Comput Optim Appl
DOI 10.1007/s10589-011-9414-2

How good are extrapolated bi-projection methods for linear feasibility problems?

Nicholas I.M. Gould

Received: 7 March 2011

Abstract We consider extrapolated projection methods for solving linear feasibility problems. Both successive and sequential methods of a two-set projection scheme are examined. The best algorithm in the class of algorithms that we considered was an extrapolated sequential method. When this was compared to an interior point method using the CUTEr/Netlib linear programming test problems it was found that the bi-projection method was fastest (or equal fastest) for 31% of the cases, while the interior point code was fastest in 71% of the cases. The interior-point method succeeded on all examples, but the best bi-projection method considered here failed to solve 37% of the problems within reasonable CPU time or iteration thresholds.

Keywords Projection methods · Convex feasibility problems

1 Introduction

This is a follow-up article to [7] in which (linear) convex feasibility problems of the form

$$\text{find} \quad x \in \mathcal{C} \overset{\text{def}}{=} \{x \mid Ax = b \text{ and } x^l \leq x \leq x^u\} \tag{1}$$

were considered. Here A is m $(\leq n)$ by n and, for simplicity, of full rank, and any of the bounds x^l and x^u may be infinite. In [7] a number of successive and simultaneous bi-projection methods are applied to (1) and compared on the well-known Netlib set

Electronic supplementary material The online version of this article (doi:10.1007/s10589-011-9414-2) contains supplementary material, which is available to authorized users.

N.I.M. Gould (✉)
Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, England, UK
e-mail: nick.gould@stfc.ac.uk

of diverse linear programming problems. While we had thought that projection methods, such as those cited in [7] and elsewhere, provide an effective general-purpose way to satisfy (1), our experiments show that, for the bi-projection methods we considered, this is not so. In particular, the failure rate—effectively observed as numerical stagnation—of the methods we considered on this test set was unacceptably high, particularly when compared with other (non-projection) optimization approaches.

Recently Censor et al. [2] have challenged these conclusions. Most particularly, the authors objected that [7] use "suboptimal" projection methods in the comparisons made. Here we extend our experiments to report on the methods recommended in [2]. While these new experiments support the view in [2] that the recommended methods are generally better than those reported on in [7], we do not find that the experiments substantively alter our overall conclusions with respect to the test-set considered.

In Sect. 2, we describe the projection methods considered, and in Sect. 3, we report on their comparative performance on the Netlib test set. We draw further conclusions in Sect. 4.

2 Projection methods

For the problem (1) of interest to us, we view the feasibility problem as that of finding a point in the intersection of the two sets

$$\mathcal{C}_A \overset{\text{def}}{=} \{x \mid Ax = b\} \quad \text{and} \quad \mathcal{C}_B \overset{\text{def}}{=} \{x \mid x^l \le x \le x^u\}.$$

The projection $p_A(x)$ of a point x into $\mathcal{C}_A$ satisfies the linear system

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} p_A(x) \\ q(x) \end{pmatrix} = \begin{pmatrix} x \\ b \end{pmatrix},$$

while the projection into $\mathcal{C}_B$ is trivially

$$p_B(x) = \text{mid}(x^l, x, x^u).$$

Simple bi-projection methods for (1) apply p_A and p_B either one-at-a-time or simultaneously to improve an estimate x_k, as we now describe; the cost of finding $p_A(x)$ usually dominates.

We focus on this specific view of the feasibility problem. Of course, there are many possible ways to formulate the feasibility problem, such as the intersection of the sets $\mathcal{C}_i = \{x \mid a_i^T x = b_i,\ x^l \le x \le x^u\}$, $i = 1, \ldots, m$, and each such view of the feasibility problem leads to a different class of projection methods.

2.1 Successive bi-projection methods

A general class of successive bi-projection methods for (1) is given in Algorithm 2.1.

Algorithm 2.1: Successive bi-projection.

Given $x_0 \in \mathcal{C}_A$, for $k = 0, 1, \ldots$ until convergence, set

$$x_{k+1} = (1 - \alpha_k)x_k + \alpha_k p_A(p_B(x_k))$$

for some $\alpha_k > 0$.

In [7], the simple choice $\alpha_k = 1$ (henceforth referred to as **A2.1**($\alpha = \mathbf{1}$)) as well as the locally-optimal choice (**A2.1**($\alpha = \mathbf{opt}$)) in which $\alpha = \alpha_k$ minimizes $\|p_B(x(\alpha)) - x(\alpha)\|_2$, where $x(\alpha) = (1-\alpha)x_k + \alpha p_A(p_B(x_k))$, were considered. The Extrapolated Alternating Projection Method (EAPM) [1] championed in [2] recommends the alternative

$$\alpha_k = \rho \begin{cases} \frac{\|p_B(x_k)-x_k\|_2^2}{\|p_A(p_B(x_k))-x_k\|_2^2} & \text{if } x_k \notin \mathcal{C}_B \text{ or} \\ 1 & \text{otherwise,} \end{cases}$$

where $\rho \in (0, 2]$. The latter choice has the strong advantage that its theoretical convergence has been established [1, Corollary 4.11].

2.2 Simultaneous bi-projection methods

An alternative general class of simultaneous bi-projection methods for (1) is given in Algorithm 2.2.

Algorithm 2.2: Simultaneous bi-projection.

Given x_0 and $\lambda \in (0, 1)$, for $k = 0, 1, \ldots$ until convergence, set

$$x_{k+1} = (1 - \alpha_k)x_k + \alpha_k \left[\lambda p_A(x_k) + (1 - \lambda) p_B(x_k)\right]$$

for some $\alpha_k > 0$.

Typically $\lambda = \frac{1}{2}$. Both the simple choice $\alpha_k = 1$ (**A2.2**($\alpha = \mathbf{1}$)) and the locally-optimal choice (**A2.2**($\alpha = \mathbf{opt}$)), in which $\alpha = \alpha_k$ minimizes $\lambda\|p_A(x(\alpha)) - x(\alpha)\|_2^2 + (1-\lambda)\|p_B(x(\alpha)) - x(\alpha)\|_2^2$, where $x(\alpha) = (1-\alpha)x_k + \alpha(\lambda p_A(x_k) + (1-\lambda)p_B(x_k))$, were considered in [7]. The Extrapolated Parallel Projection Method (EPPM) [10], for which

$$\alpha_k = \rho \begin{cases} 2\frac{\|p_A(x_k)-x_k\|^2+\|p_B(x_k)-x_k\|^2}{\|p_A(x)+p_B(x_k)-2x_k\|^2} & \text{if } x_k \notin \mathcal{C} \text{ or} \\ 1 & \text{otherwise,} \end{cases}$$

where $\rho \in (0, 2]$, is recommended as better in [2], and as before, convergence of this variation is guaranteed [3, Proposition 2.2, 10, Theorem 1.2].

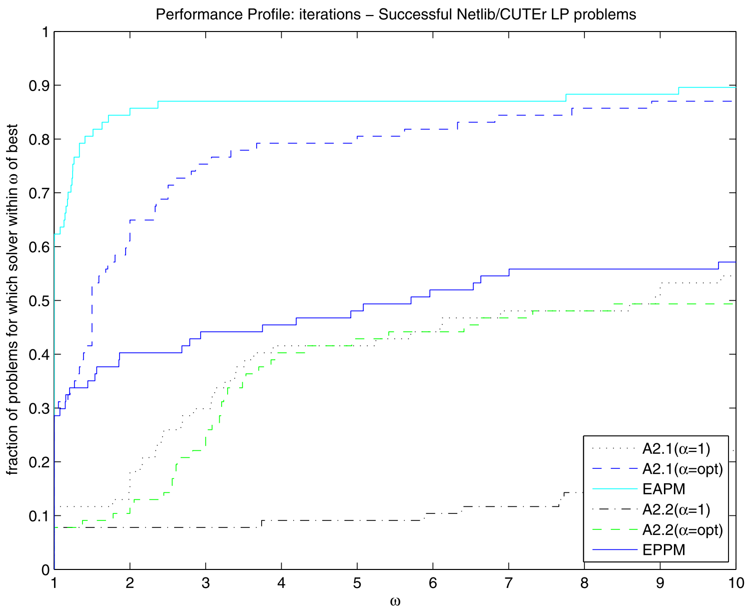

Fig. 1 Performance profile: iteration counts for the 77 problems which are successfully solved by all variants under consideration

3 Numerical experience

We modified the GALAHAD [9] fortran 2003 package `LCF`, which we previously used to obtain the results reported in [7], to include additionally the EAPM and EPPM methods described in Sect. 2. We used the value $\rho = 1.8$ that appears to work best in the experiments in [1], but note that we observed quantitatively similar results with other ρ including $\rho = 1$. See [7, Sect. 4] for general details of our implementation.

As before, we evaluate the competing methods by trying to find a feasible point for each of the sets $\mathcal{C}$ given by the complete set of 118 feasible linear programming Netlib and other linear programming test problems as distributed with CUTEr [8]—slack variables are added as necessary as described in [7] to convert the problems into the form (1). Our results are summarised in the iteration/projection and CPU performance profiles [6] given in Figs. 1 and 2, extending those in [48, Figs. 1 & 3] to include results for the EAPM and EPPM strategies. We removed the inferior $\alpha = 1$ results from the CPU comparisons to avoid clutter, but instead included the times taken by the GALAHAD interior-point package `LSQP` that can also be used to find the projection of x_0 onto $\mathcal{C}$. The complete (additional) set of results for the two extrapolated variants are given in Appendix A of the on-line attachment to this paper.

These figures and tables suggest the following:

1. Figure 1 confirms that the authors of [2] are correct in saying that EAPM and EPPM *are* better than those discussed in [7]. Indeed since the best bi-projection methods tested in [7] are heuristic, whereas EAPM and EPPM have a convincing convergence theory, this is a satisfactory outcome.

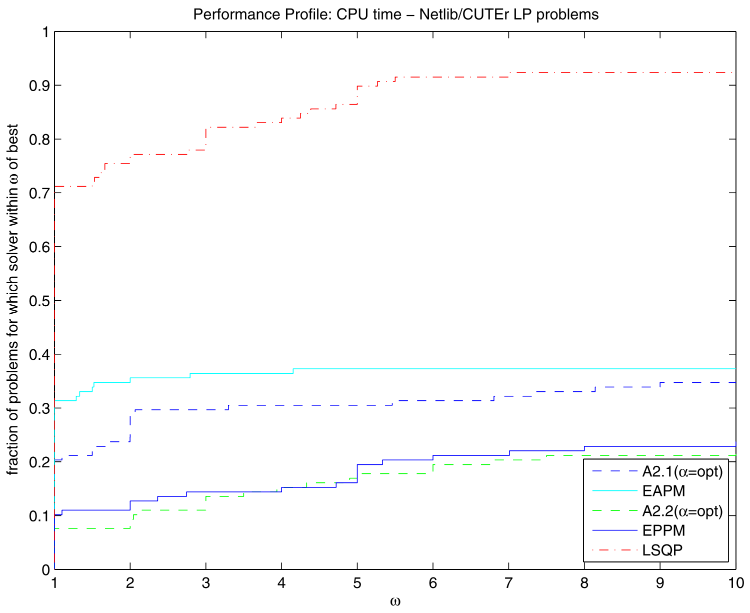

Fig. 2 Performance profile, including LSQP: CPU time for the 118 problems under consideration

2. The hierarchy of success is as one might predict in that the sequential methods generally beat the simultaneous ones. EAPM certainly beats its nearest competitor **A2.1**($\alpha =$ **opt**), but not overwhelmingly so. But certainly in some cases, EAPM takes very few iterations.
3. The reliability does not seem to improve with the extrapolation methods. When these methods work, they generally do better than their competitors. However, there is a significant failure rate (44 cases out 118 problems tested for EAPM and 43 case for EPPM) either because a CPU limit (half an hour) or iteration limit (1000000 projections) is reached. Precisely why this should be is unclear, but [2] attribute it to the examples in the Netlib set generally not having "full-dimensional" feasible sets. Certainly, the observed convergence prior to these failures is extremely slow.
4. It is of course difficult to say whether better variants of EAPM/EPPM will be found in the future, but to date, on problems of this size from this diverse source, other alternatives (such as those based on interior-point methods) appear from Fig. 2 to be both more reliable and considerably better on the CUTEr/Netlib test set.

Since the authors of [2] base many of their early conclusions [2, Sect. 2.1] on randomized problems, we took their random class of examples, and repeated their experiments as best we could—we didn't have their pseudo-random number generator, but used that from GALAHAD instead, and we had to be content with systems of order 700 by 300 since the factorization code we used was (we discovered) poorly designed for dense problems. We observed profiles of the form given in [2, Fig. 3], with EAPM requiring on average 5 iterations, **A2.1**($\alpha =$ **opt**) 8 iterations and **A2.2**($\alpha =$ **opt**) 75 iterations to obtain full machine accuracy. So while we certainly accept that our earlier implemented methods are "suboptimal", the authors' suggestion [2, Sect. 2.1] that they are not representative of the best methods is, we would argue, debatable, since the differences are relatively minor. Moreover, as the average counts on these random examples are considerably lower than most of the real-life examples from the Netlib set, this suggests that random examples may not reflect practical experience in many cases.

4 Comments and conclusions

We already mentioned in our conclusions to [7] that there are problems in important application areas for which projection methods really work well. Moreover, the authors of [2] stress that the ability to solve larger problems than those in the Netlib set is a primary advantage of projection methods. Our analysis of the projection methods EAPM and EPPM suggested by the authors of [2] indicates that these methods yield better performance than the bi-projection methods studied in [7]. Nonetheless, Fig. 2 shows that for the Netlib collection of test problems, an interior point method, on average, performs better than either EAPM or EPPM. More precisely, the interior point code was fastest in 71% of the cases.

Of course, other projection methods may well be better than the ones considered here. For example, when the feasible set is viewed as the intersection of sets of the form $\mathcal{C}_i$, then the large LPs known as `NUG20` and `NUG30` were solved more efficiently by projection techniques in [5] than by either Simplex or interior point-type methods. In [2], for example, the `ISRAEL` problem from the Netlib set can be solved very efficiently by splitting $Ax = b$ into blocks and applying multiple projections to each block successively or in sequence. We would welcome the development of generally applicable software to implement such ideas. Preliminary work along these lines is given in [4].

Acknowledgements The authors is grateful to Bill Hager for encouraging him to publish these follow-up results to [7] and for his helpful suggestions on the text herein. This work was supported by the EPSRC grant EP/E053351/1.

References

1. Bauschke, H.H., Combettes, P.L., Kruk, S.G.: Extrapolation algorithm for affine-convex feasibility problems. Numer. Algorithms **41**(3), 239–274 (2006)
2. Censor, Y., Chen, W., Combettes, P.L., Davidi, R., Herman, G.T.: On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints. Comput. Optim. Appl. (2011). doi:10.1007/s10589-011-9401-7
3. Combettes, P.L.: Hilbertian convex feasibility problem: Convergence of projection methods. Appl. Math. **35**(3), 311–330 (1997)
4. Davis, T.A., Hager, W.W.: Dual multilevel optimization. Math. Program. **112**(2), 403–425 (2008)
5. Davis, T.A., Hager, W.W.: A sparse proximal implementation of the lp dual active set algorithm. Math. Program. **112**(2), 275–301 (2008)
6. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)

7. Gould, N.I.M.: How good are projection methods for convex feasibility problems? Comput. Optim. Appl. **40**(1), 1–12 (2008)
8. Gould, N.I.M., Orban, D., Toint, Ph.L.: CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited. ACM Trans. Math. Softw. **29**(4), 373–394 (2003)
9. Gould, N.I.M., Orban, D., Toint, Ph.L.: GALAHAD—a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. ACM Trans. Math. Softw. **29**(4), 353–372 (2003)
10. Pierra, G.: Decomposition through formalization in a product space. Math. Program. **28**(1), 96–115 (1984)