Optim Eng (2016) 17:663–693
DOI 10.1007/s11081-016-9323-4

Riemannian optimization and multidisciplinary design optimization

Craig Bakker[1] · Geoffrey T. Parks[1]

Received: 31 October 2014 / Revised: 13 July 2015 / Accepted: 12 May 2016 /
Published online: 25 June 2016
© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract Riemannian Optimization (RO) generalizes standard optimization methods from Euclidean spaces to Riemannian manifolds. Multidisciplinary Design Optimization (MDO) problems exist on Riemannian manifolds, and with the differential geometry framework which we have previously developed, we can now apply RO techniques to MDO. Here, we provide background theory and a literature review for RO and give the necessary formulae to implement the Steepest Descent Method (SDM), Newton's Method (NM), and the Conjugate Gradient Method (CGM), in Riemannian form, on MDO problems. We then compare the performance of the Riemannian and Euclidean SDM, NM, and CGM algorithms on several test problems (including a satellite design problem from the MDO literature); we use a calculated step size, line search, and geodesic search in our comparisons. With the framework's induced metric, the RO algorithms are generally not as effective as their Euclidean counterparts, and line search is consistently better than geodesic search. In our post-experimental analysis, we also show how the optimization trajectories for the Riemannian SDM and CGM relate to design coupling and thereby provide some explanation for the observed optimization behaviour. This work is only a first step in applying RO to MDO, however, and the use of quasi-Newton methods and different metrics should be explored in future research.

Keywords Multidisciplinary design optimization · Differential geometry · Riemannian optimization

✉ Craig Bakker
ckrb2@cam.ac.uk

[1] Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, United Kingdom

1 Introduction

There currently exist a variety of gradient-based optimization algorithms. These algorithms are well-understood and widely used, but most have been derived for flat spaces. However, it is possible to generalize these algorithms to curved spaces: Riemannian Optimization (RO) methods are gradient-based optimization algorithms derived for Riemannian manifolds. Due to the increased mathematical technicality of these methods, and the fact that the traditional algorithms have proven to be as successful as they have, RO is not widely known or commonly used, but Multidisciplinary Design Optimization (MDO) may prove to be an ideal application opportunity.

MDO problems exist on curved spaces—the feasible design manifolds defined by the state equations (Bakker and Parks 2015a). Thus far, the literature only shows use of the traditional "flat" algorithms on MDO problems, but given that these curved spaces are, in fact, Riemannian manifolds, it makes sense to consider how RO algorithms might perform in an MDO context. With our differential geometry framework, we can now do that. In this paper, we will consider some MDO problems and compare the Riemannian algorithms' performance against that of the Euclidean algorithms. We intend to focus on convergence behaviour over computational cost (for the time being) in order to determine if further investigation into the use of these algorithms is warranted.

2 Background

2.1 Riemannian geometry

Here, we reiterate and summarize relevant portions of theory which we have previously presented in more detail (Bakker and Parks 2015a). Differential geometry is concerned with doing mathematics (such as calculus) on generally non-Euclidean spaces. In Euclidean geometry, the basic structures of space are linear: lines, planes, and their analogues in higher dimensions, but differential geometry deals with manifolds. Manifolds are abstract mathematical spaces which locally resemble the spaces described by Euclidean geometry but may have a more complicated global structure (Ivancevic and Ivancevic 2007); manifolds are like higher-dimensional versions of surfaces.

A Riemannian manifold is a differentiable manifold with a symmetric, positive definite bilinear form (known as the metric tensor). Given a Riemannian manifold M, the metric tensor g_{ij} defines an inner product, and this makes it possible to perform a number of different mathematical operations on the manifold. For example, the infinitesimal length on the manifold is

$$ds^2 = g_{ij} dw^i dw^j \tag{1}$$

where w^i are the coordinates of the manifold. Note the use of index notation with the summation convention here and in the rest of this paper unless otherwise indicated.

Our index notation includes the use of superscripts and subscripts to indicate contravariant and covariant indices, respectively. We recognize that using this convention can be disorienting from an engineering perspective, but this kind of notational bookkeeping is significant for calculating derivatives and handling coordinate system changes, among other things, so we will hold to it as much as possible; we explain this further in our foundational theory paper (Bakker and Parks 2015a).

The summation convention is that repeated indices in an expression are summed over:

$$a^i b_i = \sum_i a^i b_i \tag{2}$$

If g_{ij} is known at a point on M, it is possible to calculate the Christoffel symbols at that point:

$$\Gamma^i_{kl} = \frac{1}{2} g^{im} \left(\frac{\partial g_{mk}}{\partial w^l} + \frac{\partial g_{ml}}{\partial w^k} - \frac{\partial g_{kl}}{\partial w^m} \right) \tag{3}$$

and they feature in two important places: the covariant derivative and the geodesic equation; Γ^i_{kl} are essentially intermediate quantities—they are typically used for calculating other pieces of information. The geodesic equation is

$$\ddot{w}^i + \Gamma^i_{kl} \dot{w}^k \dot{w}^l = 0 \tag{4}$$

Solutions of the geodesic equation are the paths that particles moving along the manifold would take if they were not subjected to any external forces (Ivancevic and Ivancevic 2007). They are to curved spaces what straight lines are to flat spaces: great circles on a sphere are a familiar example.

The covariant derivative, denoted with a subscript semi-colon, does two things: it projects the derivative onto the tangent space (Boothby 1986), and it maintains the tensorial character of whatever it derivates (Ivancevic and Ivancevic 2007). $v_{i,j}$ is the derivative of v_i with respect to w^j ($\mathbf{w}$ are still the manifold coordinates), and $v_{i;j}$ is then the *covariant* derivative of v_i with respect to w^j. The some sample formulae for the covariant derivative are

$$v^i_{;k} = v^i_{,k} + \Gamma^i_{jk} v^j \tag{5}$$

$$v_{i;k} = v_{i,k} - \Gamma^j_{ik} v_j \tag{6}$$

$$t^i_{kl;q} = t^i_{kl,q} + \Gamma^i_{qs} t^s_{kl} - \Gamma^s_{kq} t^i_{sl} - \Gamma^s_{lq} t^i_{ks} \tag{7}$$

Notice how covariant and contravariant indices are differentiated differently; also, $g_{ij;k} = 0$ (Szekeres 2004).

2.2 MDO background

MDO deals with the optimization of systems which have coupled subsystems. The field originally grew out of structural optimization in aerospace design at NASA in the 1980s. Today, it is applied across a range of engineering disciplines, but aerospace remains a significant application area. In MDO, problem coupling is dealt with through decomposition and coordination strategies called architectures. Martins and Lambe (2013) provide a comprehensive review of MDO architectures and describe their functioning in detail.

A key part of performing optimization with these architectures is obtaining relevant derivative information. The Global Sensitivity Equations (GSEs) are a well-known way of obtaining the total derivatives of state variables with respect to design variables by using partial derivative information from each discipline (Sobieszczanski-Sobieski 1990). Optimum Sensitivity Analysis (OSA) can be used similarly to obtain useful derivative information in multi-level optimization problems, such as those which result from using MDO architectures (Barthelemy and Sobieszczanski-Sobieski 1983), and adjoint derivatives have also proved to be useful in computationally expensive MDO contexts (Martins et al. 2005). Martins and Hwang (2012) summarize most of the relevant information regarding derivative calculations in MDO.

MDO also has connections to a variety of related fields: optimal control (Allison and Herber 2013), multi-objective optimization (Tappeta et al. 2000), metamodelling (Paiva et al. 2010), and multi-fidelity modelling (Thokala 2005). These connections continue to be explored alongside research in basic MDO methods and MDO applications.

2.3 MDO in Riemannian geometry

In this section, we summarize the translation of MDO into Riemannian geometry (Bakker and Parks 2015a). Consider a generic MDO problem as

$$\min f(\mathbf{x}, \mathbf{y}, \mathbf{z}) \tag{8}$$

$$\mathbf{g}(\mathbf{x}, \mathbf{y}, \mathbf{z}) \leq \mathbf{0} \tag{9}$$

$$\mathbf{h}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{0} \tag{10}$$

where $\mathbf{x}$ is the vector of local design variables, $\mathbf{y}$ is the vector of state variables, $\mathbf{z}$ is the vector of global design variables, $\mathbf{g}$ is the vector of inequality constraints, and $\mathbf{h}$ is the vector of state equations in residual form; (8) is just a rearrangement of the state equations:

$$y^i = \psi^i(\mathbf{x}_{(i)}, \tilde{\mathbf{y}}_{(i)}, \mathbf{z}), \ i = 1, 2, \ldots \tag{11}$$

where $\mathbf{x}_{(i)}$ is the set of local variables for discipline i and $\tilde{\mathbf{y}}_{(i)}$ is the set of all state variables excluding y^i. A two-discipline example of this is shown in schematic form in Fig. 1. The variables can be further simplified by lumping them together: $\mathbf{w} = \{\mathbf{x}\ \mathbf{z}\}$ and $\mathbf{v} = \{\mathbf{w}\ \mathbf{y}\}$.

The total design space for this problem is $\Re^m$, where m is the total number of variables, and the feasible design space M_{feas} is an n-dimensional manifold, where n is the total number of design variables, defined by the solutions to the state

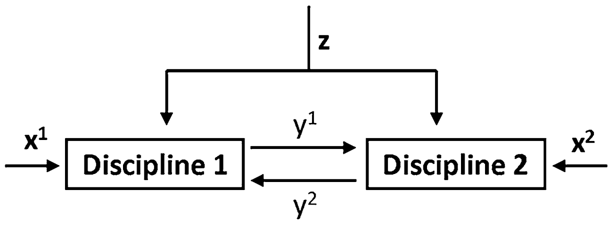

Fig. 1 A two-discipline MDO schematic

equations; M_{feas} is a subspace of the total design space. Assuming sufficient differentiability, the induced metric g_{ij} on M_{feas} is

$$g_{ij} = \frac{\partial v^k}{\partial w^i}\frac{\partial v^k}{\partial w^j} = \delta_{ij} + \frac{\partial y^k}{\partial w^i}\frac{\partial y^k}{\partial w^j} \tag{12}$$

where the derivatives of **y** with respect to **w** are calculated from the state equations by using the implicit function theorem. The Christoffel symbols are then

$$\Gamma^i_{kl} = g^{im}\frac{\partial y^s}{\partial w^m}\frac{\partial^2 y^s}{\partial w^k \partial w^l} \tag{13}$$

Turning to the objective function derivatives, we know that for a scalar function, the covariant derivative is identical to the regular derivative:

$$f_{;i} = f_{,i} = \frac{df}{dw^i} = \frac{\partial f}{\partial w^i} + \frac{\partial f}{\partial y^k}\frac{\partial y^k}{\partial w^i} \tag{14}$$

This is just the reduced gradient of f. However, taking the covariant derivative again does not result in the reduced hessian. The reduced hessian and the second covariant derivative of f are, respectively,

$$\begin{aligned} f_{,ij} = {} & \frac{\partial^2 f}{\partial w^i \partial w^j} + \frac{\partial^2 f}{\partial w^i \partial y^k}\frac{\partial y^k}{\partial w^j} + \frac{\partial^2 f}{\partial w^j \partial y^k}\frac{\partial y^k}{\partial w^i} + \frac{\partial^2 f}{\partial y^m \partial y^k}\frac{\partial y^k}{\partial w^i}\frac{\partial y^m}{\partial w^j} \\ & + \frac{\partial f}{\partial y^k}\frac{\partial^2 y^k}{\partial w^i \partial w^j} \end{aligned} \tag{15}$$

$$f_{;ij} = f_{,ij} - \Gamma^l_{ij} f_{,l} \tag{16}$$

As we have noted previously (Bakker and Parks 2015a), this differential geometry perspective allows us to consider the constrained optimization problem as an unconstrained problem on a Riemannian manifold. The resulting Lagrangian (on the Riemannian manifold) could have *inequality* constraint terms, but it would not have equality terms corresponding to the state equations because those constraints would have already been "absorbed", for lack of a better term, into the Riemannian manifold and its structure; any additional non-state equality constraints could feature in the Lagrangian, though.

Table 1 Commonly used quantities

Symbol	Denotation
f	Objective function
$\mathbf{g}$, g^i	Vector of inequality constraints
$\mathbf{h}$, h^i	Vector of state equations in residual form
$\mathbf{w}$, w^i	Vector of all design variables
$\mathbf{x}$, x^i	Vector of all local design variables
$\mathbf{z}$, z^i	Vector of all global design variables
$\mathbf{y}$, y^i	Vector of state variables (defined by the state equations)
g_{ij}, g	Metric tensor
Γ^i_{jk}	Christoffel symbols
$a_{;i}$	Covariant derivative of a
$\alpha_{(k)}$	Step length at iteration k
$\mathbf{d}_{(k)}$	Step direction at iteration k

Since the variety of variables used in this paper could prove confusing, we have provided a list of commonly used terms, in both vector (boldface) and index notation where appropriate, in Table 1.

2.4 Theoretical background for Riemannian optimization

The metric tensor can be used to raise and lower indices. Consider the inner product $\langle \mathbf{u}, \mathbf{v} \rangle$. In general, this can be expressed as $g_{ij}u^i v^j$; in flat spaces, where $g_{ij} = \delta_{ij}$,

$$g_{ij}u^i v^j = \delta_{ij}u^i v^j = u^j v^j \tag{17}$$

This procedure can actually be considered as simply taking the covariant form of $\mathbf{u}$, $u_j = g_{ij}u^i$, and doing a tensor contraction with $\mathbf{v}$: $u_j v^j$. This is relevant because of how it relates to objective function derivatives and gradients. Technically, the objective function derivative $\frac{\partial f}{\partial x^i} = f_{,i}$ is a covector, not a vector. However, the gradient is a vector, so we use the metric tensor inverse to raise the necessary index and get $g^{ij} f_{,j}$ as our gradient.

Geodesics, which are defined by (4), are an integral part of most RO derivations and proofs. In $\Re^n$, gradient-based optimization algorithms do line searches; in RO, the search is done along geodesics (at least in theory). A geodesic extending in the direction ξ at point $\mathbf{x}$ can be denoted by $\exp_{\mathbf{x}}(t\xi) : T_{\mathbf{x}}M \to M,\ t > 0$. The exponential maps vectors in $T_{\mathbf{x}}M$ to the manifold itself (Boothby 1986).

Although this notation is simple and elegant, in practice, these geodesics are difficult to calculate (Yang 2007) as they require the solution of the geodesic equation. Geodesics are often used in convergence proofs for RO algorithms (da Cruz Neto et al. 1998; Ferreira and Svaiter 2002; Luenberger 1972). In practical calculations, however, geodesics are usually approximated with a tangent step with a constraint restoration step or a quadratic approximation (Gabay 1982). Fortunately, it is still possible to do the convergence analysis with these approximations (Qi 2011).

In order to implement these more computationally tractable problems, we need retractions. Retractions are first-order approximations to the exponential: $R_{\mathbf{x}} : T_{\mathbf{x}}M \to M$ (Qi et al. 2010). For a vector ξ at a point $\mathbf{x}$,

$$R_{\mathbf{x}}(\xi) = \exp_{\mathbf{x}}(\xi) = \mathbf{x} + \xi \tag{18}$$

Baker (2008) gives three conditions for a retraction:

1. R is continuously differentiable
2. $R_{\mathbf{x}}(0_{\mathbf{x}}) = \mathbf{x}$ ($0_{\mathbf{x}}$ is the zero element on $T_{\mathbf{x}}M$)
3. $DR_{\mathbf{x}}(0_{\mathbf{x}}) = 1_{T_{\mathbf{x}}M}$, the identity mapping on $T_{\mathbf{x}}M$

Another important concept is that of parallel or vector transport. Parallel transport is the transportation of vectors between tangent spaces which preserves vector lengths and angles (Ji 2007); it applies analogously to moving tensors between the products of tangent and cotangent spaces (Whiting 2011). There is a connection which "connects" those tangent spaces. The connection is like a differential equation which specifies how quantities on the manifold change locally, and we can represent that connection with the Christoffel symbols. Parallel transport is the integration of the connection along a path (Whiting 2011), and thus parallel transport is actually defined by the connection (Li and Wang 2008).

Furthermore, for a given metric, there is a unique metric-compatible, symmetric connection (Boothby 1986). This connection may be easy to specify, but it is difficult to compute parallel transport in practice (Nishimori 2005). Therefore, Qi (2011) defines vector transport as a relaxation of parallel transport corresponding to retraction as a relaxation of the exponential mapping.

Finally, we consider convexity. In $\Re^n$, convexity of both sets and functions is defined with respect to straight lines: a function f is convex if

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y), \ t \in [0,1] \tag{19}$$

and a set S is convex if

$$tx + (1-t)y \in S \ \forall x, y \in S, \ t \in [0,1] \tag{20}$$

In Riemannian geometry, however, convexity is defined with geodesics instead of straight lines (Bento et al. 2012); assuming that there is a unique geodesic $\gamma(t)$ connecting x and y such that $\gamma(0) = x$ and $\gamma(1) = y$, a function f is convex if

$$f(\gamma(t)) \leq tf(x) + (1-t)f(y), \ t \in [0,1] \tag{21}$$

and a set S is convex if

$$\gamma(t) \in S \ \forall x, y \in S, \ t \in [0,1] \tag{22}$$

When geodesic uniqueness fails, so too does a unique definition of convexity (Udrişte 1994). For a metric-compatible connection, convexity is in fact determined by the choice of metric tensor (Rapcsák 1991): convexity is determined by the geodesics, which are determined by the connection, which is in turn determined by the metric tensor. Convexity is not, however, determined by the coordinate system used (Udrişte 1996a). This means that it is sometimes possible to transform non-convex problems into convex problems through the choice of a favourable metric tensor (Bento et al. 2012).

2.5 A review of the field

Luenberger (1972) wrote the first significant paper applying RO to equality-constrained optimization. He used a projection operator to get the objective function derivatives onto the tangent spaces (using the coordinates of the embedding space, not coordinates specifically for the manifold itself) and proved convergence for a descent method along the manifold geodesics.

What may perhaps be considered the foundational paper, however, was later written by Gabay (1982). Gabay began from a constrained optimization problem and used slack variables to convert the inequality constraints into equality constraints. This is a fairly common practice in RO (Rapcsák 1989; Tanabe 1982). He then addressed the question of partitioning the problem into dependent and independent variables. Although some problems may have a natural partition available to them, arbitrary partitions may lead to poor performance. Throughout the rest of the paper, though, he avoided partitions and used projections from the embedding spaces instead of an explicit manifold coordinate representation. This is also fairly common when applying RO to general nonlinear optimization problems (Luenberger 1972; Rapcsák and Thang 1995; Tanabe 1980). Rheinboldt (1996) is a notable exception in using a QR factorization on the constraint derivatives to get an orthonormal coordinate system at any given point on the manifold. Finally, considering two different types of geodesic approximation (a tangent step with constraint restoration and a quadratic approximation), Gabay (1982) developed a Riemannian version of a quasi-Newton method.

Since then, several authors have developed Riemannian versions of algorithms such as the Steepest Descent Method (SDM), Newton's Method (NM) and the Conjugate Gradient Method (CGM) (Smith 1994; Udrişte 1994). However, implementing geodesic searches is difficult to do in general. As a result, many of the examples given to demonstrate the algorithms use special manifolds – the Grassman and Stiefel matrix manifolds are particularly popular—where it is relatively easy to calculate the requisite quantities (Nishimori 2005; Qi 2011; Rapcsák 2002; Smith 1994).

In order to address this better, Qi (2011) looked in more detail at how to approximate the geodesic search typically specified by the RO algorithms proposed. She showed that the exponential map could be relaxed to the more general retraction while maintaining algorithm convergence and gave conditions and proofs for that convergence. Furthermore, different retractions actually produce different optimization algorithms. As Ring and Wirth (2012) later pointed out, some retractions are better than others: there is a problem-specific tradeoff between computability and the number of iterations required for convergence.

Baker (2008) continued to look at retraction-based RO while extending trust region methods to RO. Instead of directly applying Euclidean optimization algorithms to Riemannian manifolds, he used the retraction to lift the objective function on the manifold to the manifold tangent spaces and then applied the

Euclidean algorithms on those flat spaces. For example, he would use one retraction to build a trust region model in a tangent space, but he showed that a different retraction could then be used to explore that trust region model. He further showed that this process, even using two different retractions, was still convergent.

As already noted, the majority of the RO studies focused on equality-constrained optimization; slack variables would be used to turn any inequalities into equalities. There were, however, some exceptions to that general methodology. Udrişte (1996b) used a logarthmic barrier function, and Ji (2007) explored this in more detail. Ji focused on interior-point methods and self-concordant functions, and he developed damped versions of SDM, NM, and CGM to deal with these inequality-constrained RO problems. Udrişte (1994) also considered using a method of feasible directions to address the same issue.

In parallel to all of this, Tanabe (1979a, b, 1980, 1982) looked at continuous analogues to these discrete optimization processes in a Riemannian context. Typically working through the projection-based approach of Luenberger and Gabay, Tanabe used continuous versions of descent and constraint satisfaction processes (with components tangent and normal to the manifold, respectively) to define the optimization problem. He then used differential geometry to analyze those dynamical systems on the manifold; the constrained optimization could be solved with a numerical ordinary differential equation (ODE) integrator. Dean (1988) later continued on in this vein and claimed that the problems thus generated were easier to analyze than the corresponding discrete optimization algorithms.

Alongside algorithm development, there have also been a variety of pertinent theoretical works done on the subject. Yang (2007) and Qi (2010) separately worked to generalize the Euclidean Armijo search conditions to RO. Ferreira and Svaiter (2002) similarly extended Kantorovich's theorem to NM on Riemannian manifolds. A number of proofs for the convergence of geodesic-based descent methods were produced (e.g. da Cruz Neto et al. 1998), but most of these proofs required certain curvature conditions on the manifold. Wang (2011), however, provided a curvature-independent convergence proof.

On a more applied level, some authors had interesting experiences with choosing different metric tensors. Udrişte (Udrişte 1996b), for example, proposed using the hessian of the Lagrangian (for an inequality-constrained RO problem) as a metric tensor; this could work to enforce feasibility during the optimization process. The nature of manifolds with objective function hessians as their metric tensors was, however, still an open problem 20 years ago (Udrişte 1994); any current advancements in this regard are unknown to us. In general, though, it is possible to choose the metric tensor in order to take advantage of problem-specific behaviour if enough is known about the problem beforehand, and doing this does not affect the location of the optima (Ring and Wirth 2012). Munier (2007) provided an example of this: by choosing a particular metric tensor and using RO on the Rosenbrock problem, he was able to turn the problem into a convex optimization, and this produced a far more efficient solution process than the typical Euclidean methods.

There are three other areas worth mentioning, though we will not go into them in detail. Firstly, Bento et al. (2012) touched on Multi-Objective Optimization (MOO) on Riemannian manifolds in their paper. We have not seen any other work in this area, but given that single-objective RO is as established (though small) as it is, it makes sense to expand into multiple objectives. Secondly, Whiting (2011) looked at path optimization on Riemannian manifolds. This ties in to the calculus of variations and ultimately control problems in Riemannian geometry. Although we have not closely investigated this, we suspect that there exist a number of other works considering Riemannian control problems. Thirdly, Potra and Rheinboldt (1989) considered a differential geometric approach to Differential Algebraic Equations (DAEs). DAEs are related to equality-constrained optimization through constrained ODE-based optimization (Tanabe 1980). Given the presence of differential constraints in control problems, the study of these systems is probably also relevant to constrained optimal control problems.

2.6 Comments on Riemannian optimization

A variety of theoretical results and methods has been produced for RO. Essentially, if it can be done in $\mathfrak{R}^n$, it can be done analogously in Riemannian geometry (e.g. trust region methods, quasi-Newton methods), and there are a plethora of accompanying convergence proofs; this likely reflects the fact that most of the RO researchers are mathematicians, not engineers.

One of the difficulties in applying these methods to real optimization problems is finding appropriate manifold coordinate systems. Applications of the methods tend to fall into one of two groups: using projections with matrix notation to stay in the coordinate system of the embedding space (Gabay 1982; Luenberger 1972; Tanabe 1980), or working on manifolds with a special analytical structure that permits the explicit calculation of the relevant quantities (Nishimori 2005; Qi 2011; Rapcsák 2002; Smith 1994). The former perspective is general but fails to take advantage of the techniques and tools available for dealing with explicit manifold coordinates, and the latter approach is useful in specific contexts but only deals with a small subset of constrained nonlinear optimization problems. Essentially, coordinates (of some sort) are needed to do the calculations, and, in general, the mathematics are more easily grasped in a coordinate-based tensor notation than in either matrix or coordinate-free notation.

3 Applying Riemannian optimization to MDO

With our differential geometry framework, we can now apply RO algorithms to MDO; we have an explicit coordinate representation, and we can calculate all of the relevant quantities. Furthermore, although we will not do this here, we could use our previous derivations (Bakker et al. 2012) and apply these Riemannian algorithms to several different MDO architectures.

In this paper, we will only consider gradient-based optimization techniques (SDM, NM, and CGM), not metaheuristic methods. The key differences between Riemannian and Euclidean optimization techniques lie in how derivative information is handled and the path along which searching is done once a search direction has been chosen; the former, at least, would not apply to 0-th order methods. Given that metaheuristics are often designed to avoid the need for gradient information—a metaheuristic method might be chosen because the objective function is not smooth—this takes away a significant amount of potential overlap between metaheuristics and RO. For a method like Particle Swarm Optimization (Kennedy and Eberhart 1995), for example, there might still be the possibility of having the particles move along geodesics instead of straight lines, but calculating the geodesics would require second-order derivative information about the state equations, so RO methods may not be very helpful here, either. Although there may be potential crossover between metaheuristics and RO, we will not explore that here.

3.1 Algorithm formulae

Consider a general optimization algorithm to be

$$\mathbf{w}_{(k+1)} = \mathbf{w}_{(k)} + \alpha_{(k)}\mathbf{d}_{(k)} \tag{23}$$

where k in brackets indicates the iteration, $\mathbf{w}$ are the design variables, $\mathbf{d}$ is the step direction, and α is the step length. To derive SDM and NM, let us start with a second-order *Riemannian* Taylor series about (with no loss in generality) the origin:

$$f\left(w^i\right) = f(0) + f_{,i}w^i + \frac{1}{2}f_{;ij}w^iw^j \tag{24}$$

As we have already noted, the gradient is not $f_{,i}$ but $g^{ij}f_{,i}$ and thus our descent direction for SDM is $d^i = -g^{ij}f_{,j}$. If we wish to explicitly calculate a step length α, we substitute $w^i = \alpha d^i$ into (24), differentiate, solve for α, and get

$$\alpha = \frac{g^{ij}f_{,i}f_{,j}}{g^{ij}f_{,j}f_{;ik}g^{kl}f_{,l}} \tag{25}$$

For NM, we start from (24) and differentiate to find an optimal search direction d^j and step length all at once. We get

$$f_{;ij}d^j = -f_{,i} \tag{26}$$

If we let $\eta^{ij} = \left[f_{;ij}\right]^{-1}$, then $d^j = -\eta^{ij}f_{,i}$; it is implied that $\alpha = 1$ for NM. For CGM, we will not rederive the method from scratch, but we will give the formulae for Riemannian CGM (using a Fletcher-Reeves update scheme for the step direction). For CGM, the first step and any restart step is the same as that of SDM. After that,

$$\alpha_{(k)} = -\left(\frac{d^if_{,i}}{d^if_{;ij}d^j}\right)_{(k)} \tag{27}$$

$$d^i_{(k)} = -\left(g^{ij}f_{,j}\right)_{(k)} + \frac{\left(f_{,i}g^{ij}f_{,j}\right)_{(k)}}{\left(f_{,i}g^{ij}f_{,j}\right)_{(k-1)}}d^i_{(k-1)} \tag{28}$$

The subscript index in parentheses is used to denote the iteration step where necessary. Compare these with the corresponding formulae in (29), (30), and (31) for

SDM, NM, and CGM, respectively. Note the non-covariant derivatives and lack of metric tensor inverses.

$$\alpha = \frac{f_{,i}f_{,i}}{f_{,i}f_{,ik}f_{,k}}, \quad d^i = f_{,i} \tag{29}$$

$$d^j = -\eta^{ij}f_{,i}, \quad \eta^{ij} = \left[f_{,ij}\right]^{-1} \tag{30}$$

$$\alpha_{(k)} = -\left(\frac{d^i f_{,i}}{d^i f_{,ij} d^j}\right)_{(k)}, \quad d^i_{(k)} = -\left(f_{,i}\right)_{(k)} + \frac{\left(f_{,i}f_{,i}\right)_{(k)}}{\left(f_{,i}f_{,i}\right)_{(k-1)}} d^i_{(k-1)} \tag{31}$$

3.2 Algorithm approximations

In Sect. 2.4, we mentioned two kinds of approximations used in practice when applying RO algorithms: retractions to approximate geodesics and vector transport to approximate parallel transport. We will use both here.

Firstly, we will use a retraction to do a line search instead of a geodesic search in some cases. This applies to the Riemannian and Euclidean versions of SDM, NM, and CGM. A geodesic search requires integrating the geodesic equation from an initial point (the design point at that iteration) with an initial "velocity" (given by the search direction). For our purposes, the retraction $R_{\mathbf{w}}(\xi) = \mathbf{w} + \xi$, where $\mathbf{w}$ is the point in space and ξ is the vector, will suffice. This retraction is easy to calculate, and it is essentially just a typical line search performed as if the design space were flat.

Secondly, we will always use vector transport instead of parallel transport. Vector or parallel transport only applies to CGM (and not SDM or NM) because of its step update scheme incorporating information from previous iterations. Parallel transport would require us to integrate the connection, for the vector in question, along a geodesic which would itself have already been determined by integrating the geodesic equation (Szekeres 2004). Doing this would greatly increase the computational cost of the optimization, however, and lacking a compelling reason to implement parallel transport instead of vector transport, we will forego using it here. As such, we will use the vector transport $T_{\mathbf{w}_{(1)} \to \mathbf{w}_{(2)}}(\xi) = \xi$, from $\mathbf{w}_{(1)}$ to $\mathbf{w}_{(2)}$, which corresponds to our retraction as a "flat" approximation of the Riemannian concept in question.

3.3 Motivation

These RO algorithms are gradient-based optimization methods. Given the widespread use of gradient-based optimization in MDO, there is value in developing and applying new gradient-based optimization algorithms for use in MDO. Although these RO algorithms have previously been used in other contexts, they are new to MDO and therefore warrant testing on MDO problems to evaluate their performance against comparable methods.

Through the metric tensor and the covariant derivative, the RO algorithms take the feasible design manifold's properties into account in a way that the usual "flat" algorithms do not. If manifold properties do indeed affect the performance of optimization algorithms, then the RO algorithms may therefore be more effective in an MDO context because of their natural connection to manifold characteristics. We will compare the effectiveness of the RO algorithms with their Euclidean counterparts to investigate whether this is in fact the case. Numerical results one way or the other will not be conclusive, but if the RO algorithms show promise here, then perhaps further, more rigorous study will be warranted.

A cursory study of the formulae involved indicates that the RO algorithms are probably more computationally expensive than the flat algorithms. Although we recognize this now, we do not intend to address this at the present time. If the RO algorithms should prove to be the more effective option, then a more in-depth comparison of algorithm cost and the tradeoffs between cost and effectiveness may be worth doing.

3.4 Procedure

Having described the nature and operation of the Riemannian algorithms in question, we now want to do a quantitative comparison of their effectiveness and efficiency in terms of convergence percentage and iterations to convergence, respectively. We will begin with a two-dimensional illustrative problem for conceptual and visualization purposes, continue on with a two-discipline analytical MDO problem, and conclude with some different objective functions for a conceptual satellite design problem from the literature (Mesmer et al. 2013).

For the analytical problems, we will consider a calculated step size, a retraction-based line search, and a geodesic search for each algorithm (both the Riemannian and Euclidean versions). However, for the satellite design problem, we will only use a calculated step size and a retraction-based line search because of the prohibitive computational cost of applying a geodesic search in that context; see Sect. 5 for further discussion of the computational costs associated with geodesic searches. For each of these variants, we performed 100 optimization runs of each algorithm in MATLAB$^{\circledR}$ (The MathWorks Inc. R2010a) on each problem. The initial points for each optimization were generated by *sobolset* over the design space and then solved for the state variables using *fsolve* to give a feasible initial point. We then carried out the optimization with the Multidisciplinary Feasible MDO architecture (Cramer et al. 1994); we used *fsolve* to do the multidisciplinary analysis at each iteration.

Each problem only had explicit bounds on the design variables as constraints. We enforced these constraints by implementing a feasible directions search method: projected search directions were used when on design space boundaries and calculated step sizes were truncated to keep them from crossing design space boundaries. Figure 2 shows an example of how this works: a step from A to B would be truncated to point C, on the boundary, and the step from C to D would then be projected to point E. Our problems did not have non-state equality constraints (the state equations were handled using a multidisciplinary analysis to solve for the state variables), and the only inequalities present were explicit bounds on the design variables, so all of the boundaries were flat. Had equality or nonlinear inequality

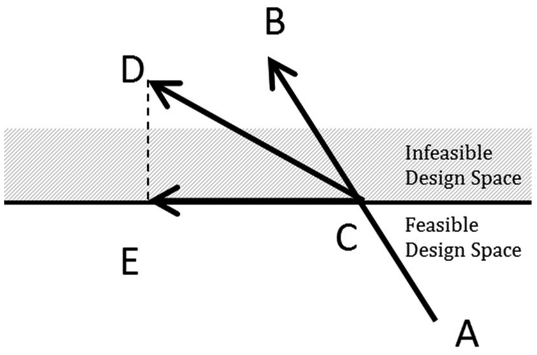

Fig. 2 Feasible search directions constraint handling diagram

constraints been present, we could have used a feasibility restoring step along with the search direction projection.

We considered the algorithm to have converged if the norm of the (projected) negative gradient was less than 10^{-3}. With gradient-based methods, we were only concerned with finding a local minimum; questions of global optimality were beyond the scope of our methods. We also terminated our algorithms if the norm of the change in the design variables was below 10^{-6} or, for the line and geodesic searches, if the search direction was an ascent direction.

In our CGM implementations, we used restarts after n iterations for a problem with n design variables. For the retraction-based line searches, we bracketed the one-dimensional minimum and then used golden section interval reduction search to find that minimum along the search direction to within a design variable tolerance of 10^{-6}. The geodesic search, however, determined the geodesic by integrating the geodesic equation using *ode15s*, a stiff MATLAB® solver (The MathWorks Inc. R2010a), until the one-dimensional minimum was found; the geodesic found either a design space boundary or a point at which the projection of the negative gradient along the search direction was below 10^{-3}. To reduce ill-conditioning, the initial direction for each geodesic search was normalized to unit length.

To summarize our implementation discussion, we have described our algorithm implementation in a step-by-step format below.

1. Choose an initial design point and solve the state equations for $\mathbf{y}$ using *fsolve*. Calculate any necessary derivative information.
2. Calculate the search direction from the derivative information for the chosen algorithm type. Project the search direction onto the boundary if the design point is on the boundary and the search direction is directed out of the feasible design space.
3. Determine the step size: calculate it analytically and truncate it if it lands outside of the feasible design space, or determine it using a line/geodesic search.
4. Solve for the new state variable values and calculate any necessary derivative information.
5. Repeat steps 2–4 until a termination criterion is met.

4 Results

4.1 Two-dimensional illustrative problem

For readability's sake, we dispense here, and in Sect. 4.2, with the subscript/superscript convention; all indices are subscripted. The first problem we considered was

$$\min f = \left(y - x_1^2\right)^2 + e^{-x_2} \tag{32}$$

$$h = x_1 e^{x_1} + (x_2 + 0.5)^3 + \sinh y = 0 \tag{33}$$

$$-1 \leq \mathbf{x} \leq 1 \tag{34}$$

This is not a true MDO problem. We created this problem for visualization and explanation purposes: its small size allowed us to illustrate optimization behaviour and algorithm implementation more easily than on higher-dimensional problems. It does not have multiple disciplines, but it has design variables and a state variable defined by a state equation (MDO problems have design variables and multiple state variables defined by state equations); we can solve its state equation in residual form with a root-finding algorithm (as can be done, in principle, with MDO problems); and we can calculate the necessary Riemannian optimization quantities using our MDO formulae. As such, *it has some of the qualities of an MDO problem*, and that makes it useful for an initial demonstration. The optimization results for a calculated step size, line search, and geodesic search are shown in Tables 2, 3, and 4, respectively.

Table 2 shows that all of the methods converged for the calculated step size. In this case, the Riemannian algorithms were all somewhat faster than their Euclidean counterparts. As would be expected, SDM was the slowest and NM was the fastest. The results in Table 3, however, now have two nonconverged runs for each of the NM, and although the Riemannian SDM is still slightly faster than Euclidean SDM, the Riemannian NM and CGM are now slightly slower than the Euclidean versions. We also note that the line search methods are all roughly two iterations faster, on

Table 2 Algorithm iterations to convergence, two-dimensional illustrative problem (calculated step size)

	SDM	SDM—RO	NM	NM—RO	CGM	CGM—RO
Maximum	17	13	15	8	11	10
Minimum	5	5	3	2	4	4
Mean	10.75	9.45	5.92	5.03	7.67	6.54
% Convergence	100	100	100	100	100	100

Table 3 Algorithm iterations to convergence, two-dimensional illustrative problem (line search)

	SDM	SDM—RO	NM	NM—RO	CGM	CGM—RO
Maximum	16	12	6	8	8	8
Minimum	2	2	2	2	2	3
Mean	8.49	8.04	3.66	3.89	5.03	5.16
% Convergence	100	100	98	98	100	100

Table 4 Algorithm iterations to convergence, two-dimensional illustrative problem (geodesic search)

	SDM	SDM—RO	NM	NM—RO	CGM	CGM—RO
Maximum	15	14	6	10	8	9
Minimum	1	1	2	2	1	1
Mean	8.94	8.16	3.81	4.34	5.30	5.65
% Convergence	100	100	98	96	100	100

average, than the calculated step size methods. Finally, as we see in Table 4, geodesic search produced results similar to line search in terms of the relative performances of the algorithms, but the geodesic search was slightly slower, across all algorithms, than the line search.

The nonconvergence in NM can be explained by considering the phase portraits of $\dot{\mathbf{x}} = \mathbf{d}$ for the Euclidean and Riemannian versions; see (Bakker et al. 2013b) for more on the use of ODEs to investigate optimization behaviour. Figure 3 shows the phase portrait for Euclidean NM, and Fig. 4 shows the phase portrait for Riemannian NM: the vectors have been normalized to show direction only, and the thick black lines indicate where the hessian (for Euclidean NM) or covariant hessian (for Riemannian NM) is singular. In some cases, the search direction for the algorithm flips directions across these lines. If the (covariant) hessian is no longer positive definite, the search direction may actually end up being an ascent direction, and this terminates the line and geodesic searches—thus why they terminated without converging.

Moreover, although this did not arise in our tests, it is clear that the calculated step size could fail to converge for the Riemannian NM as it would not for the Euclidean NM. Both algorithms have some areas of the domain where the flow is away from the optimum (i.e. the problem is nonconvex in those regions), but that flow is never directed into one of the corners for NM—near the corners, there is always a component of the flow directed away from that corner. As can be seen in Fig. 4, however, around $(x_1, x_2) = (-1, -1)$, the Riemannian NM flow is directed into the corner, and thus the algorithm would terminate there.

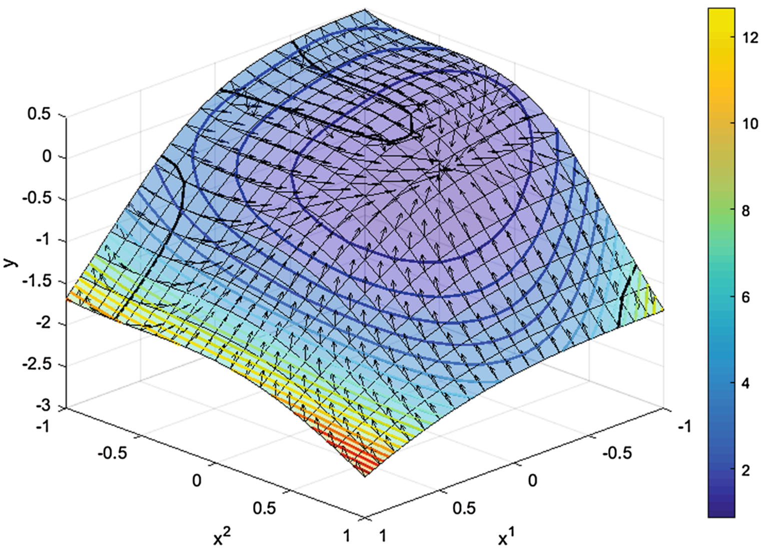

Fig. 3 Euclidean NM phase portrait, two-dimensional illustrative problem

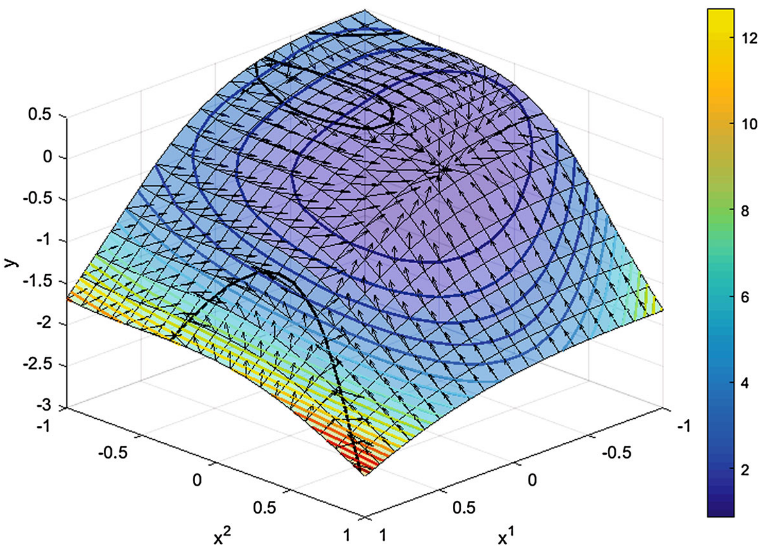

Fig. 4 Riemannian NM phase portrait, two-dimensional illustrative problem

4.2 Two-discipline analytical MDO problem

The next problem used was

$$\min f = z_1^2 + z_2^2 - y_1 + e^{-y_2} + x_1^2 \tag{35}$$

$$y_1 = x_1 - x_2^2 - z_1 z_2 + y_2^3 \tag{36}$$

$$y_2 = -x_3 x_4 + z_1 + z_2^2 - y_1 \tag{37}$$

$$-1 \leq \mathbf{w} \leq 1 \tag{38}$$

This problem is similar in form to, though greater in dimensionality than, the analytical test problem created by Sellar et al. (1996) and used elsewhere (e.g. (Bakker and Parks 2015b; Perez et al. 2004). Our state equations for disciplines 1 and 2 are in (35) and (37), respectively. If we take them and put them in residual form, we obtain

$$\mathbf{h} = \begin{Bmatrix} h_1 \\ h_2 \end{Bmatrix} = \begin{Bmatrix} y_1 - x_1 + x_2^2 + z_1 z_2 - y_2^3 \\ y_2 + x_3 x_4 - z_1 - z_2^2 + y_1 \end{Bmatrix} = \mathbf{0} \tag{39}$$

Note how this corresponds to our description in Sect. 2.3. To reiterate, our optimization algorithm takes steps in $\mathbf{w}$, and the multidisciplinary analysis then solves (39) to calculate $\mathbf{y}$.

The Euclidean and Riemannian SDM consistently failed to converge for the calculated step size: both produced optimization runs which were apparently chaotic. This can happen with SDM if the step size is too large (van den Doel and Ascher 2012). Table 5 shows the Riemannian versions performing worse than the Euclidean ones in terms of both iterations to convergence and convergence percentage.

Surprisingly, Riemannian SDM and CGM with line search produced no converged runs. Further investigation showed that the problem lay in the methods' behaviour at design space boundaries. Consider the diagram in Fig. 5: it shows the search directions $-f_{,i}$ and $-g^{ij}f_{,j}$ pointing outwards into an infeasible region (shaded in grey) along with the projections of those search directions onto the design space boundary. Since g^{ij} is positive definite, the angle between $-f_{,i}$ and $-g^{ij}f_{,j}$ is less than 90°. However, in this instance, the directions fall on opposite sides of the constraint normal, so the search directions' projections point in opposite directions. The projection of $-f_{,i}$ has to be a descent direction, so the projection of $-g^{ij}f_{,j}$ must be an ascent direction and thus the line search terminates the algorithm without converging. The same thing can and does happen with NM. See our comments at the end of Sect. 5.2 regarding why Riemannian SDM and CGM may be prone to this kind of behaviour. Occasionally, the optimization also failed because the

Table 5 Algorithm iterations to convergence, two-discipline analytical MDO problem (calculated step size)

	NM	NM—RO	CGM	CGM—RO
Maximum	13	11	96	97
Minimum	2	3	11	12
Mean	4.11	4.78	31.68	38.47
% Convergence	95	81	98	72

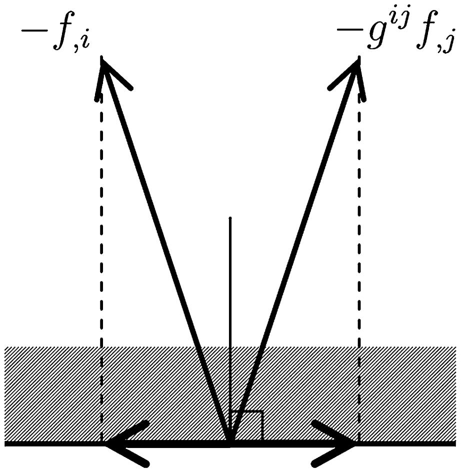

Fig. 5 Projections and direction splitting on design space boundaries

Table 6 Algorithm iterations to convergence, two-discipline analytical MDO problem (line search)

	SDM	NM	NM—RO	CGM
Maximum	20	5	7	34
Minimum	6	2	3	13
Mean	12.65	3.13	4.21	24.05
% Convergence	99	76	68	64

multidisciplinary analysis failed to converge (as with the lone SDM failure in Table 6

As with line search, the Euclidean NM was faster than the Riemannian NM and the Riemannian SDM and CGM completely failed to converge for geodesic search (Table 7). Choosing geodesic search over line search does not affect the initial search direction, and the problem lay in the search direction, not the search method. The geodesic search also produced slightly higher average iterations to convergence than the line search and reduced the convergence percentage in all algorithms save the Riemannian NM.

Table 7 Algorithm iterations to convergence, two-discipline analytical MDO Problem (geodesic search)

	SDM	NM	NM—RO	CGM
Maximum	25	5	10	33
Minimum	6	2	3	14
Mean	13.46	3.38	4.90	24.67
% Convergence	100	65	71	63

Table 8 Satellite design problem disciplines

Discipline	Number
Payload	1
Propulsion	2
Power	3
Attitude determination and control systems (ADCS)	4
Thermal	5
Structures	6

Table 9 Satellite design problem design variables

Symbol	Description	Discipline(s) used
f_{down}	Downlink frequency	1
f_{up}	Uplink frequency	1
P_t	Satellite transmitter power	1
P_{gt}	Ground transmitter power	1
D_{gr}	Ground receiving antenna diameter	1
D_{gt}	Ground transmitting antenna diameter	1
ϵ	Battery energy density	3
D_{st}	Satellite transmitting antenna diameter	1, 4, 5
D_{sr}	Satellite receiving antenna diameter	1, 4, 5

4.3 Satellite design problem

Here, we employ a satellite design problem as described by Mesmer et al. (2013).[1] A full description is also provided in (Bakker 2015). In lieu of that full description, we provide a list of the problem's disciplines in Table 8, a list of design variables in Table 9, a list of state variables in Table 10, and a Design Structure Matrix (DSM) (Browning 2001) in Table 11 to qualitatively indicate the problem's structure.

We use total mass, total cost, and the weighted sum of total mass and Signal-to-Noise Ratio (SNR) (appropriately normalized) as the three different objectives under consideration. Because of the increased computational cost of running these optimizations, however, we limited our runs to 20 iterations and forewent the geodesic search; some of the optimization runs took over an hour, and in our previous problems, the geodesic search could take an order of magnitude longer than the line search. Geodesic search was consistently worse than line search in our previous results, so we felt justified in omitting the geodesic search on this problem.

[1] The problem description is too lengthy to include here, but full details of the satellite design problem are available from the authors upon request.

Table 10 Satellite design problem state variables

Symbol	Description	Discipline of Origin
SNR	Signal-to-noise ratio	1
M_{st}	Satellite transmitting antenna mass	1
M_{sr}	Satellite receiving antenna mass	1
M_{transp}	Transponder mass	1
$M_{payload}$	Payload mass	1
$P_{payload}$	Payload power requirement	1
V_{transp}	Transponder volume	1
M_{prop}	Propellant mass	2
V_{prop}	Propellant tank volume	2
M_{SA}	Solar array mass	3
AS	Solar array area	3
M_{batt}	Battery mass	3
V_{batt}	Battery volume	3
M_{ADCS}	ADCS mass	4
P_{ADCS}	ADCS power requirement	4
V_{RW}	Reaction wheel volume	4
$M_{thermal}$	Thermal system mass	5
$P_{thermal}$	Thermal system power requirement	5
L_s	Satellite bus length	6
r_s	Satellite bus radius	6
t_s	Satellite bus thickness	6
M_{bus}	Satellite bus mass	6

Table 11 Satellite design problem DSM

	1	2	3	4	5	6
1	X	●	●	●	●	●
2		X		●	●	●
3		●	X	●	●	●
4		●	●	X	●	●
5		●	●	●	X	●
6		●		●	●	X

The calculated step size optimizations on the total mass all failed to converge except for the Riemannian SDM and CGM runs, as seen in Table 12; the handful of nonconverged runs for those algorithms failed because they produced points where the multidisciplinary analysis failed to produce a feasible design. Euclidean SDM and CGM, on the other hand, went immediately to the design space boundaries and then began taking very short steps until they ran out of iterations. As for NM, the Riemannian version diverged off into a corner of the design space because the covariant hessian was not positive definite—the negative eigenvalues of the covariant hessian were sometimes an order of magnitude larger than its positive

Table 12 Algorithm iterations to convergence, satellite design problem, total mass (calculated step size)

	SDM—RO	CGM—RO
Maximum	15	16
Minimum	5	5
Mean	9.42	9.70
% Convergence	98	97

eigenvalues—and the Euclidean NM failed to calculate a search direction because the hessian was singular everywhere.

The results in Table 13 for line search optimizations, however, were rather different. Euclidean NM still failed to calculate a search direction, and Riemannian NM still produced ascent search directions because the covariant hessians were not positive definite; SDM and CGM both performed much better. The Euclidean SDM and CGM both required significant computational time but few iterations. The algorithms would begin from an unconstrained point and then reach a new variable bound at each iteration until they finally ended up at the optimum. The nature of the problem enabled Euclidean SDM and CGM to do this, and that is why they always took four iterations to converge.

The total cost objective was more complicated than the total mass function, and this increased complexity is likely why none of the calculate step size algorithms converged. With the line search (results shown in Table 14), Riemannian and Euclidean NM failed for the same respective reasons as they did with the total mass objective, and Euclidean SDM and CGM displayed the same behaviour as before but with seven iterations rather than four. Riemannian SDM failed on nine occasions due to either direction splitting or running out of iterations, but in 80% of the runs, it took nine iterations to converge in a manner similar to SDM, CGM, and Riemannian CGM.

Following these trials, we ran optimizations on the weighted sum of the total mass and SNR. This objective, unlike the previous two, did not have a singular hessian everywhere in the design space. However, it also produced no converged

Table 13 Algorithm iterations to convergence, satellite design problem, total mass (line search)

	SDM	SDM—RO	CGM	CGM—RO
Maximum	4	8	4	8
Minimum	4	5	4	5
Mean	4	6.71	4	6.48
% Convergence	100	100	100	100

Table 14 Algorithm iterations to convergence, satellite design problem, total cost (line search)

	SDM	SDM—RO	CGM	CGM—RO
Maximum	7	20	7	9
Minimum	7	9	7	9
Mean	7	9.62	7	9
% Convergence	100	91	100	100

runs. The hessian was not singular, but it was not positive definite, and as a result, the Riemannian and Euclidean NM diverged for the calculated step size. When using line search, they terminated when the search direction became an ascent direction (within the first few iterations in all NM cases). Euclidean SDM and CGM typically ran out of iterations without converging for both line search and calculated step size because their progress at each iteration was very small. The Riemannian versions using a calculated step size failed in a similar fashion; they failed by direction splitting or running out of iterations when using line search. All of the algorithms occasionally ended up at points with nonconverged multidisciplinary analyses.

That being said, Riemannian SDM and CGM did get very close to converging on a number of occasions: the convergence criterion was that the norm of the projected gradient had to be less than 10^{-3}, and several of the runs would have counted as converged if the criterion had been less strict, as shown in Table 15. In these cases, the difference in objective function value between these runs and the true optimum was less than 0.5%. The algorithms appeared to have failed at this point due to direction splitting. The objective function derivatives in the satellite design problem were, in general, of much greater magnitudes than in the previous problems. Given that, and given the fact that the runs recorded in Table 15 all produced objective function values which were extremely close to the true optimum, it may be that the convergence criterion was more strict than necessary here.

5 Discussion

5.1 Optimization results

Geodesic search was not as effective as line search. We knew ahead of time that it would be more expensive to implement, but if it provided better performance, it might be worth developing a computationally cheaper approximation. However, such an effort would not seem to be justified. Furthermore, although the Riemannian algorithms were sometimes better than the Euclidean ones when using a calculated step size, the Euclidean algorithms were generally better overall when line search or geodesic search was used.

Both Euclidean and Riemannian NM diverged on the satellite design problem objective functions, but they did so in different ways—the covariant hessian was not singular when the hessian was, and in other optimization problems, that could be an advantage. Riemannian SDM and CGM had convergence problems because of direction splitting, however. Their convergence problem was due to the way in which the constraints were handled: penalty functions, for example, might have

Table 15 Percentage of nearly converged runs, satellite design problem, weighted sum objective (line search)

Convergence criterion	SDM—RO	CGM—RO
<0.1	7	9
<0.01	5	7

enabled the algorithms to avoid this problem. If this difficulty were overcome, the Riemannian version could be more effective as the weighted sum satellite design results hinted at. Feasible directions is a standard method and easy to implement, though, and it is particularly convenient for simple bounds on design variables. On other test problems, which we did not present here, we also found that the Riemannian algorithms had greater difficulties with ill-conditioning in $\frac{\partial \mathbf{h}}{\partial \mathbf{y}}$ due to the presence of g^{ij} and Γ^{i}_{jk}.

Given that the Riemannian algorithms are likely to have a higher cost per iteration than the Euclidean algorithms, it would seem that the Euclidean methods are better than the Riemannian ones as we have developed them here. That being said, real-world phenomena, such as nonconvexity, in our satellite design problem made it difficult for any of the algorithms to converge. Final conclusions on the relative performances of Riemannian and Euclidean algorithms would require comparisons with algorithms sophisticated enough to converge reliably on such problems.

5.2 Riemannian optimization and design coupling

We now wish to analyze the algorithms themselves further for any *a priori* performance information which we might glean. In particular, we would like to look at the algorithms' interaction with design coupling and direction splitting. Previously, we identified two different schools of thought on measuring design coupling: one focuses on design structure (i.e. the number and arrangement of variables and functions) in its evaluation of coupling, whereas the other uses design sensitivity information, and they put their respective measures of coupling to different uses (Bakker et al. 2013a); our paper on design coupling provides more detail on each school and their approach to design coupling

Here, we will focus on sensitivity-based coupling—coupling that measures how strongly the state variables depend on the design variables. Sensitivity-based coupling changes throughout the design space, unlike structure-based coupling, but we are now also making a slight innovation by considering how sensitivity changes with *search direction* as well as location. In other words, $\frac{\partial \mathbf{y}}{\partial \mathbf{w}}$ varies as $\mathbf{w}$ changes, but we will also consider how the choice of $\mathbf{d}_{(k)}$ interacts with $\frac{\partial \mathbf{y}}{\partial \mathbf{w}}$.

We begin by stepping out of Riemannian geometry briefly to examine the matrix g_{ij} and its eigenvalues. Consider a vector in the design space $\Delta\mathbf{w}$ of unit length (i.e. $\Delta w^i \Delta w^i = 1$). The (first-order approximation to the) change in the state variables corresponding to this vector is

$$\Delta\mathbf{y} = \frac{\partial \mathbf{y}}{\partial \mathbf{w}} \Delta\mathbf{w} \tag{40}$$

which is equivalent to

$$\Delta y^i = \frac{\partial y^i}{\partial w^j} \Delta w^j \tag{41}$$

Using the knowledge that $g_{ij} = \delta_{ij} + \frac{\partial y^k}{\partial w^i}\frac{\partial y^k}{\partial w^j}$, we can see that

$$\Delta w^i g_{ij} \Delta w^j = \Delta w^i \Delta w^i + \Delta y^i \Delta y^i = 1 + \Delta y^i \Delta y^i \tag{42}$$

In matrix notation, $\mathbf{g} = \mathbf{I} + \left[\frac{\partial \mathbf{y}}{\partial \mathbf{w}}\right]^T \left[\frac{\partial \mathbf{y}}{\partial \mathbf{w}}\right]$, so

$$\Delta \mathbf{w}^T \mathbf{g} \Delta \mathbf{w} = \Delta \mathbf{w}^T \Delta \mathbf{w} + \Delta \mathbf{y}^T \Delta \mathbf{y} = 1 + \Delta \mathbf{y}^T \Delta \mathbf{y} \tag{43}$$

Therefore, the eigenvalues of g_{ij} are all greater than or equal to one. Eigenvectors with eigenvalues equal to one will be in directions of constant $\mathbf{y}$ on the manifold, and eigenvectors with large associated eigenvalues will be in directions of large change in the state variables.

If the eigenvalues of g_{ij} are greater than or equal to one, then the eigenvalues of g^{ij} are less than or equal to one, and thus $\|\mathbf{g}^{-1}\Delta \mathbf{w}\| \leq \|\Delta \mathbf{w}\|$. More specifically, if λ_i is an eigenvalue for g_{ij} and ξ^i is its corresponding (normalized) eigenvector, the eigenvectors are all mutually orthogonal (because g_{ij} is symmetric), and we can represent $\Delta \mathbf{w}$ as $\Delta \mathbf{w} = \sum_i \alpha^i \xi^i$, where $\|\Delta \mathbf{w}\|^2 = \sum_i \alpha_i^2$. With this representation,

$$\mathbf{g}^{-1}\Delta \mathbf{w} = \sum_i \frac{\alpha^i}{\lambda_i}\xi^i \tag{44}$$

$$\|\mathbf{g}^{-1}\Delta \mathbf{w}\|^2 = \sum_i \frac{\alpha_i^2}{\lambda_i^2} \leq \|\Delta \mathbf{w}\|^2 = \sum_i \alpha_i^2 \tag{45}$$

In other words, multiplying a vector (we ignore the distinction between vectors and covectors here) by $\mathbf{g}^{-1}$ will never result in an increase—and typically will result in a decrease – in vector magnitude. Furthermore, even if $\mathbf{g}^{-1}\Delta \mathbf{w}$ is re-normalized so that its norm is the same as the original $\Delta \mathbf{w}$, the new vector will have had its direction changed from the original one, and the new direction will be "less coupled" than the original one: as (44) shows, the multiplication by $\mathbf{g}^{-1}$ shrinks the vector in directions of large change in the state variables proportionate to the degree of that change (represented by λ_i). "Uncoupled directions"—directions of constant $\mathbf{y}$—are left unchanged.

We can see, therefore, that the Riemannian versions of SDM and CGM will always have "flatter" trajectories, with respect to $\mathbf{y}$, than their Euclidean counterparts. Figure 6 shows an example of this: the black path corresponds to $\dot{x}^i = -f_{,i}$, and the light grey path corresponds to $\dot{x}^i = -g^{ij}f_{,j}$. The trend is easier to see in the differential equation rather than the original iterative algorithm; see Bakker et al. (2013b) for more on the use of differential equations in analyzing optimization algorithms.

We can further see that direction splitting will be a problem if a strong change in $\mathbf{y}$ corresponds to a decrease in the objective function along a boundary: $-g^{ij}f_{,j}$ will point further away from $-f_{,i}$ under such circumstances and thus be more likely to be caught on the wrong side of the constraint normal. The satellite design problem's objective functions, for example, depend strongly on the state variables, and this

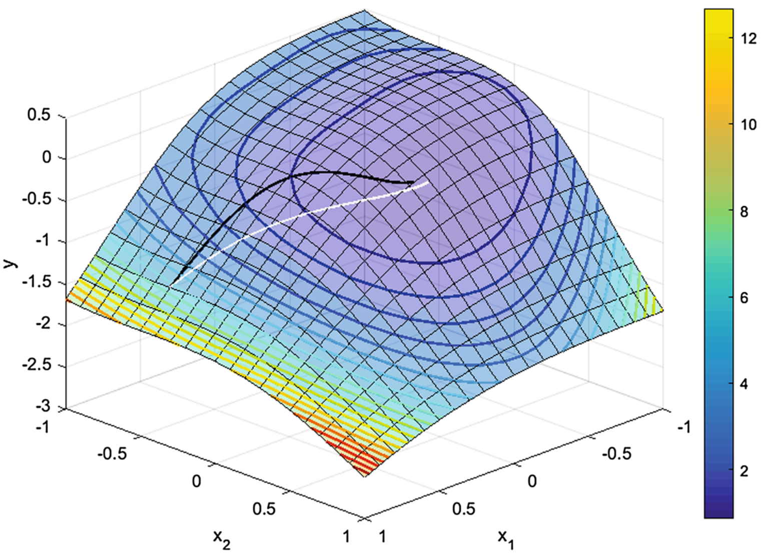

Fig. 6 Riemannian and Euclidean ODE trajectories

may explain the poor performance of the Riemannian algorithms there. Even when the algorithms did not terminate as a result of direction splitting, moving in less coupled directions would have meant less change in the objective function and thus why more iterations would have been required even when the algorithms did converge. Conversely, we can say that Riemannian SDM or CGM could be particularly effective on problems where it would be advantageous for the optimization trajectory to be "flatter" with respect to the state variables.

Although we have not come across particular design problems where this is the case, this may still be valuable information in light of the No Free Lunch (NFL) theorem (Wolpert and Macready 1997). According to the NFL theorem, no one algorithm is better than another when their respective performances are averaged over all possible optimization problems. As such, the key to improved optimization lies in being able to say, *a priori*, when one optimization algorithm will be better than another *on a given problem*. Our analysis of coupling suggests that Riemannian SDM and CGM will perform better than their Euclidean counterparts on problems where it is beneficial to have less variation in the state variables over the course of the optimization. Similarly, the results of that analysis, combined with our experimental results, show that the Euclidean versions will be better on problems where the objective function varies strongly with the state variables and the optimum lies on the design space boundary.

Unfortunately, NM is not amenable to similar analysis. Here, we begin from $f_{;ij}$ instead of g_{ij}, and we can break it up into components $f_{;ij} = f_{,ij} - \Gamma^k_{ij} f_{,k}$. However, the $-\Gamma^k_{ij} f_{,k}$ term, which determines the difference between the Euclidean and Riemannian trajectories, does not have a simple representation in terms of implicit derivatives:

$$-\Gamma^k_{ij} f_{,k} = -g^{km} \frac{\partial y^s}{\partial w^m} \frac{\partial^2 y^s}{\partial w^i \partial w^j} f_{,k} \tag{46}$$

As $f_{,k}$ goes to zero, the Euclidean and Riemannian trajectories will become more and more similar, but beyond that, there is little to say in terms of general statements.

5.3 Riemannian optimization in the broader context of MDO

RO algorithms are simply different versions of standard gradient-based optimization algorithms, and as such, they are similarly general within an MDO context. These algorithms are not part of the decomposition process—that is handled by the architecture. As with standard gradient-based algorithms, RO algorithms can then be applied to the problem after the architecture has been implemented (e.g. MDF with Riemannian CGM, or regular CGM, or another algorithm entirely). In principle, the Riemannian algorithms described and tested here can be applied wherever gradient-based algorithms are already used in MDO; a Riemannian quasi-NM algorithm could be used where a quasi-NM algorithm is currently being used, for example.

The presence of potentially different architectures does add some complexity, however. Riemannian algorithms require a metric tensor and Christoffel symbols (which are themselves produced from basic sensitivity information). As we have shown elsewhere (Bakker et al. 2012), MDF uses the original feasible design manifold, and thus the metric tensor and Christoffel symbols used with MDF are the metric tensor and Christoffel symbols for the original problem. The decompositions involved in MDO architectures alter the original manifold, though. To be more precise, the optimization processes in the different architectures occur on different manifolds—ones which are created by the decomposition process. This means that the metric tensors and Christoffel symbols will be calculated with different formulae than were shown here. However, we have already calculated those metric tensors and Christoffel symbols from basic sensitivity information for six different MDO architectures elsewhere (Bakker et al. 2012). Those calculations could be done, in principle, for any other architecture of interest, as well. In other words, the search direction for the Riemannian SDM will still be

$$d^j = -g^{ij} f_{,i} \tag{47}$$

but the g_{ij} needed to determine g^{ij} will not be calculated using (12). We show how to calculate g_{ij} for those other architectures in (Bakker et al. 2012).

RO algorithms have the same strengths and weaknesses as other gradient-based methods have in comparison to metaheuristic or hybrid methods. The comparative capabilities and limitations of gradient-based, metaheuristic, and hybrid methods, in general, are well-known. The comparative capabilities and limitations of *Riemannian and standard Euclidean methods on MDO problems* are the subject of our inquiry here.

Lest it need be said again, RO algorithms are just like regular gradient-based algorithms. The two differences lie in the modified derivative information (e.g. the covariant derivative vs. a regular gradient) and the modified search path (along geodesics vs. along straight lines). Any differences in behaviour may be traced to those two properties; in all other respects, they are the same. The burden of this paper has been to compare these single-objective, gradient-based methods in terms of their efficiency, measured by the number of iterations to convergence, and their effectiveness, measured by their percent convergence.

RO methods could therefore be incorporated into hybrid optimization methods anywhere in MDO that standard gradient-based algorithms currently are. To be even more general, the modified derivative information used in RO could be used anywhere that regular derivative information currently is, and the modified search direction used in RO could be used in some of the places where line searches currently are; the need for derivative information to calculate geodesics limits geodesic search as compared to line searches. However, implementing RO algorithms on MDO and comparing them with other gradient-based algorithms, as we have done in this paper, is both logically and chronologically prior to any extensions into hybrid optimization methods.

5.4 Recommendations and future work

We have shown our Riemannian optimization algorithms to be less efficient and less effective, on the whole, than their Euclidean counterparts. In order to be sure of Euclidean superiority, however, we would need to develop and test a Riemannian quasi-NM (the various implementations of quasi-NM form a standard against which other gradient-based algorithms are measured), consider different methods for handling inequality constraints, and test on a wider range of MDO problems. We also showed line searches to be better than geodesic searches on our test problems. That being said, the results for any Riemannian algorithm will depend on the metric being used: both the search directions and the geodesics depend on the metric (for a metric-compatible connection). We used the induced metric, but other metrics could prove to be more favourable. It could also be beneficial to look into convexifying the original optimization problem through the imposition of a particular metric. In Sect. 2.5, we mentioned this being done elsewhere, but we did not go into detail about how this might profitably be applied to MDO. Finally, our analysis of the search directions and their relationship to design coupling for Riemannian SDM and CGM was informative, and it might be similarly informative to analyze geodesic trajectories to further explain our performance results.

6 Summary

We began our paper by reviewing the background theory for our MDO differential geometry framework, the additional theory necessary for doing RO, and the RO literature. Given the historical lack of crossover between RO and MDO, we considered this to be particularly important in presenting our results to the MDO community. With this in place, we showed how to apply RO algorithms to MDO through the use of our framework.

We then tested some of these algorithms on MDO problems using our differential geometry framework's induced metric. Our results here showed geodesic search to be consistently less effective than line search, and the RO algorithms were generally not as good as the Euclidean algorithms when using a feasible directions method to handle inequality constraints. That being said, the Riemannian SDM and CGM with calculated step size proved to be a stark exception to this trend when applied to one version of our satellite design problem: they significantly outperformed the Euclidean algorithms in that case. The covariant hessian was also nonsingular in many cases when the regular hessian was singular, so the Riemannian NM was often able to calculate a step direction when the Euclidean NM could not. The nonconvexity we encountered, however, demonstrated the need to compare performance with more robust optimization algorithms (like quasi-NM) and using additional test problems for more conclusive results.

Through our analysis of the algorithms themselves, we showed how the Riemannian SDM and CGM interact with design coupling: they produce steps in directions which are less strongly coupled than their Euclidean counterparts. This has provided us with some ability to predict the performance of those methods relative to each other based on *a priori* information about the relationship between the state variables and the objective function.

Most importantly, we have now set the stage for further investigation into the application of RO to MDO—we have shown both *that* it can be done and *how* it may be done. Our preliminary work here in reviewing, testing, and analyzing RO methods, moreover, has pointed out avenues of future exploration in this area.

Acknowledgments This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Cambridge Trusts. The authors would also like to thank Mr. Hanumanthrao Kannan for his assistance in setting up the satellite design problem.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

Allison JT, Herber DR (2013) Multidisciplinary design optimization of dynamic engineering systems. In: 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA, Boston

Baker CG (2008) Riemannian manifold trust-region methods with applications to eigenproblems. PhD thesis, Florida State University, Tallahassee, Florida

Bakker C (2015) A differential geometry framework for multidisciplinary design optimization. PhD thesis, University of Cambridge, Cambridge, United Kingdom

Bakker C, Parks GT (2015a) Differential geometry tools for multidisciplinary design optimization, part I: Theory. Struct Multidiscip Optim 52:27–38

Bakker C, Parks GT (2015b) Differential geometry tools for multidisciplinary design optimization, part II: Application to QSD. Struct Multidiscip Optim 52:39–53

Bakker C, Parks GT, Jarrett JP (2012) Geometric perspectives on MDO and MDO architectures. In: 12th aviation technology, integration and operations (ATIO) conference and 14th AIAA/ISSMO multidisciplinary analysis and optimization conference, AIAA, Indianapolis

Bakker C, Parks GT, Jarrett JP (2013a) Differential geometry and design coupling in MDO. In: 54th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference, AIAA, Boston

Bakker C, Parks GT, Jarrett JP, (2013b) Optimization algorithms and ODE's in MDO. In: ASME, (2013) design engineering technical conferences and computers and information in engineering conference. ASME, Portland

Barthelemy JFM, Sobieszczanski-Sobieski J (1983) Extrapolation on optimum design based on sensitivity derivatives. AIAA J 21:797–799

Bento GC, Ferreira OP, Liveira PR (2012) Unconstrained steepest descent method for multicriteria optimization on Riemannian manifolds. J Optim Theory Appl 154:88–107

Boothby WM (1986) An introduction to differentiable manifolds and Riemannian geometry. Academic Press Inc, Boston

Browning TR (2001) Applying the design structure matrix to system decomposition and integration problems: A review and new directions. IEEE Trans Eng Manag 48:292–306

Cramer E, Dennis JE Jr, Frank PD, Lewis RM, Shubin GR (1994) Problem formulation for multidisciplinary optimization problems. SIAM J Optim 4:754–776

Da Cruz Neto JX, De Lima LL, Oliviera PR (1998) Geodesic algorithms in Riemannian geometry. Balkan J Geom Appl 3:89–100

Dean EB (1988) Continuous optimization on constraint manifolds. In: TIMS/ORSA joint national meeting, Washington, DC

Ferreira OP, Svaiter BF (2002) Kantorovich's theorem on Newton's method in Riemannian manifolds. J Complex 18:304–329

Gabay D (1982) Minimizing a differentiable function over a differential manifold. J Optim Theory Appl 37:177–219

Ivancevic VG, Ivancevic TT (2007) Applied differential geometry: a modern introduction. World Scientific Publishing Co. Pte. Ltd., Singapore

Ji H (2007) Optimization approaches on smooth manifolds. PhD thesis, Australian National University, Canberra

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: IEEE international conference on neural networks, IEEE, Perth

Li C, Wang J (2008) Newton's method for sections on Riemannian manifolds: generalized covariant α-theory. J Complex 24:423–451

Luenberger DG (1972) The gradient projection method along geodesics. Manag Sci 18:620–631

Martins JRRA, Hwang JT (2012) Review and unification of methods for computing derivatives of multidisciplinary systems. In: 53rd AIAA/ASME/ASCE/ASC structures, structural dynamics, and materials conference, AIAA, Honolulu

Martins JRRA, Lambe AB (2013) Multidisciplinary design optimization: survey of architectures. AIAA J 51:2049–2075

Martins JRRA, Alonso JJ, Reuther JJ (2005) A coupled-adjoint sensitivity analysis method for high-fidelity aero-structural design. Optim Eng 6:33–62

Mesmer BL, Bloebaum CL, Kannan H (2013) Incorporation of value-driven design in multidisciplinary design optimization. In: 10th world congress on structural and multidisciplinary optimization, ISSMO, Orlando

Munier J (2007) Steepest descent method on a Riemannian manifold: the convex case. Balk J Geom Appl 12:98–106

Nishimori Y, (2005) A note on Riemannian optimization methods on the Stiefel and the Grassman manifolds. In, (2005) international symposium on nonlinear theory and its applications, Bruges

Paiva RM, Carvalho ARD, Crawford C, Suleman A (2010) Comparison of surrogate models in a multidisciplinary optimization framework for wing design. AIAA J 48:995–1006

Perez RE, Liu HHT, Behdinan K (2004) Evaluation of multidisciplinary optimization approaches for aircraft conceptual design. In: 10th AIAA/ISSMO multidisciplinary analysis and optimization conference, AIAA, Albany

Potra FA, Rheinboldt WC (1989) Differential-geometric techniques for solving differential algebraic equations. Technical Report ICMA-89-143, University of Pittsburgh, Pittsburgh

Qi C (2011) Numerical optimization methods on Riemannian manifolds. PhD thesis, Florida State University, Tallahassee

Qi C, Gallivan KA, Absil PA (2010) An efficient BFGS algorithm for Riemannian optimization. In: 19^{th} international symposium on mathematical theory of networks and systems, Budapest

Rapcsák T (1989) Minimum problems on differentiable manifolds. Optimization 20:3–13

Rapcsák T (1991) Geodesic convexity in nonlinear optimization. J Optim Theory Appl 69:169–183

Rapcsák T (2002) On minimization on Stiefel manifolds. Eur J Oper Res 143:365–376

Rapcsák T, Thang TT (1995) Nonlinear coordinate representations of smooth optimization problems. J Optim Theory Appl 86:459–489

Rheinboldt WC (1996) Geometric notes on optimization with equality constraints. Appl Math Lett 9:83–87

Ring W, Wirth B (2012) Optimization methods on Riemannian manifolds and their application to shape space. SIAM J Optim 22:596–627

Sellar RS, Batill SM, Renaud JE (1996) Response surface based concurrent subspace optimization for multidisciplinary system design. In: 34^{th} aerospace sciences meeting and exhibit, AIAA, Reno

Smith ST (1994) Optimization techniques on Riemannian manifolds. In: Bloch A (ed) Hamiltonian and gradient flows. Algorithms and control. American Mathematical Society, Providence, pp 113–136

Sobieszczanski-Sobieski J (1990) Sensitivity of complex, internally coupled systems. AIAA J 28:153–160

Szekeres P (2004) A course in modern mathematical physics. Cambridge University Press, Cambridge

Tanabe K (1979a) Continuous Newton-Raphson method for solving an underdetermined system of nonlinear equations. Nonlinear Anal Theory Methods Appl 3:495–503

Tanabe K (1979b) Differential geometric methods in nonlinear programming. In: Lakshikantham V (ed) Applied Nonlinear Analysis. Academic Press, New York, pp 707–720

Tanabe K (1980) A geometric method in nonlinear programming. J Optim Theory Appl 30:181–210

Tanabe K (1982) Differential geometric approach to extended GRG methods with enforced feasibility in nonlinear programming: Global analysis. In: Campbell SL (ed) Recent applications of generalized inverses. Pitman Advanced Publishing Program, Boston, pp 100–137

Tappeta RV, Renaud JE, Rodríguez JF (2000) An interactive multiobjective optimization design strategy for multidisciplinary systems. In: 41st AIAA/ASME/ASCE/ASC structures, structural dynamics, and materials conference, AIAA, Atlanta

The MathWorks Inc (R2010a) Matlab®

Thokala P (2005) Variable complexity optimization. Master's thesis, University of Toronto, Toronto

Udrişte C (1994) Convex functions and optimization methods on Riemannian manifolds. Kluwer Academic Publishers, Dordrecht

Udrişte C (1996a) Riemannian convexity in programming (II). Balk J Geom Appl 1:99–109

Udrişte C (1996b) Sufficient decrease principle on Riemannian manifolds. Balk J Geom Appl 1:111–123

van den Doel K, Ascher U (2012) The chaotic nature of faster gradient descent methods. J Sci Comput 51:560–581

Wang JH (2011) Convergence of Newton's method for sections on Riemannian manifolds. J Optim Theory Appl 148:125–145

Whiting JK (2011) Path optimization using sub-Riemannian manifolds with applications to astrodynamics. PhD thesis, Massachusetts Institute of Technology, Cambridge

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82

Yang Y (2007) Globally convergent optimization algorithms on Riemannian manifolds: Uniform framework for unconstrained and constrained optimization. J Optim Theory Appl 132:245–265