Les Cahiers du GERAD ISSN: 0711–2440

Quasi-Newton and Krylov methods for the solution of trust-region subproblems

J. Bourhis, O. Cordelier, J.-P. Dussault, O. Mouhtal, D. Orban

G–2026–29

June 2026

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2026
– Bibliothèque et Archives Canada, 2026

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2026
– Library and Archives Canada, 2026

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Quasi-Newton and Krylov methods for the solution of trust-region subproblems

Johann Bourhis [a]

Oihan Cordelier [b]

Jean-Pierre Dussault [c]

Oussama Mouhtal [b]

Dominique Orban [b]

[a] *National Institute of Applied Sciences of Rennes, 35700 Rennes, France*

[b] *GERAD and Department of Mathematics and Industrial Engineering, Montréal (Qc), Canada, H3T 1J4*

[c] *GERAD and Department of Mathematics, Université de Sherbrooke, Sherbrooke (Qc), Canada, J1K 2R1*

johann.bourhis@insa-rennes.fr
oihan.cordelier@polymtl.ca
jean-pierre.dussault@usherbrooke.ca
oussama-2.mouhtal@polymtl.ca
dominique.orban@gerad.ca

June 2026
Les Cahiers du GERAD
G–2026–29

Abstract : We study the solution of symmetric positive-definite linear systems by way of families of full- and limited-memory methods. Our contributions are threefold. We first derive new relationships between the conjugate-gradient method (CG) and quasi-Newton methods of the Broyden class that refine existing results, and clarify when those methods generate the same iterates and enjoy quadratic termination. We extend this perspective to the limited-memory BFGS (LBFGS) method. Next, we examine how DIOM, a limited-memory variant of the full orthogonalization Krylov method (FOM), is akin to LBFGS in that it provides a memory lever that is critical in practical performance. Finally, we generalize LBFGS and DIOM to the computation of trust-region steps for unconstrained, potentially nonconvex, optimization. We report numerical experience on positive-definite linear systems and unconstrained optimization problems. The results show that memory is a key algorithmic lever: LBFGS and DIOM are consistently more robust than CG and often achieve comparable accuracy with fewer Hessian-vector products. They emerge as viable alternatives to CG when high accuracy is desirable or when operations with the Hessian are at a premium. The limited-memory SR1 (LSR1) method can be competitive in full-memory form, but its limited-memory variant suffers from discarded curvature information.

Keywords : Broyden Class; Quasi-Newton Method; Conjugate-Gradient Method; CG; Truncated Conjugate-Gradient Method; Limited-Memory quasi-Newton Method; LBFGS; LSR1; Full Orthogonalization Method; FOM; Direct Incomplete Orthogonalization Method; DIOM; Truncated Direct Incomplete Orthogonalization Method; Trust-Region Algorithm

Acknowledgements: Research of J.-P. Dussault and D. Orban is partially supported by an NSERC Discovery Grant.

1 Introduction

We consider the unconstrained optimization problem

$$\underset{z\in\mathbb{R}^n}{\text{minimize}}\, f(z), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. Newton's method globalized by way of a line-search or trust-region mechanism is among the most widely used techniques to identify a stationary point of (1), and computes a step by approximately minimizing a quadratic model of f about the current iterate. A popular way to compute a trust-region step is to employ the truncated conjugate-gradient (CG) method of Steihaug [35]—a variant of the original method of Hestenes and Stiefel [17] with safeguards to take the trust region and potential negative curvature into account. A similar procedure, also based on CG, exists for line-search methods [7]. One way to derive CG is as a three-term recurrence method based on the Lanczos [19] process. CG may suffer from loss of orthogonality, and, though theory predicts that it terminates in at most n iterations, may require many more to converge in practice. We exploit relationships between CG and quasi-Newton methods of the Broyden [2] class for optimization, on the one hand, and between CG and Krylov methods based on the Arnoldi [1] process on the other, to devise iterative methods that are less sensitive to loss of orthogonality.

Specifically, we investigate the use of quasi-Newton, limited-memory quasi-Newton methods and limited-memory Krylov methods to approximately minimize the quadratic

$$q(x) = \tfrac{1}{2}x^T Ax - b^T x + c, \tag{2}$$

where $A = A^T \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. When A is positive definite (PD), q is strictly convex and (2) admits a unique solution. CG computes the minimum of a strictly convex quadratic function in at most n steps in exact artithmetic [17], a property called quadratic termination. We provide new relationships between CG and quasi-Newton methods of the Broyden class applied to the minimization of a strictly convex quadratic that generalize the well-known statement of Broyden [2]: Broyden-class methods generate the same iterates as CG whenever the quasi-Newton update is positive definite and consequently enjoy quadratic termination.

We focus our numerical study on two specific methods: BFGS, one of the most widely-used Broyden-class quasi-Newton methods, and the full-orthogonalization method (FOM), which is similar in spirit to CG, but is based on the Arnoldi [1] process. Both have large memory requirements. Thus, for practical purposes, we investigate limited-memory variants. We establish that the limited-memory BFGS method (LBFGS) also generates the same iterates as CG in exact arithmetic when applied to a strictly convex quadratic, as does the direct incomplete orthogonalization method (DIOM), a limited-memory variant of FOM.

We report numerical results on ill-conditioned PD systems that show the advantage of LBFGS and DIOM over CG in the presence of loss of orthogonality. We compare LSR1 with CG on the same systems and highlight the severe limitations of LSR1 when memory is low.

We generalize our findings to potentially indefinite systems by devising truncated variants of LBFGS and DIOM, i.e., variants that account for a trust-region constraint; a constraint that confines the iterates to a Euclidean ball centered at the origin. We establish that the truncated variants are appropriate to compute a step as part of a trust-region method for unconstrained, potentially nonconvex, optimization.

Our experiments on nonlinear optimization problems, namely data-assimilation and binary classification problems, further highlight the benefits of limited-memory variants. In particular, LBFGS and DIOM often require fewer Hessian-vector products than CG and, also reduce overall runtime.

Related research

Dixon [10] showed that, for any differentiable $f : \mathbb{R}^n \to \mathbb{R}$, any two methods in the Broyden class generate parallel descent directions when using a *perfect*[1] line search, provided that the Hessian approximation remains nonsingular. Consequently, they all generate the same iterates.

If, in addition, the objective is a strictly convex quadratic, the results of Dixon [10] can be strengthened in several ways. We summarize a few of them relevant here. Broyden [2] showed that these quasi-Newton methods generate the same iterates as the conjugate gradient method of Hestenes and Stiefel [17]. Nazareth [24, §2] showed that the BFGS method with exact line search not only generates the same iterates as CG, but also the same search directions and step lengths. Forsgren and Odland [14, Proposition 1] provide necessary and sufficient conditions for a quasi-Newton method with exact line search, not necessarily of the Broyden type and not necessarily based on the secant equation, to generate search directions parallel to those of CG.

Attention also turned to limited-memory methods. Shanno [34] showed that, under an exact line search, nonlinear CG can be interpreted as a memoryless BFGS update in the sense that if the Hessian approximation is reset to the identity at each iteration, both methods generate the same iterates. His results specialize to the case of a strictly convex quadratic with exact line search, and show that the CG iterates coincide with those of the memoryless BFGS method. Ek and Forsgren [11] introduce a class of limited-memory quasi-Newton Hessian approximations that generate search directions parallel to those of CG and BFGS. However, their results do not appear to cover the limited-memory BFGS method.

Contributions

Our main message is that numerous quasi-Newton methods and certain Krylov methods coincide with CG in exact, but not in floating-point, arithmetic. Using those methods, it is possible to increase the robustness and efficiency of CG in exchange for extra storage. Specifically,

1. As long as the CG iterations are well defined, and a Broyden method does not encounter zero curvature and does not generate a singular approximation, we provide a recurrence for the coefficient of proportionality between its search direction and that of CG (Theorem 3.1). The result includes methods that generate potentially indefinite approximations, even if A is positive definite, and, in particular, the symmetric rank 1 method. The result generalizes one of Broyden [2], who restricted the Broyden parameter to nonnegative values.
2. A characterization of the coefficients of proportionality and step lengths in methods of the *convex* Broyden class as a function of the Broyden parameter (Theorem 3.2). This result shows how step lengths generated by a method in the convex class behave in an orderly manner, whereas those generated by a method that is not in the convex class, such as SR1, can behave erratically.
3. As long as the CG iterations are well defined, the directions and step lengths generated by the limited-memory BFGS method (LBFGS) with *any* memory coincide with those generated by CG. This result was previously thought to hold only for memory 1 [34].
4. We describe how the FOM Krylov method based on the Arnoldi [1] process also coincides with CG when applied to (2) for as long as the CG iterates are well defined. FOM has a limited-memory variant named DIOM, which also coincide with CG for *any* value of the memory parameter.
5. We describe how nonpositive curvature can be detected efficiently in LBFGS and DIOM so those methods can be used to solve trust-region subproblems for potentially nonconvex optimization.
6. We provide accompanying implementations of all the methods described.

[1] A line search is perfect if it is exact and guaranteed to locate the nearest local minimum along the search direction.

Notation

Matrices are denoted by uppercase Latin letters, vectors by lowercase Latin letters and scalars by Greek letters. By convention, all vectors are column vectors. Throughout, x_k denotes an iterate of a method used to minimize (2), $s_k := x_{k+1} - x_k$ is the difference between two consecutive iterates, $g_k := \nabla q(x_k) = Ax_k - b$, and $y_k := g_{k+1} - g_k = As_k$. to denote the difference between two consecutive gradients. If $A = A^T \in \mathbb{R}^{n\times n}$, we write $A \succ 0$ to indicate that A is positive definite (PD). We use the shorthand SPD for *symmetric and positive definite*. The ith component of a vector $z \in \mathbb{R}^n$, for $i = 1, \ldots, n$, is denoted $(z)_i$. All results below are derived under the assumption of exact arithmetic.

2 Background

2.1 Exact line search methods for quadratic functions

If d_k is a descent direction for q in (2) from x_k, i.e., $g_k^T d_k < 0$, performing an exact line search consists in finding a step length

$$\alpha_k \in \underset{\alpha\geq 0}{\operatorname{argmin}}\ q(x_k + \alpha d_k). \tag{3}$$

The current iterate x_k is subsequently updated according to

$$s_k = \alpha_k d_k, \quad x_{k+1} = x_k + s_k.$$

Since q is quadratic (2), if A is SPD, the unique solution to (3) is

$$\alpha_k = -g_k^T d_k / d_k^T A d_k. \tag{4}$$

Note that if A is negative definite, (4) yields the global maximum of q from x_k in the direction d_k, while if A is indefinite and $d_k^T A d_k \neq 0$, (4) yields a stationary point.

2.2 Methods of the Broyden class

Secant quasi-Newton methods generate descent directions by maintaining an approximation $H_k = H_k^T$ such that $H_k^{-1} \approx \nabla^2 q(x_k) = A$ that is updated to satisfy the secant equation

$$H_{k+1} y_k = s_k. \tag{5}$$

Whenever $s_k^T y_k \neq 0$ and $y_k^T H_k y_k \neq 0$, the Broyden family of updates is given by

$$H_{k+1} = H_k + \frac{s_k s_k^T}{s_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \phi_k\,(y_k^T H_k y_k)(v_k v_k^T), \tag{6a}$$

$$v_k = \frac{s_k}{s_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k}, \tag{6b}$$

where $\phi_k \in \mathbb{R}$ is a parameter [9]. If we let $\rho_k := 1/s_k^T y_k$, the BFGS [3, 12, 15, 33] and the DFP [6, 13] formulae result, respectively, from $\phi_k = 1$ and $\phi_k = 0$:

$$H_{k+1}^{\text{BFGS}} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T, \tag{7}$$

$$H_{k+1}^{\text{DFP}} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \rho_k s_k s_k^T. \tag{8}$$

Assuming that $(s_k - H_k y_k)^T y_k \neq 0$, the choice

$$\phi_k^{\text{SR1}} = \frac{y_k^T s_k}{(s_k - H_k y_k)^T y_k}, \tag{9}$$

yields the SR1 update [8]

$$H_{k+1}^{\text{SR1}} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}. \tag{10}$$

To avoid a proliferation of superscripts, we write H_{k+1} without explicit dependency on ϕ_k unless we refer to H_{k+1}^{BFGS}, H_{k+1}^{DFP} or H_{k+1}^{SR1}. However, it will be necessary to keep track of ϕ_k when we talk about search directions. Hence, given an approximation H_k and a value of ϕ_{k-1}, the quasi-Newton search direction is

$$d_k^{\phi_{k-1}} = -H_k g_k. \tag{11}$$

The following result states that (6) is almost always nonsingular.

Proposition 2.1 (9, Lemma 4.2). *Assume that H_k is nonsingular, $s_k^T y_k \neq 0$, and $y_k^T H_k y_k \neq 0$. Let $B_k = H_k^{-1}$. Then, H_{k+1} defined by (6) is singular if and only if $\phi_k = \phi_k^c$ where*

$$\phi_k^c = \frac{(y_k^T s_k)^2}{(y_k^T s_k)^2 - (y_k^T H_k y_k)(s_k^T B_k s_k)}. \tag{12}$$

Proposition 2.2 (26, p. 151). *Let $H_k = H_k^T \succ 0$ and $s_k^T y_k > 0$. If $\phi_k > \phi_k^c$, then $H_{k+1} = H_{k+1}^T \succ 0$. If $\phi_k < \phi_k^c$, H_{k+1} is nonsingular but may be indefinite.*

It is necessary to initialize all quasi-Newton methods with an approximation $H_0 = H_0^T$ at iteration $k = 0$. If the update used has hereditary positive definiteness, such as the updates of Proposition 2.2 with $\phi_k > \phi_k^c$, it is common to pick $H_0 \succ 0$, and typically as a multiple of the identity.

2.3 The preconditioned conjugate gradient method

Whenever $A \succ 0$ in (2), CG generates iterates by performing an exact line search along A-conjugate descent directions $\{d_k^{\text{PCG}}\}$, i.e., such that $d_i^T A d_j = 0$ for $i \neq j$. If a preconditioner $H_0 = H_0^T \succ 0$ is available, the kth preconditioned CG (PCG) descent direction is

$$d_0^{\text{PCG}} = -H_0 g_0, \qquad d_k^{\text{PCG}} = -H_0 g_k + \frac{g_k^T H_0 g_k}{g_{k-1}^T H_0 g_{k-1}} d_{k-1}^{\text{PCG}} \quad (k \geq 1). \tag{13}$$

Thus, the kth iterate of PCG is generated by performing an exact line search along these directions. The algorithm below describe this method.

Algorithm 1 PCG

$g_0 = \nabla q(x_0)$, $z_0 = -H_0 g_0$, $\rho_0 = -g_0^T z_0$, $d_0^{\text{PCG}} = z_0$
for $k = 0, 1, \ldots$ **do**
  $\alpha_k^{\text{PCG}} = \rho_k / (d_k^{\text{PCG}})^T A d_k^{\text{PCG}}$
  $x_{k+1}^{\text{PCG}} = x_k^{\text{PCG}} + \alpha_k^{\text{PCG}} d_k^{\text{PCG}}$
  $g_{k+1} = g_k + \alpha_k^{\text{PCG}} A d_k^{\text{PCG}}$ $= \nabla q(x_{k+1})$
  $z_{k+1} = -H_0 g_{k+1}$
  $\rho_{k+1}^{\text{PCG}} = -g_{k+1}^T z_{k+1}$
  $\beta_k = \rho_{k+1} / \rho_k$
  $d_{k+1}^{\text{PCG}} = z_{k+1} + \beta_{k+1} d_k^{\text{PCG}}$

CG is a special case of PCG with $H_0 = I$. In the following, we present the main results for PCG, which will be used throughout this paper. We assume that q in (2) is strictly convex and that $H_0 = H_0^T \succ 0$.

Proposition 2.3. *Let k be such that $g_k \neq 0$. The gradients calculated by PCG satisfy,*

$$g_k^T H_0 g_i = 0 \quad (i < k), \tag{14a}$$

$$g_k^T d_i^{PCG} = 0 \quad (i < k). \tag{14b}$$

Proof. The equivalence between applying PCG to the first-order optimality system $Ax = b$ and applying CG to the system $H_0^{1/2} A H_0^{1/2} x = H_0^{1/2} b$ [16, Chap. 11] allows us to invoke the standard results of [17, Theorem 5:1]. □

Proposition 2.4. *The directions generated by PCG with preconditioner $H_0 = H_0^T \succ 0$ are A-conjugate. Consequently, PCG finds the unique minimum of q in at most n steps.*

Proof. The same equivalence with standard CG used in the previous proof yields this result directly from [17, Theorem 5:2]. □

If q is a nonconvex quadratic, (4) corresponds to the step to a stationary point of q from x_k in the direction d_k. If $d_k^T A d_k \neq 0$ for all k, quadratic termination still takes place and CG finds the unique critical point of q in at most n steps. Such is the case if A is quasi-definite [36] for certain linear terms b [27].

3 Relationship between the two methods

In the next few sections, we require one or more of the following assumptions.

Assumption 3.1. In (2), $A = A^T \succ 0$.

Assumption 3.2. Methods of the Broyden class are applied with exact line search and a given initial matrix $H_0 = H_0^T \succ 0$.

Assumption 3.3. The PCG method is applied with the same preconditioner $H_0 = H_0^T \succ 0$ as the initial matrix used in methods of the Broyden class.

The following result generalizes existing results by providing a recurrence formula for the proportionality factor between a search direction generated by a Broyden method and that generated by CG. It expands and completes Broyden [2, §3 and Theorem 1], who restricts $\phi \geq 0$. The proof follows the pattern of Nazareth [24, §2].

Theorem 3.1. *Let Assumptions 3.2 and 3.3 be satisfied, but A is not necessarily PD. Let x_0 be any starting point, and assume the PCG iterations $0, \ldots, j$ are well defined. Assume that for $k = 0, \ldots, j$, $\phi_k \neq \phi_k^c$ and*

$$s_k^T A s_k \neq 0, \tag{15}$$

$$y_k^T H_k y_k \neq 0. \tag{16}$$

Then, as long as $g_k \neq 0$, there exists $\gamma_k^{\phi_{k-1}} \neq 0$ such that $d_k^{\phi_{k-1}} = \gamma_k^{\phi_{k-1}} d_k^{PCG}$, where d_k^{PCG} is the kth conjugate gradient direction preconditioned with H_0. In addition,

$$d_{k+1}^{\phi_k} = \frac{\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k + \phi_k g_{k+1}^T H_0 g_{k+1}}{\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}} d_{k+1}^{PCG} = \gamma_{k+1}^{\phi_k} d_{k+1}^{PCG}. \tag{17}$$

As a result, the iterates s_k generated by any method in the Broyden family are independent of ϕ_k and coincide with the PCG iterates.

Proof. We establish by induction that the vectors $d_k^{\phi_{k-1}}$ generated by the Broyden class methods are colinear to the vectors d_k^{PCG} and that

$$H_i g_k = H_0 g_k, \ i = 0, ..., k-1. \tag{18}$$

If the two vectors $d_k^{\phi_{k-1}}$ and d_k^{PCG} are colinear, it follows that with an exact line search the two methods calculate the same iterates $s_k = \alpha_k^{\mathrm{PCG}} d_k^{\mathrm{PCG}} = \alpha_k^{\phi_k} d_k^{\phi_{k-1}}$. For $k = 0$, $d_0^{\phi_k} = d_0^{\mathrm{PCG}} = -H_0 g_0$ and hence,

$s_0^{\text{PCG}} = s_0^{\phi_k}$. Assume that the induction hypothesis holds at iteration $k \geq 0$ and $g_{k+1} \neq 0$. Because $\phi_k \neq \phi_k^c$, Proposition 2.2 implies that $H_1, \ldots, H_k$ are nonsingular. Under the induction hypothesis, iterates $s_1, s_2, \ldots, s_k$ of all Broyden methods are the same of those generated by CG. Henceforth, $s_i^T g_{k+1} = 0$ by (14b) for $i = 0, 1, ..., k$ and by (6b) and (18),

$$v_i^T g_{k+1} = \left(\frac{s_i^T}{s_i^T y_i} - \frac{y_i^T H_i}{y_i^T H_i y_i} \right) g_{k+1} = -\frac{y_i^T H_0 g_{k+1}}{y_i^T H_i y_i}. \tag{19}$$

Then by (6) and (18),

$$\begin{aligned} H_{i+1} g_{k+1} &= H_i g_{k+1} - \frac{H_i y_i y_i^T H_i g_{k+1}}{y_i^T H_i y_i} + \phi_i (y_i^T H_i y_i) v_i v_i^T g_{k+1} \\ &= H_0 g_{k+1} - \frac{H_i y_i y_i^T H_0 g_{k+1}}{y_i^T H_i y_i} - \phi_i v_i y_i^T H_0 g_{k+1}. \end{aligned} \tag{20}$$

Furthermore, by the definition of y_k and (14a), $y_i^T H_0 g_{k+1} = 0$ for $i = 0, 1, ..., k-1$. Then $H_i g_{k+1} = H_0 g_{k+1}$ for $i = 0, 1, ..., k$, that proves (18).
Still according to (14a),

$$y_k^T H_0 g_{k+1} = g_{k+1}^T H_0 g_{k+1}. \tag{21}$$

By induction hypothesis, there exists a scalar $\gamma_k^{\phi_{k-1}} \neq 0$ such that

$$d_k^{\phi_{k-1}} = \gamma_k^{\phi_{k-1}} d_k^{\text{PCG}}. \tag{22}$$

According to (14b) and (13), we have

$$y_k^T d_k^{\text{PCG}} = \left(g_{k+1}^T - g_k^T \right) \left(-H_0 g_k + \beta_k d_{k-1}^{\text{PCG}} \right) = g_k^T H_0 g_k, \tag{23}$$

$$H_k y_k = H_k \left(g_{k+1} - g_k \right) = H_0 g_{k+1} + d_k^{\phi_{k-1}} = H_0 g_{k+1} + \gamma_k^{\phi_{k-1}} d_k^{\text{PCG}}. \tag{24}$$

So,

$$y_k^T H_k y_k = g_{k+1}^T H_0 g_{k+1} + \gamma_k^{\phi_{k-1}} g_k^T H_0 g_k. \tag{25}$$

By taking again (20) with $i = k$ and by (6b) and (21),

$$\begin{aligned} H_{k+1} g_{k+1} &= H_0 g_{k+1} - \frac{H_k y_k y_k^T H_0 g_{k+1}}{y_k^T H_k y_k} - \phi_k g_{k+1}^T H_0 g_{k+1} \left(\frac{s_k}{s_k^T y_k} - \frac{H_k y_k}{y_k^T H_k y_k} \right) \\ &= H_0 g_{k+1} - \frac{g_{k+1}^T H_0 g_{k+1} H_k y_k}{y_k^T H_k y_k} + \phi_k g_{k+1}^T H_0 g_{k+1} \left(\frac{H_k y_k}{y_k^T H_k y_k} - \frac{d_k^{\text{PCG}}}{y_k^T d_k^{\text{PCG}}} \right) \\ &= H_0 g_{k+1} \left(1 - \frac{g_{k+1}^T H_0 g_{k+1}}{y_k^T H_k y_k} + \frac{\phi_k g_{k+1}^T H_0 g_{k+1}}{y_k^T H_k y_k} \right) \\ &\quad - d_k^{\text{PCG}} \left(\frac{\gamma_k^{\phi_{k-1}} g_{k+1}^T H_0 g_{k+1}}{y_k^T H_k y_k} - \frac{\gamma_k^{\phi_{k-1}} \phi_k g_{k+1}^T H_0 g_{k+1}}{y_k^T H_k y_k} + \frac{\phi_k g_{k+1}^T H_0 g_{k+1}}{y_k^T d_k^{\text{PCG}}} \right) \\ &= \left(\frac{\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k + \phi_k g_{k+1}^T H_0 g_{k+1}}{\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}} \right) \left(H_0 g_{k+1} - \frac{g_{k+1}^T H_0 g_{k+1}}{g_k^T H_0 g_k} d_k^{\text{PCG}} \right). \end{aligned} \tag{26}$$

So we have by (11) and (13) the relationship (17). Thanks to the assumptions (15) and (16), d_{k+1}^{PCG} and $d_{k+1}^{\phi_k}$ are well-defined. Finally, $\phi_k \neq \phi_k^c$ brings by proposition 2.1 that $H_{k+1} g_{k+1} \neq 0$ and the two directions are colinear. So, with an exact line search, we have $s_{k+1} = \alpha_{k+1}^{\phi_k} d_{k+1}^{\phi_k} = \alpha_{k+1}^{\text{PCG}} d_{k+1}^{\text{PCG}}$ under the assumption that $s_{k+1}^T A s_{k+1} \neq 0$. □

The following result generalizes [2, Corollary 1] to $\phi_k \neq \phi_k^c$.

Corollary 3.1. *Let the assumptions of Theorem 3.1 be satisfied. Methods of the Broyden class inherit the conjugacy property as well as quadratic termination if $j = n$. The secant equation*

$$H_{k+1} y_i = s_i, \quad (i \leq j) \tag{27}$$

holds for all previous iterates. It follows that if n steps are performed and A is nonsingular, $H_n = A^{-1}$.

Clearly, assumptions on A in Theorem 3.1 and Corollary 3.1 hold if A is SPD.

Corollary 3.2. *Let Assumptions 3.1 and 3.2 be satisfied. A sufficient condition for (16) to hold is that $\phi_k > \phi_k^c$. Thus, (17) holds and quadratic termination occurs.*

Proof. If (2) is strictly convex, $s_k^T A s_k > 0$ and (15) holds for all k. As a result of Proposition 2.2 and because $H_0 \succ 0$, $H_k \succ 0$ for all k. Thus, $y_k^T H_k y_k > 0$ and (16) holds for all k. The result follows from Theorem 3.1. □

Lemma 3.1. *With the notation of Theorem 3.1 we can write*

$$\phi_k^c = -\gamma_k^{\phi_{k-1}} \frac{g_k^T H_0 g_k}{g_{k+1}^T H_0 g_{k+1}}. \tag{28}$$

Proof. By (12),

$$\phi_k^c = \frac{(y_k^T s_k)^2}{(y_k^T s_k)^2 - (y_k^T H_k y_k)(s_k^T B_k s_k)} = \frac{(y_k^T d_k^{\phi_{k-1}})^2}{(y_k^T d_k^{\phi_{k-1}})^2 - (y_k^T H_k y_k)(d_k^{\phi_{k-1}{}^T} B_k d_k^{\phi_{k-1}})}.$$

Furthermore, $B_k d_k^{\phi_{k-1}} = -g_k$. So, by (22),

$$d_k^{\phi_{k-1}{}^T} B_k d_k^{\phi_{k-1}} = -\gamma_k^{\phi_{k-1}} g_k^T d_k^{\text{PCG}} = \gamma_k^{\phi_{k-1}} g_k^T H_0 g_k. \tag{29}$$

It follows from (23) and (25) that

$$\begin{aligned} \phi_k^c &= \frac{\left(\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k\right)^2}{\left(\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k\right)^2 - \left(\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}\right) \gamma_k^{\phi_{k-1}} g_k^T H_0 g_k} \\ &= -\frac{\gamma_k^{\phi_{k-1}} g_k^T H_0 g_k}{g_{k+1}^T H_0 g_{k+1}}. \end{aligned}$$

□

Corollary 3.3. *Let Assumptions 3.1 and 3.2 be satisfied. Then, $\gamma_k^{\phi_{k-1}} > 0$ provided $\phi_k > \phi_k^c$.*

Proof. By (25) and Proposition 2.2, the denominator of $\gamma_k^{\phi_{k-1}}$ is positive because (2) is strictly convex, H_0 PD and $\phi_k > \phi_k^c$. If $\gamma_{k-1}^{\phi_k} > 0$, (28) implies $\gamma_k^{\phi_{k-1}} > 0$ if $\phi_k > \phi_k^c$. Finally, $\gamma_0^{\phi_k} = 1$. By induction, $\gamma_k^{\phi_{k-1}} > 0$ for all $k \geq 0$. □

Corollary 3.4. *Let Assumptions 3.1 and 3.2 be satisfied. Methods of the Broyden class with $\phi_k \geq 0$ are always well-defined.*

Proof. On a strictly convex quadratic function, it was proved by Corollary 3.3 that if $\phi_k > \phi_k^c$, then $\gamma_k^{\phi_{k-1}} > 0$ at each iteration. Thus by (28), $\phi_k^c < 0$. It follows by Corollary 3.2 that the Broyden class methods are well-defined when $\phi_k \geq 0$. □

From Theorem 3.1, if two methods of the Broyden class are applied to minimize (2), one with parameter $\phi_k^{(1)}$ and the other with parameter $\phi_k^{(2)}$, the search directions at iteration $k+1$ are related via

$$d_{k+1}^{\phi_k^{(1)}} = \frac{\gamma_{k+1}^{\phi_k^{(1)}}}{\gamma_{k+1}^{\phi_k^{(2)}}} d_{k+1}^{\phi_k^{(2)}}. \tag{30}$$

The next two results examine special cases.

Proposition 3.1. *Let Assumptions 3.1 and 3.2 be satisfied. Let d_k^{BFGS} and d_k^{DFP} be the directions generated respectively by the BFGS and DFP methods. Let γ_k^{BFGS} and γ_k^{DFP} the coefficients such that $d_k^{BFGS} = \gamma_k^{BFGS} d_k^{PCG}$ and $d_k^{DFP} = \gamma_k^{DFP} d_k^{PCG}$. Then,*

$$d_{k+1}^{BFGS} = d_{k+1}^{PCG}, \tag{31}$$

i.e., $\gamma_k^{BFGS} = 1$ for all k. In addition,

$$d_{k+1}^{DFP} = \frac{\gamma_k^{DFP} g_k^T H_0 g_k}{\gamma_k^{DFP} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}} d_{k+1}^{PCG}. \tag{32}$$

Proof. The BFGS and DFP methods correspond to $\phi_k = 1$ and $\phi_k = 0$ in (17). □

Proposition 3.2. *Let Assumptions 3.1 and 3.2 be satisfied. Let d_k^{SR1} be the direction generated by the SR1 method and let γ_k^{SR1} be the coefficient such that $d_k^{SR1} = \gamma_k^{SR1} d_k^{PCG}$. If*

$$(s_k - H_k y_k)^T y_k \neq 0, \tag{33}$$

then ϕ_k^{SR1} is well-defined and (9) exists. In addition,

$$d_{k+1}^{SR1} = \frac{\left(\gamma_k^{SR1} - \alpha_k^{PCG}\right) g_k^T H_0 g_k}{\left(\gamma_k^{SR1} - \alpha_k^{PCG}\right) g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}} d_{k+1}^{PCG}. \tag{34}$$

Proof. The assumption $(s_k - H_k y_k)^T y_k \neq 0$ is precisely the condition under which the SR1 parameter (9) is well-defined, and hence the SR1 update belongs to the Broyden family with $\phi_k = \phi_k^{\mathrm{SR1}}$. Since the iterates generated by the Broyden family coincide with those of PCG by Theorem 3.1, we have $s_k = \alpha_k^{\mathrm{PCG}} d_k^{\mathrm{PCG}}$. Using (23) and (25), with $\gamma_k^{\phi_{k-1}} = \gamma_k^{\mathrm{SR1}}$, gives

$$\begin{aligned} y_k^T s_k &= \alpha_k^{\mathrm{PCG}} y_k^T d_k^{\mathrm{PCG}} = \alpha_k^{\mathrm{PCG}} g_k^T H_0 g_k, \\ y_k^T H_k y_k &= \gamma_k^{\mathrm{SR1}} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}. \end{aligned}$$

Therefore,

$$\begin{aligned} \phi_k^{\mathrm{SR1}} &= \frac{\alpha_k^{\mathrm{PCG}} g_k^T H_0 g_k}{\left(\alpha_k^{\mathrm{PCG}} - \gamma_k^{\mathrm{SR1}}\right) g_k^T H_0 g_k - g_{k+1}^T H_0 g_{k+1}} \\ &= -\frac{\alpha_k^{\mathrm{PCG}} g_k^T H_0 g_k}{\left(\gamma_k^{\mathrm{SR1}} - \alpha_k^{\mathrm{PCG}}\right) g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}}. \end{aligned}$$

Substituting this expression for ϕ_k^{SR1} in the numerator of (17) yields

$$\begin{aligned} &\gamma_k^{\mathrm{SR1}} g_k^T H_0 g_k + \phi_k^{\mathrm{SR1}} g_{k+1}^T H_0 g_{k+1} \\ &\qquad = \gamma_k^{\mathrm{SR1}} g_k^T H_0 g_k - \frac{\alpha_k^{\mathrm{PCG}} g_k^T H_0 g_k \, g_{k+1}^T H_0 g_{k+1}}{\left(\gamma_k^{\mathrm{SR1}} - \alpha_k^{\mathrm{PCG}}\right) g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}} \end{aligned}$$

$$= \frac{\gamma_k^{\text{SR1}} g_k^T H_0 g_k \left[\left(\gamma_k^{\text{SR1}} - \alpha_k^{\text{PCG}}\right) g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1} \right] - \alpha_k^{\text{PCG}} g_k^T H_0 g_k \, g_{k+1}^T H_0 g_{k+1}}{\left(\gamma_k^{\text{SR1}} - \alpha_k^{\text{PCG}}\right) g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}}$$

$$= \frac{\gamma_k^{\text{SR1}} \left(\gamma_k^{\text{SR1}} - \alpha_k^{\text{PCG}}\right) \left(g_k^T H_0 g_k\right)^2 + \left(\gamma_k^{\text{SR1}} - \alpha_k^{\text{PCG}}\right) g_k^T H_0 g_k \, g_{k+1}^T H_0 g_{k+1}}{\left(\gamma_k^{\text{SR1}} - \alpha_k^{\text{PCG}}\right) g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}}$$

$$= \frac{\left(\gamma_k^{\text{SR1}} - \alpha_k^{\text{PCG}}\right) g_k^T H_0 g_k \left(\gamma_k^{\text{SR1}} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}\right)}{\left(\gamma_k^{\text{SR1}} - \alpha_k^{\text{PCG}}\right) g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}}.$$

The result follows by dividing the above by $\gamma_k^{\text{SR1}} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}$. □

The next result concerns methods of the *convex* Broyden class, i.e., those for which $0 \leq \phi_k \leq 1$.

Theorem 3.2. *Let Assumption 3.2 be satisfied. Consider two methods of the Broyden class, one with parameter $\phi_k^{(1)}$ and the other with parameter $\phi_k^{(2)}$ such that $0 \leq \phi_k^{(1)} \leq \phi_k^{(2)} \leq 1$ for all k. Let $\gamma_k^{(1)}$ and $\gamma_k^{(2)}$ be the coefficients defined by (17) for each method. Then*

$$0 < \gamma_k^{(1)} \leq \gamma_k^{(2)} \leq 1. \tag{35}$$

Let $\alpha_k^{(1)}$ and $\alpha_k^{(2)}$ be the optimal step lengths associated with each method at iteration k. We have

$$\alpha_k^{(2)} \leq \alpha_k^{(1)} \quad \text{if } s_k^T A s_k > 0, \tag{36a}$$

$$\alpha_k^{(1)} \leq \alpha_k^{(2)} \quad \text{if } s_k^T A s_k < 0. \tag{36b}$$

Finally, let α_k^{PCG} the optimal step length associated to PCG with preconditioner H_0 and let $\alpha_k^{\phi_k}$ be the optimal step length of any given method in the convex Broyden class. We have

$$0 < \alpha_k^{PCG} \leq \alpha_k^{\phi_k} \quad \text{if } s_k^T A s_k > 0, \tag{37a}$$

$$\alpha_k^{\phi_k} \leq \alpha_k^{PCG} < 0 \quad \text{if } s_k^T A s_k < 0. \tag{37b}$$

Proof. By Corollary 3.2 and Lemma 3.1, $\gamma_{k+1}^{\phi_k} > 0$ if $\phi_k > \phi_k^c$ and $\gamma_k^{\phi_{k-1}} > 0$. Because $\phi_k^c < 0$ when $\gamma_k^{\phi_{k-1}} \geq 0$, it is equivalent to $\phi_k \geq 0$ and $\gamma_k^{\phi_{k-1}} > 0$. In addition $\gamma_0^{\phi_k} = 1 > 0$ so $\gamma_k^{\phi_{k-1}} > 0$ for all k. By noting that

$$\gamma_k^{\phi_{k-1}} = \alpha_k^{\text{PCG}} / \alpha_k^{\phi_k}, \tag{38}$$

it follows that $\alpha_k^{\phi_k}$ has the same sign as α_k^{PCG}, and by transitivity $\alpha_k^{(1)}$ and $\alpha_k^{(2)}$ have the same sign as well. Let $\gamma_{k+1}^{(1)}$ and $\gamma_{k+1}^{(2)}$ obtained by replacing ϕ_k by $\phi_k^{(1)}$ and $\phi_k^{(2)}$ in (17). Showing by induction that for all k, $\gamma_k^{(1)} \leq \gamma_k^{(2)}$ if $0 \leq \phi_k^{(1)} \leq \phi_k^{(2)} \leq 1$. To begin, we have $\gamma_0^{(1)} = \gamma_0^{(2)} = 1$ as $d_0 = -H_0 g_0$. Assuming that $\gamma_k^{(1)} \leq \gamma_k^{(2)}$ until one $k \geq 0$ and showing that it holds true at the rank $k+1$. We have for $i = 1, 2$

$$\gamma_{k+1}^{(i)} = \frac{\gamma_k^{(i)} g_k^T H_0 g_k + \phi_k^{(i)} g_{k+1}^T H_0 g_{k+1}}{\gamma_k^{(i)} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}} = 1 + \frac{g_{k+1}^T H_0 g_{k+1} \left(\phi_k^{(i)} - 1\right)}{\gamma_k^{(i)} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}}$$

$$= 1 - \frac{g_{k+1}^T H_0 g_{k+1} \left|\phi_k^{(i)} - 1\right|}{\gamma_k^{(i)} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}}.$$

The last relation is due to the fact that $\phi_k^{(1)} \in [0, 1]$. Because $0 \leq \phi_k^{(1)} \leq \phi_k^{(2)} \leq 1$,

$$\left|\phi_k^{(1)} - 1\right| \geq \left|\phi_k^{(2)} - 1\right|. \tag{39}$$

As by induction hypothesis $0 < \gamma_k^{(1)} \leq \gamma_k^{(2)}$, then

$$\gamma_k^{(1)} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1} \leq \gamma_k^{(2)} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}. \tag{40}$$

It follows that

$$\gamma_{k+1}^{(1)} \leq 1 - \frac{g_{k+1}^T H_0 g_{k+1} \left| \phi_k^{(2)} - 1 \right|}{\gamma_k^{(2)} g_k^T H_0 g_k + g_{k+1}^T H_0 g_{k+1}} = \gamma_{k+1}^{(2)}. \tag{41}$$

In addition, it is clear that $\gamma_k^{\phi_{k-1}} \leq 1$ for all $0 \leq \phi_k \leq 1$. By induction, we have shown (35). Furthermore, by (38) we have

$$\left| \alpha_k^{(2)} \right| \leq \left| \alpha_k^{(1)} \right|. \tag{42}$$

We obtain (36) by noting that $\alpha_k^{(1)}$ and $\alpha_k^{(2)}$ have the same sign as α_k^{PCG}, which in turn has the same sign as $s_k^T A s_k$ by (4) and (13). To finish, we establish (37) by noting with (31) that $\alpha_k^{\text{PCG}} = \alpha_k^{\text{BFGS}}$ and by replacing $\alpha_k^{(2)}$ by α_k^{PCG} with $\phi_k^{(2)} = 1$. □

The step length behavior of Theorem 3.2 is illustrated in Figure 1.

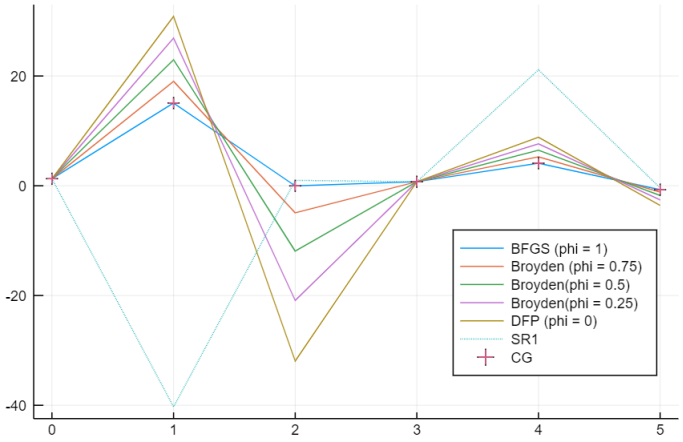

Figure 1: Optimal step lengths α_k as a function of the iteration number k. The optimal step lengths of the convex Broyden class methods are ordered whereas the optimal step lengths of SR1 show an erratic behaviour.

4 The Limited-Memory BFGS Method

Results of the previous section suggest that methods of the Broyden class could be used in place of PCG wherever it is typically used. One such usage is in the computation of steps for the minimization of a nonlinear, potentially nonconvex function. However, the usual disadvantage of Broyden methods is that they tend to generate dense approximations H_k, and are thus impractical for large-scale problems. In this section, we examine the relationship between the most widely-used limited-memory Broyden method and PCG.

The LBFGS(m) method is a limited-memory quasi-Newton method. Instead of storing a matrix H_k of size $n \times n$, we store a limited amount m of couples (s_i, y_i) to reconstruct an approximation of the product $H_k g_k$.

Let

$$U_k = I - \rho_k y_k s_k^T, \quad \text{and} \quad \rho_k = 1/y_k^T s_k. \tag{43}$$

In [20], it is shown for $m = k$ and $H_k^0 = H_0$, that the matrix

$$H_k^m = \left(U_{k-1}^T \ldots U_{k-m}^T \right) H_k^0 \left(U_{k-m} \ldots U_{k-1} \right) \tag{44}$$

$$+ \rho_{k-m}\left(U_{k-1}^T ... U_{k-m+1}^T\right) s_{k-m} s_{k-m}^T \left(U_{k-m+1} ... U_{k-1}\right)$$
$$+ \rho_{k-m+1}\left(U_{k-1}^T ... U_{k-m+2}^T\right) s_{k-m+1} s_{k-m+1}^T \left(U_{k-m+2} ... U_{k-1}\right)$$
$$+ \cdots + \rho_{k-1} s_{k-1} s_{k-1}^T$$

is exactly the one calculated by the BFGS method (7). If we store a small number of the most recent couples (s_i, y_i), we lose some information necessary to reconstruct H_k^{BFGS} but we hope to have kept relevant information to determine a good descent direction $d_k^m = -H_k^m g_k$.

As it turns out, the LBFGS(m) method possesses the same good properties as BFGS in relation with PCG.

Theorem 4.1. *Assume that A is not necessarily PD. Let x_0 be any starting point and let H_0 be any SPD matrix. Consider, on the one hand, PCG applied to (2) from x_0 with preconditioner H_0. Consider, on the other hand, the LBFGS(m) method with $m \geq 1$ and exact line search to minimize (2) with $H_k^0 = H_0$ for all k. Assume that iterations $k = 0, \ldots, j$ of both methods are well defined. If $s_k^T y_k \neq 0$ for all $k = 0, \ldots, j$, then, as long as $g_k \neq 0$, the search directions d_k^m, step lengths α_k^m, and iterates s_k generated by LBFGS(m) are identical to those generated by PCG. Consequently, if $j = n$, quadratic termination occurs.*

Proof. As in Theorem 3.1, we show that the directions d_k^m generated by LBFGS(m) are colinear to those generated by PCG (13). We note that for $k = 0$, $d_0^m = d_0^{\text{PCG}} = -H_0 g_0$. Assuming that we have the colinearity until a rank $k \geq 0$ and showing that it holds at the rank $k+1$ if $g_{k+1} \neq 0$. Under the induction hypothesis, the current points $x_1, .., x_{k+1}$ generated by LBFGS(m) are identical to those generated by CG. Henceforth, by (14b), $s_i^T g_{k+1} = 0$ for all $i = 0, ..., k$. So, we have

$$U_i g_{k+1} = \left(I - \rho_i y_i s_i^T\right) g_{k+1} = g_{k+1}, \ i = 0, ..., k. \tag{45}$$

By (14a)

$$U_i^T H_0 g_{k+1} = \left(I - \rho_i s_i y_i^T\right) H_0 g_{k+1} = H_0 g_{k+1}, \ i = 0, ..., k-1. \tag{46}$$

By applying these two relations to (44), we have,

$$H_{k+1}^m g_{k+1} = \left(U_k^T ... U_{k+1-m}^T\right) H_{k+1}^0 g_{k+1}. \tag{47}$$

With $H_{k+1}^0 = H_0$, it follows that

$$\begin{aligned} -H_{k+1}^m g_{k+1} &= -U_k^T H_0 g_{k+1} = -H_0 g_{k+1} + \rho_k s_k \left(g_{k+1}^T - g_k^T\right) H_0 g_{k+1} \\ &= -H_0 g_{k+1} + \frac{g_{k+1}^T H_0 g_{k+1}}{y_k^T s_k} s_k = -H_0 g_{k+1} + \frac{g_{k+1}^T H_0 g_{k+1}}{y_k^T d_k^{\text{PCG}}} d_k^{\text{PCG}}. \end{aligned}$$

So by (23) and (13),

$$d_{k+1}^m = -H_0 g_{k+1} + \frac{g_{k+1}^T H_0 g_{k+1}}{g_k^T H_0 g_k} d_k^{\text{PCG}} = d_{k+1}^{\text{PCG}}. \tag{48}$$

So d_{k+1}^m and d_{k+1}^{PCG} are colinear because H_0 is non singular. It follows with an exact line search that the iterates s_k are identical for both methods and for any $m \geq 1$. □

Clearly, a sufficient condition for both methods to be well defined in Theorem 4.1 is that A be PD.

Theorem 4.1 can be interpreted as follow. In the BFGS update (31), only the last couple (y_k, s_k) is useful to calculate the descent direction d_k^m in the quadratic case. However, we will see in numerical experiments that the other couples in storage are important for the robustness and accuracy of LBFGS(m).

5 The Limited-Memory Full Orthogonalization Method

In this section, we turn to another class of methods that are known, or can be shown, to coincide with PCG when applied to (2): Krylov-subspace methods based on the Arnoldi process. We first present the general framework and then specialize to the case where A is SPD.

5.1 Arnoldi Process

Given an initial vector v_1, the kth Krylov subspace is $\mathcal{K}_k(A, v_1) := \text{span}\{v_1, Av_1, \ldots, A^{k-1}v_1\}$, $k \geq 1$. The Arnoldi process builds an orthonormal basis of $\mathcal{K}_k(A, v_1)$ via Algorithm 2.

Algorithm 2 Arnoldi Process.

Require: v_1 such that $\|v_1\|_2 = 1$
1: **for** $k = 1, 2, \ldots$ **do**
2: $w = Av_k$
3: **for** $i = 1, \ldots, k$ **do**
4: set $t_{i,k} = v_i^T w$ and $w \leftarrow w - t_{i,k} v_i$
5: $t_{k+1,k} v_{k+1} = w$ $t_{k+1,k} > 0$ *such that* $\|v_{k+1}\| = 1$

By construction, k iterations of Algorithm 2 in exact arithmetic generate $V_k = \begin{bmatrix} v_1 & \ldots & v_k \end{bmatrix} \in \mathbb{R}^{n \times k}$ with orthonormal columns and a $k \times k$ upper Hessenberg matrix T_k whose nonzero entries are the $t_{i,j}$ such that

$$AV_k = V_k T_k + t_{k+1,k} v_{k+1} e_k^T, \tag{49a}$$

$$V_k^T A V_k = T_k. \tag{49b}$$

In floating-point arithmetic, (49a) holds to machine precision, but (49b) does not hold. We slightly depart from standard notation in the literature here and denote the Hessenberg matrix T_k. The reason for this notation is that, in the present context, we apply Algorithm 2 to symmetric A, in which case T_k becomes tridiagonal in exact arithmetic—but not in floating-point arithmetic.

5.2 The Full orthogonalization method (FOM)

Arnoldi-based methods are appropriate to solve a linear system $Ax = b$ where A is square, and not necessarily symmetric. Given an initial guess x_0, we let $r_0 = b - Ax_0$ be the initial residual. It is common to define $\beta = \|r_0\|_2$.

All such methods seek an approximate solution of the form

$$x_k = x_0 + V_k w_k, \tag{50}$$

where V_k is generated by Algorithm 2 with $v_1 = r_0/\beta$, and $w_k \in \mathbb{R}^k$ is chosen to enforce a suitable condition.

We can express the residual $r_k = b - Ax_k$ in terms of w_k as

$$r_k = b - Ax_k = b - A(x_0 + V_k w_k) = r_0 - AV_k w_k = \beta v_1 - AV_k w_k. \tag{51}$$

The full orthogonalization method (FOM) is one such method, and computes an approximate solution (50) by imposing the Galerkin condition on the residual, i.e., $r_k \perp \mathcal{K}_k(A, r_0)$ for $k \geq 1$. In exact arithmetic, w_k is obtained by solving the projected system

$$V_k^T r_k = V_k^T(\beta v_0 - AV_k w_k) = 0 \iff V_k^T A V_k w_k = \beta e_1, \tag{52}$$

where $e_1 = [1, 0, \ldots, 0] \in \mathbb{R}^k$. Algorithm 3 summarizes the procedure.

Our first result is well known in linear algebra circles, but perhaps not so well known in the optimization community.

Algorithm 3 Full Orthogonalization Method (FOM).

1: Choose $x_0 \in \mathbb{R}^n$, compute $r_0 = b - Ax_0$, and set $k \geq 1$.
2: Obtain V_k and T_k from Algorithm 2 initialized with $v_1 = r_0/\beta$.
3: Solve $T_k w_k = \beta e_1$ and set $x_k^{\text{FOM}} = x_0 + V_k w_k$

Proposition 5.1. *Let $A = A^T$, $b \in \mathbb{R}^n$ and $x_0 \in \mathbb{R}^n$. Suppose Algorithm 3 is well defined for values $k = 1, \ldots, j$ in the sense that each T_k is nonsingular. Then, each $x_k^{FOM} = x_k^{PCG}$ with $H_0 = I$ and the same initial x_0.*

Proof. When $A = A^T$, (49b) shows that $T_k = T_k^T$, and hence, is tridiagonal, at least in exact arithmetic. Indeed, in that case, Algorithm 2 coincides with the Lanczos [19] process for symmetric matrices. Moreover, Line 3 of Algorithm 3 yields precisely x_k^{PCG} with $H_0 = I$ [29]. □

A sufficient condition for Proposition 5.1 to hold is that A be PD. Indeed, in that case, (49b) shows that T_k is also PD, and hence nonsingular. In the remainder of this section and the next, any reference to Algorithm 1 implicitly assumes that $H_0 = I$ and that both methods are initialized from the same x_0.

Proposition 5.2. *The residual in Algorithm 3 satisfies*

$$r_0 = \beta v_1, \qquad r_k = -t_{k+1,k}(e_k^T w_k) v_{k+1} \quad (k \geq 1).$$

where e_k denotes the kth canonical basis vector of $\mathbb{R}^k$.

Proof. From (51), (49a) and $T_k w_k = \beta e_1$,

$$r_k = \beta v_1 - \big(V_k T_k + t_{k+1,k} v_{k+1} e_k^T\big) w_k = \beta v_1 - V_k \beta e_1 - t_{k+1,k} v_{k+1} e_k^T w_k.$$

Since $V_k e_1 = v_1$, the first two terms cancel, which establishes the result. □

Consider the LU factorization $T_k = L_k U_k$, where L_k, with entries $\ell_{i,j}$, is unit lower bidiagonal, and U_k, with entries $u_{i,j}$, is upper triangular. To solve $T_k w_k = \beta e_1$, we first solve $L_k z_k = \beta e_1$ followed by $U_k w_k = z_k$. It will be useful to express r_k in terms of L_k, U_k, and z_k.

Corollary 5.1. *For $k \geq 1$,*

$$r_k = -t_{k+1,k} \frac{\zeta_k}{u_{k,k}} v_{k+1} = -t_{k+1,k} \frac{\beta}{u_{k,k}} \Big(\prod_{i=2}^{k} (-\ell_{i,i-1}) \Big) v_{k+1},$$

where ζ_k is the last component of z_k.

Proof. Forward substitution in $L_k z_k = \beta e_1$ yields $\zeta_k = \beta \prod_{i=2}^{k} (-\ell_{i,i-1})$. By Proposition 5.2, we must compute $\omega_k := e_k^T w_k$, i.e., the k-th component of w_k. Back substitution into $U_k w_k = z_k$ gives $\omega_k = \zeta_k / u_{k,k}$. □

Algorithm 3 is akin to quasi-Newton methods in the sense that it typically generates a dense upper Hessenberg matrix T_k, and storage requirements grow quadratically with the number of iterations. Although we may expect that extra off-diagonals in the upper triangle of T_k that appear from loss of orthogonality between the basis vectors v_k in floating-point arithmetic would improve robustness compared to PCG, which simply ignores them, such memory requirements may be prohibitive in practice. In the next section, we review a limited-memory variant of FOM, which is then akin to limited-memory quasi-Newton methods.

In order to establish a connection with PCG, we now introduce the matrix P_k. Under the assumption that T_k is nonsingular, the update of x_k^{FOM} can be written

$$x_k^{\text{FOM}} = x_0 + V_k w_k = x_0 + V_k U_k^{-1} L_k^{-1}(\beta e_1) = x_0 + P_k z_k = x_{k-1}^{\text{FOM}} + \zeta_k p_k, \tag{53}$$

where

$$P_k := V_k U_k^{-1} = \left[p_1^{\text{DIOM}} \cdots p_k^{\text{DIOM}}\right], \quad \text{and} \quad z_k := L_k^{-1}(\beta e_1) = (\zeta_1, \ldots, \zeta_k). \tag{54}$$

Because U_k is upper triangular, the columns of P_k can be determined from the linear system with multiple right-hand sides

$$U_k^T P_k^T = V_k^T \tag{55}$$

as the iterations progress.

5.3 The direct incomplete orthogonalization method (DIOM)

Instead of orthogonalizing each new v_k against all previous ones, DIOM(m) uses a sliding window of size m, reducing storage and computational costs while retaining the most recent information of the Krylov-subspace structure. That is similar to how limited-memory quasi-Newton methods only retain the most recent curvature information. Such strategy is quite different from *restarted* FOM, which is sometimes called FOM(m), and which simply discards all the v_k vectors when it reaches iteration m, only to restart the process from x_m^{FOM} as new initial guess. By Proposition 5.1, only the first m iterations of FOM(m), if they are well defined, coincide with those of CG. Afterwards, the two methods differ. We do not investigate restarted methods further.

In DIOM(m), instead of being upper Hessenberg, T_k becomes a banded upper Hessenberg matrix with upper semi-bandwidth m, and x_k can be updated using an LU factorization that can be updated cheaply from one iteration to the next.

We summarize the resulting iteration as Algorithm 4.

Algorithm 4 DIOM(m)

Choose x_0, set $r_0 = b - Ax_0$, $v_1 \leftarrow r_0/\beta$, $\zeta_1 = \beta$, and $m \geq 1$.
for $k = 1, 2, \ldots$ **do**
  $m_k = \max(1, k - m + 1)$
  $w = Av_k$ — *incomplete orthogonalization process*
  for $i = m_k, \ldots, k$ **do**
    set $t_{i,k} = v_i^T w$ and $w \leftarrow w - t_{ik} v_i$
  $t_{k+1,k} v_{k+1} = w$ — *$t_{k+1,k} > 0$ such that $\|v_{k+1}\| = 1$*
  for $i = m_k, \ldots, k$ **do** — *update LU factorization of T_k*
    if $i = 1$ **then** $u_{1,k} = t_{1,k}$ — *update last column of U_k*
    else $u_{i,k} = t_{i,k} - \ell_{i,i-1} u_{i-1,k}$
  if $u_{k,k} = 0$ **then stop** — *abort: A is singular*
  $p_k^{\text{DIOM}} = \left(v_k - \sum_{i=m_k}^{k-1} u_{i,k} p_i^{\text{DIOM}}\right)/u_{k,k}$ — *last equation of (55)*
  $x_k^{\text{DIOM}} = x_{k-1}^{\text{DIOM}} + \zeta_k p_k^{\text{DIOM}}$ — *update iterate*
  $\ell_{k+1,k} = t_{k+1,k}/u_{k,k}$ — *update the new subdiagonal of L_k*
  $\zeta_{k+1} = -\ell_{k+1,k}\, \zeta_k$ — *last component of z_{k+1}*

Similarly to Proposition 5.1, we have the following equivalence.

Proposition 5.3. *Let $A = A^T$, $b \in \mathbb{R}^n$ and $x_0 \in \mathbb{R}^n$. Suppose Algorithm 4 is well defined for values $k = 1, \ldots, j$ in the sense that each T_k is nonsingular. Then, each $x_k^{DIOM} = x_k^{PCG}$ with $H_0 = I$ and the same initial x_0 for any $m \geq 1$.*

Proof. The proof is similar to that of Proposition 5.1 by noting that T_k is banded upper Hessenberg with upper semi-bandwidth m. Hence, in exact arithmetic, $T_k = T_k^T$ is tridiagonal, and the equivalence with PCG follows the same arguments. □

Because DIOM(m) orthogonalizes the most recent basis vector v_k against the m previous vectors, it can be interpreted as a form of reorthogonalization for PCG, albeit one that follows naturally from Algorithm 2.

If T_k, $k = 1, \ldots, j$ are nonsingular, each $u_{k,k} \neq 0$, and $U_k = D_k \hat{U}_k$, where D_k is diagonal and $\hat{U}_k$ is unit upper triangular. The elements on the diagonal of D_k are $u_{1,1}, \ldots, u_{k,k}$. By (49b) and (54), $P_k^T A P_k = U_k^{-T} L_k = \hat{U}_k^{-T} D_k^{-1} L_k$. If $A = A^T$, it must be that $\hat{U}_k = L_k^T$, and hence $P_k^T A P_k = D_k^{-1}$ is diagonal, which shows that the directions $p_1^{\text{DIOM}}, \ldots, p_k^{\text{DIOM}}$ are conjugate. In particular, if each T_k, $k = 1, \ldots, j$ were PD, as would occur, among other possible situations, if A were itself PD, D_k would also be PD, i.e., each $u_{k,k} > 0$. In the next section, we state properties of Algorithm 4 that hold when such positive curvature is observed.

5.4 DIOM(m) in the presence of positive curvature

Under the assumption that positive curvature is observed over iterations $k = 1, \ldots, j$, we show that p_k^{DIOM} is not necessarily a descent direction for (2), and relate it to d_k^{PCG}.

Lemma 5.1. *For $k = 1, 2, \ldots, j$, let $P_k^T A P_k$ be SPD. Then, $\zeta_k = (-1)^{k-1}|\zeta_k|$, $k = 1, \ldots, j$.*

Proof. For $k = 1$, $\zeta_1 = \beta = (-1)^0 \beta$. Assume that the statement holds for $k-1$, i.e., $\zeta_{k-1} = (-1)^{k-2}|\zeta_{k-1}|$. Then,

$$\zeta_k = -\ell_{k+1,k}\, \zeta_{k-1} = -\ell_{k+1,k}\, (-1)^{k-2} |\zeta_{k-1}| = (-1)^{k-1}\, \ell_{k+1,k} |\zeta_{k-1}|.$$

Because $P_k^T A P_k$ is PD, each $u_{k,k} > 0$, and hence, each $\ell_{k+1,k} \geq 0$. Thus, $\zeta_k = (-1)^{k-1}|\zeta_k|$. □

In Algorithm 4, each $p_k^{\text{DIOM}} \in \mathcal{K}_k(A, v_1)$, and hence, by the Galerkin condition,

$$r_j^T p_k^{\text{DIOM}} = 0 \quad j \geq 1, \quad k = 1, \ldots, j. \tag{56}$$

Theorem 5.1. *For $k = 1, 2, \ldots, j$, let $P_k^T A P_k$ be SPD. Then,*

$$g_0^T p_1^{DIOM} = -\beta / u_{1,1}, \qquad \text{and} \qquad g_k^T p_{k+1}^{DIOM} = g_k^T p_{k+1}^{DIOM} = (-1)^{k-1} \frac{t_{k+1,k}\, |\zeta_k|}{u_{k,k}\, u_{k+1,k+1}} \quad (k \geq 1).$$

Proof. Recall that $g_k = \nabla q(x_k) = Ax_k - b = -r_k$ for all $k \geq 0$. For $k = 0$, $p_1^{\text{DIOM}} = v_1/u_{1,1} = r_0/(\beta u_{1,1}) = -g_0/(\beta u_{1,1})$, and therefore $g_0^T p_1^{\text{DIOM}} = -\beta/u_{1,1}$.

Consider now $k \geq 1$. As in the proof of Corollary 5.1,

$$r_k = -t_{k+1,k} \frac{\zeta_k}{u_{k,k}} v_{k+1}.$$

In view of Lemma 5.1 and (56),

$$g_k^T p_{k+1}^{\text{DIOM}} = -\frac{r_k^T v_{k+1}}{u_{k,k}} = t_{k+1,k} \frac{\zeta_k}{u_{k,k}\, u_{k+1,k+1}} = t_{k+1,k} \frac{(-1)^{k-1}|\zeta_k|}{u_{k,k}\, u_{k+1,k+1}}.$$

□

Theorem 5.1 states that $g_k^T p_{k+1}^{\text{DIOM}}$ is not always positive, which means that p_{k+1}^{DIOM} is not a descent direction for every k. The following theorem recovers the CG direction from DIOM(m) vectors and coefficients.

Theorem 5.2. *For $k = 1, 2, \ldots, j$, let $P_k^T A P_k$ be SPD. Then,*

$$d_k^{PCG} = \zeta_{k+1}\, u_{k+1,k+1}\, p_{k+1}^{DIOM} \quad \text{and} \quad \alpha_k^{PCG} = 1/u_{k+1,k+1}, \quad k = 0, \ldots, j-1. \tag{57}$$

Proof. We know that $V_{k+1}^T A V_{k+1} = T_{k+1} = T_{k+1}^T \in \mathbb{R}^{(k+1)\times(k+1)}$ is tridiagonal. The elements of T_{k+1} can be expressed in terms of the CG coefficients [30, §6.7] as

$$T_{k+1} = \begin{pmatrix} \frac{1}{\alpha_0} & \frac{\sqrt{\beta_0}}{\alpha_0} & & \\ \frac{\sqrt{\beta_0}}{\alpha_0} & \frac{1}{\alpha_1} + \frac{\beta_0}{\alpha_0} & \ddots & \\ & \ddots & \ddots & \frac{\sqrt{\beta_{k-1}}}{\alpha_{k-1}} \\ & & \frac{\sqrt{\beta_{k-1}}}{\alpha_{k-1}} & \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}} \end{pmatrix},$$

from which we dropped the superscripts $^{\text{PCG}}$ for readability.

The LU factorization $T_{k+1} = L_{k+1} U_{k+1}$ with *unit* lower bidiagonal L_{k+1} and upper bidiagonal U_{k+1}. Matching the sub/super-diagonal entries yields

$$\ell_{i+1,i} = \sqrt{\beta_{i-1}}, \qquad u_{i,i+1} = \sqrt{\beta_{i-1}}/\alpha_{i-1}, \qquad i = 1, \ldots, k. \tag{58}$$

The diagonal entries satisfy, $u_{1,1} = 1/\alpha_0$ and

$$u_{j+1,j+1} + \ell_{j+1,j}\, u_{j,j+1} = \frac{1}{\alpha_j} + \frac{\beta_{j-1}}{\alpha_{j-1}}, \qquad j = 1, \ldots, k.$$

Using (58), we obtain $u_{j+1,j+1} = 1/\alpha_j$ for $j = 1, \ldots, k$.

The second part of the proof results from the above by equating the iterate updates in Algorithm 1 and Algorithm 4 by virtue of Proposition 5.3. Specifically, for $k = 0, \ldots, j-1$,

$$x_{k+1}^{\text{PCG}} = x_k^{\text{PCG}} + \alpha_k^{\text{PCG}} d_k^{\text{PCG}} = x_k^{\text{DIOM}} + \zeta_{k+1} p_{k+1}^{\text{DIOM}}.$$

□

In CG, negative curvature can be detected by examining the sign of α_k^{PCG}. Theorem 5.2 thus implies that negative curvature can be detected in DIOM(m) by examining the sign of $u_{k+1,k+1}$. A strategy similar to that of the truncated conjugate gradient of [35] can then be used with LBFGS(m) and DIOM(m) to compute trust-region steps. We examine that strategy in the next section.

6 Truncated methods for nonconvex optimization

We return to the nonlinear, and possibly nonconvex problem (1), and focus on the use of LBFGS(m) and DIOM(m) within a trust-region framework. We wish to determine empirically whether they have advantages over PCG in floating-point arithmetic.

At iteration j of a typical Newton trust-region method for (1), we constuct a quadratic model (2) of the objective about the current iterate z_j where $c = f(z_j)$, $b = -\nabla f(z_j)$, and $A = \nabla^2 f(z_j)$, or possibly an approximation to the Hessian is the latter is not available to too costly to obtain. The quadratic model is then approximately minimized while ensuring that x satisfies the trust-region constraint $\|x\|_2 \leq \Delta_j$ where $\Delta_j > 0$ is the trust-region radius. For reference, we state the subproblem as

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; f(z_j) + \nabla f(z_j)^T x + \tfrac{1}{2} x^T A_j x \quad \text{subject to} \quad \|x\|_2 \leq \Delta_j, \tag{59}$$

where $A_j = A_j^T \approx \nabla^2 f(z_j)$. Once a step x_j has been obtained, the decrease in (2) is compared the actual decrease in f to accept or reject the step, and Δ_j is updated accordingly. Algorithm 5 summarizes the main features of the procedure. We refer the interested reader to the comprehensive treatment [4] for complete details.

Algorithm 5 Newton trust-region method

1: Choose $z_0 \in \mathbb{R}^n$, $\Delta_0 > 0$, $\varepsilon > 0$, $0 < \eta_1 < \eta_2 < 1$.
2: $j = 0$
3: **while** $\|\nabla f(z_j)\| > \varepsilon$ **do**
4: Compute x_j as an approximate minimizer of (59).
5: $\rho = \dfrac{f(z_j) - f(z_j + x_j)}{q(0) - q(x_j)}$
6: **if** $\rho < \eta_1$ **then** $\Delta \leftarrow \Delta/4$ *poor adequacy: reduce the radius*
7: **else if** $\rho \geq \eta_2$ **then** $\Delta \leftarrow 2\Delta$ *very good adequacy: increase the radius*
8: **if** $\rho \geq \eta_1$ **then** $z_{j+1} = z_j + x_j$ *accept step if adequacy is sufficient*
9: **else** $z_{j+1} = z_j$ *reject step if adequacy is insufficient*
10: $j \leftarrow j + 1$
11: **return** z_j

Importantly, on Line 4 of Algorithm 5, x_j need not be an exact solution of (59), but merely induce *sufficient decrease* in the model while remaining in the trust region—see [4] for what sufficient decrease means. For our purposes, suffice it to say that the truncated conjugate gradient method of Steihaug [35] achieves sufficient decrease at the very first iteration, and is one of the most widely-used methods to compute a step in trust-region methods when the trust region is defined in the Euclidean, or in an elliptic, norm.

Because we seek to compute a step from z_j, which lies at the center of the trust-region, it is common to initialize an iterative method from $x_0 = 0$. Steihaug [35] establishes that, as long as positive curvature directions are generated, i.e., $(d_k^{\text{PCG}})^T A_j d_k^{\text{PCG}} > 0$, the model decreases monotonically along the path connecting the iterates $\{x_k\}_{k\geq 0}$, and that the iterates move further away the center of the trust region at each iteration. More precisely, $q_j(x_{k+1}) \leq q(x_k + \tau d_k^{\text{PCG}})$ for all $\tau \in [0, 1]$ and $\|x_{k+1}\|_2 \geq \|x_k\|_2$ provided that $x_0 = 0$. Thus as long as positive curvature directions are generated, either we find a minimum of q inside the trust region in at most n iterations, or there is an iteration k such that $\|x_k\|_2 < \Delta_j \leq \|x_{k+1}\|_2$. Because of the monotonicity of q, we can simply stop at the point where the segment connecting x_k to x_{k+1} meets the trust-region boundary. If nonpositive curvature should be detected at iteration k, i.e., $(d_k^{\text{PCG}})^T A_j d_k^{\text{PCG}} \leq 0$, we can see that q continues to decrease along d_k^{PCG}, and we can safely follow that direction until we hit the trust-region boundary.

By the results of the previous sections, those properties continue to hold for all methods of the Broyden class, to LBFGS(m), and to DIOM(m) applied to (59). In PCG and LBFGS(m), nonpositive curvature is detected by checking the sign of α_k, and in DIOM(m), that of $u_{k+1,k+1}$. Algorithm 6 provides the procedure for LBFGS(m), and Algorithm 7 for DIOM(m).

Algorithm 6 LBFGS(m) for trust-region subproblems

1: Set $A \leftarrow \nabla^2 f(z_j)$, $b \leftarrow -\nabla f(z_j)$, $\Delta_j > 0$, $H_0 \leftarrow I$, $x_0 \leftarrow 0$, $g_0 \leftarrow -b$
2: **for** $k = 0, 1, \ldots$ **do**
3: $d_k \leftarrow -H_k g_k$
4: Compute $\tau > 0$ such that $\|x_k + \tau s_k\|_2 = \Delta_j$ *step size to the boundary*
5: $\alpha_k = -g_k^T d_k / (d_k^T A d_k)$
6: **if** $d_k^T A d_k \leq 0$ or $\alpha_k > \tau$ **then return** $x_k + \tau d_k$ *step to boundary*
7: $s_k \leftarrow \alpha_k d_k$
8: $x_{k+1} = x_k + s_k$
9: $y_k \leftarrow \alpha_k A d_k$ $y_k = g_{k+1} - g_k = A s_k$
10: $g_{k+1} = g_k + y_k$
11: $H_{k+1} \leftarrow \text{update}(s_k, y_k, H_k)$

Algorithm 7 DIOM(m) for trust-region subproblems

Set $A \leftarrow \nabla^2 f(z_j)$, $b \leftarrow -\nabla f(z_j)$, $\Delta_j > 0$, $x_0^{\text{DIOM}} \leftarrow 0$, $v_1 \leftarrow b/\|b\|$
for $k = 1, 2, \ldots$ **do**
  $m_k = \max(1, k - m + 1)$
  Perform Lines 4–10 of Algorithm 4
  $d_k^{\text{DIOM}} = \zeta_k \Big(v_k - \sum_{i=m_k}^{k-1} u_{i,k} p_i^{\text{DIOM}} \Big)$ *avoid risking a division by zero*
  Compute $\tau > 0$ such that $\|x_{k-1}^{\text{DIOM}} + \tau d_k^{\text{DIOM}}\|_2 = \Delta_j$.
  if $u_{k,k} \leq 0$ or $1/u_{k,k} > \tau$ **then return** $x_{k-1}^{\text{DIOM}} + \tau d_k^{\text{DIOM}}$ *step to boundary*
  $x_k^{\text{DIOM}} = x_{k-1}^{\text{DIOM}} + d_k^{\text{DIOM}}/u_{k,k}$
  $\ell_{k+1,k} = t_{k+1,k}/u_{k,k}$
  $\zeta_{k+1} = -\ell_{k+1,k}\, \zeta_k$

We summarize the per-iteration computational requirements of the three methods in Table 1. PCG stores x_k, g_k, and d_k^{PCG}. LBFGS(m) stores x_k, g_k, and the m most recent pairs $\{s_i, y_i\}$. For LSR1(m), the update formula is

$$H_{k+1}^m = H_k^0 + \sum_{i=k-m+1}^{k} \frac{z_i z_i^T}{s_i^T z_i} \quad \text{with} \quad z_i = s_i - H_i^m y_i,$$

so that LSR1(m) stores x_k and g_k, together with the m most recent vectors z_i. DIOM(m) stores $\{x_k\}$, m recent Krylov basis vectors v_i, and m update vectors p_i^{DIOM}. We do not count storage for scalars. As expected, PCG has the smallest memory footprint because conjugacy is maintained through short recurrences, at least in exact arithmetic. Each method performs one operator-vector product with A per iteration. For LBFGS(m), the additional work arises from Strang's two-loop recursion used to apply the inverse-Hessian approximation to a vector, assuming $H_0 = I$ [25], and from computing $s_k^T y_k$. For LSR1(m), each vector z_i requires an application of the previous limited-memory operator H_i^m to y_i. This product is computed at the time of insertion of z_i into memory. Thus, at iteration k, only the new product $H_k^m y_k$ is required to form z_k. This product requires applying the m stored z_i to y_k, at a cost of $2mn$ scalar multiplications, and forming $s_k^T z_k$, which requires n scalar multiplications and additions. Finally, applying H_{k+1}^m to any vector v requires $2mn$ scalar multiplications, including those needed to compute $z_i^T v$ and $(z_i^T v) z_i$. The additional work per iteration is therefore $4mn + n$ scalar multiplications. For DIOM(m), they arise from the incomplete orthogonalization process on Lines 4–7, the LU factorization update on Lines 8–10 and 14–15, and the update of p_k^{DIOM} on Line 12 of Algorithm 4.

Table 1: Per-iteration complexity comparison for CG, LBFGS(m), and DIOM(m).

Method	Stored vectors	Additional work per iteration
CG	3	
LBFGS(m)	$2 + 2m$	$4mn + n$ scalar multiplications
LSR1(m)	$2 + m$	$4mn + n$ scalar multiplications
DIOM(m)	$1 + 2m$	$3mn + m$ scalar multiplications

7 Numerical experiments

In this section, we first illustrate the practical behavior of PCG, LBFGS(m), LSR1(m) and DIOM(m) on positive-definite linear systems to assess their convergence behavior in finite precision. The PCG and DIOM(m) implementations are those of [23], while the LBFGS(m) and LSR1(m) implementations are available at https://github.com/oihanc/Krylov.jl/tree/lsr1; the corresponding LBFGS and LSR1 operators are implemented in [28]. The DIOM(m) implementation was modified to include the detection and treatment of nonpositive curvature and trust-region constraint.

Next, we report results on two nonlinear problems: data assimilation and classification, using the numbers of objective evaluations, gradient evaluations, and Hessian-vector products as performance measures. We use the trust-region framework implemented in the `trunk` solver of [22].

7.1 Behavior on Positive-Definite Systems

In this section, we examine the behavior of CG, LBFGS, and DIOM for solving linear systems with positive-definite matrices with $m = 50$ and $m = n$. Table 2 summarizes the test matrices used. In all experiments, we set $x_0 = 0$ and $b = 100[1, \ldots, 1]^T$. Figure 2 shows plots of the relative residual and the quadratic objective value. We use 64-, 128-, and 256-bit floating-point arithmetic.

Table 2: Positive-definite test matrices used in the experiments.

	gr_30_30	nos5	494_bus
Dimension n	900	468	494
Conditioning $\kappa_2(A)$	1.945e+02	1.100e+04	2.415e+06

These experiments highlight differences in the numerical behavior of the methods in finite precision. In particular, LBFGS and DIOM tend to exhibit more robust convergence behavior than CG on ill-conditioned systems provided m is sufficiently large. In the plots, LBFGS(m) and and DIOM(m) are visually superposed and perform nearly identically. Only on the well-conditioned system `gr_30_30` are all the curves superposed. On the intermediate system `nos5`, we only see an advantage with DIOM and LBFGS for $m = n$. However, even as floating-point accuracy increases, PCG trails behind on the worst-conditioned system `494_bus`, even though its condition number can be considered moderate. This suggests that memory can be an effective mechanism for mitigating the effects of loss of orthogonality on ill-conditioned problems.

The results in Figure 3, where we plot the same quantities as in Figure 2 for 64-bit floating-point arithmetic, indicate that full-memory LSR1($m = n$) performs well across the different test matrices and is comparable to LBFGS and DIOM with $m = n$. However, for $m = 25, 50, 100, 200$, LSR1 deteriorates once the iteration count exceeds m: even on the well-conditioned system `gr_30_30`, with $m = 25$, a clear discrepancy between CG and LSR1 appears after about 25 iterations. The same phenomenon occurs for other memory sizes. This suggests that LSR1 may not generate directions parallel to those of CG, a point that deserves further theoretical investigation.

7.2 A data-assimilation problem

We evaluate the trust-region framework with CG, LBFGS, and DIOM on a nonlinear weighted least-squares problem arising in data assimilation:

$$\underset{z_0 \in \mathbb{R}^n}{\text{minimize}}\; f(z_0) = \tfrac{1}{2}\|z_0 - z_b\|^2_{B^{-1}} + \tfrac{1}{2}\sum_{i=1}^{N_t} \|y_i - \mathcal{H}_i(\mathcal{M}_{t_0,t_i}(z_0))\|^2_{R_i^{-1}}. \tag{60}$$

Here, $z_0 = z(t_0)$ denotes the state at the initial time t_0, $z_b \in \mathbb{R}^n$ is the background (prior) estimate, and $y_i \in \mathbb{R}^{m_i}$ denotes the observation vector at time t_i, for $i = 1, \ldots, N_t$. The nonlinear model $\mathcal{M}_{t_0,t_i}$ propagates z_0 from t_0 to t_i, while $\mathcal{H}_i$ maps the propagated state into the observation space. The SPD covariance matrices $B \in \mathbb{R}^{n\times n}$ and $R_i \in \mathbb{R}^{m_i \times m_i}$ are associated with the background and observation errors, respectively.

In our numerical experiments, we use the Lorenz-96 [21] model as the physical dynamical system $\mathcal{M}_{t_0,t_i}$—a common reference model in data assimilation. The observation operator $\mathcal{H}$ is a uniform selection operator, i.e, $\mathcal{H}(x)$ extracts a subset of x that is uniformly selected. We consider $B = \sigma_b I_n$ with $\sigma_b = 0.8$, $R_1 = R_2 = \sigma_r^2 I_m$ with $\sigma_r = 0.2$. We choose $n = 10,000$, $N_t = 2$, and $m_1 = m_2 = 500$.

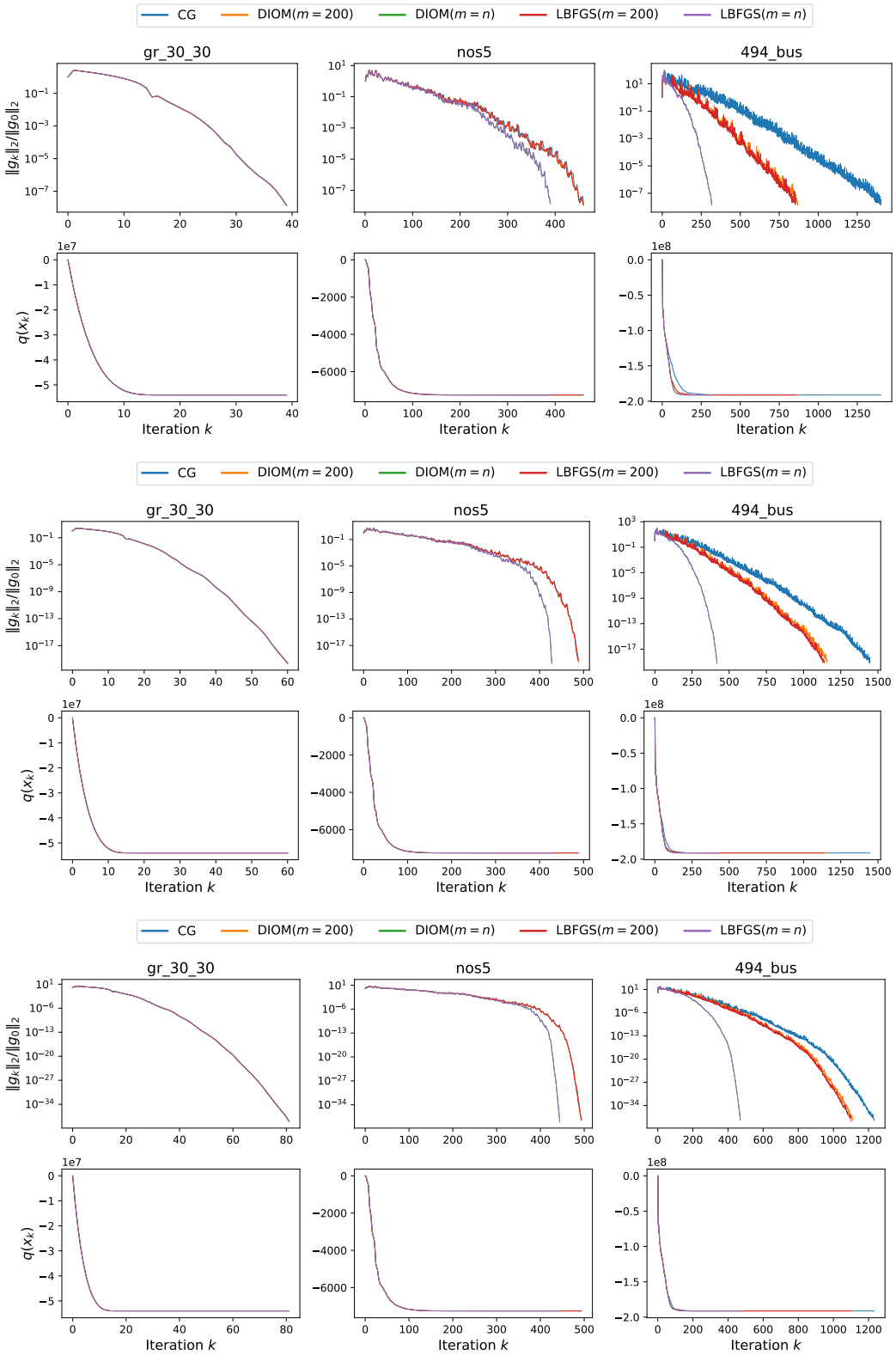

Figure 2: Convergence curves on the test systems of Table 2 in 64-bit (top), 128-bit (middle), and 256-bit (bottom) floating-point arithmetic.

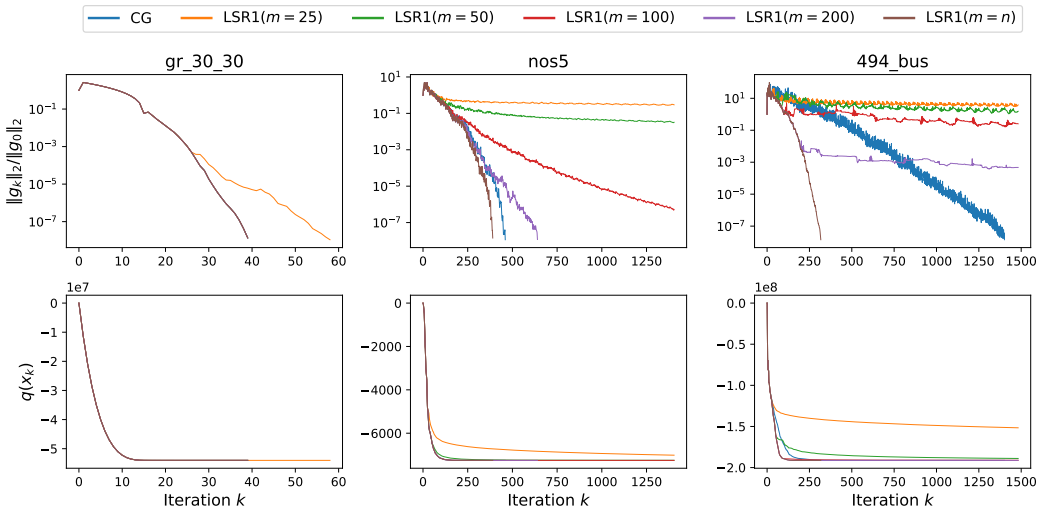

Figure 3: Convergence curves for LSR1(m) and CG on the test systems of Table 2 in 64-bit floating-point arithmetic.

Table 3: Statistics for the data assimilation problem (60). Best values are boldfaced.

	m	obj_eval	grad_eval	hprod_eval	time (s)
CG		333	**61**	1083	151.8
LBFGS	50	**293**	63	**900**	**116.7**
DIOM	50	334	68	1033	128.3
LBFGS	100	**293**	**61**	951	128.7
DIOM	100	337	64	1005	132.9

Table 3 summarizes the number of objective, gradient, and Hessian-vector evaluations, reported as obj_eval, grad_eval, and hprod_eval, respectively.

The results indicate that LBFGS is the most effective subsolver for (60), and $m = 50$ performs best, requiring the fewest objective evaluations and Hessian-vector products while also achieving the shortest runtime. DIOM also reduces the number of Hessian-vector products relative to CG and yields a shorter runtime.

7.3 A binary classification problem

We benchmark the LBFGS and DIOM subsolvers within the trust-region framework on a binary classification problem derived from the MNIST dataset. The goal is to classify images of the handwritten digits 1 and 7. Let $A \in \mathbb{R}^{n\times N}$ denote the data matrix, whose columns are vectorized images, and $b \in \{-1, 1\}^N$ denote the corresponding label vector, where $b_i = 1$ for digit 1 and $b_i = -1$ for digit 7. Here, n is the number of pixels per image and N is the number of samples. The problem is formulated as

$$\underset{z\in\mathbb{R}^n}{\text{minimize}} \;\; f(z) = \tfrac{1}{2}\|\mathbf{1} - \tanh(b \odot (A^T z))\|^2, \tag{61}$$

where $\odot$ is the elementwise product, and $\mathbf{1}$ is the vector of ones. This formulation encourages $b_i(A_i^T z)$ to be positive when sample i is correctly classified. We have $N = 13,007$, and the images have resolution 28×28, so $n = 784$. Table 4 reports our results. Overall, DIOM($m = 50$) is the fastest, while requiring the fewest Hessian-vector products. The number of objective and gradient evaluations are identical across all methods, suggesting that DIOM solves the quadratic subproblems more efficiently.

Table 4: Statistics for the data assimilation problem (61). Best values are boldfaced.

	m	obj_eval	grad_eval	hprod_eval	time (s)
CG		48	26	386	9.69687
LBFGS	50	48	26	357	8.46662
DIOM	50	48	26	**336**	**8.46505**
LBFGS	100	48	26	357	9.36065
DIOM	100	48	26	**336**	9.47864

8 Discussion

While it is well known that partial reorthogonalization in PCG improves its behavior on ill-conditioned systems [18], it is interesting that methods that are much closer to optimization, such as LBFGS, also coincide with PCG in exact arithmetic and provide a form of stabilization by way of increased memory. DIOM can be viewed as indiscriminate partial reorthogonalization, in the sense that it always occurs, and follows naturally from the Arnoldi process. It coincides with CG at one end of the spectrum, $m = 1$, and with FOM at the other, $m = \infty$. In exact arithmetic, LBFGS and DIOM coincide with CG for any value of $m \geq 1$. The present research and the work of [11] suggest investigating other limited-memory methods of the Broyden class to solve quadratic problems, e.g., in trust-region and line search methods.

Dahito and Orban [5] show that the conjugate residuals method [17], and hence MINRES [29], are appropriate to solve trust-region subproblems, can detect nonpositive curvature, and sometimes outperform CG. MINRES is based on the Lanczos process, and has a counterpart based on the Arnoldi process: GMRES [31]. GMRES admits a limited-memory variant DQGMRES [32]. We may expect that DQGMRES will provide added robustness by way of memory and reorthogonalization, and we intend to study its properties in future research on ill-conditioned nonlinear problems.

The numerical behavior of LSR1 is more delicate. In our experiments, full-memory LSR1 is competitive with the other full-memory methods, but the loss of old curvature pairs appears to have a structural effect: once information older than the memory window is discarded, the method may no longer produce directions parallel to those of CG, even on well-conditioned problems. Understanding whether this deterioration can be explained by the indefiniteness of the LSR1 inverse approximations, by a vanishing LSR1 denominator, or by a loss of conjugacy is an interesting question.

References

[1] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. Q. Appl. Math., 9:17–29, 1951.

[2] C. G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. IMA J. Appl. Math., 6(1):76–90, 03 1970.

[3] C. G. Broyden. The convergence of a class of double-rank minimization algorithms: 2. the new algorithm. IMA J. Appl. Math., 6(3):222–231, 1970.

[4] A. R. Conn, N. I. M. Gould, and P. L. Toint. Trust-Region Methods, volume 1 of MPS/SIAM Series on Optimization. SIAM, Philadelphia, 2000.

[5] M.-A. Dahito and D. Orban. The conjugate residual method in linesearch and trust-region methods. SIAM J. Optim., 29(3):1988–2025, 2019.

[6] W. C. Davidon. Variable metric method for minimization. SIAM J. Optim., 1(1):1–17, 1991.

[7] R. S. Dembo and T. Steihaug. Truncated-Newton algorithms for large-scale unconstrained optimization. Math. Program., 26(2):190–212, 1983.

[8] J. E. Dennis and J. J. Moré. A characterization of superlinear convergence and its application to quasi-newton methods. Math. Comp., 28(126):549–560, 1974.

[9] J. E. Dennis and J. J. Moré. Quasi-Newton methods, motivation and theory. SIAM Rev., 19:46–89, 1977.

[10] L. Dixon. Quasi-newton algorithms generate identical points. Math. Program., 2(1):383, 1972.

[11] D. Ek and A. Forsgren. Exact linesearch limited-memory quasi-Newton methods for minimizing a quadratic function. Comput. Optim. Appl., 79(3):789–816, 2021.

[12] R. Fletcher. A new approach to variable metric algorithms. The computer journal, 13(3):317–322, 1970.

[13] R. Fletcher and M. J. Powell. A rapidly convergent descent method for minimization. The computer journal, 6(2):163–168, 1963.

[14] A. Forsgren and T. Odland. On exact linesearch quasi-newton methods for minimizing a quadratic function. Comput. Optim. Appl., 69(1):225–241, Jan 2018.

[15] D. Goldfarb. A family of variable-metric methods derived by variational means. Math. Comp., 24(109): 23–26, 1970.

[16] G. H. Golub and C. F. Van Loan. Matrix Computations. The Johns Hopkins University Press, Baltimore, 4th edition, 2013.

[17] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand., 49(6):565–583, 1952.

[18] D. S. Horst. The Lanczos algorithm with partial reorthogonalization. Math. Comp., 42:115–142, 1984.

[19] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Natl. Bur. Stand., 45:225–280, 1950.

[20] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. Math. Program., 45:503–528, 1989.

[21] E. N. Lorenz. Predictability: A problem partly solved. In Proc. Seminar on predictability, volume 1. Reading, 1996.

[22] T. Migot, D. Monnet, D. Orban, and A. S. Siqueira. JSOSolvers.jl: Unconstrained and bound-constrained optimization solvers. J. Open Source Softw., 11(117):9467, 2026.

[23] A. Montoison and D. Orban. Krylov.jl: A Julia basket of hand-picked Krylov methods. J. Open Source Softw., 8(89):5187, 2023.

[24] L. Nazareth. A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. SIAM J. Numer. Anal., 16(5):794–800, 1979.

[25] J. Nocedal. Updating quasi-Newton matrices with limited storage. Math. Comp., 35:773–782, 1980.

[26] J. Nocedal and S. J. Wright. Numerical Optimization. Springer, New York, 2nd edition, 2006.

[27] D. Orban and M. Arioli. Iterative Solution of Symmetric Quasi-Definite Linear Systems, volume 3 of Spotlights. SIAM, Philadelphia, USA, 2017.

[28] D. Orban and A. Soares Siqueira. Linearoperators. jl. Zenodo, 2019.

[29] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. SIAM J. Numer. Anal., 12(4):617–629, 1975.

[30] Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, second edition, 2003.

[31] Y. Saad and M. Schultz. GMRES, a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. and Statist. Comput., 7:856–869, 1986.

[32] Y. Saad and K. Wu. DQGMRES: a direct quasi-minimal residual algorithm based on incomplete orthogonalization. Numer. Linear Algebra Appl., 3(4):329–343, 1996.

[33] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. Math. Comp., 24(111): 647–656, 1970.

[34] D. F. Shanno. Conjugate gradient methods with inexact searches. Math. Oper. Res., 3(3):244–256, 1978.

[35] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. SIAM J. Numer. Anal., 20(3):626–637, 1983.

[36] R. J. Vanderbei. Symmetric quasi-definite matrices. SIAM J. Optim., 5(1):100–113, 1995.