Computational Acceleration of Projection Algorithms for the Linear Best Approximation Problem

Yair Censor
Department of Mathematics, University of Haifa,
Mt. Carmel, Haifa 31905, Israel
(yair@math.haifa.ac.il)

May 2, 2005. Revised: September 18, 2005.

Abstract

This is an experimental computational account of projection algorithms for the linear best approximation problem. We focus on the sequential and simultaneous versions of Dykstra's algorithm and the Halpern-Lions-Wittmann-Bauschke algorithm for the best approximation problem from a point to the intersection of closed convex sets in the Euclidean space. These algorithms employ different iterative approaches to reach the same goal but no mathematical connection has yet been found between their algorithmic schemes. We compare these algorithms on linear best approximation test problems that we generate so that the solution will be known a priori and enable us to assess the relative computational merits of these algorithms. For the simultaneous versions we present a new component-averaging variant that substantially accelerates their initial behavior for sparse systems.

1 Introduction

The *convex feasibility problem* (CFP) is to find a point (any point) in the nonempty intersection $C := \cap_{i=1}^{m} C_i \neq \emptyset$ of a family of closed convex subsets $C_i \subseteq R^n$, $1 \leq i \leq m$, of the n-dimensional Euclidean space, see, e.g., Censor

and Zenios [19, Chapter 5]. It is a fundamental problem in many areas of mathematics and the physical sciences, see, e.g., Combettes [21, 24] and references therein. It has been used to model significant real-world problems in image reconstruction from projections, see, e.g., Herman [39], in radiation therapy treatment planning, see, e.g., Censor, Altschuler and Powlis [14] and Censor [13], and in crystallography, see Marks, Sinkler and Landree [48], to name but a few, and has been used under additional names such as *set theoretic estimation* or the *feasible set approach*. The convex sets $\{C_i\}_{i=1}^m$ represent mathematical constraints obtained from the modeling of the real-world problem. A common approach to such problems is to use *projection algorithms*, see, e.g., Bauschke and Borwein [4], which employ *orthogonal projections* (i.e., nearest point mappings) onto the individual sets C_i. The orthogonal projection $P_\Omega(z)$ of a point $z \in R^n$ onto a closed convex set $\Omega \subseteq R^n$ is a point of Ω defined by

$$P_\Omega(z) := \text{argmin}\{\| z - x \|_2 | \; x \in \Omega\}, \tag{1}$$

where $\| \cdot \|_2$ is the Euclidean norm in R^n. Frequently a *relaxation parameter* is introduced so that

$$P_{\Omega,\lambda}(z) := (1 - \lambda)z + \lambda P_\Omega(z) \tag{2}$$

is the *relaxed projection* of z onto Ω with relaxation λ.

The *best approximation problem* (BAP) is to find the projection of a given point $y \in R^n$ onto the nonempty intersection $C := \cap_{i=1}^m C_i \neq \emptyset$ of a family of closed convex subsets $C_i \subseteq R^n$, $1 \leq i \leq m$, see, e.g., Deutsch's book [27]. While in the convex feasibility problem any point in the intersection is an acceptable solution to the real-world problem, the best approximation formulation is usually appropriate if some point $y \in R^n$ has been obtained from modeling and computational efforts that initially did not take into account the constraints represented by the sets $\{C_i\}_{i=1}^m$ and now one wishes to incorporate them by seeking a point in the intersection of the convex sets which is closest to the point y.

For both problem classes, projection algorithms employ projections onto the individual convex sets in order to reach the required point in the intersection. They employ projections onto the individual sets in various ways. They may use different kinds of projections and, sometimes, even use different projections within the same algorithm. They serve to solve a variety of problems which are either of the feasibility or the optimization types. They

have different algorithmic structures, of which some are particularly suitable for parallel computing, and they demonstrate nice convergence properties and/or good initial behavior patterns. Iterative projection algorithms for the BAP, such as the algorithms of Dykstra, see, e.g., [9] and references therein, of Haugazeau, see, e.g., [5], of Halpern-Lions-Wittmann-Bauschke, see, e.g., [1] and others, are more complicated than algorithms for the CFP because they must have, in their iterative steps, some built-in "memory" mechanism to remember the original point whose projection is sought after. Is is clear that the, by now classical, alternating projections method, proposed by von Neumann [57] (see Halperin [35], Bauschke and Borwein [2], and also Kopecká and Reich [46] for a recent geometric approach), although very useful for the CFP and for the case when all the sets C_i are subspaces (see, for instance, Deutsch [27] and Bruck and Reich [11]), is not appropriate, in general, for the BAP. This can be easily seen from the example where $C_1 = \{x \in R^2 \mid x_2 \leq 0\}$, $C_2 = \{x \in R^2 \mid x_1 + x_2 \leq 0\}$ and $y = (2, 1)$. Obviously, $P_{C_2}P_{C_1}(y) \neq P_{C_1 \cap C_2}(y)$.

In this paper we focus on the sequential and simultaneous versions of Dykstra's algorithm and the Halpern-Lions-Wittmann-Bauschke (HLWB) algorithm for the BAP. These algorithms employ different iterative approaches to the BAP, and, to the best of our knowledge, no mathematical connection has yet been found between their algorithmic schemes. We briefly review the algorithms and give references to earlier theoretical work, in some of which we were involved, and then we present some toy computational results that compare the initial experimental behavior of the algorithms. Our results, as limited in scope as they are, show that the HLWB algorithms seem to be advantageous over Dykstra's algorithms. If future work with more comprehensive computations further substantiates this it might raise the question as to why have the latter been preferred over the HLWB algorithm on various occasions. These occasions include, but are not limited to, constrained least squares matrix problems, see, e.g., Birgin and Raydan [7], Escalante and Raydan [32], Higham [40], Mendoza, Raydan and Tarazaga [50] and Raydan and Tarazaga [51].

In addition to the four algorithms we propose a new component-averaging (CAV) variation for the simultaneous algorithms. This CAV idea has been previously shown to significantly accelerate simultaneous projection algorithms for systems of sparse linear equations in real-world problems of image reconstruction from projections (see Censor, Gordon and Gordon [16, 17]) and has been extensively studied theoretically (see Censor and Elfving [15]

and Jiang and Wang [45]). Here we try the CAV idea out on the simultaneous Dykstra and the simultaneous HLWB algorithms experimentally and discover that it works. Mathematical validation and higher dimensional computational work of the combination of the CAV idea with the simultaneous Dykstra and the simultaneous HLWB algorithms will be reported in future work. To evaluate the performance of the algorithms in our experiments we need test (best approximation) problems whose solutions will be known to us beforehand. Since we work with the linear BAP we do so, for linear constraints sets, by solving a linear programming problem over the constraints and then moving away from the constraints intersection along the normal of the linear objective function.

The simultaneous algorithms are *inherently parallel* schemes in that their mathematical formulations are parallel. We used this term in Butnariu, Censor and Reich [12, p. vii] to contrast such algorithms with others which are sequential in their mathematical formulation but can, sometimes, be implemented in a parallel fashion based on appropriate model decomposition (i.e., depending on the structure of the underlying problem). Being inherently parallel, these algorithms enable flexibility in the actual manner of implementation on a parallel machine. The paper is laid out as follows: Dykstra algorithms and the HLWB algorithms are presented in Sections 2 and 3, respectively. The application of the CAV idea to the simultaneous Dykstra and the simultaneous HLWB algorithms is described in Section 4. The test-problems generation process is described in Section 5 and representative numerical results and conclusions are brought in Section 6. Our experiments are of a preliminary nature and many aspects still need to be studied experimentally with regard to these algorithms.

2 Sequential and simultaneous Dykstra algorithms

The *sequential Dykstra algorithm* is an iterative procedure which (asymptotically) finds the nearest point projection (also called the orthogonal projection) of a given point onto the intersection of a given finite family of closed convex sets. It iterates by passing sequentially over the individual sets and projecting onto each one a *deflected* version of the previous iterate. The algorithm was first proposed and analyzed by Dykstra [30] and rediscovered by

Han [37]. Published work on Dykstra's algorithm includes: Boyle and Dykstra [8], Gaffke and Mathar [33], Iusem and De Pierro [44], Crombez [26], Combettes [22], Bauschke and Borwein [3], Deutsch and Hundal [28], Hundal and Deutsch [42], Han and Lou [38], Escalante and Raydan [32] and Birgin and Raydan [7]. See also Robertson, Wright and Dykstra [55], and Dykstra [31]. Censor and Reich [18] proposed a synthesis of Dykstra's algorithm with Bregman distances and obtained a new algorithm that solves the BAP with Bregman projections. However, they established convergence of the resulting Dykstra algorithm with Bregman projections only when the constraints sets are half-spaces. Shortly thereafter Bauschke and Lewis [6] provided the first proof for general closed convex constraints sets and discovered the close relationship between the Dykstra algorithm with Bregman projections and the very general and powerful algorithmic framework of Tseng [56], namely the *dual-block-coordinate-ascent* (DBCA) methods. A completely different point of departure which leads exactly to the same Dykstra algorithm with Bregman projections was studied by Bregman, Censor and Reich [9]. Combettes [25] proposed a surrogate constraint splitting algorithm that he compared with Dykstra's algorithm. In the sequel we denote by P_i the orthogonal projection P_{C_i} onto C_i.

Algorithm 1 ***The sequential Dykstra algorithm.***

1. Initialization: *Let $x^0 = y$ be the given point whose projection $P_C(y)$ onto $C := \cap_{i=1}^m C_i \neq \emptyset$ is sought after by the BAP. Initialize the auxiliary vectors $u^{0,i} := 0$, for all $i = 1, 2, \ldots, m$.*

2. Iterative step: *Choose a control index $i(k) \in \{1, 2, \ldots, m\}$ from an almost cyclic control sequence (see below). Given the current iterate x^k and the current auxiliary vectors $\{u^{k,i}\}_{i=1}^m$,*

2.1 Calculate the deflected vector:

$$z^{k+1} = x^k + u^{k,i(k)} \tag{3}$$

2.2 Project the deflected vector to obtain the next iterate:

$$x^{k+1} = P_{i(k)}(z^{k+1}) \tag{4}$$

2.3 Update the auxiliary vectors:

$$u^{k+1,i} = \begin{cases} z^{k+1} - x^{k+1}, & \text{if } i = i(k), \\ u^{k,i}, & \text{if } i \neq i(k). \end{cases} \tag{5}$$

3. Control sequence: *The indices* $\{i(k)\}_{k=0}^{\infty}$ *form an almost cyclic sequence, i.e., there exists an integer* $\rho \geq m$ *such that, for all* $r \geq 0$, $\{1, 2, \ldots, m\} \subseteq \{i_{r+1}, \ldots, i_{r+\rho}\}$.

Almost cyclic is sometimes called quasi-periodic and a commonly used almost cyclic control is the cyclic control for which $i(k) = k \bmod m + 1$, for all $k \geq 0$. The *simultaneous Dykstra's algorithm*, presented next, was developed and studied by Iusem and De Pierro [44]. The actions that it takes (simultaneously) with respect to each set are similar to those performed by the sequential algorithm and after such simultaneous deflections and projections it takes a convex combination of the intermediate points as the next iterate.

Algorithm 2 ***The simultaneous Dykstra algorithm.***

1. Initialization: *Let* $x^0 = y$ *be the given point whose projection* $P_C(y)$ *onto* $C := \cap_{i=1}^{m} C_i \neq \emptyset$ *is sought after by the BAP. Initialize the auxiliary vectors* $u^{0,i} := 0$, *for all* $i = 1, 2, \ldots, m$, *and let* $\{w_i\}_{i=1}^{m}$ *be real positive numbers (called weights) such that* $\sum_{i=1}^{m} w_i = 1$.

2. Iterative step: *Given the current iterate* x^k *and the current auxiliary vectors* $\{u^{k,i}\}_{i=1}^{m}$,

2.1 Calculate the deflected vectors, for $i = 1, 2, \ldots, m$,

$$z^{k+1,i} = x^k + u^{k,i}. \tag{6}$$

2.2 Project the deflected vectors to obtain the intermediate vectors, for $i = 1, 2, \ldots, m$,

$$x^{k+1,i} = P_i(z^{k+1,i}). \tag{7}$$

2.3 Calculate the next iterate x^{k+1} *as the convex combination of the intermediate vectors*

$$x^{k+1} = \sum_{i=1}^{m} w_i x^{k+1,i}. \tag{8}$$

2.3 Update the auxiliary vectors, for $i = 1, 2, \ldots, m$,

$$u^{k+1,i} = z^{k+1,i} - x^{k+1,i}. \tag{9}$$

In the linear case, where the sets C_i are half-spaces as in (17), the sequential and simultaneous algorithms of Dykstra are identical with the sequential and simultaneous Hildreth's algorithms, see Hildreth [41] and Lent and Censor [47] and Iusem and De Pierro [43], respectively.

3 Sequential and simultaneous Halpern-Lions-Wittmann-Bauschke algorithms

The *sequential Halpern-Lions-Wittmann-Bauschke (HLWB) algorithm* can be found in Bauschke [1]. Its origins go back to Wittmann [58] and earlier to Halpern [36] and Browder [10]. A concise historical review of the literature on this algorithm appears in Deutsch and Yamada [29] and related studies of this algorithm appear in works of Reich [52], [53] and [54]. The *simultaneous HLWB algorithm* is due to Combettes [23]. Further extensions appear in Deutsch and Yamada [29]. We need the following definition.

Definition 3 *A sequence* $\{\sigma_k\}_{k=0}^{\infty}$ *of real numbers* $0 \leq \sigma_k < 1$ *will be called a* `steering sequence` *if it satisfies the following conditions:*

$$\lim_{k\to\infty} \sigma_k = 0, \tag{10}$$

$$\sum_{k=0}^{\infty} \sigma_k = +\infty, \tag{11}$$

$$\sum_{k=0}^{\infty} \mid \sigma_k - \sigma_{k+m} \mid < +\infty. \tag{12}$$

A discussion of these conditions can be found in [1]. The sequential and simultaneous HLWB algorithms employ the parameters of a steering sequence to "force" (steer) the iterates towards the solution of the BAP.

Algorithm 4 ***The sequential Halpern-Lions-Wittmann-Bauschke algorithm.***

1. Initialization: *Let* $x^0 \in R^n$ *be an arbitrary starting point (equal to or different from the given point* y *whose projection* $P_C(y)$ *onto* $C := \cap_{i=1}^m C_i \neq \emptyset$ *is sought after by the BAP). Let* $\{\sigma_k\}_{k=0}^{\infty}$ *be a user-chosen steering sequence.*

2. Iterative step: *Choose the next control index* $i(k) \in \{1, 2, \ldots, m\}$ *(see below) and take the steering parameter* σ_k *from the chosen steering sequence. Given the current iterate* x^k*, calculate the next iterate by*

$$x^{k+1} = \sigma_k y + (1 - \sigma_k) P_{i(k)}(x^k). \tag{13}$$

3. Control sequence: *The indices* $\{i(k)\}_{k=0}^{\infty}$ *form a cyclic sequence, i.e.,* $i(k) = k \bmod m + 1$.

In every iterative step, this algorithm takes as the next iterate a point on the line segment connecting the given point y with the projection of the current iterate onto one of the constraint sets. The property (10) gradually reduces the weight of the given point y in the convex combination, as iterations proceed. The simultaneous version of the algorithm is presented next. The actions that it takes (simultaneously) with respect to each set are similar to those performed by the sequential algorithm and after such simultaneous projections are performed it takes a convex combination of the intermediate points as the next iterate.

Algorithm 5 ***The simultaneous Halpern-Lions-Wittmann-Bauschke algorithm.***

1. Initialization: *Let $x^0 = y$ be the given point y whose projection $P_C(y)$ onto $C := \cap_{i=1}^m C_i \neq \emptyset$ is sought after by the BAP. Let $\{\sigma_k\}_{k=0}^\infty$ be a user-chosen steering sequence and let $\{w_i\}_{i=1}^m$ be real positive numbers (weights) such that $\sum_{i=1}^m w_i = 1$.*

2. Iterative step: *Take the steering parameter σ_k from the chosen steering sequence. Given the current iterate x^k, calculate the next iterate:*

$$x^{k+1} = \sigma_k y + (1 - \sigma_k) \sum_{i=1}^{m} w_i P_i(x^k). \tag{14}$$

4 Simultaneous algorithms with component averaging

In both Algorithm 2 and Algorithm 5 appear summations that carry out convex combinations of projections, namely, in Equations (8) and (14), respectively. These have the general form

$$\sum_{i=1}^{m} w_i P_i(v) \tag{15}$$

where v is in each case a different point (or different points, as in (14)). If equal weights are chosen $w_i = 1/m$, for all $i = 1, 2, \ldots, m$, and if the sets C_i are as in (17) then these summations, written component-wise, reduce to the form

$$(1/m) \sum_{i=1}^{m} \frac{b_i - \langle a^i, v \rangle}{\|a^i\|^2} a^i_j. \tag{16}$$

Now, if the matrix A (whose i-th row consists of the transposed vector a^i) is sparse then only few components a^i_j in the sum (16) will be different from zero yet the sum is still divided through by m which might be very large and this will slow down the progress of the algorithm in which this convex combination taking step appears. The component-averaging (CAV) idea proposed and experimented with in [16] and further developed in [17], [15] and [45], circumvents this disturbing phenomenon by replacing in (16) the $1/m$ factor by $1/s_j$ where s_j is the number of nonzero elements in the j-th column of the matrix A. This results in the sum being divided by the actual (and much smaller) number of nonzero summands appearing in it. Related to the CAV idea are the recent papers by Gordon and Gordon [34] and by Censor et al. [20]. In contrast with all the above which are related to the CFP, we use this idea here in conjunction with Algorithm 2 and Algorithm 5 for the BAP. We use it only heuristically and report on the success of this combination. A mathematical investigation of this matter is now under way and will be published elsewhere.

5 Test-problems generation

In order to compare the practical initial behavior of Algorithms 1, 2, 4 and 5 we constructed test problems whose solutions will be known to us for the purpose of evaluation of the algorithms' behavior. Once we have a problem with a known solution we monitor the algorithms' behavior by recording the distances of the iterates to the solution. For the combination of the CAV idea with Algorithm 2 and Algorithm 5 we generate test problems in the same fashion only with randomly generated sparse matrices whose degree of sparsity is chosen by the user. All sets in all experiments were half-spaces, i.e.,

$$C_i := \{x \in R^n \mid \langle a^i, x \rangle \leq b_i\}, \text{ for } i = 1, 2, \ldots, m, \tag{17}$$

whose data a^i and b_i were created randomly. To guarantee the feasibility condition $\cap_{i=1}^m C_i \neq \emptyset$ we proceeded as follows. Upon deciding on the values of n and m we randomly generated the vectors $a^i \in R^n$ and the real numbers b_i, for $i = 1, 2, \ldots, m$. The set C_i was defined, for each i, as that half-space which includes the origin in its interior, deleting the pair (a^i, b_i) if it happened to have the origin on the bounding hyperplane and replacing it by another randomly-generated pair. In this way we knew in advance that

we have generated a feasible set of half-spaces. The next step was to use the randomly-generated feasible family of half-spaces to create a BAP whose solution would be known to us. To do so we generated randomly one more pair (a, b) to define the linear function $f(x) := \langle a, x\rangle - b$ and then applied an off-the-shelf linear optimization solver to the problem

$$\min\{f(x) \mid x \in C\}, \tag{18}$$

where C is the nonempty intersection of the randomly-generated half-spaces. Having found a solution, say z, of (18), we choose a real number θ such that the point $y := z + \theta a$ will satisfy $f(z) < f(y)$, so that the point y will be outside C and so that its projection will be $P_C(y) = z$. This is possible since $y - z$ is parallel to the normal of the supporting hyperplane to C at z. In this way we obtained a randomly-generated feasible family of half-spaces and a point y whose projection onto the set intersection C is known to us to be z. We plot for all experiments the distance of iterates from the known true solution z of the generated test problem as a function of iteration index.

There were two difficulties that we encountered while using the above procedure to generate test problems and we briefly describe how we circumvented them. On some test problems we observed that the plots of all four algorithms do not converge to zero but to another fixed positive value, which could change from one problem to another but was identical for all algorithms run on this problem. This indicated that the algorithms converge to another point in C which is at a fixed distance from the true solution z. This could occur if one picks a too large value of θ that results in a point y for which the line segment connecting z with y contains points of C, thus, putting y on the "wrong" side of the set C. To overcome this we just repeated the test problem generation phase for the same data $\{a^i, b_i\}_{i=1}^m$ but with $-\theta$, the negative of the same θ used before. This made the point y lie on the "right" side of C and the algorithms to converge to z.

The other difficulty was with the application of the `linprog` routine of MATLAB when we solved problem (18). Sometimes, quite seldom but particularly when the dimensions n and m where either low or close to each other, `linprog` failed. Without spending too much efforts on studying this phenomenon we simply preferred to use dimensions n and m that are far enough apart which reduced the occurrences of this phenomenon. Alternatively, we sometimes circumvented this by replacing f in (18) by $-f$ and re-running `linprog`. Since these difficulties are related only to the test problems gen-

eration phase we are satisfied by just being able to technically circumvent them.

6 Limited numerical results

The computations were carried out within MATLAB6.5 [49]. We generated hundreds of test problems of various dimensionalities n and m and report here on the main practical conclusions that are based on our experimental observations. Admittedly, more experimental work is needed in order to make more accurate statements about the behavior of the algorithms studied here.

6.1 Experimental details

We describe some cumulative results which are representative of our many runs and experiments. For each stopping iteration number, 20, 50, 100, 200, 400, 800 and 1,000, we constructed 40 randomly-generated BAPs, as described above, for a total of 280 BAPs. Table 1 summarize the results for dimensions $(n, m) = (24, 36)$. The numbers inside the table state for each algorithm how many times (out of the 40 BAPs created for each column) it won the race by reaching the smallest distance to the solution at the particular stopping iteration number.

Algorithm/Stopping Iteration	20	50	100	200	400	800	1000
Sequential HLWB	5	23	19	25	31	30	33
Simultaneous HLWB	0	0	0	0	0	0	0
Sequential Dykstra	35	17	21	15	9	10	7
Simultaneous Dykstra	0	0	0	0	0	0	0

Table 1. $n = 24$, $m = 36$. This information is presented graphically in Diagram 1 below.

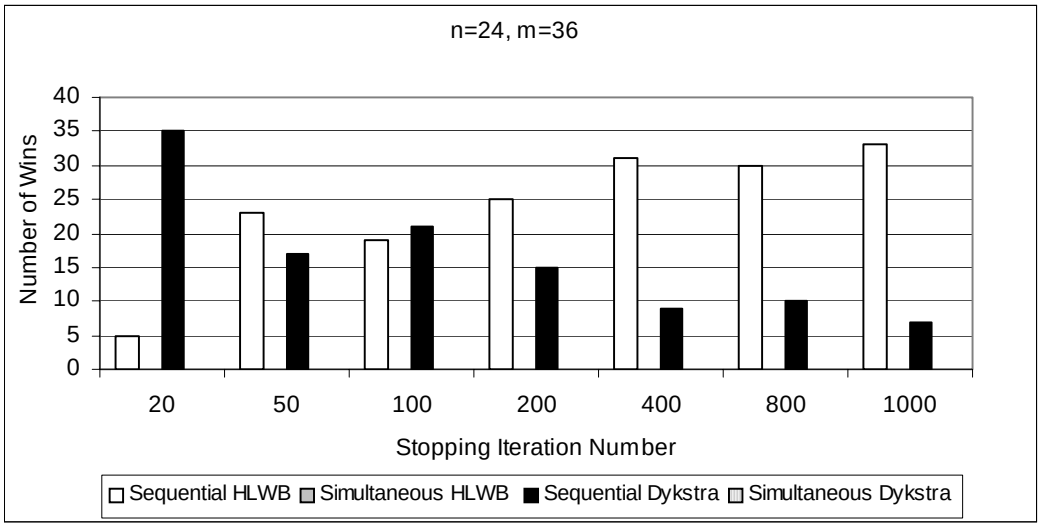

Diagram 1. The simultaneous algorithms did not win even a single run for this problem dimensions.

Figures 1 depicts the "Distance to (the true) Solution" versus "Iteration Number" behavior of the algorithms that we study for dimensions $(n, m) = (24, 36)$. The plots are for a single randomly-generated BAP but are representative of the kind of behavior that we generally observed.

In our experiments we used for the simultaneous algorithms equal weights $w_i = 1/m$ for all i. We employed no relaxation parameters in conjunction with the projections although it is known that other projection methods (such as for the CFP) are sensitive to very small or larger relaxation parameters. We counted an iteration to be one pass through all data to put the sequential and the simultaneous algorithm on equal footing in this respect. We used the most common steering sequence for the HLWB algorithms, namely, $\sigma_k = \dfrac{1}{k+1}$ for all $k \geq 0$.

6.2 Experimental conclusions

No single winner amongst the four algorithms that we worked with can be crowned. The algorithms performed differently on problems of different dimensionalities but several trends were clearly recognizable in our experiments. All statements are made, of course, about the initial behavior of the algorithms.

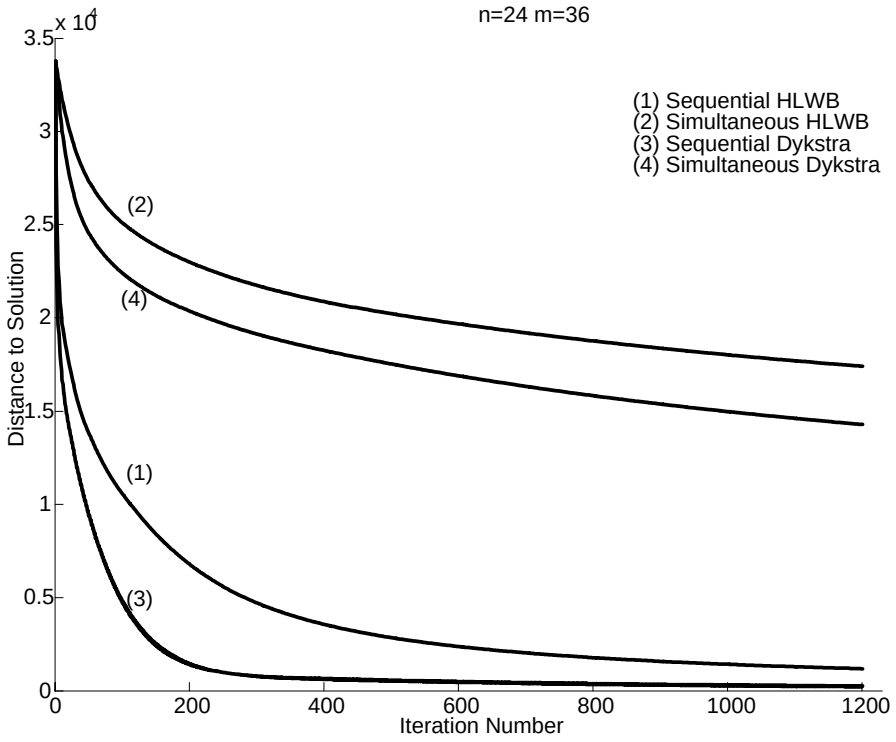

Figure 1: The gap in performance between sequential and simultaneous algorithms, demonstrated here, has been noticed to grow as the dimensionality of the problem increases.

1. Generally, the simultaneous algorithms are slower then the sequential ones, their distance-to-solution versus iteration-index plots lag considerably behind the sequential versions of these algorithms. This is true for one-processor computations, as we did here. The inherent parallelism of the simultaneous algorithms can be exploited to speed up their clock-time achievements, but we did not study this here. This observation is in concert with what has been reported with regard to other iterative algorithms of sequential and simultaneous nature, see, e.g., Censor, Gordon and Gordon [16, 17].

2. The sequential Dykstra algorithm exhibits an oscillatory behavior in most runs, this can be seen in Figures 2–4.

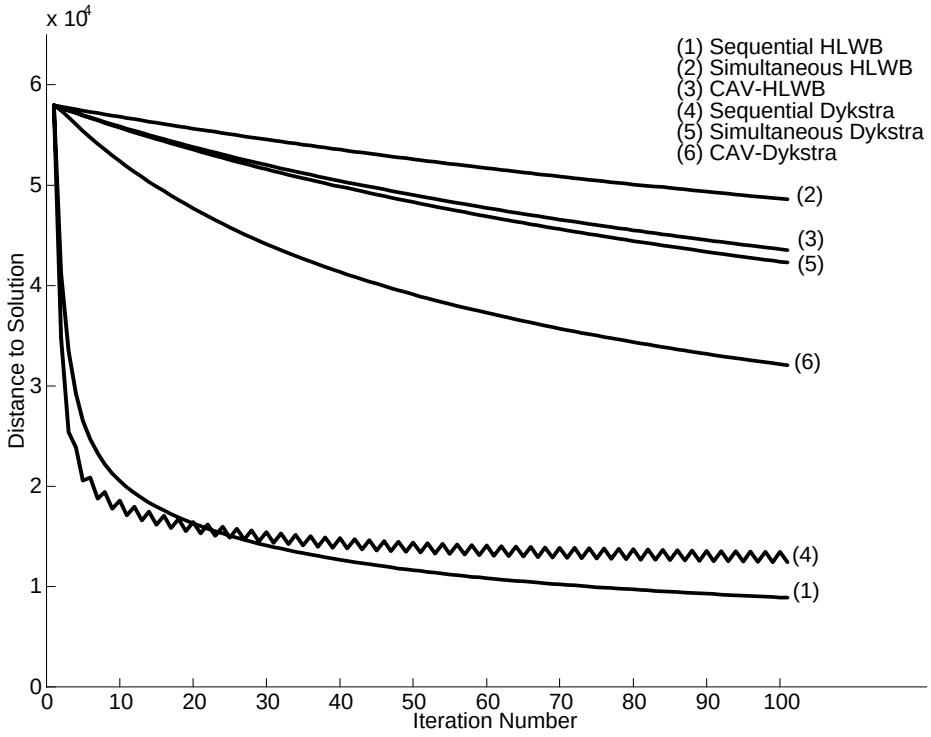

Figure 2: With $n = 100$ and $m = 200$ these are typical plots for a matrix 70% to 80% of whose elements are nonzero (i.e., low sparsity).

3. In spite of the previous point, the initial drop of distance-to-solution is strongest in the sequential Dykstra algorithm, although in the longer run the sequential HLWB algorithm wins over the sequential Dykstra algorithm. This preferential behavior of the sequential Dykstra algorithm must be weighed against its increased memory demands though.

4. Figures 2–4 demonstrate the effect of the CAV idea on the simultaneous algorithms. As the sparsity of the matrix A grows the computational acceleration becomes more pronounced, as can be seen by observing how plots (3) and (6) distance themselves from plots (2) and (5) and get nearer to plots (1) and (4). All plots represent single processor work.

5. Our experiments are of a preliminary nature and leave room for a great

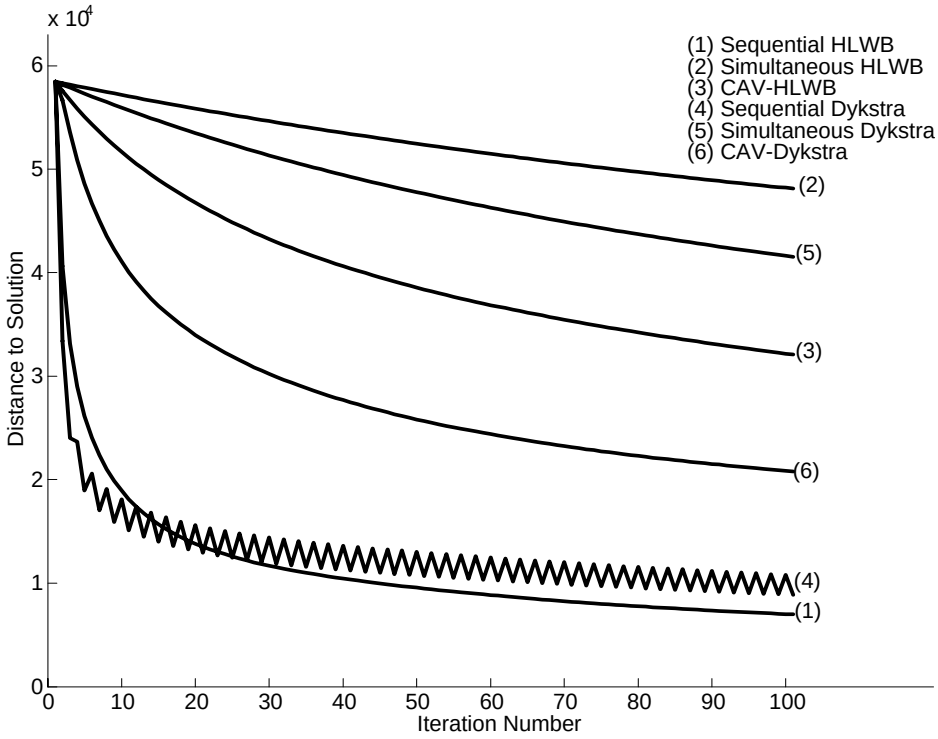

Figure 3: With $n = 100$ and $m = 200$ these are typical plots for a matrix 30% to 40% of whose elelents are nonzero (i.e., medium sparsity).

deal of future experimental work. The problem sizes (n, m) are still very small and must be extended by orders of magnitude before conclusions can be drawn that will be useful for application fields such as those mentioned in the introduction. We plan to do so in the near future. Further work need also to be done to assess the effects of weights $\{w_i\}_{i=1}^m$ on the algorithmic behavior and one must also introduce into the projection steps of the algorithm relaxation parameters and experiment with them. Not less important are the effects of different steering sequences on the HLWB algorithms and they need to be experimented with as well. Extension to convex sets which are not necessarily linear would also be very interesting.

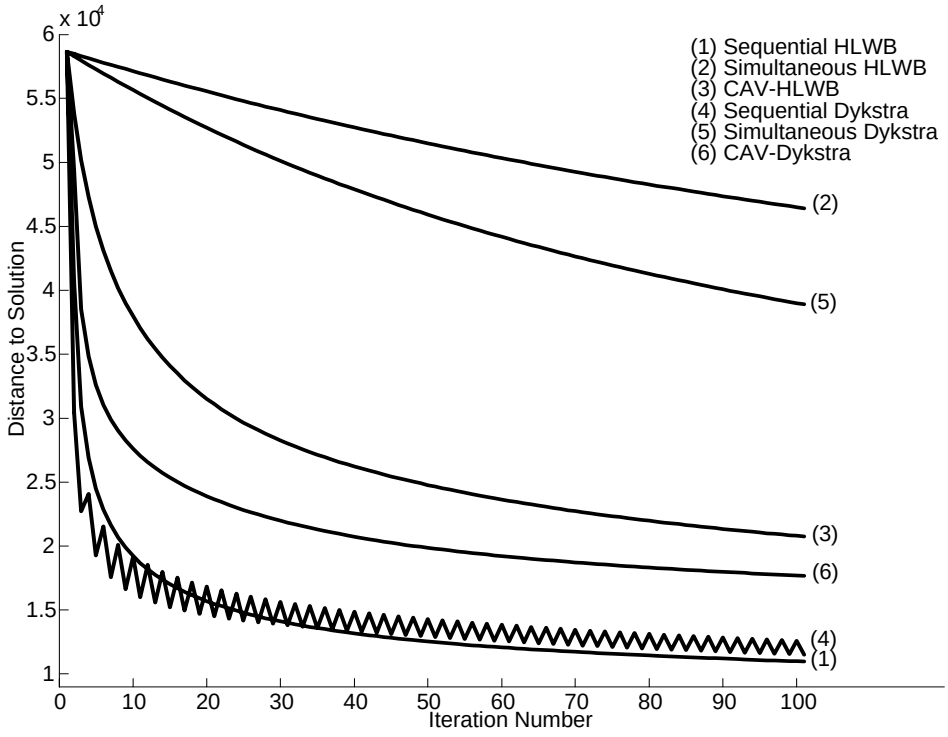

Figure 4: With $n = 100$ and $m = 200$ these are typical plots for a matrix 5% to 10% of whose elelents are nonzero (i.e., high sparsity).

Acknowledgments. We thank Yaniv Zaks for preliminary programming, Arik F. Hatwell for his skillful programming and Dan Gordon, from the Department of Computer Science at the University of Haifa, for his remarks on generating test examples. This research is supported by grant No. 2003275 from the United States-Israel Binational Science Foundation (BSF) and by a National Institutes of Health (NIH) grant No. HL70472. Part of this work was done at the Center for Computational Mathematics and Scientific Computation (CCMSC) at the University of Haifa and supported by research grant No. 522/04 from the Israel Science Foundation (ISF).

References

[1] H.H. Bauschke, "The approximation of fixed points of compositions of nonexpansive mappings in Hilbert space," *Journal of Mathematical Analysis and Applications* **202** (1996), 150–159.

[2] H.H. Bauschke and J.M. Borwein, "On the convergence of von Neumann's alternating projection algorithm for two sets", *Set-Valued Analysis* **1** (1993), 185–212.

[3] H.H. Bauschke and J.M. Borwein, "Dykstra's alternating projection algorithm for two sets", *Journal of Approximation Theory* **79** (1994), 418–443.

[4] H.H. Bauschke and J.M. Borwein, "On projection algorithms for solving convex feasibility problems", *SIAM Review* **38** (1996), 367–426.

[5] H.H. Bauschke and P.L. Combettes, "A weak-to-strong convergence principle for Fejér-monotone methods in Hilbert spaces", *Mathematics of Operations Research* **26** (2001), 248–264.

[6] H.H. Bauschke and A.S. Lewis, "Dykstra's algorithm with Bregman projections: a convergence proof", *Optimization* **48** (2000), 409–427.

[7] E.G. Birgin and M. Raydan, "Robust stopping criteria for Dykstra's algorithm", *SIAM Journal on Scientific Computing* **26** (2004), 1405–1414.

[8] J.P. Boyle and R.L. Dykstra, "A method for finding projections onto the intersection of convex sets in Hilbert spaces", in: *Advances in Order Restricted Statistical Inference*, Lecture Notes in Statistics, Springer-Verlag, Berlin, Germany, Volume **37** (1986), 28–47.

[9] L.M. Bregman, Y. Censor and S. Reich, "Dykstra's algorithm as the nonlinear extension of Bregman's optimization method", *Journal of Convex Analysis* **6** (1999), 319–333.

[10] F.E. Browder, "Convergence of approximations to fixed points of nonexpansive nonlinear mappings in Banach spaces", *Arch. Rational Mech. Anal.* **24** (1967), 82–90.

[11] R.E. Bruck and S. Reich, "Nonexpansive projections and resolvents of accretive operators in Banach spaces", *Houston Journal of Mathematics* **3** (1977), 459–470.

[12] D. Butnariu, Y. Censor and S. Reich (Editors), *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*, Studies in Computational Mathematics 8, Elsevier Science Publishers, Amsterdam, The Netherlands, 2001.

[13] Y. Censor, "Mathematical optimization for the inverse problem of intensity-modulated radiation therapy", in: J.R. Palta and T.R. Mackie (Editors), *Intensity-Modulated Radiation Therapy: The State of The Art*, American Association of Physicists in Medicine (AAPM), Medical Physics Monograph No. 29, Medical Physics Publishing, Madison, Wisconsin, USA, 2003, pp. 25–49.

[14] Y. Censor, M.D. Altschuler and W.D. Powlis, "On the use of Cimmino's simultaneous projections method for computing a solution of the inverse problem in radiation therapy treatment planning", *Inverse Problems* **4** (1988), 607–623.

[15] Y. Censor and T. Elfving, "Block-iterative algorithms with diagonally scaled oblique projections for the linear feasibility problem", *SIAM Journal on Matrix Analysis and Applications* **24**, (2002), 40–58.

[16] Y. Censor, D. Gordon and R. Gordon, "Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems", *Parallel Computing* **27** (2001), 777–808.

[17] Y. Censor, D. Gordon and R. Gordon, "BICAV: A block-iterative, parallel algorithm for sparse systems with pixel-related weighting", *IEEE Transactions on Medical Imaging* **20** (2001), 1050–1060.

[18] Y. Censor and S. Reich, "The Dykstra algorithm with Bregman projections", *Communications in Applied Analysis* **2** (1998), 407–419.

[19] Y. Censor and S.A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press, New York, NY, USA, 1997.

[20] Y. Censor, T. Elfving, G.T. Herman and N. Touraj, "On diagonally-relaxed orthogonal projection methods", Technical Report, September 2005.

[21] P.L. Combettes, "The foundations of set-theoretic estimation", *Proceedings of the IEEE* **81** (1993), 182–208.

[22] P.L. Combettes, "Signal recovery by best feasible approximation", *IEEE Transactions on Image Processing* **IP-2** (1993), 269–271.

[23] P.L. Combettes, "Construction d'un point fixe commun à famille de contractions fermes", *C.R. Acad. Sci. Paris, Sèr. I Math.* **320** (1995), 1385–1390.

[24] P.L. Combettes, "The convex feasibility problem in image recovery", *Advances in Imaging and Electron Physics* **95** (1996), 155–270.

[25] P.L. Combettes, "A block-iterative surrogate constraint splitting method for quadratic signal recovery", *IEEE Transactions on Signal Processing* **51** (2003), 1771–1782.

[26] G. Crombez, "Finding projections onto the intersection of convex sets in Hilbert spaces", *Numerical Functional Analysis and Optimization* **16** (1995), 637–652.

[27] F. Deutsch, *Best Approximation in Inner Product Spaces*, Springer-Verlag, New York, NY, USA, 2001.

[28] F. Deutsch and H. Hundal, "The rate of convergence of Dykstra's cyclic projections algorithm: The polyhedral case", *Numerical Functional Analysis and Optimization* **15** (1994), 537–565.

[29] F. Deutsch and I. Yamada, "Minimizing certain convex functions over the intersection of the fixed point sets of nonexpansive mappings", *Numerical Functional Analysis and Optimization* **19** (1998), 33–56.

[30] R.L. Dykstra, "An algorithm for restricted least squares regression", *Journal of the American Statistical Association* **78** (1983), 837–842.

[31] R.L. Dykstra, "An iterative procedure for obtaining I-projections onto the intersection of convex sets", *The Annals of Probability* **13** (1985), 975–984.

[32] R. Escalante and M. Raydan, "Dykstra's algorithm for a constrained least-squares matrix problem", *Numerical Linear Algebra with Applications* **3** (1996), 459–471.

[33] N. Gaffke and R. Mathar, "A cyclic projection algorithm via duality", *Metrika* **36** (1989), 29–54.

[34] D. Gordon and R. Gordon, "Component-averaged row projections: A robust block-parallel scheme for sparse linear systems, Technical Report, University of Haifa (2005), *SIAM Journal on Scientific Computing,* to appear.

[35] I. Halperin, "The product of projection operators", *Acta Scientiarum Mathematicarum (Szeged)* **23** (1962), 96–99.

[36] B. Halpern, "Fixed points of nonexpanding maps", *Bulletin of the American Mathematical Society* **73** (1967), 957–961.

[37] S.-P. Han, "A successive projection method", *Mathematical Programming* **40** (1988), 1–14.

[38] S.-P. Han and G. Lou, "A parallel algorithm for a class of convex programs", *SIAM Journal on Control and Optimization* **26** (1988), 345–355.

[39] G.T. Herman, *Image Reconstruction From Projections: The Fundamentals of Computerized Tomography*, Academic Press, New York, NY, USA, 1980.

[40] N.J. Higham, "Computing the nearest correlation matrix – a problem from finance", *IMA Journal of Numerical Analysis* **22** (2002), 329–343.

[41] C. Hildreth, "A quadratic programming procedure", *Naval Research Logistics Quarterly* **4** (1957), 79–85. Erratum, ibid., p. 361.

[42] H. Hundal and F. Deutsch, "Two generalizations of Dykstra's cyclic projection algorithm", *Mathematical Programming* **77** (1997), 335–355.

[43] A.N. Iusem and A.R. De Pierro, "A simultaneous iterative method for computing projections on polyhedra", *SIAM Journal on Control and Optimization* **25** (1987), 231–243.

[44] A.N. Iusem and A.R. De Pierro, "On the convergence of Han's method for convex programming with quadratic objective", *Mathematical Programming* **52** (1991), 265–284.

[45] M. Jiang and G. Wang, "Convergence studies on iterative algorithms for image reconstruction", *IEEE Transactions on Medical Imaging* **22** (2003), 569–579.

[46] E. Kopecká and S. Reich, "A note on the von Neumann alternating projections algorithm", *Journal of Nonlinear and Convex Analysis* **5** (2004), 379–386.

[47] A. Lent and Y. Censor, "Extensions of Hildreth's row-action method for quadratic programming", *SIAM Journal on Control and Optimization* **18** (1980), 444–454.

[48] L.D. Marks, W. Sinkler and E. Landree, "A feasible set approach to the crystallographic phase problem", *Acta Crystallographica* **A55** (1999), 601–612.

[49] MATLAB6.5, The MathWorks, Inc., http://www.mathworks.com/.

[50] M. Mendoza, R. Raydan and P. Tarazaga, "Computing the nearest diagonally dominant matrix", *Numerical Linear Algebra with Applications* **5** (1998), 461–474.

[51] M. Raydan and P. Tarazaga, "Primal and polar approach for computing the symmetric diagonally dominant projection", *Numerical Linear Algebra with Applications* **9** (2002), 333–345.

[52] S. Reich, "Strong convergence theorems for resolvents of accretive operators in Banach spaces", *Journal of Mathematical Analysis and Applications* **75** (1980), 287–292.

[53] S. Reich, "Approximating fixed points of holomorphic mappings", *Mathematica Japonica* **37** (1992), 457–459.

[54] S. Reich, "Approximating fixed points of nonexpansive mappings", *Panamerican Mathematical Journal* **4** (1994), 23–28.

[55] T. Robertson, F.T. Wright and R.L. Dykstra, *Order Restricted Statistical Inference*, John Wiley & Sons, Chichester, UK, 1988.

[56] P. Tseng, Dual coordinate ascent methods for non-strictly convex minimization", *Mathematical Programming* **59** (1993), 231–247.

[57] J. von Neumann, "On rings of operators. Reduction theory", *Annals of Mathematics* **50** (1949), 401–485.

[58] R. Wittmann, "Approximation of fixed points of nonexpansive mappings", *Arch. Math.* **58** (1992), 486–491.