Solving Large Scale Cubic Regularization by a Generalized Eigenvalue Problem

Felix Lieder

04.10.2019

Mathematisches Institut, Heinrich-Heine-Universität , D–40225 Düsseldorf, Germany,
✉ lieder@opt.uni-duesseldorf.de

Abstract. Cubic Regularization methods have several favorable properties. In particular under mild assumptions, they are globally convergent towards critical points with second order necessary conditions satisfied. Their adoption among practitioners, however, does not yet match the strong theoretical results. One of the reasons for this discrepancy may be additional implementation complexity needed to solve the occurring sub-problems. In this paper we show that this complexity can be essentially eliminated by reducing the sub-problem to a generalized eigenvalue problem. The resulting algorithm is not only robust, due to existing highly advanced eigenvalue solvers, but also provides a new way of employing second order methods in the large scale case.

Key words: Cubic Regularization, Generalized Eigenvalue Problem, Large Scale

1 Introduction

We consider the cubic regularization problem

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; g^T s + \tfrac{1}{2} s^T H s + \tfrac{\sigma}{3}||s||_2^3 \tag{1}$$

for some vector $g \in \mathbb{R}^n$, symmetric matrix $H = H^T \in \mathbb{R}^{n\times n}$ and positive $\sigma > 0$. Throughout this paper, the Euclidean norm of a vector $s \in \mathbb{R}^n$ is denoted by $||s||_2 := \sqrt{s^T s}$, where s^T denotes the transpose of s. Although (1) is generally not convex, its infimum is always attained by some global minimizer. Specifically it was shown in [3] (Theorem 3.1) that the condition

$$(H + \sigma||s_*||_2 I)s_* = -g \tag{2}$$

and the semidefinite inequality

$$H + \sigma||s_*||_2 I \succeq 0 \tag{3}$$

are necessary and sufficient for $s_* \in \mathbb{R}^n$ to be a globally optimal solution of (1). Note that (2) is equivalent to a zero gradient of the objective function in (1), but (3) is generally strictly stronger than the standard second order necessary condition of requiring a positive semidefinite Hessian of the objective in (1) (which would, for non-zero $s \in \mathbb{R}^n$, be equal to $H + \sigma||s||_2 I + \sigma\frac{ss^T}{||s||_2}$). The main interest in Problem (1) comes from unconstrained optimization, where it is used to compute search directions. Essentially, this approach was proposed first by Griewank in [9], but only gained wide attention due to Nesterov and Polyak [12], who showed that in particular cubic regularization of Newtons method (see Algorithm A1 below) enjoys global convergence to second order critical points under mild conditions.

Algorithm A1 Cubic regularization of Newton method

1. **Input:** $x_0 \in \mathbb{R}^n$, twice differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ with L-Lipschitz Hessian
2. **Compute:** A solution $s_k \in \mathbb{R}^n$ of (1) with $g = \nabla f(x_k)$ and $H = \nabla^2 f(x_k)$ and $\sigma = \frac{L}{2}$
3. **Set** $x_{k+1} := x_k + s_k$, set $k := k + 1$ and go to 2.

Then Cartis, Gould and Toint [3],[4] generalized the earlier results to an adaptive setting, which may have spiked interest among researchers even further: Since then, the framework has been extended with iterative [2], block [7], accelerated [13], stochastic [15] and numerous more variants. Their adoption among practitioners however, seems to be rather slow. The two main reasons may be the following: First the difficulty of implementing an accurate solver for (1) and second limitations of existing implementations when dealing with large scale problems. Here, we will address both issues by showing that $s_* \in \mathbb{R}^n$ solving (1), can be easily calculated via a generalized eigenvalue problem, where we only require one (specific) eigenvalue to be found. Moreover, our approach is easy to implement and well suitable for large scale problems by using an approximate iterative generalized eigenvalue solver. In particular only (typically few) matrix vector products Hv are employed and therefore the matrix H is not required to be explicitly available. This work was motivated by recent similar results for trust region methods [1]. In section 2 we will present a basic version of the algorithm, show its well definedness in section 3, discuss some implementation details in section 4, present numerical experiments in section 5 and conclude in section 6.

2 Generalized Eigenvalue Problem

We define the symmetric matrix pencil $M : \mathbb{C} \to \mathbb{C}^{2(n+1)\times 2(n+1)}$ via

$$M(\lambda) := \underbrace{\begin{pmatrix} 0 & 0 & 0 & -g^T \\ 0 & \sigma I & 0 & -H \\ 0 & 0 & \sigma & 0 \\ -g & -H & 0 & 0 \end{pmatrix}}_{\in \mathbb{R}^{2(n+1)\times 2(n+1)}} - \lambda \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & I \\ 1 & 0 & 0 & 0 \\ 0 & I & 0 & 0 \end{pmatrix}}_{\in \mathbb{R}^{2(n+1)\times 2(n+1)}} \tag{4}$$

where $I \in \mathbb{R}^{n\times n}$ denotes the identity matrix. Let us now consider the generalized eigenvalue problem $det(M(\lambda)) \overset{!}{=} 0$, where det denotes the determinant. Recall that $\lambda_* \in \mathbb{C}$ is called an eigenvalue of the matrix pencil M if it satisfies $det(M(\lambda_*)) = 0$. Likewise $v \in \mathbb{C}^{2(n+1)}$ is called an eigenvector of M if $M(\lambda_*)v = 0$ is satisfied. Note that $det(M(\lambda))$ is a polynomial of exactly degree $2(n + 1)$ and will therefore have the same number of (possibly complex) roots, i.e. eigenvalues. Specifically we will show that the globally optimal solutions of (1) can be derived from the largest real eigenvalue of M and its corresponding eigenvector.

2.1 Basic Algorithm

Here we present Algorithm A2 for globally solving (1) via a generalized eigenvalue decomposition. In Step 2 of Algorithm A2 we require the main computational effort of finding the largest generalized eigenvalue and corresponding eigenvector. This may be performed by iterative methods (which are very well developed, see e.g. [10]) and can be much more efficient than existing full methods, especially when the matrix $H \in \mathbb{R}^{n\times n}$ is large and sparse. In the unlikely "hard case" in Step 4, we can again employ an iterative method, such as MINRES-QLP [5], to compute the auxiliary vector d.

Algorithm A2 Minimization of Cubic Regularization via Generalized Eigenvalue Problem

1. **Input:** $g \in \mathbb{R}^n$, $H = H^T \in \mathbb{R}^{n\times n}$, $\sigma > 0$
2. **Compute:** The largest real generalized eigenvalue $\lambda_* \geq 0$ and corresponding eigenvector $v = \begin{pmatrix} v_1 & v_2^T & v_3 & v_4^T \end{pmatrix}^T$ with $v_1, v_3 \in \mathbb{R}$ and $v_2, v_4 \in \mathbb{R}^n$ such that
$$\begin{pmatrix} 0 & 0 & 0 & -g^T \\ 0 & \sigma I & 0 & -H \\ 0 & 0 & \sigma & 0 \\ -g & -H & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda_* \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & I \\ 1 & 0 & 0 & 0 \\ 0 & I & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$
3. **If** $v_1 \neq 0$ **(Generic Case):** Set $s_* := \frac{v_2}{v_1}$
4. **Else** $v_1 = 0$ **(Hard Case):** Set $s_* := d + \frac{v_4}{||v_4||_2}\sqrt{(\frac{\lambda_*}{\sigma})^2 - ||d||_2^2}$ for $d := -(H + \lambda_* I)^+ g$.
5. **Output:** $s_* \in \mathbb{R}^n$ a global solution to $\min_{s\in\mathbb{R}^n} g^T s + \frac{1}{2} s^T H s + \frac{\sigma}{3}||s||_2^3$

Our main result is following statement.

Theorem 2.1. *Algorithm A2 is well-defined.*

The proof of Theorem 2.1 is treated in the next section and its subsections. In particular we will show that Algorithm A2 is well defined in infinite precision. In order to make Algorithm A2 stable in finite precision; i.e. accounting for rounding errors or, more broadly, an inexact execution; steps 3 and 4 require an appropriate modification. Specifically in the generic case, we can avoid rounding errors by computing

$$s_* := -\frac{\operatorname{sign}(g^T v_4)\lambda_*}{\sigma} \frac{v_2}{||v_2||_2} \tag{5}$$

instead. Here, similar to [1], $g^T v_4 = 0$ or $v_2 = 0$ correspond to either the hard case or the case $s_* = 0$. In the hard case it is advisable to use a stable formula to find $t_* \in \mathbb{R}$ that solves $||d + t_* v_4||_2 = \frac{\lambda}{\sigma}$ (In finite precision the root with smaller absolute value is preferable.) and then set $s_* := d + t_* v_4$. This may be necessary, because in finite precision $|d^T v_4| = \epsilon > 0$ can occur, while in infinite precision $d \perp v_4$ holds true, making it equivalent to our formula in step 4 . In our implementation actually distinguishing between the generic and the hard (or "almost hard ") case is done by considering the quantity $\frac{|g^T v_4|}{||g||_2 ||v_4||_2}$. More precisely if $|g^T v_4| \leq \delta ||g||_2 ||v_4||_2$ is satisfied for some (small) tolerance $\delta > 0$, then it is treated as the hard case, otherwise as the generic case. For convenience Algorithm A3, the resulting stabilized version of Algorithm A2, is detailed in subsection 4.3 below.

3 Correspondence

In this section we will show that Algorithm A2 is well defined. This will be done in multiple parts. A correspondence between critical points of (1) and real eigenvectors (and therefore real eigenvalues) of (4) is proven first. Next we will show how the largest real eigenvalue of M and its associated eigenvector gives rise to an optimal solution of (1).

3.1 Constructing Real Eigenvectors from Critical Points

We start by showing that critical points of (1), i.e. points that satisfy (2), natively correspond to a subset of eigenvalues and eigenvectors of (4). Recall that (1) is always solvable, implying that at least one critical point exists. Thus the following result implies in particular that there exists at least one pair of real eigenvalue and eigenvector of (4).

Lemma 3.1. *If $s \in \mathbb{R}^n$ satisfies (2), i.e. $(H + \sigma ||s||_2 I)s = -g$, then $\lambda = \sigma ||s||_2$ is an eigenvalue of (4), i.e. $det(M(\sigma||s||_2)) = 0$. As a consequence M has at least one non-negative eigenvalue; the corresponding eigenvector may be chosen as a real vector.*

Proof. Let $s \in \mathbb{R}^n$ satisfy $(H + \sigma||s||_2 I)s = -g$. We then distinguish two cases.
Case $det(H + \sigma||s||_2 I) = 0$: There exists $0 \neq d \in \mathbb{R}^n$ with $(H + \sigma||s||_2 I)d = 0$. Furthermore $-g^T d = s^T (H + \sigma||s||_2 I)d = 0$ holds true and therefore $v := \begin{pmatrix} 0 & 0^T & 0 & d \end{pmatrix}^T \neq 0$ satisfies

$$M(\sigma||s||_2)v = \begin{pmatrix} -g^T d \\ -(H + \sigma||s||_2 I)d \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{6}$$

implying that v is an eigenvector of M with eigenvalue $\lambda = \sigma||s||_2$.

Case $det(H + \sigma||s||_2 I) \neq 0$: There exists $d \in \mathbb{R}^n$ which satisfies $(H + \sigma||s||_2)d = \sigma s$. Together with $(H + \sigma||s||_2 I)s = -g$ this shows that

$$v := \begin{pmatrix} 1 & s^T & ||s||_2 & d^T \end{pmatrix}^T \neq 0$$

satisfies

$$M(\sigma||s||_2)v = \begin{pmatrix} -\sigma||s||_2^2 - g^T d \\ \sigma s - (H + \sigma||s||_2)d \\ \sigma||s||_2 - \sigma||s||_2 \\ -g - (H + \sigma||s||_2)s \end{pmatrix} = \begin{pmatrix} -\sigma||s||_2^2 + s^T(H + \sigma||s||_2)d \\ \sigma s - \sigma s \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{7}$$

again implying that v is an eigenvector of M with eigenvalue $\lambda = \sigma||s||_2$. This concludes our proof.

■

3.2 Constructing Optimal Solutions from Real Eigenvectors

Now if we find some non-negative eigenvalue $\lambda \in \mathbb{R}$ (which exists by Lemma 3.1) with eigenvector $v := \begin{pmatrix} v_1 & v_2^T & v_3 & v_4^T \end{pmatrix}^T \in \mathbb{R}^{2(n+1)}$ partitioned according to $v_1, v_3 \in \mathbb{R}$ and $v_2, v_4 \in \mathbb{R}^n$, then we can derive the following:

Case $(v_1 \neq 0)$**:** By rescaling v, we can assume that $v_1 = 1$ holds true. We then use

$$0 = M(\lambda)v = \begin{pmatrix} -\lambda v_3 - g^T v_4 \\ \sigma v_2 - (H + \lambda I)v_4 \\ -\lambda + \sigma v_3 \\ -g - (H + \lambda I)v_2 \end{pmatrix} \tag{8}$$

to show

$$\lambda^2 = \lambda\sigma v_3 = \sigma(-g^T)v_4 = \sigma v_2^T (H + \lambda I)v_4 = \sigma^2 v_2^T v_2 = \sigma^2 ||v_2||_2^2 \tag{9}$$

which implies $\lambda = \sigma||v_2||_2$ (since $\lambda \geq 0$) and therefore

$$(H + \sigma||v_2||_2 I)v_2 = (H + \lambda I)v_2 = -g \tag{10}$$

i.e. (2) is satisfied for v_2.

Case $v_1 = 0$**:** This is the Hard Case. We obtain from $\sigma > 0$ and the third row of

$$0 = M(\lambda)v = \begin{pmatrix} -\lambda v_3 - g^T v_4 \\ \sigma v_2 - (H + \lambda I)v_4 \\ \sigma v_3 \\ -(H + \lambda I)v_2 \end{pmatrix} \tag{11}$$

that $v_3 = 0$. The first row then implies

$$g^T v_4 = 0 \tag{12}$$

The second and fourth row show that $\sigma||v_2||_2^2 = \sigma v_2^T v_2 = v_4^T (H + \lambda I)v_2 = 0$, i.e. $v_2 = 0$. The second row then implies

$$(H + \lambda I)v_4 = 0 \tag{13}$$

and since $v \neq 0$ is an eigenvector of M, we obtain $v_4 \neq 0$. Therefore 0 is an eigenvalue of $H + \lambda I$ with eigenvector $v_4 \in \mathbb{R}^n$. In order to construct an optimal solution of (1) we use the following corollary.

Corollary 3.2. *Let $s_* \in \mathbb{R}^n$ be a globally optimal solution of* (1) *and let λ_* be the largest real eigenvalue of* (4)*. Then*

$$\lambda_* = \sigma||s_*||_2 \tag{14}$$

holds true.

Proof. By the previous Lemma we know that $\sigma||s_*||_2 \geq 0$ is some (non-negative) eigenvalue of M and therefore we have $\sigma||s_*||_2 \leq \lambda_*$. We may now choose a real eigenvector $v :=$

$\begin{pmatrix} v_1 & v_2^T & v_3 & v_4^T \end{pmatrix}^T \in \mathbb{R}^{2(n+1)}$ of M corresponding to λ_* partitioned according to $v_1, v_3 \in \mathbb{R}$ and $v_2, v_4 \in \mathbb{R}^n$. If we assume that $\sigma||s_*||_2 < \lambda_*$ then, in combination with (3), we obtain

$$H + \lambda_* I \succ H + \sigma||s_*||_2 I \succeq 0 \tag{15}$$

which means that we must be in the case $v_1 \neq 0$ (Otherwise (13) would be a contradiction). Therefore we may assume $v_1 = 1$, and according to (9) and (10) $v_2 \in \mathbb{R}^n$ then satisfies $(H + \sigma||v_2||_2 I)v_2 = -g$ and $\sigma||v_2||_2 = \lambda_*$. Using the latter equality and (15), we obtain $H + \sigma||v_2||_2 I \succ 0$, i.e. $v_2 \in \mathbb{R}^n$ is the unique solution of (1) and therefore $v_2 = s_*$ which leads to a contradiction via $\sigma||s_*||_2 < \lambda_* = \sigma||v_2||_2 = \sigma||s_*||_2$. Therefore $\sigma||s_*||_2 = \lambda_*$ holds true.

■

Remark 3.3. *If $\lambda_* \geq 0$ is the largest real eigenvalue of M and $v_1 \neq 0$ then it follows $H + \lambda_* I \succeq 0$ and from* (9) *and* (10) *that $s_* := \frac{v_2}{v_1}$ is a globally optimal solution of* (1)*. The Hard Case $v_1 = 0$ requires additional computations as shown below.*

3.3 Dealing with the Hard Case

Let $\lambda_* \geq 0$ denote the largest real eigenvalue of M. We now assume that $v_1 = 0$ holds in the solution of Step 2 in Algorithm A2. By Corollary 3.2, it follows that $\lambda_* = \sigma||s_*||_2$ for some $s_* \in \mathbb{R}^n$ satisfying (2) and (3). We conclude that $-g \in range(H + \lambda_* I)$. By taking the least square, minimum norm solution $d := -(H + \lambda_* I)^+ g$, we make sure that $||d||_2 \leq ||s_*||_2 = \frac{\lambda_*}{\sigma}$ and $(H + \lambda_* I)d = -g$ hold true.

If $H + \lambda_* I$ is regular, then we are not actually in the hard case and $s_* = d = \frac{v_2}{v_1}$ is the unique solution to (1).

If $H + \lambda_* I$ is singular, we take an eigenvector $0 \neq u \in \mathbb{R}^n$ such that $(H + \lambda_* I)u = 0$ and define $s(t) := d + tu$. Due to Equation (13), $u = v_4$ is a viable choice. By choosing $t_* = \frac{\sqrt{(\frac{\lambda_*}{\sigma})^2 - ||d||_2^2}}{||u||_2}$ we obtain

$$||s(t_*)||_2^2 = ||d + t_* u||_2^2 = ||d||_2^2 + t_* \underbrace{d^T u}_{=0} + t_*^2 ||u||_2^2 = ||d||_2^2 + ((\tfrac{\lambda_*}{\sigma})^2 - ||d||_2^2) = (\tfrac{\lambda_*}{\sigma})^2 = ||s_*||_2^2 \tag{16}$$

and therefore $s(t_*)$ satisfies

$$(H + \sigma||s(t_*)||_2)s(t_*) = (H + \lambda_* I)(d + t_* u) = \underbrace{(H + \lambda_* I)d}_{=-g} + t_* \underbrace{(H + \lambda_* I)u}_{=0} = -g \tag{17}$$

as well as $H + \sigma||s(t_*)||_2 I = H + \sigma||s_*||_2 I \succeq 0$ showing that $s(t_*)$ satisfies both (2) as well as (3) and therefore is a globally optimal solution of (1). This justifies Step 4 in Algorithm A2.

4 Implementation Details

The generalized eigenvalue problem in step 2 of Algorithm A2 can be transformed to a standard eigenvalue problem

$$\begin{pmatrix} 0 & 0 & \sigma & 0 \\ -g & -H & 0 & 0 \\ 0 & 0 & 0 & -g^T \\ 0 & \sigma I & 0 & -H \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda_* \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \tag{18}$$

by multiplying with the matrix on the right hand side (which is self-inverse). Depending on the availability of eigensolvers, solving (18) may be preferable. The MATLAB 2017b routine `eigs` for example performs an equivalent transformation internally when computing generalized eigenvalues of M. Therefore working with (18) directly was usually 5% to 20% faster in our testing. Computing the largest real eigenvalue can then be implemented in a "matrix-free" fashion, through MATLABs internal `eigs` routine via

```
[v,lambda]=eigs(@(x) ApplyMatrix(H,g,sigma,n,x),2*(n+1),1,'largestreal');
```

where the function `ApplyMatrix` implements the multiplication with the matrix on the left-handside of (18). Showing that this will lead in fact to the largest real eigenvalue (and not only the one with the largest real part) is discussed in the next subsection. Let us, however, briefly sketch one scenario in which above transformation is less favorable: Note that one may adapt for the slightly more general setting of

$$\underset{s\in\mathbb{R}^n}{\text{minimize}} \;\; g^T s + \tfrac{1}{2} s^T H s + \tfrac{\sigma}{3}(s^T Q s)^{\frac{3}{2}} \tag{19}$$

where $Q = Q^T \in \mathbb{R}^{n\times n}$ is positive definite. (This is actually the original setting of Griewank in [9].) Changing the variables from s to $Q^{-\frac{1}{2}}s$ then applying our previous work and then changing the variables back leads to the generalized

$$\begin{pmatrix} 0 & 0 & 0 & -g^T \\ 0 & \sigma Q & 0 & -H \\ 0 & 0 & \sigma & 0 \\ -g & -H & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & Q \\ 1 & 0 & 0 & 0 \\ 0 & Q & 0 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} \tag{20}$$

eigenvalue problem. Although in theory one could explicitly multiply with the inverse of the matrix on the right-hand side, this is likely to destroy sparsity on the left-handside. Here we would therefore recommend either tackling the generalized eigenvalue problem (20) directly or hiding the transformation in a "matrix-free" fashion (via implicit back-solves) in order to solve (19).

4.1 Rightmost Eigenvalue is Real

From Corollary 3.2 it follows that we are only interested in the largest real eigenvalue of M. Here we show that this eigenvalue is also the one with the largest real part (i.e. the rightmost), which significantly simplifies any implementation details. Let $\lambda \in \mathbb{C}$ be an eigenvalue of M with eigenvector $v = \begin{pmatrix} v_1 & v_2^T & v_3 & v_4^T \end{pmatrix}^T \in \mathbb{C}^{2(n+1)}$. If $v_1 = 0$ then $\lambda \in \mathbb{R}$ follows, because of Equation (11) respectively (13): λ is also an eigenvalue of the symmetric matrix H which has only real eigenvalues. In this case the eigenvector can be choosen real $v \in \mathbb{R}^{2(n+1)}$. For $\lambda \in \mathbb{C}$ with imaginary part $Im(\lambda) \neq 0$ we can therefore assume $v_1 = 1$ and $det(H + \lambda I) \neq 0$. From (8), respectively(9) and (10), we obtain

$$\lambda^2 = \sigma\lambda v_3 = \sigma(-g)^T v_4 = \sigma^2(-g)^T(H + \lambda I)^{-1}v_2 = \sigma^2 g^T(H + \lambda I)^{-1}(H + \lambda I)^{-1}g \tag{21}$$

which can be rewritten in terms of the spectral decomposition $H = UDU^T$ (diagonal $D \in \mathbb{R}^{n\times n}$ and orthogonal $U \in \mathbb{R}^{n\times n}$). With $\tilde{g} := U^T g \in \mathbb{R}^n$ Equation (21) is equivalent to

$$0 = \lambda^2 - g^T U(D + \lambda)^{-2}U^T g = \lambda^2 - \sum_{i=1}^{n} \frac{\tilde{g}_i^2}{(D_{ii} + \lambda)^2}. \tag{22}$$

Now let $\lambda_* \geq 0$ be the largest real eigenvalue of M. For $\lambda = a + bi,\ a, b \in \mathbb{R}$ with $\lambda_* \leq a$ and $b > 0$ we have $-D_{ii} \leq \lambda_*$ and the imaginary part of above equation reads as

$$0 = Im((a + bi)^2 - \sum_{i=1}^{n} \frac{\tilde{g}_i^2(D_{ii} + (a - bi))^2}{|D_{ii} + \lambda|^4}) = 2ab + 2\underbrace{\sum_{i=1}^{n} \frac{\tilde{g}_i^2(D_{ii} + a)b}{|D_{ii} + \lambda|^4}}_{\geq 0}. \tag{23}$$

For $a > 0$, we have $ab > 0$ and therefore a contradiction. $a < 0$ can not happen (because $\lambda_* \geq 0$ holds true). Finally for $a = 0$ we obtain $\lambda_* = 0$ and therefore $s = 0$ which implies $g = 0$ and $\tilde{g} = 0$ and therefore $\lambda = 0$, i.e. a contradiction to $b > 0$. In summary, it follows that λ can not be positioned right of λ_* in the complex plane. Note that λ_* may still have higher algebraic multiplicity, which may hurt the performance and accuracy of some eigenvalue solvers. For this we refer to the next section.

4.2 Adding Extra Accuracy

In some cases an eigenvalue solver may fail to deliver the desired accuracy in step 2 of Algorithm A2. Here we point out two simple ways, that may help to regain this lost accuracy. In the hard case one may try to increase accuracy by finding a better eigenvalue/eigenvector approximation: Recall from (13), that in the hard case $v_4 \in \mathbb{R}^n$ is an (approximate) eigenvector of H to the eigenvalue $\max(-\lambda_{min}(H), 0) = \lambda_* = \sigma||s_*||_2$. Therefore we can use an iterative eigensolver starting at $\frac{v_4}{||v_4||_2}$ to improve the approximation accuracy. In MATLAB this is easily done by two lines of code:

```
[v4,lambda_min]=eigs(H,1,'smallestreal','Startvector',v4/norm(v4));
lambda=max(-lambda_min,0);
```

(24)

In the generic case, better accuracy can be reached by performing an extra Newton step: If $s \in \mathbb{R}^n$ is an approximate minimizer of (1) then

$$\bar{s} = s - (H + \sigma(||s||_2 I + \tfrac{ss^T}{||s||_2}))^{-1}(Hs + \sigma||s||_2 s + g) \tag{25}$$

can be approximated by a conjugate gradient method and leads in our experiments typically to much higher accuracy, if needed.

4.3 Stabilized Algorithm

For convenience we now state a stabilized version of Algorithm A2 following the description in subsection 2.1 and employing the transformed problem (18). Our implementation also uses the steps for adding extra accuracy from the previous section. For most applications however, this high accuracy may be unnecessary. This is why these steps are marked as optional.

Algorithm A3 Stabilized Version of Algorithm A2.

---

1. Input: $g \in \mathbb{R}^n$, $H = H^T \in \mathbb{R}^{n\times n}$, $\sigma > 0$ and (small) tolerance $\delta > 0$

2. Approximate: The rightmost eigenvalue $\lambda_* \geq 0$ and corresponding eigenvector $v = \begin{pmatrix} v_1 & v_2^T & v_3 & v_4^T \end{pmatrix}^T$ with $v_1, v_3 \in \mathbb{R}$ and $v_2, v_4 \in \mathbb{R}^n$ such that

$$\begin{pmatrix} 0 & 0 & \sigma & 0 \\ -g & -H & 0 & 0 \\ 0 & 0 & 0 & -g^T \\ 0 & \sigma I & 0 & -H \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \lambda_* \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix}$$

via an iterative method.

3. Assume Generic Case: Set

$$s_* := -\frac{\text{sign}(g^T v_4)\lambda_*}{\sigma} \frac{v_2}{||v_2||_2}$$

with the convention $\frac{0}{0} := 0$.

4. If $|g^T v_4| \leq \delta ||g||_2 ||v_4||_2$ (Treat as Hard Case):

o) (optional) Use (24) to increase accuracy.

a) Use iterative method minresQLP from [6] to approximate $d := -(H + \lambda_* I)^+ g$.

b) Find solution $t^* \in \mathbb{R}$ to the quadratic equation $||d + tv_4||_2^2 = (\frac{\lambda}{\sigma})^2$.

c) Set $\bar{s} := d + t_* v_4$.

d) Doublecheck: If $g^T \bar{s} + \frac{1}{2}\bar{s}^T H \bar{s} + \frac{\sigma}{3}||\bar{s}||_2^3 \leq g^T s_* + \frac{1}{2} s_*^T H s_* + \frac{\sigma}{3}||s_*||_2^3$ then set $s_* := \bar{s}$.

(optional): Use an iterative method to increase accuracy via extra Newton step(s) according to (25).

5. Output: $s_* \in \mathbb{R}^n$ an approximate global solution to $\min_{s \in \mathbb{R}^n} g^T s + \frac{1}{2} s^T H s + \frac{\sigma}{3}||s||_2^3$

---

5 Numerical Experiments

We implemented Algorithm A3 in MATLAB 2017b which we will refer to as `CUBgep`. Unless otherwise stated the regularization parameter is set to one, i.e. $\sigma = 1$. The parameter δ for detecting the hard case is set to 10^{-5}. All of the following experiments were performed on a PC with an Intel quad-core i7-4770 CPU, 32 Gigabyte of DDR3 RAM running Ubuntu 18.04. To gain some intuition about running times, we compare `CUBgep` to our own implementation of the well known Moré-Sorensen approach (see [11]), which (in the generic case) finds a root $\lambda_* \in \mathbb{R}_+$ of

$$\phi(\lambda) := \frac{\sigma}{\lambda} - \frac{1}{(H+\lambda I)^{-1}g} \tag{26}$$

on the interval $I = (\max(\lambda_{\min}(-H), 0), \infty)$ via Newtons method and then sets $s_* := (H + \lambda_* I)^{-1}g$. The procedure can, in principle, be performed by computing multiple Cholesky factorizations (at least one per Newton step see section 6.1 of [3] for further details). This however comes with the disadvantage of not being able to easily handle the hard case. We therefore chose to reuse one full eigenvalue decomposition of H. The eigenvalue decomposition took about 10 to 15 times longer to compute than one Cholesky decomposition. The overall computational advantage of a Cholesky-based approach typically range at a factor of two to three, but as mentioned, does not cover properly handling the hard case.
Both our implementations guarantee (3) up to machine precision. We will therefore only report averages over the violation of (2) as well as over running time: For every dimension n we average the gradient norms $||Hs + \sigma||s||_2 s + g||_2$ (here s denotes the output of the solver), as well as solution times in seconds of 100 problems.

5.1 Generic Case Results

We randomly generated g by `randn(n,1)` and symmetric $n \times n$ matrices H with about 1% density by `sprandsym(n,density);` for dimensions n between 100 and 10000.

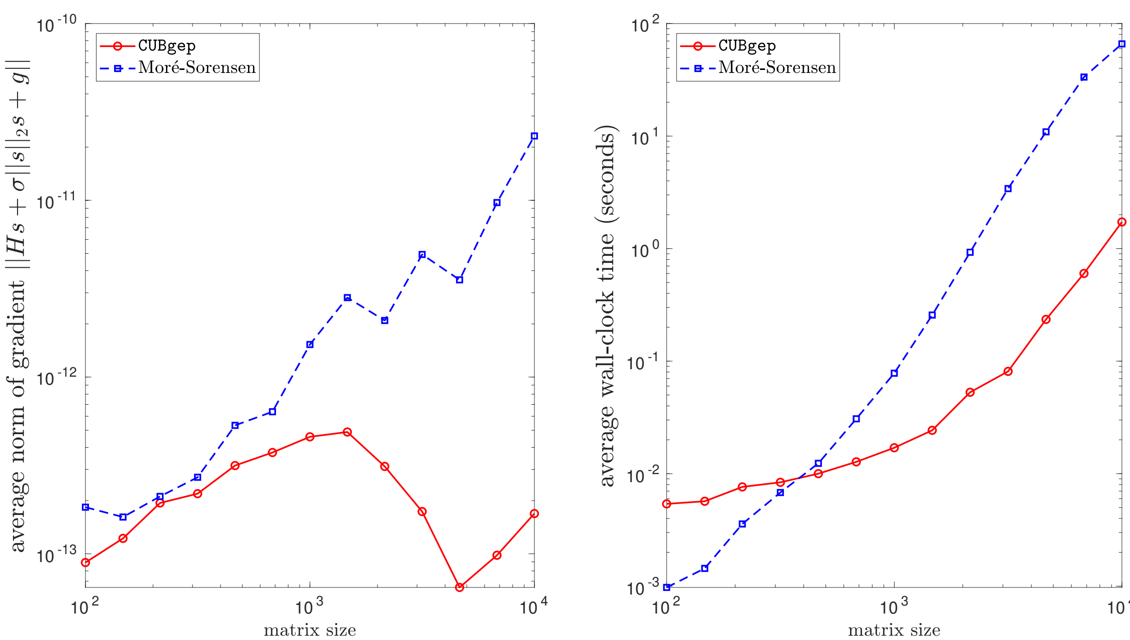

Figure 1: Results medium sparse and medium scale problems, `H=sprandsym(n,1e-2); g=randn(n,1);` $\sigma = 1$;

For these medium sized and medium sparse problems, similar results for both algorithms can be seen in Figure 1. The average violation of first order optimality for both algorithms

is below 10^{-10}, which seems rather reasonable for most applications. For `CUBgep`, the techniques for adding extra accuracy, described in section 4.2, were used if $||Hs+\sigma||s||_2 s+g||_2$ was above a threshold of 10^{-12}, explaining the graphs kink for matrix sizes above $n > 10^3$. This resulted in excellent accuracy, while leaving performance almost untouched. We note that `CUBgep` was much faster than the Moré-Sorensen approach for larger problems (up to a factor of 100). For smaller problems, the overhead of using sparse matrices did prove problematic: Since sparse-matrix-dense-vector multiplication is not multithreaded in the used MATLAB version, we essentially lost a factor of 4 here (assuming perfect scaling of our quad-core CPU). This may for example be fixed by either using full matrices in small dimensions or by using an appropriate routine from Intels Math Kernel Library 2019, such as mkl_sparse_d_mm via a mex file for large dimensions. Using such routines could potentially divide the execution time by the number of available processing cores (which we will not explore further here, due to lack of appropriate hardware). The execution time of both algorithms scale with the matrix sizes. The reasons for this depend on the algorithms and differ considerably: The execution time of the Moré-Sorensen approach is mostly dominated by the time spent on computing an eigenvalue decomposition, which grows cubic with the matrix size. On the other hand `CUBgep` spends most time on multiplications with H, which is proportional to the number of non-zero (nnz) entries in H, which in turn is roughly $nnz(H) = n^2 density(H) \approx \frac{n^2}{100}$. This proportion makes `CUBgep` very well suited for very sparse large scale problems: Consider Figure 2 for which large symmetric sparse matrices with approximately 10 entries per row/column were generated via `H=sprandsym(n, 10/n)`. Again g was generated by `randn(n,1)`. The Moré-Sorensen approach shall be omitted for this particular example as it is not feasible for the larger matrix sizes of up to 10^7.
Execution time of `CUBgep` scales almost perfectly with the number of non-zeros $nnz(H) = n^2 density(H) \approx \frac{10n^2}{n} = 10n$ on these problems: Tenfold the matrix size n results in just over ten times the execution time.

5.2 Hard Case Results

For testing the hard case, we randomly generated dense problems with known globally optimal solution according to

```
sopt=randn(n,1); V=orth(randn(n));
d=max(randn(n,1),-sigma*norm(sopt));
d(1,1)=-sigma*norm(sopt);
H=V*diag(d)*V.'; H=0.5*(H+H.');
g=-V*((d+sigma*norm(sopt)).*sopt);
sopt=V*sopt;
```

(27)

which not only makes sure that we are in the hard case, but also allowed us to compare objective values. Both routines detected the hard case correctly and yielded objective values coinciding with the optimal value up to machine precision. The least-square minimum norm solution in step 4 of Algorithm A3 was computed with `minresQLP.m` from [6]. Note that there also exists a Python implementation. For these fully dense problems, Figure 3 shows that `CUBgep` is generally more accurate and up to a factor 20 faster than the Moré-Sorensen

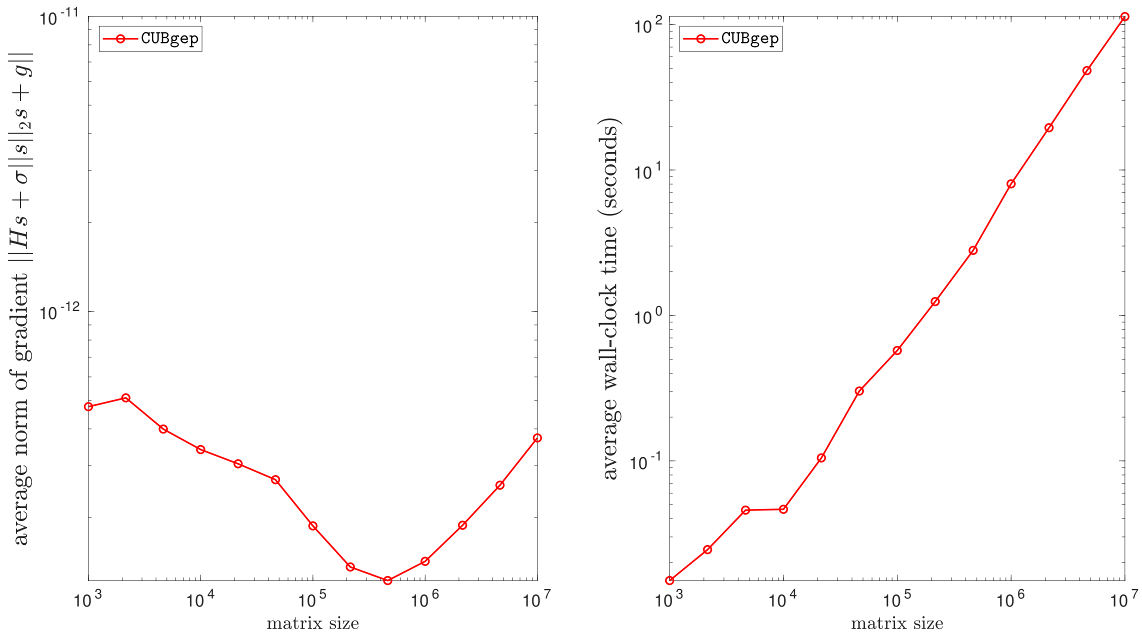

Figure 2: Results for large scale problems with approximately 10 entries per row/column: `H=sprandsym(n, 10/n); g=randn(n,1);` $\sigma = 1$;

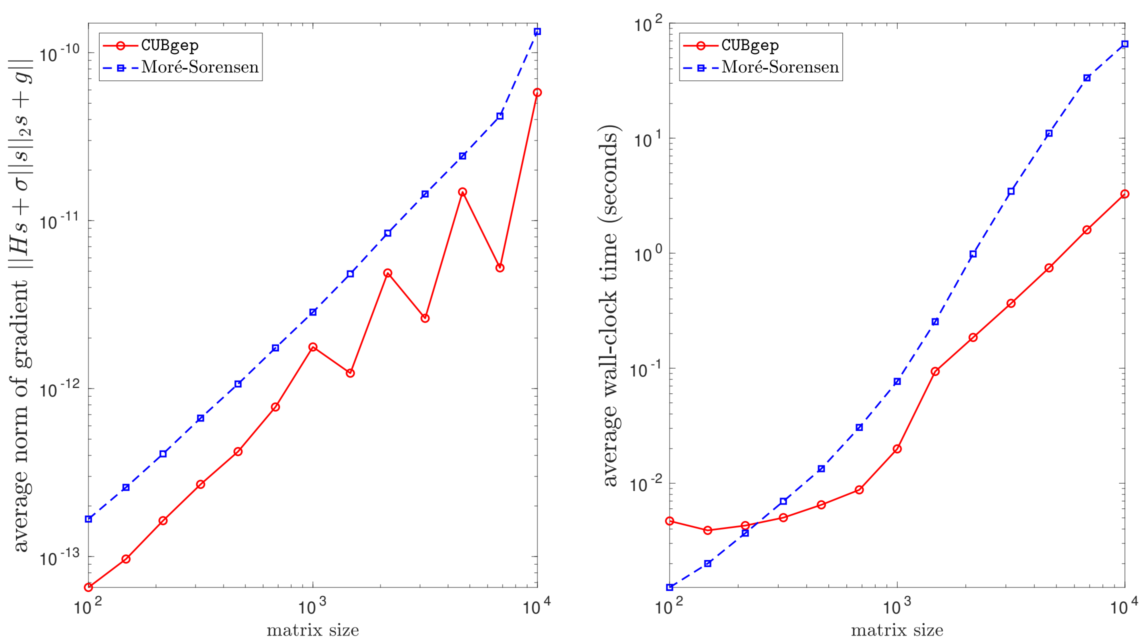

Figure 3: Results for hard case problems generated with (27)

approach. This gap is expected to grow further with increasing dimensions until the Moré-Sorensen approach becomes infeasible due to memory and computational constraints. This substantiates our belief that `CUBgep` is well suited, both for the hard case and for large scale problems.

6 Conclusion

Building up on the work presented in [1], we developed a fast, reliable and easy-to-implement way of globally solving the cubic regularization problem (1). For unconstrained optimization, the approach is of particular interest when Hessian vector products are available at reasonable computational cost, which for example may be true in machine learning, see e.g. [8], or in particle simulations, see e.g. [14]. The main part of Algorithm A2 or Algorithm A3 consists of computing the largest real (rightmost) eigenvalue and eigenvector of a real $2(n+1)$ block matrix, a problem that is well understood and solvable in huge dimensions. In fact in many instances of our numerical experiments, generating the data took significantly longer than solving the resulting problems. We hope that this shift in paradigms, helps a broad adoption of cubic regularization Newton-type methods in general.

Acknowledgment

I would like to thank Florian Jarre for careful proofreading of this paper.

References

[1] Adachi, S., Iwata, S., Nakatsukasa, Y., & Takeda, A. (2017). Solving the trust-region subproblem by a generalized eigenvalue problem. SIAM Journal on Optimization, 27(1), 269-291.

[2] Bianconcini, T., Liuzzi, G., Morini, B., & Sciandrone, M. (2015). On the use of iterative methods in cubic regularization for unconstrained optimization. Computational Optimization and Applications, 60(1), 35-57.

[3] Cartis, C., Gould, N. I., & Toint, P. L. (2011). Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. Mathematical Programming, 127(2), 245-295.

[4] Cartis, C., Gould, N. I., & Toint, P. L. (2011). Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function-and derivative-evaluation complexity. Mathematical programming, 130(2), 295-319.

[5] Choi, S. C. T., Paige, C. C., & Saunders, M. A. (2011). MINRES-QLP: A Krylov subspace method for indefinite or singular symmetric systems. SIAM Journal on Scientific Computing, 33(4), 1810-1836.

[6] `https://web.stanford.edu/group/SOL/software/minresqlp/`

[7] Doikov, N., & Richtarik, P. (2018, July). Randomized Block Cubic Newton Method. In International Conference on Machine Learning (pp. 1289-1297).

[8] Gower, R. M., Kovalev, D., Lieder, F., & Richtárik, P. (2019). RSN: Randomized Subspace Newton. arXiv preprint arXiv:1905.10874.

[9] Griewank, A. (1981). The modification of Newtons method for unconstrained optimization by bounding cubic terms. Technical report NA/12.

[10] Lehoucq, R. B., Sorensen, D. C., & Yang, C. (1998). ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods (Vol. 6). Siam.

[11] Moré, J. J., & Sorensen, D. C. (1983). Computing a trust region step. SIAM Journal on Scientific and Statistical Computing, 4(3), 553-572.

[12] Nesterov, Y., & Polyak, B. T. (2006). Cubic regularization of Newton method and its global performance. Mathematical Programming, 108(1), 177-205.

[13] Nesterov, Y. (2008). Accelerating the cubic regularization of Newtons method on convex problems. Mathematical Programming, 112(1), 159-181.

[14] Schlegel, H. B. (2011). Geometry optimization. Wiley Interdisciplinary Reviews: Computational Molecular Science, 1(5), 790-809.

[15] Tripuraneni, N., Stern, M., Jin, C., Regier, J., & Jordan, M. I. (2018). Stochastic cubic regularization for fast nonconvex optimization. In Advances in neural information processing systems (pp. 2899-2908).