CSIAM Trans. Appl. Math.
doi: 10.4208/csiam-am.2020-0008

Vol. **1**, No. 1, pp. 142-154
March 2020

An Implicit Evaluation Method of Vector 2-Norms Arising from Sphere Constrained Quadratic Optimizations

T. Sogabe[1,*], A. Suzuki[2] and S.-L. Zhang[1]

[1] *Department of Applied Physics, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan.*
[2] *Department of Computational Science and Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan.*

Received 13 February 2020; Accepted 20 March 2020

Abstract. An implicit evaluation method of vector 2-norms is presented for function evaluations arising from sphere constrained quadratic optimizations. The efficiency of the method in terms of computational costs mainly comes from the well-known shifted conjugate gradient method, and the robustness of the method comes from the fact that it never suffers from cancellations when the coefficient matrix is symmetric positive definite. Numerical experiments indicates that the method is promising for reducing computational costs of Ye's hybrid method for solving sphere constrained quadratic optimizations.

AMS subject classifications: 65F10, 65K05, 90C20

Key words: Shifted linear systems, symmetric positive definite, the shifted conjugate gradient method, sphere constrained quadratic optimizations, Ye's hybrid method.

1 Introduction

Given a symmetric positive definite matrix $A \in \mathbb{R}^{N\times N}$ and a vector $b \in \mathbb{R}^N$, finding a root of the function

$$f(\sigma) = \|(A+\sigma I)^{-1}b\|_2 - 1 \tag{1.1}$$

over the set of all positive real numbers $\mathbb{R}^+$ is of prime importance in solving sphere constrained quadratic optimizations (SQO hereafter)

$$\begin{aligned}&\text{minimize} \quad q(x) = \frac{1}{2}(x,Ax) - (b,x)\\ &\text{subject to} \quad x \in S = \{x \in \mathbb{R}^N : \|x\|_2 \le 1\}.\end{aligned}$$

*Corresponding author. *Email addresses:* `sogabe@na.nuap.nagoya-u.ac.jp` (T. Sogabe), `suzuki@na.cse.nagoya-u.ac.jp` (A. Suzuki), `zhang@na.nuap.nagoya-u.ac.jp` (S.-L. Zhang)

Here, the symbol $(\cdot,\cdot)$ denotes the standard dot product, i.e., $(x,y)=x^{\mathrm{T}}y$. The SQO problems have a rich variety of applications such as trust region algorithms for nonlinear programming (see [1] and references therein), regularization methods for ill-posed problems [8], and graph partitioning problems [6].

Of many root-finding algorithms, e.g. the bisection method, the false position method, and Newton's method, one of the most well-known root-finding algorithms for (1.1) is Ye's hybrid method [10] that is combining Newton's method and a special binary search. The most time consuming part of Ye's hybrid method is evaluating the function (1.1) for some given values, i.e.,

$$f(\sigma_\ell)=\|(A+\sigma_\ell I)^{-1}b\|_2-1, \quad \ell=1,2,\cdots,m. \tag{1.2}$$

For computing the 2-norms in (1.2), one may choose sparse direct solvers (see, e.g., [3]), since the sparse pattern of $A+\sigma_\ell I$ does not change for all $\sigma_\ell\in\mathbb{R}^+$ and thus one symbolic factorization may be enough. This is very cost-efficient in this respect. The approach, however, requires m times numerical factorizations that will be highly expensive. The Householder triangularization of $A+\sigma_\ell I$ may be more useful when the matrices are small, since the triangularizations can be done simultaneously, i.e., $Q^{\mathrm{T}}(A+\sigma_\ell I)Q=T+\sigma_\ell I$ for $\ell=1,\cdots,m$, where T is a tridiagonal matrix, and the Cholesky factorizations of tridiagonal matrices $T+\sigma_\ell I$ for $\ell=1,\cdots,m$ can be done in $\mathcal{O}(mN)$. When matrices are large, the memory requirement of $\mathcal{O}(N^2)$ may be a bottleneck.

The 2-norms in (1.2) can be viewed as the solution 2-norms $\|x^{(\ell)}\|_2$ of the following, so-called, shifted linear systems:

$$(A+\sigma_\ell I)x^{(\ell)}=b, \quad \ell=1,2,\cdots,m. \tag{1.3}$$

Krylov subspace methods for solving shifted linear systems have received much attention since only one Krylov subspace $K_n(A,b):=\mathrm{span}\{b,Ab,\cdots,A^{n-1}b\}$ is required because of the shift-invariance property $K_n(A,b)=K_n(A+\sigma_\ell I)$ (see, an excellent survey, [9]). Since the coefficient matrices are symmetric positive definite, it is natural to use the shifted Conjugate Gradient (shifted CG) method for solving (1.3) whose algorithm in a complete form is described in [7]; the symmetric version of the shifted Bi-CG [5], and an accurate variant of the shifted CG method was proposed for the case $(A^{\mathrm{T}}A+\sigma_\ell I)x^{(\ell)}=A^{\mathrm{T}}b$ [4]. This motivates us to use the shifted CG method for efficiently evaluating the function (1.1).

The purpose of the paper is to give an efficient evaluation method for computing the solution 2-norms $\|x^{(\ell)}\|_2$ of (1.3) based on the shifted CG method. The paper is organized as follows: In the next section, a brief explanation of the shifted CG method is given. In Section 3, explicit and implicit evaluation methods for evaluating function values (1.2) are described. The main contribution of the paper is the implicit evaluation method. In Section 4, the explicit/implicit evaluation methods are modified so that they can be applicable to Ye's hybrid method. In Section 5, the results of some numerical experiments are described. Finally some concluding remarks are made in Section 6.

2 The shifted CG method

The shifted CG method is a powerful solver for shifted linear systems with symmetric positive definite matrices (1.3). Let $x_0^{(\ell)}$ be initial guesses for the shifted linear systems (1.3) so that the corresponding initial residual vectors $r_0^{(\ell)} := b-(A+\sigma_\ell I)x_0^{(\ell)}$ are collinear, i.e., $\{r_0^{(\ell)}\}_{\ell=1}^m$ lie in one dimensional subspace:

$$c_1 r_0^{(1)} = c_2 r_0^{(2)} = \cdots = c_m r_0^{(m)}, \quad c_1,c_2,\cdots,c_m \in \mathbb{R}.$$

Then, the shifted COCG method finds approximate solutions over the following affine space:

$$x_n^{(\ell)} \in x_0^{(\ell)} + K_n(A+\sigma_\ell I, r_0^{(\ell)}), \tag{2.1}$$

where the nth residual vector $r_n^{(\ell)} := b-(A+\sigma_\ell I)x_n^{(\ell)}$ satisfies

$$r_n^{(\ell)} \in K_{n+1}(A+\sigma_\ell I, r_0^{(\ell)}) \perp K_n(A+\sigma_\ell I, r_0^{(\ell)}). \tag{2.2}$$

Here $K_n(A,b)$ is an n-dimensional Krylov subspace given by $\text{span}\{b,Ab,\cdots,A^{n-1}b\}$. Since $r_0^{(\ell)}$'s are collinear, it is well known that shift-invariance property of Krylov subspaces holds: $K_n(A+\sigma_i I, r_0^{(i)}) = K_n(A+\sigma_j I, r_0^{(j)})$ for all $i,j=1,2,\cdots,m$. This means that it is enough to generate only one Krylov subspace, leading to cost efficient algorithm shown in Algorithm 1.

There are some remarks on Algorithm 1: For simplicity, all initial guesses are zeros so that all initial residual vectors are collinear; in exact precision arithmetic all residual vectors $r_n^{(\ell)}$ for all ℓ,n are the same as true CG residual vectors; the seed system is chosen as $Ax=b$, since from the following proposition shown by Frommer, all shifted linear systems are solved within the required number of iteration steps for solving the seed system by the CG method.

Proposition 2.1 ([5]). *Let A be symmetric positive definite, then $1/\pi_n^{(\ell)} \in (0,1)$. From the relation $r_n^{(\ell)} = r_n/\pi_n^{(\ell)}$, it follows that $\|r_n^{(\ell)}\|_2 < \|r_n\|_2$.*

3 Explicit/Implicit evaluation methods

This section describes explicit and implicit evaluation methods for (1.1).

3.1 Explicit evaluation method

Explicit evaluation is explicitly evaluating the solution 2-norms after convergence. The algorithm is summarized in Algorithm 2.

The symbol $n(\ell)$ in Algorithm 2 denotes the required number of iteration steps for each shifted linear system (1.3).

Algorithm 1 The shifted Conjugate Gradient method

```
$x_0^{(\ell)} = p_{-1}^{(\ell)} = \mathbf{0}$, $r_0 = b$,
$\beta_{-1} = 0$, $\pi_0^{(\ell)} = \pi_{-1}^{(\ell)} = \alpha_{-1} = 1$,
for $n = 0,1,\cdots$, until $\|r_n\|_2 \le \epsilon \|b\|_2$ do:
  $p_n = r_n + \beta_{n-1} p_{n-1}$,
  $\alpha_n = \frac{(r_n,r_n)}{(p_n,Ap_n)}$,
  (begin shifted system)
    for $\ell = 1,2,\cdots,m$
      if $\|r_n^{(\ell)}\|_2 > \epsilon \|b\|_2$ then
      $\pi_{n+1}^{(\ell)} = (1+\alpha_n\sigma_\ell)\pi_n^{(\ell)} + \frac{\beta_{n-1}}{\alpha_{n-1}}\alpha_n\big(\pi_n^{(\ell)} - \pi_{n-1}^{(\ell)}\big)$,
      $\beta_{n-1}^{(\ell)} = \left(\frac{\pi_{n-1}^{(\ell)}}{\pi_n^{(\ell)}}\right)^2 \beta_{n-1}$,
      $\alpha_n^{(\ell)} = \frac{\pi_n^{(\ell)}}{\pi_{n+1}^{(\ell)}}\alpha_n$,
      $p_n^{(\ell)} = \frac{1}{\pi_n^{(\ell)}} r_n + \beta_{n-1}^{(\ell)} p_{n-1}^{(\ell)}$,
      $x_{n+1}^{(\ell)} = x_n^{(\ell)} + \alpha_n^{(\ell)} p_n^{(\ell)}$,
      end if
    end for
  (end shifted system)
  $r_{n+1} = r_n - \alpha_n A p_n$,
  $\beta_n = \frac{(r_{n+1},r_{n+1})}{(r_n,r_n)}$.
end for
```

Algorithm 2 Explicit evaluation method for $\|x_{n(\ell)}^{(\ell)}\|_2$

```
Step 1: Run Algorithm 1.
Step 2: Compute $\|x_{n(\ell)}^{(\ell)}\|_2 = (x_{n(\ell)}^{(\ell)}, x_{n(\ell)}^{(\ell)})^{1/2}$ for $\ell = 1,2,\cdots,m$.
```

3.2 Implicit evaluation method

This subsection describes implicit evaluation method for $\|x_{n(\ell)}^{(\ell)}\|_2$. For simplicity the symbol $n(\ell)$ is abbreviated as n. The key proposition for the implicit evaluation method is given below.

Proposition 3.1. *Let $D_n^{(\ell)}$ be an $n \times n$ diagonal matrix with $\|r_0^{(\ell)}\|_2/\pi_0^{(\ell)},\cdots,\|r_{n-1}^{(\ell)}\|_2/\pi_{n-1}^{(\ell)}$ on the diagonal; $B_n^{(\ell)}$ be an $n \times n$ upper diagonal matrix with $1,\cdots,1$ on the diagonal and*

$-\beta_0^{(\ell)},\cdots,-\beta_{n-2}^{(\ell)}$ on the upper diagonal; $a_n^{(\ell)}$ be a vector of length n with $\alpha_0^{(\ell)},\cdots,\alpha_{n-1}^{(\ell)}$. Then solution 2-norms of (1.3) *are evaluated by the following method:*

$$\|x_n^{(\ell)}\|_2 = \|D_n^{(\ell)}(B_n^{(\ell)})^{-1}a_n^{(\ell)}\|_2, \tag{3.1}$$

where $r_k^{(\ell)}$, $\alpha_k^{(\ell)}$, $\beta_k^{(\ell)}$ for all k,ℓ are generated in Algorithm 1.

Proof. It follows from recurrences relation for $p_n^{(\ell)}$ in Algorithm 1 that we have

$$[r_0/\pi_0^{(\ell)},\cdots,r_{n-1}/\pi_{n-1}^{(\ell)}] = [p_0^{(\ell)},\cdots,p_{n-1}^{(\ell)}]B_n^{(\ell)}. \tag{3.2}$$

Since $x_n^{(\ell)} = [p_0^{(\ell)},\cdots,p_{n-1}^{(\ell)}]a^{(\ell)}$, it follows that

$$\begin{aligned} x_n^{(\ell)} &= [p_0^{(\ell)},\cdots,p_{n-1}^{(\ell)}]a_n^{(\ell)} \\ &= [r_0/\pi_0^{(\ell)},\cdots,r_{n-1}/\pi_{n-1}^{(\ell)}](B_n^{(\ell)})^{-1}a_n^{(\ell)} \quad \text{(from (3.2))} \\ &= [r_0/\|r_0\|_2,\cdots,r_{n-1}/\|r_{n-1}\|_2]D^{(\ell)}(B_n^{(\ell)})^{-1}a_n^{(\ell)}. \end{aligned}$$

The $r_k^{(\ell)}$'s are CG residual vectors to be orthogonal to each other, and thus column vectors of the matrix $Q_n := [r_0/\|r_0\|_2,\cdots,r_{n-1}/\|r_{n-1}\|_2]$ are orthonormal. The solution 2-norms are, therefore, given by

$$\|x_n^{(\ell)}\|_2 = \|Q_n D_n^{(\ell)}(B_n^{(\ell)})^{-1}a_n^{(\ell)}\|_2 = \|D_n^{(\ell)}(B_n^{(\ell)})^{-1}a_n^{(\ell)}\|_2,$$

which concludes the proof. □

Eq. (3.1) leads to the following implicit evaluation method (Algorithm 3) for all solution 2-norms of the shifted linear systems.

Algorithm 3 Implicit evaluation method for $\|x_{n(\ell)}^{(\ell)}\|_2$

```
Step 1: Run Algorithm 1 without computing $p_n^{(\ell)}$ and $x_n^{(\ell)}$.
Step 2: Compute $b_{n(\ell)}^{(\ell)} := (B_{n(\ell)}^{(\ell)})^{-1}a_{n(\ell)}^{(\ell)}$ by back substitution.
Step 3: Compute $\|D_{n(\ell)}^{(\ell)} b_{n(\ell)}^{(\ell)}\|_2$.
```

We compare Algorithms 2 and 3 in terms of computational costs. The shifted part and computing 2-norms in Algorithm 2 require $\mathcal{O}(N\tilde{n}m)$ operations, and that in Algorithm 3 requires $\mathcal{O}(\tilde{n}m)$ operations. Here, $\tilde{n}$ denotes average number of required iterations for each shifted linear system.

From Proposition 3.1, we readily have the following corollary that implies numerical robustness of the implicit evaluation method.

Corollary 3.1. *Steps* 2 *and* 3 *in Algorithm* 3 *never suffer from cancellation.*

Proof. It is enough to show $\alpha_k^{(\ell)}, \beta_k^{(\ell)} \in \mathbb{R}^+$. Firstly, $\alpha_k^{(\ell)} \in \mathbb{R}^+$, since $\alpha_k^{(\ell)} = \|r_k^{(\ell)}\|_2^2/(p_k^{(\ell)}, (A+\sigma_\ell I)p_k^{(\ell)})$, where $(A+\sigma_\ell I)$ is symmetric positive definite. Secondly, $\beta_k^{(\ell)} = \|r_{k+1}^{(\ell)}\|_2^2/\|r_k^{(\ell)}\|_2^2 \in \mathbb{R}^+$. □

4 Application

In this section, we describe an application of Algorithms 2 and 3. These algorithms may substantially reduce computational costs of Ye's hybrid algorithm [10] that is widely known as an approximate root finding algorithm arising in the SQO problems.

4.1 Ye's hybrid algorithm

Ye's hybrid algorithm finds an approximate root of $f(\sigma)$ in (1.1) and then run root-finding algorithm such as Newton's method with the computed approximate root. The good initial guess of Ye's hybrid algorithm is given in Algorithm 4.

Algorithm 4 The initial guess of Ye's hybrid algorithm

```
Step 1: Set $\epsilon$ to be small value, set $\beta := 1+1/12$ and compute $\hat{\beta}_i := \beta^{2^i}$ for $i = 0,1,\cdots,K$,
        where $K := \lceil \log_2(\log_2(\|b\|_2/\epsilon^3)) - \log_2(\log_2\beta) \rceil$ and $\lceil\cdot\rceil$ is the ceiling function.
        Set $k := K$, $\ell := 1$, $\xi := \epsilon^3$.
Step 2: Set $\sigma_\ell := \hat{\beta}_{k-1}\xi$ and compute $g(\sigma_\ell) := \|(A+\sigma_\ell I)^{-1}b\|_2^2 - 1$.
Step 3: If $g(\sigma_\ell) > 0$ then $\xi := \hat{\beta}_{k-1}\xi$.
Step 4: Set $k := k-1$, $\ell := \ell+1$ and goto Step 2.
```

Note that the function used in Ye's hybrid algorithm is $g(\sigma) = \|(A+\sigma I)^{-1}b\|_2^2 - 1$ instead of $f(\sigma)$. The resulting value ξ in Algorithm 4 is an approximate root of $g(\sigma)$, i.e., a root of $g(\sigma)$ is in the interval $[\xi, \beta\xi]$, and thus the value ξ can be a good initial guess.

4.2 Application of the explicit/implicit evaluation methods to Ye's hybrid algorithm

We should note that (i) the explicit/implicit evaluation methods require Algorithm 1, and Algorithm 1 solves shifted linear systems in not sequential but in parallel manner, which means the algorithm is suitable for parallel evaluations of $g(\sigma_\ell)$ for $\ell = 1,2,\cdots,m$; (ii) Ye's hybrid algorithm, however, require sequential evaluations, i.e., after evaluation of $g(\sigma_k)$, it requires $g(\sigma_{k+1})$.

From the above note, Algorithms 2 and 3 requires a modification so that these algorithms can be applicable to Ye's hybrid algorithm. To this end, Algorithm 2 is modified as Algorithm 5.

Algorithm 5 A modified version of the explicit evaluation method for $\|x^{(\ell)}_{n(\ell)}\|_2$

```
Run Algorithm 1 only for $\ell=1$, and then compute $\|x^{(1)}_{n(1)}\|_2$.
Set $n_{\rm its}:=n(1)$ and store $\alpha_k,\beta_k,r_k$ for $k=0,1,\cdots,n_{\rm its}$ and $p_{n_{\rm its}}$.
for $\ell=2,3,\cdots$
  if $\|r^{(\ell)}_{n_{\rm its}}\|_2(=\|r_{n_{\rm its}}\|_2/|\pi^{(\ell)}_{n_{\rm its}}|)>\epsilon$ then
    run CG part of Algorithm 1 from $n_{\rm its}$ iteration step,
    store additional information: $\alpha_k,\beta_k,r_k$ for $k=n_{\rm its},n_{\rm its}+1,c\cdots,n(\ell)$, and $p_{n(\ell)}$,
    set $n_{\rm its}:=n(\ell)$,
  end if
  run shift part of Algorithm 1 using $\alpha_k,\beta_k,r_k$ for $k=0,1,\cdots,n(\ell)$,
  run Step 2 in Algorithm 2.
end for
```

Similarly, Algorithm 3 is modified as Algorithm 6.

Algorithm 6 A modified version of the implicit evaluation method for $\|x^{(\ell)}_{n(\ell)}\|_2$

```
Run Algorithm 1 only for $\ell=1$ without computing $p^{(1)}_n$, $x^{(1)}_n$, and then run Steps 2 and 3
in Algorithm 3.
Set $n_{\rm its}:=n(1)$ and store $\alpha_k,\beta_k$, for $k=0,1,\cdots,n_{\rm its}$ and $p_{n_{\rm its}},r_{n_{\rm its}}$.
for $\ell=2,3,\cdots$
  if $\|r^{(\ell)}_{n_{\rm its}}\|_2(=\|r_{n_{\rm its}}\|_2/|\pi^{(\ell)}_{n_{\rm its}}|)>\epsilon$ then
    run CG part of Algorithm 1 from $n_{\rm its}$ iteration step,
    store additional information: $\alpha_k,\beta_k$, for $k=n_{\rm its}+1,n_{\rm its}+2,\cdots,n(\ell)$, and $p_{n(\ell)},r_{n(\ell)}$,
    set $n_{\rm its}:=n(\ell)$,
  end if
  run Steps 2 and 3 in Algorithm 3 using $\alpha_k,\beta_k$ for $k=0,1,\cdots,n(\ell)$.
end for
```

Algorithm 6 has an additional advantage over Algorithm 5 in that the required memory in Algorithm 6 is of $\mathcal{O}(N)$, whereas $\mathcal{O}(n_{\rm its}N)$ in Algorithm 5.

5 Numerical experiments

Numerical experiments were performed on MATLAB 5.3.1. Test matrices we used are small in order to know the condition numbers. Flops are given for the evaluation of algorithms instead of CPU time.

5.1 Example 1

The first test matrix A is bcsstk09 that comes from Matrix Market

`http://math.nist.gov/MatrixMarket/`

The matrix size is 1083×1083 and the number of nonzeros is 18437. We used the shifted matrix $A - 7102.229I$ that is symmetric positive definite and the right-hand side b to be all ones. The nonzero structure of the matrix is given in Fig. 1.

Algorithm 4 with $\epsilon = 0.0001$ required 9 shifted linear systems as shown in the first column of Table 1. The corresponding condition numbers are shown in the third column.

Table 1: Solution 2-norms for shifted linear systems and these condition numbers.

Shift value	Solution 2-norm	$\kappa_2(A+\sigma_\ell I)$
σ_1: 7.927e-004	Direct: **1.59477**5024955985e+004	7.953e+010
	Explicit: **1.5947795**79679995e+004	
	Implicit: **1.5947795**75580611e+004	
σ_2: 2.232e+001	Direct: **6.073309881**220007e–001	3.028e+006
	Explicit: **6.0733098808069**45e–001	
	Implicit: **6.0733098808069**30e–001	
σ_3: 1.330e–001	Direct: **1.0186254**58467804e+002	5.079e+008
	Explicit: **1.018625439 9**46537e+002	
	Implicit: **1.0186254399**29811e+002	
σ_4: 1.723e+000	Direct: **7.8668255 8**5488928e+000	3.923e+007
	Explicit: **7.866825587 3**34573e+000	
	Implicit: **7.8668255873**24572e+000	
σ_5: 6.201e+000	Direct: **2.1858292 8**0249282e+000	1.090e+007
	Explicit: **2.1858292832209**41e+000	
	Implicit: **2.1858292832209**29e+000	
σ_6: 1.176e+001	Direct: **1.152182531**617289e+000	5.746e+006
	Explicit: **1.15218253190911**3e+000	
	Implicit: **1.15218253190911**4e+000	
σ_7: 1.620e+001	Direct: **8.365144**399127212e–001	4.171e+006
	Explicit: **8.3651444008965**51e–001	
	Implicit: **8.3651444008965**37e–001	
σ_8: 1.381e+001	Direct: **9.817420371**340140e–001	4.896e+006
	Explicit: **9.81742037170573**8e–001	
	Implicit: **9.81742037170573**7e–001	
σ_9: 1.274e+001	Direct: **1.063553505**983321e+000	5.304e+006
	Explicit: **1.06355350523408**7e+000	
	Implicit: **1.06355350523408**1e+000	

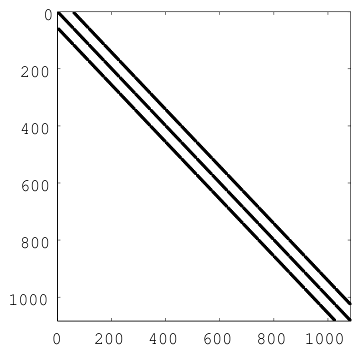

Figure 1: Sparse matrix structure for bcsstk09.

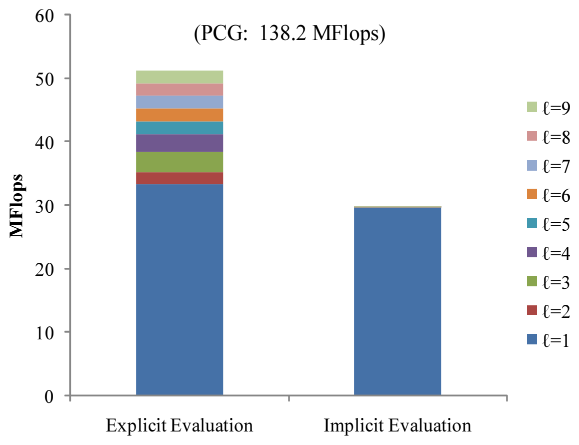

Figure 2: Operation counts of explicit/implicit evaluations for bcsstk09.

The result of the comparison in terms of accuracy among direct, explicit, and explicit evaluations is given in the second column, where direct evaluation means that all shifted linear systems were solved by the LU decomposition and then the solution 2-norms were computed. We see from Table 1 that the number of the same digits of solution 2-norms by direct, explicit, and implicit evaluations tends to be $15-\log_{10}\kappa_2(A+\sigma_\ell I)$. This implies that in this case explicit and implicit evaluation methods work well in terms of accuracy.

Next, the result of computational costs is shown in Fig. 2. "PCG" in Fig. 2 means that the conjugate gradient method with incomplete Cholesky factorization preconditioner using dropping tolerance 0.001, and the required number of operations was 138.2 MFlops for solving all shifted linear systems. The explicit evaluation method required 51.3 MFlops that was much smaller than that of the PCG method. The implicit evaluation method required 29.9 MFlops that was smaller than that of the explicit one. The main reason is that the implicit evaluation method does not require vector updates in its shifted parts.

5.2 Example 2

The second test matrix A is s3rmt3m3 that also comes from Matrix Market. The matrix size is 5357×5357 and the number of nonzeros is 207123. We used the original matrix A that is symmetric positive definite and the right-hand side b to be all ones. The nonzero structure of the matrix is given in Fig. 3.

As shown in the first column of Table 2, Algorithm 4 with $\epsilon=0.003$ required 9 shifted linear systems. The corresponding condition numbers are listed in the third column.

Table 2: Solution 2-norms for shifted linear systems and these condition numbers.

Shift value	Solution 2-norm	$\kappa_2(A+\sigma_\ell I)$
σ_1: 2.140e+001	Direct: **3.16565551061775**7e+000	4.495e+002
	Explicit: **3.16565551061775**5e+000	
	Implicit: **3.16565551061775**1e+000	
σ_2: 6.026e+005	Direct: **1.21460932094981**7e–004	1.016e+000
	Explicit: **1.21460932094981**5e–004	
	Implicit: **1.21460932094981**4e–004	
σ_3: 3.591e+003	Direct: **2.03123267410639**4e–002	3.673e+000
	Explicit: **2.03123267410639**7e–002	
	Implicit: **2.03123267410639**6e–002	
σ_4: 2.772e+002	Direct: **2.59032361055767**7e–001	3.562e+001
	Explicit: **2.59032361055767**4e–001	
	Implicit: **2.5903236105576**69e–001	
σ_5: 7.703e+001	Direct: **9.1415792396475**25e–001	1.256e+002
	Explicit: **9.1415792396475**35e–001	
	Implicit: **9.1415792396475**65e–001	
σ_6: 4.060e+001	Direct: **1.7071266170875**68e+000	2.374e+002
	Explicit: **1.70712661708757**5e+000	
	Implicit: **1.70712661708757**1e+000	
σ_7: 5.593e+001	Direct: **1.250078635467547**e+000	1.726e+002
	Explicit: **1.250078635467547**e+000	
	Implicit: **1.25007863546754**4e+000	
σ_8: 6.563e+001	Direct: **1.06916484648190**7e+000	1.472e+002
	Explicit: **1.069164846481908**e+000	
	Implicit: **1.069164846481908**e+000	
σ_9:7.110e+001	Direct: **9.8866334616169**62e–001	1.360e+002
	Explicit: **9.8866334616169**18e–001	
	Implicit: **9.8866334616169**63e–001	

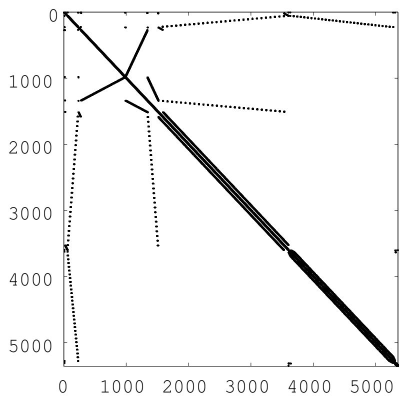

Figure 3: Sparse matrix structure for s3rmt3m3.

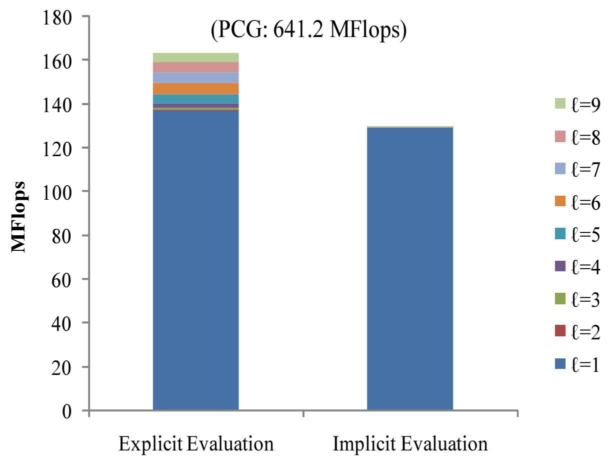

Figure 4: Operation counts of explicit/implicit evaluations for s3rmt3m3.

The result of the comparison in terms of accuracy among direct, explicit, and explicit evaluations is given in the second column. From Table 2, the number of the same digits of solution 2-norms by direct, explicit, and implicit evaluations tends to be $15-\log_{10}\kappa_2(A+\sigma_\ell I)$, which is a similar tendency to that seen in the previous example.

The result of computational costs is shown in Fig. 4. "PCG" in Fig. 4 means that the conjugate gradient method with incomplete Cholesky factorization preconditioner using dropping tolerance 0.001, and the required number of operations was 641.2 MFlops for solving all shifted linear systems. The explicit evaluation method required 163.3 MFlops. The implicit evaluation method required 129.8 MFlops that was smaller than that of the explicit one. We also see from the result that accelerating the CG part in Algorithm 1 will lead to further efficient computation for implicit/explicit evaluation methods.

6 Concluding remarks

In this paper, the implicit evaluation method of vector 2-norms was proposed for function evaluations arising from sphere constrained quadratic optimizations. The method is based on the shifted conjugate gradient method, and the total computational costs are less than that of the explicit one because of no requirement of the vector updates in the shifted part. In terms of accuracy, we have shown that the method never suffers from cancellation when the coefficient matrix is symmetric positive definite. Furthermore, we have modified the explicit/implicit evaluation methods so that they can be applicable to Ye's hybrid method for solving sphere constrained quadratic optimizations. In this case, we have seen that the implicit method is more memory efficient than the explicit one.

From numerical experiments we have learned that the implicit method is promising for reducing computational costs of Ye's hybrid method, and that accelerating the CG part in the shifted CG method will be a key for more efficient computation of the implicit evaluation method. To achieve this, a deflation technique may be a method of choice (see, e.g., [2]).

Finally, future work is described below, which was inspired by useful and insightful comments of the anonymous referee. Since one of the commonly used methods in practice for the SQO problem is the truncated CG method [11], both theoretical and practical comparison arising from nonlinear programming problem and machine learning (logistic regression) between the truncated CG method and our algorithms will be important future work. Furthermore, an extension of the presented algorithms to the case where A is positive indefinite will also be interesting future work. In this case, we may need to consider some theory such as look-ahead Lanczos process to avoid breakdown.

Acknowledgments

The authors appreciate the anonymous referee for the useful and insightful comments that enhanced the quality of the manuscript.

This work was partially supported by JSPS KAKENHI (Grant Numbers 17H02829, 18H05392, 19K12002). The authors sincerely thank Prof. Yinyu Ye at Stanford University for his valuable comments on sphere constrained quadratic optimizations.

References

[1] CONN, A. R., GOULD, N. & TOINT, PH. L. (2000) *Trust-Region Methods*. Philadelphia: SIAM.
[2] DARNELL, D., MORGAN, R. B., & WILCOX, W. (2008) Deflated GMRES for systems with multiple shifts and multiple right-hand sides *Lin. Alg. Appl.* **429**, 2415–2434.
[3] DAVIS, T. A. (2006) *Direct Methods for Sparse Linear System*. Philadelphia: SIAM.
[4] ESHOF, J. & SLEIJPEN, G. L. G. (2004) Accurate conjugate gradient methods for families of shifted linear systems. *Appl. Numer. Math.*, **49**, 17–37.

[5] FROMMER, A. (2003) BiCGStab(ℓ) for families of shifted linear systems. *Computing*, **70**, 87–109.

[6] HAGER, W. W. & KRYLYUK, Y. (1999) Graph partitioning and continuous quadratic programming. *SIAM J. Discrete. Math.*, **12**, 500–523.

[7] JEGERLEHNER, B. (1996) Krylov space solvers for shifted linear systems. Hep-lat/9612014.

[8] MENKE, W. (1989) *Geophysical Data Analysis: Discrete Inverse Theory*. San Diego: Academic Press.

[9] SIMONCINI, V. & SZYLD, D. B. (2007) Recent computational developments in Krylov Subspace Methods for linear systems. *Numer. Lin. Alg. Appl.*, **14**, 1–59.

[10] YE, Y. (1992) A new complexity result on minimization of a quadratic function with a sphere constraint. *in C. Floudas and P. Pardalos eds., Recent Advances in Global Optimization*. New Jersey: Princeton University Press.

[11] YUAN, Y. (2000) On the truncated conjugate gradient method. *Math. Program.*, Ser. A **87**, 561–573.