Xiamen University
Florida State University
Université catholique de Louvain
Northeastern University

ROPTLIB: Riemannian Manifold Optimization Library

User Manual

Author:
Wen Huang
Collaborators:
Kyle A. Gallivan
P.-A. Absil
Paul Hand

Affiliations:
Xiamen University

Florida State University
Université catholique de Louvain
Northeastern University

August 10, 2020

Contents

1 Introduction

The Riemannian manifold optimization library ROPTLIB is used to optimize a cost function defined on a Riemannian manifold. State of the art algorithms, shown in Table 1, are included. The package is written in C++ using the standard linear algebra libraries BLAS and LAPACK. It can be used in a C++ environment, a Matlab environment and a Julia environment. Users only need to provide the cost function, the gradient function, and the action of the Hessian (if a Newton method is used) in C++, Matlab or Julia. The package optimizes a given cost function using some parameters specified by users, e.g., the domain manifold, algorithm, stopping criterion.

Table 1: Riemannian algorithms in ROPTLIB. Note that ManPG and AManPG requires the ambient space of the manifold to be $\mathbb{R}^n$.

Name	literature	used objects	smoothness of the cost function
Riemannian trust-region Newton (RTRNewton)	[ABG07]	Fun/Grad/Hess	Smooth
Riemannian trust-region symmetric rank-one update (RTRSR1)	[HAG15]	Fun/Grad	Smooth
Limited-memory RTRSR1 (LRTRSR1)	[HAG15]	Fun/Grad	Smooth
Riemannian trust-region steepest descent (RTRSD)	[AMS08]	Fun/Grad	Smooth
Riemannian line-search Newton (RNewton)	[AMS08]	Fun/Grad/Hess	Smooth
Riemannian Broyden family (RBroydenFamily)	[HGA15]	Fun/Grad	Smooth
Riemannian BFGS (RWRBFGS and RBFGS)	[RW12] [HGA15]	Fun/Grad	Smooth
Limited-memory RBFGS (LRBFGS)	[HGA15]	Fun/Grad	Smooth
Riemannian conjugate gradients (RCG)	[NW06] [AMS08] [SI13]	Fun/Grad	Smooth
Riemannian steepest descent (RSD)	[AMS08]	Fun/Grad	Smooth
Riemannian gradient sampling (RGS)	[Hua13] [HU16]	Fun/Grad	L-continuous
Subgradient Riemannian (L)BFGS ((L)RBFGSLPSub)	[HHY18]	Fun/Grad	L-continuous
Riemannian Proximal gradient method (ManPG)	[CMMCSZ20, HW19]	Fun/Grad	$f + g$
Accelerated Riemannian proximal gradient method (AManPG)	[HW19]	Fun/Grad	$f + g$

2 Installation and First Example

The package has been tested on Windows 10, Ubuntu 18.04.4 and MacOS Mojave 10.14.6 when the code is compiled in Matlab and C++ environment alone. It has been tested on Ubuntu 18.04.4 and MacOS Mojave 10.14.6 when the code is compiled in Julia[1].

ROPTLIB can be installed with or without the fast fourier transform package FFTW. If one needs to install ROPTLIB with FFTW, then the command `#define ROPTLIB_WITH_FFTW` in `./Others/def.h` needs to be uncommented out.

2.1 Compiling in Matlab

The command "mex -setup" in Matlab sets up the MEX environment properly. Users are not required to install BLAS and LAPACK in this case since those libraries are included in Matlab.

[1]ROPTLIB on an Ubuntu system needs dependencies. When installing ROPTLIB in Ubuntu, if "gl.h" is missing, using the command "sudo apt-get install mesa-common-dev"; if "-lgl" is not defined, then use the command "sudo apt-get install build-essential libgl1-mesa-dev".

Windows: The C++ compiler in windows 7 SDK is used in our tests. It can be downloaded from `https://www.microsoft.com/en-us/download/details.aspx?id=8279`. In the installation of windows 7 SDK, one error may occur. If the error occurs, then open ”program and feature” and uninstall Microsoft Visual C++ 2010 x64 Redistributable-10.0.40219 and uninstall Microsoft Visual C++ 2010 x86 Redistributable-10.0.40219. If Net Framework 4 can not be installed, then one can uninstall Net Framework 4.6 to fix this problem.

Install FFTW by following the instruction at `http://www.fftw.org/install/windows.html`. The following is the steps that are used during the test.

1. Download 64-bit version from above webpage and unzip the file.
2. Find the lib.exe file. On my computer, the directory is
   `C:\Program Files (x86)\Microsoft Visual Studio 14.0\VC\bin\`.
3. Open cmd window and go to the directory of the downloaded file. Use commands:
   `PATH\lib /machine:x64 /def:libfftw3-3.def`
   `PATH\lib /machine:x64 /def:libfftw3f-3.def`
   `PATH\lib /machine:x64 /def:libfftw3l-3.def` to generate libraries, where PATH denotes the directory of the lib.exe.
4. Copy .lib and .dll files to the folder: `ROPTLIB\BinaryFiles`.
5. See paragraph “Tests in Matlab for all platforms” for testing.

Ubuntu: The steps to set up in terminal are:

1. Install BLAS, LAPACK and FFTW:
   ```
   sudo apt-get install build-essential
   sudo apt-get install liblapack*
   sudo apt-get install libblas*
   sudo apt-get install libfftw3*
   ```
2. Download the corresponding version of ROPTLIB and go the the directory of ROPTLIB.
3. See paragraph “Tests in Matlab for all platforms” for testing.

MAC: The steps to set up in Matlab (tested for Matlab R2019a) are:

1. Download Xcode from official webpage or App Store and install it.
2. Download FFTW (fftw-3.3.8.tar.gz in our test) from `http://www.fftw.org/download.html`. Unzip the file.
3. Install FFTW by following the instruction in `http://www.fftw.org/fftw3_doc/Installation-on-Unix.html#Installation-on-Unix`. In our test, the steps are
   (a) `./configure`
   (b) `make`
   (c) `make install`

4. Download the corresponding version of ROPTLIB and go the the directory of ROPTLIB.

5. If you can error "xcrun: error: SDK "macosx******" cannot be located", then use command "sudo xcode-select –switch /Applications/Xcode.app/" in terminal to fix this problem.

6. See paragraph "Tests in Matlab for all platforms" for testing.

Tests in Matlab for all platforms: To compile ROPTLIB and run a test example, first go to the root directory, i.e., `/ROPTLIB/`. Run "GenerateMyMex.m". A file called "MyMex.m" will be created or updated. By default, ROPTLIB is not link to FFTW library and therefore fast FFT is not supported. If one needs to use FFTW library, then make sure FFTW has been installed properly and use command "GenerateMyMex(1)" to generate MyMex file. The MyMex file is used to compile the package. The command "MyMex" followed by the name of any test files in `/ROPTLIB/test/` compiles the test file. For example, run "MyMex TestStieBrockett" to test the Brockett cost function on the Stiefel manifold [AMS08, Section 4.8]. A binary file "TestStieBrockett.***" will be generated in `/ROPTLIB/test/BinaryFiles/` , where the suffix *** depends on the systems. Finally, use the command "TestStieBrockett" to run the binary file. The commands and results can be found in Listing 1. The explanations of the notation can be found in Appendix B.

Listing 1: Test code

```
>> GenerateMyMex
Generate MyMex.m file...
>> MyMex TestStieBrockett
Building with 'g++-4.7'.
MEX completed successfully.
>> n = 12; p = 4; B = randn(n, n); B = B + B'; D = (p:-1:1)';
>> Xinitial = orth(randn(n, p));
>> SolverParams.method = 'RBFGS'; SolverParams.IsCheckParams = 1; SolverParams.Verbose = 1;
>> HasHHR = 0; ParamSet = 1;
>> [Xopt, f, gf, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime] = TestStieBrockett(B, D,
    Xinitial, HasHHR, ParamSet, SolverParams);
(n, p):12,4
GENERAL PARAMETERS:
Stop_Criterion:       GRAD_F_0[YES],    Tolerance      :          1e-06[YES]
Max_Iteration :            500[YES],    Min_Iteration :              0[YES]
OutputGap     :              1[YES],    Verbose          :    FINALRESULT[YES]
LINE SEARCH TYPE METHODS PARAMETERS:
LineSearch_LS :         ARMIJO[YES],    LS_alpha      :         0.0001[YES]
LS_ratio1     :            0.1[YES],    LS_ratio2     :            0.9[YES]
Initstepsize  :              1[YES]
Minstepsize   :    2.22045e-16[YES],    Maxstepsize   :           1000[YES]
RBFGS METHOD PARAMETERS:
nu            :         0.0001[YES],    mu            :              1[YES]
isconvex      :              0[YES]
=======================RBFGS=======================
Iter:51,f:-5.809e+01,|gf|:1.912e-05,|gf|/|gf0|:4.108e-07,time:0.00e+00,nf:57,ng:52,nR:56,nH
    :51,nV(nVp):51(51),
```

2.2 Compiling in Julia

Ubuntu: We first generate a shared library by g++. Julia uses this shared library to call ROPTLIB through a C++ interface Cxx. The details are as follows:

1. In order to compile ROPTLIB in Julia, the "Cxx" library at `https://github.com/JuliaInterop/Cxx.jl` is required. Julia and "Cxx" are installing by following the instruction on `https:`

//julialang.org/ and https://github.com/Keno/Cxx.jl. The steps when this user manual is generated are given below for completeness (it takes a few hours):

- Download and install julia v1.3.1 from https://julialang.org/downloads/oldreleases/
- Run julia by "./julia" and in the command line of julia
- Type right bracket "]" to enter the Pkg mode and then use the command

  ```
  add Cxx
  ```

  to install "Cxx".

2. Download the latest version of ROPTLIB and go to the directory of ROPTLIB.

3. Open ROPTLIB/Makefile and make sure "ROOTPATH" is set to be the correct directory of ROPTLIB and JULIA_DIR is the directory of Julia.

4. Install BLAS, LAPACK and FFTW:

   ```
   sudo apt-get install build-essential
   sudo apt-get install liblapack*
   sudo apt-get install libblas*
   sudo apt-get install libfftw3*
   ```

5. Run "make JuliaROPTLIB TP=DriverJuliaProb" to obtain a shared library of ROPTLIB for Julia.

6. Open the downloaded Julia. Go to the directory of ROPTLIB in Julia using command

   ```
   julia> cd("directory_of_ROPTLIB")
   ```

7. Open ROPTLIB/Julia/BeginROPTLIB.jl and make sure that the path of ROPTLIB is correct and the path of head files of Julia is correct.

8. Run ROPTLIB/Julia/BeginROPTLIB.jl by the command "include("Julia/BeginROPTLIB.jl")" to import ROPTLIB into Julia.

9. Type right bracket "]" to enter the Pkg mode and then use the command "add Random" and "add SparseArrays" to add the two packages. These packages are used in the test file.

10. Run JTestStieBrockett.jl by the command "include("Julia/JTestStieBrockett.jl")" to run an example.

MAC: The installation follows the steps as those for Ubuntu except the installation of BLAS, LAPACK and FFTW in Step 4.

1. Download BLAS and LAPACK from http://www.netlib.org/blas/ and http://www.netlib.org/lapack/;

2. Unzip both packages. In terminal, run "make" in both folders to generate *.a libraries;

3. Rename BLAS and LAPACK libraries to "libblas.a" and "liblapack.a" respectively;

4. Download FFTW (fftw-3.3.6-pl2.tar.gz in our test) from `http://www.fftw.org/download.html`. Unzip the file.

5. Install FFTW by following the instruction in `http://www.fftw.org/fftw3_doc/Installation-on-Unix.html#Installation-on-Unix`. In our test, the steps are

   (a) `./configure`

   (b) `make`

   (c) `make install`

2.3 Compiling in a Stand-alone C++ Enviroment

Windows: Users must first install BLAS and LAPACK. For details on a Windows installation, see the links: `http://www.fi.muni.cz/~xsvobod2/misc/lapack/` and `http://www.netlib.org/`. The steps of compiling this code in Windows 7 using IDE Visual Studio Express 2013 are: i) Click "PROJECT→properties"; ii) add directory of ROPTLIB and the directories of header files of BLAS and LAPACK to "Configuration properties→VC++ Directories→General→Include directories"; iii) install FFTW by following the steps in Section 2.1, iv) add the libraries of BLAS, LAPACK, and FFTW to "Configuration properties → Linker→ Input → Additional Dependencies". To compile and run a test file, press F5 or ctrl + F5 to compile and run the test problems in `./test/DriverCpp.cpp`.

Ubuntu: The steps to set up in terminal are:

1. Install BLAS, LAPACK and FFTW:

```
sudo apt-get install build-essential
sudo apt-get install liblapack*
sudo apt-get install libblas*
sudo apt-get install libfftw3*
```

2. Download the latest version of ROPTLIB and go the the directory of ROPTLIB.

3. Run Makefile to generate a binary file for a test problem, i.e., using the command:

```
make ROPTLIB TP=DriverCpp
```

4. Run "./DriverCpp" to see the test results for `./test/DriverCpp.cpp`.

MAC: The steps to set up in Xcode (tested for Xcode 11.3) are:

1. Download Xcode from official webpage or App Store and install it.

2. Download FFTW (fftw-3.3.8.tar.gz in our test) from `http://www.fftw.org/download.html`. Unzip the file.

3. Install FFTW by following the instruction in `http://www.fftw.org/fftw3_doc/Installation-on-Unix.html#Installation-on-Unix`. In our test, the steps are

   (a) `./configure`

(b) `make`

(c) `make install`

4. Download the corresponding version of ROPTLIB and go the the directory of ROPTLIB.

5. Open Xcode, create a "MacOS/command line tool" project, we name this project "ROPTLIB_mac", delete the default "main.cpp" file in the project. Then add ROPTLIB to this project by simply dragging the subfolders: "Manifolds, Others, Problems, Solvers, and Test" to the project in Xcode. Choose "Create groups" and check the option "Add to targets", in the pop-up window.

6. In Xcode, link FFTW library "libfftw3.a" and "libfftw3f.a" by adding them to
`Build Phases/Link Binary With Libraries`
Note that the FFTW libraries are in `\usr\local\lib` by default

7. In Xcode, add `\usr\local\lib` (the default installation folder of FFTW) to
`Building Settings/Search Paths/Library Search Paths`

8. Two approaches to install BLAS and LAPACK

   (a) Note that the blas and lapack has been installed in MacOS. In Xcode, link BLAS, and LAPACK by adding "libblas.tbd" and "liblapack.tbd" to
   `Build Phases/Link Binary With Libraries`
   However, this approach does not support single precision float point operations.

   (b) Download BLAS and LAPACK from `http://www.netlib.org/blas/` and `http://www.netlib.org/lapack/`; Unzip both packages. In terminal, run "make" in both folders to generate *.a libraries; Rename BLAS and LAPACK libraries to "libblas.a" and "liblapack.a" respectively; In Xcode, link BLAS, and LAPACK by adding "libblas.a" and "liblapack.a" to
   `Build Phases/Link Binary With Libraries`
   Note that if an error "undefined symbol: *gfortran*" occurs when compiling ROPTLIB, then gfortran needs to be installed. Download gfortran from `https://github.com/fxcoudert/gfortran-for-macOS/releases`. Install it and the default installation folder is `/usr/loocal/gfortran`. Add `/usr/local/gfortran/lib` to
   `Building Settings/Search Paths Library Search Path`
   and add "libgfortran.a" to
   `Build Phases/Link Binary With Libraries` This approach supports both double and single precision float point operations.

9. In Xcode, add the path of ROPTLIB and paths of the header files of BLAS and LAPACK (in /ROPTLIB/cwrapper/*) to
`Building Settings/Search Paths/Header Search Paths` and
`Building Settings/Search Paths/User Header Search Paths`

10. Compile and run the project. The driver is in `./test/DriverCpp.cpp`

3 For Matlab Users

3.1 Test Problems and Matlab Interface

ROPTLIB contains three parts, including problem definition, manifold and solver. In order to use this package, a user must define a problem by providing functions of a cost function and specify a domain manifold and a solver. If the gradient and the action of the Hessian are not provided, then numerical gradient and action of the Hessian are computed automatically. For the sake of efficiency, we encourage users to provide gradient and action of Hessian.

Two problems are used as examples. The first is the Brockett cost function on the Stiefel manifold $\mathrm{St}(p,n) = \{X \in \mathbb{R}^{n\times p} | X^T X = I_p\}$ [AMS08, Section 4.8]

$$\min_{X\in\mathrm{St}(p,n)} \mathrm{trace}(X^T B X D) \tag{3.1}$$

where $B \in \mathbb{R}^{n\times n}$, $B = B^T$, $D = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_p)$ and $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$. The second is a summation of three Brockett cost functions

$$\min_{(X_1,X_2,X_3)\in\mathrm{St}(p,n)\times\mathrm{St}(p,n)\times\mathrm{St}(q,m)} \mathrm{trace}(X_1^T B_1 X_1 D_1) + \mathrm{trace}(X_2^T B_2 X_2 D_2) + \mathrm{trace}(X_3^T B_3 X_3 D_3) \tag{3.2}$$

where $B_1, B_2 \in \mathbb{R}^{n\times n}$, $B_3 \in \mathbb{R}^{m\times m}$, $B_1 = B_1^T$, $B_2 = B_2^T$, $B_3 = B_3^T$, $D_1 = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_p)$, $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_p$, $D_2 = \mathrm{diag}(\nu_1, \nu_2, \ldots, \nu_p)$, $\nu_1 \geq \nu_2 \geq \ldots \geq \nu_p$, $D_3 = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_q)$, and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_q$. Problem (3.2) is used to illustrate an implementation for a problem on a product manifold.

First, use the command "MyMex DriverMexProb" to generate a binary file. This binary can be called from Matlab by inputting function handles and parameter structures. We have wrapped this function by a script `/ROPTLIB/Matlab/DriverOPT.m`. DriverOPT.m is used to check correctness of the input parameters.

Listing 2: Matlab interface

```
[FinalIterate, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times,
    eigHess] = DriverOPT(fhandle, gfhandle, Hesshandle, PreConhandle,
                              SolverParams, ManiParams, HasHHR, initialIterate)
```

The script DriverOPT can be called by Listing 2, where "initialIterate" and "finalIterate" are structures that contain initial and final iterates respectively; "fv" is the final cost function value; "gfv" is the norm of the final gradient; "gfgf0" is the norm of the final gradient over the norm of the initial gradient; "iter", "nf", "ng", "nR", "nV/nVp", "nH" donote the number of iterations, the number of function evaluations, the number of gradient evaluations, the number of retraction evaluations, the number of vector transports (expensive/cheap)[2], and the number of evaluations of the action of the Hessian respectively; "ComTime" denotes the total computational time; "funs", "grads", and "times" are arrays that store the function values, norms of gradients/directions and the accumulated computational time at each iteration. "eigHess" is an array of length 4. The first two values are the smallest and the largest eigenvalues of the Riemannian Hessian at the initial iterate and the latter two are the two values at the final iterate. "fhandle", "gfhandle",

[2]Two numbers of vector transports are reported. The first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$.

"Hesshandle" and "PreConhandle" are function handles of cost function, its Euclidean gradient, the action of its Euclidean Hessian, and a preconditioner. The latter three function handles are allowed to be empty, i.e., "[]". If they are empty, then numerical gradient, numerical action of Hessian and no preconditioner are used. "SolverParams" and "ManiParams" are structures that specify parameters of the solver and manifold respectively; and "HasHHR" indicates whether the locking condition [HGA15, (2.8)] is satisfied using the testing approach in [HGA15, Section 4.1].

3.2 A Simple Example

An example for Brockett cost function (3.1) is given in Listing 3 and the code can be found in `/ROPTLIB/Matlab/ForMatlab/MTestStieBrockett.m`.[3] First, the cost function, the Euclidean gradient and action of the Euclidean Hessian are given from line 32 to line 43. Their function handles are assigned from line 5 to line 7. Iterates and tangent vectors are stored as structures with the field "main", as shown in line 18 and line 34. In order to store temporary data to save computations, users can put the temporary data on an iterate with a different field. For example, the Brockett cost function is $\mathrm{trace}(X^T BXD)$ and the Euclidean gradient is $2BXD$. It is required to evaluate BXD in the cost function evaluation. Therefore, one can use the result from the function evaluation to reduce computation in the gradient evaluation. This can be seen from the definitions of f(x, B, D) and gf(x, B, D) in line 33 and line 38. All fields for each solver and manifold are defined in Appendices B and C.

Note that besides using the default line search algorithms and the default stopping criteria, users are allowed to define their own stopping criterion and line search algorithm. Lines 10 to 12 specify the stopping criterion and line search algorithm using the functions defined from line 24 to line 30. The input variables x, eta, $t0$, $s0$ and $output$ defined in

```
output = LinesearchInput(x, eta, t0, s0)
```

represent the current iterate, the search direction, the suggested initial stepsize and the initial slope respectively. If the parameter of line search solvers, *IsPureLSInput* (see Appendix B), is set to be false, then the step size found by "LinesearchInput" will be used as the initial step size for a backtracking algorithm. Otherwise, the step size will be the accepted step size. The variables x, $funs$, ngf and $ngf0$ defined in

```
output = IsStopped(x, funs, ngf, ngf0)
```

represent the current iterate, the arrays of function values, the norm of the gradient at x and the norm of the gradient at the initial iterate respectively.

Listing 3: Test Brockett

```
1 function [FinalX, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times
      ] = testBrockett()
2   n = 5; p = 2;                  % size of the Stiefel manifold
3   B = randn(n, n); B = B + B';   % data matrix
4   D = sparse(diag(p : -1 : 1));  % data matrix
5   fhandle = @(x)f(x, B, D);      % cost function handle
6   gfhandle = @(x)gf(x, B, D);    % gradient
7   Hesshandle = @(x, eta)Hess(x, eta, B, D); % Hessian
8
9   SolverParams.method = 'RSD';   % Use RSD solver
```

[3] The code in the file may not be exactly the same as that in the Listings. The code in the file tests more parameters and runs more/different algorithms. Therefore, the differences are minor and should not cause confusion.

```
  SolverParams.IsStopped = @IsStopped; % Don't use one of the default stopping criteria. Use the one specified by the IsStopped function handle.
  SolverParams.LineSearch_LS = 5; % Don't use one of the default line search algorithm. Use the one specified by the LinesearchInput function handle.
  SolverParams.LinesearchInput = @LinesearchInput;

  ManiParams.name = 'Stiefel';     % Domain is the Stiefel manifold
  ManiParams.n = n;                % assign size to manifold parameter
  ManiParams.p = p;                % assign size to manifold parameter

  initialX.main = orth(randn(n, p));  % initial iterate

  % call the driver
  [FinalX, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times] = DriverOPT(fhandle, gfhandle, Hesshandle, SolverParams, ManiParams, initialX);
end

function output = LinesearchInput(x, eta, t0, s0)
    output = 1;
end

function output = IsStopped(x, funs, ngf, ngf0)
    output = ngf / ngf0 < 1e-5;
end

function [output, x] = f(x, B, D)
  x.BUD = B * x.main * D;
  output = x.main(:)' * x.BUD(:);
end

function [output, x] = gf(x, B, D)
  output.main = 2 * x.BUD;
end

function [output, x] = Hess(x, eta, B, D)
  output.main = 2 * B * eta.main * D;
end
```

3.3 An Example for a Product of Manifolds

An example for a summation of three Brockett cost functions is given in Listing 4, and the associated code can be found in /ROPTLIB/Matlab/ForMatlab/MTestProdStieSumBrockett.m.[4] An array of structures is used to specify a product of manifolds. Suppose the manifold $\mathcal{M}$ is $\mathcal{M}_1^{t_1} \times \mathcal{M}_2^{t_2} \times \ldots \times \mathcal{M}_s^{t_s} := \mathcal{M}_1 \times \ldots \times \mathcal{M}_1 \times \mathcal{M}_2 \times \ldots \mathcal{M}_2 \times \ldots \times \mathcal{M}_s \times \ldots \times \mathcal{M}_s$, where the number of $\mathcal{M}_i$ is t_i. Then the structure specifying parameters of manifolds is an array with length s and the field "numofmani" in i-th element of the array is assigned to be t_i. One example can be found in the function "testSumBrockett()" of Listing 4 from line 20 to line 27.

All components of an iterate of products of manifolds are stored in a consecutive memory. Suppose the length of the i-th component of iterate in product of manifold $\mathcal{M}_1 \times \mathcal{M}_2 \times \ldots \times \mathcal{M}_w$ is ℓ_i. The i-th component of the iterate is stored in the space from $\sum_{j=1}^{i-1} \ell_j + 1$ to $\sum_{j=1}^{i} \ell_j$ in the field "main" of the iterate structure. The same method is used to store tangent vectors. An example is given in lines 29 to 31, 40 to 42, 49, 56 to 58 and 62 in Listing 4.

Listing 4: Test Summation of Brockett

[4]The code in the file may not be exactly the same as that in the Listings. The code in the file tests more parameters and runs more/different algorithms. Therefore, the differences are minor and should not cause confusion.

```matlab
function [FinalX, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times, eigHess] = testSumBrockett()
  n = 5;
  p = 2;
  m = 6;
  q = 3;
  B1 = randn(n, n); B1 = B1 + B1';
  D1 = sparse(diag(p : -1 : 1));
  B2 = randn(n, n); B2 = B2 + B2';
  D2 = sparse(diag(p : -1 : 1));
  B3 = randn(m, m); B3 = B3 + B3';
  D3 = sparse(diag(q : -1 : 1));

  fhandle = @(x)f(x, B1, D1, B2, D2, B3, D3);
  gfhandle = @(x)gf(x, B1, D1, B2, D2, B3, D3);
  Hesshandle = @(x, eta)Hess(x, eta, B1, D1, B2, D2, B3, D3);

  SolverParams.method = 'RSD';

  % Set up domain of manifold, St(p, n)^2 \times St(q, m)
  ManiParams(1).name = 'Stiefel';
  ManiParams(1).numofmani = 2;      % the number of St(p, n) is two
  ManiParams(1).n = n;
  ManiParams(1).p = p;
  ManiParams(2).name = 'Stiefel';
  ManiParams(2).numofmani = 1;   % the number of St(q, m) is one
  ManiParams(2).n = m;
  ManiParams(2).p = q;

  % generate initial iterate
  X1 = orth(randn(n, p)); X2 = orth(randn(n, p)); X3 = orth(randn(m, q));
  initialX.main = [X1(:); X2(:); X3(:)];

  [FinalX, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times] =
      DriverOPT(fhandle, gfhandle, Hesshandle, SolverParams, ManiParams, initialX);
end

function [output, x] = f(x, B1, D1, B2, D2, B3, D3)
  n = size(B1, 1); p = size(D1, 1);
  m = size(B3, 1); q = size(D3, 1);

  X1 = reshape(x.main(1 : n * p), n, p);
  X2 = reshape(x.main(n * p + 1 : 2 * n * p), n, p);
  X3 = reshape(x.main(2 * n * p + 1 : 2 * n * p + m * q), m, q);

  x.BUD1 = B1 * X1 * D1; x.BUD2 = B2 * X2 * D2; x.BUD3 = B3 * X3 * D3;
  output = X1(:)' * x.BUD1(:) + X2(:)' * x.BUD2(:) + X3(:)' * x.BUD3(:);
end

function [output, x] = gf(x, B1, D1, B2, D2, B3, D3)
  output.main = [x.BUD1(:); x.BUD2(:); x.BUD3(:)];
  output.main = 2 * output.main;
end

function [output, x] = Hess(x, eta, B1, D1, B2, D2, B3, D3)
  n = size(B1, 1); p = size(D1, 1);
  m = size(B3, 1); q = size(D3, 1);
  eta1 = reshape(eta.main(1 : n * p), n, p);
  eta2 = reshape(eta.main(n * p + 1 : 2 * n * p), n, p);
  eta3 = reshape(eta.main(2 * n * p + 1 : 2 * n * p + m * q), m, q);
  xi1 = 2 * B1 * eta1 * D1;
  xi2 = 2 * B2 * eta2 * D2;
  xi3 = 2 * B3 * eta3 * D3;
  output.main = [xi1(:); xi2(:); xi3(:)];
end
```

3.4 Checking the Correctness of the Gradient and the Action of the Hessian

ROPTLIB provides a function to test the correctness of the gradient and the action of the Hessian. Let $\hat{f}_x(\eta_x)$ be $f(R_x(\eta_x))$. If $f \in C^2$, then using Taylor's theorem yields

$$
\begin{aligned}
\hat{f}_x(\eta_x) =& \hat{f}_x(0_x) + \langle \operatorname{grad} \hat{f}_x(0_x), \eta_x \rangle + \frac{1}{2} \langle \operatorname{Hess} \hat{f}_x(0_x)[\eta_x], \eta_x \rangle + o(\|\eta_x\|^2) \\
=& f(x) + \langle \operatorname{grad} f(x), \eta_x \rangle + \frac{1}{2} \langle \operatorname{Hess} \hat{f}_x(0_x)[\eta_x], \eta_x \rangle + o(\|\eta_x\|^2).
\end{aligned}
$$

If the retraction R is a second-order retraction or x is a stationary point of f, then $\operatorname{Hess} \hat{f}_x(0_x) = \operatorname{Hess} f(x)$ by [AMS08, Propositions 5.5.5 and 5.5.6]. It follows that

$$
f(y) = f(x) + \langle \operatorname{grad} f(x), \eta_x \rangle + \frac{1}{2} \langle \operatorname{Hess} f(R_x(\eta_x))[\eta_x], \eta_x \rangle + o(\|\eta_x\|^2),
$$

where $y = R_x(\eta_x)$. The function in this package computes

$$
(f(y) - f(x)) / \langle \operatorname{grad} f(x), \eta_x \rangle \tag{3.3}
$$

and

$$
(f(y) - f(x) - \langle \operatorname{grad} f(x), \eta_x \rangle) / (0.5 \langle \operatorname{Hess} f(R_x(\eta_x))[\eta_x], \eta_x \rangle) \tag{3.4}
$$

for $\eta_x = \alpha \xi$ such that $\|\xi\| = 1$, α decreases from 100 to $100 * 2^{-35}$. Suppose there exists an interval of α such that numerical errors do not have significant effect and the values of α are sufficiently small so that the higher order term is negligible. If (3.3) is approximately 1 in the interval, then $\operatorname{grad} f$ is probably correct. Likewise, if (3.4) is approximately 1 in the interval, the retraction R is a second-order retraction or x is a stationary point of f, then the $\operatorname{Hess} f$ is probably correct.

To run the function, users must set the field "IsCheckGradHess" to 1 in the solver's parameters. For example, adding the command "SolverParams.IsCheckGradHess = 1" in line 13 of the function "testBrockett()" in Listing 3 sets "IsCheckGradHess" to 1. Two sets of values (3.3) and (3.4) are output. One is at the initial iterate and the other at the final iterate obtained by the solver. The values of (3.3) at the initial iterate indicates if the Riemannian gradient and Euclidean gradient are correct and the values of (3.4) at the final iterate indicates if the actions of the Riemannian Hessian and Euclidean Hessian are correct.

4 For Julia Users

4.1 A Simple Example

In Julia, a shared library of ROPTLIB needs to be generated first (see Section 2 for details). ROPTLIB then can be added to Julia by running `ROPTLIB/Julia/BeginROPTLIB.jl`. The interface in Julia is given by

```
[FinalIterate, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times,
    eigHess] = DriverJuliaOPT(Handles, SolverParams, ManiParams, HasHHR, initialIterate,
    solution)
```

where the notation is the same as those in Listing 2 except that "Handles" is a composite type containing all the function names (see an example in lines 8 to 10 in Listing 5).

Listing 5 shows an example to optimize the Brockett cost function in (3.1). The code is available in `/ROPTLIB/Julia/JTestStieBrockett.jl`. The manifold is specified from lines 1 to 12. Note that the name and size of a manifold are defined to be an array. This is done to make the code compatible for a product of manifolds. See `/ROPTLIB/Julia/JTestProdStieSumBrockett.jl` or Section 4.2 for an example on a product of manifolds and related information. The names of the cost function, gradient, action of Hessian, stopping criterion, and line search algorithm are given from lines 15 to 17. The functions are defined later from lines 44 to 85. The solver-related parameters are defined from lines 22 to 30. Unlike *ManiParams* and *FunHandles*, an object **Sparams** of *SolverParams* has been defined in BeginROPTLIB.jl. Therefore, users do not need to create an object of type *SolverParams* but need only modify **Sparams**. The default values of **Sparams** can be found in Appendix B.

The Julia interface also supports sharing data across functions. As shown in line 51 of Listing 5, the temporary data is stored in the object **outTmp**. In the gradient evaluation, the temporary data is given in the object **inTmp** and can be used to avoid redundant computations.

Note that size information about all data is not explicitly stored with the data in ROPTLIB and, therefore, it is required to reshape the data, as shown in lines 30, 37, and 43. This has little impact on the efficiency of ROPTLIB since the data in memory do not change when reshaped.

Listing 5: Test Brockett

```
1  # set domain manifold to be the Stiefel manifold.
2  # Note that every parameter in ManiParams is an array. The idea to use an array
3  # is to make the framework compatible with produce of manifolds. See details in
4  # JTestProdStieSumBrockett.jl
5  mani = "Stiefel"; numoftypes = 1;
6  ms = [-1] # -1 means that m is not used in Stiefel manifold.
7  numofmani = [1]
8  # The size is R^{5 \times 3}
9  ns = [5];
10 ps = [3];
11 paramsets = [2];
12 Mparams = ManiParams(1, length(ManArr), pointer(ManArr), pointer(numofmani), pointer(
       paramsets), pointer(UseDefaultArr), pointer(ns), pointer(ps))
13
14 # set function handles
15 fname = "func"
16 gfname = "gfunc"
17 hfname = "hfunc"
18 isstopped = "stopfunc" # or empty string "" if use a default one
19 LinesearchInput = "LSfunc" # or empty string "" if use a default one
20 Handles = FunHandles(pointer(fname), pointer(gfname), pointer(hfname), pointer(isstopped),
       pointer(LinesearchInput))
21
22 # set solvers by modifying the default one.
23 method = "LRBFGS"
24 Sparams.name = pointer(method)
25 Sparams.OutputGap = 1
26 Sparams.LineSearch_LS = 5
27 Sparams.Max_Iteration = 50
28 Sparams.IsPureLSInput = 0
29 Sparams.Stop_Criterion = 0
30 Sparams.IsCheckGradHess = 1
31
32 # use locking condition or not
33 HasHHR = 0
34
35 # Initial iterate and problem
36 srand(1)
```

```
n = ns[1]
p = ps[1]
B = randn(n, n)
B = B + B'
D = sparse(diagm(linspace(p, 1, p)))
initialX = qr(randn(ns[1], ps[1]))[1]

# Define function handles
# The function names are assigned to the "FunHandles" struct.
# See lines 17-21
function func(x, inTmp)
    x = reshape(x, n, p) # All the input argument is a vector. One has to reshape it to have a proper size
    outTmp = B * x * D
    fx = vecdot(x, outTmp)
    return (fx, outTmp) # The temparary data "outTmp" will replace the "inTmp"
end

function gfunc(x, inTmp) # The inTmp is the temparary data computed in "func".
    inTmp = reshape(inTmp, n, p) # All the input argument is a vector. One has to reshape it to have a proper size
    gf = 2.0::Float64 * inTmp
    return (gf, []) # If one does not want to change the temparary data, then let the outTmp be an empty array.
end

function hfunc(x, inTmp, eta)
    eta = reshape(eta, n, p) # All the input argument is a vector. One has to reshape it to have a proper size
    result = 2.0::Float64 * B * eta * D
    return (result, []) # If one does not want to change the temparary data, then let the outTmp be an empty array.
end

# Users can define their own stopping criterion by passing the name
# of the function to the "isstopped" field in the object of structure FunHandles
function stopfunc(x, gf, fx, ngfx, ngfx0)
# x: the current iterate
# gf: the gradient at x
# gx: the function value at x
# ngfx: the norm of gradient at x
# ngfx0: the norm of gradient at the initial iterate
    return (ngfx / ngfx0 < 1e-6)
end

# Users can define their own line search method by passing the name
# of the function to the "LinesearchInput" field in the object of structure FunHandles
function LSfunc(x, eta, t0, s0)
# x: the current iterate
# eta: the search direction
# t0: the initial step size
# s0: the slope of the line search scalar function at zero
    return 1.0::Float64
end

# Call the solver and get results. See the user manual for details about the outputs.
(FinalIterate, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times,
    eigHess) = DriverJuliaOPT(Handles, Sparams, Mparams, HasHHR, initialX)
```

4.2 An Example for a Product of Manifolds

An example for minimzing a summation of Brockett cost functions 3.2 is given in Listing 7. The setting is the same as that in Section 4.1. All the data are stored in consecutive memory and the shape information is not explicitly stored with the data. Therefore, as in e.g., lines 58 to 60 and lines 72 to 74, each component is obtained by extracting and reshaping the input variables using knowledge of the manner in which the initial data was specified and the relevant manifold parameters.

As shown in Table 35 of Appendix C, the size information of a manifold is specified by at most three letters, m, n, and p. The example shown in Listing 7 only involves p and n. Here we show what if m is also involved. Suppose the product manifold is $(\mathrm{St}(p, n))^2 \times \mathbb{R}^{m \times n}$. The code to generate such a product manifold is given in Listing 6. If a manifold does not need a value of a letter, such as the Stiefel manifold does not need a value of m, then its corresponding value can be set to be any value and we use 0 in the code.

Listing 6: Generate the product of manifolds

```
# set domain manifold to be the product of manifolds: \left(\St(p, n)\right)^2 \times \mathbb{R}^{m \times n}.
manis = "Stiefel, Euclidean"
numoftypes = 2
numofmani = [2, 1] # Orthogonal group has power 3, therefore, the corresponding number is set to be 3.
St_p = 3, St_n = 5, Euc_m = 4, Euc_n = 6
ms = [0, Euc_m]; # first one is for Stiefel, Second one is for Oblique and the last one is for Orthogonal group
ns = [St_n, Euc_n];
ps = [St_p, 0];
paramsets = [1, 1];
Mparams = ManiParams(1, numoftypes, pointer(manis), pointer(numofmani), pointer(paramsets), pointer(ms), pointer(ns), pointer(ps))
```

Listing 7: Test summation of Brockett

```
1  # set domain manifold to be the product of Stiefel manifolds: St(p, n)^2 \times St(q, m).
2  manis = "Stiefel, Stiefel"
3  numoftypes = 2
4  ms = [-1, -1] # -1 means that ms are not used in the Stiefel manifolds.
5  numofmani = [2, 1] # St(p, n) has power 2, therefore, the corresponding number is set to be 2.
6  # p = 3, n = 5
7  # q = 2, m = 6
8  ns = [5, 6];
9  ps = [3, 2];
10 paramsets = [1, 1];
11 Mparams = ManiParams(1, numoftypes, pointer(manis), pointer(numofmani), pointer(paramsets), pointer(ms), pointer(ns), pointer(ps))
12
13 # set function handles
14 fname = "func_P"
15 gfname = "gfunc_P"
16 hfname = "hfunc_P"
17 isstopped = "stopfunc_P"
18 LinesearchInput = "LSfunc_P"
19 Handles = FunHandles(pointer(fname), pointer(gfname), pointer(hfname), pointer(isstopped), pointer(LinesearchInput))
20
21 # set solvers by modifying the default one.
22 method = "LRBFGS"
23 Sparams.name = pointer(method)
24 Sparams.OutputGap = 1
25 Sparams.LineSearch_LS = 5
26 Sparams.Max_Iteration = 50
27 Sparams.IsPureLSInput = 0
28
```

```julia
29 # use locking condition or not
30 HasHHR = 0
31
32 # Initial iterate and problem
33 srand(1)
34 n = ns[1]
35 p = ps[1]
36 m = ns[2]
37 q = ps[2]
38 B1 = randn(n, n)
39 B1 = B1 + B1'
40 D1 = sparse(diagm(linspace(p, 1, p)))
41 B2 = randn(n, n)
42 B2 = B2 + B2'
43 D2 = sparse(diagm(linspace(p, 1, p)))
44 B3 = randn(m, m)
45 B3 = B3 + B3'
46 D3 = sparse(diagm(linspace(q, 1, q)))
47
48 initialX1 = qr(randn(ns[1], ps[1]))[1]
49 initialX2 = qr(randn(ns[1], ps[1]))[1]
50 initialX3 = qr(randn(ns[2], ps[2]))[1]
51 initialX = [reshape(initialX1, n * p, 1); reshape(initialX2, n * p, 1); reshape(initialX3, m * q, 1)]
52
53
54 # Define function handles
55 # The function names are assigned to the "FunHandles" struct.
56 # See lines 17-21
57 function func_P(x, inTmp) # All the input argument is a vector.
58     x1 = reshape(view(x, 1 : n * p), n, p)
59     x2 = reshape(view(x, n * p + 1 : 2 * n * p), n, p)
60     x3 = reshape(view(x, 2 * n * p + 1 : 2 * n * p + m * q), m, q)
61     outTmp = [reshape(B1 * x1 * D1, n * p, 1); reshape(B2 * x2 * D2, n * p, 1); reshape(B3 * x3 * D3, m * q, 1)]
62     fx = vecdot(x, outTmp)
63     return (fx, outTmp) # The temparary data "outTmp" will replace the "inTmp"
64 end
65
66 function gfunc_P(x, inTmp)
67     gf = 2.0::Float64 * inTmp
68     return (gf, []) # If one does not want to change the temparary data, then let the outTmp be an empty array.
69 end
70
71 function hfunc_P(x, inTmp, eta)
72     eta1 = reshape(view(eta, 1:n*p), n, p) # All the input argument is a vector. One has to reshape it to have a proper size
73     eta2 = reshape(view(eta, n*p+1:2*n*p), n, p)
74     eta3 = reshape(view(eta, 2*n*p+1:2*n*p+m*q), m, q)
75
76     result = [reshape(2.0::Float64 * B1 * eta1 * D1, n * p, 1); reshape(2.0::Float64 * B2 * eta2 * D2, n * p, 1); reshape(2.0::Float64 * B3 * eta3 * D3, m * q, 1)]
77     return (result, []) # If one does not want to change the temparary data, then let the outTmp be an empty array.
78 end
79
80 function stopfunc_P(x, gf, fx, ngfx, ngfx0)
81     return (ngfx / ngfx0 < 1e-6)
82 end
83
84 function LSfunc_P(x, eta, t0, s0)
85     return 1.0::Float64
86 end
87
```

```
(FinalIterate, fv, gfv, gfgf0, iter, nf, ng, nR, nV, nVp, nH, ComTime, funs, grads, times) =
    DriverJuliaOPT(Handles, Sparams, Mparams, HasHHR, initialX)
```

5 For C++ Users

The classes in the package and their relationships are given in Figures 1 to 4. All the classes that store data inherit an abstract class, *SmartSpace*. The copy-on-write strategy is used in *SmartSpace*. Its derived class *Element* is the main container class in ROPTLIB. Some commonly-used linear algebra operations are also supported in this class, such as matrix multiplication, QR, SVD factorazition. In the abstract class *Manifold*, all functions only related to manifolds are declared, e.g., retraction, vector transport. Some of these functions are also given default definitions, e.g., the default metric is the Frobenius inner product. The abstract class *Problem* contains all prototypes of the cost function, the Riemannian gradient, the Euclidean gradient, the action of the Riemannian Hessian and the action of the Euclidean Hessian. It not only automatically chooses functions that have been overridden (polymorphism), but also includes a function to check the correctness of the gradient and the action of the Hessian, see Section 3.4. The domain of a problem must also be specified using one of the manifold classes. Note that class *mexProblem* is a bridge between C++ and Matlab. It uses function handles of Matlab and produces C++ functions. Each solver accepts an object of *Problem* and an object of *Variable* (an initial iterate), and outputs a final iterate based on the given parameters.

Users must write a problem class by inheriting the abstract class `/ROPTLIB/Problems/Problem.h` and override either functions of cost function, Riemannian gradient and action of Riemannian Hessian

```
virtual realdp f(const Variable &x) const;
virtual Vector &RieGrad(const Variable &x, Vector *result) const;
virtual Vector &RieHessianEta(const Variable &x, const Vector &etax, Vector *result) const;
```

or functions of cost function, Euclidean gradient and action of Euclidean Hessian.

```
virtual realdp f(const Variable &x) const;
virtual Vector &EucGrad(const Variable &x, Vector *result) const;
virtual Vector &EucHessianEta(const Variable &x, const Vector &etax, Vector *result) const;
```

Note that the "realdp" in ROPTLIB is defined to be either "double" or "float" depending whether single precision or double precision float point is used. One can specify the precision by modifying Line 13 in "def.h".

Throughout this section, a class or a routine is written in *this font* and an object is written in **this font**.

5.1 A Simple Example

An example for the Brockett cost function (3.1) is given in Listings 8, 9 and 10. Listings 8 and 9 give details of two files, StieBrockett.h and StieBrockett.cpp, which inherit the class *Problem* and define the Brockett problem. The Euclidean gradient and the action of the Euclidean Hessian are overridden. Listing 10 gives a test file for the Brockett cost function minimization problem. Those codes can be found in `/ROPTLIB/Problems/StieBrockett*` and `/ROPTLIB/test/TestStieBrockett.cpp`.[5]

[5]The code in the file may not be exactly the same as that in the Listings. The code in the file tests more parameters and runs more/different algorithms. Therefore, the differences are minor and should not cause confusion.

For any class derived from *SmartSpace*, any one of the following three functions can be used to obtain a double pointer to the data:

```
virtual const realdp *ObtainReadData(void) const;
virtual realdp *ObtainWriteEntireData(void);
virtual realdp *ObtainWritePartialData(void);
```

ObtainReadData returns a constant pointer and users are not allowed to modify the data. This is the fastest way to access the data but users have the most limited authority. The memory functions *ObtainWriteEntireData* and *ObtainWritePartialData* are allowed to access the data and modify them. *ObtainWriteEntireData* may not preserve the old data in memory and this function is used when users want to completely overwrite the data. *ObtainWritePartialData* guarantees that the memory retains the old data. This is the most inefficient approach but it preserves the old data information and is used if users only partially modify the data.

C++ code provides a way to share information in the computation of the cost function, the gradients and the actions of the Hessians. One example can be found in Listing 9. The object **BxD** in Line 24 is used to store the shared information. It is attached to **x** in Line 27. In Line 33, the data in **BxD** can be obtained by the same name "BxD". More examples can be found in files under directory `/ROPTLIB/Problems/`.

Users are allowed to define a line search algorithm and a stopping criterion. Lines 5 to 15 in Listing 10 show an example of a definition of a line search algorithm and stopping criterion. The input variables are the same as those in Matlab, see Section 3.2. Their function pointers are assigned to the solvers on lines 45 and 47 in Listing 10. The false value of the parameter *IsPureLSInput* in line 46 indicates that the step size returned by the user-specified function is used as an initial step size in a back tracking algorithm to satisfy the Armijo condition. If the value is true, then the step size is used as the accepted step size. Note that in this case, users must guarantee that the step size is sufficient for convergence.

C++ codes, of course, support checking correctness of the gradients and the actions of the Hessians. An example is given in line 53 of Listing 10.

Listing 8: File "StieBrockett.h" for test Brockett in C++

```
// File: StieBrockett.h

#ifndef STIEBROCKETT_H
#define STIEBROCKETT_H

#include "Manifolds/Stiefel.h"
#include "Problems/Problem.h"
#include "Others/def.h"

/*Define the namespace*/
namespace ROPTLIB{

    class StieBrockett : public Problem{
    public:
        StieBrockett(Vector inB, Vector inD);
        virtual ~StieBrockett();
        virtual realdp f(const Variable &x) const;

        virtual Vector &EucGrad(const Variable &x, Vector *result) const;
        virtual Vector &EucHessianEta(const Variable &x, const Vector &etax, Vector *result)
             const;

        Vector B;
        Vector D;
```

```
        integer n;
        integer p;
    };
}; /*end of ROPTLIB namespace*/
#endif /* end of STIEBROCKETT_H */
```

Listing 9: File "StieBrockett.cpp" for test Brockett in C++

```
// File: StieBrockett.cpp

#include "Problems/StieBrockett.h"

/*Define the namespace*/
namespace ROPTLIB{

    StieBrockett::StieBrockett(Vector inB, Vector inD)
    {
        B = inB;
        D = inD;
        n = B.Getrow();
        p = D.Getlength();

        NumGradHess = false;
    };

    StieBrockett::~StieBrockett(void)
    {
    };

    realdp StieBrockett::f(const Variable &x) const
    {
        Vector BxD(n, p); BxD.AlphaABaddBetaThis(1, B, GLOBAL::N, x, GLOBAL::N, 0);
        D.DiagTimesM(BxD, GLOBAL::R); /* BxD = B * x; BxD = D.GetDiagTimesM(BxD, "R");*/
        realdp result = x.DotProduct(BxD);
        x.AddToFields("BxD", BxD);
        return result;
    };

    Vector &StieBrockett::EucGrad(const Variable &x, Vector *result) const
    {
        *result = x.Field("BxD");
        result->ScalarTimesThis(2);
        return *result;
    };

    Vector &StieBrockett::EucHessianEta(const Variable &x, const Vector &etax, Vector *
        result) const
    {
        result->AlphaABaddBetaThis(2, B, GLOBAL::N, etax, GLOBAL::N, 0);
        D.DiagTimesM(*result, GLOBAL::R);
        return *result; /* 2 * D.GetDiagTimesM(B * etax, "R"); */
    };
}; /*end of ROPTLIB namespace*/
```

Listing 10: File "TestStieBrockett.cpp" for test Brockett in C++

```
// File: TestStieBrockett.cpp

using namespace ROPTLIB;

/*User-specified linesearch algorithm*/
double LinesearchInput(integer iter, const Variable &x1, const Vector &exeta1, realdp
    initialstepsize, realdp initialslope, const Problem *prob, const Solvers *solver)
{
```

```
    return 1;
}

/*User-specified stopping criterion*/
bool MyStop(const Variable &x, const Vector &funSeries, integer lengthSeries, realdp
    finalval, realdp initval, const Problem *prob, const Solvers *solver)
{
    return (finalval / initval < 1e-6);
};

void testStieBrockett(void)
{
    // size of the Stiefel manifold
    integer n = 5, p = 2;
    // Generate the matrices in the Brockett problem.
    Vector B(n, n), D(p);
    B.RandGaussian();
    B = B + B.GetTranspose();
       realdp *Dptr = D.ObtainWriteEntireData();
    /*D is a diagonal matrix.*/
    for (integer i = 0; i < p; i++)
        Dptr[i] = static_cast<realdp> (i + 1);

    // Define the manifold
    Stiefel Domain(n, p);
    //Grassmann Domain(n, p);
    Variable StieX = Domain.RandominManifold();

    // Define the Brockett problem
    StieBrockett Prob(B, D);
    /*The domain of the problem is a Stiefel manifold*/
    Prob.SetDomain(&Domain);
    /*Output the parameters of the domain manifold*/
    Domain.CheckParams();

    RSD *RSDsolver = new RSD(&Prob, &StieX);
    RSDsolver->Verbose = FINALRESULT;//--- FINALRESULT;
    RSDsolver->LineSearch_LS = LSSM_INPUTFUN;
    RSDsolver->LinesearchInput = &LinesearchInput;
    RSDsolver->IsPureLSInput = false;
    RSDsolver->StopPtr = &MyStop;
    RSDsolver->Max_Iteration = 100;
    RSDsolver->OutputGap = 1;
    RSDsolver->CheckParams();
    RSDsolver->Run();

    Prob.CheckGradHessian(RSDsolver->GetXopt());

    delete RSDsolver;
}
```

5.2 An Example for a Product of Manifolds

This section gives the C++ code for the problem (3.2) defined on a product of manifolds (see Section 3.3). The codes in Listing 11, 12 and 13 can be found in `/ROPTLIB/Problems/ProdStieSumBrockett*` and `/ROPTLIB/test/TestProdStieSumBrockett.cpp`.[6]

The codes defining a product of manifolds and a point on the manifold is given from line 31 to line 41 of Listing 13. The space for all components required by a point on a product of manifolds is

[6]The code in the file may not be exactly the same as that in the Listings. The code in the file tests more parameters and runs more/different algorithms. Therefore, the differences are minor and should not cause confusion.

stored in consecutive memory locations. A pointer to a segment of memory with length of $2np+mq$ doubles is obtained. The first np doubles are the first component of the iterate. The next np doubles are the second component and the last mq doubles are the last component of the iterate.

Since $\mathbf{x}$ is a point on a product of manifolds, it has multiple components on each manifold. Each component can be obtained by using the member function *Element* &*GetElement(integer)*, e.g., Line 31 of Listing 12. If users want to overwrite data that is pointed to by a pointer obtained by *GetElement(integer)*, then it is required to first use *NewMemoryonWrite(void)* or *CopyOnWrite(void)*. Otherwise, the data on product manifolds would not use consecutive memory and would cause errors. See for example, Lines 51 and 61 in Listing 12.

Listing 11: File "ProdStieSumBrockett.h" for test summation of Brockett in C++

```
// File: ProdStieSumBrockett.h

#ifndef PRODSTIESUMBROCKETT_H
#define PRODSTIESUMBROCKETT_H

#include "Manifolds/Stiefel.h"
#include "Manifolds/MultiManifolds.h"
#include "Problems/Problem.h"
#include "Others/def.h"

/*Define the namespace*/
namespace ROPTLIB{

    class ProdStieSumBrockett : public Problem{
    public:
        ProdStieSumBrockett(Vector inB1, Vector inD1, Vector inB2, Vector inD2, Vector inB3,
             Vector inD3);
        virtual ~ProdStieSumBrockett();
        virtual realdp f(const Variable &x) const;

        virtual Vector &EucGrad(const Variable &x, Vector *result) const;
        virtual Vector &EucHessianEta(const Variable &x, const Vector &etax, Vector *result)
             const;

        Vector B1;
        Vector D1;
        Vector B2;
        Vector D2;
        Vector B3;
        Vector D3;
        integer n;
        integer p;
        integer m;
        integer q;
    };
}; /*end of ROPTLIB namespace*/

#endif /* end of PRODSTIESUMBROCKETT_H */
```

Listing 12: File "ProdStieSumBrockett.cpp" for test summation of Brockett in C++

```
1  // File: ProdStieSumBrockett.cpp
2
3  #include "Problems/ProdStieSumBrockett.h"
4
5  /*Define the namespace*/
6  namespace ROPTLIB{
7
8      ProdStieSumBrockett::ProdStieSumBrockett(Vector inB1, Vector inD1, Vector inB2, Vector
           inD2, Vector inB3, Vector inD3)
```

```
    {
        B1 = inB1;
        D1 = inD1;
        B2 = inB2;
        D2 = inD2;
        B3 = inB3;
        D3 = inD3;

        n = B1.Getrow();
        p = D1.Getlength();
        m = B3.Getrow();
        q = D3.Getlength();

        NumGradHess = false;
    };

    ProdStieSumBrockett::~ProdStieSumBrockett(void)
    {
    };

    realdp ProdStieSumBrockett::f(const Variable &x) const
    {
        Vector B1x1D1(n, p); B1x1D1.AlphaABaddBetaThis(1, B1, GLOBAL::N, x.GetElement(0),
            GLOBAL::N, 0); /* B1x1D1 = B1 * x.GetElement(0); */
        D1.DiagTimesM(B1x1D1, GLOBAL::R);
        realdp result = x.GetElement(0).DotProduct(B1x1D1);
        x.AddToFields("B1x1D1", B1x1D1);

        Vector B2x2D2(n, p); B2x2D2.AlphaABaddBetaThis(1, B2, GLOBAL::N, x.GetElement(1),
            GLOBAL::N, 0); /* B2x2D2 = B2 * x.GetElement(1); */
        D2.DiagTimesM(B2x2D2, GLOBAL::R);
        result += x.GetElement(1).DotProduct(B2x2D2);
        x.AddToFields("B2x2D2", B2x2D2);

        Vector B3x3D3(m, q); B3x3D3.AlphaABaddBetaThis(1, B3, GLOBAL::N, x.GetElement(2),
            GLOBAL::N, 0); /* B3x3D3 = B3 * x.GetElement(2); */
        D3.DiagTimesM(B3x3D3, GLOBAL::R);
        result += x.GetElement(2).DotProduct(B3x3D3);
        x.AddToFields("B3x3D3", B3x3D3);

        return result;
    };

    Vector &ProdStieSumBrockett::EucGrad(const Variable &x, Vector *result) const
    {
        result->NewMemoryOnWrite();
        result->GetElement(0) = x.Field("B1x1D1");
        result->GetElement(1) = x.Field("B2x2D2");
        result->GetElement(2) = x.Field("B3x3D3");
        Domain->ScalarTimesVector(x, 2, *result, result);
        return *result;
    };

    Vector &ProdStieSumBrockett::EucHessianEta(const Variable &x, const Vector &etax, Vector
         *result) const
    {
        result->NewMemoryOnWrite();
        result->GetElement(0).AlphaABaddBetaThis(2, B1, GLOBAL::N, etax.GetElement(0),
            GLOBAL::N, 0);
        D1.DiagTimesM(result->GetElement(0), GLOBAL::R);
        result->GetElement(1).AlphaABaddBetaThis(2, B2, GLOBAL::N, etax.GetElement(1),
            GLOBAL::N, 0);
        D2.DiagTimesM(result->GetElement(1), GLOBAL::R);
        result->GetElement(2).AlphaABaddBetaThis(2, B3, GLOBAL::N, etax.GetElement(2),
            GLOBAL::N, 0);
```

```
        D3.DiagTimesM(result->GetElement(2), GLOBAL::R);

        return *result;
    };
}; /*end of ROPTLIB namespace*/
```

Listing 13: File "TestProdStieSumBrockett.cpp" for test summation of Brockett in C++

```
// File: TestProdStieSumBrockett.cpp

#include "test/TestProdStieSumBrockett.h"

using namespace ROPTLIB;

void testProdStieSumBrockett(void)
{
    // size of the Stiefel manifold
    integer n = 4, p = 2, m = 3, q = 2;

    Vector B1(n, n), B2(n, n), B3(m, m);
    B1.RandGaussian(); B1 = B1 + B1.GetTranspose();
    B2.RandGaussian(); B2 = B2 + B2.GetTranspose();
    B3.RandGaussian(); B3 = B3 + B3.GetTranspose();
    Vector D1(p), D2(p), D3(q);
    realdp *D1ptr = D1.ObtainWriteEntireData();
    realdp *D2ptr = D2.ObtainWriteEntireData();
    realdp *D3ptr = D3.ObtainWriteEntireData();

    for (integer i = 0; i < p; i++)
    {
        D1ptr[i] = static_cast<realdp> (i + 1);
        D2ptr[i] = D1ptr[i];
    }
    for (integer i = 0; i < q; i++)
    {
        D3ptr[i] = static_cast<realdp> (i + 1);
    }

    // number of manifolds in product of manifold
    integer numoftypes = 2; // two kinds of manifolds
    integer numofmani1 = 2; // the first one has two
    integer numofmani2 = 1; // the second one has one

    // Define the Stiefel manifold
    Stiefel mani1(n, p);
    Stiefel mani2(m, q);
    ProductManifold Domain(numoftypes, &mani1, numofmani1, &mani2, numofmani2);
    // Obtain an initial iterate
    Variable ProdX = Domain.RandominManifold();

    // Define the Brockett problem
    ProdStieSumBrockett Prob(B1, D1, B2, D2, B3, D3);

    // Set the domain of the problem to be the product of Stiefel manifolds
    Prob.SetDomain(&Domain);

    // output the parameters of the manifold of domain
    Domain.CheckParams();

    Prob.SetNumGradHess(true);
    Prob.CheckGradHessian(ProdX);
    LRBFGS *LRBFGSsolver = new LRBFGS(&Prob, &ProdX);
    LRBFGSsolver->Verbose = ITERRESULT; //ITERRESULT;//
    LRBFGSsolver->Max_Iteration = 2000;
    LRBFGSsolver->CheckParams();
```

```
    LRBFGSsolver->Run();
    Prob.CheckGradHessian(LRBFGSsolver->GetXopt());

    delete LRBFGSsolver;
}
```

A Relationships among Classes in the Package

A.1 Manifold-related Classes

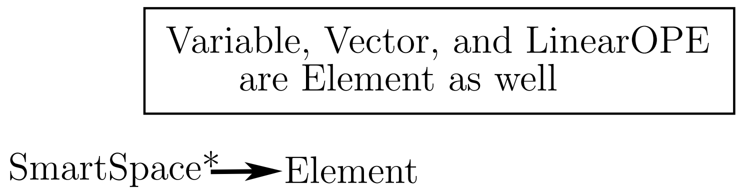

Figure 1: The class hierarchy of space-related classes in ROPTLIB. Note that Variable, Vector, *and* LinearOPE *are defined to be* Element.

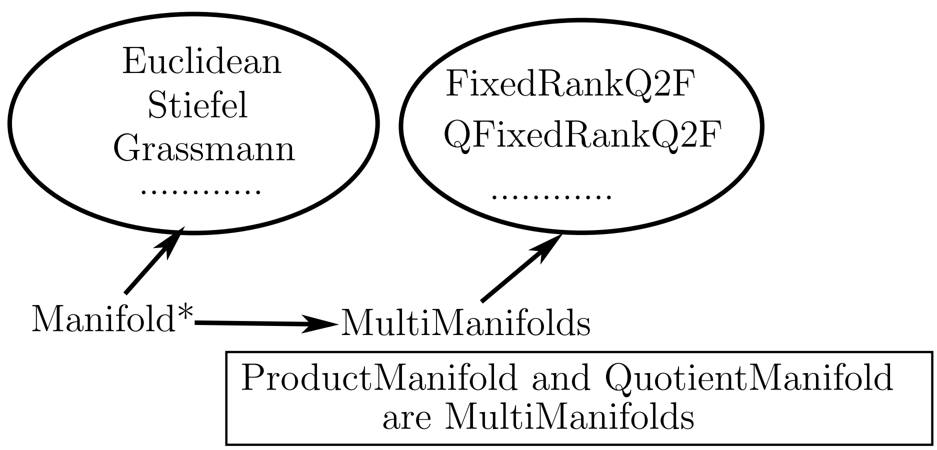

Figure 2: The class hierarchy of manifold-related classes in ROPTLIB. We refer to the documentation in the code for detailed explanations of the functions. ProductManifold and QuotientManifold are defined to be MultiManifolds.

A.2 Problem-related Classes

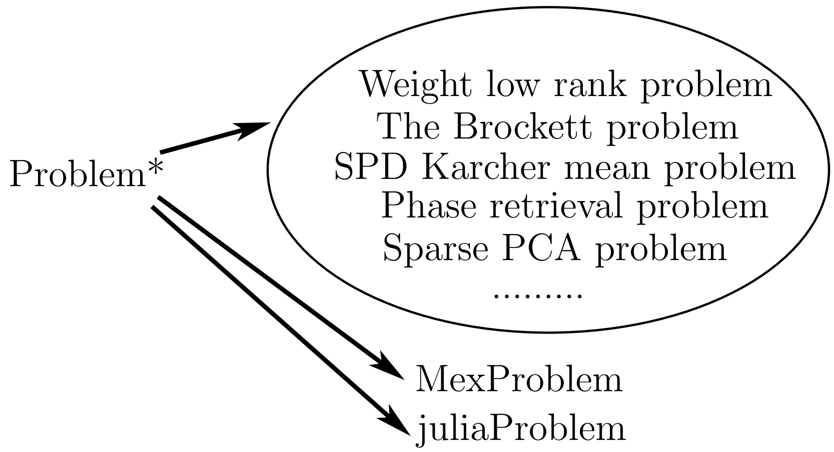

Figure 3: The class hierarchy of problem-related classes in ROPTLIB. We refer to the documentation in the code for detailed explanations of the functions. MexProblem and juliaProblem are problems for Matlab and Julia interfaces.

A.3 Solver-related Classes

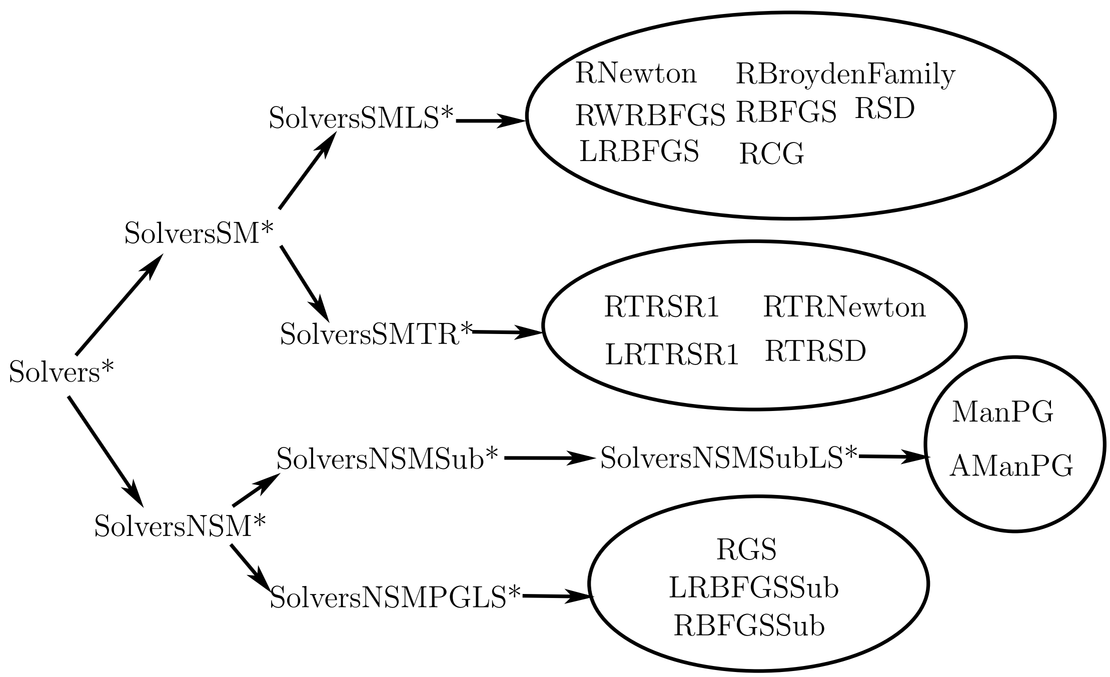

Figure 4: The class hierarchy of solver-related classes in ROPTLIB. We refer to the documentation in the code for detailed explanations of the functions.

B Input Parameters and Output Notation of Solvers

B.1 RTRNewton

Table 2: Input Parameters of RTRNewton

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\\|\operatorname{grad} f(x_i)\\|$ GRAD_F_0 / 2 : $\\|\operatorname{grad} f(x_i)\\|/\\|\operatorname{grad} f(x_0)\\|$
Tolerance	Algorithm stops if "Stop_Criterion" $<$ tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
Acceptence_Rho	Accept candidate if Rho $>$ Acceptence_Rho	0.1	between 0 and 0.25, i.e., $\in (0, 0.25)$
Shrinked_tau	coefficient in reducing radius	0.25	between 0 and 1, i.e., $\in (0, 1)$
Magnified_tau	coefficient in increasing radius	2	greater than 1
minimum_Delta	minimum allowed radius	machine eps	greater than 0 and smaller than or equal to maximum_Delta
maximum_Delta	maximum allowed radius	10000	greater than or equal to minimum_Delta
useRand	whether use Rand in truncate conjugate gradient	false / 0	false / 0 or true / 1
Min_Inner_Iter	minimum number of iterations in truncate conjugate gradient	0	greater than or equal to ZERO and smaller than or equal to Max_Inner_Iter
Max_Inner_Iter	maximum number of iterations in truncate conjugate gradient	1000	greater than or equal to Min_Inner_Iter
theta	in [AMS08, (7.10)]	1	greater than or equal to 0
kappa	in [AMS08, (7.10)]	0.1	between 0 and 1, i.e., $\in (0, 1)$
initial_Delta	initial radius	1	greater than 0

Table 3: Output notation of RTRNewton. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\\| \operatorname{grad} f(x_i) \\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
nH	the number of actions of Hessian
rho	[AMS08, (7.7)]
radius	the radius of trust region
tCGstatus	status of truncate conjugate gradient
innerIter	the number of iterations in truncate conjugate gradient

B.2 RTRSR1

Table 4: Input Parameters of RTRSR1

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\\| \operatorname{grad} f(x_i) \\|$ GRAD_F_0 / 2 : $\\| \operatorname{grad} f(x_i) \\| / \\| \operatorname{grad} f(x_0) \\|$
Tolerance	Algorithm stops if "Stop_Criterion" < tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
Acceptence_Rho	Accept candidate if Rho > Acceptence_Rho	0.1	between 0 and 0.25, i.e., $\in (0, 0.25)$
Shrinked_tau	coefficient in reducing radius	0.25	between 0 and 1, i.e., $\in (0, 1)$
Magnified_tau	coefficient in increasing radius	2	greater than 1

minimum_Delta	minimum allowed radius	machine eps	greater than 0 and smaller than or equal to maximum_Delta
maximum_Delta	maximum allowed radius	10000	greater than or equal to minimum_Delta
useRand	whether use Rand in truncate conjugate gradient	false / 0	false / 0 or true / 1
Min_Inner_Iter	minimum number of iterations in truncate conjugate gradient	0	greater than or equal to ZERO and smaller than or equal to Max_Inner_Iter
Max_Inner_Iter	maximum number of iterations in truncate conjugate gradient	1000	greater than or equal to Min_Inner_Iter
theta	in [AMS08, (7.10)]	0.1	greater than or equal to 0
kappa	in [AMS08, (7.10)]	0.1	between 0 and 1, i.e., $\in (0, 1)$
initial_Delta	initial radius	1	greater than 0
isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1

Table 5: Output notation of RTRSR1. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\\| \operatorname{grad} f(x_i)\\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
nH	the number of actions of Hessian
rho	[AMS08, (7.7)]
radius	the radius of trust region
tCGstatus	status of truncate conjugate gradient
innerIter	the number of iterations in truncate conjugate gradient
inpss	$\langle s_i, s_i \rangle$
IsUpdateHessian	Whether update Hessian approximation or not

B.3 LRTRSR1

Table 6: Input Parameters of LRTRSR1

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\\| \operatorname{grad} f(x_i)\\|$ GRAD_F_0 / 2 : $\\| \operatorname{grad} f(x_i)\\|/\\| \operatorname{grad} f(x_0)\\|$

Tolerance	Algorithm stops if "Stop_Criterion" < tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
Acceptence_Rho	Accept candidate if Rho > Acceptence_Rho	0.1	between 0 and 0.25, i.e., $\in (0, 0.25)$
Shrinked_tau	coefficient in reducing radius	0.25	between 0 and 1, i.e., $\in (0, 1)$
Magnified_tau	coefficient in increasing radius	2	greater than 1
minimum_Delta	minimum allowed radius	machine eps	greater than 0 and smaller than or equal to maximum_Delta
maximum_Delta	maximum allowed radius	10000	greater than or equal to minimum_Delta
useRand	whether use Rand in truncate conjugate gradient	false / 0	false / 0 or true / 1
Min_Inner_Iter	minimum number of iterations in truncate conjugate gradient	0	greater than or equal to ZERO and smaller than or equal to Max_Inner_Iter
Max_Inner_Iter	maximum number of iterations in truncate conjugate gradient	1000	greater than or equal to Min_Inner_Iter
theta	in [AMS08, (7.10)]	0.1	greater than or equal to 0
kappa	in [AMS08, (7.10)]	0.1	between 0 and 1, i.e., $\in (0, 1)$
initial_Delta	initial radius	1	greater than 0
isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1
LengthSY	the same as ℓ in [HGA15, Algorithm 2]	4	greater than or equal to 0

Table 7: Output notation of LRTRSR1. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations

ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
nH	the number of actions of Hessian
rho	[AMS08, (7.7)]
radius	the radius of trust region
tCGstatus	status of truncate conjugate gradient
innerIter	the number of iterations in truncate conjugate gradient
gamma	$\langle y_i, y_i \rangle / \langle s_i, y_i \rangle$
inpss	$\langle s_i, s_i \rangle$
inpsy	$\langle s_i, y_i \rangle$
inpyy	$\langle y_i, y_i \rangle$
IsUpdateHessian	Whether update Hessian approximation or not

B.4 RTRSD

Table 8: Input Parameters of RTRSD

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\\|\operatorname{grad} f(x_i)\\|$ GRAD_F_0 / 2 : $\\|\operatorname{grad} f(x_i)\\|/\\|\operatorname{grad} f(x_0)\\|$
Tolerance	Algorithm stops if "Stop_Criterion" $<$ tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
Acceptence_Rho	Accept candidate if Rho $>$ Acceptence_Rho	0.1	between 0 and 0.25, i.e., $\in (0, 0.25)$
Shrinked_tau	coefficient in reducing radius	0.25	between 0 and 1, i.e., $\in (0, 1)$
Magnified_tau	coefficient in increasing radius	2	greater than 1
minimum_Delta	minimum allowed radius	machine eps	greater than 0 and smaller than or equal to maximum_Delta
maximum_Delta	maximum allowed radius	10000	greater than or equal to minimum_Delta
useRand	whether use Rand in truncate conjugate gradient	false / 0	false / 0 or true / 1

Min_Inner_Iter	minimum number of iterations in truncate conjugate gradient	0	greater than or equal to ZERO and smaller than or equal to Max_Inner_Iter
Max_Inner_Iter	maximum number of iterations in truncate conjugate gradient	1000	greater than or equal to Min_Inner_Iter
theta	in [AMS08, (7.10)]	0.1	greater than or equal to 0
kappa	in [AMS08, (7.10)]	0.9	between 0 and 1, i.e., $\in (0, 1)$
initial_Delta	initial radius	1	greater than 0

Table 9: Output notation of RTRSD. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
nH	the number of actions of Hessian
rho	[AMS08, (7.7)]
radius	the radius of trust region
tCGstatus	status of truncate conjugate gradient
innerIter	the number of iterations in truncate conjugate gradient

B.5 RNewton

Table 10: Input Parameters of RNewton

Name of field	Interpretation	Default value C++/(Matlab,Julia)	Applicable values C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\|\operatorname{grad} f(x_i)\|$ GRAD_F_0 / 2 : $\|\operatorname{grad} f(x_i)\|/\|\operatorname{grad} f(x_0)\|$
Tolerance	Algorithm stops if "Stop_Criterion" $<$ tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result

			ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60*60*24*365$	greater than 0
LineSearch_LS	Algorithm in linesearch	ARMIJO / 0	ARMIJO / 0 : Back tracking WOLFE / 1 : [DS83, Algorithm A6.3.1mod] STRONGWOLFE / 2 : [NW06, Algorithm 3.5] EXACT / 3 : scaled BFGS INPUTFUN / 4 : Given by users
IsPureLSInput	Whether backtracking is used for step size given by users' algorithm	false / 0	false / 0 or true / 1
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Accuracy	fixed the stepsize if $\|gf_k\|/\|gf_0\| <$ accuracy	0	between 0 and 1, i.e., $\in [0, 1]$
Finalstepsize	Use this step size if $\|gf_k\|/\|gf_0\| <$ accuracy	1	all real number (negative number means the stepsize by method in "Initstepsize" is used)
LS_ratio1	coefficient in the Armijo condition	0.1	between 0 and 1, i.e., $\in (0, 1)$
LS_ratio2	coefficient in the Armijo condition	0.9	between 0 and 1, i.e., $\in (0, 1)$
Initstepsize	initial step size in first iteration	1	greater than 0
Num_pre_funs	the number of computed functions values stored for nonmonotonic linesearch	0	greater than or equal to 0
InitSteptype	Initial step size	QUADINTMOD / 3	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
useRand	whether use Rand in truncate conjugate gradient	false / 0	false / 0 or true / 1
Min_Inner_Iter	minimum number of iterations in truncate conjugate gradient	0	greater than or equal to ZERO and smaller than or equal to Max_Inner_Iter
Max_Inner_Iter	maximum number of iterations in truncate conjugate gradient	1000	greater than or equal to Min_Inner_Iter
theta	in [AMS08, (7.10)]	1	greater than or equal to 0
kappa	in [AMS08, (7.10)]	0.1	between 0 and 1, i.e., $\in (0, 1)$

Table 11: Output notation of RNewton. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
nH	the number of actions of Hessian
tCGstatus	status of truncate conjugate gradient
innerIter	the number of iterations in truncate conjugate gradient

B.6 RBroydenFamily

Table 12: Input Parameters of RBroydenFamily

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\|\operatorname{grad} f(x_i)\|$ GRAD_F_0 / 2 : $\|\operatorname{grad} f(x_i)\|/\|\operatorname{grad} f(x_0)\|$
Tolerance	Algorithm stops if "Stop_Criterion" $<$ tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LineSearch_LS	Algorithm in linesearch	ARMIJO / 0	ARMIJO / 0 : Back tracking WOLFE / 1 : [DS83, Algorithm A6.3.1mod] STRONGWOLFE / 2 : [NW06, Algorithm 3.5] EXACT / 3 : scaled BFGS INPUTFUN / 4 : Given by users

IsPureLSInput	Whether backtracking is used for step size given by users' algorithm	false / 0	false / 0 or true / 1
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Accuracy	fixed the stepsize if $\|gf_k\|/\|gf_0\| <$ accuracy	0	between 0 and 1, i.e., $\in [0, 1]$
Finalstepsize	Use this step size if $\|gf_k\|/\|gf_0\| <$ accuracy	1	all real number (negative number means the stepsize by method in "Initstepsize" is used)
LS_ratio1	coefficient in the Armijo condition	0.1	between 0 and 1, i.e., $\in (0, 1)$
LS_ratio2	coefficient in the Armijo condition	0.9	between 0 and 1, i.e., $\in (0, 1)$
Initstepsize	initial step size in first iteration	1	greater than 0
Num_pre_funs	the number of computed functions values stored for nonmonotonic linesearch	0	greater than or equal to 0
InitSteptype	Initial step size	QUADINTMOD / 3	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1
nu	the same as ϵ in [LF01, (3.2)]	10^{-4}	greater than or equal to 0 and smaller than 1
mu	the same as α in [LF01, (3.2)]	1	greater than or equal to 0

Table 13: Output notation of RBroydenFamily. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search

stepsize	the final stepsize
betay	$\alpha_i \eta_i / \mathcal{T}_{R_{\alpha_i \eta_i}}(\alpha_i \eta_i)$ see [HGA15, Step 6 of Algorithm 1]
Phic	the coefficient ϕ_i in the update [HGA15, (2.3)]
inpss	$\langle s_i, s_i \rangle$
inpsy	$\langle s_i, y_i \rangle$
IsUpdateHessian	Whether update inverse Hessian approximation or not

B.7 RWRBFGS

Table 14: Input Parameters of RWRBFGS

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\\|\operatorname{grad} f(x_i)\\|$ GRAD_F_0 / 2 : $\\|\operatorname{grad} f(x_i)\\| / \\|\operatorname{grad} f(x_0)\\|$
Tolerance	Algorithm stops if “Stop_Criterion” < tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every “OutputGap” iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every “OutputGap” iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LineSearch_LS	Algorithm in linesearch	ARMIJO / 0	ARMIJO / 0 : Back tracking WOLFE / 1 : [DS83, Algorithm A6.3.1mod] STRONGWOLFE / 2 : [NW06, Algorithm 3.5] EXACT / 3 : scaled BFGS INPUTFUN / 4 : Given by users
IsPureLSInput	Whether backtracking is used for step size given by users’ algorithm	false / 0	false / 0 or true / 1
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Accuracy	fixed the stepsize if $\\|gf_k\\|/\\|gf_0\\|$ < accuracy	0	between 0 and 1, i.e., $\in [0, 1]$

Finalstepsize	Use this step size if $\|gf_k\|/\|gf_0\| <$ accuracy	1	all real number (negative number means the stepsize by method in "Initstepsize" is used)
LS_ratio1	coefficient in the Armijo condition	0.1	between 0 and 1, i.e., $\in (0, 1)$
LS_ratio2	coefficient in the Armijo condition	0.9	between 0 and 1, i.e., $\in (0, 1)$
Initstepsize	initial step size in first iteration	1	greater than 0
Num_pre_funs	the number of computed functions values stored for nonmonotonic linesearch	0	greater than or equal to 0
InitSteptype	Initial step size	QUADINTMOD / 3	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1
nu	the same as ϵ in [LF01, (3.2)]	10^{-4}	greater than or equal to 0 and smaller than 1
mu	the same as α in [LF01, (3.2)]	1	greater than or equal to 0

Table 15: Output notation of RWRBFGS. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
inpss	$\langle s_i, s_i \rangle$
inpsy	$\langle s_i, y_i \rangle$
IsUpdateHessian	Whether update inverse Hessian approximation or not

B.8 RBFGS

Table 16: Input Parameters of RBFGS

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1

IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\|\operatorname{grad} f(x_i)\|$ GRAD_F_0 / 2 : $\|\operatorname{grad} f(x_i)\|/\|\operatorname{grad} f(x_0)\|$ PSSUBGRAD / 3 : See [LO13, Section 6.3]
Tolerance	Algorithm stops if "Stop_Criterion" $<$ tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LineSearch_LS	Algorithm in linesearch	ARMIJO / 0	ARMIJO / 0 : Back tracking WOLFE / 1 : [DS83, Algorithm A6.3.1mod] STRONGWOLFE / 2 : [NW06, Algorithm 3.5] EXACT / 3 : scaled BFGS INPUTFUN / 4 : Given by users
IsPureLSInput	Whether backtracking is used for step size given by users' algorithm	false / 0	false / 0 or true / 1
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Accuracy	fixed the stepsize if $\|gf_k\|/\|gf_0\| <$ accuracy	0	between 0 and 1, i.e., $\in [0, 1]$
Finalstepsize	Use this step size if $\|gf_k\|/\|gf_0\| <$ accuracy	1	all real number (negative number means the stepsize by method in "Initstepsize" is used)
LS_ratio1	coefficient in the Armijo condition	0.1	between 0 and 1, i.e., $\in (0, 1)$
LS_ratio2	coefficient in the Armijo condition	0.9	between 0 and 1, i.e., $\in (0, 1)$
Initstepsize	initial step size in first iteration	1	greater than 0
Num_pre_funs	the number of computed functions values stored for nonmonotonic linesearch	0	greater than or equal to 0
InitSteptype	Initial step size	QUADINTMOD / 3	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y)

			QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1
nu	the same as ϵ in [LF01, (3.2)]	10^{-4}	greater than or equal to 0 and smaller than 1
mu	the same as α in [LF01, (3.2)]	1	greater than or equal to 0
Diffx	the same as τ_x in [LO13, Section 6.3]	10^{-6}	greater than 0

Table 17: Output notation of RBFGS. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\\|\operatorname{grad} f(x_i)\\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
betay	$\alpha_i \eta_i / \mathcal{T}_{R_{\alpha_i \eta_i}}(\alpha_i \eta_i)$ see [HGA15, Step 6 of Algorithm 1]
inpss	$\langle s_i, s_i \rangle$
inpsy	$\langle s_i, y_i \rangle$
IsUpdateHessian	Whether update inverse Hessian approximation or not

B.9 LRBFGS

Table 18: Input Parameters of LRBFGS

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\\|\operatorname{grad} f(x_i)\\|$ GRAD_F_0 / 2 : $\\|\operatorname{grad} f(x_i)\\| / \\|\operatorname{grad} f(x_0)\\|$
Tolerance	Algorithm stops if "Stop_Criterion" < tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1

Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LineSearch_LS	Algorithm in linesearch	ARMIJO / 0	ARMIJO / 0 : Back tracking WOLFE / 1 : [DS83, Algorithm A6.3.1mod] STRONGWOLFE / 2 : [NW06, Algorithm 3.5] EXACT / 3 : scaled BFGS INPUTFUN / 4 : Given by users
IsPureLSInput	Whether backtracking is used for step size given by users' algorithm	false / 0	false / 0 or true / 1
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Accuracy	fixed the stepsize if $\\|gf_k\\|/\\|gf_0\\| <$ accuracy	0	between 0 and 1, i.e., $\in [0, 1]$
Finalstepsize	Use this step size if $\\|gf_k\\|/\\|gf_0\\| <$ accuracy	1	all real number (negative number means the stepsize by method in "Initstepsize" is used)
LS_ratio1	coefficient in the Armijo condition	0.1	between 0 and 1, i.e., $\in (0, 1)$
LS_ratio2	coefficient in the Armijo condition	0.9	between 0 and 1, i.e., $\in (0, 1)$
Initstepsize	initial step size in first iteration	1	greater than 0
Num_pre_funs	the number of computed functions values stored for nonmonotonic linesearch	0	greater than or equal to 0
InitSteptype	Initial step size	QUADINTMOD / 3	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1
nu	the same as ϵ in [LF01, (3.2)]	10^{-4}	greater than or equal to 0 and smaller than 1
mu	the same as α in [LF01, (3.2)]	1	greater than or equal to 0
LengthSY	the same as ℓ in [HGA15, Algorithm 2]	4	greater than or equal to 0

Table 19: Output notation of LRBFGS. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\\| \operatorname{grad} f(x_i) \\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport [7]
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
betay	$\alpha_i \eta_i / \mathcal{T}_{R_{\alpha_i \eta_i}} (\alpha_i \eta_i)$ see [HGA15, Step 6 of Algorithm 1]
rho	$1/\langle s_i, y_i \rangle$
gamma	$\langle s_i, y_i \rangle / \langle y_i, y_i \rangle$
inpss	$\langle s_i, s_i \rangle$
inpsy	$\langle s_i, y_i \rangle$
IsUpdateHessian	Whether update inverse Hessian approximation or not

B.10 RCG

Table 20: Input Parameters of RCG

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\\| \operatorname{grad} f(x_i) \\|$ GRAD_F_0 / 2 : $\\| \operatorname{grad} f(x_i) \\| / \\| \operatorname{grad} f(x_0) \\|$
Tolerance	Algorithm stops if "Stop_Criterion" < tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LineSearch_LS	Algorithm in linesearch	ARMIJO / 0	ARMIJO / 0 : Back tracking WOLFE / 1 : [DS83, Algorithm A6.3.1mod]

			STRONGWOLFE / 2 : [NW06, Algorithm 3.5] EXACT / 3 : scaled BFGS INPUTFUN / 4 : Given by users
IsPureLSInput	Whether backtracking is used for step size given by users' algorithm	false / 0	false / 0 or true / 1
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Accuracy	fixed the stepsize if $\|gf_k\|/\|gf_0\| <$ accuracy	0	between 0 and 1, i.e., $\in [0, 1]$
Finalstepsize	Use this step size if $\|gf_k\|/\|gf_0\| <$ accuracy	1	all real number (negative number means the stepsize by method in "Initstepsize" is used)
LS_ratio1	coefficient in the Armijo condition	0.1	between 0 and 1, i.e., $\in (0, 1)$
LS_ratio2	coefficient in the Armijo condition	0.9	between 0 and 1, i.e., $\in (0, 1)$
Initstepsize	initial step size in first iteration	1	greater than 0
Num_pre_funs	the number of computed functions values stored for nonmonotonic linesearch	0	greater than or equal to 0
InitSteptype	Initial step size	BBSTEP / 1	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
RCGmethod	method in choosing β in [AMS08, (8.26)]	HESTENES_STIEFEL / 2	FLETCHER_REEVES / 0 : [AMS08, (8.28)] POLAK_RIBIERE_MOD / 1 : Riemannian generalization of [NW06, (5.45)] HESTENES_STIEFEL / 2 : Riemannian generalization of [NW06, (5.46)] FR_PR / 3 : Riemannian generalization of [NW06, (5.48)] DAI_YUAN / 4 : Riemannian generalization of [NW06, (5.49)] HAGER_ZHANG / 5 : Riemannian generalization of [NW06, (5.50)]
ManDim	search direction is reset every "ManDim" iterations	machine maximum integer	greater than or equal to 0

Table 21: Output notation of RCG. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations

f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
sigma	the coefficient between grad $f(x_i)$ and $\mathcal{T}_{\alpha_i \eta_i}(\eta_i)$

B.11 RSD

Table 22: Input Parameters of RSD

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Stop_Criterion	Stopping criterion	GRAD_F_0 / 2	FUN_REL / 0 : $(f(x_{i-1}) - f(x_i))/f(x_i)$ GRAD_F / 1 : $\|\operatorname{grad} f(x_i)\|$ GRAD_F_0 / 2 : $\|\operatorname{grad} f(x_i)\|/\|\operatorname{grad} f(x_0)\|$
Tolerance	Algorithm stops if "Stop_Criterion" $<$ tolerance	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LineSearch_LS	Algorithm in linesearch	ARMIJO / 0	ARMIJO / 0 : Back tracking WOLFE / 1 : [DS83, Algorithm A6.3.1mod] STRONGWOLFE / 2 : [NW06, Algorithm 3.5] EXACT / 3 : scaled BFGS INPUTFUN / 4 : Given by users
IsPureLSInput	Whether backtracking is used for step size given by users' algorithm	false / 0	false / 0 or true / 1
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$

LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0,1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Accuracy	fixed the stepsize if $\|gf_k\|/\|gf_0\| <$ accuracy	0	between 0 and 1, i.e., $\in [0,1]$
Finalstepsize	Use this step size if $\|gf_k\|/\|gf_0\| <$ accuracy	1	all real number (negative number means the stepsize by method in "Initstepsize" is used)
LS_ratio1	coefficient in the Armijo condition	0.1	between 0 and 1, i.e., $\in (0,1)$
LS_ratio2	coefficient in the Armijo condition	0.9	between 0 and 1, i.e., $\in (0,1)$
Initstepsize	initial step size in first iteration	1	greater than 0
Num_pre_funs	the number of computed functions values stored for nonmonotonic linesearch	0	greater than or equal to 0
InitSteptype	Initial step size	BBSTEP / 1	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]

Table 23: Output notation of RSD. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize

B.12 RBFGSSub

Table 24: Input Parameters of RBFGSLPSub

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1

IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Tolerance	Algorithm stops if $\|gf\|_P <$ tolerance and Eps equals Min_Eps	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Initstepsize	initial step size in first iteration	1	greater than 0
InitSteptype	Initial step size	ONESTEP / 0	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1
lambdaLower	λ in [HHY18]	10^{-2}	greater than 0 and smaller than lambdaUpper
lambdaUpper	Λ in [HHY18]	10^{2}	greater than lambdaLower
Eps	ϵ in [HHY18]	1	in $(0, 1)$
Theta_eps	θ_δ in [HHY18]	0.01	in $(0, 1)$
Min_Eps	lower bound of ϵ	10^{-6}	in $(0, 1)$
Del	δ in [HHY18]	1	in $(0, 1)$
Theta_del	θ_δ in [HHY18]	0.01	in $(0, 1)$

Table 25: Output notation of RBFGSLPSub. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\|\operatorname{grad} f(x_i)\|$
time	computational time (second)
nf	the number of function evaluations

ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
betay	$\alpha_i \eta_i / \mathcal{T}_{R_{\alpha_i \eta_i}}(\alpha_i \eta_i)$ see [HGA15, Step 6 of Algorithm 1]
inpss	$\langle s_i, s_i \rangle$
inpsy	$\langle s_i, y_i \rangle$
IsUpdateHessian	Whether update inverse Hessian approximation or not
nsubprob	The number of solving quadratic programming problem

B.13 LRBFGSSub

Table 26: Input Parameters of RBFGSLPSub

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Tolerance	Algorithm stops if $\|gf\|_P <$ tolerance and Eps equals Min_Eps	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every “OutputGap” iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every “OutputGap” iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Initstepsize	initial step size in first iteration	1	greater than 0
InitSteptype	Initial step size	ONESTEP / 0	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]

isconvex	whether the cost function is convex	false / 0	false / 0 or true / 1
lambdaLower	λ in [HHY18]	10^{-2}	greater than 0 and smaller than lambdaUpper
lambdaUpper	Λ in [HHY18]	10^{2}	greater than lambdaLower
Eps	ϵ in [HHY18]	1	in $(0, 1)$
Theta_eps	θ_δ in [HHY18]	0.01	in $(0, 1)$
Min_Eps	lower bound of ϵ	10^{-6}	in $(0, 1)$
Del	δ in [HHY18]	1	in $(0, 1)$
Theta_del	θ_δ in [HHY18]	0.01	in $(0, 1)$
LengthSY	The same as ℓ in [HGA15, Algorithm 2]	2	greater than or equal to 0

Table 27: Output notation of RBFGSLPSub. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\\| \operatorname{grad} f(x_i) \\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
betay	$\alpha_i \eta_i / \mathcal{T}_{R_{\alpha_i \eta_i}}(\alpha_i \eta_i)$ see [HGA15, Step 6 of Algorithm 1]
inpss	$\langle s_i, s_i \rangle$
inpsy	$\langle s_i, y_i \rangle$
IsUpdateHessian	Whether update inverse Hessian approximation or not
nsubprob	The number of solving quadratic programming problem

B.14 RGS

Table 28: Input Parameters of RBFGSLPSub

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Tolerance	Algorithm stops if $\\|gf\\|_P <$ tolerance and Eps equals Min_Eps	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration

OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60*60*24*365$	greater than 0
LS_alpha	coefficient in the Wolfe first condition	0.0001	between 0 and 0.5, i.e. $\in (0, 0.5)$
LS_beta	coefficient in the Wolfe second condition	0.999	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	machine eps	greater than 0 and smaller than or equal to Maxstepsize
Maxstepsize	maximum allowed step size	1000	greater than or equal to Minstepsize
Initstepsize	initial step size in first iteration	1	greater than 0
InitSteptype	Initial step size	ONESTEP / 0	ONESTEP / 0 : use one BBSTEP / 1 : g(s, s) / g(s, y) QUADINT / 2 : [NW06, (3.60)] QUADINTMOD / 3 : [NW06, page 60]
Eps	ϵ in [HHY18]	1	in $(0, 1)$
Theta_eps	θ_δ in [HHY18]	0.01	in $(0, 1)$
Min_Eps	lower bound of ϵ	10^{-6}	in $(0, 1)$
Del	δ in [HHY18]	1	in $(0, 1)$
Theta_del	θ_δ in [HHY18]	0.01	in $(0, 1)$

Table 29: Output notation of RBFGSLPSub. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
gf	$\\| \operatorname{grad} f(x_i) \\|$
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
LSstatus	status of line search result
initslope	initial slope in line search
newslope	the slope of final point in line search
initstepsize	initial step size in line search
stepsize	the final stepsize
nsubprob	The number of solving quadratic programming problem

B.15 ManPG

Table 30: Input Parameters of ManPG

Name of field	Interpretation	Default value	Applicable values
		C++/(Matlab,Julia)	C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Tolerance	Algorithm stops if $\|gf\|_P <$ tolerance and Eps equals Min_Eps	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
Stop_Criterion	Stopping criterion	DIR_F_0 / 2	FUN_REL / 0 : $\|f(x_{i-1}) - f(x_i)\|/(\|f(x_i)\| + 1)$ DIR_F / 1 : $\|\eta_{x_i}\|$ DIR_F_0 / 2 : $\|\eta_{x_i}\|/\|\eta_{x_0}\|\|$
RPGLSVariant	Adaptive or fixed L, see [CMMCSZ20]	ADALIPSCHITZ / 1	REGULAR / 0 : fixed L ADALIPSCHITZ / 1 : adaptive L
LS_alpha	coefficient in the Armijo condition	0.0001	between 0 and 1, i.e. $\in (0, 1)$
LS_ratio	shrinking parameter in line search	0.1	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	0.02	between 0 and 1, i.e., $\in (0, 1]$

Table 31: Output notation of ManPG. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
—nd—	$\|\eta_{x_i}\|$
t0	initial step size
t	accepted step size
s0	initial slope
s	slope at accepted step size
time	computational time (second)
nf	the number of function evaluations
ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
SMlambda	tolerance in CG of semi-smooth Newton
SMtol	tolerance of semi-smooth Newton
SMiter	number of iterations in semi-smooth Newton
SMCGiter	total number of CG iterations in semi-smooth Newton

B.16 AManPG

Table 32: Input Parameters of AManPG

Name of field	Interpretation	Default value C++/(Matlab,Julia)	Applicable values C++/(Matlab,Julia) : interpretation
IsCheckParams	output parameters of Solvers	Matlab and Julia only : 0	0 or 1
IsCheckGradHess	Check the correctness of gradient and Hessian	Matlab and Julia only : 0	0 or 1
Tolerance	Algorithm stops if $\|gf\|_P <$ tolerance and Eps equals Min_Eps	10^{-6}	greater than 0
Min_Iteration	minimum number of iterations	0	greater than or equal to 0 and smaller than or equal to Max_Iteration
Max_Iteration	maximum number of iterations	500	greater than or equal to Min_Iteration
OutputGap	Output every "OutputGap" iterations	1	greater than or equal to 1
Verbose	output information	ITERRESULT / 2	NOOUTPUT / 0 : no output FINALRESULT / 1 : Only final result ITERRESULT / 2 : Output every "OutputGap" iterations DETAILED / 3: Output Detailed information
TimeBound	maximum computational time	$60 * 60 * 24 * 365$	greater than 0
Stop_Criterion	Stopping criterion	DIR_F_0 / 2	FUN_REL / 0 : $\|f(x_{i-1}) - f(x_i)\|/(\|f(x_i)\| + 1)$ DIR_F / 1 : $\|\eta_{x_i}\|$ DIR_F_0 / 2 : $\|\eta_{x_i}\|/\|\eta_{x_0}\|\|\|$
RPGLSVariant	Adaptive or fixed L, see [CMMCSZ20]	ADALIPSCHITZ / 1	REGULAR / 0 : fixed L ADALIPSCHITZ / 1 : adaptive L
LS_alpha	coefficient in the Armijo condition	0.0001	between 0 and 1, i.e. $\in (0, 1)$
LS_ratio	shrinking parameter in line search	0.1	between 0 and 1, i.e., $\in (0, 1)$
Minstepsize	minimum allowed step size	0.02	between 0 and 1, i.e., $\in (0, 1]$
SGIterGap	check safeguard every "SGIterGap" iterations	5	a positive iteger

Table 33: Output notation of AManPG. Note that the first time an action of a vector transport $\mathcal{T}_\eta$ is computed will usually have higher complexity than subsequent times. Specifically, if $\mathcal{T}_\eta \xi_1$ has been computed, then evaluating $\mathcal{T}_\eta \xi_2$ usually can use some results from computations of $\mathcal{T}_\eta \xi_1$. nV denotes the number of evaluations of vector transport first time. nVp denotes the number of other times.

Notation	Interpretation
i	the number of iterations
f	function value
df/f	$(f(x_{i-1}) - f(x_i))/f(x_i)$
—nd—	$\|\eta_{x_i}\|$
t0	initial step size
t	accepted step size
s0	initial slope
s	slope at accepted step size
time	computational time (second)
nf	the number of function evaluations

ng	the number of gradient evaluations
nR	the number of retraction evaluations
nV/nVp	the number of actions of vector transport
SMlambda	tolerance in CG of semi-smooth Newton
SMtol	tolerance of semi-smooth Newton
SMiter	number of iterations in semi-smooth Newton
SMCGiter	total number of CG iterations in semi-smooth Newton

C Manifold Parameters

Table 34: Parameters for Matlab and Julia. An example can be found in Lines 11 to 13 of Listing 3.

Manifolds	Name of field	Applicable values
Quotient manifold $\mathbb{C}_*^{m\times p} \times \mathbb{C}_*^{n\times p}/\mathrm{GL}(p) \simeq \mathbb{C}_p^{m\times n}$	name	'CFixedRankQ2F'
	m	positive integer
	n	positive integer
	p	positive integer
Complex Stiefel manifold $\mathrm{CSt}(p, n) = \{X \in \mathbb{C}^{n\times p} \mid X^H X = I_p\}$	name	'CStiefel'
	n	positive integer
	p	positive integer
	ParamSet	see Table 35
Quotient manifold $\mathbb{C}_*^{n\times p}/U_p \simeq \mathcal{S}_p^{n\times n}$	name	'CSymFixedRankQ'
	n	positive integer
	p	positive integer
	ParamSet	see Table 36
Euclidean space $\mathbb{R}^{m\times n}$	name	'Euclidean'
	m	positive integer
	n	positive integer
Fixed rank manifold $\mathbb{R}_p^{m\times n}$	name	'FixedRankE'
	m	positive integer
	n	positive integer
	p	positive integer
Quotient manifold $\mathbb{R}_*^{m\times p} \times \mathbb{R}_*^{n\times p}/\mathrm{GL}(p) \simeq \mathbb{R}_p^{m\times n}$	name	'FixedRankQ2F'
	m	positive integer
	n	positive integer
	p	positive integer
Grassmann manifold: $Gr(p, n)$	name	'Grassmann'
	n	positive integer
	p	positive integer
The manifold of symmetric positive definite matrices $\mathbb{S}_n$	name	'SPDManifold'
	n	positive integer
	ParamSet	see Table 37
Unit sphere $\mathbb{S}^n = \{x \in \mathbb{R}^n \mid x^T x = 1\}$	name	'Sphere'
	n	positive integer
	ParamSet	see Table 38
Stiefel manifold $\mathrm{St}(p, n) = \{X \in \mathbb{R}^{n\times p} \mid X^T X = I_p\}$	name	'Stiefel'
	n	positive integer
	p	positive integer and smaller than or equal to n
	ParamSet	see Table 39
Quotient manifold $\mathbb{R}_*^{n\times p}/O_p \simeq \mathcal{S}_p^{n\times n}$	name	'SymFixedRankQ'
	n	positive integer
	p	positive integer
	ParamSet	see Table 40

Table 35: The complex Stiefel manifold

Matlab ParamSet value	C++ Member function	Parameters	Values
1	ChooseParamsSet1() (default)	Metric	Euclidean
		Retraction	qf retraction [AMS08, (4.8)]
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
2	ChooseParamsSet2()	Metric	Euclidean
		Retraction	qf retraction [AMS08, (4.8)]
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	no
3	ChooseParamsSet3()	Metric	Euclidean
		Retraction	polar retraction [AMS08, (4.7)]
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
4	ChooseParamsSet4()	Metric	Euclidean
		Retraction	polar retraction [AMS08, (4.7)]
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	no

Table 36: Quotient manifold $\mathbb{C}_*^{n\times p}/U_p$ which is diffeomorphic to the set of fixed rank Hermitian matrices $\mathcal{S}_p^{n\times n}$

Matlab ParamSet value	C++ Member function	Parameters	Values
1	ChooseParamsSet1() (default)	Metric	Euclidean metric of $\mathcal{S}_p^{n\times n}$
		Retraction	by projection
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
2	ChooseParamsSet2()	Metric	metric in [HGZ17]
		Retraction	by projection
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
3	ChooseParamsSet3()	Metric	Euclidean metric of $\mathcal{S}_p^{n\times n}$
		Retraction	by projection
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	yes
4	ChooseParamsSet4()	Metric	metric in [HGZ17]
		Retraction	by projection
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	yes

Table 37: The manifold of symmetric positive definite matrices $\mathbb{S}_n$

Matlab ParamSet value	C++ Member function	Parameters	Values
1	ChooseParamsSet1() (default)	Metric	Affine invariance
		Retraction	A second order retraction in [YHAG19]
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
2	ChooseParamsSet2()	Metric	Euclidean
		Retraction	A second order retraction in [YHAG19]
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
3	ChooseParamsSet3()	Metric	Affine invariance
		Retraction	Exponential mapping of AF metric
		Vector transport	parallel translation of AF metric
		Use intrinsic approach [Hua13, §9.5]	no
4	ChooseParamsSet4()	Metric	Euclidean
		Retraction	Exponential mapping of AF metric
		Vector transport	parallel translation of AF metric
		Use intrinsic approach [Hua13, §9.5]	no

Table 38: Unit sphere

Matlab ParamSet value	C++ Member function	Parameters	Values
1	ChooseStieParamsSet1() (default)	Metric	Euclidean
		Retraction	qf retraction [AMS08, (4.8)]
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
2	ChooseSphereParamsSet2()	Metric	Euclidean
		Retraction	exponential mapping [AMS08, (5.25)]
		Vector transport	parallel translation [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	no
3	ChooseSphereParamsSet3()	Metric	Euclidean
		Retraction	qf retraction [AMS08, (4.8)]
		Vector transport	parallel translation [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	no
4	ChooseSphereParamsSet4()	Metric	Euclidean
		Retraction	qf retraction [AMS08, (4.8)]
		Vector transport	parallel translation [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	no

Table 39: The Stiefel manifold

Matlab ParamSet value	C++ Member function	Parameters	Values
1	ChooseParamsSet1() (default)	Metric	Euclidean
		Retraction	qf retraction [AMS08, (4.8)]
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
2	ChooseParamsSet2()	Metric	Euclidean
		Retraction	qf retraction [AMS08, (4.8)]
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	no
3	ChooseParamsSet3()	Metric	Euclidean
		Retraction	polar retraction [AMS08, (4.7)]
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
4	ChooseParamsSet4()	Metric	Euclidean
		Retraction	polar retraction [AMS08, (4.7)]
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	no
5	ChooseParamsSet5()	Metric	Euclidean
		Retraction	Cayley retraction [Zhu17]
		Vector transport	Cayley vector transport [Zhu17]
		Use intrinsic approach [Hua13, §9.5]	no

Table 40: Quotient manifold $\mathbb{R}_*^{n\times p}/U_p$ which is diffeomorphic to the set of fixed rank symmetric matrices $\mathcal{S}_p^{n\times n}$

Matlab ParamSet value	C++ Member function	Parameters	Values
1	ChooseParamsSet1() (default)	Metric	Euclidean metric of $\mathcal{S}_p^{n\times n}$
		Retraction	by projection
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
2	ChooseParamsSet2()	Metric	metric in [HGZ17]
		Retraction	by projection
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
3	ChooseParamsSet2()	Metric	Euclidean metric of $\mathbb{R}^{n\times p}$
		Retraction	by projection
		Vector transport	by parallelization [HAG15, (2.3.1)]
		Use intrinsic approach [Hua13, §9.5]	yes
4	ChooseParamsSet3()	Metric	Euclidean metric of $\mathcal{S}_p^{n\times n}$
		Retraction	by projection
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	yes
5	ChooseParamsSet4()	Metric	metric in [HGZ17]
		Retraction	by projection
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	yes
6	ChooseParamsSet4()	Metric	Euclidean metric of $\mathbb{R}^{n\times p}$
		Retraction	by projection
		Vector transport	by projection
		Use intrinsic approach [Hua13, §9.5]	yes

References

[ABG07] P.-A. Absil, C. G. Baker, and K. A. Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007.

[AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, Princeton, NJ, 2008.

[CMMCSZ20] Shixiang Chen, Shiqian Ma, Anthony Man-Cho So, and Tong Zhang. Proximal gradient method for nonsmooth optimization over the Stiefel manifold. *SIAM Journal on Optimization*, 30(1):210–239, 2020.

[DS83] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Springer, New Jersey, 1983.

[HAG15] W. Huang, P.-A. Absil, and K. A. Gallivan. A Riemannian symmetric rank-one trust-region method. *Mathematical Programming*, 150(2):179–216, February 2015.

[HGA15] W. Huang, K. A. Gallivan, and P.-A. Absil. A Broyden Class of Quasi-Newton Methods for Riemannian Optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015.

[HGZ17] Wen Huang, K. A. Gallivan, and Xiangxiong Zhang. Solving PhaseLift by low rank Riemannian optimization methods for complex semidefinite constraints. *SIAM Journal on Scientific Computing*, 39(5):B840–B859, 2017.

[HHY18] S. Hosseini, W. Huang, and R. Yousefpour. Line search algorithms for locally Lipschitz functions on Riemannian manifolds. *SIAM Journal on Optimization*, 28(1):596–619, 2018.

[HU16] S. Hosseini and A. Uschmajew. A Riemannian gradient sampling algorithm for nonsmooth optimization on manifolds. *Institut für Numerische Simulation*, page INS Preprint No. 1607, 2016.

[Hua13] W. Huang. *Optimization algorithms on Riemannian manifolds with applications*. PhD thesis, Florida State University, Department of Mathematics, 2013.

[HW19] W. Huang and K. Wei. Extending FISTA to Riemannian optimization for sparse PCA. arXiv:1909.05485, 2019.

[LF01] D.-H. Li and M. Fukushima. On the global convergence of the BFGS method for nonconvex unconstrained optimization problems. *SIAM Journal on Optimization*, 11(4):1054–1064, January 2001. doi:10.1137/S1052623499354242.

[LO13] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming*, 141(1-2):135–163, February 2013. doi:10.1007/s10107-012-0514-2.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[RW12] W. Ring and B. Wirth. Optimization methods on Riemannian manifolds and their application to shape space. *SIAM Journal on Optimization*, 22(2):596–627, January 2012. doi:10.1137/11082885X.

[SI13] H. Sato and T. Iwai. A Riemannian optimization approach to the matrix singular value decomposition. *SIAM Journal on Optimization*, 23(1):188–212, 2013.

[YHAG19] Xinru Yuan, Wen Huang, P.-A. Absil, and K. A. Gallivan. Computing the matrix geometric mean: Riemannian vs Euclidean conditioning, implementation techniques, and a Riemannian BFGS method. Technical Report UCL-INMA-2019.05, U.C.Louvain, 2019.

[Zhu17] X. Zhu. A Riemannian conjugate gradient method for optimization on the Stiefel manifold. *Computational Optimization and Applications*, 67(1):73–110, 2017.