Cooperative concurrent asynchronous computation of the solution of symmetric linear systems

Amit Bhaya, Pierre-Alexandre Bliman, Fernando Pazos

Noname manuscript No.
(will be inserted by the editor)

Cooperative concurrent asynchronous computation of the solution of symmetric linear systems

Amit Bhaya · Pierre-Alexandre Bliman · Fernando Pazos

Received: date / Accepted: date

Abstract This paper extends the idea of Brezinski's hybrid acceleration procedure, for the solution of a system of linear equations with a symmetric coefficient matrix of dimension n, to a new context called cooperative computation, involving m agents ($m \ll n$), each one concurrently computing the solution of the whole system, using an iterative method. Cooperation occurs between the agents through the communication, periodic or probabilistic, of the estimate of each agent to one randomly chosen agent, thus characterizing the computation as concurrent and asynchronous. Every time one agent receives solution estimates from the others, it carries out a least squares computation, involving a small linear system of dimension m, in order to replace its current solution estimate by an affine combination of the received estimates, and the algorithm continues until a stopping criterion is met. In addition, an autocooperative algorithm, in which estimates are updated using affine combinations of current and past estimates, is also proposed. The proposed algorithms are shown to be efficient for certain matrices, specifically in relation to the popular Barzilai–Borwein algorithm, through numerical experiments.

Keywords Distributed multi-agent optimization · Gradient descent methods · Hybrid procedures

1 Introduction

Motivated by the ubiquitous appearance of large scale networks, there has been much recent interest in distributed control and computation involving multiple agents connected through a network. In the class of problems studied in this context, there has been a recent focus on so-called cooperative distributed multi-agent optimization, in which the goal is to collectively optimize a global objective, under the constraints that there is no access to centralized information, so that the algorithms designed for such networks should rely only on local information. In addition, given the applications, such as wireless networks, there is usually also a requirement of robustness against changes in network connectivity (link failures) and structure (node failures). More formally, following [22], a multi-agent network model consists of m agents exchanging information over a connected network. Each agent i has a local convex scalar objective function $f_i(\mathbf{x})$, with $f_i : \mathbb{R}^n \to \mathbb{R}$, and a nonempty local convex constraint set, known by this agent only. The vector $\mathbf{x} \in \mathbb{R}^n$ represents a global decision vector that the agents are collectively trying to decide on. The goal of the agents is to cooperatively optimize a global objective function, denoted by $f(\mathbf{x})$, which is a combination of the local objective functions, i.e., $f(\mathbf{x}) = T(f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$,

This work was supported in part by FAPERJ (CNE grant to the first author), and CNPq (BPP and EU grant to first author, as well as a PNPD fellowship to the third author). The second author's work was supported by Inria. A preliminary version of this paper appeared in the IEEE Conference on Decision and Control, December 2010, Atlanta, USA [7].

Bhaya, Pazos
Department of Electrical Engineering, Federal University of Rio de Janeiro, PEE/COPPE/UFRJ, PO Box 68504, Rio de Janeiro, RJ 21945-970, Brazil
Tel.: +55-21-2562 8078
Fax: +55-21-2562 8080
E-mail: amit@nacad.ufrj.br, quini.coppe@gmail.com

Bliman
Institut National de Recherche en Informatique et en Automatique (Inria), Domaine de Voluceau, Rocquencourt BP 105, 78153 Le Chesnay Cedex, France E-mail: pierre-alexandre.bliman@inria.fr

where a common choice of T is $\sum_i f_i(\mathbf{x})$. Roughly speaking, these two paradigms of cooperative computation are concerned with, respectively, the solution of problems of decentralized optimization of the type $\min_{x_i} \sum_i f_i(\mathbf{x})$, or of 'worst case' optimization of the type $\max_i \min_{x_i} f_i(x_i)$.

In the field of computation, the emphasis has been mainly on the paradigm of concurrent computing in which some computational task is subdivided into as many subtasks as there are available agents (processors). The subdivision naturally induces a communication structure or graph connecting processors and the challenge is to achieve a subdivision that maximizes concurrency of tasks (hence minimizing total computational time), while simultaneously minimizing communication overhead. This paradigm arose as a consequence of the usual architecture of most early multiprocessor machines, in which interprocessor communication is a much slower operation than a mathematical operation carried out in the same processor. Disadvantages of this approach arise from the difficulty of effectively decomposing a large task into minimally connected subtasks, difficulties of analysis and the need for synchronization barriers at which all processors wait for the slowest one, in order to exchange information with the correct time stamps (i.e., without asymmetric delays).

With the advent of ever larger on-chip memory and multicore processors that allow multithread programming, it is now possible to propose a new paradigm in which each thread, with access to a common memory computes its own estimate of the solution to the whole problem (i.e., decomposition of the problem into subproblems is avoided) and the threads exchange information amongst themselves, this being the cooperative step. The design of a cooperative algorithm has the objective of ensuring that exchanged information is used by the threads in such a way as to reduce overall convergence time.

The idea of information exchange between two iterative processes was introduced into numerical linear algebra long before the advent of multicore processors by Brezinski [16] under the name of *hybrid procedures*, defined as (we quote) "a combination of two arbitrary approximate solutions with coefficients summing up to one...(so that) the combination only depends on one parameter whose value is chosen in order to minimize the Euclidean norm of the residual vector obtained by the hybrid procedure... The two approximate solutions which are combined in a hybrid procedure are usually obtained by two iterative methods". The objective of minimizing the residue is to accelerate convergence of the overall hybrid procedure.

Brezinksi and coauthors followed up on this idea in several papers. In [2] properties of convergence of the hybrid procedure are studied ([2, sec. 3]). Hybrid procedures are also used by Brezinski in [12] and [14]. The concept of multiparameter iteration is also introduced by Brezinski in [15] and [11], where a descent matrix is used instead of a descent direction. In [13] Brezinski considers the equation $\mathbf{AX} = \mathbf{B} \in \mathbb{R}^{n\times m}$ and defines and studies block iterative methods which handle the linear system, with m different right hand sides, simultaneously instead of treating them separately. The same paper also considers a block hybrid procedure to find the solution of the block system ([13, sec. 3]). It should be noted, however, that all of these papers consider the hybrid procedure in a convergence acceleration context rather than in the context of concurrent, asynchronous implementation.

Thus, the contribution of this paper is to reconsider the idea of a hybrid procedure in the context of concurrent, asynchronous implementation, in which each iterative method is employed by an agent which cooperates, under some communication scheme, with the other agents in order to compute the desired solution. The basic idea can be expressed simply as follows. For linear equations, affine combinations of solution vectors ($\mathbf{x}_i$) correspond to affine combinations of residues ($\mathbf{r}_i := \mathbf{A}\mathbf{x}_i - \mathbf{b}$). Thus, thinking in terms of several agents cooperating to solve the linear system, whenever any agent is in possession of residues from some other (possibly neighboring) agents, it can compute a residue with norm less than or equal to its own, by simply computing the vector of smallest norm in the affine subspace spanned by the residue vectors available to it. This is a simple minimization problem which has an explicit closed form solution (of the least squares type) not just for the 2-norm, but also for weighted 2-norms, discussed further below. Thus the key step in the cooperative algorithm proposed in this paper is as follows. Every time one agent receives approximations of solutions from one or more agents, the receiving agent applies the closed form solution to the minimization problem described above and replaces its current value by the computed affine combination. Computation continues until a stopping criterion, e.g., residual norm below a prespecified tolerance, is met. This paper studies deterministic and probabilistic communication variants and shows, through numerical experiments, that they can be quite efficient, specifically in relation to the well known and much used Barzilai–Borwein algorithm ([5]), particularly for ill-conditioned matrices. Notice that the cooperative paradigm, which seeks to solve $\min_i \min_{x_i} f_i(x_i)$, is quite different from the distributed optimization problems mentioned above.

Other motivations for the consideration of the paradigm of cooperative computation are as follows. For the simple model problem of solution of linear systems, there is the well known optimal conjugate gradient (CG) algorithm, which, however, is inherently sequential. It is also a second order method, in the sense that, for a linear system of dimension n, it uses two coupled iterations, one in the direction of the gradient of the scalar

quadratic function $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} - \mathbf{b}^\mathsf{T}\mathbf{x}$ and the other in a suitably defined conjugate direction. In the class of first order methods that use only gradient information, there is the well known steepest descent (SD) algorithm, the related orthomin (OM) algorithm, and the Barzilai–Borwein (BB) algorithm, which uses delayed gradient information and has been extensively investigated due to its surprisingly good convergence properties. These first order methods are also all essentially sequential and do not admit concurrent implementations. In contrast, the cooperative computation (CC) algorithm studied in this paper, is inherently concurrent, with asynchronous communication between agents. On the other hand, for two agents, it also works with two vectors of size n, like the CG algorithm, but with simultaneous updates by each processor instead of the sequential updates required by the CG algorithm.

Many attempts to improve the SD, OM, BB methods have been made (see e.g. [4,19,18]). An intuitively attractive way to achieve this is by including more 'knowledge' about the behavior of the $\mathbf{A}$ matrix acquired during the previous iterations, especially values of the previous residues. Here our approach is different, as it is based on exchanging information between *different* trajectories of essentially the same dynamical system. Also note that the cooperative concurrent approach proposed in this paper is different from the standard parallel computation approach, in which a (large) computational task is typically decomposed into smaller computation tasks that are then carried out concurrently and, in a synchronized step, put together to generate the next iterate.

This paper is organized as follows. Section 2 discusses some preliminaries, while section 3 studies optimal affine combinations of residual vectors. Section 4 presents the general form of the cooperative computation algorithm and section 5 the results of numerical experiments. Section 6 presents some concluding remarks. Finally, two appendices end the paper with studies about the performance of the CC algorithms and with several properties and lemmas, respectively.

2 Preliminaries

One approach to solving the equation

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

where $\mathbf{A}$ is a symmetric positive definite matrix of dimension n (typically large and sparse), is to minimize the convex quadratic function, also referred to as the *cost function* and defined as:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} - \mathbf{b}^\mathsf{T}\mathbf{x} \tag{2}$$

since the unique optimal point that minimizes f is $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$. Several algorithms are based on the standard idea of generating a sequence of points, starting from an arbitrary initial guess, and proceeding in the descent direction (negative gradient of $f(\mathbf{x})$), with an adequate choice of the step size. In mathematical terms:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \rho_k \mathbf{r}_k \tag{3}$$

where $\mathbf{r}_k = \nabla f(\mathbf{x}_k) = \mathbf{A}\mathbf{x}_k - \mathbf{b}$ is called the residue and ρ_k is the step size. Note that, from (3):

$$\mathbf{r}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k - \rho_k \mathbf{r}_k) - \mathbf{b} = (\mathbf{I} - \rho_k \mathbf{A})\mathbf{r}_k \tag{4}$$

Table I lists some of the common choices for the step size ρ_k. The notation $\|\mathbf{r}\|_{\mathbf{A}^{-1}}$ refers to the so-called $\mathbf{A}^{-1}$-norm, defined as $\sqrt{\mathbf{r}^\mathsf{T}\mathbf{A}^{-1}\mathbf{r}}$. Note that $\|\mathbf{r}\|^2_{\mathbf{A}^{-1}} = 2f(\mathbf{x}) + \mathbf{b}^\mathsf{T}\mathbf{x}^*$. These step sizes, being inverses of Rayleigh quotients, are all between the values: $0 < (1/\lambda_{\max}(\mathbf{A})) \leq \rho_k \leq (1/\lambda_{\min}(\mathbf{A}))$. Some other, more elaborate, choices and the comparison of these choices with each other and with the Barzilai–Borwein algorithm are given in [6].

Table 1 Choice of step size for different methods: steepest descent (SD), Orthomin (OM), Barzilai–Borwein (BB)

Method	Stepsize choice ρ_k	Convergence
SD	$\frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k}$	monotonic in $\|\mathbf{r}\|_{\mathbf{A}^{-1}}$
OM	$\frac{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\mathsf{T}\mathbf{A}^2\mathbf{r}_k}$	monotonic in $\|\mathbf{r}\|_2$ and $\|\mathbf{r}\|_{\mathbf{A}^{-1}}$
BB	$\frac{\mathbf{r}_{k-1}^\mathsf{T}\mathbf{r}_{k-1}}{\mathbf{r}_{k-1}^\mathsf{T}\mathbf{A}\mathbf{r}_{k-1}}$	nonmonotonic

The SD and OM algorithms are classical ([21]), and, although simple, have the disadvantage of slow convergence when the condition number of $\mathbf{A}$ is large ([3,23]). The BB algorithm has been much studied because of its remarkable improvement over the SD and OM algorithms ([25,4,17] and references therein) and proofs of its convergence can be found in [20]. However, a complete explanation of why this simple modification of the SD algorithm improves its performance considerably has not yet been found, although it has been suggested that the improvement is connected to its nonmonotonic convergence, as well as to the fact that it does not produce iterates that get trapped in a low dimensional subspace [4]. Like the SD and CG methods, the BB method tends to become slow for ill-conditioned problems [20,17]. The other method that can be derived from an optimization perspective is the famous conjugate gradient (CG) method, which can be written as:

$$\begin{aligned} \mathbf{r}_{k+1} &= \mathbf{r}_k - \rho_k A\mathbf{p}_k \\ \mathbf{p}_{k+1} &= \mathbf{r}_{k+1} - \beta_k \mathbf{p}_k \end{aligned} \tag{5}$$

This method has the properties of very fast convergence, theoretically in at most n iterations, where n is the problem dimension ([21]), superior to that of the BB algorithm ([6]). It is also appropriate for use with large, sparse matrices, since it uses only matrix-vector products in its computations. Notice, however, that the CG algorithm works with a state space of dimension twice that of the vector $\mathbf{x}$, since it uses coupled iterations in both $\mathbf{r}$ and $\mathbf{p}$. This is interpreted from a control perspective in [9,10]. Note also that the coupling between the $\mathbf{r}$ and $\mathbf{p}$ iterations (5) is of the Gauss-Seidel type, i.e, the updated value $\mathbf{r}_{k+1}$ is used *immediately after* it is computed in the $\mathbf{p}$ update. This is crucial to the speed of convergence of the CG algorithm and also implies that it has an inherently sequential structure, that does not permit concurrent computation of $\mathbf{r}$ and $\mathbf{p}$.

In this context, the affine combination cooperative method studied in this paper aims to combine the advantages of the methods mentioned above, while attempting to avoid some of the disadvantages. Specifically, by using affinely combined SD/OM algorithms, the monotonically decreasing behavior of some norm of the residual vector is retained, but speed of convergence can be enhanced. The algorithm runs concurrently on all agents, with a low level of communication between them, that can be realized, for example, asynchronously at random intervals, between random pairs of agents.

3 Optimal affine combinations of solution estimates

Let $\mathbf{x}_1, \ldots, \mathbf{x}_m$ be m estimates of the solution $\mathbf{x}^*$ to the linear system (1), available at some stage of computation of the solution by m agents using iterative methods, which could be the same or different. The question that we wish to address is the following. If communication between agents occurs, can each agent, receiving one or more solution estimates, improve its own estimate using the information it receives? The answer is affirmative, and is explained below.

Given the solution estimate $\mathbf{x}_i$, let $\mathbf{r}_i = \mathbf{A}\mathbf{x}_i - \mathbf{b}$ be the corresponding residual vector and $\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}^*$ the corresponding error vector, for $i \in \{1, \cdots, m\}$.

For future use, we put the m estimate, residual and error vectors into matrices in $\mathbb{R}^{n\times m}$ defined as:

$$\mathbf{X} := [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_m] \tag{6}$$

$$\mathbf{R} := [\mathbf{r}_1\ \mathbf{r}_2\ \cdots\ \mathbf{r}_m] \tag{7}$$

$$\mathbf{E} := [\mathbf{e}_1\ \mathbf{e}_2\ \cdots\ \mathbf{e}_m] \tag{8}$$

Given a set of vectors $\mathbf{v}_i$ and a set of real numbers α_i, $i = 1, \ldots, m$, such that $\sum_{i=1}^m \alpha_i = 1$, an *affine combination* of the vectors $\mathbf{v}_i$ is defined as $\sum_{i=1}^m \alpha_i \mathbf{v}_i$.

The key observation that affine combinations of estimates correspond exactly to affine combinations of their residuals can be stated mathematically as follows: *the affine combination of residuals* $\mathbf{r}_{\text{aff}} = \sum_{i=1}^m \alpha_i \mathbf{r}_i$ *is the residual that corresponds to the same affine combination of the estimates* $\mathbf{x}_i$*, i.e.,* $\mathbf{r}_{\text{aff}} = \mathbf{A}\mathbf{x}_{\text{aff}} - \mathbf{b}$*, where* $\mathbf{x}_{\text{aff}} := \sum_{i=1}^m \alpha_i \mathbf{x}_i$.

3.1 Minimizing affine combinations of residual in the 2-norm

In the context of iterative solution methods which aim to minimize the residual error norm, an estimate that produces a smaller residual norm is preferable to one that produces a larger norm. Thus, from the observation just made, the problem of finding an optimal combination of estimates, i.e., one that produces the smallest residual

norm, is equivalent to the problem of finding the residue of smallest norm that belongs to the affine subspace generated by the m residual vectors, where this subspace can be defined as:

$$\{\alpha_1 \mathbf{r}_1 + \cdots + \alpha_m \mathbf{r}_m \text{ s.t. } \alpha_1 + \cdots + \alpha_m = 1\}$$

and denoting a vector of ones by $\mathbf{1}_m := [1 \ \cdots \ 1]^\intercal \in \mathbb{R}^m$, the problem can be formulated as:

$$\min_{\text{s.t.}\boldsymbol{\alpha}^\intercal \mathbf{1}_m = 1} \|\mathbf{R}\boldsymbol{\alpha}\|_2^2 \tag{9}$$

This constrained minimization problem is solved by writing the Lagrangian

$$F(\boldsymbol{\alpha}, \mu) = \boldsymbol{\alpha}^\intercal \mathbf{R}^\intercal \mathbf{R} \boldsymbol{\alpha} + \mu(\boldsymbol{\alpha}^\intercal \mathbf{1}_m - 1)$$

where μ is the Lagrange multiplier. Setting the partial derivatives of F to zero yields:

$$\begin{array}{l} \frac{\partial F(\boldsymbol{\alpha},\mu)}{\partial \boldsymbol{\alpha}} = 2\mathbf{R}^\intercal \mathbf{R}\boldsymbol{\alpha} + \mu \mathbf{1}_m = \mathbf{0} \\ \frac{\partial F(\boldsymbol{\alpha},\mu)}{\partial \mu} = \boldsymbol{\alpha}^\intercal \mathbf{1}_m - 1 = 0 \end{array}$$

Solving for $\boldsymbol{\alpha}$ in terms of μ:

$$\boldsymbol{\alpha} = -\frac{\mu}{2}(\mathbf{R}^\intercal \mathbf{R})^{-1} \mathbf{1}_m$$

which implies $-\frac{\mu}{2}\mathbf{1}_m^\intercal (\mathbf{R}^\intercal \mathbf{R})^{-1}\mathbf{1}_m - 1 = 0$, that is $\mu = -\frac{2}{\mathbf{1}_m^\intercal (\mathbf{R}^\intercal \mathbf{R})^{-1}\mathbf{1}_m}$ and finally

$$\boldsymbol{\alpha} = \frac{(\mathbf{R}^\intercal \mathbf{R})^{-1}\mathbf{1}_m}{\mathbf{1}_m^\intercal (\mathbf{R}^\intercal \mathbf{R})^{-1}\mathbf{1}_m} := \hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1 \cdots \hat{\alpha}_m]^\intercal \tag{10}$$

The residual vector with smallest norm in the affine subspace is:

$$\hat{\mathbf{r}} = \mathbf{R}\hat{\boldsymbol{\alpha}} = \frac{\mathbf{R}(\mathbf{R}^\intercal \mathbf{R})^{-1}\mathbf{1}_m}{\mathbf{1}_m^\intercal (\mathbf{R}^\intercal \mathbf{R})^{-1}\mathbf{1}_m} = \hat{\alpha}_1 \mathbf{r}_1 + \cdots + \hat{\alpha}_m \mathbf{r}_m \tag{11}$$

3.2 Minimizing the cost function of affine combinations of estimates

The optimal affine combination (10-11) minimizes the norm of the residue. This procedure is particularly interesting when agents use an algorithm like *orthomin*, which produces a sequence of estimates with residues of monotonically decreasing 2-norm. However, the *steepest descent* algorithm does not produce such a sequence of estimates, but rather one with monotonically decreasing cost function values, $f(\mathbf{x})$ or, equivalently, monotonically decreasing weighted norms $\|\mathbf{r}\|_{A^{-1}}^2 = 2f(\mathbf{x}) + \mathbf{b}^\intercal \mathbf{x}^*$. Thus, the performance of cooperative SD algorithms may be improved upon by using optimal affine combinations which seek to minimize the cost function $f(\mathbf{x})$; i.e. the problem can be now defined as:

$$\min_{\text{s.t. } \boldsymbol{\alpha}^\intercal \mathbf{1}_m = 1} f(\mathbf{X}\boldsymbol{\alpha}) \tag{12}$$

Writing the Lagrangian $F(\boldsymbol{\alpha}, \mu) = f(\mathbf{X}\boldsymbol{\alpha}) + \mu(\boldsymbol{\alpha}^\intercal \mathbf{1}_m - 1) = \frac{1}{2}\boldsymbol{\alpha}^\intercal \mathbf{X}^\intercal \mathbf{A}\mathbf{X}\boldsymbol{\alpha} - \mathbf{b}^\intercal \mathbf{X}\boldsymbol{\alpha} + \mu(\boldsymbol{\alpha}^\intercal \mathbf{1}_m - 1)$ and setting its partial derivatives to zero yields:

$$\begin{array}{l} \frac{\partial F(\boldsymbol{\alpha},\mu)}{\partial \boldsymbol{\alpha}} = \mathbf{X}^\intercal \mathbf{A}\mathbf{X}\boldsymbol{\alpha} - \mathbf{X}^\intercal \mathbf{b} + \mu \mathbf{1}_m = \mathbf{0} \\ \frac{\partial F(\boldsymbol{\alpha},\mu)}{\partial \mu} = \boldsymbol{\alpha}^\intercal \mathbf{1}_m - 1 = 0 \end{array}$$

This leads to the solution

$$\boldsymbol{\alpha} = \frac{1 - \mathbf{1}_m^\intercal (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{b}}{\mathbf{1}_m^\intercal (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{1}_m} (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{1}_m + (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{b} := \hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1 \cdots \hat{\alpha}_m]^\intercal \tag{13}$$

The corresponding affine combination of the estimates is then:

$$\hat{\mathbf{x}} = \mathbf{X}\hat{\boldsymbol{\alpha}} = \frac{1 - \mathbf{1}_m^\intercal (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{b}}{\mathbf{1}_m^\intercal (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{1}_m} \mathbf{X}(\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{1}_m + \mathbf{X}(\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{b} \tag{14}$$

and the corresponding residual vector is:

$$\begin{array}{l} \hat{\mathbf{r}} = \mathbf{A}\hat{\mathbf{x}} - \mathbf{b} = \frac{1 - \mathbf{1}_m^\intercal (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{b}}{\mathbf{1}_m^\intercal (\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{1}_m} \mathbf{A}\mathbf{X}(\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{1}_m + \mathbf{A}\mathbf{X}(\mathbf{X}^\intercal \mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal \mathbf{b} - \mathbf{b} \\ \quad = \mathbf{R}\hat{\boldsymbol{\alpha}} \end{array} \tag{15}$$

Remark 1 In the two previous subsections, the optimal affine combinations presented minimize $\|\mathbf{R}\boldsymbol{\alpha}\|$ and $f(\mathbf{X}\boldsymbol{\alpha})$ $(= \|\mathbf{R}\boldsymbol{\alpha}\|_{A^{-1}})$, respectively. We unify these two cases by using the notation $\|\cdot\|_{\mathbf{A}^z}$, $z \in \mathbb{Z}$ where z usually takes the values $0, \pm 1$, with the convention that the A^0-norm is defined to be the 2-norm. This leads to the unified general problem:

$$\min_{\text{s.t. } \mathbf{1}_m^\intercal \boldsymbol{\alpha}=1} \|\mathbf{R}\boldsymbol{\alpha}\|_{\mathbf{A}^z} \tag{16}$$

4 Cooperative computation algorithms

This section defines two types of cooperative algorithms: Type 1a, with periodic deterministic information exchange; Type 1b, with randomized information exchange, and finally type 2, which is a class of autocooperative algorithms, in which the algorithm uses an optimal affine combination of its own current and delayed iterates, so that it can be thought of as cooperating with itself (thus $m = 1$). The algorithms are presented by means of pseudocode and an appropriate nomenclature is also introduced for each variant, consisting of a placeholder for each choice made in the algorithm and a superscript which represents a specific value for this choice. Capital letters (S, O) denote the algorithm chosen by each agent and lower case letters (z, p, d, t) represent parameter choices (respectively, norm, probability, deterministic period, and tolerance), according to the following description:

- Type 1a. $S^q O^{m-q} z^a d^N$: q agents using **S**teepest descent; $m-q$ agents using **O**rthomin; $a = 0$ implies use of 2-norm in optimal affine combination (eqs. 10-11), $a = -1$ implies use of A^{-1}-norm (eqs. 13-14); superscript $N \in \mathbb{N}$ (of letter d) denotes that communication occurs after N iterations.
- Type 1b. $S^q O^{m-q} z^a p^M$: q agents using **S**teepest descent; $m-q$ agents using **O**rthomin; here again a indicates the norm minimized by the affine combination; superscript M (of letter p) taking values in $[0, 1]$ denotes the probability of communication.
- Type 2. $S(O) z^a t^\epsilon$ one agent using **S**teepest descent or **O**rthomin; here too a indicates the norm minimized by the affine combination; the superscript ϵ (of letter t) denotes the tolerance used (see line 7 of algorithm 3).

Thus, for example, in the first class of algorithms, the notation $S^2 O^1 z^0 d^{10}$ signifies that a total of $2+1 = 3 = m$ agents cooperate, with 2 using the **S**teepest descent algorithm, one using the **O**rthomin algorithm, the 2-norm is used to obtain the optimal affine combination and **d**eterministic information exchange occurs every ten iterations.

We also define $\mathbf{x}_{k,i}$ to be the value of the estimate of the ith agent at iteration k.

Algorithm 1 Type 1a $S^q O^{m-q} z^a d^N$ cooperative computation algorithm (deterministic)

1: $k = 0$
2: For each agent, set $\mathbf{x}_{k,i}$, $i = 1, \ldots, m$ to the given initial value.
3: **while** stopping criterion is not met **do**
4: $\quad \mathbf{x}_{k+1,i} = \mathbf{x}_{k,i} - \rho_{k,i}\mathbf{r}_{k,i}$, where $\mathbf{r}_{k,i} = \mathbf{A}\mathbf{x}_{k,i} - \mathbf{b}$, and $\begin{cases} \rho_{k,i} = \dfrac{\mathbf{r}_{k,i}^\intercal \mathbf{r}_{k,i}}{\mathbf{r}_{k,i}^\intercal \mathbf{A}\mathbf{r}_{k,i}}, & i = 1, \ldots, q \quad \text{(SD)} \\ \rho_{k,i} = \dfrac{\mathbf{r}_{k,i}^\intercal \mathbf{A}\mathbf{r}_{k,i}}{\mathbf{r}_{k,i}^\intercal \mathbf{A}^2\mathbf{r}_{k,i}}, & i = q+1, \ldots, m \quad \text{(OM)} \end{cases}$
5: $\quad k = k + 1$
6: $\quad$ **if** k is a multiple of N **then**
7: $\quad\quad$ Choose an agent randomly, say the jth, which will update its estimate using $\hat{\mathbf{x}}$ the optimal affine combination (specified by a)
8: $\quad\quad$ Calculate the optimal affine combination $\hat{\mathbf{x}} = \mathbf{X}_k\hat{\boldsymbol{\alpha}} = [\mathbf{x}_{k,1}\, \mathbf{x}_{k,2} \cdots \mathbf{x}_{k,m}]\hat{\boldsymbol{\alpha}}$, where $\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}^\intercal \mathbf{1}_m = 1} \|\mathbf{R}_k\boldsymbol{\alpha}\|_{\mathbf{A}^a}$
9: $\quad\quad$ $\mathbf{x}_{k+1,j} = \hat{\mathbf{x}}$, the other agents maintain their estimates $\mathbf{x}_{k+1,i} = \mathbf{x}_{k,i}$, $i = 1, \ldots, j-1, j+1, \ldots, m$
10: $\quad\quad$ $k = k + 1$
11: $\quad$ **end if**
12: **end while**

In algorithm 3, the reason to use an optimal affine combination between the present estimate and a two step delayed estimate can be explained as follows. Akaike [3] showed that, when gradient descent algorithms are used, the error $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$ (as well as the residue $\mathbf{r}_k = \mathbf{A}\mathbf{e}_k$) tends to lie in a space spanned by the eigenvector associated with the smallest eigenvalue and the eigenvector associated with the largest eigenvalue. In addition, Akaike shows that asymptotically $\mathbf{r}_k$ alternates between these eigendirections and hence the vector $\mathbf{r}_k$ tends to have the same direction that the vector $\mathbf{r}_{k-2}$. The parallelism between these vectors can be detected by the conditions $\rho_k \simeq \rho_{k-2}$ or $\mathbf{r}_k^\intercal \mathbf{r}_{k-2}/\|\mathbf{r}_k\|\|\mathbf{r}_{k-2}\| \simeq 1$ (see [4]). If the parallelism is exact, it is easy to prove (see lemma 2 in the appendix 2) that $\hat{\mathbf{r}} = [\mathbf{r}_k\ \mathbf{r}_{k-2}]\hat{\boldsymbol{\alpha}} = \mathbf{0}$ and hence $\hat{\mathbf{x}} = \mathbf{x}^*$.

Algorithm 2 Type 1b $S^qO^{m-q}z^ap^M$ cooperative computation algorithm (probabilistic)

$k = 0$
For each agent, set $\mathbf{x}_{k,i}$, $i = 1, \ldots, m$ to the given initial value.
while stopping criterion is not met **do**
    $\mathbf{x}_{k+1,i} = \mathbf{x}_{k,i} - \rho_{k,i}\mathbf{r}_{k,i}$, where $\mathbf{r}_{k,i} = \mathbf{A}\mathbf{x}_{k,i} - \mathbf{b}$, and $\begin{cases} \rho_{k,i} = \frac{\mathbf{r}_{k,i}^\mathsf{T}\mathbf{r}_{k,i}}{\mathbf{r}_{k,i}^\mathsf{T}\mathbf{A}\mathbf{r}_{k,i}}, & i = 1, \ldots, q \quad \text{(SD)} \\ \rho_{k,i} = \frac{\mathbf{r}_{k,i}^\mathsf{T}\mathbf{A}\mathbf{r}_{k,i}}{\mathbf{r}_{k,i}^\mathsf{T}\mathbf{A}^2\mathbf{r}_{k,i}}, & i = q+1, \ldots, m \quad \text{(OM)} \end{cases}$
    $k = k + 1$
    if random$(0, 1) < M$ **then**
        Choose an agent randomly, say the jth, which will update its estimate using $\hat{\mathbf{x}}$ the optimal affine combination (specified by a)
        Calculate the optimal affine combination $\hat{\mathbf{x}} = \mathbf{X}_k\hat{\boldsymbol{\alpha}} = [\mathbf{x}_{k,1}\, \mathbf{x}_{k,2} \cdots \mathbf{x}_{k,m}]\hat{\boldsymbol{\alpha}}$, where $\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}^\mathsf{T}\mathbf{1}_m=1} \|\mathbf{R}_k\boldsymbol{\alpha}\|_{\mathbf{A}^a}$
        $\mathbf{x}_{k+1,j} = \hat{\mathbf{x}}$. The other agents maintain their estimates i.e., $\mathbf{x}_{k+1,i} = \mathbf{x}_{k,i}$, $\mathbf{x}_{k+1,i} = \mathbf{x}_{k,i}$, $i = 1, \ldots, j-1, j+1, \ldots, m$
        $k = k + 1$
    end if
end while

Algorithm 3 Type 2 $S(O)z^at^\epsilon$ autocooperative computation algorithm

$k = 0$
Set $\mathbf{x}_k$, $\mathbf{x}_{k-1}$ to the given initial values.
Choose tolerance ϵ
while stopping criterion is not met **do**
    $\mathbf{x}_{k+1} = \mathbf{x}_k - \rho_k\mathbf{r}_k$, where $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k - \mathbf{b}$, and $\begin{cases} \rho_k = \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k} & \text{(SD)} \\ \rho_k = \frac{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\mathsf{T}\mathbf{A}^2\mathbf{r}_k} & \text{(OM)} \end{cases}$
    $k = k + 1$
    if $|\rho_k - \rho_{k-2}| < \epsilon$ or **if** $1 - \mathbf{r}_k^\mathsf{T}\mathbf{r}_{k-2}/\|\mathbf{r}_k\|\|\mathbf{r}_{k-2}\| < \epsilon$ **then**
        $\mathbf{x}_{k+1} = \hat{\mathbf{x}} = [\mathbf{x}_k \;\; \mathbf{x}_{k-2}]\hat{\boldsymbol{\alpha}}$, where $\hat{\boldsymbol{\alpha}} = \arg\min_{\boldsymbol{\alpha}^\mathsf{T}\mathbf{1}_2=1} \|[\mathbf{r}_k \;\; \mathbf{r}_{k-2}]\boldsymbol{\alpha}\|_{\mathbf{A}^a}$.
        $k = k + 1$
    end if
end while

Brezinski and Redivo-Zaglia first presented a version of the cooperative computation algorithm using optimal affine combination (10-11) in [16], calling their method a hybrid procedure. The authors considered only the affine combination of two residual vectors ($m = 2$). The different cases studied in [16] are particular cases of the general CC algorithm presented here. The authors of [16] also considered the affine combination between an estimate and a one step delayed estimate (i.e. between $\mathbf{r}_k$ and $\mathbf{r}_{k-1}$), without noting the advantage of using the two step delayed residue $\mathbf{r}_{k-2}$ when a gradient descent method is used to update the estimate.

Moreover, in [16] the authors proposed the application of the method at all the iterations (considered here as the deterministic model with $N = 1$, or the probabilistic model with $M = 1$). As experimental results below will show, in the case of concurrent and asynchronous communication, a low frequency of communication (i.e., small values of M and large values of N) seems to be more efficient than a high one.

5 Experimental results

The algorithms presented in the former sections, as well as the *conjugate gradient* (CG) algorithm and the *Barzilai-Borwein* (BB) algorithm will be tested using MATLAB 7. We emphasize that the CG algorithm is implemented only in order to obtain a sequential "gold standard" against which to compare the concurrent algorithms being proposed and studied in this paper.

5.1 Experiments with randomly generated matrices of small dimensions

In a first series of experiments, a suite of small randomly generated symmetric positive definite matrices will be used. The dimensions of the matrices are 50, 100, 200 and 300, respectively, and they have condition numbers of 10^3, 10^4 and 10^5. Table 2 shows the names used for each matrix. The reason to use matrices of small dimensions in this first series of experiments is to understand the performance of the CC algorithms with respect to the location of the eigenvalues, the condition number, among other characteristics. Matrices of small dimensions allow the manipulation of these quantities.

Table 2 Properties of randomly generated s.p.d. test matrices: size, condition number

Matrix	Dimension	Condition no.
$\mathbf{A}1$	50	10^3
$\mathbf{A}2$	50	10^4
$\mathbf{A}3$	50	10^5
$\mathbf{B}1$	100	10^3
$\mathbf{B}2$	100	10^4
$\mathbf{B}3$	100	10^5
$\mathbf{C}1$	200	10^3
$\mathbf{C}2$	200	10^4
$\mathbf{C}3$	200	10^5
$\mathbf{D}1$	300	10^3
$\mathbf{D}2$	300	10^4
$\mathbf{D}3$	300	10^5

The matrices are generated in the following way: a smallest eigenvalue λ_1 is randomly chosen between 1 and 100. The largest eigenvalue is calculated as $\lambda_n = \kappa\lambda_1$, where κ is the condition number. The other eigenvalues are randomly chosen between λ_1 and λ_n. Next, a diagonal matrix is generated as $\boldsymbol{\Lambda} = \text{diag}\,(\lambda_1, \ldots, \lambda_n)$, and 4 random orthonormal matrices (one for each dimension) $\mathbf{U} \in \mathbb{R}^{n\times n}$ are generated. Finally, the matrices presented in Table 2 are generated as $\mathbf{U}^\intercal\boldsymbol{\Lambda}\mathbf{U}$. In addition, four random right hand sides $\mathbf{b} \in \mathbb{R}^n$ are generated (one for each dimension).

In all the cases, the algorithms will be tested from 20 different initial points per agent. All these initial points are localized on a hypersphere of norm 1 surrounding the solution point, that is $\|\mathbf{x}_{0_j,i} - \mathbf{x}^*\| = 1,\ \forall j \in \{1, \ldots, 20\},\ \forall i \in \{1, \ldots, m\}$.

The matrices used and reported in Table 2, as well as the right hand sides $\mathbf{b}$ and the initial points used are available on request.

The following conditions will be observed:

- In all the cases, the stopping criterion is that at least one agent have a norm of its residue smaller than 10^{-4}.
- In the CG algorithm the initial condition $\mathbf{p}_0 = \mathbf{r}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b}$ will be used.
- In the BB algorithm ρ_{-1} will be chosen as $\rho_0 = \frac{\mathbf{r}_0^\intercal\mathbf{r}_0}{\mathbf{r}_0^\intercal\mathbf{A}\mathbf{r}_0}$.

The algorithms tested are CG, BB, $Oz^0t^{0.1}$, $S^1O^1z^0p^{0.2}$, $S^1O^1z^0d^6$, $S^1O^2z^0p^{0.2}$, $S^1O^2z^0d^5$, $O^3z^0p^{0.2}$, $O^3z^0d^6$, $S^1O^1z^{-1}p^{0.1}$, $S^1O^1z^{-1}d^{10}$, $S^2O^1z^{-1}p^{0.1}$, $S^2O^1z^{-1}d^{10}$, $S^1O^2z^{-1}p^{0.1}$ and $S^1O^2z^{-1}d^{10}$. Many other CC algorithms were also tested, as well as modifications of the algorithm such as one in which a fixed predetermined agent always receives the information (instead of choosing the receptor agent randomly) or choosing different norms in order to optimize the affine combination according to the agent which receives the information (instead of always minimizing the same norm); however, the results were worse than the presented here and for that reason they are omitted. We remark that a low number of agents is used in the CC algorithms tested (up to 3) in order to obtain a low computational cost for the optimal affine combinations. These calculations require the solution of $m-$dimensional linear systems, and the results can be explicitly formulated for systems of small dimensions, otherwise, the solution of an $m-$dimensional linear system by LU factorization requires the calculation of $\frac{m(m+1)(2m+1)}{6} - m$ scalar products and additions and $\frac{m(m+1)}{2}$ scalar divisions ([24, p. 15]). The communication cost, which is neglected in our experiments, also increases with the number of agents, as well as with an increasing communication frequency.

Table 3 shows the results of the experiments, where the mean number of iterations from every initial point and the standard deviation are reported.

The number of floating point operations (scalar products plus scalar divisions) calculated by each algorithm (in the case of the CC algorithms, by each agent) were also counted in this first series of experiments. However, this number is not presented in Table 3 because the performance of the algorithms in terms of the floating point operations is similar to that quantified by the number of iterations. The reason that the number of floating point operations is roughly proportional to the number of iterations can be explained by the formulas presented in Table 4.

The cooperative computation algorithms of the type 1 reported in Table 4 assume that the solution of m-dimensional linear systems is realized using a gaussian elimination by LU factorization. However, as pointed out before in this section, if the number of agents m is small enough, then the use of the closed-form linear system solution results in a lower flop count. This is the case for the CC algorithms of the type 2, in which affine

Table 3 Mean number of iterations (upper number) and standard deviation (lower number) for each algorithm. Trajectories began at 20 different initial points per agent, all of them with norm one surrounding the solution point. $\mathbf{b} \neq \mathbf{0}$. The minimum number of iterations in each case is in boldface and blue (excepting the CG). A dash means that the algorithm does not converge in 10000 iterations.

matrix	CG	BB	$Oz^0t^{0.1}$	$S^1O^1z^0p^{0.2}$	$S^1O^1z^0d^6$	$S^1O^2z^0p^{0.2}$	$S^1O^2z^0d^5$	$O^3z^0p^{0.2}$	$O^3z^0d^6$	$S^1O^1z^{-1}p^{0.1}$	$S^1O^1z^{-1}d^{10}$	$S^2O^1z^{-1}p^{0.1}$	$S^2O^1z^{-1}d^{10}$	$S^1O^2z^{-1}p^{0.1}$	$S^1O^2z^{-1}d^{10}$
A1	43.95	245.1	413.6	285.55	344.25	167.60	144.95	168.25	163.90	193.85	181.55	127.8	126.4	**125.50**	127.85
	0.22	58.39	65.06	92.07	127.82	26.23	22.51	42.72	28.44	40.69	44.82	24.34	22.55	16.91	21.84
A2	51	1055	2159	1393	1638	573.4	603.65	861.30	669.90	945.40	843.65	567	529.5	444.80	**414.15**
	0	234.16	422.99	314.19	434.78	89.65	77.86	293.68	145.87	164.41	198.48	185	224.3	68.18	75.51
A3	47	2475	1131	1730	2134	190	176.65	153	144.80	378.35	426.55	119.9	118.85	122.55	**114.15**
	0	996.66	256.96	661.28	1250	42.21	38.83	28.54	22.30	282.11	509.91	20.47	27.63	31.30	24.52
B1	54.8	264.8	1350	282.55	365.2	154.05	138.55	157.70	152.35	187.05	183.1	127.2	128.45	120.95	**107.45**
	0.41	51.87	384.10	75.61	119.50	34.10	20.60	30.94	24.53	49.90	38.40	22.69	29.95	18.34	10.61
B2	65.95	966.20	1130	1082	1338	499.70	478.55	280.30	276.70	632.8	568.35	200.75	**194.6**	205.25	275.55
	0.22	250.84	181.43	226.07	363.73	141.47	130.13	56.48	57.80	330.28	244.81	33.67	54.03	33.05	234.67
B3	80.10	3238	1350	1435	2772	277.15	243.05	296.25	269	606.3	401.8	232.65	223.4	204.3	**196.85**
	0.85	1366	384.10	647.59	1382	48.29	41.85	89.20	82.56	567.74	207.20	42.23	41.28	30.79	32.75
C1	92.25	**291.9**	542.75	403.35	462.85	359.80	356.50	518.60	406.45	383.95	351.15	385.15	349.1	344.55	323.1
	0.55	38.53	44.95	53.73	72.68	34.45	37.84	128.66	65.43	45.56	36.30	59.09	45.41	53.79	31.95
C2	102.45	1048	1859	1355	1827	717	725.15	1043	822.95	1030	948.45	661.05	581.9	587.8	**518.95**
	0.60	212.49	346.19	319.80	311.80	92.76	93.04	210.38	156.64	349.09	204.68	136.57	131.94	108.3	94.38
C3	105.25	3672	5421	3817	5618	725.85	652.20	871.30	708.40	1641	1526	548.9	**480.4**	491.1	486.15
	0.55	1150	2782	1564	2448	114.12	135.65	220.29	131.17	734.26	839.73	281.27	83.77	71.49	192.35
D1	68.55	270.55	363.35	305	342.80	192.05	180.05	209.40	193.30	244.90	237.75	149.8	151.05	143	**135.1**
	1.27	45.45	53.25	50.78	84.93	42.63	33.74	45.07	51.38	52.64	41.10	32.25	29.38	18.20	26.50
D2	161.6	920.20	1474	1183	1469	837.10	739.55	1043	922.95	949.6	835.95	767.85	679.35	706.85	**592.4**
	1.84	191.66	269.27	132.44	209.90	174.81	111.72	222.22	201.31	207.89	165.76	150.23	150.48	137.74	85.76
D3	135.55	3477	5593	4702	5640	1618	1569	1807	**1465**	91514	-	5286	5394	1639	1749
	0.51	1057	1824	1209	1691	366.83	267.75	483.88	431.51	193300	-	8042	8956	1176	1154

Table 4 Number of floating point operations (scalar products plus scalar divisions) calculated by different algorithms to solve symmetric n-dimensional linear systems. *it*: number of iterations. *ac*: number of affine combinations. In the CC algorithms m is the number of agents and the iteration count excludes the iterations in which an affine combination is calculated (in algorithms 1 to 3 the step in which an affine combination is calculated is considered as an additional iteration), so the number of operations per agent is the same as that carried out by the SD-OM algorithms per iteration plus the operations calculated in the affine combinations.

algorithm	number of floating point operations
CG	$n^2 + it(2n^2 + 7n + 2)$
BB	$2n^2 + it(n^2 + 4n + 1)$
SD-OM	$n^2 + it(n^2 + 4n + 1)$
CC Type 1, $a = 0$	$n^2 + it(n^2 + 4n + 1) + ac\left(\frac{m^3}{3} + \left(\frac{n}{2} + 1\right)m^2 + \left(\frac{5}{2}n - \frac{1}{3}\right)m + 1\right)$
CC Type 1, $a = -1$	$n^2 + it(n^2 + 4n + 1) + ac\left(\frac{2}{3}m^3 + \left(\frac{n}{2} + 2\right)m^2 + \left(n^2 + \frac{7}{2}n + \frac{1}{3}\right)m + 2\right)$
CC Type 2, $a = 0$	$n^2 + it(n^2 + 4n + 1) + ac(7n + 2)$
CC Type 2, $a = -1$	$n^2 + it(n^2 + 4n + 1) + ac(2n^2 + 9n + 11)$

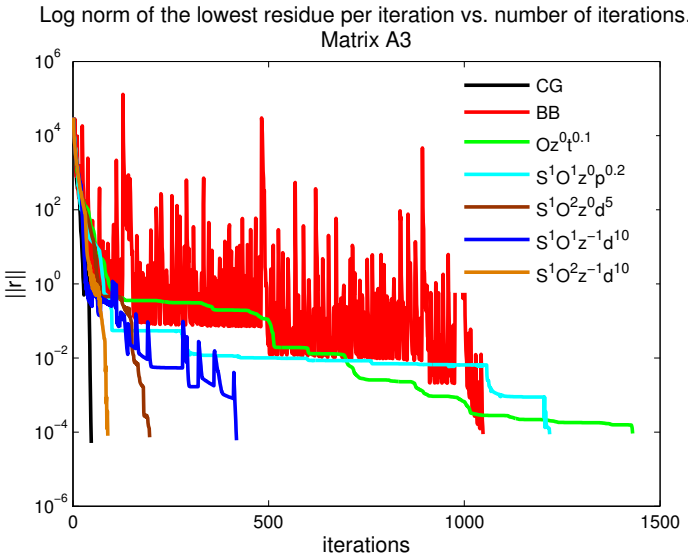

Fig. 1 Plot of the logarithm of the lowest residual norm per agent and per iteration versus iteration number for the CG, BB, $S^1O^1z^0p^{0.2}$, $Oz^0t^{0.1}$, $S^1O^1z^{-1}d^{10}$, $S^1O^2z^{-1}d^{10}$ and $S^1O^2z^0d^5$ algorithms for matrix $\mathbf{A}3$, with a fixed vector $\mathbf{b} \in \mathbb{R}^{50}$ and from initial points $\mathbf{x}_{0,i} \in \mathbb{R}^{50}$, $i = 1, \ldots, 3$.

combinations are realized between the present estimate of the agent and the two step delayed estimate. The small number of agents m used by the CC algorithms in the experiments realized in this first series (up to 3), as well as the infrequent exchange of information (which implies a small value of the parameter *ac* (number of affine combinations) in Table 4), is the reason that the number of floating point operations calculated by the agents when gradient descent methods are used is greater than the operations calculated in the affine combinations.

For illustrative purposes, figures 1 to 4 show the norm of the residual vector (in the cases with more than one agent, at each iteration the residue plotted is chosen as that with lowest norm among the m estimates) versus iteration number for several algorithms with the matrices $\mathbf{A}3$, $\mathbf{B}2$, $\mathbf{C}2$ and $\mathbf{D}1$, respectively. The initial points chosen are located at a hypersphere of norm one surrounding the solution point. The vectors $\mathbf{b}$ are the same used in Table 3. Note that, from the initial points chosen, none of the algorithms tested converge faster than the CG algorithm, but in all the cases several of the CC algorithms converge faster than the BB algorithm.

Table 3 shows that the performance of the CC algorithms does not always improve when the dimension of the matrix decreases; for example, all the algorithms tested work better with the matrix $\mathbf{D}1 \in \mathbb{R}^{300\times300}$ than with the matrix $\mathbf{C}1 \in \mathbb{R}^{200\times200}$ (although both have $\kappa = 10^3$). It was also noted that performances do not necessarily improve when the condition number decreases; for example, with the matrix $\mathbf{A}3 \in \mathbb{R}^{50\times50}$ ($\kappa = 10^5$) almost all the performances are better than with the matrix $\mathbf{A}2 \in \mathbb{R}^{50\times50}$ ($\kappa = 10^4$).

A large number of tests, not presented here for lack of space, for randomly generated matrices (according to the recipe given above) show that the performance of the CC algorithms depends strongly on the location of the second eigenvalue λ_2.

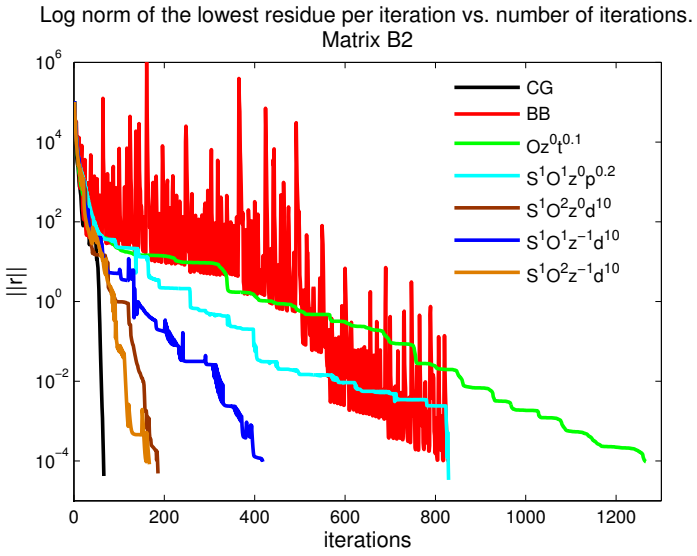

Fig. 2 Plot of the logarithm of the lowest residual norm per agent and per iteration versus iteration number for the CG, BB, $S^1O^1z^0p^{0.2}$, $Oz^0t^{0.1}$, $S^1O^1z^{-1}d^{10}$, $S^1O^2z^{-1}d^{10}$ and $S^1O^2z^0d^5$ algorithms for matrix $\mathbf{B}2$, with a fixed vector $\mathbf{b} \in \mathbb{R}^{100}$ and from initial points $\mathbf{x}_{0,i} \in \mathbb{R}^{100}$, $i = 1, \ldots, 3$.

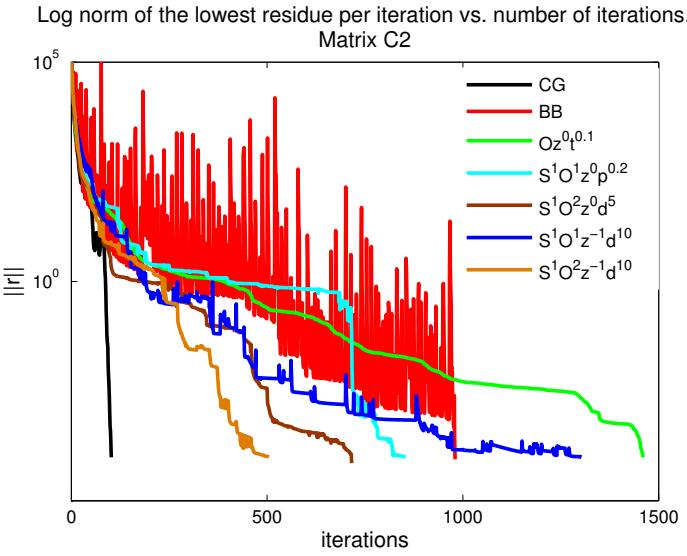

Fig. 3 Plot of the logarithm of the lowest residual norm per agent and per iteration versus iteration number for the CG, BB, $S^1O^1z^0p^{0.2}$, $Oz^0t^{0.1}$, $S^1O^1z^{-1}d^{10}$, $S^1O^2z^{-1}d^{10}$ and $S^1O^2z^0d^5$ algorithms for matrix $\mathbf{C}2$, with a fixed vector $\mathbf{b} \in \mathbb{R}^{200}$ and from initial points $\mathbf{x}_{0,i} \in \mathbb{R}^{200}$, $i = 1, \ldots, 3$.

A study of the relation between the performance of the CC algorithms and the location of the eigenvalues is presented in appendix 1.

5.2 Experiments with standard test matrices

In the series of experiments reported in this section a standard suite of small matrices which arise from real applications will be used in order to illustrate the behavior of the CC algorithms on this set. The matrices used in the experiments reported in Table 5 were taken from [1]. The right hand side $\mathbf{b}$ was randomly chosen for each matrix. The initial points, stopping criterion and the conditions observed by the algorithms are the same used in the experiments reported in Section 5.1.

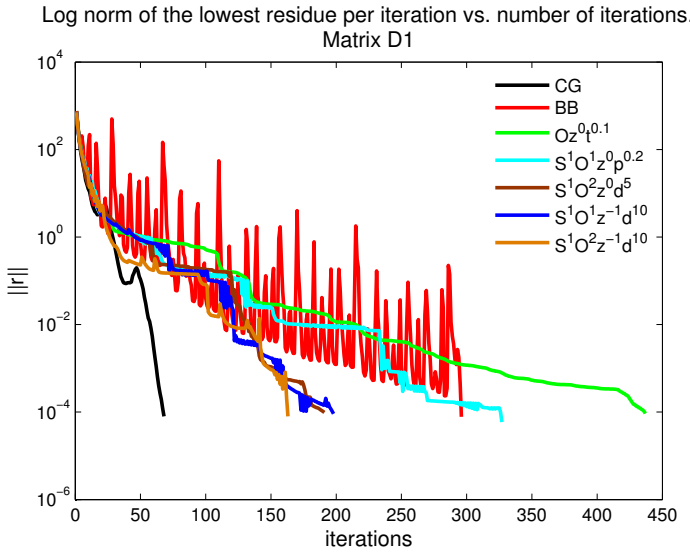

Fig. 4 Plot of the logarithm of the lowest residual norm per agent and per iteration versus iteration number for the CG, BB, $S^1O^1z^0p^{0.2}$, $Oz^0t^{0.1}$, $S^1O^1z^{-1}d^{10}$, $S^1O^2z^{-1}d^{10}$ and $S^1O^2z^0d^5$ algorithms for matrix $\mathbf{D}1$, with a fixed vector $\mathbf{b} \in \mathbb{R}^{300}$ and from initial points $\mathbf{x}_{0,i} \in \mathbb{R}^{300}$, $i = 1, \ldots, 3$.

In table 5, note that there always exists a cooperative computation algorithm with better performance than the BB algorithm for all the matrices tested. Moreover, the autocooperative computation algorithm of the type 2 $Oz^0t^{0.1}$ presents a lower mean number of iterations than those presented by the CG algorithm for three of the matrices tested.

In the experiments reported in Table 6 matrices of large dimensions also extracted from [1] were used. The algorithms were implemented in C language. The machine used in the experiments made with the first two matrices reported in Table 6 was a Silicon Graphics SGI Altix ICE 8400. This machine has 64 CPUs, with each CPU being a Quad Core Intel Xeon X5355 (Clovertown) running at 2.66 GHz. However, since our experiments were designed to test the possibilities of multi-thread implementations, with each thread corresponding to one agent, and the number of agents was kept very small, all experiments were run on the 4 cores of one of the CPUs of the Altix machine, or on up to 8 cores on 2 CPUs. The experiments realized with the last five matrices were made in a laptop Dell Vostro 3560 with a CPU Intel Core i7-3632 QM running at 2.20 GHz. The only criterion used to choose the matrices was to check whether all the algorithms being compared converged in less than 100000 iterations.

The results reported in table 6 show that the $S^1O^2z^0d^5$ algorithm converges faster (in wall clock time) than all the others it is tested against, for all matrices except the second, for which sequential CG is the fastest. This indicates that the family of the proposed CC algorithms becomes more competitive as matrix size increases and is therefore worth testing more extensively in practical applications.

5.3 Experiments varying from two through ten number of agents

In this section the behavior of the CC algorithms of the type 1 with an increasing number of agents will be studied. We will use some small matrices presented in Section 5.2 extracted from [1], and the algorithms tested are $S^qO^1z^{-1}d^{10},\ q \in \{1, \ldots, 9\}$ and $S^1O^iz^0p^{0.2},\ i \in \{1, \ldots, 9\}$. These algorithms presented a low number of iterations with $q = i = 2$ in the experiments reported in Section 5.2. All the algorithms will be tested from 20 different initial points per agent, all of them with norm one surrounding the solution point, i.e. $\|\mathbf{x}_{0_j,i} - \mathbf{x}^*\| = 1, \forall j \in \{1, \ldots, 20\}, i \in \{1, \ldots, m\}$, and we will use the same stopping criterion used in the previous experiments. The BB algorithm will be also tested from the 20 initial points used by the first agent of the CC algorithms. The goal of this series of experiments is to compare the mean number of iterations needed to met the stopping condition from every initial point by each CC algorithm tested as well as the number of floating point operations (scalar products plus scalar divisions) calculated by each CC algorithm with respect to those presented by the BB algorithm. The iteration speedup and the floating point operation speedup as a function of the number of agents used by the CC algorithms tested are presented in Figures 5 to 8.

Table 5 Mean number of iterations (upper number) and standard deviation (lower number) for each algorithm. Trajectories began at 20 different initial points per agent, all of them with norm one surrounding the solution point. $\mathbf{b} \neq \mathbf{0}$. The minimum number of iterations in each case is in boldface and blue (excepting the CG). A dash means that the algorithms do not converge in 100000 iterations.

matrix	n	κ	CG	BB	$Oz^0t^{0.1}$	$S^1Oz^0p^{0.2}$	$S^1Oz^0d^6$	$S^1Oz^0p^{0.2}$	$S^1Oz^0d^5$	$O^3z^0p^{0.2}$	$O^3z^0d^6$	$S^1Oz^{-1}p^{0.1}$	$S^1Oz^{-1}d^{10}$	$S^2Oz^{-1}p^{0.1}$	$S^2Oz^{-1}d^{10}$	$S^1O^2z^{-1}p^{0.1}$	$S^1O^2z^{-1}d^{10}$
Bai\mhdb416	416	3.99e9	42.65	96.75	**93.8**	153.9	175.4	131.7	124.7	236.35	154.7	774.2	1489	344.7	287.25	346.85	337.05
			3.42	8.79	7.28	28.05	40.96	21.61	25.49	131.50	21.52	944.32	2261	127.88	78.18	125.26	187.89
FIDAP\ex5	27	6.65e7	110.1	1558600	12654	44769	200460	**11950**	20390	104390	76397	13733	13696	25095	22555	19538	18502
			6.58	306500	8221	9891	38668	1975	4534	31148	22930	4.717	3875	5795	5068	4809	3769
HB\494bus	494	2.41e6	994.3	6453	8238	12284	19541	6695	**6147**	10294	8220	8559	9403	8952	8707	8227	7182
			34.44	2478	2744	3948	6527	1398	1376	2550	2118	3617	3750	2546	2607	2222	1623
HB\bcsstm06	420	3.45e6	110.9	4576	7283	6258	10999	**1169**	1303	2358	1782	2992	3476	1851	1892	1849	1667
			0.30	2205	4842	2905	5395	294.09	328.64	822.77	350.22	2023	2096	780.92	521.21	512.57	643.36
HB\nos3	960	3.77e4	212	601.8	423.65	828.15	1006	474.85	**420.65**	759	578.05	866.2	880.30	651.80	579.15	612.45	507.95
			18.82	307.01	150.48	234.78	393.31	130.72	83.33	233.90	192.90	437.62	426.68	196.44	173.00	157.58	130.38
Pothen\mesh3e1	289	8.92	15	18.2	**4.9**	29.1	28.95	27.55	25.9	32.15	32	28.85	28.25	28.45	29.05	29.85	26.20
			0	0.41	0.30	3.66	4.08	3.88	3.46	2.36	3.37	3.66	3.04	4.81	3.66	2.88	2.54
HB\nos6	675	7.65e6	966.9	46363	31308	63531	163360	22586	23211	49273	29932	26827	25239	28723	27898	**21863**	23346
			27.92	12240	5980	16669	33710	2198	2989	6214	4737	7348	7719	5756	6573	3645	5086
Pothen\mesh1em6	48	6.10	11.85	14.8	**4.05**	22.05	21.65	20.40	18.9	20.45	19.40	21.7	21.25	20.7	20.05	20.85	19.95
			0.36	0.89	0.22	3.70	2.10	2.08	2.63	3.05	2.43	2.0	2.35	2.67	2.45	2.03	2.35
HB\plat362	362	2.17e11	51.7	41.7	**27.6**	55.15	57.10	51.8	45.65	72.40	63.7	-	-	-	-	-	-
			7.52	7.79	6.14	8.08	13.42	10.65	7.16	19.71	18.91	-	-	-	-	-	-

Table 6 Mean time (column t, in seconds) and mean number of iterations (column it.) for each algorithm. Trajectories began at 20 different randomly chosen initial points per agent, $\mathbf{b} \neq \mathbf{0}$. The minimum mean time, now including the CG algorithm, for each matrix is in boldface and blue. A dash means that the algorithms do not converge in 100000 iterations.

matrix	n	CG		BB		$S^1O^1z^0p^{0.2}$		$S^1O^2z^0d^5$		$S^1O^1z^{-1}d^{10}$		$S^1O^2z^{-1}d^{10}$	
		t	it.	t	it.	t	it.	t	it.	t	it.	t	it.
ND\nd12k	36000	120.8	1008.45	184.8	1121.05	75.096	1466.65	**73.8**	1113.30	-	-	-	-
GHS_psdef\cvxbpq1	50000	**32.85**	7818.85	1054.8	33258.85	42.6	21809.50	43.69	12972.65	37.54	12903.30	38.95	11619.70
Boeing\bcsstm39	46772	0.7246	196.6	0.4549	245.5	0.4624	303.2	**0.3367**	227.2	0.5399	266.55	0.5523	250.45
Rothberg\cfd2	123440	4.7023	183.15	1.8743	144.4	2.3896	222.4	**1.5683**	151.3	3.1135	220.55	3.6775	239.15
GHS_psdef\wathen100	30401	0.3255	209.05	0.2048	261.75	0.2350	360.7	**0.1672**	267.45	0.2346	274.15	0.2420	259.6
GHS_psdef\gridgena	48962	6.4859	1609.5	5.1078	2533.8	6.8167	4053.4	**4.0392**	2504.8	5.3147	2398.1	5.3057	2194.8
JGD_Trefethen\Trefethen20000b	19999	0.8517	1266.6	0.8298	2466.9	1.1499	4098.8	**0.6556**	2436.4	0.8646	2338.2	0.8044	1994.5

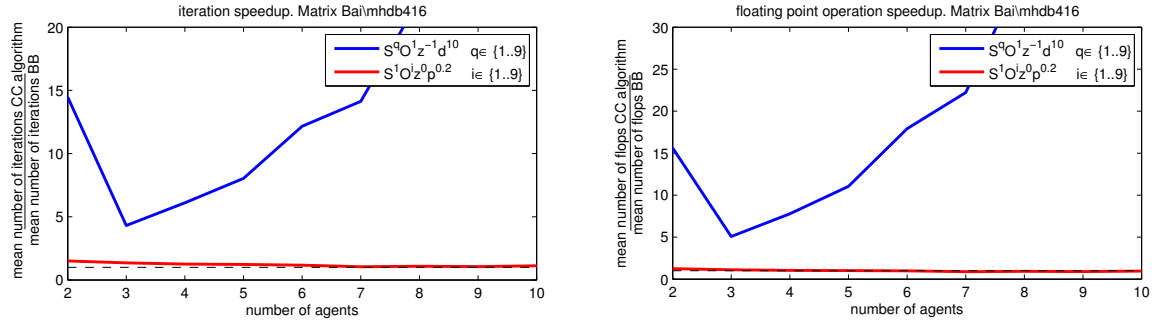

Fig. 5 Iteration speedup (left) and floating point operation speedup (right) presented by the CC algorithms tested with respect to those presented by the BB algorithm as a function of the number of agents. Matrix Bai\mhdb416. The dashed black line indicates speedup equal to 1.

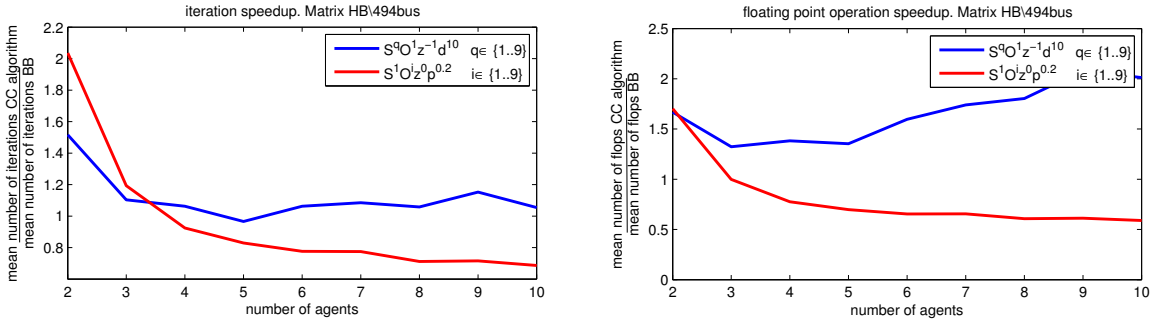

Fig. 6 Iteration speedup (left) and floating point operation speedup (right) presented by the CC algorithms tested with respect to those presented by the BB algorithm as a function of the number of agents. Matrix HB\494bus.

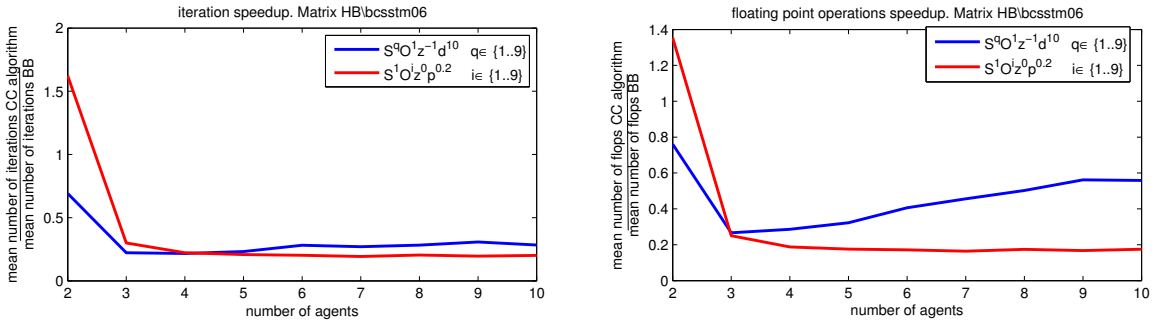

Fig. 7 Iteration speedup (left) and floating point operation speedup (right) presented by the CC algorithms tested with respect to those presented by the BB algorithm as a function of the number of agents. Matrix HB\bcsstm06.

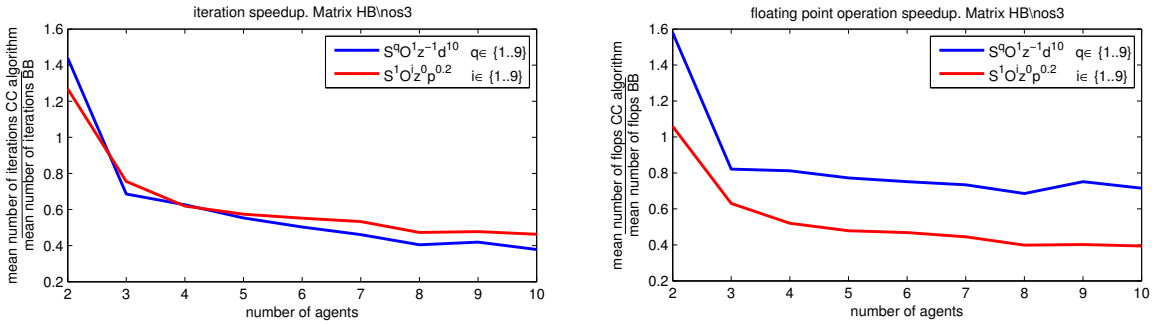

Fig. 8 Iteration speedup (left) and floating point operation speedup (right) presented by the CC algorithms tested with respect to those presented by the BB algorithm as a function of the number of agents. Matrix HB\nos3.

With the matrix Bai\mhdb416 (Figure 5), the algorithm $S^1O^iz^0p^{0.2}$ calculates a smaller number of floating point operations than that calculated by the BB algorithm for all $i \geq 5$. However, the iteration speedup is not smaller than one for any number of agents tested.

5.4 Conclusions from the experiments

The experiments carried out with the matrices described in Tables 2 and 5, with corresponding results presented in Tables 3, 5, 6 and 7 (in appendix 1), as well as in the figures 1 to 12, allow us to arrive at some conclusions:

1) In the cases where more than one agent was used, the probabilistic and the deterministic models were tested. From the results obtained in the experiments reported, there seems to exist a probability M which is as efficient as using a deterministic model with communication every N iterations, as can be observed by comparing corresponding columns in Tables 3-5. This implies that deterministic algorithms are just as good as probabilistic ones, for adequate parameter choices.

2) For both of the models, in all cases the best probability M and the best number of iterations N between information exchanges indicate that an infrequent exchange of information is usually the best strategy, e.g. $M = 0.1$ and $N = 10$. This means that a frequent use of an optimal affine combination, for example in every iteration as proposed in [16], is less efficient.

3) The experiments reported in section 5.3 seem to indicate that the use of a small number of agents is usually the best strategy, because the speedups seem to reach a minimum value with a small number of agents, principally the floating point operation speedups in the CC algorithms which minimize the $\mathbf{A}^{-1}$ norm. The communication cost, which was neglected in the tests reported here, also increases with the number of agents.

6 Conclusions

This paper introduced a simple paradigm of cooperative computation for the special case of iterative solution of symmetric linear systems. The resulting cooperative computation algorithm has excellent convergence properties, comparable to that of the optimal sequential conjugate gradient algorithm, for a subset of symmetric matrices. For this set, which remains to be characterized theoretically, the cooperative computation algorithm is partially concurrent and can perform well with infrequent asynchronous communication between the processors or agents, each of which has to solve the whole linear system, but only needs to exchange the current solution approximation with the other agents. As far as the authors know, this is the only such concurrent algorithm for the class of symmetric linear systems which comes close to being as efficient as the conjugate gradient algorithm. In light of the present encouraging results, the authors consider it interesting to explore further how sparse asynchronous information exchange mechanisms can speed up optimization algorithms, as well as preconditioning schemes able to manipulate the location of the eigenvalues in order to improve their performances.

Acknowledgements The first author would like to thank Prof Uri Ascher for interesting discussions about the Barzilai–Borwein algorithm, and also for making preprints of his work on this algorithm available. The third author would like to thank Leonardo Ferreira for advice regarding implementation of the algorithms. All authors would like to thank the reviewers for their constructive comments.

Appendix 1: Experiments with different values of λ_2

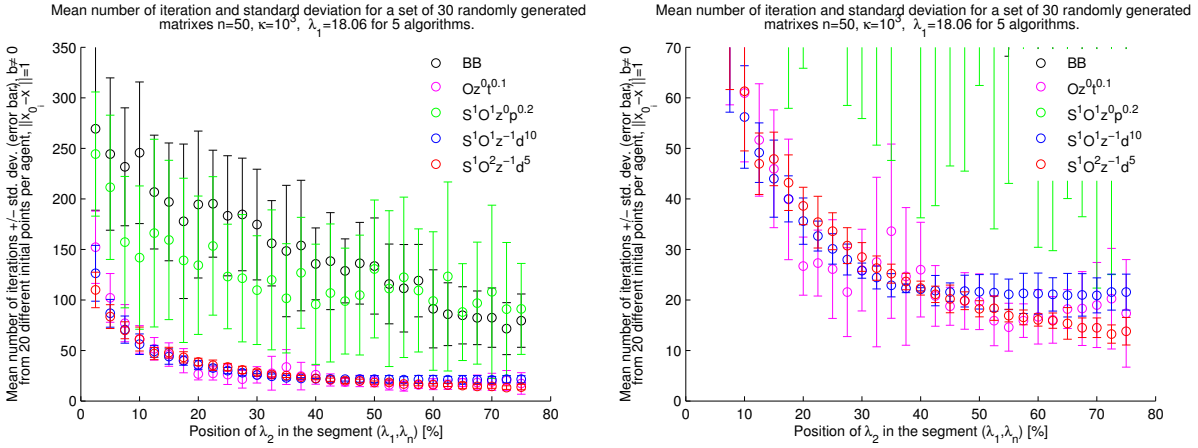

Fig. 9 Statistics of the algorithms BB, $Oz^0t^{0.1}$, $S^1O^1z^0p^{0.2}$, $S^1O^2z^0d^5$ and $S^1O^1z^{-1}d^{10}$ for a set of 30 random matrices of dimension 50, condition number 1000, with the mean number of iterations and standard deviation calculated from 20 different initial points per agent, all of them located on a hypersphere such that $\|\mathbf{x}_{0_j,i} - \mathbf{x}^*\| = 1$. The right hand side $\mathbf{b} \in \mathbb{R}^{50}$ is randomly chosen. The second eigenvalue increases for each matrix. The figure to the right is a zoom of the one to the left.

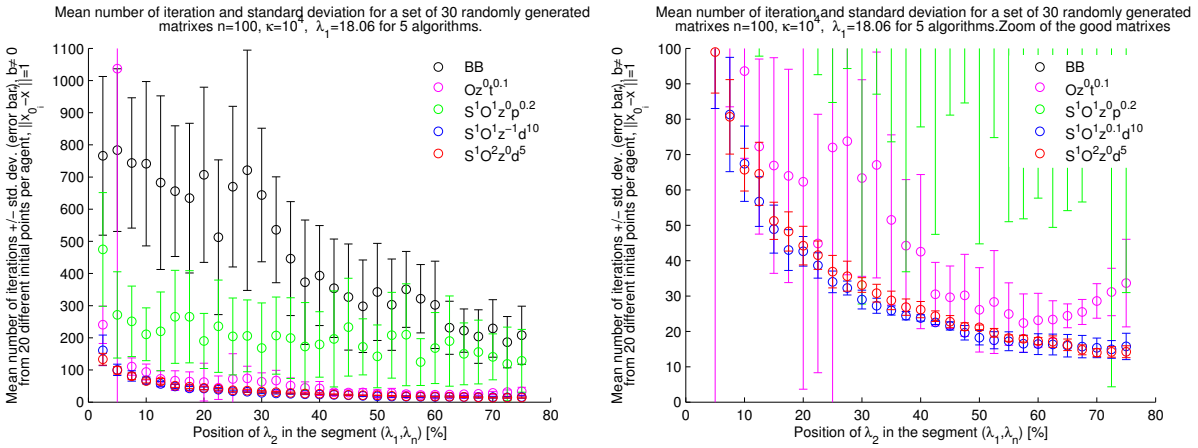

Fig. 10 Statistics of the algorithms BB, $Oz^0t^{0.1}$, $S^1O^1z^0p^{0.2}$, $S^1O^2z^0d^5$ and $S^1O^1z^{-1}d^{10}$ for a set of 30 random matrices of dimension 100, condition number 10^4, with the mean number of iterations and standard deviation calculated from 20 different initial points per agent, all of them located on a hypersphere such that $\|\mathbf{x}_{0_j,i} - \mathbf{x}^*\| = 1$. The right hand side $\mathbf{b} \in \mathbb{R}^{100}$ is randomly chosen. The second eigenvalue increases for each matrix. The figure to the right is a zoom of the one to the left.

In the section 5.1 we affirmed that the performance of the cooperative computation algorithms is strongly dependent on the location of the second lowest eigenvalue λ_2. In order to illustrate this affirmation, we repeat the tests presented in section 5.1 with the matrices $\overline{\mathbf{A}j}$, $\overline{\mathbf{B}j}$, $\overline{\mathbf{C}j}$ and $\overline{\mathbf{D}j}$, $j = 1, \ldots, 3$. These matrices have the same dimensions, condition number, largest and smallest eigenvalues as those of the matrices reported in Table 2. In addition, the four random orthonormal matrices $\mathbf{U} \in \mathbb{R}^{n\times n}$ generated to create the test matrices as $\mathbf{U}^\mathsf{T}\boldsymbol{\Lambda}\mathbf{U}$ are also the same used in the former section. The only difference lies in the fact that the matrices $\overline{\mathbf{A}j}$, $\overline{\mathbf{B}j}$, $\overline{\mathbf{C}j}$ and $\overline{\mathbf{D}j}$, $j = 1, \ldots, 3$ have the eigenvalue λ_2 chosen as $\lambda_2 = \lambda_1(0.9 + 0.1\kappa)$. The other eigenvalues are also randomly chosen as $\lambda_i \in (\lambda_2, \lambda_n),\ \forall i \in \{3, \ldots, n-1\}$, which implies that the eigenvalues λ_i, $i = 2, \cdots, n-1$, are grouped closer to the largest eigenvalue. The initial points and the right hand sides $\mathbf{b}$ used are also those used in the first series of experiments. The matrices used in the experiment presented in this section are also available on request. The results of this test are presented in Table 7.

Note that here the mean number of iterations of the CC algorithms are very much lower than those presented when λ_2 is randomly chosen. Except for the algorithm $S^1O^1z^0d^6$ with the matrix $\overline{\mathbf{D}1}$, in all the cases the cooperative computation algorithms present a mean number of iterations lower than those presented by the BB algorithm.

The following experiments, whose results are presented in figures 9 to 12, show tests for randomly generated matrices reported in Table 2 with different values of λ_2. In order to produce each figure, 30 matrices are randomly generated according to the following recipe: a smallest eigenvalue λ_1 is randomly chosen between 0 to 100, the largest eigenvalue is calculated as $\lambda_n = \kappa\lambda_1$, a random orthonormal matrix $\mathbf{U} \in \mathbb{R}^{n\times n}$ is generated; for each matrix, the second eigenvalue is chosen as $\lambda_2 = (1-\gamma)\lambda_1 + \gamma\lambda_n$, $\gamma \in [0.025, 0.75]$, the other eigenvalues are randomly chosen as $\lambda_i \in (\lambda_2, \lambda_n)$, $\forall i \in \{3, \ldots, n-1\}$, and the matrices are generated as $\mathbf{U}^\mathsf{T}\boldsymbol{\Lambda}\mathbf{U}$, where $\boldsymbol{\Lambda} = \text{diag}\ (\lambda_1, \ldots, \lambda_n)$. The right hand sides $\mathbf{b} \in \mathbb{R}^n$ and the initial points are the same used in Tables 3 and 7.

Note that all the algorithms improve their performances when λ_2 increases, but the CC algorithms, specially $S^1O^1z^{-1}d^{10}$ and $S^1O^2z^0d^5$, achieve much better results. Moreover, these algorithms present a performance which improves monotonically with the increasing of λ_2.

The reason for this behavior is explained in the following lemma:

Table 7 Mean number of iterations (upper number) and standard deviation (lower number) for each algorithm. Trajectories began at 20 different initial points per agent, all of them with norm one surrounding the solution point. $\mathbf{b} \neq \mathbf{0}$. The minimum number of iterations in each case is in boldface and blue (excepting the CG). The difference with Table 3 lies in the fact that here $\lambda_2 = 0.9\lambda_1 + 0.1\lambda_n$.

matrix	CG	BB	$Oz^0t^{0.1}$	$S^1O^1z^0p^{0.2}$	$S^1O^1z^0d^6$	$S^1O^2z^0p^{0.2}$	$S^1O^2z^0d^5$	$O^3z^0p^{0.2}$	$O^3z^0d^6$	$S^1O^1z^{-1}p^{0.1}$	$S^1O^1z^{-1}d^{10}$	$S^2O^1z^{-1}p^{0.1}$	$S^2O^1z^{-1}d^{10}$	$S^1O^2z^{-1}p^{0.1}$	$S^1O^2z^{-1}d^{10}$
$\overline{\mathbf{A1}}$	31.3	262.25	147.45	145.95	236.75	67.55	55.95	62.20	57.3	65.60	56.6	57.05	56.1	56.85	**47.1**
	0.47	78.09	22.73	51.94	128.99	11.73	8.31	11.68	9.88	10.79	7.10	12.05	8.57	10.51	6.82
$\overline{\mathbf{A2}}$	36	917.85	226.9	265.25	431.05	74.85	69.35	78.40	69.45	72.90	70.35	67.3	60.1	65.55	**59.15**
	0.32	293.91	47.01	142.36	309.72	10.17	5.81	8.59	7.89	13.54	12.40	10.55	7.50	12.28	5.18
$\overline{\mathbf{A3}}$	35.25	2016	746.70	307.95	719.85	77.90	70.80	64.65	64.40	73.30	71.15	61.15	52.4	59.65	**48.3**
	0.44	721.86	1278	210.33	421.15	12.71	8.42	9.97	10.00	10.36	13.60	15.5	5.3	15.51	5.04
$\overline{\mathbf{B1}}$	37.65	264.40	156.25	175.4	249.3	69.70	64.15	71	65.8	70.15	58.80	60.75	55.95	57.85	**54.95**
	0.48	72.93	24.11	104.17	111.39	12.69	5.94	8.44	9.92	14.98	10.47	8.07	9.28	7.92	6.32
$\overline{\mathbf{B2}}$	38.3	766	199.55	234.65	491.20	76.10	67.20	75.55	73.5	76.8	70.35	63.3	57.95	62	**57.9**
	0.47	341.15	57.16	113.67	365.05	8.61	5.60	11.87	8.1	10.52	9.77	9.62	10.62	6.80	7.15
$\overline{\mathbf{B3}}$	49	2692	510.05	352.4	514.85	84	88.55	86.35	84.4	87	77.05	69.55	74.85	77.45	**67.7**
	0.32	846.92	796.74	240.93	300.85	10.34	15.33	9.28	12.61	17.03	11.62	8.81	15.43	18.17	10.29
$\overline{\mathbf{C1}}$	37	245.85	147.95	158	234.30	68.10	59.90	69.20	69.55	70	58.75	55.8	54.2	55.4	**51.15**
	0	74.74	25.31	64.73	118.99	8.44	8.40	10.15	11.61	18.22	10.63	4.98	10.57	8.79	8.26
$\overline{\mathbf{C2}}$	44	831.95	252.05	199.7	419.15	81.15	75.55	84.05	81	76.8	70.30	67.65	64.25	64.7	**62.25**
	0	268.56	93.45	98.84	225.32	11.92	7.47	11.36	12.46	9.53	10.29	12.28	10.18	12.16	10.17
$\overline{\mathbf{C3}}$	49	2961	421.45	410.75	659.20	92.50	84.85	87.05	83.35	79.65	75.2	75.4	71.35	69.9	**66.80**
	0	971.57	425.07	245.02	431.26	14.49	9.69	13.34	13.83	13.71	15.46	11.17	11.9	8.67	8.53
$\overline{\mathbf{D1}}$	34.4	182.4	123.30	142.85	193.4	61.60	60.25	65.70	62.05	59	52.3	53.55	49.25	53.2	**48.6**
	0.50	56.87	23.33	58.88	118.46	8.95	6.34	6.30	8.46	9.61	8.71	7.05	5.33	9.91	6.10
$\overline{\mathbf{D2}}$	41	618.15	191.80	245.75	410.90	74.25	66.65	77.05	69.05	65.10	63.85	60.5	62.65	61.6	**52.8**
	0	158.45	65.56	131.15	269.82	9.03	7.56	10	8.55	9.48	10.22	9.31	9.12	10.08	6.87
$\overline{\mathbf{D3}}$	50.9	2443	479.55	323.25	646.75	87.85	87.30	101.9	87.95	86.15	75.95	78.50	73.75	73.6	**69.15**
	0.30	1061	600.25	226.07	401.44	11.37	8.74	12.07	9.13	12.43	11.28	10.01	12.37	10.27	11.20

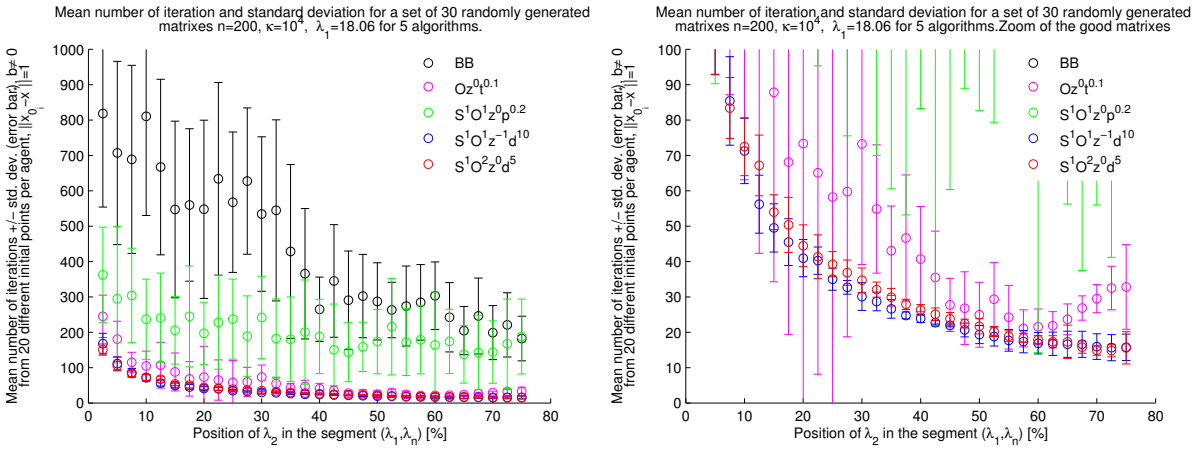

Fig. 11 Statistics of the algorithms BB, $Oz^0t^{0.1}$, $S^1O^1z^0p^{0.2}$, $S^1O^2z^0d^5$ and $S^1O^1z^{-1}d^{10}$ for a set of 30 random matrices of dimension 200, condition number 10^4, with the mean number of iterations and standard deviation calculated from 20 different initial points per agent, all of them located on a hypersphere such that $\|\mathbf{x}_{0_j,i} - \mathbf{x}^*\| = 1$. The right hand side $\mathbf{b} \in \mathbb{R}^{200}$ is randomly chosen. The second eigenvalue increases for each matrix. The figure to the right is a zoom of the one to the left.

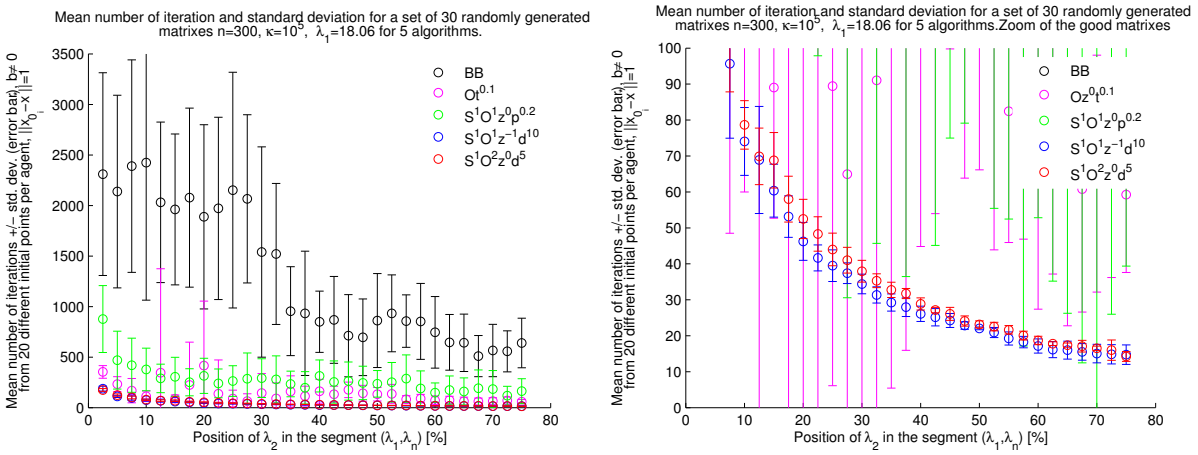

Fig. 12 Statistics of the algorithms BB, $Oz^0t^{0.1}$, $S^1O^1z^0p^{0.2}$, $S^1O^2z^0d^5$ and $S^1O^1z^{-1}d^{10}$ for a set of 30 random matrices of dimension 300, condition number 10^5, with the mean number of iterations and standard deviation calculated from 20 different initial points per agent, all of them located on a hypersphere such that $\|\mathbf{x}_{0_j,i} - \mathbf{x}^*\| = 1$. The right hand side $\mathbf{b} \in \mathbb{R}^{300}$ is randomly chosen. The second eigenvalue increases for each matrix. The figure to the right is a zoom of the one to the left.

Lemma 1 *The value of the scalar quadratic function $f(\hat{\mathbf{x}})$, where the point $\hat{\mathbf{x}}$ is obtained applying the optimal affine combination (10-11) or (13-14) between m agents decreases when the eigenvalues of the matrix $\mathbf{A}$ increase.*
(See the proof in the appendix 2.)

Note that lemma 1 indicates that the increase of *any one* of the eigenvalues produces the decrease of the value of $f(\hat{\mathbf{x}})$. Of course, the increase of λ_n increases the condition number, worsening the performances of the gradient descent algorithms. The increase of λ_1, keeping the other eigenvalues constant, decreases the condition number, improving the performance of the gradient descent algorithms as well as decreasing the A^{-1}-norm of the optimal affine combination. For this reason, the comparisons are made keeping the condition number constant, and the increase of the other eigenvalues decreases the A^{-1}-norm of the optimal affine combination without worsening the performance of the gradient descent methods.

Table 8 shows some of these comparisons. A matrix $\mathbf{A} \in \mathbb{R}^{100\times100}$ is generated as in the former sections with $\lambda_1 = 75.51$, $\lambda_{100} = 755140$, so the condition number is $\kappa = 10^4$. The algorithms were tested from 20 initial points per agent such that $\|\mathbf{x}_{0_j,i} - \mathbf{x}^*\| = 1$. The right hand side and the initial points were the same used by the matrix $\mathbf{B}2$ and $\overline{\mathbf{B}2}$. The first row shows the mean number of iterations used by the algorithms $S^2O^1z^{-1}d^{10}$, SD, OM, BB and CG. In the second row all the eigenvalues were decreased at 10% of their original values. Note that the number of iteration decreases in the gradient methods, but increases in the CC algorithms, because the A^{-1}-norm of the optimal affine combination calculated every 10 iterations increases. In the third row the eigenvalues were increased 10 times with respect to their original values. Note that, as the performance of the gradient descent methods is worse, the number of iteration of the CC algorithm does not decrease, although every affine combination presents a lower A^{-1}-norm, as lemma 1 indicates. In the fourth row the smallest and the largest eigenvalues have their original values, and all the other ones are given by $\lambda_i \geq 0.9\lambda_1 + 0.1\lambda_{100}$, $i = 2, \ldots, 99$. Note that here, as seen in Table 7, the performance of the CC algorithm improves whereas the gradient descent methods continue presenting almost the same number of iterations. This trend is more explicit with the increase of λ_2, as the fifth and sixth rows show.

Table 8 Mean number of iterations for each algorithm. Trajectories began at 20 different initial points per agent, all of them with norm one surrounding $\mathbf{x}^*$. $\mathbf{A} \in \mathbb{R}^{100\times 100}$, $\lambda_1 = 75.51$, $\lambda_{100} = 755140$, $\lambda_i \in (\lambda_1, \lambda_{100})$, $i = 2, \ldots, 99$ is randomly chosen, $\mathbf{b} \neq \mathbf{0}$.

	$S^2O^1z^{-1}d^{10}$	SD	OM	BB	CG
λ_i original $\forall i$	450	54225	53141	993.15	68.75
$\lambda_i \leftarrow 0.1\lambda_i$ original $\forall i$	472.05	42874	41681	878.3	66.35
$\lambda_i \leftarrow 10\lambda_i$ original $\forall i$	556.7	65602	64602	1128.9	71.1
$\lambda_2 \leftarrow 0.9\lambda_1 + 0.1\lambda_{100}$	64.35	54159	52917	850.25	44.3
$\lambda_2 \leftarrow 0.8\lambda_1 + 0.2\lambda_{100}$	36.9	54675	53204	804.95	30
$\lambda_2 \leftarrow 0.7\lambda_1 + 0.3\lambda_{100}$	31.05	54687	53144	638.4	26

Appendix 2: Properties of optimal affine combinations (Proofs of lemmas)

The proofs of the following properties are straightforward and they are omitted here.

Property 1 The point in the $\mathbf{x}$ space which has residue $\hat{\mathbf{r}} = \mathbf{R}\hat{\boldsymbol{\alpha}}$ is $\hat{\mathbf{x}} = \mathbf{X}\hat{\boldsymbol{\alpha}} = \mathbf{A}\hat{\mathbf{r}} - \mathbf{b}$. This point has error $\hat{\mathbf{e}} = \mathbf{E}\hat{\boldsymbol{\alpha}} = \hat{\mathbf{x}} - \mathbf{x}^*$.

Property 2 $\|\hat{\mathbf{r}}\|^2 = \frac{1}{\mathbf{1}_m^\intercal(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{1}_m}$

Property 3 For all $i,\ j \in \{1, \ldots, m\},\ i \neq j, \hat{\mathbf{r}}^\intercal(\mathbf{r}_i - \mathbf{r}_j) = 0$.

Proof of lemma 1:

The point which minimizes the quadratic function $f(\mathbf{x})$ is the same as that which minimizes $\|\mathbf{r}\|^2_{\mathbf{A}^{-1}} = 2f(\mathbf{x}) + \mathbf{b}^\intercal\mathbf{x}^*$.

Denoting $\mathbf{U} := [\mathbf{u}_1 \cdots \mathbf{u}_n]$ a matrix with orthonormal eigenvectors as columns, $\mathbf{A} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\intercal$ where $\boldsymbol{\Lambda}$ is a diagonal matrix with the eigenvalues; every residue of the agents can be represented in the base of these eigenvectors: $\mathbf{r}_i = \mathbf{U}\mathbf{d}_i,\ \forall i \in \{1, \ldots, m\}$. Denoting $\mathbf{D} := [\mathbf{d}_1 \cdots \mathbf{d}_m] \in \mathbb{R}^{n\times m}$, thus $\mathbf{R} = \mathbf{U}\mathbf{D} = \mathbf{A}\mathbf{X} - \mathbf{b}\mathbf{1}_m^\intercal$.

If $\hat{\mathbf{r}}$ is calculated as in (11):

$$\hat{\mathbf{r}} = \frac{\mathbf{R}(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{1}_m}{\mathbf{1}_m^\intercal(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{1}_m} = \frac{\mathbf{U}\mathbf{D}(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{1}_m}{\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{1}_m}$$

Hence:

$$\|\hat{\mathbf{r}}\|^2_{\mathbf{A}^{-1}} = \frac{\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{D}^\intercal\mathbf{U}^\intercal\mathbf{A}^{-1}\mathbf{U}\mathbf{D}(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{1}_m}{(\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{1}_m)^2} = \frac{\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{D}^\intercal\boldsymbol{\Lambda}^{-1}\mathbf{D}(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{1}_m}{(\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{1}_m)^2} \tag{17}$$

and denoting $\mathbf{c} := \mathbf{D}(\mathbf{D}^\intercal\mathbf{D})^{-1}\mathbf{1}_m \in \mathbb{R}^n$:

$$\|\hat{\mathbf{r}}\|^2_{\mathbf{A}^{-1}} = \frac{c_1^2/\lambda_1 + \cdots + c_n^2/\lambda_n}{(c_1^2 + \cdots + c_n^2)^2} \tag{18}$$

The application of this result when $\hat{\mathbf{r}} = \mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\mathbf{1}_m^\intercal$ and $\hat{\mathbf{x}}$ is calculated as in (14) is difficult. But this point $\hat{\mathbf{x}}$, which minimizes $f(\mathbf{X}\boldsymbol{\alpha})$ in the affine subspace defined as $\boldsymbol{\alpha}^\intercal\mathbf{1}_m = 1$ is the same which minimizes $\|\mathbf{R}\boldsymbol{\alpha}\|_{\mathbf{A}^{-1}}$ in the same space. Applying the same technique as that used to obtain (11) and (14) yields:

$$\hat{\mathbf{r}} = \mathbf{R}\hat{\boldsymbol{\alpha}} = \frac{\mathbf{R}(\mathbf{R}^\intercal\mathbf{A}^{-1}\mathbf{R})^{-1}\mathbf{1}_m}{\mathbf{1}_m^\intercal(\mathbf{R}^\intercal\mathbf{A}^{-1}\mathbf{R})^{-1}\mathbf{1}_m} \tag{19}$$

Of course, (19) cannot be used to calculate $\hat{\mathbf{r}}$ instead (14) because it is expressed in terms of $\mathbf{A}^{-1}$. From (19):

$$\begin{aligned}\|\hat{\mathbf{r}}\|^2_{\mathbf{A}^{-1}} &= \frac{\mathbf{1}_m^\intercal(\mathbf{R}^\intercal\mathbf{A}^{-1}\mathbf{R})^{-1}\mathbf{R}^\intercal\mathbf{A}^{-1}\mathbf{R}(\mathbf{R}^\intercal\mathbf{A}^{-1}\mathbf{R})^{-1}\mathbf{1}_m}{(\mathbf{1}_m^\intercal(\mathbf{R}^\intercal\mathbf{A}^{-1}\mathbf{R})^{-1}\mathbf{1}_m)^2} = \frac{\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\mathbf{U}^\intercal\mathbf{A}^{-1}\mathbf{U}\mathbf{D})^{-1}\mathbf{1}_m}{(\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\mathbf{U}^\intercal\mathbf{A}^{-1}\mathbf{U}\mathbf{D})^{-1}\mathbf{1}_m)^2} \\ &= \frac{\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\boldsymbol{\Lambda}^{-1}\mathbf{D})^{-1}\mathbf{1}_m}{(\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\boldsymbol{\Lambda}^{-1}\mathbf{D})^{-1}\mathbf{1}_m)^2} = \frac{1}{\mathbf{1}_m^\intercal(\mathbf{D}^\intercal\boldsymbol{\Lambda}^{-1}\mathbf{D})^{-1}\mathbf{1}_m} = \frac{\det(\mathbf{D}^\intercal\boldsymbol{\Lambda}^{-1}\mathbf{D})}{\mathbf{1}_m^\intercal\text{adj}\,(\mathbf{D}^\intercal\boldsymbol{\Lambda}^{-1}\mathbf{D})\mathbf{1}_m} = \frac{\det(\mathbf{D}^\intercal\boldsymbol{\Lambda}^{-1}\mathbf{D})}{d'_1/\lambda_1 + \cdots + d'_n/\lambda_n}\end{aligned} \tag{20}$$

where adj denotes the classical adjoint matrix and d'_i are coefficients which depend on the matrix $\mathbf{D}$. Note that the numerator of this expression is composed of terms of the type $d''_i/(\lambda_1^a\lambda_2^b\cdots\lambda_n^j)$, where d''_i are also coefficients which depend on $\mathbf{D}$ and $a + b + \cdots + j = m$, thus, each term of the determinant has a denominator of degree m, which implies that $\|\hat{\mathbf{r}}\|_{\mathbf{A}^{-1}}$ tends to zero when λ_i tends to infinity for all $i \in \{1, \ldots, n\}$.

Lemma 2 *If* $\mathbf{R} = [\mathbf{r}_1\ \mathbf{r}_2] \in \mathbb{R}^{n\times 2}$ *and* $\mathbf{r}_2 = q\mathbf{r}_1$ *with* $q \neq 1$, *then* $\hat{\mathbf{r}} := \mathbf{R}\hat{\boldsymbol{\alpha}} = \mathbf{0}$, *with* $\hat{\boldsymbol{\alpha}}$ *calculated as in (10).*

Proof:

$$\mathbf{R}^\intercal\mathbf{R} = \begin{bmatrix} 1 & q \\ q & q^2 \end{bmatrix} \|\mathbf{r}_1\|^2$$

$$\hat{\boldsymbol{\alpha}} = \begin{bmatrix} q^2 - q \\ 1 - q \end{bmatrix} \frac{1}{1+q^2-2q}$$

$$\hat{\mathbf{r}} = \mathbf{R}\hat{\boldsymbol{\alpha}} = \frac{\mathbf{r}_1(q^2-q) + q\mathbf{r}_1(1-q)}{1+q^2-2q} = \mathbf{0}$$

Lemma 3 *If $\mathbf{R} = [\mathbf{r}_1\ \mathbf{r}_2]$, the angle between $\mathbf{r}_1$ and $\mathbf{r}_2$ which maximizes $\|\hat{\mathbf{r}}\|$ is an angle θ such that $\cos\theta = \frac{\|\mathbf{r}_1\|}{\|\mathbf{r}_2\|}$ or $\cos\theta = \frac{\|\mathbf{r}_2\|}{\|\mathbf{r}_1\|}$.*

Proof: From property 2

$$\|\hat{\mathbf{r}}\|^2 = \frac{1}{\mathbf{1}_2^\intercal(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{1}_2} = \frac{\|\mathbf{r}_1\|^2\|\mathbf{r}_2\|^2 - (\mathbf{r}_1^\intercal\mathbf{r}_2)^2}{\|\mathbf{r}_1 - \mathbf{r}_2\|^2} = \frac{\|\mathbf{r}_1\|^2\|\mathbf{r}_2\|^2(1-\cos^2\theta)}{\|\mathbf{r}_1\|^2 + \|\mathbf{r}_2\|^2 - 2\|\mathbf{r}_1\|\|\mathbf{r}_2\|\cos\theta} \tag{21}$$

Note that (21) also proves lemma 2, because if $\mathbf{r}_1 \neq \mathbf{r}_2$ and $\theta \in \{0, \pi\}$, then $\|\hat{\mathbf{r}}\| = 0$. Differentiating (21) with respect to θ and equating to zero yields in the following second degree expression:

$$\cos^2\theta - \frac{\|\mathbf{r}_1\|^2 + \|\mathbf{r}_2\|^2}{\|\mathbf{r}_1\|\|\mathbf{r}_2\|}\cos\theta + 1 = 0$$

which has roots such that $\cos\theta \in \left\{\frac{\|\mathbf{r}_1\|}{\|\mathbf{r}_2\|}, \frac{\|\mathbf{r}_2\|}{\|\mathbf{r}_1\|}\right\}$, which means that a vector coincides with the projection of the other vector onto it. With this result, we can affirm:

$$\|\hat{\mathbf{r}}\|^2 \leq \frac{\|\mathbf{r}_1\|^2\|\mathbf{r}_2\|^2(1-\frac{\|\mathbf{r}_1\|^2}{\|\mathbf{r}_2\|^2})}{\|\mathbf{r}_1\|^2 + \|\mathbf{r}_2\|^2 - 2\|\mathbf{r}_1\|^2} = \|\mathbf{r}_1\|^2 \quad \textit{and} \quad \|\hat{\mathbf{r}}\|^2 \leq \frac{\|\mathbf{r}_1\|^2\|\mathbf{r}_2\|^2(1-\frac{\|\mathbf{r}_2\|^2}{\|\mathbf{r}_1\|^2})}{\|\mathbf{r}_1\|^2 + \|\mathbf{r}_2\|^2 - 2\|\mathbf{r}_2\|^2} = \|\mathbf{r}_2\|^2 \tag{22}$$

Lemma 4 *$\hat{\mathbf{r}} = \mathbf{R}\hat{\boldsymbol{\alpha}}$, with $\hat{\boldsymbol{\alpha}}$ calculated as in (13) is perpendicular to $\mathbf{x}_i - \mathbf{x}_j$ for all $i,\ j \in \{1, \cdots, m\},\ i \neq j$.*
Proof:

Denoting $\mathbf{v}_i = [0 \cdots \underbrace{1}_{i^{th}\ column} \cdots 0]^\intercal$:

$$\begin{aligned}(\mathbf{x}_i - \mathbf{x}_j)^\intercal\hat{\mathbf{r}} &= (\mathbf{v}_i - \mathbf{v}_j)^\intercal\mathbf{X}^\intercal\left[\frac{1-\mathbf{1}_p^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{b}}{\mathbf{1}_p^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_p}\mathbf{A}\mathbf{X}(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_p + \mathbf{A}\mathbf{X}(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{b} - \mathbf{b}\right] \\ &= (\mathbf{v}_i - \mathbf{v}_j)^\intercal\left[\frac{1-\mathbf{1}_p^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{b}}{\mathbf{1}_p^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_p}\mathbf{1}_p + \mathbf{X}^\intercal\mathbf{b} - \mathbf{X}^\intercal\mathbf{b}\right] = \frac{1-\mathbf{1}_p^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{b}}{\mathbf{1}_p^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_p}(\mathbf{v}_i - \mathbf{v}_j)^\intercal\mathbf{1}_p = 0\end{aligned}$$

Lemma 5 *The point $\hat{\mathbf{x}}$ with residue $\hat{\mathbf{r}}$, obtained by optimal affine combination (10-11) applied to $\mathbf{R} = [\mathbf{r}_k\ \ (I - \rho_k\mathbf{A})\mathbf{r}_k]$, does not depend on the choice of ρ_k. Moreover, $\hat{\mathbf{x}} = \mathbf{x}_k - \rho_k\mathbf{r}_k$ where $\rho_k = \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k}$, that is, an orthomin step applied from $\mathbf{x}_k$.*

Proof:

$\mathbf{R} = [\mathbf{r}_k\ \ (I - \rho_k\mathbf{A})\mathbf{r}_k] = \mathbf{A}\mathbf{X} - \mathbf{b}\mathbf{1}_2^\intercal$ *where* $\mathbf{X} = [\mathbf{x}_k\ \ \mathbf{x}_k - \rho_k\mathbf{r}_k]$

$$\mathbf{R}^\intercal\mathbf{R} = \begin{bmatrix}\|\mathbf{r}_k\|^2 & \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k \\ \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k & \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})^2\mathbf{r}_k\end{bmatrix}$$

$$\hat{\boldsymbol{\alpha}} = \frac{(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{1}_2}{\mathbf{1}_2^\intercal(\mathbf{R}^\intercal\mathbf{R})^{-1}\mathbf{1}_2} = \frac{\begin{bmatrix}\mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})^2\mathbf{r}_k - \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k \\ \|\mathbf{r}_k\|^2 - \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k\end{bmatrix}}{\|\mathbf{r}_k\|^2 + \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})^2\mathbf{r}_k - 2\mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k}$$

$$\begin{aligned}\hat{\mathbf{r}} = \mathbf{R}\hat{\boldsymbol{\alpha}} &= \frac{\mathbf{r}_k[\mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})^2\mathbf{r}_k - \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k] + (I-\rho_k\mathbf{A})\mathbf{r}_k[\|\mathbf{r}_k\|^2 - \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k]}{\|\mathbf{r}_k\|^2 + \mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})^2\mathbf{r}_k - 2\mathbf{r}_k^\intercal(I-\rho_k\mathbf{A})\mathbf{r}_k} \\ &= \frac{\mathbf{r}_k(-\rho_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k + \rho_k^2\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k) + (I-\rho_k\mathbf{A})\mathbf{r}_k(\rho_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k)}{\rho_k^2\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k} \\ &= \frac{\rho_k^2\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k\mathbf{r}_k - \rho_k^2\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k\mathbf{A}\mathbf{r}_k}{\rho_k^2\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k} = \mathbf{r}_k - \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k}\mathbf{A}\mathbf{r}_k = \left(I - \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k}\mathbf{A}\right)\mathbf{r}_k\end{aligned}$$

Therefore, by property 1:

$$\hat{\mathbf{x}} = \mathbf{X}\hat{\boldsymbol{\alpha}} = \mathbf{A}^{-1}(\hat{\mathbf{r}} + \mathbf{b}) = \mathbf{A}^{-1}\left(\mathbf{r}_k + \mathbf{b} - \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k}\mathbf{A}\mathbf{r}_k\right) = \mathbf{x}_k - \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\intercal\mathbf{A}^2\mathbf{r}_k}\mathbf{r}_k$$

Lemma 6 *The point $\hat{\mathbf{x}}$ with residue $\hat{\mathbf{r}}$, obtained by optimal affine combination (13-14) applied to $\mathbf{X} = [\mathbf{x}_k\ \ \mathbf{x}_k - \rho_k\mathbf{r}_k]$, does not depend on the choice of ρ_k. Moreover, $\hat{\mathbf{x}} = \mathbf{x}_k - \rho_k\mathbf{r}_k$, where $\rho_k = \frac{\mathbf{r}_k^\intercal\mathbf{r}_k}{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k}$, that is, a steepest descent step applied from $\mathbf{x}_k$.*

Proof:

$$\mathbf{X} = [\mathbf{x}_k\ \mathbf{x}_k - \rho_k\mathbf{r}_k]$$

$$\mathbf{X}^\intercal\mathbf{A}\mathbf{X} = \begin{bmatrix}\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k & \mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k - \rho_k\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k \\ \mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k - \rho_k\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k & \mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k - 2\rho_k\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k + \rho_k^2\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k\end{bmatrix}$$

$$(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1} = \frac{\begin{bmatrix}\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k - 2\rho_k\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k + \rho_k^2\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k & -\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k + \rho_k\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k \\ -\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k + \rho_k\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k & \mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\end{bmatrix}}{\rho_k^2(\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k - (\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k)^2)}$$

$$\mathbf{1}_2^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_2 = \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k}{\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k - (\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k)^2}$$

$$\mathbf{1}_2^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{b} = \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k\mathbf{x}_k^\intercal\mathbf{b} - \mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k\mathbf{r}_k^\intercal\mathbf{b}}{\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k - (\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k)^2}$$

$$\frac{1-\mathbf{1}_2^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{b}}{\mathbf{1}_2^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_2}\mathbf{X}^\intercal(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_2 = \frac{\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k - (\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k)^2 - \mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k\mathbf{x}_k^\intercal\mathbf{b} + \mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k\mathbf{r}_k^\intercal\mathbf{b})(\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k\ \mathbf{x}_k - \mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k\ \mathbf{r}_k)}{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k(\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k - (\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k)^2)}$$

$$\mathbf{X}(\mathbf{X}^\intercal\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\intercal\mathbf{b} = \frac{\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k\mathbf{x}_k^\intercal\mathbf{b} - \mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{b}}{\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k - (\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k)^2}\ \mathbf{x}_k - \frac{\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k\mathbf{x}_k^\intercal\mathbf{b} - \mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{b}}{\mathbf{x}_k^\intercal\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\intercal\mathbf{A}\mathbf{r}_k - (\mathbf{x}_k^\intercal\mathbf{A}\mathbf{r}_k)^2}\ \mathbf{r}_k$$

$$\hat{\mathbf{x}} = \mathbf{x}_k + \frac{-\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k+(\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k)^3+(\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k-(\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k)^2)\mathbf{r}_k^\mathsf{T}\mathbf{b}}{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k(\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{x}_k\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k-(\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k)^2)}\ \mathbf{r}_k$$
$$= \mathbf{x}_k + \frac{\mathbf{r}_k^\mathsf{T}\mathbf{b}-\mathbf{x}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k}{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k}\mathbf{r}_k \ = \ \mathbf{x}_k + \frac{\mathbf{r}_k^\mathsf{T}(\mathbf{b}-\mathbf{A}\mathbf{x}_k)}{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k}\mathbf{r}_k = \mathbf{x}_k - \frac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{r}_k^\mathsf{T}\mathbf{A}\mathbf{r}_k}\mathbf{r}_k$$

Lemma 7 *If* $\mathbf{E} = [\mathbf{e}_1\ \mathbf{e}_2] \in \mathbb{R}^{n\times 2}$ *and* $\mathbf{e}_2 = q\mathbf{e}_1$ *with* $q \neq 1$ *then* $\hat{\mathbf{e}} := \mathbf{E}\hat{\boldsymbol{\alpha}} = \mathbf{0}$*, with* $\hat{\boldsymbol{\alpha}}$ *calculated as in (13), which implies that* $\hat{\mathbf{x}} = \mathbf{X}\hat{\boldsymbol{\alpha}} = \mathbf{x}^*$.

Proof:

$\mathbf{e}_2 = q\mathbf{e}_1 \ \Rightarrow\ \mathbf{x}_2 - \mathbf{x}^* = q(\mathbf{x}_1 - \mathbf{x}^*) \ \Rightarrow\ \mathbf{x}_2 = q\mathbf{x}_1 + (1-q)\mathbf{x}^*$

$\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2] = [\mathbf{x}_1\ q\mathbf{x}_1 + (1-q)\mathbf{x}^*]$

$$\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 & q\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 + (1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^* \\ q\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 + (1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^* & q^2\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 + 2q(1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^* + (1-q)^2\mathbf{x}^{*\mathsf{T}}\mathbf{A}\mathbf{x}^* \end{bmatrix}$$

$$(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X})^{-1} = \frac{\begin{bmatrix} q^2\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 + 2q(1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^* + (1-q)^2\mathbf{x}^{*\mathsf{T}}\mathbf{A}\mathbf{x}^* & -q\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 - (1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^* \\ -q\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 - (1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^* & \mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 \end{bmatrix}}{(1-q)^2(\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{A}\mathbf{x}^* - (\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^*)^2)}$$

$$\mathbf{1}_2^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_2 = \frac{\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1 - 2\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^* + \mathbf{x}^{*\mathsf{T}}\mathbf{A}\mathbf{x}^*}{\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{A}\mathbf{x}^* - (\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}^*)^2}$$

$$(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{b} = \frac{\begin{bmatrix} q(1-q)(\mathbf{x}_1^\mathsf{T}\mathbf{b})^2 + (1-q)^2\mathbf{x}^{*\mathsf{T}}\mathbf{b}\mathbf{x}_1^\mathsf{T}\mathbf{b} - q(1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b} - (1-q)^2\mathbf{x}_1^\mathsf{T}\mathbf{b}\mathbf{x}^{*\mathsf{T}}\mathbf{b} \\ -(1-q)(\mathbf{x}_1^\mathsf{T}\mathbf{b})^2 + (1-q)\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b} \end{bmatrix}}{(1-q)^2(\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b} - (\mathbf{x}_1^\mathsf{T}\mathbf{b})^2)}$$
$$= \frac{\begin{bmatrix} q(1-q)((\mathbf{x}_1^\mathsf{T}\mathbf{b})^2 - \mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b}) \\ (1-q)(-(\mathbf{x}_1^\mathsf{T}\mathbf{b})^2 + \mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b}) \end{bmatrix}}{(1-q)^2(\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b} - (\mathbf{x}_1^\mathsf{T}\mathbf{b})^2)} = \frac{((\mathbf{x}_1^\mathsf{T}\mathbf{b})^2 - \mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b})\begin{bmatrix} q \\ -1 \end{bmatrix}}{(1-q)(\mathbf{x}_1^\mathsf{T}\mathbf{A}\mathbf{x}_1\mathbf{x}^{*\mathsf{T}}\mathbf{b} - (\mathbf{x}_1^\mathsf{T}\mathbf{b})^2)} = \frac{[q\ -1]^\mathsf{T}}{q-1}$$

where the fact that $\mathbf{A}\mathbf{x}^* = \mathbf{b}$ *has been used.*

$$\mathbf{1}_2^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{b} = \frac{q-1}{q-1} = 1$$

$$\frac{1-\mathbf{1}_2^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{b}}{\mathbf{1}_2^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X})^{-1}\mathbf{1}_2} = 0$$

$$\hat{\boldsymbol{\alpha}} = (\mathbf{X}^\mathsf{T}\mathbf{A}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{b} = \frac{[q\ -1]^\mathsf{T}}{q-1}$$

$$\hat{\mathbf{e}} = \mathbf{E}\hat{\boldsymbol{\alpha}} = [\mathbf{e}_1\ \ q\mathbf{e}_1]\hat{\boldsymbol{\alpha}} = \mathbf{e}_1[1\ \ q]\frac{\begin{bmatrix} q \\ -1 \end{bmatrix}}{q-1} = \mathbf{0}$$

Therefore:

$\hat{\mathbf{e}} = \mathbf{E}\hat{\boldsymbol{\alpha}} = [\mathbf{x}_1 - \mathbf{x}^*\ \mathbf{x}_2 - \mathbf{x}^*]\hat{\boldsymbol{\alpha}} = (\mathbf{X} - \mathbf{x}^*\mathbf{1}_2^\mathsf{T})\hat{\boldsymbol{\alpha}} = \hat{\mathbf{x}} - \mathbf{x}^* = \mathbf{0}$

$\Rightarrow\ \hat{\mathbf{x}} = \mathbf{x}^*$

References

1. Sparse matrix collection. Available at http://www.cise.ufl.edu/research/sparse/matrices/ (2009)
2. Abkowicz, A., Brezinski, C.: Acceleration properties of the hybrid procedure for solving linear systems. Applicationes Mathematicae **4**(23), 417–432 (1996)
3. Akaike, H.: On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method. Ann. Inst. Stat. Math. **11**, 1–16 (1959)
4. Ascher, U., van den Doel, K., Huang, H., Svaiter, B.: On fast integration to steady state and earlier times. Mathematical Modelling and Numerical Analysis **43**, 689–708 (2009)
5. Barzilai, J., Borwein, J.M.: Two point step size gradient methods. IMA J. Numer. Anal. **49**, 141–148 (1988)
6. Bhaya, A., Bliman, P.A., Pazos, F.: Control-theoretic design of iterative methods for symmetric linear systems of equations. In: Proc. of the 48th IEEE Conference on Decision and Control, pp. 115–120. Shanghai, China (2009)
7. Bhaya, A., Bliman, P.A., Pazos, F.: Cooperative parallel asynchronous computation of the solution of symmetric linear systems. In: Proc. of the 49th IEEE Conference on Decision and Control. Atlanta, USA (2010)
8. Bhaya, A., Kaszkurewicz, E.: Iterative methods as dynamical systems with feedback control. In: Proc. 42nd IEEE Conference on Decision and Control, pp. 2374–2380. Maui, Hawaii, USA (2003)
9. Bhaya, A., Kaszkurewicz, E.: Control perspectives on numerical algorithms and matrix problems. Advances in Control. SIAM, Philadelphia (2006)
10. Bhaya, A., Kaszkurewicz, E.: A control-theoretic approach to the design of zero finding numerical methods. IEEE Transactions on Automatic Control **52**(6), 1014–1026 (2007)
11. Brezinski, C.: Multiparameter descent methods. Linear Algebra and its Applications **296**, 113–141 (1999)
12. Brezinski, C.: Acceleration procedures for matrix iterative methods. Numerical Algorithms **25**, 63–73 (2000)
13. Brezinski, C.: Block descent methods and hybrid procedures for linear systems. Numerical Agorithms **29**, 21–32 (2002)
14. Brezinski, C., Chehab, J.P.: Nonlinear hybrid procedures and fixed point iterations. Numer. Funct. Anal. Optimization **19**, 465–487 (1998)
15. Brezinski, C., Chehab, J.P.: Multiparameter iterative schemes for the solution of systems of linear and nonlinear equations. SIAM J. Sci. Comp. **20**(6), 2140–2159 (1999)
16. Brezinski, C., Redivo-Zaglia, M.: Hibrid procedures for solving linear systems. Numerische Mathematik (67), 1–19 (1994)
17. Dai, Y.H., Liao, L.Z.: R-linear convergence of the Barzilai and Borwein gradient method. IMA Journal of numerical analysis **22**, 1–10 (2002)
18. van den Doel, K., Ascher, U.: The chaotic nature of faster gradient descent methods. Journal Sci. Comput. **51**, 560–581 (2012). DOI: 10.1007/s10915-011-9521-3

19. Fletcher, R.: A limited memory steepest descent method. Technical Report, Edinburgh Research Group in Optimization ERGO 09-014, Dept. of Mathematics, University of Dundee, Dundee DD1 4HN, Scotland, UK (2009)
20. Friedlander, A., Martínez, J.M., Molina, B., Raydan, M.: Gradient method with retards and generalizations. SIAM Journal on Numerical Analysis **36**(1), 275–289 (1999)
21. Greenbaum, A.: Iterative methods for solving linear systems. SIAM, Philadelphia (1997)
22. Nedic, A., Ozdaglar, A.: Convex Optimization in Signal Processing and Communications, chap. Cooperative Distributed Multi-agent Optimization, pp. 340–386. Cambridge University Press (2010)
23. Pronzato, L., Wynn, H.P., Zhigljavsky, A.A.: Dynamical Search: Applications of Dynamical Systems in Search and Optimization. Chapman and Hall/CRC, Boca Raton (2000)
24. G. Strang, *Linear algebra and its applications*, Harcourt Brace Jovanovich, San Diego, California, 1988.
25. Zhou, B., Gao, L., Dai, Y.H.: Gradient methods with adaptive step-sizes. Computational optimization and applications **35**, 69–86 (2006)