GSOC 2017 MRPT: Project #8

Implement a state-of-the-art graph-slam solver, SE-Sync

Jesus Briales
University of Malaga
jbriales@uma.es

Jose-Luis Blanco
University of Almeria
jlblanco@ual.es

1. Preamble

Project #8 is a challenging as well as rewarding project. Because of this, we consider necessary for the willing participants to prove a minimum knowledge and performance matching the required basis to build upon in this project.

The candidates are expected to handle:

- Optimization within ROPTLIB.
- More traditional linear algebra: Solving large sparse linear systems and eigenvalue problems.

Because of this, we propose the subsequent assessment exercises. The task in Section 2 is compulsory if you want to opt to this project, whereas the tasks in Section 3 are optional to give you further space for proving your expertise.

Note that these tasks are essential building blocks within Project #8, so their completion brings you closer to the final goal of the project. Keep also in mind that aspects such as efficiency and clarity of the code will be taken into account for selection. So, do your best! :)

For questions regarding this project and these assessment exercises you may contact any of the mentors.

1.1. Exercises data

We provide several data matrices Q at `http://mapir.isa.uma.es/jbriales/GSOC_project8_data.zip` that will serve as input for the proposed tasks. These matrics are positive-semidefinite $(d \times d)$-*block-sparse* $(n \times n)$-matrices, yielding an overall $nd \times nd$ dimension of the complete matrix.

Each matrix is given as a list of (i, j, val) triplets stored in a text file, whose name `<dataset>_d<d>_n<n>.txt` indicates the dimension parameters n and d for the problem as well. Note that n and d vary for the different cases.

In order to load the matrix into the C++ program we suggest that you read the triplets from the text files and use then to define an `Eigen::SparseMatrix` object[1].

2. Riemannian optimization within ROPTLIB

The problem we want you to solve is

$$f^\star = \min_{X \in \mathcal{M}} \frac{1}{2} \operatorname{tr}(X^\top Q X), \tag{1}$$

where the optimization domain is

$$\underset{\substack{[d \times p] \\ (n \times 1)}}{\underline{X}} \in \operatorname{stack}(St(p,d)^n)^\top \equiv \Big[\underbrace{St(p,d) \ \cdots \ St(p,d)}_{\times n} \Big]^\top \subset \mathbb{R}^{nd \times p}.$$

[1] Check the complete tutorial or the Quick Reference for Sparse Matrices in Eigen.

From its definition above, the variable X lies within a manifold characterized as the stacked representation of the product of n Stiefel manifolds[2]: $St(p,d) \equiv \{V \in \mathbb{R}^{p\times d} : V^\top V = I_d\}$. This justifies the interest in using a library tailored to optimization on manifolds such as ROPTLIB. Note we introduce a use-specified integer parameter $p \geq d$ which will be essential in the SE-Sync approach. We also ask you to solve the problem above employing the Riemannian trust-regions method, which is already available as an option in ROPTLIB.

So, overall, the outcome of this exercise should be a working executable that receives the matrix file for Q and the dimension parameters n, d and p. There should be also an optional input X_0 to be used as initialization:

```
$ solve_rotsync <Q_file_path> <n> <d> <p> [X0_file_path]
```

If X_0 is not provided, you should generate it randomly as a *valid point* inside the specified manifold $\mathscr{M}$ (this should be straightforward in ROPTLIB). The output X should be saved into a file.

Some related and useful references are the C++ examples in the User Manual of ROPTLIB. We strongly recommend the use of the tools provided by ROPTLIB to check correctness of the defined problem!

3. Linear algebra

3.1. Solving a linear system via Cholesky decomposition

Solve the linear problem

$$QX = B \tag{2}$$

where the data matrices $Q \in \mathbb{S}^{nd}$ are the same as for the prior exercise, $B \in \mathbb{R}^{nd\times p}$ is a $nd \times p$ matrix (p a user parameter) and $X \in \mathbb{R}^{nd\times p}$ is the unknown to solve. We encourage you to solve the problem using some kind of decomposition, *e.g.* the Cholesky decomposition, in order to solve *many* problems with the same Q. An advisable C++ library for performing this kind of linear algebra is Eigen (already lying at the core of MRPT).

The expected output is an application that reads the matrix file for Q and the dimension parameter p. There should be also an optional input B:

```
$ solve_linsys <Q_file_path> <p> [B_file_path]
```

If B is not provided, you should generate it as a random $nd \times p$ matrix. The output X should be saved into a file.

3.2. Solving eigenvectors

Find the eigenvector corresponding to the smallest eigenvalue of the matrices Q. For this purpose, note that Q is symmetric and positive semidefinite. This allows to find the smallest eigenvalue as the most negative one (see *e.g.* the `eigs` documentation in Matlab).

Since we are interested only in the smallest eigenvalue and the scalability of the problem (Q may become really large) we discourage the computation of a complete eigenvalue decomposition. We suggest you use instead some dedicated library for large scale eigenvalue problems, *e.g.* the traditional ARPACK, its C++ interface ARPACK++, or maybe the more recent Eigen-based (and thus purely C++) Spectra library. Alternative (efficient) proposals are accepted.

The expected output is an application that reads the matrix file and returns the minimum eigenvalue in the screen (saving the corresponding eigenvector to a file):

```
$ solve_mineig <Q_file_path>
```

[2]The Stiefel manifold $St(p,d)$ is the set of $p \times d$ matrices whose columns are orthonormal.