RAL-TR-2006-015

CCLRC

How good are projection methods for convex feasibility problems?

N I M Gould

September 21, 2006

Council for the Central Laboratory of the Research Councils

Library and Information Services
CCLRC Rutherford Appleton Laboratory
Chilton Didcot
Oxfordshire OX11 0QX
UK
Tel: +44 (0)1235 445384
Fax: +44(0)1235 446403
Email: library@rl.ac.uk

CCLRC reports are available online at:
http://www.clrc.ac.uk/Activity/ACTIVITY=Publications;SECTION=225;

ISSN 1358-6254

RAL-TR-2006-015

How good are projection methods for convex feasibility problems?

Nicholas I. M. Gould[1,2,3]

ABSTRACT

We consider simple projection methods for solving convex feasibility problems. Both successive and sequential methods are considered, and heuristics to improve these are suggested. Unfortunately, particularly given the large literature which might make one think otherwise, numerical tests indicate that in general none of the variants considered are especially effective or competitive with more sophisticated alternatives.

---

[1] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, UK.
Email: n.i.m.gould@rl.ac.uk

[2] Current reports available from "http://www.numerical.rl.ac.uk/reports/reports.shtml".

[3] This work was supported by the EPSRC grant GR/S42170.

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxfordshire OX11 0QX

September 21, 2006

1 Introduction

One of the basic tasks in computational science is to find a point satisfying a given set of equations and/or inequalities. In general this may be a very hard problem, but fortunately for many realistic examples, the solution set of feasible points is convex. Over the years, a number of simple methods have been proposed for such problems, in which a sequence of projections into simpler regions defined by subsets of the required equations/inequalities are combined in appropriate ways. Many of these methods have strong convergence guarantees [2,6,9], and their simplicity makes them appear most appealing for large-scale computation. However, despite the large number of theoretical papers devoted to generalizations and convergence issues, there appears to have been little effort to investigate how they really perform in practice. In this paper, we attempt to do so in perhaps the most favourable circumstances, for which the feasible region is defined as the intersection of two convex sets and for which individual projections may be easily computed but for which the joint projection may not.

The particular case we investigate is the linear-feasibility problem. Here we wish to find a point $x \in \mathbb{R}^n$ satisfying

$$Ax = b \tag{1.1a}$$

$$\text{and} \quad x^l \leq x \leq x^u, \tag{1.1b}$$

where A is m ($\leq n$) by n and, for simplicity, of full rank, and any/all of the bounds x^l and x^u may be infinite. Given an initial point x_0, many of the methods described in the literature aim to find the x which is "closest" to x_0 in a well-defined sense; in the methods we will consider here, closest will be measured in terms of minimizing $\|x - x_0\|_2$.

The problem (1.1) is important both in its own right, and as a way of means to solving other optimization problems—for example, the solution set for the linear-programming problem (in standard form),

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \;\; c^T x \;\; \text{subject to} \;\; Ax = b \;\; \text{and} \;\; x \geq 0$$

may be expressed as

$$\begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ c^T & -b^T & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} b \\ c \\ 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} x \\ z \end{pmatrix} \geq 0,$$

where the unknowns y and z are Lagrange multipliers and dual variables respectively.

The paper is organised as follows. Sequential and successive projection methods are reviewed in §2.1, and suggestions on how they might be accelerated for the special problem (1.1) are proposed in §3. The results of numerical experiments are reported in §4, and conclusions drawn in \$5.

2 Projection methods

We now examine the basic classes of projection methods that have been proposed. We consider the general problem for which $\mathcal{C}_i$, $i \in \mathcal{S} \stackrel{\rm def}{=} \{1, \ldots, s\}$, are closed, convex sets, and we wish to find a point $x \in \mathcal{C} \stackrel{\rm def}{=} \bigcap_{i \in \mathcal{S}} \mathcal{C}_i$. For simplicity, we restrict ourselves to finite-dimensional spaces, but note that much of what we say can be generalized to Hilbert space [1–3].

2.1 Simple projections

We denote the (orthogonal) projection $P_i(x)$ of x onto $\mathcal{C}_i$ as

$$P_i(x) = \ \arg\min_{y \in \mathcal{C}_i} \ F(y, x), \tag{2.2}$$

where $F(y, x) = \frac{1}{2}\|y - x\|_2^2$. This is an important special case of a very general framework in which $F(y, x) = f(y) - f(x) - \langle \nabla_x f(x), y - x \rangle$ for a given inner product $\langle \cdot, \cdot \rangle$, where the differentiable f is a so-called Bregman function [9, §2.1]. Our fundamental assumption will be that we are able to solve (2.2) at some acceptable cost for each $i \in \mathcal{S}$.

For the system (1.1) of most interest to us, if we let

$$\mathcal{C}_1 = \mathcal{C}_B \ \stackrel{\rm def}{=} \ \{x \ | \ x^l \leq x \leq x^u\} \tag{2.3a}$$

$$\text{and } \ \mathcal{C}_2 = \mathcal{C}_A \ \stackrel{\rm def}{=} \ \{x \ | \ Ax = b\}, \tag{2.3b}$$

then

$$P_1(x) = P_B(x) \stackrel{\rm def}{=} \text{mid}(x^l, x, x^u), \tag{2.4}$$

the compenent-wise median of the vectors x^l, x and x^u, while $P_2(x) = P_A(x) \stackrel{\rm def}{=} p(x)$, where $(p(x), q(x))$ satisfy the linear system

$$\begin{pmatrix} I & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} p(x) \\ q(x) \end{pmatrix} = \begin{pmatrix} x \\ b \end{pmatrix}. \tag{2.5}$$

Certainly (2.4) is trivial, while in many cases (2.5) may be solved by a suitable factorization. More generally, given finite collections of sets $\{\mathcal{B}_j\}_{j=1}^{s_b}$ and $\{\mathcal{L}_i\}_{i=1}^{s_l}$ for which $\bigcup_j \mathcal{B}_j = \mathcal{N} \stackrel{\rm def}{=} \{1, \ldots, n\}$ and $\bigcup_i \mathcal{L}_i = \mathcal{M} \stackrel{\rm def}{=} \{1, \ldots, m\}$, we might choose

$$\mathcal{C}_j \ = \ \{x \ | \ x_i^l \leq x_i \leq x_i^u \ \text{ for all } \ i \in \mathcal{B}_j\}, \ \ j = 1, \ldots s_b, \ \text{ and} \tag{2.6a}$$

$$\mathcal{C}_{s_b+i} \ = \ \{x \ | \ a_j^T x = b_j \ \text{ for all } \ j \in \mathcal{L}_i\}, \ \ i = 1, \ldots s_l. \tag{2.6b}$$

In this case

$$(P_j(x))_i = \left\{ \begin{array}{cc} \text{mid}(x_i^l, x_i, x_i^u) & \text{for } \ i \in \mathcal{B}_j \\ x_i & \text{for } \ i \notin \mathcal{B}_j \end{array} \right\} \ \text{ for } \ j = 1, \ldots s_b \tag{2.7}$$

and $P_{s_b+i}(x) = p_i(x)$, where $(p_i(x), q_i(x))$ satisfy the linear system

$$\begin{pmatrix} I & A_{\mathcal{B}_i}^T \\ A_{\mathcal{B}_i} & 0 \end{pmatrix} \begin{pmatrix} p_i(x) \\ q_i(x) \end{pmatrix} = \begin{pmatrix} x \\ b_{\mathcal{B}_i} \end{pmatrix} \tag{2.8}$$

and $A_{\mathcal{B}_i}$ and $b_{\mathcal{B}_i}$ denote the rows of A and entries of b indexed by $\mathcal{B}_i$, for $i = 1, \ldots s_l$. In the extreme case $\mathcal{B}_j = \{j\}$, $j = 1, \ldots n$, and $\mathcal{L}_i = \{i\}$, $i = 1, \ldots m$, all the projections are onto spaces, or segments of spaces, of dimension one and thus trivial to compute.

2.2 Successive projection methods

Armed with suitable ways to apply projections, perhaps the simplest way to try to find a point in $\mathcal{C}$ is to project successively onto the sets $\mathcal{C}_i$. In particular, we have the following scheme.

Algorithm 2.1: Successive projection.

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

$$x_{k+1} = (1 - \alpha_k)x_k + \alpha_k P_{j(k)}(x_k),$$

where $j(k) = k \bmod s + 1$, for some $\alpha_k > \epsilon$.

This method is sometimes known as cyclic or alternating projection. The simplest case, when $\alpha_k \equiv 1$, was first analysed for the case where $\{\mathcal{C}_i\}_{i \in \mathcal{S}}$ are affine subspaces by von Neumann [31] (when $s = 2$) and Halperin [20] (for $s > 2$); in this case the algorithm converges to the (orthogonal) projection $P(x_0)$ of x_0 onto $\mathcal{C}$ when $\mathcal{C} \neq \emptyset$. Subsequently, the convergence for more general sets has been addressed, and in particular for polyhedral $\mathcal{C}_i$ (which includes the descriptions of (1.1) given in §2.1), convergence to a point within $\mathcal{C} \neq \emptyset$ at a globally linear rate has been established [3, Thm. 5.6.3]. For affine subspaces $\mathcal{C}_i$ with non-empty mutual intersection, the linear convergence factor is the "cosine of the angles between the subspaces", [3, Thm. 5.7.8], and one might anticipate that a similar factor involving projections onto "active" constraints would hold in the polyhedral case. [11]. If the projection $P(x_0)$ is required in the non-affine case, it may be necessary to add recursively-computed outward normals before applying each projection onto non-affine $\mathcal{C}_i$—this is Dykstra's algorithm [15] (see also [21]), and when $\mathcal{C}_i = \{x \mid a_i^T x \geq b_i\}$, this is equivalent to Hildreth's method [22].

The extension to non-unit α_k is due to Gubin, Polyak and Raik [19]. Convergence for general $\alpha_k \in [\epsilon, 2-\epsilon]$ has been established for special classes of $\mathcal{C}_i$ [9, §5.2]. Generalisations in which $j(k)$ need only be ℓ infinitely often for each $\ell \in \mathcal{S}$, but not necessarily cyclically so, have also been suggested. For the very special case for which we simply wish to find a solution to $Ax = b$, and $\mathcal{C}_i = \{x | a_i^T x = b_i\}$, $i = 1, \ldots, m$, the above algorithm is that of Kaczmarz [26] when $\alpha_k = 1$.

2.3 Simultaneous projection methods

Rather than successively projecting the current iterate onto individual constraint sets is to project simultaneously onto all of the sets, and then form a strictly convex sum of these projections. We summarize this as follows.

Algorithm 2.2: Simultaneous projection.

Given x_0, and $\epsilon \in (0,1)$, choose

$$\lambda_i > 0,\ i \in \mathcal{S},\ \text{ for which } \sum_{i \in \mathcal{S}} \lambda_i = 1.$$

For $k = 0, 1, \ldots$ until convergence, set

$$x_{k+1} = (1 - \alpha_k) x_k + \alpha_k \sum_{i \in \mathcal{S}} \lambda_i P_i(x_k)$$

for some $\alpha_k \geq \epsilon$.

The advantage here is that convergence is ensured even when $\mathcal{C} = \emptyset$, and in the latter case x_k converges to the point for which

$$\sum_{i \in \mathcal{S}} \lambda_i \|P_i(x) - x\|_2^2$$

is minimized. The method may actually be interpreted in the simultaneous projection framework within an appropriate product space when $\alpha_k = 1$ [29], and thus convergence ensured using known results. In particular, for the case of affine sets, this is Cimmino's method [10]. Convergence for any $\alpha_k \in [\epsilon, 2 - \epsilon]$ has been established for various classes of $\mathcal{C}_i$ [27, 30].

If the projection $P(x_0)$ is required, as for the successive projection case it may be necessary to add recursively-computed outward normals before applying each projection onto non-affine $\mathcal{C}_i$ [24, 25].

3 Stepsize choice for the linear-feasibility problem

We now return to the linear-feasibility problem (1.1), and consider the methods proposed in §2 for the bi-projection case for which $\mathcal{C}_1 = \mathcal{C}_B$ and $\mathcal{C}_2 = \mathcal{C}_A$ as in (2.3). In this case, it is convenient to rewrite Algorithm 2.1 as follows.

Algorithm 3.1: Successive bi-projection.

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

$$x_{k+1} = (1 - \alpha_k^A)x_k^B + \alpha_k P_A(x_k^B), \tag{3.9}$$

where

$$x_k^B = (1 - \alpha_k^B)x_k + \alpha_k^B P_B(x_k),$$

for some $\alpha_k^A, \alpha_k^B > \epsilon$.

While, as we have indicated, it is usual to pick the stepsizes $\alpha_k^A, \alpha_k^B \in [\epsilon, 2-\epsilon]$, there is some evidence to suggest that larger stepsizes may be beneficial [29]. In particular, Bauschke and Kruk [5] present numerical results that suggest picking $\alpha_k^A = 1$ and $\alpha_k^B = 2$ is better than other $\alpha_k^A, \alpha_k^B \in [0, 2]$, along with an accompanying proof of convergence for this case. More recently, Bauschke, Combettes and Kruk [4] have investigated the $\alpha_k^A = 1$ case in more detail—since projection onto C_A is linear, (3.9) may be written as

$$x_{k+1} = (1 - \alpha_k)x_k + \alpha_k P_A(P_B(x_k))$$

so long as $x_k \in \mathcal{C}_A$, where we have replaced α_k^B by α_k— and have established convergence for α_k (significantly) larger than 2.

How should we pick α_k? Since $\{x_k\}_{k\geq 1} \in \mathcal{C}_A$, it is convenient to measure convergence in terms of $\|P_B(x_k) - x_k\|_2$. Thus, if we let

$$x(\alpha) = (1 - \alpha)x_k + \alpha P_A(P_B(x_k)),$$

it seems reasonable to pick α_k to minimize $\phi(\alpha) = \|P_B(x(\alpha)) - x(\alpha)\|_2^2$.

Since ϕ is a differentiable convex piecewise-quadratic function of α, a global minimizer is easy to find. Indeed, each component of $P_B(x(\alpha)) - x(\alpha)$ has derivative discontinuities at at most two values ("breakpoints") of α. Thus arranging the at-most $2n$ breakpoints of $P_B(x(\alpha)) - x(\alpha)$ using a heapsort [32], the resulting quadratic may be examined between increasing pairs of breakpoints until the global minimizer is found. The slope and curvature of the quadratic may be updated trivially as breakpoints are traversed, and the overall cost of the stepsize determination is $O(n \log n)$ integer and floating-point operations.

Turning to simultaneous projection, Algorithm 2.2 simplifies as follows.

Algorithm 3.2: Simultaneous bi-projection.

Given x_0, $\lambda \in (0,1)$ and $\epsilon \in (0,1)$, 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 $\alpha_k \geq \epsilon$.

Here progress is measured in terms of $\lambda\|P_A(x) - x\|_2^2 + (1-\lambda)\|P_B(x) - x\|_2^2$. Thus it would seem reasonable to consider the line

$$x(\alpha) = (1 - \alpha)x_k + \alpha z_k, \quad \text{where} \quad z_k = \lambda P_A(x_k) + (1 - \lambda)P_B(x_k),$$

and to pick α_k to minimize $\psi(\alpha) = \lambda\|P_A(x(\alpha)) - x(\alpha)\|_2^2 + (1 - \lambda)\|P_B(x(\alpha)) - x(\alpha)\|_2^2$. As for the successive bi-projection case, ψ is a differentiable convex piecewise-quadratic function of α and thus its global minimizer is easy to find.

One small implementational issue is that in order to evaluate $\psi(\alpha)$, it would appear at first sight that we need to evaluate

$$P_A((1 - \alpha)x_k + \alpha z_k) = (1 - \alpha)P_A(x_k) + \alpha P_A(z_k)$$

and thus that we need to compute (potentially expensive) projections at both x_k and z_k. But of course we may subsequently recur

$$P_A(x_{k+1}) = (1 - \alpha_k)P_A(x_k) + \alpha_k P_A(z_k)$$

to start the next iteration, and thus each iteration actually only requires a single projection $P_A(z_k)$ into $\mathcal{C}_A$.

4 Numerical experience

We now turn to our numerical experience with the algorithms we discussed in Section 3 for the linear feasibility problem (1.1). We have implemented both Algorithms 3.1 and 3.2 as a Fortran 95 module `LCF` (`L`inearly-`C`onstrained `F`easibility) as part of the upcoming release 2.0 of the nonlinear optimization library GALAHAD [18].

`LCF` is actually designed to find a feasible point for the general linear constraint set

$$c^l \leq Ax \leq c^u \quad \text{and} \quad x^l \leq x \leq x^u. \tag{4.1}$$

Here any or all of the components of the bounds c^l, c^u, x^l and x^u may be identical or infinite, thus allowing equality constraints and fixed or free variables. Although the details

and bookkeeping are more complicated than for (1.1), the underlying method is essentially to introduce slack variables c and to find a feasible point for the equivalent set

$$Ax - c = 0 \text{ and } \begin{pmatrix} x^l \\ c^l \end{pmatrix} \leq \begin{pmatrix} x \\ c \end{pmatrix} \leq \begin{pmatrix} x^u \\ c^u \end{pmatrix}. \tag{4.2}$$

The dominant cost per iteration of each algorithm is, of course, the cost of computing the projection $P_A(x)$. As we suggested in §2.1, this projection may be calculated as the solution to the symmetric, indefinite (augmented) linear system (2.5), but sometimes it is more convenient to obtain it using the range-space/Schur complement approach, for which

$$p(x) = x - A^T q(x), \text{ where } AA^T q(x) = Ax - b$$

and the crucial matrix AA^T is positive definite. `LCF` uses another GALAHAD module `SBLS` (itself built on top of the HSL [23] package `MA27` [14]) to find the projection. `SBLS` uses the Schur complement method by default, but switches to the augmented system approach if A has any relatively dense columns or if AA^T proves to be too ill-conditioned. `MA27` may be trivially replaced by the more powerful (but commercial) package `MA57` [13] if desired.

We consider four strategies. The first two correspond to Algorithm 3.1, the first (later denoted **A3.1**($\alpha = \mathbf{1}$)) with $\alpha_k^A = \alpha_k^B = 1$ being the original von Neumann [31] approach, while the second (**A3.1**($\alpha = \mathbf{opt}$)) is with $\alpha_k^A = 1$ and α_k^B being chosen to minimize $\phi(\alpha)$ as described in §3. The second pair of strategies correspond to Algorithm 3.2 with $\lambda = \frac{1}{2}$, the first (**A3.2**($\alpha = \mathbf{1}$)) using the traditional $\alpha_k = 1$, and the second (**A3.2**($\alpha = \mathbf{opt}$)) with α_k chosen to minimize $\psi(\alpha)$.

In our tests, we start from the (usually infeasible) point x_0 for which $(x_0)_i = x_i^l - 1$, if x_i^l is finite, otherwise $(x_0)_i = x_i^u + 1$, if x_i^u is finite, and otherwise $(x_0)_i = 0$. The corresponding $c_0 = Ax_0$. Each run is terminated as soon as $\max(\|P_A(x,c) - (x,c)\|_2, \|P_B(x,c) - (x,c)\|_2)$ is smaller than a prescribed stopping tolerance, here 0.000001 In addition, a CPU time limit of half an hour, and an iteration bound of one million iterations is imposed to preclude unacceptably poor performance.

No preprocessing of the problems is performed by default, aside from identifying and removing dependent linear equality constraints. We do this by calling yet another GALAHAD module `FDC`. This factorizes

$$\begin{pmatrix} \alpha I & A^T \\ A & 0 \end{pmatrix} = LBL^T$$

using `MA27`, and assesses rank deficiency from tiny eigenvalues of the one-by-one and two-by-two diagonal blocks that make up B—a value $\alpha = 0.01$ is used by default. We recognize that this is not as robust as, for example, a singular-value decomposition or a rank-revealing QR factorization, but fortunately has proved remarkably reliable in our tests.

All of our experiments were performed on a single processor of a 3.06 GHz Dell Precision 650 Workstation. The codes were compiled using full optimization with the Intel ifort compiler, and the computations were performed in double precision.

We consider the complete set of linear programming Netlib and other linear programming test problems as distributed with CUTEr [17]. Our intention here is simply to

evaluate projection methods for a diverse and important set of real-life examples. In particular, since almost all of these problems are known to have feasible points, we aim to find such a point efficiently.

For comparison purposes, we also apply the GALAHAD interior-point package `LSQP` to find the projection of x_0 onto $\mathcal{C}$.

The complete set of results for our four variants are given in Appendix A. It is immediately clear that none of these methods is generally reliable or fast. We break down the failures for each variant in Table 4.1.

variant	$> 10^6$ iterations	> 1800 CPU seconds
A3.1(α=1)	36	15
A3.1(α=opt)	29	15
A3.2(α=1)	40	18
A3.2(α=opt)	32	15

Table 4.1: Types of failures for each algorithm tested.

By way of comparison, in Appendix B we give results for `LSQP`, which solves all 120 problems with the exception of the (allegedly) infeasible `BCDOUT` and `MPSBCD03`. To illustrate the relative performance of the five methods, we plot their CPU time performance profiles [12], for the 118 problems solved by at least one of the methods, in Figure 4.1. It is

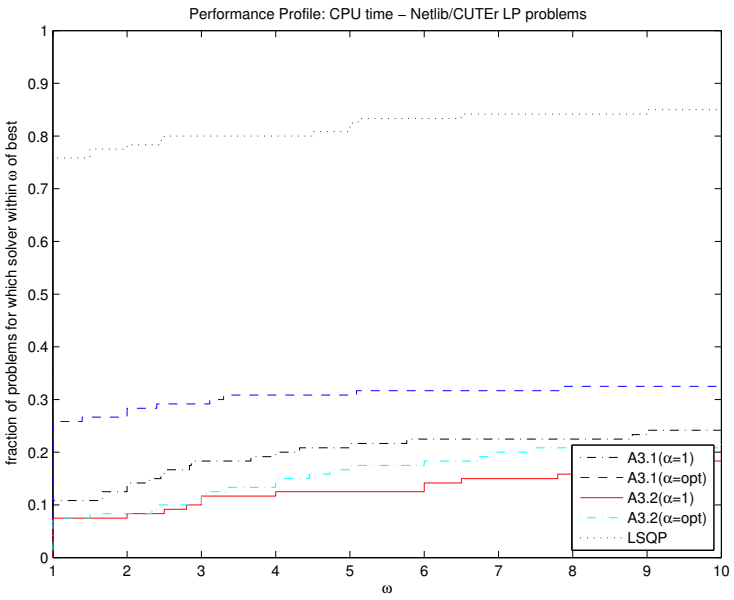

Figure 4.1: Performance profile, $p(\omega)$ including `LSQP`: CPU time for the 118 problems under consideration.

immediately clear that, at least for this test set, the interior-point method is vastly superior

to all four projection methods. Closer scrutiny of the detailed results in the appendices indicates that even for the subset of problems for which all succeed, the interior-point approach is still a clear overall winner.

We now turn to assessing the projection methods. Since there is little point in comparing the variants on problems for which all fail, we restrict ourselves to the 77 out of the 120 problems that were solved by at least one variant. In Figures 4.2 and 4.3, we plot the performance profiles for CPU times and iteration counts for the projection methods. These

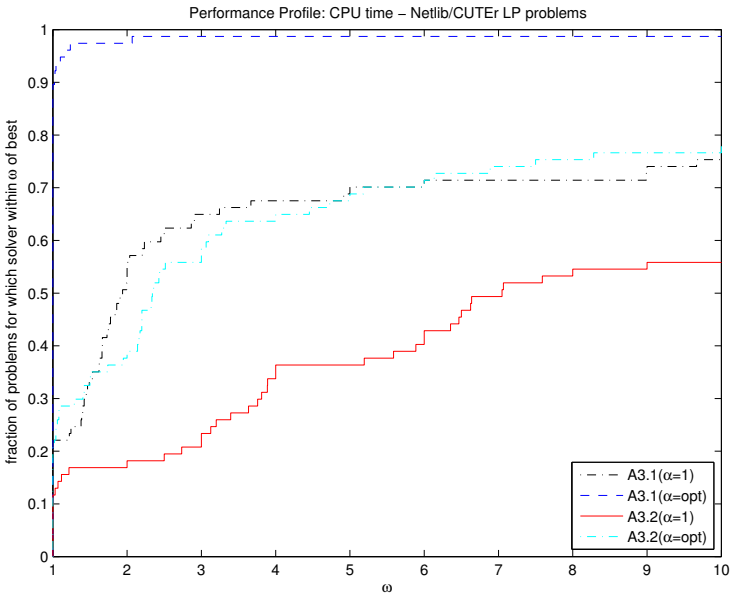

Figure 4.2: Performance profile, $p(\omega)$: CPU time for the 77 problems under consideration.

indicate that the successive projection methods are generally superior to the simultaneous ones, and that picking the stepsize to be locally optimal is better than the traditional unit step. Again, examining the tables in Appendix A in detail, the (often pitifully slow) linear convergence of the methods is apparent.

5 Comments and conclusions

When we started this study, we were under the impression that projection methods would be generally applicable techniques for solving real-life problems. In particular, numerical experience with (small) random problems [4, 5, 7] suggests that modest numbers of iterations are required to achieve reasonable accuracy for methods of the sort we have described here. This is contrary to our numerical experience, and simply suggests to us that there is a significant difference between random and real-life problems (similar observations have been made for linear equations, where random problems tend to be well-conditioned [16], and thus often easier to solve than those from applications). The only real hint we found in the literature that these methods might be expensive is for an example from (N-convex)

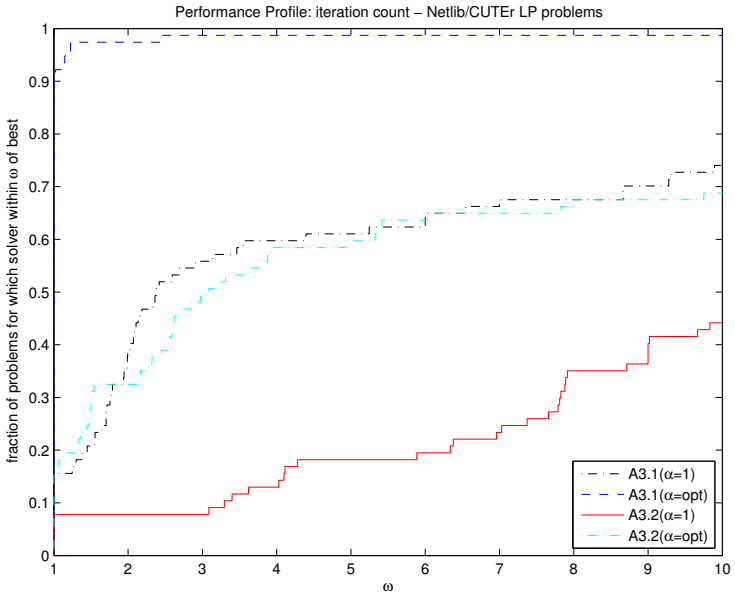

Figure 4.3: Performance profile, $p(\omega)$: Iteration counts for the 77 problems under consideration.

regression [28]. With hindsight, since for linear systems Kaczmarz's method [26] is essentially the Gauss-Seidel iteration, while Cimmino's [10] may be viewed as Jacobi's method for the normal equations, it is perhaps not surprising that these simple projection methods do not perform particularly well; some form of (low-cost) acceleration is undoubtedly and urgently needed.

We do not want to suggest here that successive and simultaneous projection methods are not useful, since in particular they appear to have been applied successfully for many years in medical imaging, radiation therapy planning and signal processing [2, 8, 9]. But our experience suggests that despite the large literature devoted to theoretical analysis, they should not be considered as the method of choice for a given application without further strong empirical evidence to support such a claim.

We also need to be cautious in extrapolating our findings on problems involving two constraint sets to the more general case. Our argument is simply that this is the setting which we had anticipated would put the methods in the best light, and that projecting into a larger number of sets would likely have increase iteration counts (if not costs).

Of course the observation that successive projection methods tend to require fewer iterations/projections than simultaneous methods is not new [5]; in defense of the latter, it is easier to exploit their natural parallelism, especially when a point in the intersection of a large number of sets is required.

Acknowledgements

The author would like to thank Coralia Cartis and Iain Duff for their helpful comments and suggestions on this work.

References

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

[2] H. H. Bauschke and J. M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Review*, 38(3):367–426, 1996.

[3] H. H. Bauschke, J. M. Borwein, and A. S. Lewis. The method of cyclic projections for closed convex sets in Hilbert space. *Contemporary Mathematics*, 204:1–38, 1997.

[4] H. H. Bauschke, P. L. Combettes, and S. G. Kruk. Extrapolation algorithm for affine-convex feasibility problems. *Numerical Algorithms*, (to appear), 2006.

[5] H. H. Bauschke and S. G. Kruk. The method of reflection-projection for convex feasibility problems with an obtuse cone. *Journal of Optimization Theory and Applications*, 120(3):503–531, 2004.

[6] Y. Censor. Row-action methods for huge and sparse systems and their applications. *SIAM Review*, 23(4):444–466, 1981.

[7] Y. Censor. Computational acceleration of projection algorithms for the linear best approximation problem. *Linear Algebra and its Applications*, 416:111–123, 2006.

[8] Y. Censor and G. T. Herman. On some optimization techniques in image reconstruction from projections. *Applied Numerical Mathematics*, 3:365–391, 1987.

[9] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms and Applications*. Oxford University Press, Oxford, 1997.

[10] G. Cimmino. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. *Ricerca Scientifica, XVI Seria II, Anno IX*, 1:326–333, 1938.

[11] F. Deutsch and H. Hundal. The rate of convergence of Dykstra's cyclic projections algorithm: the polyhedral case. *Numerical Functional Analysis and Optimization*, 15(5/6):537–565, 1994.

[12] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[13] I. S. Duff. MA57 - a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software*, 30(2):118–144, 2004.

[14] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software*, 9(3):302–325, 1983.

[15] R. L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statatistical Association*, 78:837–842, 1983.

[16] A. Edelman. Eigenvalues and condition numbers of random matrices. *SIAM Journal on Matrix Analysis and Applications*, 9(4):543–560, 1988.

[17] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

[18] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD—a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4):353–372, 2003.

[19] L. G. Gubin, B. T. Polyak, and E. V. Raik. The method of projections for finding the common point of convex sets. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(6):1–24, 1967.

[20] I. Halperin. The product of projection operators. *Acta Scientiarum Mathematicarum*, 23:96–99, 1962.

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

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

[23] HSL. A collection of Fortran codes for large-scale scientific computation, 2004. See http://www.cse.clrc.ac.uk/nag/hsl/.

[24] A. N. Iusem and A. R. De Pierro. A simultaneous iterative method for computing projections on polyhedra. *SIAM Journal on Control and Optimization*, 25(1):231–243, 1987.

[25] A. N. Iusem and A. R. De Pierro. On the convergence of Han's method for convex-programming with quadratic objective. *Mathematical Programming, Series B*, 52(2):265–284, 1991.

[26] S. Kaczmarz. Angenährte Auflösung von Systemen linearer Gleichungen. *Bulletin internationel de l'Academie Polonaise des Sciences et des Lettres*, A35:355–357, 1937.

[27] Y. I. Merzlyakov. On a relaxation method of solving systems of linear inequalities. *U. S. S. R. Computational Mathematics and Mathematical Physics*, 2:504–510, 1962.

[28] C. Perkins. A convergence analysis of Dykstra's algorithm for polyhedral sets. *SIAM Journal on Numerical Analysis*, 40(2):792–804, 2002.

[29] G. Pierra. Decomposition through formalization in a product space. *Mathematical Programming*, 28(1):96–115, 1984.

[30] A. R. De Pierro and A. N. Iusem. A simultaneous projections method for linear inequalities. *Linear Algebra and its Applications*, 64:243–253, 1985.

[31] J. von Neumann. *Functional Operators, Volume II.* Princeton University Press, Princeton, USA, 1950.

[32] J. W. J. Williams. Algorithm 232, Heapsort. *Communications of the ACM*, 7:347–348, 1964.

Appendix A

For reference, we give details of the performance of each of the four variants tested on the complete set of Netlib and CUTEr linear programming test sets. The columns marked n and m give the numbers of variables and general linear constraints, status gives the exit status (0 for success, -10 when the iteration limit is exceeded and -11 when the CPU time limit is reached), and error gives the value of $\max(\|P_A(x)-x\|_2, \|P_B(x)-x\|_2)$ achieved on termination. A -1 in the columns recording the number of iterations required to achieve the indicated error simply indicates that this accuracy was not achieved.

Table A.1: Complete results for the variant A3.1($\alpha = 1$)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
25FV47	1571	821	0	1.0E-05	173375	224428	275456	326473	377485	674.2
80BAU3B	9799	2262	0	1.0E-05	1595	2065	2534	3004	3474	17.2
ADLITTLE	97	56	0	9.7E-06	77	133	189	244	300	0.0
AFIRO	32	27	0	9.9E-06	132	197	261	326	390	0.0
AGG2	302	516	-10	4.1E+03	-1	-1	-1	-1	1000001	746.4
AGG3	302	516	-10	2.6E+03	-1	-1	-1	-1	1000001	773.2
AGG	163	488	-10	9.6E+00	-1	-1	-1	-1	1000001	431.4
BANDM	472	305	0	1.0E-05	194566	345486	496407	647327	798248	273.7
BCDOUT	5940	5414	-11	2.2E+01	-1	-1	-1	-1	70611	1800.0
BEACONFD	262	173	0	9.9E-06	48	56	64	72	79	0.0
BLEND	83	74	0	1.0E-05	20852	28255	35657	43060	50463	4.1
BNL1	1175	643	-10	1.6E+00	-1	-1	-1	-1	1000001	897.9
BNL2	3489	2324	-11	2.3E-01	-1	-1	-1	-1	432602	1800.0
BOEING1	384	351	-10	1.6E+03	-1	-1	-1	-1	1000001	493.2
BOEING2	143	166	-10	4.0E+01	-1	-1	-1	-1	1000001	180.5
BORE3D	315	233	-10	1.9E+02	-1	-1	-1	-1	1000001	224.9
BRANDY	249	220	0	1.0E-05	167449	208991	250533	292075	333617	87.0
CAPRI	353	271	-10	3.9E+01	-1	-1	-1	-1	1000001	309.0
CYCLE	2857	1903	-10	2.0E+03	-1	-1	-1	-1	1000001	3997.9
CZPROB	3523	929	-10	1.0E+01	-1	-1	-1	-1	1000001	1334.6
D2Q06C	5167	2171	-11	6.5E+03	-1	-1	-1	-1	257829	1800.0
D6CUBE	6184	415	0	9.9E-06	359	529	700	870	1041	112.0
DEGEN2	534	444	0	9.7E-06	138	200	262	323	385	0.3
DEGEN3	1818	1503	0	1.0E-05	3261	9205	15767	23024	30281	174.7

Table A.1: Complete results for the variant A3.1($\alpha = 1$) (continued)

name	n	m	status	error	iterations until error < 0.1	0.01	0.001	0.0001	0.00001	CPU time
DEGENLPA	20	15	-10	3.1E-03	2	2	-1	-1	1000001	15.9
DEGENLPB	20	15	-10	3.1E-03	2	2	-1	-1	1000001	15.9
DFL001	12230	6071	0	1.0E-05	1301	2073	2843	3614	4385	193.6
E226	282	223	-10	5.0E-04	373552	645677	917711	-1	1000001	326.9
ETAMACRO	688	400	-10	8.3E+00	-1	-1	-1	-1	1000001	430.2
FFFFF800	854	524	-10	4.8E+02	-1	-1	-1	-1	1000001	876.2
FINNIS	614	497	0	1.0E-05	91072	165011	239126	313170	387207	183.0
FIT1D	1026	24	-10	1.3E+04	-1	-1	-1	-1	1000001	1057.4
FIT1P	1677	627	-10	1.8E+01	-1	-1	-1	-1	1000001	1986.0
FIT2D	10500	25	-11	2.2E+00	-1	-1	-1	-1	157135	1800.0
FIT2P	13525	3000	-11	8.7E+00	-1	-1	-1	-1	248297	1800.0
FORPLAN	421	161	-10	8.4E+01	-1	-1	-1	-1	1000001	405.7
GANGES	1681	1309	0	1.0E-05	1570	1953	6454	20280	33688	50.8
GFRD-PNC	1092	616	-10	7.1E+03	-1	-1	-1	-1	1000001	428.7
GOFFIN	51	50	0	0.0E+00	1	1	1	1	1	0.0
GREENBEA	5405	2392	-11	1.6E+01	-1	-1	-1	-1	342880	1800.0
GREENBEB	5405	2392	-11	2.1E+01	-1	-1	-1	-1	328029	1800.0
GROW15	645	300	0	0.0E+00	1	1	1	1	1	0.0
GROW22	946	440	0	0.0E+00	1	1	1	1	1	0.0
GROW7	301	140	0	0.0E+00	1	1	1	1	1	0.0
ISRAEL	142	174	-10	1.0E+05	-1	-1	-1	-1	1000001	315.6
KB2	41	43	0	1.0E-05	265378	362081	458781	555480	652180	33.0
LINSPANH	97	33	0	9.9E-06	110	181	327	466	605	0.0
LOTFI	308	153	0	1.0E-05	249095	312330	375564	438799	502033	82.5
MAKELA4	21	40	0	0.0E+00	1	1	1	1	1	0.0
MAROS-R7	9408	3136	0	9.2E-06	75	92	110	128	146	9.2
MAROS	1443	846	-10	1.8E+05	-1	-1	-1	-1	1000001	1597.8
MODSZK1	1620	687	0	9.8E-06	348	464	581	697	814	0.6
MPSBCD03	5940	5414	-11	6.5E+01	-1	-1	-1	-1	73671	1800.0
NESM	2923	662	-10	3.6E-05	309522	507157	708215	909416	1000001	1849.4
OET1	3	1002	0	1.0E-05	7	15	41	128	392	0.1
OET3	4	1002	0	1.0E-05	3	6	18	63	206	0.1
PEROLD	1376	625	-10	3.7E+04	-1	-1	-1	-1	1000001	1150.3
PILOT4	1000	410	-10	9.6E+02	-1	-1	-1	-1	1000001	731.5
PILOT87	4883	2030	-11	2.4E+01	-1	-1	-1	-1	97421	1800.0
PILOT-JA	1988	940	-11	4.5E+04	-1	-1	-1	-1	309142	1800.0
PILOTNOV	2172	975	-11	5.8E+03	-1	-1	-1	-1	463143	1800.0
PILOT	3652	1441	-11	1.7E+00	-1	-1	-1	-1	215132	1800.0
PILOT-WE	2789	722	-10	2.1E+04	-1	-1	-1	-1	1000001	1474.1
PT	2	501	0	9.9E-06	2	9	36	126	413	0.1
QAP12	8856	3192	0	2.2E-16	2	2	2	2	2	6.6
QAP15	22275	6330	0	2.2E-16	2	2	2	2	2	42.7
QAP8	1632	912	0	1.1E-16	2	2	2	2	2	0.2
QPBD_OUT	263	211	-10	5.3E-04	404741	665957	927172	-1	1000001	305.7
READING2	6003	4000	0	1.0E-05	9841	16072	22304	28535	34766	131.3
RECIPELP	180	91	0	1.0E-05	193232	308332	423422	538502	653574	59.0
S277-280	4	4	0	9.7E-06	4	27	71	114	158	0.0
SC105	103	105	0	9.9E-06	31	63	99	139	182	0.0
SC205	203	205	0	9.9E-06	31	104	204	312	419	0.1
SC50A	48	50	0	1.0E-05	29	47	68	100	135	0.0
SC50B	48	50	0	9.9E-06	66	100	137	193	254	0.0
SCAGR25	500	471	0	1.0E-05	24126	41945	59764	77583	95402	31.7
SCAGR7	140	129	0	1.0E-05	2654	3809	4964	6120	7275	0.7
SCFXM1	457	330	0	1.0E-05	134264	172493	210722	248952	287181	108.8
SCFXM2	914	660	-10	2.5E+00	-1	-1	-1	-1	1000001	800.2
SCFXM3	1371	990	-10	2.5E+00	-1	-1	-1	-1	1000001	1268.5
SCORPION	358	388	0	9.8E-06	26	93	188	285	383	0.1
SCRS8	1169	490	-10	4.9E+01	-1	-1	-1	-1	1000001	516.6
SCSD1	760	77	0	7.6E-06	5	8	11	15	18	0.0

Table A.1: Complete results for the variant A3.1($\alpha = 1$) (continued)

name	n	m	status	error	iterations until error < 0.1	0.01	0.001	0.0001	0.00001	CPU time
SCSD6	1350	147	0	7.6E-06	5	8	11	15	18	0.0
SCSD8	2750	397	0	6.2E-06	8	11	14	17	21	0.1
SCTAP1	480	300	0	1.0E-05	11369	21901	32453	43008	53564	14.5
SCTAP2	1880	1090	0	1.0E-05	11833	19277	28120	37825	47484	58.6
SCTAP3	2480	1480	0	1.0E-05	13517	21398	30471	40448	50630	87.2
SEBA	1028	515	-10	1.4E+01	-1	-1	-1	-1	1000001	440.1
SHARE1B	225	117	-10	3.0E+03	-1	-1	-1	-1	1000001	152.7
SHARE2B	79	96	0	1.0E-05	70958	97385	123812	150240	176667	19.3
SHELL	1775	536	0	1.0E-05	2944	3867	4790	5713	6636	3.5
SHIP04L	2118	402	0	1.0E-05	22022	69012	116258	163545	210837	164.2
SHIP04S	1458	402	0	1.0E-05	22624	73064	124103	175142	226182	133.3
SHIP08L	4283	778	0	1.0E-05	4705	17631	30591	43550	56510	100.6
SHIP08S	2387	778	0	1.0E-05	5010	18498	32076	45655	59235	61.4
SHIP12L	5427	1151	0	1.0E-05	11974	121450	247445	373445	499446	1217.0
SHIP12S	2763	1151	0	1.0E-05	12314	122003	248287	374573	500859	677.9
SIERRA	2036	1227	0	9.9E-06	352	477	601	725	850	1.1
SIPOW1M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW1	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW3	4	2000	0	1.0E-05	13	141	932	1969	3016	2.4
SIPOW4	4	2000	0	1.0E-05	6	78	493	1107	1739	1.5
SSEBLIN	194	72	0	1.0E-05	883	1168	1452	1737	2021	0.1
STAIR	467	356	0	1.0E-05	2632	3722	4811	5900	6989	4.0
STANDATA	1075	359	-10	1.7E+00	-1	-1	-1	-1	1000001	432.1
STANDGUB	1184	361	-10	1.7E+00	-1	-1	-1	-1	1000001	455.1
STANDMPS	1075	467	-10	6.0E-04	617553	789533	961512	-1	1000001	518.8
STOCFOR1	111	117	0	9.9E-06	269	328	386	444	502	0.1
STOCFOR2	2031	2157	0	1.0E-05	55584	82085	108585	135084	161584	344.6
STOCFOR3	15695	16675	-11	3.5E-01	-1	-1	-1	-1	81177	1800.0
TESTDECK	14	15	0	8.8E-06	6	11	16	21	26	0.0
TFI2	3	101	0	1.0E-05	3	24	70	118	231	0.0
TRUSS	8806	1000	0	5.6E-06	12	16	19	22	26	0.3
TUFF	587	333	-10	6.1E+03	-1	-1	-1	-1	1000001	516.1
VTP-BASE	203	198	-10	1.2E+03	-1	-1	-1	-1	1000001	164.9
WOOD1P	2594	244	-11	2.5E-02	215461	-1	-1	-1	259417	1800.0
WOODW	8405	1098	-11	1.2E-04	88560	115112	141664	-1	166484	1800.0

Table A.2: Complete results for the variant A3.1(α =opt)

name	n	m	status	error	iterations until error < 0.1	0.01	0.001	0.0001	0.00001	CPU time
25FV47	1571	821	0	1.0E-05	83209	128140	178742	229332	279917	502.8
80BAU3B	9799	2262	0	9.9E-06	788	1020	1251	1481	1677	9.8
ADLITTLE	97	56	0	8.4E-06	35	53	77	95	127	0.0
AFIRO	32	27	0	9.7E-06	56	83	110	137	164	0.0
AGG2	302	516	-10	1.6E+02	-1	-1	-1	-1	1000001	800.3
AGG3	302	516	-10	9.8E-01	-1	-1	-1	-1	1000001	804.2
AGG	163	488	-10	1.8E+01	-1	-1	-1	-1	1000001	480.1
BANDM	472	305	0	1.0E-05	69745	161407	312327	463248	614168	220.0
BCDOUT	5940	5414	-11	2.2E+01	-1	-1	-1	-1	73763	1800.0
BEACONFD	262	173	0	8.1E-06	24	28	32	35	37	0.0
BLEND	83	74	0	1.0E-05	3971	5233	8271	15676	23079	2.0
BNL1	1175	643	-10	9.8E-01	-1	-1	-1	-1	1000001	967.7
BNL2	3489	2324	-11	9.3E-02	367516	-1	-1	-1	449957	1800.0
BOEING1	384	351	-10	1.2E+03	-1	-1	-1	-1	1000001	543.8
BOEING2	143	166	-10	4.0E+01	-1	-1	-1	-1	1000001	189.5
BORE3D	315	233	-10	8.6E+01	-1	-1	-1	-1	1000001	239.0
BRANDY	249	220	0	1.0E-05	77228	105500	147042	188584	230126	61.1

Table A.2: Complete results for the variant A3.1(α =opt) (continued)

name	n	m	status	error	iterations until error < 0.1	0.01	0.001	0.0001	0.00001	CPU time
CAPRI	353	271	-10	1.2E+00	-1	-1	-1	-1	1000001	331.3
CYCLE	2857	1903	-11	2.1E+03	-1	-1	-1	-1	430052	1800.0
CZPROB	3523	929	-10	2.1E+00	-1	-1	-1	-1	1000001	1462.2
D2Q06C	5167	2171	-11	8.8E+02	-1	-1	-1	-1	243723	1800.0
D6CUBE	6184	415	0	9.9E-06	182	267	353	437	607	114.6
DEGEN2	534	444	0	9.3E-06	52	75	97	121	143	0.1
DEGEN3	1818	1503	0	1.0E-05	1361	4739	8378	14969	24273	142.9
DEGENLPA	20	15	-10	3.1E-03	2	2	-1	-1	1000001	15.3
DEGENLPB	20	15	-10	3.1E-03	2	2	-1	-1	1000001	15.4
DFL001	12230	6071	0	1.0E-05	635	1015	1396	1779	2261	109.0
E226	282	223	0	1.0E-05	90143	146112	202028	257940	313851	106.2
ETAMACRO	688	400	-10	4.2E+00	-1	-1	-1	-1	1000001	449.7
FFFFF800	854	524	-10	5.4E+01	-1	-1	-1	-1	1000001	933.3
FINNIS	614	497	0	1.0E-05	45307	78929	99746	118514	166047	83.0
FIT1D	1026	24	-10	1.2E+04	-1	-1	-1	-1	1000001	1150.3
FIT1P	1677	627	-11	1.5E+01	-1	-1	-1	-1	883478	1800.0
FIT2D	10500	25	-11	9.5E-03	112585	152790	-1	-1	153662	1800.0
FIT2P	13525	3000	-11	8.4E+00	-1	-1	-1	-1	259068	1800.0
FORPLAN	421	161	-10	2.1E+01	-1	-1	-1	-1	1000001	430.5
GANGES	1681	1309	0	1.0E-05	544	603	680	1133	1671	2.8
GFRD-PNC	1092	616	-10	3.6E+03	-1	-1	-1	-1	1000001	495.5
GOFFIN	51	50	0	0.0E+00	1	1	1	1	1	0.0
GREENBEA	5405	2392	-11	1.0E+00	-1	-1	-1	-1	325198	1800.0
GREENBEB	5405	2392	-11	8.0E+00	-1	-1	-1	-1	310565	1800.0
GROW15	645	300	0	0.0E+00	1	1	1	1	1	0.0
GROW22	946	440	0	0.0E+00	1	1	1	1	1	0.0
GROW7	301	140	0	0.0E+00	1	1	1	1	1	0.0
ISRAEL	142	174	-10	3.8E+04	-1	-1	-1	-1	1000001	341.0
KB2	41	43	0	1.0E-05	45761	129276	225976	322675	419394	22.1
LINSPANH	97	33	0	1.0E-05	53	84	160	229	298	0.0
LOTFI	308	153	0	1.0E-05	35918	62965	126076	189188	252299	42.5
MAKELA4	21	40	0	0.0E+00	1	1	1	1	1	0.0
MAROS-R7	9408	3136	0	8.9E-06	38	47	57	66	75	6.6
MAROS	1443	846	-10	1.1E+05	-1	-1	-1	-1	1000001	1736.4
MODSZK1	1620	687	0	9.6E-06	128	159	179	209	235	0.2
MPSBCD03	5940	5414	-11	6.5E+01	-1	-1	-1	-1	73114	1800.0
NESM	2923	662	0	1.0E-05	153695	308511	505675	703056	900441	1720.7
OET1	3	1002	0	2.2E-15	3	3	3	3	3	0.0
OET3	4	1002	0	4.0E-14	3	3	3	3	3	0.0
PEROLD	1376	625	-10	3.4E+04	-1	-1	-1	-1	1000001	1198.7
PILOT4	1000	410	-10	8.4E+02	-1	-1	-1	-1	1000001	774.8
PILOT87	4883	2030	-11	8.0E+00	-1	-1	-1	-1	92277	1800.0
PILOT-JA	1988	940	-11	4.5E+04	-1	-1	-1	-1	300455	1800.0
PILOTNOV	2172	975	-11	5.3E+03	-1	-1	-1	-1	453725	1800.0
PILOT	3652	1441	-11	8.4E-01	-1	-1	-1	-1	206525	1800.0
PILOT-WE	2789	722	-10	2.1E+04	-1	-1	-1	-1	1000001	1650.2
PT	2	501	0	1.7E-16	2	3	3	3	3	0.0
QAP12	8856	3192	0	2.2E-16	2	2	2	2	2	6.5
QAP15	22275	6330	0	2.2E-16	2	2	2	2	2	42.0
QAP8	1632	912	0	1.1E-16	2	2	2	2	2	0.2
QPBD_OUT	263	211	0	1.0E-05	87384	143035	198456	253825	309182	98.0
READING2	6003	4000	0	2.3E-12	3	3	3	3	3	0.1
RECIPELP	180	91	0	1.0E-05	28207	110365	225457	340540	455613	42.1
S277-280	4	4	0	8.9E-16	3	3	3	3	3	0.0
SC105	103	105	0	1.9E-14	4	4	4	4	4	0.0
SC205	203	205	0	2.5E-06	5	13	24	35	45	0.0
SC50A	48	50	0	3.8E-15	3	3	3	3	3	0.0
SC50B	48	50	0	9.6E-13	3	3	3	3	3	0.0
SCAGR25	500	471	0	1.0E-05	5264	8253	11205	14183	26890	9.8

Table A.2: Complete results for the variant A3.1(α =opt) (continued)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
SCAGR7	140	129	0	9.9E-06	648	889	1147	1396	1656	0.2
SCFXM1	457	330	0	1.0E-05	44192	70621	108850	147080	185309	74.1
SCFXM2	914	660	0	1.0E-05	500441	652654	653048	667405	706673	631.7
SCFXM3	1371	990	-10	4.7E-03	461737	805453	-1	-1	1000001	1378.5
SCORPION	358	388	0	5.8E-06	9	21	44	64	73	0.0
SCRS8	1169	490	-10	3.3E+00	-1	-1	-1	-1	1000001	577.4
SCSD1	760	77	0	2.4E-16	3	3	3	3	3	0.0
SCSD6	1350	147	0	6.6E-15	3	3	3	3	3	0.0
SCSD8	2750	397	0	8.4E-15	3	3	3	3	3	0.0
SCTAP1	480	300	0	1.0E-05	3582	5934	8836	19381	29926	8.7
SCTAP2	1880	1090	0	1.0E-05	651	1138	1454	2821	5117	6.5
SCTAP3	2480	1480	0	1.0E-05	2047	4075	6267	11695	17422	30.0
SEBA	1028	515	-10	2.6E-02	642786	-1	-1	-1	1000001	546.8
SHARE1B	225	117	-10	1.5E+03	-1	-1	-1	-1	1000001	167.1
SHARE2B	79	96	0	1.0E-05	14383	24418	50845	77272	103700	11.9
SHELL	1775	536	0	9.3E-06	574	679	795	912	1014	0.7
SHIP04L	2118	402	0	1.0E-05	4717	14936	25700	53363	100096	88.3
SHIP04S	1458	402	0	1.0E-05	4608	16200	28768	59470	108357	70.0
SHIP08L	4283	778	0	1.0E-05	58	2838	6347	9940	21785	45.1
SHIP08S	2387	778	0	1.0E-05	741	4528	8408	12157	24479	30.8
SHIP12L	5427	1151	0	1.0E-05	3111	29623	59749	176935	302918	792.8
SHIP12S	2763	1151	0	1.0E-05	3607	25726	51038	166910	293196	417.2
SIERRA	2036	1227	0	9.7E-06	178	240	302	366	428	0.6
SIPOW1M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW1	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW3	4	2000	0	1.4E-08	3	5	7	7	7	0.0
SIPOW4	4	2000	0	3.3E-06	3	6	8	13	15	0.0
SSEBLIN	194	72	0	9.9E-06	443	585	728	871	1013	0.1
STAIR	467	356	0	1.0E-05	1027	1290	1518	1976	3944	2.3
STANDATA	1075	359	-10	2.0E-02	742830	-1	-1	-1	1000001	466.2
STANDGUB	1184	361	-10	2.0E-02	742830	-1	-1	-1	1000001	489.0
STANDMPS	1075	467	0	1.0E-05	296372	468361	640350	812340	984329	539.9
STOCFOR1	111	117	0	9.8E-06	73	90	106	112	160	0.0
STOCFOR2	2031	2157	0	1.0E-05	1595	3349	8901	14458	20020	44.0
STOCFOR3	15695	16675	0	1.0E-05	4793	6172	10012	15539	21071	490.4
TESTDECK	14	15	0	1.3E-15	3	3	3	3	3	0.0
TFI2	3	101	0	3.3E-06	3	5	7	8	10	0.0
TRUSS	8806	1000	0	2.0E-13	3	3	3	3	3	0.2
TUFF	587	333	-10	3.5E+03	-1	-1	-1	-1	1000001	554.5
VTP-BASE	203	198	-10	9.6E+02	-1	-1	-1	-1	1000001	179.8
WOOD1P	2594	244	-11	1.9E-03	170874	244189	-1	-1	296471	1800.0
WOODW	8405	1098	0	1.0E-05	53878	80431	106983	133535	160088	1174.5

Table A.3: Complete results for the variant A3.2($\alpha = 1$)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
25FV47	1571	821	0	1.0E-05	180749	378928	581830	785653	989661	1658.2
80BAU3B	9799	2262	0	1.0E-05	4843	6723	8603	10482	12362	57.9
ADLITTLE	97	56	0	1.0E-05	188	308	507	780	1589	0.1
AFIRO	32	27	0	9.9E-06	260	519	779	1038	1298	0.0
AGG2	302	516	-10	1.7E+02	-1	-1	-1	-1	1000001	701.7
AGG3	302	516	-10	1.6E+02	-1	-1	-1	-1	1000001	691.9
AGG	163	488	-10	5.5E+00	-1	-1	-1	-1	1000001	404.8
BANDM	472	305	-10	7.0E-04	166725	405874	904943	-1	1000001	327.3
BCDOUT	5940	5414	-11	1.6E+01	-1	-1	-1	-1	77041	1800.0
BEACONFD	262	173	0	9.5E-06	108	139	171	203	236	0.1

Table A.3: Complete results for the variant A3.2($\alpha = 1$) (continued)

name	n	m	status	error	iterations until error < 0.1	0.01	0.001	0.0001	0.00001	CPU time
BLEND	83	74	0	1.0E-05	28105	57426	87042	116658	146274	11.2
BNL1	1175	643	-10	5.4E-01	-1	-1	-1	-1	1000001	832.7
BNL2	3489	2324	-11	3.4E-01	-1	-1	-1	-1	499798	1800.0
BOEING1	384	351	-10	7.8E-01	-1	-1	-1	-1	1000001	469.1
BOEING2	143	166	-10	1.1E-02	251241	-1	-1	-1	1000001	171.5
BORE3D	315	233	-10	1.7E+00	-1	-1	-1	-1	1000001	214.8
BRANDY	249	220	0	1.0E-05	282088	447816	613984	780153	946321	233.1
CAPRI	353	271	-10	9.7E-01	-1	-1	-1	-1	1000001	296.7
CYCLE	2857	1903	-11	5.9E+00	-1	-1	-1	-1	480986	1800.0
CZPROB	3523	929	-10	2.8E+00	-1	-1	-1	-1	1000001	1231.2
D2Q06C	5167	2171	-11	5.1E+01	-1	-1	-1	-1	266919	1800.0
D6CUBE	6184	415	0	1.0E-05	72	448	972	1516	2064	115.5
DEGEN2	534	444	0	1.0E-05	416	664	911	1159	1406	0.9
DEGEN3	1818	1503	0	1.0E-05	1115	16815	41019	70245	99471	556.1
DEGENLPA	20	15	-10	2.7E-05	4	10	17	26	1000001	13.6
DEGENLPB	20	15	-10	2.7E-05	4	10	17	26	1000001	13.7
DFL001	12230	6071	0	1.0E-05	3705	6719	9774	12833	15892	692.6
E226	282	223	-10	1.5E-04	164642	390665	663872	-1	1000001	310.7
ETAMACRO	688	400	-10	2.0E-01	-1	-1	-1	-1	1000001	466.2
FFFFF800	854	524	-10	4.3E+01	-1	-1	-1	-1	1000001	800.5
FINNIS	614	497	-10	3.8E-04	279520	576024	874989	-1	1000001	449.3
FIT1D	1026	24	-10	9.4E-01	-1	-1	-1	-1	1000001	960.4
FIT1P	1677	627	-11	7.3E-03	5347	15848	-1	-1	975909	1800.0
FIT2D	10500	25	-11	2.9E-01	-1	-1	-1	-1	137872	1800.1
FIT2P	13525	3000	-11	2.8E-03	5431	20771	-1	-1	293220	1800.0
FORPLAN	421	161	-10	2.5E+00	-1	-1	-1	-1	1000001	386.1
GANGES	1681	1309	0	1.0E-05	6885	8449	20355	66147	119036	167.2
GFRD-PNC	1092	616	-10	1.4E+01	-1	-1	-1	-1	1000001	410.1
GOFFIN	51	50	0	0.0E+00	1	1	1	1	1	0.0
GREENBEA	5405	2392	-11	3.6E+00	-1	-1	-1	-1	368553	1800.0
GREENBEB	5405	2392	-11	4.0E+00	-1	-1	-1	-1	362425	1800.0
GROW15	645	300	0	5.2E-06	9	13	16	19	23	0.0
GROW22	946	440	0	6.2E-06	10	13	16	19	23	0.0
GROW7	301	140	0	7.3E-06	9	12	15	19	22	0.0
ISRAEL	142	174	-10	5.3E+01	-1	-1	-1	-1	1000001	295.2
KB2	41	43	-10	4.1E-04	118791	468377	851403	-1	1000001	47.9
LINSPANH	97	33	0	1.0E-05	392	574	961	1517	2074	0.1
LOTFI	308	153	-10	4.2E-04	399540	652476	905412	-1	1000001	156.2
MAKELA4	21	40	0	0.0E+00	1	1	1	1	1	0.0
MAROS-R7	9408	3136	0	9.9E-06	309	379	449	520	591	24.8
MAROS	1443	846	-10	3.9E+01	-1	-1	-1	-1	1000001	1502.9
MODSZK1	1620	687	0	1.0E-05	1189	1644	2105	2570	3035	2.2
MPSBCD03	5940	5414	-11	6.5E+01	-1	-1	-1	-1	78467	1800.0
NESM	2923	662	-10	5.8E-03	393910	877389	-1	-1	1000001	1722.5
OET1	3	1002	0	1.0E-05	37	83	209	531	1355	0.5
OET3	4	1002	0	1.0E-05	16	42	115	307	797	0.3
PEROLD	1376	625	-10	1.6E+01	-1	-1	-1	-1	1000001	1058.5
PILOT4	1000	410	-10	2.8E-01	-1	-1	-1	-1	1000001	718.0
PILOT87	4883	2030	-11	2.4E+00	-1	-1	-1	-1	98392	1800.0
PILOT-JA	1988	940	-11	2.6E+00	-1	-1	-1	-1	308957	1800.0
PILOTNOV	2172	975	-11	1.2E+00	-1	-1	-1	-1	482334	1800.0
PILOT	3652	1441	-11	9.0E-01	-1	-1	-1	-1	226390	1800.0
PILOT-WE	2789	722	-10	2.8E+00	-1	-1	-1	-1	1000001	1407.7
PT	2	501	0	1.0E-05	21	72	206	549	1415	0.2
QAP12	8856	3192	0	9.3E-06	5	8	12	15	18	7.3
QAP15	22275	6330	0	9.3E-06	5	8	12	15	18	44.9
QAP8	1632	912	0	9.3E-06	5	8	12	15	18	0.3
QPBD_OUT	263	211	-10	7.6E-05	163602	388587	626336	950003	1000001	286.6
READING2	6003	4000	0	1.0E-05	243	1210	5797	24378	49304	177.4

Table A.3: Complete results for the variant A3.2($\alpha = 1$) (continued)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
RECIPELP	180	91	-10	1.2E-04	16257	129940	585861	-1	1000001	86.2
S277-280	4	4	0	9.9E-06	15	69	239	414	590	0.0
SC105	103	105	0	1.0E-05	70	152	285	428	570	0.0
SC205	203	205	0	9.9E-06	70	166	410	739	1068	0.1
SC50A	48	50	0	9.7E-06	72	146	224	302	381	0.0
SC50B	48	50	0	9.9E-06	141	280	420	561	702	0.0
SCAGR25	500	471	0	1.0E-05	119972	191249	262526	333803	405080	127.3
SCAGR7	140	129	0	1.0E-05	10651	15466	20281	25096	29911	2.5
SCFXM1	457	330	0	1.0E-05	135099	288017	440935	593853	746771	269.4
SCFXM2	914	660	-10	4.9E+00	-1	-1	-1	-1	1000001	760.3
SCFXM3	1371	990	-10	7.0E+00	-1	-1	-1	-1	1000001	1245.7
SCORPION	358	388	0	1.0E-05	150	416	755	1131	1517	0.4
SCRS8	1169	490	-10	8.9E-02	837270	-1	-1	-1	1000001	482.1
SCSD1	760	77	0	9.6E-06	13	28	42	57	71	0.0
SCSD6	1350	147	0	9.8E-06	21	36	50	65	79	0.1
SCSD8	2750	397	0	8.8E-06	37	51	66	80	95	0.1
SCTAP1	480	300	0	1.0E-05	5707	20669	42507	66117	92420	23.9
SCTAP2	1880	1090	0	1.0E-05	6475	33412	62762	93231	124536	142.6
SCTAP3	2480	1480	0	1.0E-05	9316	38023	69214	101873	135700	212.1
SEBA	1028	515	-10	3.3E+00	-1	-1	-1	-1	1000001	493.7
SHARE1B	225	117	-10	2.1E+01	-1	-1	-1	-1	1000001	145.1
SHARE2B	79	96	0	1.0E-05	44682	131864	235962	340067	444172	46.3
SHELL	1775	536	0	1.0E-05	9224	12918	16612	20306	23999	12.2
SHIP04L	2118	402	0	1.0E-05	34227	221723	410870	600371	789880	586.2
SHIP04S	1458	402	0	1.0E-05	34823	233815	437972	642130	846289	463.5
SHIP08L	4283	778	0	1.0E-05	3480	55095	106923	158762	210602	342.0
SHIP08S	2387	778	0	1.0E-05	4206	57842	112153	166471	220789	216.9
SHIP12L	5427	1151	-11	1.9E-03	37906	408713	-1	-1	777499	1800.0
SHIP12S	2763	1151	-10	6.8E-04	38888	410604	915817	-1	1000001	1254.3
SIERRA	2036	1227	0	1.0E-05	1284	1783	2281	2780	3279	4.0
SIPOW1M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW1	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW3	4	2000	0	1.0E-05	108	576	2851	6870	11101	8.2
SIPOW4	4	2000	0	1.0E-05	45	269	1279	3714	6249	4.8
SSEBLIN	194	72	0	1.0E-05	3369	4509	5648	6788	7928	0.5
STAIR	467	356	0	1.0E-05	5790	10143	14500	18858	23215	11.9
STANDATA	1075	359	-10	2.1E-01	-1	-1	-1	-1	1000001	418.9
STANDGUB	1184	361	-10	2.1E-01	-1	-1	-1	-1	1000001	1602.8
STANDMPS	1075	467	-10	1.6E-02	465855	-1	-1	-1	1000001	500.9
STOCFOR1	111	117	0	9.9E-06	458	692	926	1160	1394	0.1
STOCFOR2	2031	2157	0	1.0E-05	2697	14712	27745	40778	53812	111.4
STOCFOR3	15695	16675	-11	9.6E-03	12045	84953	-1	-1	86553	1800.0
TESTDECK	14	15	0	9.9E-06	20	47	74	100	127	0.0
TFI2	3	101	0	1.0E-05	10	62	230	442	886	0.1
TRUSS	8806	1000	0	9.5E-06	55	70	84	99	113	0.6
TUFF	587	333	-10	4.6E+00	-1	-1	-1	-1	1000001	485.8
VTP-BASE	203	198	-10	1.0E+01	-1	-1	-1	-1	1000001	157.4
WOOD1P	2594	244	-11	2.7E-04	13	78	167723	-1	334070	1800.0
WOODW	8405	1098	-11	2.8E-05	6635	37414	116366	219982	278510	1800.0

Table A.4: Complete results for the variant A3.2(α =opt)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
25FV47	1571	821	0	1.0E-05	90067	153857	198699	224294	273241	488.4
80BAU3B	9799	2262	0	9.9E-06	2383	3303	4227	5106	5554	30.4
ADLITTLE	97	56	0	1.0E-05	101	161	255	369	688	0.1

Table A.4: Complete results for the variant A3.2(α =opt) (continued)

name	n	m	status	error	iterations until error < 0.1	0.01	0.001	0.0001	0.00001	CPU time
AFIRO	32	27	0	1.0E-05	130	259	388	518	644	0.0
AGG2	302	516	-10	2.7E+01	-1	-1	-1	-1	1000001	739.6
AGG3	302	516	-10	3.8E+01	-1	-1	-1	-1	1000001	750.1
AGG	163	488	-10	5.1E+00	-1	-1	-1	-1	1000001	450.6
BANDM	472	305	0	1.0E-05	81941	164903	277272	300781	693636	237.7
BCDOUT	5940	5414	-11	1.6E+01	-1	-1	-1	-1	74072	1800.0
BEACONFD	262	173	0	9.5E-06	46	59	71	85	97	0.0
BLEND	83	74	0	9.6E-06	14052	28738	41621	48839	52370	4.4
BNL1	1175	643	-10	3.7E-01	-1	-1	-1	-1	1000001	923.9
BNL2	3489	2324	-11	1.5E-01	-1	-1	-1	-1	457738	1800.0
BOEING1	384	351	-10	6.0E-01	-1	-1	-1	-1	1000001	513.9
BOEING2	143	166	-10	6.2E-03	123587	622124	-1	-1	1000001	189.2
BORE3D	315	233	-10	7.9E-01	-1	-1	-1	-1	1000001	228.6
BRANDY	249	220	0	1.0E-05	140071	216979	259187	275671	348119	106.4
CAPRI	353	271	-10	1.7E-01	-1	-1	-1	-1	1000001	321.5
CYCLE	2857	1903	-11	4.7E+00	-1	-1	-1	-1	432253	1800.0
CZPROB	3523	929	-10	1.6E+00	-1	-1	-1	-1	1000001	1423.4
D2Q06C	5167	2171	-11	3.1E+01	-1	-1	-1	-1	250480	1800.0
D6CUBE	6184	415	0	9.6E-06	40	225	485	680	811	112.0
DEGEN2	534	444	0	9.9E-06	209	333	457	582	705	0.5
DEGEN3	1818	1503	0	1.0E-05	555	8234	15819	21390	25976	148.8
DEGENLPA	20	15	-10	2.7E-05	4	6	9	13	1000001	15.1
DEGENLPB	20	15	-10	2.7E-05	4	6	9	13	1000001	15.0
DFL001	12230	6071	0	1.0E-05	1840	3333	4715	5656	6154	274.2
E226	282	223	-10	1.1E-05	83426	167804	253590	285813	1000001	327.9
ETAMACRO	688	400	-10	9.0E-02	866299	-1	-1	-1	1000001	545.4
FFFFF800	854	524	-10	6.8E+00	-1	-1	-1	-1	1000001	949.2
FINNIS	614	497	0	1.0E-05	127132	218760	287340	342339	512275	254.4
FIT1D	1026	24	-10	9.2E-01	-1	-1	-1	-1	1000001	1046.2
FIT1P	1677	627	-11	6.0E-03	2160	6394	-1	-1	919920	1800.0
FIT2D	10500	25	-11	5.9E-03	106263	151151	-1	-1	159259	1800.0
FIT2P	13525	3000	-11	2.7E-03	2658	9394	-1	-1	262115	1800.0
FORPLAN	421	161	-10	9.9E-01	-1	-1	-1	-1	1000001	406.6
GANGES	1681	1309	0	1.0E-05	3436	4217	8682	25209	50188	78.2
GFRD-PNC	1092	616	-10	1.1E+01	-1	-1	-1	-1	1000001	603.7
GOFFIN	51	50	0	0.0E+00	1	1	1	1	1	0.0
GREENBEA	5405	2392	-11	1.6E+00	-1	-1	-1	-1	337336	1800.0
GREENBEB	5405	2392	-11	2.0E+00	-1	-1	-1	-1	337497	1800.0
GROW15	645	300	0	9.4E-06	6	9	22	41	67	0.1
GROW22	946	440	0	9.9E-06	6	10	25	63	146	0.1
GROW7	301	140	0	9.5E-06	6	9	18	38	67	0.0
ISRAEL	142	174	-10	4.4E+01	-1	-1	-1	-1	1000001	323.7
KB2	41	43	0	1.0E-05	59454	234273	339567	405732	647248	33.3
LINSPANH	97	33	0	9.7E-06	197	289	484	758	886	0.0
LOTFI	308	153	0	1.0E-05	194892	303813	367117	419720	546559	93.6
MAKELA4	21	40	0	0.0E+00	1	1	1	1	1	0.0
MAROS-R7	9408	3136	0	9.7E-06	154	188	222	256	291	14.3
MAROS	1443	846	-10	2.8E+01	-1	-1	-1	-1	1000001	1641.7
MODSZK1	1620	687	0	9.6E-06	496	686	880	1074	1251	1.0
MPSBCD03	5940	5414	-11	6.5E+01	-1	-1	-1	-1	76240	1800.0
NESM	2923	662	-11	1.2E-05	177911	298985	439695	559143	964285	1800.0
OET1	3	1002	0	9.7E-06	18	36	80	196	469	0.2
OET3	4	1002	0	9.9E-06	9	20	50	132	336	0.1
PEROLD	1376	625	-10	1.5E+01	-1	-1	-1	-1	1000001	1173.8
PILOT4	1000	410	-10	2.6E-01	-1	-1	-1	-1	1000001	761.6
PILOT87	4883	2030	-11	1.4E+00	-1	-1	-1	-1	95640	1800.0
PILOT-JA	1988	940	-11	2.6E+00	-1	-1	-1	-1	302848	1800.0
PILOTNOV	2172	975	-11	1.2E+00	-1	-1	-1	-1	454180	1800.0
PILOT	3652	1441	-11	7.0E-01	-1	-1	-1	-1	213275	1800.0

Table A.4: Complete results for the variant A3.2(α =opt) (continued)

name	n	m	status	error	iterations until error < 0.1	0.01	0.001	0.0001	0.00001	CPU time
PILOT-WE	2789	722	-10	2.5E+00	-1	-1	-1	-1	1000001	1598.0
PT	2	501	0	1.0E-05	12	36	96	252	640	0.1
QAP12	8856	3192	0	2.3E-14	3	3	3	3	3	6.9
QAP15	22275	6330	0	6.4E-15	3	3	3	3	3	42.9
QAP8	1632	912	0	5.3E-15	3	3	3	3	3	0.2
QPBD_OUT	263	211	-10	1.3E-05	81650	169594	272221	280824	1000001	305.2
READING2	6003	4000	0	1.0E-05	127	611	2899	12109	24385	94.0
RECIPELP	180	91	0	1.0E-05	8118	47726	168745	259012	372163	45.6
S277-280	4	4	0	9.9E-06	8	34	116	200	284	0.0
SC105	103	105	0	9.7E-06	37	81	148	220	290	0.0
SC205	203	205	0	9.8E-06	38	106	282	468	654	0.1
SC50A	48	50	0	9.8E-06	38	75	114	154	193	0.0
SC50B	48	50	0	9.9E-06	72	140	209	278	348	0.0
SCAGR25	500	471	0	1.0E-05	54896	89579	110917	127113	145284	50.6
SCAGR7	140	129	0	1.0E-05	5040	7292	9542	11802	12957	1.2
SCFXM1	457	330	0	1.0E-05	62975	118497	156561	186296	251351	95.6
SCFXM2	914	660	-10	1.8E+00	-1	-1	-1	-1	1000001	818.6
SCFXM3	1371	990	-10	2.3E+00	-1	-1	-1	-1	1000001	1312.3
SCORPION	358	388	0	1.0E-05	74	206	372	556	746	0.2
SCRS8	1169	490	-10	2.2E-02	281414	-1	-1	-1	1000001	534.2
SCSD1	760	77	0	2.9E-15	2	3	4	4	4	0.0
SCSD6	1350	147	0	8.9E-06	4	8	14	19	24	0.0
SCSD8	2750	397	0	7.5E-06	14	19	24	29	34	0.1
SCTAP1	480	300	0	1.0E-05	2795	11015	26506	35556	43985	12.4
SCTAP2	1880	1090	0	6.7E-06	2683	15104	25130	29816	31062	40.0
SCTAP3	2480	1480	0	1.0E-05	4081	17848	28084	32492	37567	65.3
SEBA	1028	515	-10	9.8E-01	-1	-1	-1	-1	1000001	690.5
SHARE1B	225	117	-10	7.5E+00	-1	-1	-1	-1	1000001	158.4
SHARE2B	79	96	0	1.0E-05	22489	62287	93839	114263	149915	16.7
SHELL	1775	536	0	9.9E-06	4363	6151	7952	9194	9887	5.9
SHIP04L	2118	402	0	1.0E-05	16755	78789	126213	173405	259458	206.6
SHIP04S	1458	402	0	1.0E-05	16877	83631	134711	179479	277756	164.9
SHIP08L	4283	778	0	1.0E-05	1762	27214	40436	53258	61863	109.9
SHIP08S	2387	778	0	1.0E-05	2143	27455	41451	54825	63955	67.6
SHIP12L	5427	1151	0	1.0E-05	18894	176311	339604	465090	703179	1697.2
SHIP12S	2763	1151	0	1.0E-05	19293	127867	262609	394938	752056	1012.8
SIERRA	2036	1227	0	9.9E-06	539	748	956	1165	1374	2.0
SIPOW1M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW1	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2M	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW2	2	2000	0	0.0E+00	1	1	1	1	1	0.0
SIPOW3	4	2000	0	9.8E-06	42	210	1030	2472	3484	2.8
SIPOW4	4	2000	0	9.9E-06	18	102	478	1388	2298	2.0
SSEBLIN	194	72	0	9.9E-06	1598	2138	2678	3218	3665	0.3
STAIR	467	356	0	1.0E-05	2879	5035	7187	8851	9653	5.3
STANDATA	1075	359	-10	1.4E-04	454498	782152	897690	-1	1000001	478.6
STANDGUB	1184	361	-10	2.3E-04	473632	807096	921795	-1	1000001	633.7
STANDMPS	1075	467	0	1.0E-05	171899	357050	379349	379430	878366	490.5
STOCFOR1	111	117	0	9.6E-06	198	298	398	498	600	0.1
STOCFOR2	2031	2157	0	1.0E-05	1292	5466	10054	14503	16329	35.6
STOCFOR3	15695	16675	0	1.0E-05	5820	9518	13112	16200	18360	418.9
TESTDECK	14	15	0	8.9E-06	11	20	29	38	47	0.0
TFI2	3	101	0	1.0E-05	6	28	96	182	312	0.0
TRUSS	8806	1000	0	5.4E-06	20	25	30	34	40	0.4
TUFF	587	333	-10	3.6E+00	-1	-1	-1	-1	1000001	517.1
VTP-BASE	203	198	-10	8.8E+00	-1	-1	-1	-1	1000001	186.0
WOOD1P	2594	244	0	1.0E-05	6	38	19602	30945	190147	1055.9
WOODW	8405	1098	0	1.0E-05	3349	12823	24953	29086	65872	567.9

Appendix B

By way of comparison, we give equivalent results for the GALAHAD separable convex quadratic programming package LSQP applied to the same test set—since an objective function is required, $\|x - x_0\|_2^2$ is used. Each run is terminated when primal and dual optimality errors are smaller than 0.00001, and thus in particular the maximum infeasibility is no larger than this value. The column headings are exactly as in Appendix A, except that iterations indicates the number of iterations/factorizations required. The two problems with return status −6 appear to be infeasible.

Table B.1: Complete results for LSQP

name	n	m	status	iterations	CPU time
25FV47	1571	821	0	109	1.2
80BAU3B	9799	2262	0	131	3.0
ADLITTLE	97	56	0	29	0.0
AFIRO	32	27	0	19	0.0
AGG2	302	516	0	61	0.3
AGG3	302	516	0	48	0.3
AGG	163	488	0	91	0.2
BANDM	472	305	0	37	0.1
BCDOUT	5940	5414	-6	486	115.9
BEACONFD	262	173	0	23	0.0
BLEND	83	74	0	33	0.0
BNL1	1175	643	0	267	1.4
BNL2	3489	2324	0	174	6.3
BOEING1	384	351	0	72	0.2
BOEING2	143	166	0	57	0.1
BORE3D	315	233	0	31	0.0
BRANDY	249	220	0	51	0.1
CAPRI	353	271	0	170	0.3
CYCLE	2857	1903	0	139	3.9
CZPROB	3523	929	0	121	0.6
D2Q06C	5167	2171	0	160	10.0
D6CUBE	6184	415	0	76	108.2
DEGEN2	534	444	0	21	0.1
DEGEN3	1818	1503	0	27	1.8
DEGENLPA	20	15	0	13	0.0
DEGENLPB	20	15	0	13	0.0
DFL001	12230	6071	0	61	174.0
E226	282	223	0	109	0.2
ETAMACRO	688	400	0	60	0.2
FFFFF800	854	524	0	65	0.3
FINNIS	614	497	0	55	0.1
FIT1D	1026	24	0	69	0.5
FIT1P	1677	627	0	18	2.7
FIT2D	10500	25	0	91	7.2
FIT2P	13525	3000	0	18	0.8
FORPLAN	421	161	0	40	0.1
GANGES	1681	1309	0	23	0.1
GFRD-PNC	1092	616	0	82	0.1
GOFFIN	51	50	0	17	0.0
GREENBEA	5405	2392	0	279	6.8
GREENBEB	5405	2392	0	505	40.4
GROW15	645	300	0	20	0.1
GROW22	946	440	0	20	0.1
GROW7	301	140	0	20	0.0
ISRAEL	142	174	0	98	0.3
KB2	41	43	0	29	0.0

Table B.1: Complete results for LSQP (continued)

name	n	m	status	iterations	CPU time
LINSPANH	97	33	0	20	0.0
LOTFI	308	153	0	24	0.0
MAKELA4	21	40	0	17	0.0
MAROS-R7	9408	3136	0	19	18.9
MAROS	1443	846	0	302	2.1
MODSZK1	1620	687	0	31	0.1
MPSBCD03	5940	5414	-6	201	41.2
NESM	2923	662	0	52	0.5
OET1	3	1002	0	16	0.0
OET3	4	1002	0	15	0.0
PEROLD	1376	625	0	4910	58.6
PILOT4	1000	410	0	176	1.1
PILOT87	4883	2030	0	355	195.7
PILOT-JA	1988	940	0	1377	27.3
PILOTNOV	2172	975	0	264	4.8
PILOT	3652	1441	0	178	25.2
PILOT-WE	2789	722	0	531	4.5
PT	2	501	0	18	0.0
QAP12	8856	3192	0	4	15.4
QAP15	22275	6330	0	4	110.5
QAP8	1632	912	0	5	0.5
QPBD_OUT	263	211	0	106	0.2
READING2	6003	4000	0	4	0.1
RECIPELP	180	91	0	47	0.0
S277-280	4	4	0	16	0.0
SC105	103	105	0	27	0.0
SC205	203	205	0	27	0.0
SC50A	48	50	0	25	0.0
SC50B	48	50	0	23	0.0
SCAGR25	500	471	0	44	0.1
SCAGR7	140	129	0	20	0.0
SCFXM1	457	330	0	84	0.1
SCFXM2	914	660	0	145	0.5
SCFXM3	1371	990	0	170	0.8
SCORPION	358	388	0	27	0.0
SCRS8	1169	490	0	61	0.1
SCSD1	760	77	0	19	0.0
SCSD6	1350	147	0	18	0.1
SCSD8	2750	397	0	17	0.1
SCTAP1	480	300	0	60	0.1
SCTAP2	1880	1090	0	65	0.4
SCTAP3	2480	1480	0	65	0.5
SEBA	1028	515	0	124	0.3
SHARE1B	225	117	0	258	0.1
SHARE2B	79	96	0	51	0.0
SHELL	1775	536	0	55	0.1
SHIP04L	2118	402	0	31	0.1
SHIP04S	1458	402	0	32	0.1
SHIP08L	4283	778	0	32	0.3
SHIP08S	2387	778	0	29	0.2
SHIP12L	5427	1151	0	41	0.4
SHIP12S	2763	1151	0	33	0.2
SIERRA	2036	1227	0	45	0.3
SIPOW1M	2	2000	0	6	0.0
SIPOW1	2	2000	0	6	0.0
SIPOW2M	2	2000	0	13	0.1
SIPOW2	2	2000	0	13	0.1
SIPOW3	4	2000	0	23	0.1
SIPOW4	4	2000	0	17	0.1
SSEBLIN	194	72	0	23	0.0

Table B.1: Complete results for LSQP (continued)

name	n	m	status	iterations	CPU time
STAIR	467	356	0	44	0.1
STANDATA	1075	359	0	50	0.5
STANDGUB	1184	361	0	50	0.5
STANDMPS	1075	467	0	71	0.3
STOCFOR1	111	117	0	31	0.0
STOCFOR2	2031	2157	0	41	0.4
STOCFOR3	15695	16675	0	62	6.3
TESTDECK	14	15	0	17	0.0
TFI2	3	101	0	16	0.0
TRUSS	8806	1000	0	17	0.6
TUFF	587	333	0	162	0.7
VTP-BASE	203	198	0	29	0.0
WOOD1P	2594	244	0	73	21.2
WOODW	8405	1098	0	67	144.2