TECHNISCHE UNIVERSITÄT
CHEMNITZ

Technische Universität Chemnitz
Fakultät für Mathematik
Professur Numerische Mathematik
(Partielle Differentialgleichungen)

The Performance of the Riemannian Symmetric Rank-One Method in Julia

Computerpraktikum

submitted by Tom-Christian Riemer
student number 444019
referee PD Dr. Ronny Bergmann

date of submission 31.01.2021

Contents

1 Introduction

Optimization on Riemannian manifolds, also called Riemannian optimization, concerns finding an optimum of a real-valued function f defined over a manifold. It can be thought of as unconstrained optimization on a constrained space.
It is shown how the SR1 update is generalized for operators on the tangent space of a manifold and the performance of a Riemannian SR1 quasi-Newton method implemented in the package `Manopt.jl` (available at `https://manoptjl.org`, Bergmann, 2019) is compared with that of a Riemannian BFGS method.

2 The Euclidean Symmetric Rank-One Quasi-Newton Method

In the Euclidean optimization a key problem is minimizing a real-valued function f over the Euclidean space $\mathbb{R}^n$ ($n \geq 1$), i.e. our focus and efforts are centred on solving

$$\min f(x), \quad x \in \mathbb{R}^n \tag{2.0.1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ is a smooth function. In this chapter we focus on smooth functions, by which we generally mean functions whose second derivatives exist and are continuous or formally $f \in C^2(\mathbb{R}^n)$, unless otherwise stated. Equation (2.0.1) is called a (nonlinear) unconstrained optimization problem. To solve the problem Equation (2.0.1) we want to use quasi-Newton methods, which belong to the class of line search methods. These can be formulated as algorithms where the next iterate is obtained by the iterative update scheme

$$x_{k+1} = x_k + \alpha_k d_k.$$

This means these methods start with an initial point $x_0 \in \mathbb{R}^n$ and produce a sequence of iterates $\{x_k\}_k$ that we hope will converge towards a minimum of Equation (2.0.1). The algorithms follow the strategy of first determining a so-called search direction $d_k \in \mathbb{R}^n$ and then a suitable stepsize $\alpha_k > 0$ is searched for along this search direction d_k.
In quasi-Newton methods,

$$d_k = -{H_k}^{-1}\nabla f(x_k) = -B_k \nabla f(x_k)$$

is used as search direction, where the matrix $H_k \in \mathbb{R}^{n\times n}$ approximates the action of the objective's Hessian $\nabla^2 f(\cdot)$ at the current iterate x_k in the direction of s_k and $B_k = {H_k}^{-1}$. These matrices are not calculated anew in each iteration, but H_k or B_k are updated to new matrices $H_{k+1}, B_{k+1} \in \mathbb{R}^{n\times n}$ using the information about the curvature of the objective function f obtained during the iteration. It is required that matrices H_{k+1} and B_{k+1} generated by the update fulfill the so-called quasi-Newton equation, which reads as

$$H_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k) \quad \text{or} \quad B_{k+1}(\nabla f(x_{k+1}) - \nabla f(x_k)) = x_{k+1} - x_k.$$

For the sake of simplicity, we introduce the notations $s_k = x_{k+1} - x_k \in \mathbb{R}^n$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \in \mathbb{R}^n$, thus we obtain

$$H_{k+1}s_k = y_k \quad \text{or} \quad B_{k+1}y_k = s_k. \tag{2.0.2}$$

The fact that the matrices H_{k+1} and B_{k+1} satisfy the quasi-Newton equation, Equation (2.0.2), is the distinguishing feature of quasi-Newton methods.
The idea now is to find a convenient formula, which requires at most the evaluation of the gradient $\nabla f(\cdot)$ of the objective function at each iterate, for updating the matrix H_k or B_k to a new matrix H_{k+1} or B_{k+1} that satisfies the quasi-Newton equation Equation (2.0.2). Instead of recomputing H_{k+1} or

B_{k+1} from scratch at every iteration, we apply a simple modification that combines the most recently observed information about the objective function with the existing knowledge embedded in H_k or B_k Nocedal, Wright, 2006, p. 139.
There are different update formulae, which of course differentiate the quasi-Newton methods. Probably the best-known method is based on the BFGS update, where the matrix H_{k+1}^{BFGS} or B_{k+1}^{BFGS} differs from its predecessor H_k^{BFGS} or B_k^{BFGS} by a rank-two matrix:

$$H_{k+1}^{\text{BFGS}} = H_k^{\text{BFGS}} + \frac{y_k y_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k} - \frac{H_k^{\text{BFGS}} s_k s_k^{\mathrm{T}} H_k^{\text{BFGS}}}{s_k^{\mathrm{T}} H_k^{\text{BFGS}} s_k} \tag{2.0.3}$$

or

$$B_{k+1}^{\text{BFGS}} = \left(I_{n\times n} - \frac{s_k y_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k}\right) B_k^{\text{BFGS}} \left(I_{n\times n} - \frac{y_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k}\right) + \frac{s_k s_k^{\mathrm{T}}}{s_k^{\mathrm{T}} y_k}. \tag{2.0.4}$$

If the matrix H_k^{BFGS} or B_k^{BFGS} is symmetric positive definite (= $s.p.d.$), and the so-called curvature condition holds, which requires that

$$s_k^{\mathrm{T}} y_k > 0 \tag{2.0.5}$$

is satisfied, then H_{k+1}^{BFGS} or B_{k+1}^{BFGS} is also $s.p.d.$ and satisfies the quasi-Newton equation Equation (2.0.2). But there is also a simpler rank-one update that maintains symmetry of the matrix H_k or B_k and satisfies the quasi-Newton equation Equation (2.0.2). Unlike the rank-two BFGS update, this symmetric rank-one, or short SR1, update does not guarantee that the updated matrix H_{k+1} or B_{k+1} maintains positive definiteness. Nevertheless good numerical results have been obtained with algorithms based on SR1, therefore we now show a possible derivation of this formula, which is so simple that it has been rediscovered a number of times.
A general symmetric rank-one update has the form

$$H_{k+1} = H_k + \sigma v v^{\mathrm{T}},$$

where $v \in \mathbb{R}^n$ and $\sigma \in \{-1, 1\}$. The task now is to determine v and σ so that H_{k+1} satisfies the quasi-Newton equation, Equation (2.0.2). By substituting into Equation (2.0.2), we obtain

$$H_k s_k + [\sigma v^{\mathrm{T}} s_k] v = y_k. \tag{2.0.6}$$

Since $[\sigma v^{\mathrm{T}} s_k]$ is a scalar, v must be a multiple of $y_k - H_k s_k$, i.e. $v = \delta(y_k - H_k s_k)$ for some $\delta \in \mathbb{R}$. By substituting this form of v into Equation (2.0.6), we obtain

$$(y_k - H_k s_k) = \sigma \delta^2 \, [s_k^{\mathrm{T}}(y_k - H_k s_k)](y_k - H_k s_k), \tag{2.0.7}$$

and it is clear that this equation is satisfied if (and only if) we choose the parameters δ and σ to be

$$\sigma = \operatorname{sgn}(s_k^{\mathrm{T}}(y_k - H_k s_k)), \quad \delta = \pm |s_k^{\mathrm{T}}(y_k - H_k s_k)|^{-\frac{1}{2}}.$$

Hence, the only symmetric rank-one update formula that satisfies the quasi-Newton equation, Equation (2.0.2), is given by

$$H_{k+1}^{\text{SR1}} = H_k^{\text{SR1}} + \frac{(y_k - H_k^{\text{SR1}} s_k)(y_k - H_k^{\text{SR1}} s_k)^{\mathrm{T}}}{(y_k - H_k^{\text{SR1}} s_k)^{\mathrm{T}} s_k}. \tag{2.0.8}$$

By applying the Sherman–Morrison-Woodbury formula (cf. **SunYuan:2006**), we obtain the corresponding update formula for the approximation of Hessian inverse $\nabla^2 f(x_{k+1})^{-1}$:

$$B_{k+1}^{\mathrm{SR1}} = B_k^{\mathrm{SR1}} + \frac{(s_k - B_k^{\mathrm{SR1}} y_k)(s_k - B_k^{\mathrm{SR1}} y_k)^{\mathrm{T}}}{(s_k - B_k^{\mathrm{SR1}} y_k)^{\mathrm{T}} y_k}. \tag{2.0.9}$$

It is easy to see in that even if H_k^{SR1} is positive definite, H_{k+1}^{SR1} may not have the same property (this holds also for B_k^{SR1} and B_{k+1}^{SR1}). If and only if $(y_k - H_k^{\mathrm{SR1}} s_k)^{\mathrm{T}} s_k > 0$, the SR1 update retains positive definiteness. However, this condition is difficult to guarantee. This means that H_{k+1}^{SR1} or B_{k+1}^{SR1} may no longer be invertible. Moreover, $d_{k+1} = -{H_{k+1}^{\mathrm{SR1}}}^{-1} \nabla f(x_{k+1}) = -B_{k+1}^{\mathrm{SR1}} \nabla f(x_{k+1})$ is not necessarily a descent direction.
The main drawback of the SR1 update formula is that the denominator in Equation (2.0.8) or Equation (2.0.9) can vanish. In fact, even when the objective function f is convex and quadratic, there may be steps on which there is no symmetric rank-one update that satisfies the quasi-Newton equation Equation (2.0.2). This disadvantage results in serious numerical difficulties, which restrict the applications of it. Nevertheless, the SR1 update formula has the following advantages:

(*i*) A simple safeguard seems to adequately prevent the breakdown of the method and the occurrence of numerical instabilities.

(*ii*) The matrices generated by the SR1 formula, Equation (2.0.8), tend to be good approximations to the true Hessian matrix.

(*iii*) In quasi-Newton methods for constrained problems, it may not be possible to impose Equation (2.0.5), and thus the BFGS update, Equation (2.0.3), is not recommended.

For the vanishing denominator in Equation (2.0.8) we introduce a strategy to prevent a method using the SR1 update from breaking down. It has been observed in practice that it performs well simply by skipping the update if the denominator is small. More specifically, the update Equation (2.0.8) is applied only if

$$|(y_k - H_k^{\mathrm{SR1}} s_k)^{\mathrm{T}} s_k| \geq r \, \|s_k\| \|y_k - H_k^{\mathrm{SR1}} s_k\| \tag{2.0.10}$$

holds, where $r \in (0,1)$ is a small number, e.g. $r = 10^{-8}$. Most implementations of the SR1 update use a skipping rule of this kind. The condition $(y_k - H_k^{\mathrm{SR1}} s_k)^{\mathrm{T}} s_k \approx 0$ occurs infrequently, since it requires certain vectors to be aligned in a specific way. When it occurs, skipping the update appears to have no negative effects on the iteration, since the skipping condition implies that $s_k^{\mathrm{T}} \tilde{G} s_k \approx s_k^{\mathrm{T}} H_k^{\mathrm{SR1}} s_k$, where $\tilde{G}$ is the average Hessian over the last step, meaning that the curvature of H_k^{SR1} along s_k is already correct.
In summary, a general quasi-Newton method using the inverse SR1 update, Equation (2.0.9), is as follows:

Algorithm 1 Inverse SR1 Method

1: Continuously differentiable real-valued function f on $\mathbb{R}^n$, bounded below; initial iterate $x_0 \in \mathbb{R}^n$; initial $s.p.d.$ matrix $B_0^{\mathrm{SR1}} \in \mathbb{R}^{n\times n}$; convergence tolerance $\varepsilon > 0$. Set $k = 0$.
2: **while** $\|\nabla f(x_k)\| > \varepsilon$ **do**
3: Compute the search direction $d_k = -B_k^{\mathrm{SR1}} \nabla f(x_k)$.
4: Determine a suitable stepsize $\alpha_k > 0$.
5: Set $x_{k+1} = x_k + \alpha_k d_k$.
6: Set $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.
7: Compute $B_{k+1}^{\mathrm{SR1}} \in \mathbb{R}^{n\times n}$ by means of Equation (2.0.9).
8: Set $k = k + 1$.
9: **end while**
10: **Return** x_k.

One of the main advantages of the SR1 method is its ability to generate good Hessian approximations. We first consider the case of a quadratic objective function. We have the following statement:

Theorem 2.0.1 (Nocedal, Wright, 2006, Theorem 6.1.). *Suppose that $f\colon \mathbb{R}^n \to \mathbb{R}$ is the strongly convex quadratic function $f(x) = \frac{1}{2}x^{\mathrm{T}}Ax + b^{\mathrm{T}}x$, where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite. Then for any starting point x_0 and any symmetric starting matrix $B_0^{\mathrm{SR1}} \in \mathbb{R}^{n\times n}$, the iterates $\{x_k\}_k$ generated by Algorithm 1 with constant stepsize $\alpha_k = 1$ converge to the minimizer in at most n steps, provided that $(s_k - B_k^{\mathrm{SR1}} y_k)^{\mathrm{T}} y_k \neq 0$ for all k. Moreover, if n steps are performed, and if the search directions d_k are linearly independent, then $B_n^{\mathrm{SR1}} = A^{-1}$.*

The distinguishing feature of the SR1 quasi-Newton method with constant stepsize is its natural quadratic termination, which means that for a quadratic function, the method terminates within n steps.
In the proof it is shown that the SR1 update possesses the so-called hereditary property, i.e.

$$B_k^{\mathrm{SR1}} y_i = s_i, \quad i = 0, \cdots, k-1. \tag{2.0.11}$$

This relation shows that when f is quadratic, the quasi-Newton equation, Equation (2.0.2), is satisfied along all previous search directions, regardless of how the line search is performed. A result like this can be established for the BFGS update, Equation (2.0.4), only under the restrictive assumption that an exact line search is used.
For general nonlinear functions, the SR1 update continues to generate good Hessian approximations under certain conditions:

Theorem 2.0.2 (Nocedal, Wright, 2006, Theorem 6.2.). *Suppose that f is twice continuously differentiable, and that its Hessian is bounded and Lipschitz continuous in a neighborhood of a point x^*. Let $\{x_k\}_k$ be any sequence of iterates such that $x_k \to x^*$ for some $x^* \in \mathbb{R}^n$. Suppose in addition that the inequality Equation (2.0.10) holds for all k, for some $r \in (0,1)$, and that the steps s_k are uniformly linearly independent. Then the matrices H_k^{SR1} generated by Equation (2.0.8) satisfy*

$$\lim_{k\to\infty} \|H_k^{\mathrm{SR1}} - \nabla^2 f(x^*)\| = 0.$$

"Uniformly linearly independent steps" means, that the steps do not tend to fall in a subspace of a dimension less than n. This assumption is usually, but not always, satisfied in practice Nocedal, Wright, 2006, p. 144-149.
The convergence properties of the SR1 quasi-Newton method are not as well understood as those of the BFGS method. No global results or local superlinear results have been established, except the mentioned results for quadratic functions.

3 The Riemannian Symmetric Rank-One Quasi-Newton Method

3.1 Preliminaries

From now on we consider Riemannian optimization problems, which consider finding an optimum of a real-valued function f defined on a Riemannian manifold $\mathcal{M}$, i.e.

$$\min f(x), \quad x \in \mathcal{M}.$$

From now on we assume that $\mathcal{M}$ is a n-dimensional geodesically complete Riemannian manifold. We further assume that the manifold $\mathcal{M}$ is embedded in a real-valued space (e.g. $\mathcal{M} \subseteq \mathbb{R}^m$) and connected. Further we assume that $f\colon \mathcal{M} \to \mathbb{R}$ is a twice continuously differentiable function, i.e. $f \in C^2(\mathcal{M})$. Riemannian quasi-Newton methods belong to the class of Riemannian line search methods in which, similar to their Euclidean counterparts, at first a tangent vector $\eta_k \in \mathcal{T}_{x_k}\mathcal{M}$ as search direction is determined and then a suitable stepsize $\alpha_k > 0$ is searched for along a curve $\gamma(\alpha) = \mathrm{retr}_{x_k}(\alpha\eta_k)$ on the manifold $\mathcal{M}$, which is defined by a chosen retraction $\mathrm{retr}\colon \mathcal{TM} \to \mathcal{M}$ (see Absil, Mahony, Sepulchre, 2007, Definition 4.1.1). Therefore the choice of a computationally efficient retraction is an important decision in the design of high-performance numerical algorithms on manifolds Absil, Mahony, Sepulchre, 2007, p. 54. We further assume that the exponential map $\exp\colon \mathcal{TM} \to \mathcal{M}$ (see Absil, Mahony, Sepulchre, 2007, p. 102-103), which is a retraction, is always available on the manifold $\mathcal{M}$. Therefore, from now on we use the following iterative update scheme:

$$x_{k+1} = \exp_{x_k}(\alpha_k \eta_k).$$

In Riemannian quasi-Newton methods, the search direction $\eta_k \in \mathcal{T}_{x_k}\mathcal{M}$ is obtained by

$$\eta_k = -\mathcal{H}_k^{-1}[\mathrm{grad}\, f(x_k)] = -\mathcal{B}_k[\mathrm{grad}\, f(x_k)],$$

where $\mathrm{grad}\, f(x_k)$ is the Riemannian gradient (see Absil, Mahony, Sepulchre, 2007, p. 46) of the objective function f at $x_k \in \mathcal{M}$ and $\mathcal{H}_k, \mathcal{B}_k\colon \mathcal{T}_{x_k}\mathcal{M} \to \mathcal{T}_{x_k}\mathcal{M}$ are linear self-adjoint operators and $\mathcal{H}_k^{-1}[\cdot] = \mathcal{B}_k[\cdot]$. Here $\mathcal{H}_k[\cdot]$ tries to approximate the action of the Riemannian Hessian $\mathrm{Hess}\, f(x_k)[\cdot]$ (see Absil, Mahony, Sepulchre, 2007, Definition 5.5.1) and $\mathcal{B}_k[\cdot]$ approximates the action of $\mathrm{Hess}\, f(x_k)^{-1}[\cdot]$. At the end of each iteration, these operators are updated to new operators $\mathcal{H}_{k+1}, \mathcal{B}_{k+1}\colon \mathcal{T}_{x_{k+1}}\mathcal{M} \to \mathcal{T}_{x_{k+1}}\mathcal{M}$ on the upcoming tangent space $\mathcal{T}_{x_{k+1}}\mathcal{M}$. It is required that these satisfy some form of the Riemannian quasi-Newton equation. For this we need the parallel transport $\mathrm{P}_{y\leftarrow x}\colon \mathcal{T}_x\mathcal{M} \to \mathcal{T}_y\mathcal{M}$ (see Absil, Mahony, Sepulchre, 2007, p. 169-170), which is a vector transport (see Absil, Mahony, Sepulchre, 2007, Definition 8.1.1) to which the exponential map $\exp$ is the associated retraction. By introducing the tangent vector $s_k = \mathrm{P}_{x_k,\alpha_k\eta_k}(\alpha_k\eta_k) \in \mathcal{T}_{x_{k+1}}\mathcal{M}$, which is the transported search direction times the

stepsize and thus represents the connection between the two iterates x_k and x_{k+1}, and the tangent vector $y_k = \operatorname{grad} f(x_{k+1}) - \mathrm{P}_{x_k,\alpha_k\eta_k}(\operatorname{grad} f(x_k)) \in \mathcal{T}_{x_{k+1}}\mathcal{M}$, which represents the difference of the gradients in $\mathcal{T}_{x_{k+1}}\mathcal{M}$, the Riemannian quasi-Newton equation is given by

$$\mathcal{H}_{k+1}[s_k] = y_k \quad \text{or} \quad \mathcal{B}_{k+1}[y_k] = s_k.$$

As in the Euclidean case, there are different update formulae for operators so that they satisfy the Riemannian quasi-Newton equation. In (almost) all of them, the previously collected information in $\mathcal{H}_k[\cdot]$ is taken over, but this is a operator on the tangent space $\mathcal{T}_{x_k}\mathcal{M}$. To overcome this obstacle, we transport it into the appropriate tangent space $\mathcal{T}_{x_{k+1}}\mathcal{M}$, i.e.

$$\widetilde{\mathcal{H}}_k = \mathrm{P}_{x_{k+1}\leftarrow x_k} \circ \mathcal{H}_k \circ \mathrm{P}_{x_k\leftarrow x_{k+1}} \colon \mathcal{T}_{x_{k+1}}\mathcal{M} \to \mathcal{T}_{x_{k+1}}\mathcal{M}.$$

We use the same method and notation for $\widetilde{\mathcal{B}}_k \colon \mathcal{T}_{x_{k+1}}\mathcal{M} \to \mathcal{T}_{x_{k+1}}\mathcal{M}$. The idea now is to find a convenient update formula for operators, which can be written with $\widetilde{\mathcal{H}}_k$ or $\widetilde{\mathcal{B}}_k$ and y_k, s_k. To be able to create rank-one operators, we introduce the musical isomorphism $\flat \colon \mathcal{T}_{x_{k+1}}\mathcal{M} \ni \eta_{x_{k+1}} \mapsto \eta^{\flat}_{x_{k+1}} \in \mathcal{T}^*_{x_{k+1}}\mathcal{M}$ (see Bergmann et al., 2020, p. 6). Put simply, it means: $\eta^{\flat}_{x_{k+1}} \in \mathcal{T}^*_{x_{k+1}}\mathcal{M}$ represents the flat of $\eta_{x_{k+1}} \in \mathcal{T}_{x_{k+1}}\mathcal{M}$, i.e., $\eta^{\flat}_{x_{k+1}} \colon \mathcal{T}_{x_{k+1}}\mathcal{M} \to \mathbb{R},\ \xi_{x_{k+1}} \mapsto \eta^{\flat}_{x_{k+1}}[\xi_{x_{k+1}}] = g_{x_{k+1}}(\eta_{x_{k+1}}, \xi_{x_{k+1}})$. This generalizes the notion of the transpose. As noted earlier, there are several quasi-Newton update formulae for operators that have been generalized from the Euclidean case to the Riemannian setup. The generalization of the efficient BFGS update is given by

$$\mathcal{H}^{RBFGS}_{k+1}[\cdot] = \widetilde{\mathcal{H}}^{RBFGS}_{k}[\cdot] + \frac{y_k y_k^{\flat}[\cdot]}{s_k^{\flat}[y_k]} - \frac{\widetilde{\mathcal{H}}^{RBFGS}_{k}[s_k] s_k^{\flat}(\widetilde{\mathcal{H}}^{RBFGS}_{k}[\cdot])}{s_k^{\flat}(\widetilde{\mathcal{H}}^{RBFGS}_{k}[s_k])} \tag{3.1.1}$$

and the update for the approximation of the Hessian inverse $\operatorname{Hess} f(x_{k+1})^{-1}[\cdot]$ is given by

$$\mathcal{B}^{RBFGS}_{k+1}[\cdot] = \Big(\mathrm{id}_{\mathcal{T}_{x_{k+1}}\mathcal{M}}[\cdot] - \frac{s_k y_k^{\flat}[\cdot]}{s_k^{\flat}[y_k]}\Big)\widetilde{\mathcal{B}}^{RBFGS}_{k}[\cdot]\Big(\mathrm{id}_{\mathcal{T}_{x_{k+1}}\mathcal{M}}[\cdot] - \frac{y_k s_k^{\flat}[\cdot]}{s_k^{\flat}[y_k]}\Big) + \frac{s_k s_k^{\flat}[\cdot]}{s_k^{\flat}[y_k]}. \tag{3.1.2}$$

These update formulae produce positive definite self-adjoint operators $\mathcal{H}^{RBFGS}_{k+1}[\cdot]$ or $\mathcal{B}^{RBFGS}_{k+1}[\cdot]$, if the operator $\mathcal{H}^{RBFGS}_{k}[\cdot]$ or $\mathcal{B}^{RBFGS}_{k}[\cdot]$ is positive definite self-adjoint, an isometric vector transport (which is the parallel transport) is used in the definitions of s_k, y_k, $\widetilde{\mathcal{H}}^{RBFGS}_{k}[\cdot]$, $\widetilde{\mathcal{B}}^{RBFGS}_{k}[\cdot]$, and the Riemannian curvature condition holds, which requires:

$$g_{x_{k+1}}(s_k, y_k) > 0. \tag{3.1.3}$$

In the setup of this work, the Riemannian curvature condition, Equation (3.1.3), is satisfied if a stepsize $\alpha_k > 0$ is chosen that fulfills the following version of the generalization of the Wolfe conditions:

$$f(\exp_{x_k}(\alpha_k\eta_k)) \le f(x_k) + c_1\alpha_k g_{x_k}(\operatorname{grad} f(x_k), \eta_k) \tag{3.1.4}$$

and

$$g_{\exp_{x_k}(\alpha_k\eta_k)}(\operatorname{grad} f(\exp_{x_k}(\alpha_k\eta_k)), \mathrm{P}_{x_k,\alpha_k\eta_k}(\eta_k)) \ge c_2 g_{x_k}(\operatorname{grad} f(x_k), \eta_k). \tag{3.1.5}$$

with $0 < c_1 < c_2 < 1$.

3.2 Riemannian Symmetric Rank-One Update

The direct SR1 update for operators on tangent spaces of the manifold which approximates the action of the Riemannian Hessian $\operatorname{Hess} f(x_{k+1})[\cdot]$ is given by

$$\mathcal{H}_{k+1}^{\mathrm{RSR1}}[\cdot] = \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[\cdot] + \frac{(y_k - \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[s_k])(y_k - \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[s_k])^\flat[\cdot]}{(y_k - \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[s_k])^\flat[s_k]} \tag{3.2.1}$$

Huang, 2013, p. 18, and it is not difficult to derive the update formula for the approximation of the Hessian inverse $\operatorname{Hess} f(x_{k+1})^{-1}[\cdot]$, which is given by

$$\mathcal{B}_{k+1}^{\mathrm{RSR1}}[\cdot] = \widetilde{\mathcal{B}}_k^{\mathrm{SR1}}[\cdot] + \frac{(s_k - \widetilde{\mathcal{B}}_k^{\mathrm{SR1}}[y_k])(s_k - \widetilde{\mathcal{B}}_k^{\mathrm{SR1}}[y_k])^\flat[\cdot]}{(s_k - \widetilde{\mathcal{B}}_k^{\mathrm{SR1}}[y_k])^\flat[y_k]}. \tag{3.2.2}$$

We see immediately that both Equation (3.2.1) and Equation (3.2.2) create self-adjoint operators if $\widetilde{\mathcal{H}}_k^{\mathrm{SR1}}$ or $\widetilde{\mathcal{B}}_k^{\mathrm{SR1}}$ are self-adjoint (which are self-adjoint if $\mathcal{H}_k^{\mathrm{SR1}}$ or $\mathcal{B}_k^{\mathrm{SR1}}$ are self-adjoint, since the parallel transport P is an isometric vector transport). Furthermore, we see that the update formula Equation (3.2.1) produces a positive definite operator if $\widetilde{\mathcal{H}}_k^{\mathrm{SR1}}$ is positive definite (which is positive definite if $\mathcal{H}_k^{\mathrm{SR1}}$ is positive definite, since the parallel transport P is an isometric vector transport) and $(y_k - \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[s_k])^\flat[s_k] = g_{x_{k+1}}(y_k - \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[s_k], s_k) > 0$ holds (the same of course for Equation (3.2.2)). This shows that, just as in the Euclidean case, the search direction $\eta_k \in \mathcal{T}_{x_k}\mathcal{M}$ does not always have to be a descent direction and that there is also the possibility that this method can break down if the numerator in Equation (3.2.1) is equal to zero. In order to overcome this, Equation (2.0.10) can be transferred to the Riemannian setup. This means that Equation (3.2.1) is only executed if

$$|g_{x_{k+1}}(y_k - \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[s_k], s_k)| \geq r \, \|s_k\|_{x_{k+1}} \|y_k - \widetilde{\mathcal{H}}_k^{\mathrm{SR1}}[s_k]\|_{x_{k+1}} \tag{3.2.3}$$

holds, where $g_{x_{k+1}}(\cdot,\cdot)\colon \mathcal{T}_{x_{k+1}}\mathcal{M} \times \mathcal{T}_{x_{k+1}}\mathcal{M} \to \mathbb{R}$ is the inner product on $\mathcal{T}_{x_{k+1}}\mathcal{M}$ (see Bergmann et al., 2020, p. 6) and $\|\cdot\|_{x_{k+1}} = \sqrt{g_{x_{k+1}}(\cdot,\cdot)}$ denotes the resulting norm on $\mathcal{T}_{x_{k+1}}\mathcal{M}$. The factor r is again in the interval $(0,1)$.
In summary, a Riemannian inverse SR1 quasi-Newton method, short inverse RSR1 method, which uses the approximation of the Hessian inverse $\operatorname{Hess} f(x_{k+1})^{-1}[\cdot]$, is as follows:

Algorithm 2 Inverse SR1 Method

1: Continuously differentiable real-valued function f on $\mathcal{M}$, bounded below; initial iterate $x_0 \in \mathcal{M}$; initial positive definite and self-adjoint operator $\mathcal{B}_0^{\mathrm{SR1}}\colon\ \mathcal{T}_{x_0}\mathcal{M} \to \mathcal{T}_{x_0}\mathcal{M}$; convergence tolerance $\varepsilon > 0$. Set $k = 0$.
2: **while** $\|\nabla f(x_k)\|_{x_k} > \varepsilon$ **do**
3: Compute the search direction $\eta_k = -\mathcal{B}_k^{\mathrm{SR1}}[\operatorname{grad} f(x_k)]$.
4: Determine a suitable stepsize $\alpha_k > 0$.
5: Set $x_{k+1} = \exp_{x_k}(\alpha_k \eta_k)$.
6: Set $\widetilde{\mathcal{B}}_k^{\mathrm{SR1}} = \mathrm{P}_{x_k,\alpha_k\eta_k} \circ \mathcal{B}_k^{\mathrm{SR1}} \circ \mathrm{P}_{x_k,\alpha_k\eta_k}{}^{-1}$, $s_k = \mathrm{P}_{x_k,\alpha_k\eta_k}(\alpha_k\eta_k)$ and $y_k = \operatorname{grad} f(x_{k+1}) - \mathrm{P}_{x_k,\alpha_k\eta_k}(\operatorname{grad} f(x_k))$.
7: Compute $\mathcal{B}_{k+1}^{\mathrm{SR1}}\colon\ \mathcal{T}_{x_{k+1}}\mathcal{M} \to \mathcal{T}_{x_{k+1}}\mathcal{M}$ by means of Equation (3.2.2).
8: Set $k = k + 1$.
9: **end while**
10: **Return** x_k.

The statements about the termination of quadratic functions and the generation of good approximations known from the Euclidean case have not yet been transferred to the Riemannian setup. Nevertheless, we can make the bold assumption that this is feasible under reasonable assumptions.

4 Numerics

To test the performance of the Riemannian SR1 quasi-Newton method, implemented in the package `Manopt.jl`, we consider the Rayleigh quotient minimization problem on the sphere $\mathbb{S}^{n-1}$. For a symmetric matrix $A \in \mathbb{R}^{n\times n}$, the unit-norm eigenvector, $v \in \mathbb{R}^n$, corresponding to the smallest eigenvalue, defines the two global minima, $\pm v$, of the Rayleigh quotient

$$\begin{aligned} f\colon\ \mathbb{S}^{n-1} &\to \mathbb{R} \\ x &\mapsto x^{\mathrm{T}}Ax \end{aligned} \tag{4.0.1}$$

with its gradient

$$\operatorname{grad} f(x) = 2(Ax - xx^{\mathrm{T}}Ax).$$

To apply the RSR1 method, implemented in `Manopt.jl`, to the optimization problem defined by the cost function Equation (4.0.1) for $n = 100$, Listing 4.1 must be executed in Julia. The problem is defined by setting `A_symm = ( A + A' )/ 2`, where the elements of `A` are drawn from the standard normal distribution using Julia's `randn(n,n)` with seed 42. With `random_point(M)`, a random point is created on the given manifold `M`. The stopping criterion requires to abort the method, that either the norm of the gradient is less than 10^{-6} or 1000 iterations have been run.

Listing 4.1: The Rayleigh quotient minimization experiment in Julia for $n = 100$.

```
using Manopt, Manifolds, Random
Random.seed!(42)
n = 100
A = randn(n,n)
A_symm = ( A + A' ) / 2
M = Sphere(n - 1)
F(X::Array{Float64,1}) = X' * A_symm * X
grad_F(X::Array{Float64,1}) = 2 * ( A_symm * X - X * X' * A_symm *
   X )
x = random_point(M)

quasi_Newton(M,
    F,
    grad_F,
    x;
    memory_size = -1,
    direction_update = SR1(),
    stopping_criterion = StopWhenAny(
        StopAfterIteration(max(1000)),
```

```
            StopWhenGradientNormLess(10^(-6)),
        )
```

To test the performance, we execute an experiment in Julia, where Listing 4.1 depending on the parameter `n` is used. We compare the average number of iterations per call and the average time needed per call of the RSR1, the inverse RSR1, the RBFGS and the inverse RBFGS method, which are all implemented in the package `Manopt.jl`. All methods follow the sequence of Algorithm 2, only at the end the corresponding operator update of the method is used, i.e. the RBFGS method uses Equation (3.1.1), the inverse RBFGS method uses Equation (3.1.2), the RSR1 method uses Equation (3.2.1) and the inverse RSR1 method uses Equation (3.2.2). In all methods, a stepsize $\alpha_k > 0$ is determined in each iteration that fulfills Equation (3.1.4) and Equation (3.1.5), with $c_1 = 10^{-4}$ and $c_2 = 0.999$. The numerical experiment is implemented in the toolbox `Manopt.jl`. It runs on a Lenovo ThinkPad L490, 64 bit Windows system, 1.8 Ghz Intel Core i7-8565U, 32 GB RAM, with Julia 1.5.2.

Manifold	$\mathbb{S}^{99}$		$\mathbb{S}^{299}$	
	Time	Iterations	Time	Iterations
RBFGS	248.525 ms	79	2.318 s	88
Inverse RBFGS	97.434 ms	79	888.005 ms	88
RSR1	450.167 ms	105	5.630 s	295
Inverse RSR1	336.362 ms	115	3.737 s	309

Table 4.0.1: Comparison of the quasi-Newton methods.

Chapter 4 contains the results of the various quasi-Newton methods on $\mathbb{S}^{99}$ and $\mathbb{S}^{299}$. For the time measurement the package BenchmarkTools.jl was used. The time was measured with a benchmark of 10 samples and 1 evaluation per sample using a random point on the manifold. The iterations are the average of the measured iterations from 10 random runs.
We note that the RSR1 method was aborted twice on the manifold $\mathbb{S}^{99}$ and three times on the manifold $\mathbb{S}^{299}$, because the norm of the gradient of f was not smaller than 10^{-6} until 1000 iterations. These results were not taken into account in the calculation of the average iterations and are not considered valid. A total of 10 runs were made in which the RSR1 method stopped due to the norm of the gradient. This occurrence is not surprising, since no convergence statements are known about the Riemannian SR1 quasi-Newton method and one cannot generally assume fast local convergence.
In the results from Chapter 4, it is immediately apparent that the inverse RBFGS method is the clear winner. Both the RBFGS method and the inverse RBFGS method need less time and fewer iterations for both manifolds than the RSR1 and the inverse RSR1 method. This is not very surprising as the BFGS method is considered the most efficient quasi-Newton method in the Euclidean case.
The lower number of iterations can be explained by the fact that by choosing a stepsize $\alpha_k > 0$ that satisfies the Wolfe conditions, the Riemannian curvature condition, Equation (3.1.3), is satisfied and thus both RBFGS methods consistently produce positive definite operators. This in turn leads to the fact that η_k is a descent direction in each iteration, which leads to a steady descent of the objective function f. This, of course, favours faster convergence. In the RSR1 and the inverse RSR1 method, this property is not given, which seems to be important, since on $\mathbb{S}^{299}$ the RSR1 and the inverse RSR1

method need three times more iterations than the RBFGS and inverse RBFGS method.
The large time difference can be attributed to the higher number of iterations on the one hand and on the other hand it can be more difficult to find a suitable stepsize if η_k is not a descent direction, which can be the case with the RSR1 and the inverse RSR1 method.
Interestingly, both RBFGS methods have the same average number of iterations, but the inverse RBFGS method takes less than half the time. This is most likely due to the solving of the system $\mathcal{H}_k^{RBFGS}[\eta_k] = \operatorname{grad} f(x_k)$, which can be (very) costly and occurs in every iteration of the RBFGS method. The same phenomenon can be observed with the RSR1 and the inverse RSR1 method. The inverse RSR1 method even needs slightly more iterations, but less time.

5 Conclusion

This work summarizes how the Symmetric Rank-One update formula is generalized for the Riemannian setup. An implementation of a Riemannian SR1 quasi-Newton method and an inverse Riemannian SR1 quasi-Newton method was undertaken in the package `Manopt.jl`. An experiment was conducted in the programming language Julia in which the RSR1 method was found to converge with a reasonable number of iterations and in a reasonable time under certain conditions. But this method is not nearly as efficient as the RBFGS method, which is based on a rank-two update. Nevertheless, there is further interest in investigating the Riemannian SR1 update, as it can also be used in other algorithms such as the Riemannian trust region method.

Literature

Absil, P.-A.; R. Mahony; R. Sepulchre (2007). *Optimization Algorithms on Matrix Manifolds*. USA: Princeton University Press.

Bergmann, R. (2019). *Manopt.jl - Optimization on manifolds in Julia*. URL: https://ronnybergmann.net/projects/manoptjl.html.

Bergmann, R.; R. Herzog; M. Louzeiro Silva; D. Tenbrinck; J. Vidal-Núñez (2020). *Fenchel duality theory and a primal-dual algorithm on Riemannian manifolds*. arXiv: 1908.02022.

Huang, W. (2013). "Optimization Algorithms on Riemannian Manifolds with Applications". URL: http://purl.flvc.org/fsu/fd/FSU_migr_etd-8809.

Nocedal, J.; S. J. Wright (2006). *Numerical Optimization*. Second Edition. Springer. DOI: 10.1007/978-0-387-40065-5.