Accepted: 31 May 2020

Compact quasi-Newton preconditioners for symmetric positive definite linear systems

Luca Bergamaschi[1] | **José Marín**[2] | **Ángeles Martínez**[3]

[1]Department of Civil Environmental and Architectural Engineering, University of Padova, Padova, Italy

[2]Instituto de Matemática Multidisciplinar, Departamento de Matemática Aplicada, Universitat Politècnica de València, València, Spain

[3]Department of Mathematics and Earth Sciences, University of Trieste, Trieste, Italy

Correspondence
Ángeles Martínez, Department of Mathematics and Earth Sciences, University of Trieste, Via Valerio 12/1, 34127 Trieste, Italy.
Email: amartinez@units.it

Summary
In this paper, preconditioners for the conjugate gradient method are studied to solve the Newton system with symmetric positive definite Jacobian. In particular, we define a sequence of preconditioners built by means of Symmetric Rank one (SR1) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) low-rank updates. We develop conditions under which the SR1 update maintains the preconditioner symmetric positive definite. Spectral analysis of the SR1 preconditioned Jacobians shows an improved eigenvalue distribution as the Newton iteration proceeds. A compact matrix formulation of the preconditioner update is developed which reduces the cost of its application and is more suitable to parallel implementation. Some notes on the implementation of the corresponding Inexact Newton method are given and some numerical results on a number of model problems illustrate the efficiency of the proposed preconditioners.

KEYWORDS
conjugate gradient, inexact Newton method, quasi-Newton matrices, sequence of preconditioners

1 | INTRODUCTION

The purpose of this work is to develop efficient preconditioners for the linear systems arising in the inexact Newton method. The Newton method computes the solution of a system of nonlinear equations $\boldsymbol{F}(\boldsymbol{x})=0$, with $\boldsymbol{F}:\mathbb{R}^n\to\mathbb{R}^n$ iteratively by generating a sequence of approximations. Assuming that each component of $\boldsymbol{F}, f_i$, is differentiable, the Jacobian matrix $J(\boldsymbol{x})$ is defined by

$$J(\boldsymbol{x})_{ij}=\frac{\partial f_i}{\partial x_j}(\boldsymbol{x}),$$

and the Newton step is written as

$$\begin{cases} J(\boldsymbol{x}_k)\boldsymbol{s}_k=-\boldsymbol{F}(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\boldsymbol{s}_k \end{cases}. \tag{1}$$

Thus, the method involves the solution of a linear system with the Jacobian matrix. In this work it will be assumed that the Jacobian is a symmetric positive definite (SPD) matrix, as it happens in a number of applications such as, for instance, unconstrained optimization of convex problems, discretization of nonlinear Partial Differential Equations (PDEs) by, for example, the Picard method,[1] or eigensolution of SPD matrices using the (simplified) Jacobi–Davidson method.[2-4] When J is large and sparse, iterative methods are recommended to solve the linear system with the Jacobian matrix in (1).

The issue of devising preconditioners for the sequences of linear systems arising in the application of an (Inexact) Newton method to solve nonlinear problems has been addressed, for example, in References 5,6. To construct a sequence of preconditioners $\{P_k\}$ for the preconditioned conjugate gradient (PCG) solution of the Newton systems we use Quasi-Newton approximations of the Jacobian matrix as in Reference 7. In particular we develop and compare two approaches based on the symmetric Quasi-Newton formulas: SR1 and Broyden-Fletcher-Goldfarb-Shanno (BFGS). These low-rank corrections have been used as preconditioners in previous papers. We mention, among the others,[8] where the BFGS preconditioner is named *balancing preconditioner*,[9] where the authors use a single-vector version of the SR1 update, and the work in[10] in which both BFGS and SR1 updates are reviewed and used in the framework of iterative eigensolvers. We also mention the work[11] where a short survey on the existing literature on low-rank preconditioner updates is made. In Reference 12 a bounded deterioration property has been proved for the BFGS preconditioners which implies that $\|I - P_k J(\boldsymbol{x}_k)\|$ can be made as small as desired depending both on the closeness of $\boldsymbol{x}_0$ to the exact solution and of the initial preconditioner P_0 to the inverse of the initial Jacobian. BFGS-based sequences of preconditioners have been successfully used in the acceleration of inner iterative solvers in the framework of eigensolution of large and sparse SPD matrices.[3]

Differently from the BFGS sequence the SR1 update formula is not expected to provide SPD preconditioners even if the initial one is so. In this work, we will give some conditions under which it is possible to prove symmetric positive definiteness of the SR1 sequence. Moreover, since in the SR1 case the bounded deterioration property does not generally hold, we will prove that the spectral distribution of the preconditioned matrix at step $k+1$: $P_{k+1}J(\boldsymbol{x}_{k+1})$ is not worst than that of the preconditioned matrix at step k.

Application of this kind of preconditioners requires, in addition to the application of the initial approximation of $J(\boldsymbol{x}_0)^{-1}$, a number of scalar products that may considerably slow down the PCG iteration, even if limited memory variants are considered. In view of a parallel implementation, when a high number of processors is employed, large number of scalar products usually reveals a bottleneck of the overall efficiency. To address this problem, we have developed a compact version of the two classes of preconditioners, following the work in Reference 13 where analogous formula are developed in a more general framework, but not in connection with the idea of preconditioning. These matrix versions of the preconditioner updates allow for the use of BLAS-2 kernels for their applications and, at the same time, may reduce considerably the number of communications among processors.

The remainder of the paper is as follows: in Section 2 we review the theoretical properties of the BFGS sequences of preconditioners and we develop a matrix version of the preconditioner update. Section 3 analyzes the condition under which the SR1 sequence is SPD, shows the theoretical properties of the preconditioned matrices and develops a matrix version of the preconditioner update. In Section 4 we present some numerical results which give evidence of the computational gain provided by the compact formulas and of the good behavior of the SR1 update to accelerate the PCG method. In Section 5 we draw some conclusions.

Notation: Throughout the paper we will use the Euclidean norm for vectors and the Frobenius norm ($\|.\|_F$) for matrices.

2 | BFGS RECURRENCE AS A PRECONDITIONER

BFGS-type formulas are used in the context of Quasi-Newton methods to approximate the true Jacobian as the iteration progresses. At a given Newton step, let B_k be an approximation of the Jacobian matrix J_k, while P_k denotes its inverse, that is, $P_k = B_k^{-1}$. By denoting $\boldsymbol{F}_k = \boldsymbol{F}(\boldsymbol{x}_k)$, let us define the vectors $\boldsymbol{y}_k = \boldsymbol{F}_{k+1} - \boldsymbol{F}_k$, $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$. The BFGS formula is derived by imposing the following conditions on the inverse Jacobian matrix:

- P_{k+1} must be symmetric and positive definite.
- It must satisfy the secant condition, $P_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k$.
- Closeness condition: among all the symmetric matrices satisfying the secant equation, P_{k+1} must be the closest matrix to P_k. Using the Frobenius norm:

$$\begin{aligned} P_{k+1} &= \operatorname{argmin}_P \|P - P_k\|_F \\ \text{subject to } & P = P^T, \quad P\boldsymbol{y}_k = \boldsymbol{s}_k. \end{aligned} \tag{2}$$

The unique solution to (2) is (see Reference 14)

$$P_{k+1} = V_k^T P_k V_k + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^T, \tag{3}$$

where

$$V_k = (I - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^T), \quad \rho_k = 1/\boldsymbol{y}_k^T \boldsymbol{s}_k. \tag{4}$$

By applying the Sherman–Morrison–Woodbury formula to (3) an update formula for the Jacobian matrix approximation B_k is obtained as

$$B_{k+1} = B_k - \frac{B_k \boldsymbol{s}_k \boldsymbol{s}_k^T B_k}{\boldsymbol{s}_k^T B_k \boldsymbol{s}_k} + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}. \tag{5}$$

Moreover, in order to compute the approximate Jacobian efficiently, especially when the number of nonlinear iterations is large, limited-memory Quasi-Newton methods were introduced. L-BFGS methods store only a few pairs of vectors $\{\boldsymbol{s}_i, \boldsymbol{y}_i\}$ instead of all vectors generated during the iteration process, that is, early iteration pairs are discarded. Therefore, the computational and memory storage requirements at each Newton iteration are reduced. This may be helpful for solving large-scale nonlinear problems. More precisely, from Equations (3) and (4) we can see that the matrix P_{k+1} is obtained by updating the P_k with the pair $\{\boldsymbol{s}_k, \boldsymbol{y}_k\}$. After the new iterate is computed, the oldest vector pair in the set is deleted and replaced by the new pair. Thus, the set of vector pairs includes information from the $k_{\max}$ most recent iterations. Moreover if $(k \ mod \ k_{\max}) = 0$, a new initial preconditioner is computed as $P_0 \equiv P_k$. Starting with an initial matrix P_0, Formula (3) can be written as

$$\begin{aligned} P_k = {} & (V_{k-1}^T \cdots V_{k-k_{\max}}^T) P_0 (V_{k-k_{\max}} \cdots V_{k-1}) \\ & + \rho_{k-k_{\max}} (V_{k-1}^T \cdots V_{k-k_{\max}+1}^T) \boldsymbol{s}_{k-k_{\max}} \boldsymbol{s}_{k-k_{\max}}^T (V_{k-k_{\max}+1} \cdots V_{k-1}) \\ & + \rho_{k-k_{\max}+1} (V_{k-1}^T \cdots V_{k-k_{\max}+2}^T) \boldsymbol{s}_{k-k_{\max}+1} \boldsymbol{s}_{k-k_{\max}+1}^T (V_{k-k_{\max}+2} \cdots V_{k-1}) \\ & \vdots \\ & + \rho_{k-1} \boldsymbol{s}_{k-1} \boldsymbol{s}_{k-1}^T. \end{aligned} \tag{6}$$

From this expression, the computation of the product $\hat{\boldsymbol{r}} = P_k \boldsymbol{r}$ for a given residual vector $\boldsymbol{r}$ can be done recursively[14] by performing a sequence of inner products and vector summations involving $\boldsymbol{F}_k$ and the pairs $\{\boldsymbol{s}_i, \boldsymbol{y}_i\}$, $i = k - k_{\max}, \ldots, k - 1$, see Algorithm 1. The computational cost is (at most) $2k_{\max}$ daxpys and $2k_{\max}$ dot products.

Algorithm 1. L-BFGS two-loop recursion to compute $\hat{\boldsymbol{r}} = P_k \boldsymbol{r}$

```
for $i = k - 1, \ldots, k - k_{\max}$ do
    $\alpha_i = \rho_i \boldsymbol{s}_i^T \boldsymbol{r}$
    $\boldsymbol{r} := \boldsymbol{r} - \alpha_i \boldsymbol{y}_i$
end for
$\hat{\boldsymbol{r}} = P_0 \boldsymbol{r}$
for $i = k - k_{\max}, \ldots, k - 1$ do
    $\beta = \rho_i \boldsymbol{y}_i^T \hat{\boldsymbol{r}}$
    $\hat{\boldsymbol{r}} := \hat{\boldsymbol{r}} + (\alpha_i - \beta) \boldsymbol{s}_i$
end for
```

The implementation of the inexact Newton method[15] requires the definition of a stopping criterion for the linear solver (1) based on the nonlinear residual. Thus, the linear iteration is stopped with the test

$$\|J(\boldsymbol{x}_k)\boldsymbol{s}_k + F(\boldsymbol{x}_k)\| \le \eta_k \|F(\boldsymbol{x}_k)\|. \tag{7}$$

The sequence $\{\eta_k\}$ will be chosen such that $\eta_k = O(\|\boldsymbol{F}(\boldsymbol{x}_k)\|)$. This will guarantee quadratic convergence of the inexact Newton method and as a consequence, the following result.

Proposition 1. *Define* $\boldsymbol{e}_k = \boldsymbol{x}_* - \boldsymbol{x}_k$. *There exist* $\delta > 0$ *and* $0 < r < 1$ *such that if* $\|\boldsymbol{e}_0\| < \delta$ *then* $\|\boldsymbol{e}_{k+1}\| \le r \, \|\boldsymbol{e}_k\|$ *for every k.*

If the Jacobian matrices are SPD and so is P_0, then P_k obtained with the BFGS update is also SPD under the condition $\boldsymbol{s}_k^T \boldsymbol{y}_k > 0$ (see lemma 4.1.1 in Reference 16). Let $\boldsymbol{e}_0 = \boldsymbol{x}_* - \boldsymbol{x}_0$, $E_k = J_k^{-1} - P_k$. The next result establishes that $\|I - P_k J(\boldsymbol{x}_k)\|_F$ can be made arbitrarily small by suitable choices of the initial guess $\boldsymbol{x}_0$ and the initial preconditioner P_0.

Theorem 1 (theorem 3.6 in Reference 12). *For a fixed* $\varepsilon_1 > 0$, *there are* δ_0, δ_B *such that if* $\|\boldsymbol{e}_0\| < \delta_0$ *and* $\|E_0\|_F < \delta_B$ *then*

$$\|I - P_k J_k\|_F < \varepsilon_1.$$

2.1 | Compact inverse representation of the BFGS update formula

Following References 14,17 we write a compact formula for the direct update

$$B_k = B_0 - \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix} \begin{bmatrix} S_k^T B_0 S_k & L_k \\ L_k^T & -D_k \end{bmatrix}^{-1} \begin{bmatrix} S_k^T B_0 \\ Y_k^T \end{bmatrix}, \tag{8}$$

where

$$S_k = \begin{bmatrix} \boldsymbol{s}_0, & \dots, & \boldsymbol{s}_{k-1} \end{bmatrix}, \quad Y_k = \begin{bmatrix} \boldsymbol{y}_0, & \dots, & \boldsymbol{y}_{k-1} \end{bmatrix},$$

and

$$(L_k)_{(i,j)} = \begin{cases} \boldsymbol{s}_{i-1}^T \boldsymbol{y}_{j-1} & \text{if } i > j, \\ 0 & \text{otherwise} \end{cases}, \quad D_k = \operatorname{diag}\begin{bmatrix} \boldsymbol{s}_0^T \boldsymbol{y}_0, & \dots, & \boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}.$$

By applying the Sherman–Woodbury–Morrison formula to (8) one obtains an explicit formula for the inverse of B_k, that is, P_k. We have

$$P_k = P_0 - \begin{bmatrix} S_k & Z_k \end{bmatrix} \begin{bmatrix} R_k^{-T} H_k R_k^{-1} & -R_k^{-T} \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^T \\ Z_k^T \end{bmatrix},$$

where S_k, Y_k are as before and

$$Z_k = P_0 Y_k, \quad (R_k)_{i,j} = \begin{cases} \boldsymbol{s}_{i-1}^T \boldsymbol{y}_{j-1} & \text{if } i \le j, \\ 0 & \text{otherwise} \end{cases}, \quad H_k = D_k + Y_k^T P_0 Y_k. \tag{9}$$

2.2 | Computation of $\hat{\boldsymbol{r}} = P_k \boldsymbol{r}$

At step k, previously to the PCG solution of the kth Newton system, we have to compute $\boldsymbol{z}_{k-1} = P_0 \boldsymbol{y}_{k-1}$ by solving the linear system $B_0 \boldsymbol{z}_{k-1} = \boldsymbol{y}_{k-1}$. Then we have to form recursively

$$S_k = \begin{bmatrix} S_{k-1} \ \boldsymbol{s}_{k-1} \end{bmatrix}, \quad Z_k = \begin{bmatrix} Z_{k-1} \ \boldsymbol{z}_{k-1} \end{bmatrix},$$

$$R_k = \begin{bmatrix} R_{k-1} & S_{k-1}^T \boldsymbol{y}_{k-1} \\ 0 & \boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}, \quad H_k = \begin{bmatrix} H_{k-1} & Z_{k-1}^T \boldsymbol{y}_{k-1} \\ \boldsymbol{y}_{k-1}^T Z_{k-1} & (s_{k-1} + \boldsymbol{z}_{k-1})^T \boldsymbol{y}_{k-1} \end{bmatrix}.$$

During the PCG solver, application of P_k to the residual vector takes the steps recalled in Algorithm 2.

Algorithm 2. Computation of $\hat{\boldsymbol{r}} = P_k \boldsymbol{r}$ with compact (L-)BFGS

```
1. Solve $B_0 \hat{\boldsymbol{r}} = \boldsymbol{r}$;
2. $\boldsymbol{w}_1 = S_k^T \boldsymbol{r}$;
3. $\boldsymbol{w}_2 = Z_k^T \boldsymbol{r}$;
4. Solve $R_k \boldsymbol{q}_2 = \boldsymbol{w}_1$;
5. Solve $R_k^T \boldsymbol{q}_1 = \boldsymbol{w}_2 - H_k \boldsymbol{q}_2$;
5. $\hat{\boldsymbol{r}} = \hat{\boldsymbol{r}} - S_k \boldsymbol{q}_1 - Z_k \boldsymbol{q}_2$.
```

2.3 | Limited memory implementation L-BFGS

If $k > k_{\max}$ then the update of matrices S_k,Z_k,U_k, and H_k changes as follows (in MATLAB notation). First, rewrite the matrices by shifting elements corresponding to the last $k_{\max} - 1$ updates:

$$\begin{aligned} S_{k-1} &\leftarrow S_{k-1}(:, 2 : k_{\max}) \\ Z_{k-1} &\leftarrow Z_{k-1}(:, 2 : k_{\max}) \\ R_{k-1} &\leftarrow R_{k-1}(2 : k_{\max}, 2 : k_{\max}) \\ H_{k-1} &\leftarrow H_{k-1}(2 : k_{\max}, 2 : k_{\max}). \end{aligned}$$

Then, update the matrices with the new entries,

$$S_k = \begin{bmatrix} S_{k-1} \ \boldsymbol{s}_{k-1} \end{bmatrix}, \quad Z_k = \begin{bmatrix} Z_{k-1} \ \boldsymbol{z}_{k-1} \end{bmatrix},$$

$$R_k = \begin{bmatrix} R_{k-1} & S_{k-1}^T \boldsymbol{y}_{k-1} \\ 0 & \boldsymbol{s}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}, \quad H_k = \begin{bmatrix} H_{k-1} & Z_{k-1}^T \boldsymbol{y}_{k-1} \\ \boldsymbol{y}_{k-1}^T Z_{k-1} & (\boldsymbol{s}_{k-1} + \boldsymbol{z}_{k-1})^T \boldsymbol{y}_{k-1} \end{bmatrix}.$$

3 | PRECONDITIONER BASED ON SR1 RECURRENCE

In the L-BFGS updating formula, the matrix B_{k+1} (or P_{k+1}) differs from its predecessor B_k (or P_k) by a rank-2 matrix. There is a simpler rank-1 update that maintains symmetry of the matrix and allows it to satisfy the secant equation. But unlike the rank-2 formulas, this symmetric-rank-1 update, also called SR1, does not guarantee that the updated matrix maintains positive definiteness. The direct SR1 update formula is given by (see Reference 14)

$$B_{k+1} = B_k + \frac{(\boldsymbol{y}_k - B_k \boldsymbol{s}_k)(\boldsymbol{y}_k - B_k \boldsymbol{s}_k)^T}{(\boldsymbol{y}_k - B_k \boldsymbol{s}_k)^T \boldsymbol{s}_k}. \tag{10}$$

By applying the Sherman–Morrison formula, one can obtain the corresponding update formula for the preconditioner (we will omit subscript $_k$ in vectors $\boldsymbol{s}$ and $\boldsymbol{y}$ from now on)

$$P_{k+1} = P_k + \frac{(\boldsymbol{s} - P_k \boldsymbol{y})(\boldsymbol{s} - P_k \boldsymbol{y})^T}{\boldsymbol{y}^T (\boldsymbol{s} - P_k \boldsymbol{y})}, \tag{11}$$

satisfying the secant condition

$$P_{k+1} \boldsymbol{y} = \boldsymbol{s}.$$

Formula (11) does not generally provide a sequence of SPD matrices and may not be well defined since the scalar $\boldsymbol{y}^T(\boldsymbol{s} - P_k \boldsymbol{y})$ can in principle be zero. In the sequel of this section we will discuss these two issues and we will prove that $P_{k+1} J(\boldsymbol{x}_{k+1})$ has a more favorable eigenvalue distribution than $P_k J(\boldsymbol{x}_k)$.

Let us denote with Ω an open subset of $\mathbb{R}^n$, we will make the following *standard assumptions* on $\boldsymbol{F}$ which we will assume to hold throughout this section.

Standard assumptions:

1. Equation $\boldsymbol{F}(\boldsymbol{x}) = 0$ has a solution $\boldsymbol{x}^*$.
2. $J(\boldsymbol{x}) : \Omega \to \mathbb{R}^{n\times n}$ is Lipschitz continuous with Lipschitz constant γ.
3. $J_* \equiv J(\boldsymbol{x}_*)$ is nonsingular (Set $\alpha = \|{J_*}^{-1}\|_F$) .

Notation. It is now convenient to indicate with the subscript c every vector or matrix referring to the current iterate and with the subscript $+$ every quantity referring to the new iterate $k+1$. Moreover we will use J_+ and J_c instead of $J(\boldsymbol{x}_{k+1})$ and $J(\boldsymbol{x}_k)$ and $\boldsymbol{F}_+$,$\boldsymbol{F}_c$ instead of $\boldsymbol{F}(\boldsymbol{x}_{k+1})$, $\boldsymbol{F}(\boldsymbol{x}_k)$.

Following this notation Newton's method can be stated as

$$\begin{aligned} J_c\boldsymbol{s} &= -\boldsymbol{F}_c \\ \boldsymbol{x}_+ &= \boldsymbol{x}_c + \boldsymbol{s}. \end{aligned} \tag{12}$$

We define the error vectors $\boldsymbol{e}_+ = \boldsymbol{x}_* - \boldsymbol{x}_+$,$\boldsymbol{e}_c = \boldsymbol{x}_* - \boldsymbol{x}_c$.

Lemma 1. *Let $\|\boldsymbol{e}_c\| \le \delta_0 < \frac{1}{\alpha\gamma}$. Then J_c is invertible and*

$$\|J_c^{-1}\|_F \le \frac{\alpha}{1-\alpha\gamma\delta_0} \equiv \alpha_c.$$

Proof. Follows from theorem 1.2.1 in Reference 18, also known as Banach lemma. ■

The following Lemma will provide relations between the vectors $\boldsymbol{s}, \boldsymbol{y}$, and $\boldsymbol{e}_c$.

Lemma 2. *There exists $\delta > 0$ such that if*

$$0 < \|\boldsymbol{x}_c - \boldsymbol{x}_*\| \le \delta,$$

then the following relations hold

$$\|\boldsymbol{s}\| \le 2\|\boldsymbol{e}_c\|. \tag{13}$$

$$\boldsymbol{y} = J_c\boldsymbol{s} + \Delta_1\boldsymbol{s}, \quad \text{with} \quad \|\Delta_1\|_F \le \gamma\|\boldsymbol{e}_c\| \tag{14}$$

$$\|\boldsymbol{y}\| \ge \frac{1-\alpha_c\gamma\delta}{\alpha_c}\|\boldsymbol{s}\| \equiv c_1\|\boldsymbol{s}\|. \tag{15}$$

Proof. From $\boldsymbol{s} = \boldsymbol{e}_c - \boldsymbol{e}_+$ we have, by Proposition 1,

$$\|\boldsymbol{s}\| \le \|\boldsymbol{e}_+\| + \|\boldsymbol{e}_c\| \le (1+r)\|\boldsymbol{e}_c\| \le 2\|\boldsymbol{e}_c\|.$$

Now from the standard assumptions, and from the fundamental theorem of calculus we have that

$$\boldsymbol{y} = \boldsymbol{F}_+ - \boldsymbol{F}_c = \int_0^1 J(\boldsymbol{x}_c + t\boldsymbol{s})\boldsymbol{s}dt = J_c\boldsymbol{s} + \int_0^1 (J(\boldsymbol{x}_c + t\boldsymbol{s}) - J_c)\boldsymbol{s}dt = J_c\boldsymbol{s} + \Delta_1\boldsymbol{s}, \tag{16}$$

where $\Delta_1 = \int_0^1 (J(\boldsymbol{x}_c + t\boldsymbol{s}) - J_c)dt$. From the standard assumptions, (13) and Proposition 1

$$\|\Delta_1\|_F \le \int_0^1 \|(J(\boldsymbol{x}_c + t\boldsymbol{s}) - J_c)\|_F dt \le \int_0^1 \gamma\|\boldsymbol{x}_c + t\boldsymbol{s} - \boldsymbol{x}_c\|dt = \gamma\int_0^1 t\|\boldsymbol{s}\|dt \le \gamma\|\boldsymbol{e}_c\|.$$

Finally from

$$\|J_c^{-1}\|\|\boldsymbol{y}\| \ge \|J_c^{-1}\boldsymbol{y}\| = \|\boldsymbol{s} + J_c^{-1}\Delta_1\boldsymbol{s}\| \ge \|\boldsymbol{s}\| - \|J_c^{-1}\Delta_1\boldsymbol{s}\| \ge (1 - \alpha_c\gamma\|\boldsymbol{e}_c\|)\|\boldsymbol{s}\|$$

if $\delta < \frac{1}{\alpha_c \gamma}$ then we have

$$\|\boldsymbol{y}\| \geq \frac{(1-\alpha_c\gamma\delta)\|\boldsymbol{s}\|}{\alpha_c}.$$

■

3.1 | Well definition and positive definiteness of P_{k+1}

We now give some evidence that the sequence $\{P_k\}$ can be made SPD provided that P_0 is so. Assume that P_c is SPD. Then P_+ is SPD if the denominator in (11) is positive. Using the results of the previous Lemma we can rewrite the denominator in (11), using $\boldsymbol{s} = J_c^{-1}\boldsymbol{y} - J_c^{-1}\Delta_1\boldsymbol{s}$ from (14), as

$$\boldsymbol{y}^T(\boldsymbol{s} - P_c\boldsymbol{y}) = \boldsymbol{y}^T\boldsymbol{s} - \boldsymbol{y}^T P_c \boldsymbol{y} = \boldsymbol{y}^T J_c^{-1}\boldsymbol{y} - \boldsymbol{y}^T P_c\boldsymbol{y} - \boldsymbol{y}^T J_c^{-1}\Delta_1\boldsymbol{s}.$$

Dividing both sides by $\|\boldsymbol{y}\|^2$ we have that

$$\frac{\boldsymbol{y}^T(\boldsymbol{s}-P_c\boldsymbol{y})}{\|\boldsymbol{y}\|^2} = \frac{\boldsymbol{y}^T J_c^{-1}\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} - \frac{\boldsymbol{y}^T P_c\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{y}} - \frac{\boldsymbol{y}^T J_c^{-1}\Delta_1\boldsymbol{s}}{\boldsymbol{y}^T\boldsymbol{y}} \geq q_{J_c^{-1}}(\boldsymbol{y}) - q_{P_c}(\boldsymbol{y}) - c_2\delta, \tag{17}$$

where $q_A(\boldsymbol{y})$ denotes the Rayleigh quotient for a given SPD matrix A and $c_2 = \frac{\gamma(1+\alpha_c\gamma\delta)}{c_1^2}$ since

$$\left|\frac{\boldsymbol{y}^T J_c^{-1}\Delta_1\boldsymbol{s}}{\boldsymbol{y}^T\boldsymbol{y}}\right| \leq \frac{(1+\alpha_c\gamma\delta)(\gamma\delta)\|\boldsymbol{s}\|^2}{c_1^2\|\boldsymbol{s}\|^2} = \frac{(1+\alpha_c\gamma\delta)\gamma}{c_1^2}\delta = c_2\delta.$$

Remark 1. The expression on the right of (17) is expected to be positive for the following reason. The initial preconditioner P_0 is usually computed as the inverse of a Cholesky factorization of $A \equiv J(\boldsymbol{x}_0)$, that is, $P_0 = (LL^T)^{-1}$. As known this kind of preconditioner is able to capture the largest eigenvalues of the coefficient matrix rather than the lowest ones.[19] As a consequence the typical spectrum of the preconditioned matrices is an interval $[\alpha, \beta]$ with α close to zero and β slightly larger than 1. The sign of the eigenvalues of $A^{-1} - (LL^T)^{-1} = L^{-T}(L^TA^{-1}L - I)L^{-1}$ is related to the sign of those of $L^TA^{-1}L - I$ which are therefore larger than $\frac{1}{\beta} - 1$. As an example we report the extremal eigenvalues of the preconditioned matrix arising from the FD discretization of the Poisson equation with size $n = 39\,204$ preconditioned with the Cholesky factorizations with either no fill-in or with drop tolerances of $\tau = 10^{-3}, 10^{-5}$.

	α	β
IC(0)	8.504×10^{-4}	1.2057
$\tau = 1e-3$	2.253×10^{-2}	1.1445
$\tau = 1e-5$	0.5097	1.0998

In practice, it is rather easy to guarantee that

$$d = \min_{\boldsymbol{y}\in\mathbb{R}^n} q_{A^{-1}}(\boldsymbol{y}) - q_{(LL^T)^{-1}}(\boldsymbol{y}) = \lambda_{\min}(A^{-1} - (LL^T)^{-1})$$

is bounded away from zero by first roughly estimating β and then simply scaling the preconditioner factor L by a scalar to satisfy $\beta < 1$. This is described in the following pseudocode:

Algorithm 3. Scaling of the initial preconditioner factor to guarantee that the SR1 update is SPD

```
1. Estimate the largest eigenvalue $\tilde{\beta}$ of $L^{-1}AL^{-T}$ with a few iterations of the Lanczos method.
2. scale_factor = $1.2\tilde{\beta}$
3. Set $L = L \cdot \sqrt{\text{scale\_factor}}$.
```

The approximate eigenvalue $\tilde{\beta}$ must be increased by a safety factor 1.2 since the Lanczos process underestimates the true eigenvalue.

3.2 | Effects of the low-rank update on the eigenvalues of P_+J_+

The following Theorem will state that the SR1 correction is able to set approximately to 1 one eigenvalue of the preconditioned matrix.

Theorem 2. *If P_c is SPD and δ is small enough so that the conclusions of Lemma 2 hold, if moreover the denominator in (11) is positive (with $\|P_+\| = \alpha_1$) and if $0 < \|\boldsymbol{e}_c\| \le \delta$ then*

$$P_+J_+\boldsymbol{s} = \boldsymbol{s} + \boldsymbol{w}, \quad \text{with} \quad \|\boldsymbol{w}\| \le 6\alpha_1\gamma\delta^2.$$

Proof. The secant condition $P_+\boldsymbol{y} = \boldsymbol{s}$ can be rewritten using (16) as

$$P_+(J_c\boldsymbol{s} + \Delta_1\boldsymbol{s}) = \boldsymbol{s},$$

which is equivalent to

$$P_+J_+\boldsymbol{s} = \boldsymbol{s} - P_+\Delta_1\boldsymbol{s} + P_+(J_+ - J_c)\boldsymbol{s} = \boldsymbol{s} + \boldsymbol{w},$$

with

$$\|\boldsymbol{w}\| \le \alpha_1\gamma\delta\|\boldsymbol{s}\| + \alpha_1\gamma\|\boldsymbol{s}\|^2 \le \text{ (using } \|\boldsymbol{s}\| \le 2\|\boldsymbol{e}_c\| \le 2\delta) \le 6\alpha_1\gamma\delta^2.$$

■

The previous theorem shows that the preconditioned matrix P_+J_+ has an approximate eigenvector $\boldsymbol{s}$ corresponding to the approximate eigenvalue 1, the goodness of the approximation depending on how $\boldsymbol{x}_c$ is close to the exact solution $\boldsymbol{x}_*$. Regarding all other eigenvalues we can observe that P_+J_+ is similar to the SPD matrix $J_+^{1/2}P_+J_+^{1/2}$ so we denote by $\hat{J}_+$ this new matrix and by $\hat{J}_c$ its analogous at step k, namely $J_c^{1/2}P_cJ_c^{1/2}$. The eigenvalue distribution of $\hat{J}_+$ is described by Theorem 3. Before proceeding we recall a linear algebra result, stated in lemma 4.2 of Reference 20 for positive operators on an Hilbert space, which we restate for finite dimensional operators.

Lemma 3. *Let A,B SPD and $\zeta = \max\{\lambda_{\min}(A), \lambda_{\min}(B)\}$. Then*

$$\|\sqrt{B} - \sqrt{A}\|_F \le \frac{1}{\sqrt{\zeta}}\|B - A\|_F.$$

Proof. Suppose $\zeta = \lambda_{\min}(B)$ and let $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ the eigenvectors of B. Then using the fact that the trace is invariant by similarity transformation:

$$\begin{aligned}
\|B - A\|_F^2 = \mathrm{Tr}((B-A)^2) &= \sum_{i=1}^n \boldsymbol{v}_i^T(B-A)^2\boldsymbol{v}_i \\
&= \sum_{i=1}^n \boldsymbol{v}_i^T(\lambda_i I - A)^2\boldsymbol{v}_i \\
&= \sum_{i=1}^n \boldsymbol{v}_i^T\left(\sqrt{\lambda_i}I - \sqrt{A}\right)^2\left(\sqrt{\lambda_i}I + \sqrt{A}\right)^2\boldsymbol{v}_i \\
&\ge \sum_{i=1}^n \left(\zeta\boldsymbol{v}_i^T\left(\sqrt{\lambda_i}I - \sqrt{A}\right)^2\boldsymbol{v}_i\right) \\
&= \zeta\sum_{i=1}^n \left(\boldsymbol{v}_i^T\left(\sqrt{B} - \sqrt{A}\right)^2\boldsymbol{v}_i\right) \\
&= \zeta\|\sqrt{B} - \sqrt{A}\|_F^2.
\end{aligned}$$

■

We now premise the following

Lemma 4. *If δ is small enough so that the conclusions of Lemma 2 hold, then*

$$\hat{J}_+ = \hat{J}_c + \boldsymbol{z}\boldsymbol{z}^T + \Delta_2, \tag{18}$$

where

$$\boldsymbol{z} = \frac{J_c^{1/2}(\boldsymbol{s} - P_c\boldsymbol{y})}{\sqrt{\boldsymbol{y}^T(\boldsymbol{s} - P_c\boldsymbol{y})}}, \quad \Delta_{J^{1/2}} = J_+^{1/2} - J_c^{1/2}. \tag{19}$$

$$\Delta_2 = \Delta_{J^{1/2}} P_+ \Delta_{J^{1/2}} + \Delta_{J^{1/2}} P_+ + P_+ \Delta_{J^{1/2}}. \tag{20}$$

$$\|\Delta_2\| \le c_3\delta, \quad \text{with} \quad c_3 = \frac{4\alpha_1\gamma}{\lambda_{\min}(J_c)}\left(\gamma\delta + \sqrt{\lambda_{\min}(J_c)}\right). \tag{21}$$

Proof. Multiplying (11) by $J_+^{1/2}$ on the left and on the right yields

$$\hat{J}_+ = J_+^{1/2} P_c J_+^{1/2} + \frac{J_+^{1/2}(\boldsymbol{s} - P_c\boldsymbol{y})(\boldsymbol{s} - P_c\boldsymbol{y})^T J_+^{1/2}}{\boldsymbol{y}^T(\boldsymbol{s} - P_c\boldsymbol{y})}.$$

Writing $J_+^{1/2} = J_c^{1/2} + (J_+^{1/2} - J_c^{1/2})$ and substituting at the right hand side yield (18) with definitions (19) and (20). To obtain (21) we apply Lemma 3 and the Lipschitz continuity of the Jacobians:

$$\|J_+^{1/2} - J_c^{1/2}\|_F \le \frac{1}{\sqrt{\lambda_{\min}(J_c)}}\|J_+ - J_c\|_F \le \frac{2\gamma\delta}{\sqrt{\lambda_{\min}(J_c)}}.$$

Hence

$$\|\Delta_2\|_F \le \|P_+\|_F(\|\Delta_{J^{1/2}}\|_F^2 + 2\|\Delta_{J^{1/2}}\|_F) \le \alpha_1\left(\frac{4\gamma^2\delta^2}{\lambda_{\min}(J_c)} + \frac{4\gamma\delta}{\sqrt{\lambda_{\min}(J_c)}}\right) = \delta\frac{4\alpha_1\gamma}{\lambda_{\min}(J_c)}(\gamma\delta + \sqrt{\lambda_{\min}(J_c)}) \equiv c_3\delta.$$

■

We are able to state the final theorem regarding eigenvalue distribution for $\hat{J}_+$, assuming all eigenvalues of $\hat{J}_+$ and $\hat{J}_c$ are ordered increasingly. Denote furthermore with $\boldsymbol{v}_1$ the eigenvector of $\hat{J}_c$ corresponding to $\lambda_1(\hat{J}_c)$

Theorem 3.

$$\lambda_1(\hat{J}_+) \ge \lambda_1(\hat{J}_c) + \sqrt{\lambda_2(\hat{J}_c) - \lambda_1(\hat{J}_c)}|\boldsymbol{v}_1^T\boldsymbol{z}| - c_3\delta \tag{22}$$

$$\lambda_k(\hat{J}_c) - c_3\delta \le \lambda_k(\hat{J}_+) \le \lambda_{k+1}(\hat{J}_c) + c_3\delta, \quad k = 2, \dots, n-1, \tag{23}$$

$$\lambda_n(\hat{J}_+) \le \lambda_n(\hat{J}_c) + \|\boldsymbol{z}\|^2 + c_3\delta. \tag{24}$$

Proof. Bound (22) comes from corollary 2.3 in Reference 21. Bounds (23) and (24) are a direct consequence of Weyl's theorem (Reference 22, theorem 8.1.8). ■

Using the previous theorem we can state a bounded deterioration property regarding the condition number of the preconditioned Jacobians:

Corollary 1. *If*

$$\delta < \frac{\sqrt{\lambda_2(\hat{J}_c) - \lambda_1(\hat{J}_c)}|\boldsymbol{v}_1^T\boldsymbol{z}|}{c_3}, \tag{25}$$

then

$$\kappa(\hat{J}_+) \le \left(1 + \frac{\|\boldsymbol{z}\|^2}{\lambda_n(\hat{J}_c)} + O(\delta)\right)\kappa(\hat{J}_c).$$

Proof. If (25) holds, then from (22) we obtain $\lambda_1(\hat{J}_+) \geq \lambda_1(\hat{J}_c)$ hence

$$\kappa(\hat{J}_+) = \frac{\lambda_n(\hat{J}_+)}{\lambda_1(\hat{J}_+)} \leq \frac{\lambda_n(\hat{J}_c) + \|\boldsymbol{z}\|^2 + c_3\delta}{\lambda_1(\hat{J}_c)} = \frac{\lambda_n(\hat{J}_c) + \|\boldsymbol{z}\|^2 + c_3\delta}{\lambda_n(\hat{J}_c)}\kappa(\hat{J}_c).$$

■

Summarizing the findings of this section, P_+J_+ has an approximate eigenvalue at one thus improving the spectral properties of P_cJ_c. Moreover, the condition number of P_+J_+ may be only slightly worse than that of P_cJ_c, depending on $\|\boldsymbol{z}\|$, which in turn can be controlled by the boundedness of the denominator of $\boldsymbol{z}$ in (19) as per Remark 1.

3.3 | Compact representation of the SR1 update

To develop the compact representation of the inverse update (11) we recall the definitions in (9):

$$Z_k = P_0 Y_k, \quad (R_k)_{i,j} = \begin{cases} \boldsymbol{s}_{i-1}^T \boldsymbol{y}_{j-1} & \text{if } i \leq j, \\ 0 & \text{otherwise} \end{cases}, \quad H_k = D_k + Y_k^T P_0 Y_k.$$

and set

$$M_k = R_k + R_k^T - H_k, \quad Q_k = S_k - Z_k.$$

Then the compact formula for the SR1 update in inverse form is[17]

$$P_k = P_0 + Q_k M_k^{-1} Q_k^T. \tag{26}$$

Matrix M_k can be computed using the following recursive characterization

$$M_k = \begin{bmatrix} M_{k-1} & \boldsymbol{w}_{k-1} \\ \boldsymbol{w}_{k-1}^T & \boldsymbol{q}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}, \tag{27}$$

where $\boldsymbol{w}_{k-1} = Q_{k-1}^T \boldsymbol{y}_{k-1}$. Therefore, the inverse update only requires the computation of the vector $\boldsymbol{q}_{k-1} = \boldsymbol{s}_{k-1} - P_0\boldsymbol{y}_{k-1}$ and the matrix update (27).

A simple strategy to prevent break down in the SR1 is just skipping the update if the denominator in (10) or (11) is small. That is, the update is applied if

$$|\boldsymbol{s}_k^T(\boldsymbol{y}_k - B_k\boldsymbol{s}_k)| \geq r\|\boldsymbol{s}_k\|\,\|\boldsymbol{y}_k - B_k\boldsymbol{s}_k\|, \tag{28}$$

where $r \in (0,1)$ is a small number, for instance $r = 10^{-4}$.

3.4 | Computation of $\hat{\boldsymbol{r}} = P_k\boldsymbol{r}$

At step k, previously to the PCG solution of the kth Newton system, we have to compute $\boldsymbol{q}_{k-1} = \boldsymbol{s}_{k-1} - \boldsymbol{z}_{k-1}$ by solving the linear system $B_0\boldsymbol{z}_{k-1} = \boldsymbol{y}_{k-1}$. Then we have to form recursively

$$Q_k = [Q_{k-1}, \boldsymbol{q}_{k-1}], \quad M_k = \begin{bmatrix} M_{k-1} & Q_{k-1}^T \boldsymbol{y}_{k-1} \\ \boldsymbol{y}_{k-1}^T Q_{k-1} & \boldsymbol{q}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}.$$

Algorithm 4 shows the application of P_k to the residual vector $\boldsymbol{r}$. The storage memory and number of operations needed are just half of the ones required with the compact BFGS formula.

Algorithm 4. Computation of $\hat{\boldsymbol{r}} = P_k \boldsymbol{r}$ with compact (L-)SR1

```
1. Solve $B_0 \hat{\boldsymbol{r}} = \boldsymbol{r}$;
2. $\boldsymbol{w}_1 = Q_k^T \boldsymbol{r}$;
3. Solve $M_k \boldsymbol{w}_2 = \boldsymbol{w}_1$;
4. $\hat{\boldsymbol{r}} = \hat{\boldsymbol{r}} + Q_k \boldsymbol{w}_2$;
```

3.5 | Limited memory implementation L-SR1

If $k > k_{\max}$ then the update of matrices Q and M changes as follows (in MATLAB notation):

$$Q_{k-1} \leftarrow Q_{k-1}(:, 2 : k_{\max})$$
$$M_{k-1} \leftarrow M_{k-1}(2 : k_{\max}, 2 : k_{\max})$$
$$Q_k = [Q_{k-1}, \boldsymbol{q}_{k-1}], \quad M_k = \begin{bmatrix} M_{k-1} & Q_{k-1}^T \boldsymbol{y}_{k-1} \\ \boldsymbol{y}_{k-1}^T Q_{k-1} & \boldsymbol{q}_{k-1}^T \boldsymbol{y}_{k-1} \end{bmatrix}.$$

4 | NUMERICAL EXPERIMENTS

In this section we present the results of numerical experiments for solving different nonlinear test problems of large size. As initial preconditioner P_0 an incomplete Cholesky factorization (IC) of the initial Jacobian was often used. Therefore, its application requires the solution of two triangular linear systems. Additionally, some results with Jacobi preconditioning, that is, $B_0 = \text{diag}(J(\boldsymbol{x}_0))$, are also presented. The numerical experiments are performed with MATLAB running in a Windows OS PC equipped with an Intel i5-6600k CPU processor and 16Gb RAM. The CPU times are measured in seconds with the functions `tic` and `toc`.

In Sections 4.1 and 4.2 the IC preconditioner was computed with the `ichol` function with drop tolerances 0.1 and 0.01. For the solution of the linear systems in (1) the MATLAB function `pcg` was used with stopping criterion the relative residual norm below 10^{-6} and allowing a maximum of 2000 iterations. The nonlinear iterations were stopped whenever $\|F(\boldsymbol{x}_k)\| \leq 10^{-10}\|F(\boldsymbol{x}_0)\|$. The initial solution is the vector $\boldsymbol{x}_0 = [0.1, \ldots, 0.1]^T$.

The results are presented either with graphics or tables. In the tables, the total number of linear iterations (totlin) and total CPU time required to solve the problem are reported. The number of nonlinear iterations (nlit) is recalled in the caption since it is independent of the selected preconditioner.

The objectives of the experiments were

- to compare standard vs compact formulas
- to study the convergence behavior with different values of $k_{\max}$ (size of the update pair set or update window)
- to compare the acceleration provided by L-BFGS and L-SR1 compact formulas.

All these aspects have been investigated considering a number of nonlinear problems.

4.1 | Discrete Bratu and modified PHI-2 problems

The discrete Bratu problem consists in solving the nonlinear problem

$$A\boldsymbol{u} = \lambda D(\boldsymbol{u}), \quad D = \text{diag}(\exp(u_1), \ldots, \exp(u_n))$$

where A is an SPD matrix arising from a two-dimensional discretization of the diffusion equation on a unitary domain, and λ is a real parameter.

The second problem considered in this work corresponds to the nonlinear problem

$$A\boldsymbol{u} = \lambda D(\boldsymbol{u}), \quad D = \mathrm{diag}(u_1^3, \dots, u_n^3).$$

This problem is a variant of the PHI-2 equation[12] to maintain the Jacobian SPD. As for the Bratu problem, A is the discretization of the Laplacian operator, $\lambda = -1$.

Figure 1 shows a comparison between the standard and compact L-BFGS formulas for one of the different discretizations of the Bratu problem tested. It shows clearly that the compact implementation can achieve better computational performance for increasing values of $k_{\max}$, the size of the update window. In our experiments, as it could be expected, it was observed that the difference in performance increases when a large number of linear iterations must be performed, situation that is likely to happen when the nonlinear iteration method is approaching the solution (since in these model problems the ill-conditioning of the Jacobians increases toward the end of the Newton process), or when a poor initial preconditioner is used. Similar results were encountered for the other problems tested, also for the L-SR1 formulas. In this case, the gain using the compact formulas is less evident since the number of vectors used to update of the preconditioners is half the number of vectors used with L-BFGS.

Figure 2 shows a comparison between the L-BFGS and L-SR1 formulas in their compact version. It is worth to note that the number of iterations spent by the L-SR1 method is similar, or even smaller in many occasions, than the one required by the L-BFGS. This nice behavior gives experimental evidence of the theoretical findings for the SR1 update presented in Section 3. As a consequence, the total computational CPU time spent by the Newton method with the L-SR1 was smaller in all our experiments compared to the L-BFGS formula.

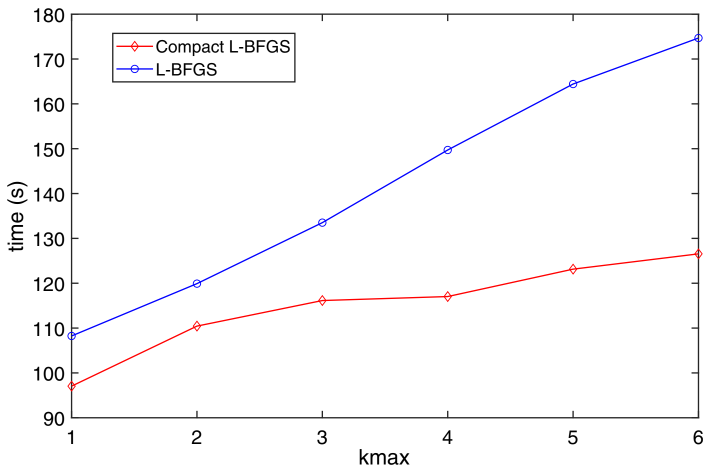

FIGURE 1 Bratu problem $n = 747\,003$, $nnz = 3\,731\,023$. Compact vs standard L-BFGS formulas. Initial preconditioner IC(0.01)

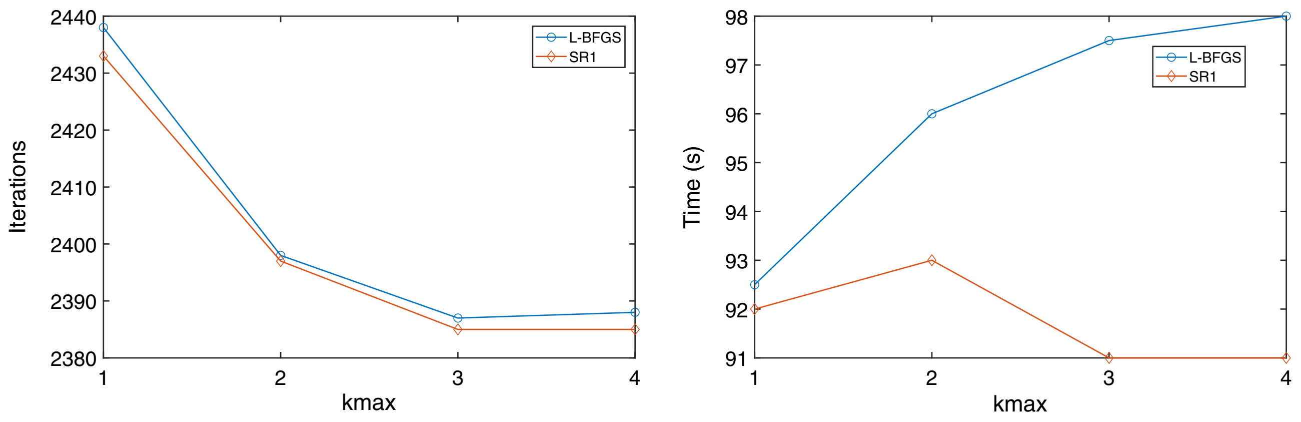

FIGURE 2 Compact L-BFGS vs L-SR1. Phi-2 problem $n = 747\,003$, $nnz = 3\,731\,023$. Initial preconditioner IC(0.01)

TABLE 1 Overall number of linear iterations and CPU time for the three-dimensional finite element (FE) problem with different initial preconditioners, update formulas and $k_{\max}$ values. Number of nonlinear iterations nlit = 15

IC(0.01)	k_{max}	totlin	CPU	IC(0.1)	k_{max}	totlin	CPU	Jacobi	k_{max}	totlin	CPU
No-update	0	3 728	69.1	No-update	0	4 248	66.6	No-update	0	9 124	133.2
L-BFGS	1	3 341	65.0	L-BFGS	1	3 703	63.5	L-BFGS	1	7 779	123.9
	2	3 194	70.1		2	3 583	67.7		2	7 501	136.7
	3	3 192	71.7		3	3 519	69.2		3	7 362	138.8
	4	3 192	73.5		4	3 513	71.1		4	7 339	139.4
L-SR1	1	3 351	63.8	L-SR1	1	3 673	63.7	L-SR1	1	7 781	123.6
	2	3 164	63.2		2	3 518	60.9		2	7 357	119.6
	3	3 152	64.8		3	3 501	62.3		3	7 192	119.1
	4	3 147	65.1		4	3 493	63.2		4	7 197	119.2

4.2 | Three-dimensional Finite Element problem

The third problem considered was a modification of the Bratu problem, where the linear term corresponding to the matrix A arises from a three-dimensional finite element discretization of Darcy's law in porous media with heterogeneous hydraulic conductivity coefficient. The size of the problem is $n = 268\,515$ with a number of nonzero elements $nnz = 3\,881\,337$. For this problem, together with IC preconditioning with drop tolerances 0.1 and 0.01, Jacobi diagonal preconditioning was used to show the effect of starting with a poor preconditioner.

Table 1 compares the L-BFGS and L-SR1 methods with initial IC and Jacobi preconditioners and also the no-update case. It can be observed, analogously to the previous problems, that updating by means of the L-SR1 formula yields the best performance with respect to both total number of linear iterations and CPU time. Although IC(0.01) yields the best results in terms of number of iterations, the best CPU time was achieved with drop tolerance 0.1, since in this case the number of nonzero elements of the preconditioner is halved. The time spent to update the preconditioner was negligible compared with the total amount, so it is not indicated. It is worth noting that for these experiments the best results were obtained with $k_{\max} = 2$ when using L-SR1. We finally highlight that in all previous experiments, the SR1 update was always well defined, with the denominator in (11) positive, and the test (28) always satisfied, in all Newton iterations.

4.3 | Application to eigenvalue computation

Given an SPD matrix A, to compute its leftmost eigenpair, the Newton method in the unit sphere[23] or Newton–Grassmann method, constructs a sequence of vectors $\{\boldsymbol{u}_k\}$ by solving the linear systems

$$J_k\boldsymbol{s} = -\boldsymbol{r}_k, \quad \text{where} \quad J_k = (I - \boldsymbol{u}_k\boldsymbol{u}_k^T)(A - \theta_k I)(I - \boldsymbol{u}_k\boldsymbol{u}_k^T),$$
$$\boldsymbol{r}_k = -(A\boldsymbol{u}_k - \theta_k\boldsymbol{u}_k), \quad \text{and} \quad \theta_k = \frac{\boldsymbol{u}_k^T A\boldsymbol{u}_k}{\boldsymbol{u}_k^T\boldsymbol{u}_k} \tag{29}$$

on a subspace orthogonal to $\boldsymbol{u}_k$. Then the next approximation is set as $\boldsymbol{u}_{k+1}=\boldsymbol{t}\|\boldsymbol{t}\|^{-1}$ where $\boldsymbol{t}=\boldsymbol{u}_k+\boldsymbol{s}$. Linear system (29) is shown to be better conditioned than the one with $A - \theta_k I$. The same linear system represents the *correction equation* in the well-known Jacobi–Davidson method,[24] which in its turn can be viewed as an accelerated inexact Newton method.[25] When A is SPD and the leftmost eigenpairs are being sought, it has been proved in Reference 2 that the PCG method can be employed in the solution of the correction equation. In this work we follow the PCG implementation of algorithm 5.1 of Reference 2.

We tried our updated preconditioners for the PCG solver within the Newton–Grassmann method to compute the leftmost eigenpair of two large and sparse SPD matrices whose characteristics are reported in Table 2. Note that the

Matrix	Where it comes from	n	nz	Drop tolerance
MONTE-CARLO	2D-MFE stochastic PDE	77120	384320	10^{-3}
EMILIA-923	3D-FE elasticity problem	923136	41 005206	10^{-5}

Abbreviations: MFE, mixed finite element; 2D, Two dimensional; 3D, three dimensional.

TABLE 2 Main characteristics of the matrices used in the tests

	$k_{\max} = 10$			$k_{\max} = 5$		
	nlit	totlin	CPU	nlit	totlin	CPU
L-BFGS	14	269	4.83	15	329	5.66
L-SR1	13	312	4.72	14	340	5.07
No update	21	585	7.50			

TABLE 3 Outer/inner iterations an CPU time to evaluate the leftmost eigenpair of matrix `monte-carlo`.

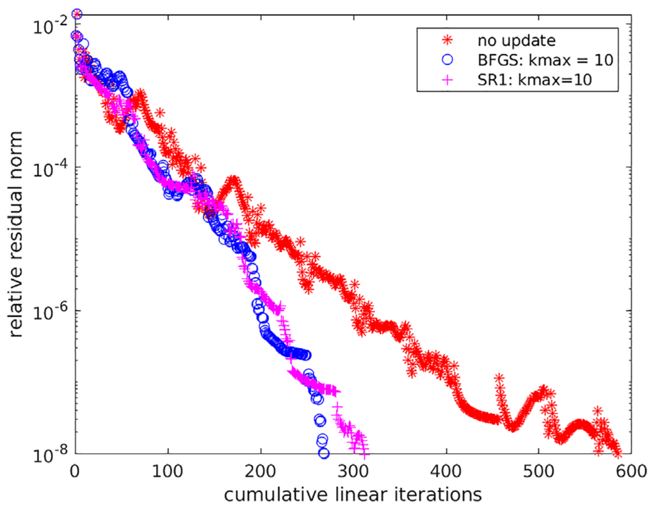

FIGURE 3 Nonlinear converge profile vs cumulative linear iteration number with different preconditioning strategies. Matrix `monte-carlo`

solution to (29) is not needed to be very accurate. We then set the maximum number of inner CG iterations equal to 50. The outer Newton process is stopped instead when the following test is satisfied

$$\|A\boldsymbol{u}_k - \theta_k \boldsymbol{u}_k\| = \|\boldsymbol{r}_k\| < \theta_k 10^{-8}.$$

The initial Newton vector has been computed satisfying $\|\boldsymbol{r}_0\| < \theta_0 10^{-2}$. In both cases the preconditioner is computed once and for all at the beginning of the Newton iteration as an incomplete Cholesky factorization with fill-in of *matrix A*.

4.3.1 | Matrix `Monte-Carlo`

In Table 3 we report the number of outer iterations (nlit) the overall number of the linear iterations (totlin) and the CPU time. We compared the no-update case and the L-BFGS and L-SR1 updates, both with $k_{\max} = \{5, 10\}$. Moreover we plot, in Figure 3 the nonlinear relative residual $\|\boldsymbol{r}_k\|\theta_k^{-1}$ vs the number of cumulative linear iterations across all the outer Newton iterations.

From Table 3 and Figure 3 we can appreciate the remarkable improvement provided by the compact low-rank updates in terms of inner/outer iterations and CPU time. Moreover, once again, the L-SR1 update provides comparable results as the L-BFGS correction being (slightly) the most convenient option in terms of CPU time.

4.3.2 | Matrix `emilia-923`

We report in Table 4 the results for the larger matrix `emilia-923` where the L-BFGS acceleration is shown to provide outstanding improvement in both inner/outer iteration number and CPU time, which is also accounted for by Figure 4.

TABLE 4 Outer/inner iterations an CPU time to evaluate the leftmost eigenpair of matrix `emilia-923`. For all the update formulas $k_{\max} = 10$ has been set

	nlit	totlin	CPU
L-BFGS	15	224	183.22
L-SR1 (no scaling)	[a]	[a]	[a]
L-SR1 (with scaling)	16	286	194.82
No Update	21	647	371.20

[a]PCG breakdown found at outer iteration #28.

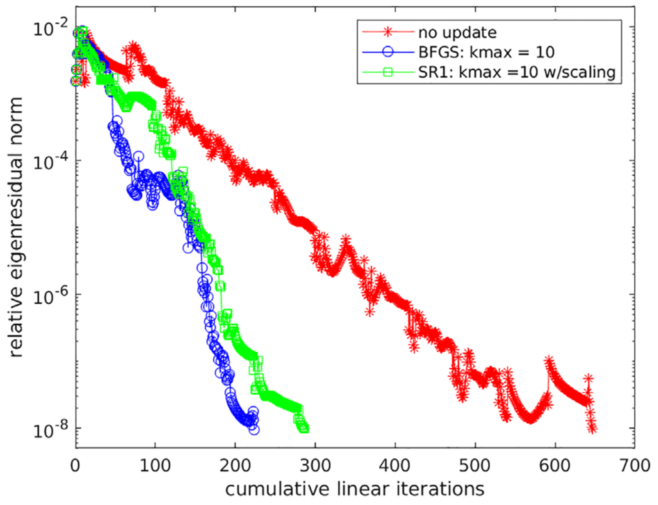

FIGURE 4 Nonlinear converge profile vs cumulative linear iteration number with different preconditioning strategies. Matrix `emilia-923`

The L-SR1 update did not lead to convergence (in fact at nonlinear iteration #28 the PCG method stopped due to a breakdown). In the light of Remark 1, to ensure the positive definiteness of the L-SR1 update, we scaled in this case the Cholesky triangular matrix by a factor 1.4, after roughly estimating the largest eigenvalue of $(LL^T)^{-1}J_0$, as described in Algorithm 3. With this simple modification, the L-SR1 update provides a similar performance as that of the L-BFGS update.

As a general comment, regarding Newton method in eigenvalue computation, the effects of the low-rank updates are much more pronounced as compared to the previous nonlinear problems. Moreover, the relatively high number of nonlinear iterations and the close to indefinite linear systems to be solved at each Newton step suggest the use of higher values of the $k_{\max}$ parameter to improve the properties of the low-rank updates.

5 | CONCLUSIONS

In this paper a compact version of the rank-two L-BFGS and the rank-one SR1 formulas has been used to update an initial preconditioner for the solution of nonlinear systems within the Inexact Newton method. One purpose of this study was to show that the compact formulation, which allows application of the preconditioner in matrix form, can improve the performance obtained in the PCG iteration. The other objective of the paper was to show that the rank-one update SR1, despite not being guaranteed to provide SPD preconditioners in all cases, can yet provide SPD sequences by simply scaling the initial (Cholesky) preconditioner. Moreover, it has been shown that the SR1-updated preconditioner is able to shift some eigenvalues toward one, and a bounded deterioration property regarding the condition number of the preconditioned Jacobians has been stated.

In practice, the construction of the sequence of preconditioners is based on well-known limited memory techniques in order to keep under control the amount of memory, and also the computational time needed to compute and apply the update. From the numerical experiments we can conclude that both the L-BFGS and L-SR1 formulas benefit from the use of compact (block) forms. More interestingly, the experiments have shown that the L-SR1 may be a good alternative to the L-BFGS formula for nonlinear problems with SPD Jacobian. Work is undergoing to provide a parallel implementation of the compact limited memory Quasi-Newton preconditioners to improve a given sparse approximate inverse initial preconditioner in the framework, for example, of sequences of linear systems arising in the eigensolution of very large and sparse matrices in the lines of References 4,26.

ACKNOWLEDGEMENTS

This work was supported by the Spanish Ministerio de Economía y Competitividad under grants MTM2014-58159-P, MTM2017-85669-P, and MTM2017-90682-REDT. The first and third authors have been also partially supported by the INdAM Research group GNCS, 2020 Project: Optimization and advanced linear algebra for problems arising from PDEs.

ORCID

Luca Bergamaschi https://orcid.org/0000-0001-8273-9674
José Marín https://orcid.org/0000-0002-7825-2836
Ángeles Martínez https://orcid.org/0000-0003-4826-1114

REFERENCES

1. Bergamaschi L, Putti M. Mixed finite elements and Newton-type linearization for the solution of Richard's equation. Int J Numer Methods Eng. 1999;45(8):1025–1046.
2. Notay Y. Combination of Jacobi-Davidson and conjugate gradients for the partial symmetric eigenproblem. Numer Linear Algebra Appl. 2002;9(1):21–44.
3. Bergamaschi L, Martinez A. Efficiently preconditioned Inexact Newton methods for large symmetric eigenvalue problems. Optim Methods Softw. 2015;30:301–322.
4. Bergamaschi L, Martinez A. Parallel RFSAI-BFGS preconditioners for large symmetric eigenproblems. J Appl Math. 2013;2013:767042 10 pages.
5. Martinez JM. A theory of secant preconditioners. Math Comput 1993;60(202):681–698.
6. Morales JL, Nocedal J. Automatic preconditioning by limited memory Quasi-Newton updating. SIAM J Optim. 2000;10(4): 1079–1096.
7. Bergamaschi L, Bru R, Martinez A, Putti M. Quasi-Newton preconditioners for the inexact Newton method. Electron Trans Numer Anal. 2006;23:76–87.
8. Nabben R, Vuik C. A comparison of deflation and the balancing preconditioner. SIAM J Sci Comput. 2006;27(5):1742–1759.
9. Freitag MA, Spence A. A tuned preconditioner for inexact inverse iteration applied to Hermitian eigenvalue problems. IMA J Numer Anal. 2008;28(3):522–551.
10. Martinez A. Tuned preconditioners for the eigensolution of large SPD matrices arising in engineering problems. Numer Linear Algebra Appl. 2016;23(3):427–443.
11. Gratton S, Sartenaer A, Tshimanga J. On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides. SIAM J Optim. 2011;21(3):912–935.
12. Bergamaschi L, Bru R, Martinez A. Low-rank update of preconditioners for the inexact Newton method with SPD Jacobian. Math Comput Model. 2011;54(7–8):1863–1873.
13. DeGuchy O, Erway JB, Marcia RF. Compact representation of the full Broyden class of quasi-Newton updates. Numer Linear Algebra Appl. 2018;25(5):E2186.
14. Nocedal J, and Wright SJ. Numerical optimization. Springer Series in Operations Research Springer-Verlag. New York, NY: 1999. /https://doi.org/10.1007/b98874.
15. Dembo RS, Eisenstat SC, Steihaug T. Inexact Newton methods. SIAM J Numer Anal. 1982;19(2):400–408.
16. Kelley CT. Iterative methods for optimization. Frontiers in applied mathematics. Volume 18. Philadelphia, PA: SIAM, 1999. https://doi.org/10.1137/1.9781611970920.
17. Byrd RH, Nocedal J, Schnabel RB. Representations of quasi-Newton matrices and their use in limited memory methods. Math Program. 1994;63(1):129–156.
18. Kelley CT. Iterative methods for optimization. Frontiers in applied mathematics. Volume 16. Philadelphia, PA: SIAM, 1995.
19. Bergamaschi L. A survey of low-rank updates of preconditioners for sequences of symmetric linear systems. Algorithms. 2020;34(2):100.
20. Powers RT, Stormer E. Free states of the canonical anticommutation relations. Commun Math Phys. 1970;16(1):1–33. https://projecteuclid.org:443/euclid.cmp/1103842028.
21. Ipsen I, Nadler B. Refined perturbation bounds for eigenvalues of Hermitian and Non-Hermitian matrices. SIAM J Matrix Anal Appl. 2009;31(1):40–53.
22. Golub GH, van Loan CF. Matrix computation. Baltimore, Maryland: Johns Hopkins University Press, 1991.
23. Simoncini V, Eldén L. Inexact Rayleigh quotient-type methods for eigenvalue computations. BIT Numer Math. 2002;42(1): 159–182.
24. Sleijpen GLG, van der Vorst HA. A Jacobi-Davidson method for linear eigenvalue problems. SIAM J Matrix Anal Appl. 1996;17(2): 401–425.

25. Tapia RA, Dennis JE, Schäfermeyer JP. Inverse, shifted inverse, and Rayleigh quotient iteration as Newton's method. SIAM Rev. 2018;60(1):3–55.
26. Bergamaschi L, Martinez A. Spectral acceleration of parallel iterative eigensolvers for large scale scientific computing. Adv Parallel Comput. 2018;32:107–116.