u) = 0$ to full accuracy and only then executes the adjoint iteration (11) the convergence rate will be very similar. The reason is that the Jacobian

$$\partial \bar{G}(y,\bar{y},u)/\partial \bar{y} \;=\; G_y^T \;=\; [\partial G(y,u)/\partial y]^T$$

has of course exactly the same spectral properties as G_y. Since, for fixed (y,u), the adjoint $\bar{G}$ is linear with respect to $\bar{y}$ we may even expect a more regular convergence pattern for the adjoints than for the underlying normal iteration. It was first observed in [6] that if G describes Newton's method so that $G_y(y,u)$ vanishes at any feasible (y,u), then the adjoints will converge in a single step, which amounts simply to an application of the implicit function theorem. For slowly converging solvers on the other hand, it can often be observed that executing the adjoint iteration simultaneously with the underlying state space iteration does not slow the former down significantly. This simultaneous iteration certainly promises a significant reduction in wall-clock time, especially on a multiprocessor machine.

The Coupled Iteration

For the subsequent analysis we write the combination of (9) and (11) in a Hamiltonian like fashion as

$$\left[y_{k+1}^T, \bar{y}_{k+1}\right] \;=\; \nabla N(y_k,\bar{y}_k,u) \;=\; [N_{\bar{y}}(y_k,\bar{y}_k,u), N_y(y_k,\bar{y}_k,u)] \tag{13}$$

where N is defined as

$$N(y, \bar{y}, u) \;\equiv\; \bar{y}\, G(y, u) + \mu\, f(y, u) \tag{14}$$

When we set $c(y, u) = G(y, u) - y$ then the function N is related to the Lagrange function L as defined in (4) by the shift

$$\bar{y}\; y = N(y, \bar{y}, u) - L(y, \bar{y}, u)$$

Consequently, we have

$$N_y = L_y + \bar{y} \quad \text{and} \quad N_{\bar{y}} = L_{\bar{y}} + y \quad \text{but} \quad N_u = L_u$$

and, for the subsequent analysis more importantly, all second derivatives are identical:

$$N_{yy} = L_{yy},\; N_{yu} = L_{yu}, N_{uu} = L_{uu}$$

In the more general situation $c(y, u) = P(y, u)[G(y, u) - y]$ the second derivatives are related in such a way that second order necessary and sufficiency conditions applied to N and L are still equivalent.

In any case we note that N is linear in the weighting $(\bar{y}, \mu)$ and thus homogeneous in μ, provided $\bar{y}$ solves the adjoint equation $N_y(y, \bar{y}, u) = \bar{y}$ or is initialized and consistently updated proportionally to μ. Hence we can make N and its derivatives arbitrarily small compared to G by reducing the weighting μ on the objective appropriately.

A side benefit of iterating on $\bar{y}_k$ simultaneously with y_k is that the Lagrangian value

$$\mu\, f(y_k, u) + \bar{y}_{k+1}\,(y_{k+1} - y_k) \;=\; \mu\, f(y_*(u), u) + \mathcal{O}(\|\bar{y}_{k+1} - \bar{y}_*\| \|y_{k+1} - y_*\|) \tag{15}$$

converges much faster than $\mu\, f(y_k, u)$ towards the limit $\mu\, f(y_*(u), u)$. This result was derived in [9] using a very general duality argument.

Adjoint Convergence Analysis

The Jacobian of the coupled iteration (13) takes the block triangular form

$$J_k \;\equiv\; \frac{\partial(y_{k+1}, \bar{y}_{k+1})}{\partial(y_k, \bar{y}_k)} \;=\; \begin{bmatrix} G_y(y_k, u) & 0 \\ N_{yy}(y_k, \bar{y}_k, u) & G_y^T(y_k, u) \end{bmatrix}$$

By elementary arguments one derives that in the l-2 matrix norm

$$\|J_k\| \;\le\; \max\left(\|G_y\|, \|G_y^T\|\right) + \|N_{yy}\| \;=\; \rho + \mathcal{O}(\mu)$$

In other words for sufficiently small μ the combined iteration (13) is, like G, also a contraction on the product space $Y \times \bar{Y}$. Using the homogeneity of all adjoint quantities with respect to μ, one can then deduce that for any fixed μ also

$$\bar{\rho}_* \;\equiv\; R\big\{\bar{y}_k - \bar{y}_*\big\}_{k \in \mathbb{N}} \;\equiv\; \limsup_k \sqrt[k]{\|\bar{y}_k - \bar{y}_*\|} \;\le\; \rho$$

This R-linear convergence result was generalized in [9] to contractive iterations of the form $y_{k+1} = G_k(y_k, u) = y_k - P_k\, c(y_k, u)$ coupled with the corresponding adjoints. Here the matrices P_k may vary discontinuously from step to step as is for example the case for quasi-Newton methods.

On the other hand, the following observation was made in [11] for twice continuously differentiable $G(y, u)$ and $f(y, u)$. Since J_k is block triangular and has in its diagonal the Jacobian G_y and its transposed G_y^T, every eigenvalue of G_y is a double eigenvalue of J_k, which has thus generically Jordan blocks of size 2. As a consequence, for fixed u the adjoints $\bar{y}_k$ and $\bar{u}_k$ converge a little slower than the normal iterates y_k in that in general

$$\|\bar{u}_k - \bar{u}_*\| \;\sim\; \|\bar{y}_k - \bar{y}_*\| \;\sim\; k\;\|y_k - y_*\| \tag{16}$$

In other words, despite the common R-factor, the Lagrange multipliers $\bar{y}_k$ and the corresponding reduced gradients $\bar{u}_k$ lag somewhat behind the normal iterates y_k. In hindsight that is not all that surprising since the adjoints are heading for a moving target, namely the solution of an equation parameterized by y_k. A practical consequence seems that any optimization algorithms that is based on the approximate reduced gradients $\bar{u}_k$ will likely exhibit tangential convergence to constrained optima. In other words feasibility will be reached faster than adjoint feasibility and thus optimality, a pattern that is familiar from NLP solvers based on exact Jacobians but (secant) approximated Hessians.

Reduced Hessian as Second Order Adjoints

Differentiating (13) and (12) once more with respect to $x = (y, u)$ in some direction $(\dot{y}_k, \dot{u})$ one obtains an iteration of the form

$$\begin{bmatrix} \dot{y}_{k+1} \\ \dot{\bar{y}}_{k+1} \\ \dot{\bar{u}}_{k+1} \end{bmatrix} \;=\; \begin{bmatrix} G_y(y_k,u)\dot{y}_k + G_u(y_k,\bar{y}_k,u)\dot{u} \\ \dot{\bar{y}}_k G_y(y_k,u) + (N_{yy}(y_k,\bar{y}_k,u)\dot{y}_k + N_{yu}(y_k,\bar{y}_k,u)\dot{u})^T \\ \dot{\bar{y}}_k G_u(y_k,u) + (N_{uy}(y_k,\bar{y}_k,u)\dot{y}_k + N_{uu}(y_k,\bar{y}_k,u)\dot{u})^T \end{bmatrix} \tag{17}$$

Here we have separated on the right hand side the leading terms that are dependent on $\dot{y}_k \in \mathbb{R}^n$ and $\dot{\bar{y}}_k^T \in \mathbb{R}^n$, respectively. Since they are defined again in terms of the contractive state space Jacobian G_y one gets once more linear convergence as follows.

Proposition 1. *With $Z_* \equiv (I - G_y)^{-1} G_u \in \mathbb{R}^{m\times n}$ the derivatives $\dot{y}_k \in \mathbb{R}^n, \dot{\bar{y}}_k^T \in \mathbb{R}^n$ and $\dot{\bar{u}}_k^T \in \mathbb{R}^m$ converge with R-Factors $\leq \rho$ such that in the limit*

$$\lim_k \begin{bmatrix} \dot{y}_k \\ \dot{\bar{y}}_k \\ \dot{\bar{u}}_k \end{bmatrix} \;=\; \begin{bmatrix} Z_*\dot{u} \\ \dot{u}^T[Z_*^T, I]N_{xy}(I - G_y)^{-1} \\ \dot{u}^T[Z_*^T, I]N_{xx}[Z_*^T, I]^T \end{bmatrix}$$

Proof. The fact of linear convergence with R-factor $\leq \rho$ was proven in [9]. The limits can be derived in the current framework as follows. Simply replacing in (17)

all subscripts k by $*$ and collecting the unknown vector onto the left side we obtain linear equations for $\dot{y}_*, \dot{\bar{y}}_*$ and $\dot{\bar{u}}_*$, all in terms of the matrix $I - G_y$ or its transpose. The first equation has the right hand side G_u, which immediately yields the solution $(I - G_y)^{-1} G_u \dot{u} = Z_* \dot{u}$. Substituting this result into the second equation we obtain the transposed right hand side

$$N_{yx} \dot{x}_* = N_{yy} \dot{y}_* + N_{yu} \dot{u}_* = (N_{yy} Z_* + N_{yu}) \dot{u}_* = N_{yx} \begin{bmatrix} Z_* \\ I \end{bmatrix} \dot{u}$$

Transposition and post-multiplication by $(I - G_y)^{-1}$ yields the assertion. Substitution of this result into the third yields firstly the term $\bar{y}_* G_u = \dot{u}^T [Z_*^T, I] N_{xy} (I - G_y)^{-1} G_u = \dot{u}^T [Z_*^T, I] N_{xy} Z_*$ to which we have to add $\dot{x}_*^T N_{xu} = \dot{u}_*^T (Z_*^T, I) N_{xu}$. The sum can than be written as the asserted partitioned matrix product. ∎

The proposition states that for each selected direction $\dot{u}$ in the domain space, the first derivative vectors $\dot{y}_k$ converge to the state space direction $\dot{y}_* = Z_* \dot{u}$, which makes $\dot{x} = (\dot{y}_*, \dot{u})$ a feasible direction in the full space $X = Y \times U$. The corresponding adjoint vectors $\dot{\bar{y}}_k$ converge to the product of the Lagrangian Hessian and the vector $\dot{x} = (\dot{y}_*, \dot{u})$, except for the nonsingular transformation $I - G_y$. The corresponding second order adjoints $\dot{\bar{u}}_k$ converge to the transposed direction $\dot{u}^T$ multiplied by the reduced Hessian. As in Section 1 we will denote the latter by $H_*(1) = [Z_*^T, I] N_{xx} [Z_*^T, I]^T \in \mathbb{R}^{m \times m}$ for reasons that will become clear later.

While the iteration (17) looks reasonably complicated it can also be realized in a completely automatic fashion, which does not increase the complexity by more than a constant factor. If one replaces $\dot{u}$ in the above results by the identy I_m on the design space the whole reduced Hessian $H_*(1)$ is obtained as $\dot{\bar{u}}_*$ by what is sometimes called the *vector mode* of automatic differentiation. Naturally the cost will then also grow by a factor of m, namely the dimension of the design space. In our test calculations we have so far made that effort, though we prefer a different projected Hessian denoted by $H_*(-1)$ that can be obtained using a slight variation of (17).

The analysis in [11] shows that the errors in $\dot{\bar{y}}_k$ and thus also in $\dot{\bar{u}}_k$ lag asymptotically behind those in y_k by a factor proportional to k^2 so that

$$\|\dot{\bar{u}}_k - \dot{\bar{u}}_*\| \;\sim\; \|\dot{\bar{y}}_k - \dot{\bar{y}}_*\| \;\sim\; k^2 \;\|y_k - y_*\| \tag{18}$$

In view of (16) this effect is not surprising and probably not so serious, because the accuracy of reduced Hessians is not nearly as critical as that of reduced gradients for rapid and robust convergence.

Inexact Block Gauss-Seidel

Now suppose we decide to always execute the coupled step (13) $r > 0$ times, before executing one optimization step of the form (7). Schematically, we may write this approach as

$$\cdots \to [\mathbf{normal}, \mathbf{adjoint}]^r \to \mathbf{optim} \to \cdots \tag{19}$$

Here [**normal**, **adjoint**] represents (13) with $u_{k+1} = u_k$ remaining unchanged and **optim** represents the update

$$u_{k+1} \;=\; u_k - H_k^{-1}\,[\bar{y}_k\, G_u(y_k, u) + \mu\, f_u(y_k, u)]^T \tag{20}$$

with $y_{k+1} = y_k$ and $\bar{y}_{k+1} = \bar{y}_k$ remaining unchanged. (Thus k is employed to count individual steps rather than whole cycles.)

When we select r very large the first two equation blocks in the KKT systems are solved quite accurately, so that we have a block Gauss-Seidel type method. However, it is clearly doubtful whether it makes sense to expend a large computational effort for gaining feasibility and adjoint feasibility at a design argument u that may still be far from the optimal. This observation leads naturally to the concept of 'one-shot' optimization as it is called in the aerodynamics literature [16]. The basic idea is to adjust the design u as frequently as possible in order to gain optimality more or less simultaneously with feasibility.

From an 'algebraic' point of view it seems attractive to adjust u at every step. Here we tacitly assume that changing u from a current value u_k to a different design u_{k+1} causes no extra cost for the subsequent evaluation of the iteration function G at (y_{k+1}, u_{k+1}). However, in applications there may be a significant setup cost for example due to the need to regrid after changes in a wing design. In that situation our hypothesis that $\bar{u}_{k+1}$ costs little extra on top of the evaluation of $\bar{y}_{k+1}$ may also be a little naive. To model these effects one would have to consider some partial preevaluation of $G(y, u)$ and then distinguish the cost of evaluating G and its derivatives with or without a change in u. For simplicity we shy away from doing so, but will keep this effect in mind. On a theoretical level one may refine G as a subcyles of l normal steps, such that its combined cost clearly dominates the setup cost incurred by changing the design. In any case we may realistically assume that $m = \dim(u)$ is so small that the linear algebra associated with the optimization step especially storing and manipulating the preconditioning matrix H is comparatively cheap.

Finally, as we have observed in the previous section for an RSQP-like method it may make sense to perform more normal steps than coupled ones so that one arrives at a the following most general scheme

$$\cdots \to [\mathbf{normal}]^s \to [\mathbf{normal}, \mathbf{adjoint}]^r \to \mathbf{optim} \to \cdots \tag{21}$$

Here **normal** represents (9) with $\bar{y}_{k+1} = \bar{y}_k$ and $u_{k+1} = u_k$ remaining unchanged.

Now the key question is naturally how to first select the repetition numbers s and r and then how to best define and compute the matrix H_k in (20). Since there is no separate line-search it should absorb the step multiplier and achieve the invariance with respect to linear transformations on the design space. Naturally, when $\rho^{(s+r)}$ is still close to 1.0 we must choose H_k large and thus H_k^{-1} small such that the design variables u_k are adjusted rather cautiously. Otherwise the iteration may diverge immediately as was we have observed on our test problem. On the other hand we

wish to choose H_k small enough such that the overall optimization proceeds at a reasonable pace that is not much smaller than normal iteration for pure simulation at a fixed design u. While there are some experimental studies and corresponding recommendations for particular application areas we are not aware of a systematic study on the best choice of the repetition numbers s, r and the possible choices of H_k. There is a natural tendency for optimization people to assume that the reduced Hessian $H_k = H_*$ as defined in (8) is the optimal choice. However, we believe this only to be true when ρ^r is quite small so that a large degree of normal and adjoint feasibility is restored before each optimization step. It certainly is not valid at the other extreme namely for the one-step method

$$\cdots \to [\mathbf{normal}, \mathbf{adjoint}, \mathbf{optim}] \to \cdots \tag{22}$$

In contrast to the scheme above with $s = 0$ and $r = 1$ we assume here that the reduced gradient (12) is still evaluated with the old values $(y_k, \bar{y}_k)$ rather than the new versions $(y_{k+1}, \bar{y}_{k+1})$ just obtained in the same cycle. As we have discussed above, from an algebraic view point this value $\bar{y}_k \; G_u(y_k, u_k) + f_u(y_k, u_k)$ is a cheap by-product of evaluating $[G, f]$ and $\bar{G}$ at $(y_k, \bar{y}_k, u_k)$, but obtaining the newer reduced gradient $\bar{y}_{k+1} \; G_u(y_{k+1}, u_k) + f_u(y_k, u_k)$ would require a completely new evaluation of the same quantities at $(y_{k+1}, \bar{y}_{k+1}, u_k)$. While this simplification may cause the loss of of superlinear convergence for an RSQP like method, we believe that it makes very little difference for the convergence speed in a pseudo-Newton setting.

4 Necessary Convergence Condition on H_*

Throughout this section we assume that we are within the neighborhood of a KKT point $(y_*, \bar{y}_*, u_*)$, i.e. a solution to (3) in which the contractivity assumption (10) holds and G as well as f are twice Lipschitz continuously differentiable. The contractivity implies linear independence of the constraints $G(y, u) - y = 0$ and thus uniqueness of the multipliers $\bar{y}_*$ for given (y_*, u_*). Using the function $N(y, \bar{y}, u)$ defined above we may write one step of the one-step scheme (22) as

$$\begin{bmatrix} y_{k+1} \\ \bar{y}_{k+1} \\ u_{k+1} \end{bmatrix} = \begin{bmatrix} G(y_k, u_k) \\ N_y(y_k, \bar{y}_k, u_k)^T \\ u_k - H_k^{-1}[N_u(y_k, \bar{y}_k, u_k)]^T \end{bmatrix} \tag{23}$$

Omitting the common argument $(y_*, \bar{y}_*, u_*)$ and assuming that the H_k converge to a limit H_* we obtain at the fixed point the block 3×3 Jacobian

$$\begin{aligned} \hat{J}_* &= \left. \frac{\partial(y_{k+1}, \bar{y}_{k+1}, u_{k+1})}{\partial(y_k, \bar{y_k}, u_k)} \right|_{(y_*, \bar{y}_*, u_*)} \\ &= \begin{bmatrix} G_y & 0 & G_u \\ N_{yy} & G_y^T & N_{yu} \\ -H_*^{-1} N_{uy} & -H_*^{-1} G_u^T & I - H_*^{-1} N_{uu} \end{bmatrix} \end{aligned} \tag{24}$$

The eigenvalues of the system Jacobian can be characterized as those of its Schur complement.

Proposition 2. *For any nonsingular $H_* = H_*^T$ the eigenvalues of the matrix $\hat{J}_*$ are either eigenvalues of G_y or solve the nonlinear eigenvalue problem* $\det(M(\lambda)) = 0$ *with*

$$M(\lambda) \;\equiv\; (\lambda - 1)H_* + [-G_u^T(G_y^T - \lambda I)^{-1}, I]N_{xx} \begin{bmatrix} -(G_y - \lambda I)^{-1}G_u \\ I \end{bmatrix} \tag{25}$$

where $N_{xx} = \nabla^2_{(y,u)}N \in \mathbb{R}^{(n+m)\times(n+m)}$.

Proof. The eigenvalues λ must solve the characteristic equation

$$0 = \det \begin{bmatrix} G_y - \lambda I & 0 & G_u \\ N_{yy} & G_y^T - \lambda I & N_{yu} \\ -H_*^{-1}N_{uy} & -H_*^{-1}G_u^T & (1-\lambda)I - H_*^{-1}N_{uu} \end{bmatrix} \tag{26}$$

Without changing the roots we can premultiply the last block row by the the negative of the nonsingular matrix $-H_*$, which now occurs in place of I. Unless a λ is eigenvalue of G_y the leading 2×2 block has the inverse

$$\begin{bmatrix} (G_y - \lambda I)^{-1} & 0 \\ -(G_y - \lambda I)^{-T}N_{yy}\,(G_y - \lambda I)^{-1} & (G_y - \lambda I)^{-T} \end{bmatrix} \tag{27}$$

Using this block to eliminate the last block row and block column we obtain after some rearrangements as Schur complement exactly the negative of matrix $M(\lambda)$. ∎

Now we look for conditions that exclude the existence of eigenvalues with modulus greater than or equal to 1. To this end we denote the right term in $M(\lambda)$ by

$$H(\lambda) \;\equiv\; [Z(\lambda)^T, I]\; N_{xx}\; [Z(\lambda)^T, I]^T$$

where

$$Z(\lambda) \;=\; [\lambda I - G_y]^{-1}G_u \in \mathbb{R}^{n\times m}$$

Since we may write the state constraints as $G(y,u) - y = 0$ the columns of the particular matrix $Z(1)$ represent feasible directions restricted to the state space. Therefore the positive definiteness property of $H(1)$ being positive definite, denoted by $H(1) \succ 0$, corresponds to the usual second order sufficiency condition at constrained minima. More generally the rows of $[Z(\lambda)^T, I]$ span the tangent space of the manifold $\{G(y,u) = \lambda\, y\}$ and $H(\lambda)$ represents the projection of N_{xx} onto that m-dimensional subspace.

The nicest situation occurs when G_y has n distinct real eigenvalues and our optimization problem is strongly convex in that the Lagrangian Hessian N_{xx} is positive definite on the whole space $X = \mathbb{R}^{n+m}$. Then the rational function $\det M(\lambda)$ has n

double-poles in the interval $(-1, 1)$ and for positive definite H_* we may expect two eigenvalues of $\hat{J}_*$ close to each one of them. Of those only the ones beyond the largest and below the smallest eigenvalue of G_y may destroy contractivity. However, there is no guarantee that there may not be complex eigenvalues of modulus greater than 1.0 even in this convenient scenario. Therefore the necessary conditions for convergence derived in the following proposition are probably quite far from sufficient.

Proposition 3. *Provided $H(1) \succeq 0$, it is a necessary condition for $\hat{J}_*$ to have no real eigenvalues $\lambda \geq 1$ that $H_* \succ 0$ and $\det(H(1)) \neq 0$. Moreover, then for $\hat{J}_*$ also to have no real eigenvalues $\lambda \leq -1$ it is necessary that $H_* \succeq \frac{1}{2}H(-1)$.*

Proof. Since $M(1) = H(1)$ the singularity of $H(1)$ would immediately imply that $\lambda = 1$ is an eigenvalue of $\hat{J}_*$, which establishes $\det(H(1)) \neq 0$. As clearly $Z(\lambda) = \mathcal{O}(|\lambda|^{-1})$ we have $H(\lambda) = N_{uu} + \mathcal{O}(|\lambda|^{-1})$ and the leading term $(\lambda - 1)H_*$ of $M(\lambda)$ as defined in (25) dominates $M(\lambda)$ for large $|\lambda|$. Now if H_* had a negative eigenvalue the matrix $M(\lambda)$, which reduces to $H(1) = M(1)$ at $\lambda = 1$, would have to become singular for some $\lambda \geq 1$. Because H_* is by definition symmetric and nonsingular we must therefore have $H_* \succ 0$. Now suppose the second condition was violated so that $M(-1) = -2H_* + H(-1)$ has a positive eigenvalue. Then there would have to be a real eigenvalue $\lambda < -1$ since $(\lambda - 1)H_*$ and thus $M(\lambda)$ must be negative definite for large negative λ. ∎

According to Proposition 2 it is necessary for the iteration (23) to be contractive that

$$0 \prec 2\,H_* \succ H(-1) \;=\; [G_u^T(I + G_y)^{-T}, I]N_{xx}[(I + G_y)^{-1}G_u, I]^T$$

provided the second order sufficiency conditions are satisfied at the KKT point in question. If the underlying optimization problem is strongly convex then $H(-1)$ is like all $H(\lambda)$ positive definite and it looks like a reasonable candidate for the preconditioner H_*. On the other hand there is no reason, why the standard projected Hessian $H(1)$ should be greater than $H(-1)/2$ and if it fails to do so then using it in (23) must prevent even local convergence. On our test problem the standard choice $H_* = H_*(1)$ leads to immediate blow up of the iteration, while $H_* = H_*(-1)$ yields full step convergence without any line-search. While the problem is not everywhere convex we found that eventually all components of the Lagrange multiplier vector $\bar{y}_k$ were positive so that naturally both $H_* = H_*(1)$ and $H_* = H_*(-1)$ allowed a Cholesky factorization.

To make the surprising result of our eigenvalue analysis a little more intuitive we offer the following, admittedly very vague explanation. The matrix $Z(1) = (I - G_y)^{-1}G_u$ spans the tangent space of $\{G(y, u) = y\}$, which is rich in the eigenvectors of G_y for which the corresponding eigenvalues are close to $+1.0$. We may call them mononotic modes in contrast to the alternating modes associated with eigenvalues close to -1.0 The latter are richly represented in $Z(-1) = (I + G_y)^{-1}G_u$ whose columns span the tangent space of $\{G(y, u) = -y\}$. By preconditioning the

approximate reduced gradient with $H(-1) = Z(-1)^T N_{xx} Z(-1)$ we seem to be monitoring more the effects of these alternating modes. Their strength is perhaps more detrimental to a regular convergence behavior than that of the monotonic modes.

To actually compute approximations $Z_k \approx Z(-1) \in \mathbb{R}^{n\times m}$ and $H_k \approx H_*(-1) \in \mathbb{R}^{m\times m}$ one my use the following modification of the vector iteration (17).

$$\begin{bmatrix} -Z_{k+1} \\ -\bar{Z}_{k+1} \\ H_{k+1} \end{bmatrix} = \begin{bmatrix} G_y(y_k,u)Z_k + G_u(y_k,\bar{y}_k,u) \\ \bar{Z}_k G_y(y_k,u) + (N_{yy}(y_k,\bar{y}_k,u)Z_k + N_{yu}(y_k,\bar{y}_k,u))^T \\ \bar{Z}_k G_u(y_k,u) + (N_{uy}(y_k,\bar{y}_k,u)Z_k + N_{uu}(y_k,\bar{y}_k,u))^T \end{bmatrix} \tag{28}$$

Here the matrix $\bar{Z}_k \in \mathbb{R}^{m\times n}$ represents the adjoint of Z_k. The two minus signs in front of Z_{k+1} and $\bar{Z}_{k+1}$ on the left are quite deliberate and result from setting $\lambda = -1$. Of course we can move them to the right, especially in order to verify that we have again linear contractions in the state space Jacobian. Leaving them on the left only illustrates that we can use standard differentiation procedures to evaluate the right hand sides and then only have to flip signs to get the correct values for Z_{k+1} and $\bar{Z}_{k+1}$.

Finally, let us consider what happens when the design variable u is replaced by $C\,u$ with $C \in \mathbb{R}^{m\times m}$ and $\det(C) \neq 0$. Then all differentiations with respect to the new design variable will 'throw out' the extra factor C according to the chainrule. In particular the reduced gradient $\bar{u}$ and the matrices $Z(\lambda)$ will be multiplied from the right by C, and $H(\lambda)$ will be multiplied from the right by C and from the left by C^T. The same will be true for any preconditioner H_* that we define as a linear combination of one or several of the $H(\lambda)$. Then it can be checked without too much trouble that the corrections to the design variables according to (20) are C^{-1} times those for the original setting and thus unchanged except for the coordinate transformation. Hence we have indeed achieved invariance with respect to the inner product norm and the design space U.

5 Numerical Verification

The following results were obtained on a variant of the Bratu equation frequently used in combustion modeling on the unit square.

$$\Delta_x y(x) + e^{y(x)} = 0 \quad \text{for} \quad x = (\zeta,\eta) \in [0,1]^2 \quad \text{s.}t.$$

$$y(0,\eta) = y(1,\eta), \quad y(\zeta,0) = \sin(2\pi\zeta), \quad y(\zeta,1) = u(\zeta) \quad \text{for} \quad \zeta,\eta \in [0,1]$$

Hence the problem is periodic with respect to the horizontal coordinate ζ and has Dirichlet boundary conditions on the lower and upper edge of the unit square. The function u is viewed as a boundary control that can be varied to minimize the objective

$$f(y,u) = \int_0^1 \left[\frac{\partial y(\eta,\zeta)}{\partial \eta}\bigg|_{\eta=1} - 4 - \cos(2\pi\zeta) \right]^2 d\zeta + \sigma \int_0^1 \left[u(\zeta)^2 + u'(\zeta)^2 \right] d\zeta$$

In the following calculations we used $\sigma = 0.001$ and the control= design u is set initially to the constant $u(\zeta) = 2.2$. This value is not all that far from the fold point where solutions cease to exist due to overheating of the system.

We use a central difference discretization with the meshwidth 1/12.0 so that the resulting algebraic system involves 144 equations in as many variables. Since the nonlinearities occur only on the diagonal one can easily implement Jacobi's method to obtain the basic function $G(y, u)$. For this simple example we also coded by hand the corresponding adjoint iteration function $\bar{G}$ defined in (11) and even the somewhat more complicated right hand side of (17). The results were later confirmed using the automatic differentiation tool ADOL-C [10].

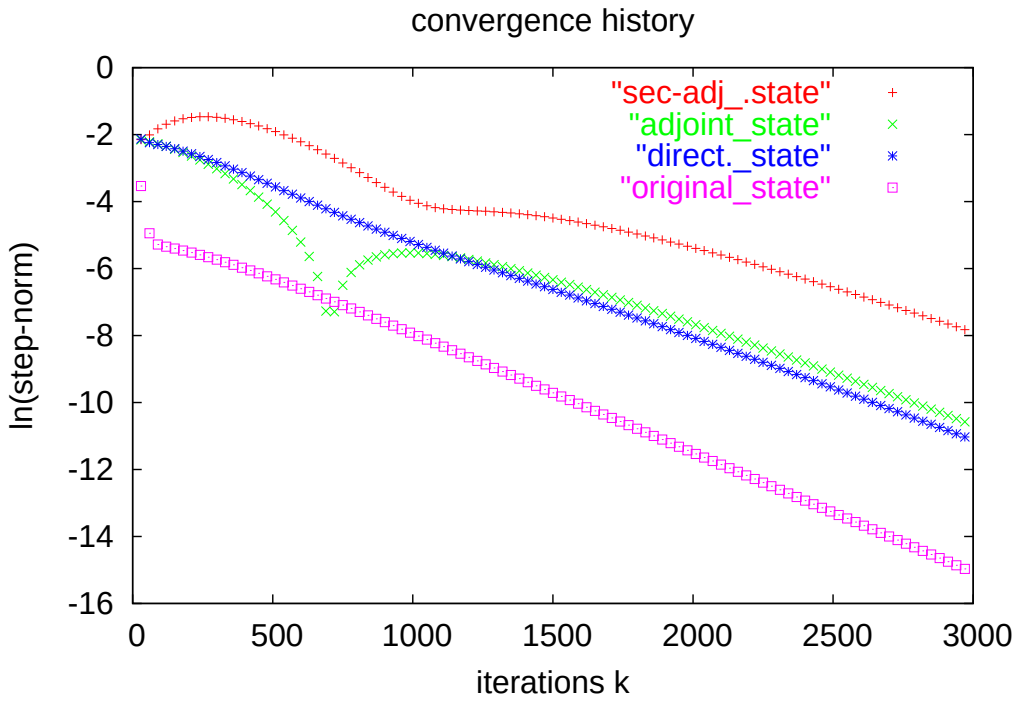

Figure 1: Convergence pattern for fixed design $u = 2.2$

As can be seen in Fig.1 the convergence of the Jacobi method is rather slow with the common R-factor being about $(1-1/300)$. The lowest curve represents the natural logarithms of the Euclidean norm ratios $\|y_{k+1} - y_k\|/\|y_1 - y_0\|$, which provide some indication of the norm ratios $\|y_k - y_*\|/\|y_0 - y_*\|$. In view of the very slow convergence this relation need certainly not be very close. Nevertheless the theory is basically confirmed with the first derivatives $\|\dot{y}_{k+1} - \dot{y}_k\|/\|\dot{y}_1 - \dot{y}_0\|$ and $\|\bar{y}_{k+1} - \bar{y}_k\|/\|\bar{y}_1 - \bar{y}_0\|$ lagging somewhat behind, and the second derivatives $\|\ddot{y}_{k+1} - \ddot{y}_k\|/\|\ddot{y}_1 - \ddot{y}_0\|$ coming in last.

On closer inspection it was found in [11] that the ratio between these derivative quantities and the original iterates confirms after an initial transition phase the

asymptotic relations (16) and (18). While the adjoints $\bar{y}_k$ were defined as vectors, the direct differentiation was performed simultaneously with respect to all components of the discretized u so that the quantity $\dot{u}$ occurring in (17) was in fact the identity matrix of order 12. Consequently, $\dot{y}_k = Z_k$ and $\dot{\bar{y}}_k = \bar{Z}_k$ had also 12 times as many components as the underlying y_k and $\bar{y}_k$, which had both $n = 144$ components.

Without and with the sign switch the Hessians H_k converged for fixed u quite rapidly to the respective limits $H(1)$ and $H(-1)$ displayed in Fig.2 and Fig. 3. Both are Toeplitz matrices and look rather similar to discretized Laplacians in 1D. It was shown in [12] by Hazra and Schulz that the reduced Hessian must have this property exactly on the original test problem without the nonlinear term, which causes only a relatively compact perturbation. By comparison the diagonal elements in $H(-1)$ are about 50% larger than those of $H(1)$, which results in slightly shorter and thus more cautious optimization steps. Moreover, there are also structural differences as $H(1)$ is apparently an M-matrix, whereas the subdiagonals of $H(-1)$ have alternating signs and don't decline quite as rapidly as those of $H(1)$.

$$\begin{bmatrix}
34.90 & -27.62 & -1.11 & -0.38 & -0.15 & -0.07 & -0.06 & -0.07 & -0.15 & -0.38 & -1.11 & -3.62 \\
-27.62 & 58.90 & -27.62 & -1.11 & -0.38 & -0.15 & -0.07 & -0.06 & -0.07 & -0.15 & -0.38 & -1.11 \\
-1.11 & -27.62 & 58.90 & -27.62 & -1.11 & -0.38 & -0.15 & -0.07 & -0.06 & -0.07 & -0.15 & -0.38 \\
-0.38 & -1.11 & -27.62 & 58.90 & -27.62 & -1.11 & -0.38 & -0.15 & -0.07 & -0.06 & -0.07 & -0.15 \\
-0.15 & -0.38 & -1.11 & -27.62 & 58.90 & -27.62 & -1.11 & -0.38 & -0.15 & -0.07 & -0.06 & -0.07 \\
-0.07 & -0.15 & -0.38 & -1.11 & -27.62 & 58.90 & -27.62 & -1.11 & -0.38 & -0.15 & -0.07 & -0.06 \\
-0.06 & -0.07 & -0.15 & -0.38 & -1.11 & -27.62 & 58.90 & -27.62 & -1.11 & -0.38 & -0.15 & -0.07 \\
-0.07 & -0.06 & -0.07 & -0.15 & -0.38 & -1.11 & -27.62 & 58.90 & -27.62 & -1.11 & -0.38 & -0.15 \\
-0.15 & -0.07 & -0.06 & -0.07 & -0.15 & -0.38 & -1.11 & -27.62 & 58.90 & -27.62 & -1.11 & -0.38 \\
-0.38 & -0.15 & -0.07 & -0.06 & -0.07 & -0.15 & -0.38 & -1.11 & -27.62 & 58.90 & -27.62 & -1.11 \\
-1.11 & -0.38 & -0.15 & -0.07 & -0.06 & -0.07 & -0.15 & -0.38 & -1.11 & -27.62 & 58.90 & -27.62 \\
-3.62 & -1.11 & -0.38 & -0.15 & -0.07 & -0.06 & -0.07 & -0.15 & -0.38 & -1.11 & -27.62 & 34.90
\end{bmatrix}$$

Figure 2: The reduced Hessian $H_*(1)$

$$\begin{bmatrix}
69.81 & -33.54 & 4.82 & -2.81 & 1.88 & -1.46 & 1.34 & -1.46 & 1.88 & -2.81 & 4.82 & -9.54 \\
-33.54 & 93.81 & -33.54 & 4.82 & -2.81 & 1.88 & -1.46 & 1.34 & -1.46 & 1.88 & -2.81 & 4.82 \\
4.82 & -33.54 & 93.81 & -33.54 & 4.82 & -2.81 & 1.88 & -1.46 & 1.34 & -1.46 & 1.88 & -2.81 \\
-2.81 & 4.82 & -33.54 & 93.81 & -33.54 & 4.82 & -2.81 & 1.88 & -1.46 & 1.34 & -1.46 & 1.88 \\
1.88 & -2.81 & 4.82 & -33.54 & 93.81 & -33.54 & 4.82 & -2.81 & 1.88 & -1.46 & 1.34 & -1.46 \\
-1.46 & 1.88 & -2.81 & 4.82 & -33.54 & 93.81 & -33.54 & 4.82 & -2.81 & 1.88 & -1.46 & 1.34 \\
1.34 & -1.46 & 1.88 & -2.81 & 4.82 & -33.54 & 93.81 & -33.54 & 4.82 & -2.81 & 1.88 & -1.46 \\
-1.46 & 1.34 & -1.46 & 1.88 & -2.81 & 4.82 & -33.54 & 93.81 & -33.54 & 4.82 & -2.81 & 1.88 \\
1.88 & -1.46 & 1.34 & -1.46 & 1.88 & -2.81 & 4.82 & -33.54 & 93.81 & -33.54 & 4.82 & -2.81 \\
-2.81 & 1.88 & -1.46 & 1.34 & -1.46 & 1.88 & -2.81 & 4.82 & -33.54 & 93.81 & -33.54 & 4.82 \\
4.82 & -2.81 & 1.88 & -1.46 & 1.34 & -1.46 & 1.88 & -2.81 & 4.82 & -33.54 & 93.81 & -33.54 \\
-9.54 & 4.82 & -2.81 & 1.88 & -1.46 & 1.34 & -1.46 & 1.88 & -2.81 & 4.82 & -33.54 & 69.81
\end{bmatrix}$$

Figure 3: The projected Hessian $H_*(-1)$

In any case, as soon as the optimization correction (7) is added using either approximations $H_k \approx H(1)$ or $H_k \approx H(-1)$ the convergence behavior is very different. The first, seemingly natural choice results in an immediate blow up whereas the second achieves convergence of the full steps without any line-search. The resulting values of the objective function $f(y_k, u_k)$, its Lagrange correction according to (15) and the Euclidean norm of the reduced gradient $\bar{u}_k$ are displayed in Fig. 4. As one can see the optimum is reached with about 6 Figure accuracy after some 1600 iterations, which is roughly twice as many as it took for the simulation with constant design u to reach that level of accuracy. The reason for the subsequent oscillation in the reduced gradient remains as yet a little mysterious. In order to make the method converge with steps of the form $-\alpha H_*(1)^{-1}\bar{u}_k$ we had to use a step multiplier α smaller than 0.04. On the other hand we could only use steps $-\alpha H_*(-1)^{-1}\bar{u}_k$ with α up to about 1.3 before convergence was lost. This seems to indicate that $-H_*(-1)^{-1}\bar{u}_k$ is of a fairly optimal size.

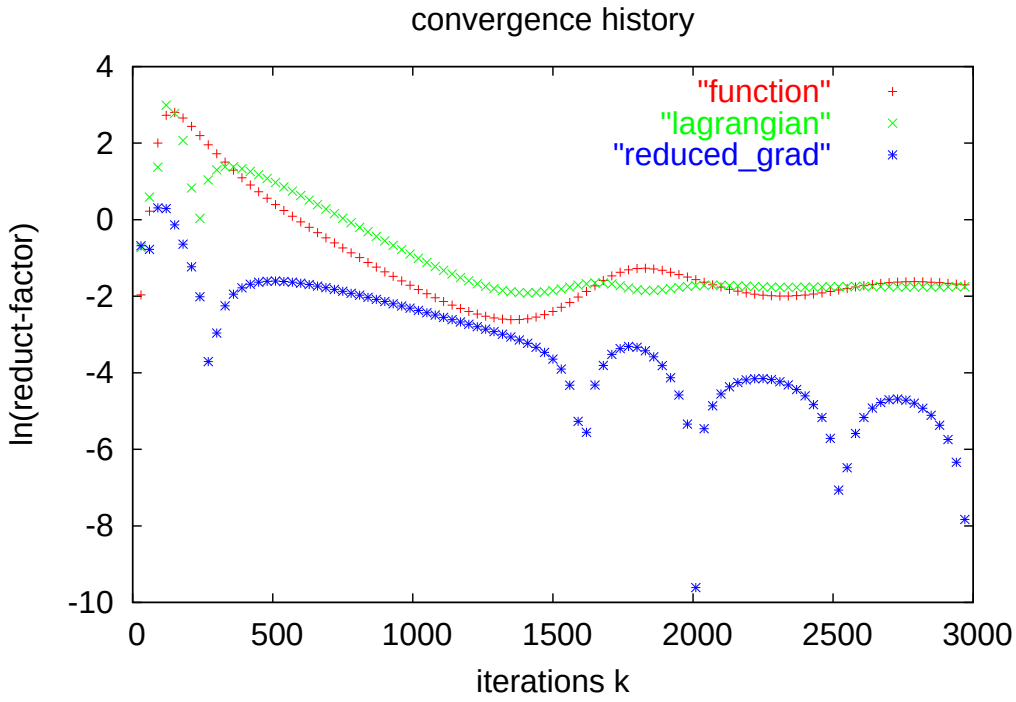

Figure 4: One-step one-shot convergence with variable design

5.1 Summary and Outlook

We have considered the problem of minimizing an objective function subject to a very large dimensional state equation with a separate design space. After reviewing some reduced SQP variants we considered one-shot methods based on a rather

slowly converging state equation solver. This approach promises a convenient transition from simulation to optimization at a limited increase in computational cost. Based on the convergence analysis in [9] for fixed design, we addressed the task of selecting a suitable preconditioner H_* for the approximate reduced gradient to allow an adjustment of the design variable at each iteration. This was referred to as one-step one-shot design. Our eigenvalue analysis yielded certain necessary convergence conditions, which are not necessarily satisfied by the conventional reduced Hessian $H_*(1)$. A more suitable choice seems to be another projection $H_*(-1)$, though there is certainly no guarantee even of local convergence either.

In any case our current approach of evaluating either one of the projected Hessians as second order vector adjoints is certainly too expensive. To avoid a cost that grows proportional to the number of design variables we will pursue a low rank updating approach. Another possible improvement concerns the spacing of optimization steps. There is some indication that normal and adjoint iterations should be repeated at least twice before each correction on the design variables. That would turn the alternating modes of the state space Jacobian into monotone modes and thus possibly favor the reduced Hessian as preconditioner. However, this simple remedy need not work if there are complex eigenvalues as one should certainly expect in general. In general one would like to obtain a relation between the contractivity factor ρ of the given simulation code and the best convergence rate on can achieve for a multi-step optimization scheme of the most general form (21). For that one can not simply ignore complex eigenvalues as we have done so far. Of course, there are many unresolved nonlocal issue, chiefly how to force convergence and how to deal with inequalities.

References

[1] L.T. Biegler, J. Nocedal, C. Schmid (1995), "A reduced Hessian method for large-scale constrained optimization," *SIAM J. on Optimization*, 5, 314-347.

[2] G. Biros and O. Ghattas (2000), "A Lagrange-Newto-Krylov-Schur Method for PDE-Constrained Optimization," *SIAG/OPT Views-and-News*, 11(2), 1-6.

[3] T.F. Coleman and A.R. Conn (1984), "On the local convergence of a quasi-Newton method for the nonlinear programming problem," *SIAM J. Num. Anal.*, 21, 755-769.

[4] Francois Courty, (2003), "Optimization différentiable en mécanique des fluides numérique," Ph.D. thesis, Université Paris-Sud.

[5] P. Deuflhard and F. Potra (1992), "Asymptotic mesh independence of Newton–Galerkin methods via a refined Mysovskii theorem," *SIAM Journal on Numerical Analysis*, 29(5), 1395–1412.

[6] J.Ch. Gilbert (1992), "Automatic Differentiation and Iterative Processes," *Optim. Methods Softw.*, 1, 13–21.

[7] A. Griewank. (2000), *Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, Vol. 19.

[8] A. Griewank (2003), "A mathematical view of automatic differentiation," *Acta Numerica*, 12, 312–398.

[9] A. Griewank and C. Faure (2002), "Reduced functions, gradients and Hessian from fixed point iteration for state equations," *Numerical Algorithms*, 30(2), 113–139.

[10] A. Griewank, D. Juedes, and J. Utke (1996), "ADOL-C, a package for the automatic differentiation of algorithms written in C/C++," *TOMS*, 22(2), 131–167.

[11] A. Griewank and A. Ponomarenko (2004), "Time-lag in derivative convergence for fixed point iterations," *Proceedings of CARI'04, 7th African Conference on Research in Computer Science*, 295–304.

[12] S.B. Hazra and V. Schulz (2004), "Simultaneous pseudo-timestepping for PDE-model based optimization problems", *Bit Numerical Math.*, 44(3), 457-472.

[13] M. Heinkenschloss and L.N. Vicente (1999), "An interface between optimization and application for the numerical solution of optimal control problems", *ACM Transactions on Math. Software*, 25(2), 157-190.

[14] B. Mohammadi and O. Pironneau. (2001), *Applied Shape Optimization for Fluids*, Numerical Mathematics and Scientific Computation, Cladenen Press, Oxford.

[15] J.M. Ortega and W.C. Rheinboldt. (1979), *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York.

[16] S. Ta'asan and G. Kuruvila and M.D. Salas (1992), "Aerodynamic design and optimization in one shot", in *30th AIAA Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, Reno, Paper 91-0025.