Science & Technology Facilities Council

Technical Report
RAL-TR-2011-009

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

NIM Gould

May 2011

RAL Library
STFC Rutherford Appleton Laboratory
R61
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358- 6254

RAL-TR-2011-0xx

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

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

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.

---

[1] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, UK.
Email: nick.gould@stfc.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 EP/E053351/1.

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

May 10, 2011

1 Introduction

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

$$\text{find } x \in \mathcal{C} \stackrel{\text{def}}{=} \{x \mid Ax = b \text{ and } x^l \leq x \leq x^u\} \tag{1.1}$$

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 (1.1) and compared on the well-known Netlib set 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 (1.1), 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 §2, we describe the projection methods considered, and in §3, we report on their comparative performance on the Netlib test set. We draw further conclusions in §4.

2 Projection methods

For the problem (1.1) of interest to us, we view the feasibility problem as that of finding a point in the intersection of the two sets

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

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 (1.1) 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 feasiblity problem. Of course, there are many possible ways to formulate the feasibility problem, such as the intersection of the sets $\mathcal{C}_i = \{x \mid a_i^T x = b_i,\, x^l \leq x \leq x^u\}$, $i = 1, \ldots, m$, 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 (1.1) 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

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

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

2.2 Simultaneous bi-projection methods

An alternative general class of simultaneous bi-projection methods for (1.1) 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

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

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

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 §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, §4] for general details of our implementation.

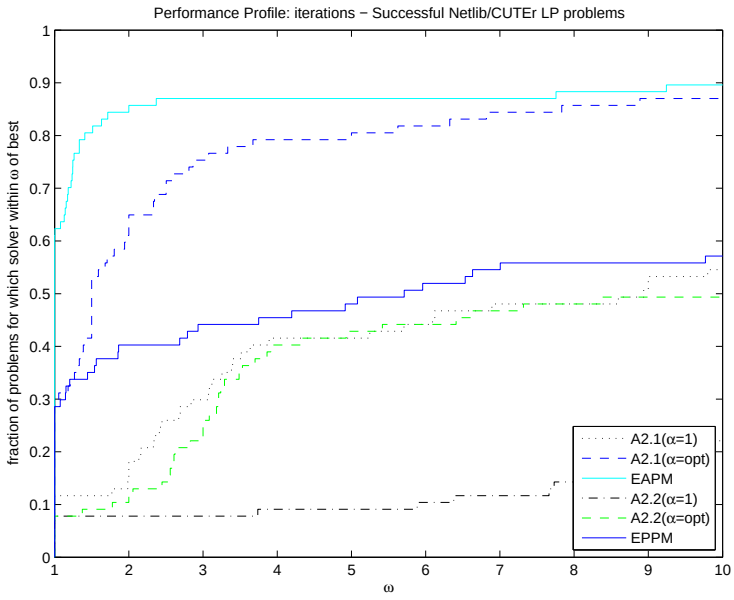

Figure 3.1: Performance profile: iteration counts for the 77 problems which are successfully solved by all variants under consideration.

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 (1.1). Our results are summarised in the iteration/projection and CPU performance profiles [6] given in Figures 3.1 and 3.2, 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) results are presented in Tables A.1 and A.2 in Appendix A.

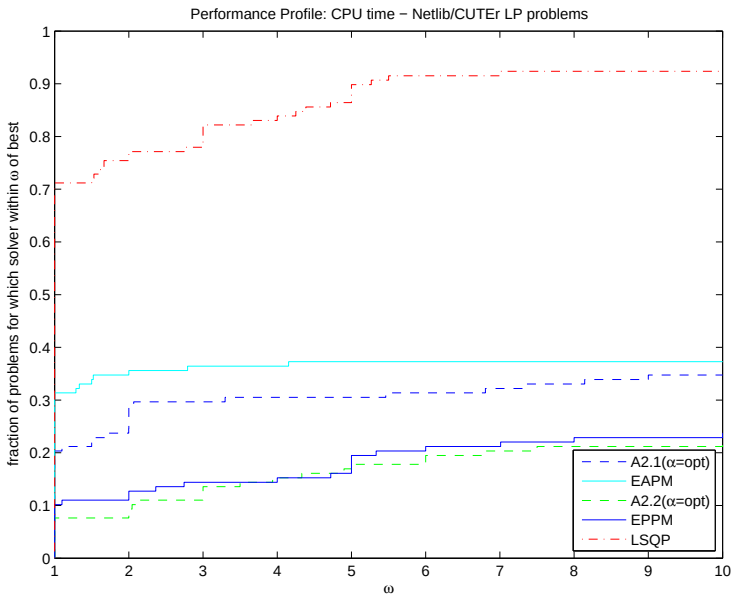

Figure 3.2: Performance profile, including `LSQP`: CPU time for the 118 problems under consideration.

These figures and tables suggest the following:

1. Figure 3.1 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.

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**($\alpha = \mathbf{opt}$), 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 Figures 3.2 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, §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**($\alpha = \mathbf{opt}$) 8 iterations and **A2.2**($\alpha = \mathbf{opt}$) 75 iterations to obtain full machine accuracy. So while we certainly accept that our earlier implemented methods are "suboptimal", the authors' suggestion [2, §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, Figures 3.2 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].

Acknowledgement

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.

References

[1] H. H. Bauschke, P. L. Combettes, and S. G. Kruk. Extrapolation algorithm for affine-convex feasibility problems. *Numerical Algorithms*, 41(3):239–274, 2006.

[2] Y. Censor, W. Chen, P. L. Combettes, R. Davidi, and G. T. Herman. On the effectiveness of projection methods for convex feasibility problems with linear inequality constraints. *Computational Optimization and Applications*, (to appear), 2011.

[3] P. L. Combettes. Hilbertian convex feasibility problem: Convergence of projection methods. *Appllied Mathematics*, 35(3):311–330, 1997.

[4] T. A. Davis and W. W. Hager. Dual multilevel optimization. *Mathematical Programming*, 112(2):403–425, 2008.

[5] T. A. Davis and W. W. Hager. A sparse proximal implementation of the lp dual active set algorithm. *Mathematical Programming*, 112(2):275–301, 2008.

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

[7] N. I. M. Gould. How good are projection methods for convex feasibility problems? *Computational Optimization and Applications*, 40(1):1–12, 2008.

[8] 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.

[9] 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.

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

Appendix A

For reference, we give details of the performance of the authors improved variants EAPM and EEPM tested on the same (complete) set of Netlib and CUTEr linear programming test sets used in [7]; these should be compared with [7, online Appendix A]. 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. As before, these additional experiments were performed on a single processor of a 3.06 GHz Dell Precision 650 Workstation. The codes are simply those used to obtain our previous results with additional input options to request EAPM and EEPM. They were compiled using full optimization with the Intel ifort compiler, and the computation was performed in double precision.

Table A.1: Complete results for EAPM

					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	22425	50171	77993	105859	133725	173.2
80BAU3B	9799	2262	0	9.9E-06	642	715	949	1184	1419	5.9
ADLITTLE	97	56	0	9.9E-06	30	42	125	213	301	0.0
AFIRO	32	27	0	7.1E-15	11	12	13	13	13	0.0
AGG2	302	516	-10	3.6E-01	-1	-1	-1	-1	1000001	701.8
AGG3	302	516	-10	1.7E-01	-1	-1	-1	-1	1000001	689.4
AGG	163	488	-10	1.3E+02	-1	-1	-1	-1	1000001	366.4
BANDM	472	305	0	1.0E-05	14722	45130	95403	170832	246262	61.4
BCDOUT	5940	5414	-11	1.6E+01	-1	-1	-1	-1	28963	1800.0
BEACONFD	262	173	0	3.6E-06	47	48	51	54	56	0.0
BLEND	83	74	0	1.0E-05	3878	7693	11508	15322	19137	1.4
BNL1	1175	643	-10	1.9E-01	-1	-1	-1	-1	1000001	627.0
BNL2	3489	2324	-11	1.7E-01	-1	-1	-1	-1	406917	1800.0
BOEING1	384	351	-10	8.6E-01	-1	-1	-1	-1	1000001	345.3
BOEING2	143	166	-10	1.1E-02	55078	-1	-1	-1	1000001	142.4
BORE3D	315	233	-10	6.9E-01	-1	-1	-1	-1	1000001	192.7
BRANDY	249	220	0	1.0E-05	30379	50611	71348	92091	112833	20.7
CAPRI	353	271	-10	1.6E-02	583824	-1	-1	-1	1000001	314.6
CYCLE	2857	1903	-11	5.0E+00	-1	-1	-1	-1	338222	1800.0
CZPROB	3523	929	-10	7.6E-01	-1	-1	-1	-1	1000001	1192.1
D2Q06C	5167	2171	-11	2.3E+01	-1	-1	-1	-1	120074	1800.0
D6CUBE	6184	415	0	7.0E-06	13	20	25	28	32	0.0
DEGEN2	534	444	-10	1.5E+20	-1	-1	-1	-1	1000001	608.7
DEGEN3	1818	1503	0	1.0E-05	41	59	293	2849	9399	27.9
DEGENLPA	20	15	-10	1.4E-05	1	1	1	1	1000001	11.2
DEGENLPB	20	15	-10	1.4E-05	1	1	1	1	1000001	11.2
DFL001	12230	6071	0	1.0E-05	510	879	1252	1626	1999	366.6
E226	282	223	-10	6.2E-05	26302	55736	101868	844064	1000001	202.6
ETAMACRO	688	400	-10	9.4E-02	905852	-1	-1	-1	1000001	569.0
FFFFF800	854	524	-10	1.4E+00	-1	-1	-1	-1	1000001	664.4
FINNIS	614	497	0	1.0E-05	34658	72280	112082	152170	192224	59.3
FIT1D	1026	24	-10	5.2E-01	-1	-1	-1	-1	1000001	492.3
FIT1P	1677	627	-10	1.0E-02	12	-1	-1	-1	1000001	1240.5
FIT2D	10500	25	0	1.0E-05	27814	47774	67730	87684	107638	541.7
FIT2P	13525	3000	-11	1.1E-02	184	-1	-1	-1	171100	1800.0
FORPLAN	421	161	-10	5.5E-01	-1	-1	-1	-1	1000001	233.8

Table A.1: Complete results for EAPM (continued)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
GANGES	1681	1309	0	1.0E-05	769	960	2704	9078	15446	13.7
GFRD-PNC	1092	616	-10	8.6E+00	-1	-1	-1	-1	1000001	289.0
GOFFIN	51	50	0	0.0E+00	1	1	1	1	1	0.0
GREENBEA	5405	2392	-11	6.2E-02	322621	-1	-1	-1	358103	1800.0
GREENBEB	5405	2392	-11	5.9E-01	-1	-1	-1	-1	347878	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	2.4E+01	-1	-1	-1	-1	1000001	210.1
KB2	41	43	0	1.0E-05	18133	66837	115008	163342	211691	6.7
LINSPANH	97	33	0	9.6E-06	73	79	89	93	106	0.0
LOTFI	308	153	0	1.0E-05	47411	78963	110514	142066	173617	19.5
MAKELA4	21	40	0	0.0E+00	1	1	1	1	1	0.0
MAROS-R7	9408	3136	0	1.5E-11	11	11	11	11	11	0.6
MAROS	1443	846	-10	2.3E+01	-1	-1	-1	-1	1000001	1604.8
MODSZK1	1620	687	0	7.1E-06	21	23	26	28	30	0.0
MPSBCD03	5940	5414	-11	5.8E+02	-1	-1	-1	-1	37228	1800.0
NESM	2923	662	0	1.0E-05	1380	3033	4786	6545	8305	13.0
OET1	3	1002	0	3.1E-15	2	2	2	2	2	0.0
OET3	4	1002	0	5.6E-16	2	2	2	2	2	0.0
PEROLD	1376	625	-10	2.7E+01	-1	-1	-1	-1	1000001	1269.4
PILOT4	1000	410	-10	4.9E-01	-1	-1	-1	-1	1000001	1006.6
PILOT87	4883	2030	-11	2.6E+00	-1	-1	-1	-1	30601	1800.0
PILOT-JA	1988	940	-11	4.6E+00	-1	-1	-1	-1	205118	1800.0
PILOTNOV	2172	975	-11	1.2E+00	-1	-1	-1	-1	587830	1800.0
PILOT	3652	1441	-11	1.5E+00	-1	-1	-1	-1	73010	1800.0
PILOT-WE	2789	722	-10	6.3E+00	-1	-1	-1	-1	1000001	1304.7
PT	2	501	0	2.2E-16	1	2	2	2	2	0.0
QAP12	8856	3192	0	8.7E-16	1	1	1	1	1	3.3
QAP15	22275	6330	0	1.3E-15	1	1	1	1	1	20.3
QAP8	1632	912	0	2.8E-15	1	1	1	1	1	0.1
QPBD_OUT	263	211	-10	6.4E-05	26174	55275	105025	849423	1000001	192.1
READING2	6003	4000	0	4.1E-14	3	4	4	4	4	0.0
RECIPELP	180	91	0	1.0E-05	14077	51823	124853	197920	271020	29.2
S277-280	4	4	0	6.7E-16	2	2	2	2	2	0.0
SC105	103	105	0	1.2E-14	3	3	3	3	3	0.0
SC205	203	205	0	2.2E-16	4	8	8	8	8	0.0
SC50A	48	50	0	4.5E-14	3	4	4	4	4	0.0
SC50B	48	50	0	4.5E-12	5	6	6	6	6	0.0
SCAGR25	500	471	0	1.0E-05	6691	10668	14644	18620	22595	4.4
SCAGR7	140	129	0	9.9E-06	445	633	820	1007	1194	0.0
SCFXM1	457	330	0	1.0E-05	15021	30919	46937	62957	78975	19.8
SCFXM2	914	660	-10	9.1E-03	695704	988566	-1	-1	1000001	496.5
SCFXM3	1371	990	-10	9.3E-03	696911	990822	-1	-1	1000001	751.4
SCORPION	358	388	-10	1.6E+25	-1	-1	-1	-1	1000001	286.5
SCRS8	1169	490	-10	2.4E-03	3909	508599	-1	-1	1000001	329.0
SCSD1	760	77	0	2.4E-16	1	2	2	2	2	0.0
SCSD6	1350	147	0	2.9E-15	2	2	2	2	2	0.0
SCSD8	2750	397	0	1.4E-14	2	2	2	2	2	0.0
SCTAP1	480	300	0	1.0E-05	747	4463	11130	17927	24723	4.4
SCTAP2	1880	1090	0	1.0E-05	353	1202	2158	3491	4846	3.3
SCTAP3	2480	1480	0	1.0E-05	484	1574	2899	4272	5665	5.3
SEBA	1028	515	-10	3.4E-03	562938	860780	-1	-1	1000001	447.2
SHARE1B	225	117	-10	5.1E-01	-1	-1	-1	-1	1000001	92.3
SHARE2B	79	96	0	1.0E-05	5477	17872	31068	44263	57458	3.8
SHELL	1775	536	0	9.9E-06	388	484	580	676	772	0.3
SHIP04L	2118	402	0	1.0E-05	5256	28325	51407	74489	97571	40.3
SHIP04S	1458	402	0	1.0E-05	5810	29140	52476	75814	99151	31.1
SHIP08L	4283	778	0	1.0E-05	26	31	36	45	84	0.0
SHIP08S	2387	778	0	9.7E-06	31	35	40	57	81	0.0

Table A.1: Complete results for EAPM (continued)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
SHIP12L	5427	1151	0	1.0E-05	6170	12634	19319	26084	32850	35.8
SHIP12S	2763	1151	0	1.0E-05	6180	25487	88570	151654	214738	146.9
SIERRA	2036	1227	-10	7.9E+20	-1	-1	-1	-1	1000001	1205.0
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	8.9E-15	2	3	3	3	3	0.0
SIPOW4	4	2000	0	2.2E-14	2	3	3	3	3	0.0
SSEBLIN	194	72	0	8.7E-06	68	79	91	102	114	0.0
STAIR	467	356	0	1.0E-05	852	1700	2554	3408	4262	1.5
STANDATA	1075	359	0	1.0E-05	57692	239178	420826	604734	788646	199.3
STANDGUB	1184	361	0	1.0E-05	56680	238698	420852	605292	789734	207.0
STANDMPS	1075	467	0	1.0E-05	36186	85836	171500	257442	343388	109.3
STOCFOR1	111	117	0	8.7E-06	30	42	54	66	82	0.0
STOCFOR2	2031	2157	0	1.0E-05	147	1313	4103	6893	9683	11.8
STOCFOR3	15695	16675	0	1.0E-05	1137	2227	4239	7045	9850	102.1
TESTDECK	14	15	0	8.9E-16	2	2	2	2	2	0.0
TFI2	3	101	0	4.4E-15	2	3	3	3	3	0.0
TRUSS	8806	1000	0	5.7E-13	2	2	2	2	2	0.0
TUFF	587	333	-10	3.2E+00	-1	-1	-1	-1	1000001	1251.4
VTP-BASE	203	198	-10	1.2E+01	-1	-1	-1	-1	1000001	197.2
WOOD1P	2594	244	0	1.0E-05	2	619	31753	68181	104609	616.8
WOODW	8405	1098	0	1.0E-05	45	107	5032	12760	25314	65.0

Table A.2: Complete results for EEPM

					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	33873	52247	70623	88999	107375	47.8
80BAU3B	9799	2262	0	9.9E-06	2805	3380	3524	3668	3812	5.9
ADLITTLE	97	56	0	9.9E-06	842	1098	1356	1612	1870	0.0
AFIRO	32	27	0	9.6E-06	40	62	83	105	127	0.0
AGG2	302	516	-10	2.8E-01	-1	-1	-1	-1	1000001	281.3
AGG3	302	516	-10	3.7E-02	911277	-1	-1	-1	1000001	282.1
AGG	163	488	-10	1.2E+03	-1	-1	-1	-1	1000001	185.0
BANDM	472	305	0	1.0E-05	37114	88888	140970	193052	245134	25.8
BCDOUT	5940	5414	-11	1.6E+01	-1	-1	-1	-1	66646	1800.0
BEACONFD	262	173	0	9.3E-06	100	122	144	166	188	0.0
BLEND	83	74	0	1.0E-05	5481	8153	10825	13499	16171	0.4
BNL1	1175	643	-10	2.1E+00	-1	-1	-1	-1	1000001	241.9
BNL2	3489	2324	-10	4.9E-01	-1	-1	-1	-1	1000001	917.9
BOEING1	384	351	-10	7.2E+00	-1	-1	-1	-1	1000001	144.4
BOEING2	143	166	-10	1.5E-01	97236	-1	-1	-1	1000001	56.1
BORE3D	315	233	-10	2.2E+01	-1	-1	-1	-1	1000001	100.3
BRANDY	249	220	0	1.0E-05	40025	54667	69309	83949	98591	7.5
CAPRI	353	271	-10	1.1E+00	-1	-1	-1	-1	1000001	167.0
CYCLE	2857	1903	-11	1.0E+01	-1	-1	-1	-1	729014	1800.0
CZPROB	3523	929	-10	2.2E+00	-1	-1	-1	-1	1000001	565.9
D2Q06C	5167	2171	-11	2.4E+00	-1	-1	-1	-1	247089	1800.0
D6CUBE	6184	415	0	9.6E-06	42	55	76	98	120	0.2
DEGEN2	534	444	0	9.3E-06	75	97	119	141	164	0.0
DEGEN3	1818	1503	0	9.8E-06	91	113	135	158	183	0.7
DEGENLPA	20	15	-10	2.2E-04	33	53	75	-1	1000001	18.3
DEGENLPB	20	15	-10	2.2E-04	33	53	75	-1	1000001	17.7
DFL001	12230	6071	0	1.0E-05	493	725	957	1189	1421	142.3
E226	282	223	-10	4.1E-04	64160	175798	723642	-1	1000001	102.4
ETAMACRO	688	400	-10	6.5E+00	-1	-1	-1	-1	1000001	347.8
FFFFF800	854	524	-10	9.6E+00	-1	-1	-1	-1	1000001	409.9

Table A.2: Complete results EEPM (continued)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
FINNIS	614	497	0	1.0E-05	254381	412323	558387	668308	696478	99.9
FIT1D	1026	24	-10	5.2E+00	-1	-1	-1	-1	1000001	235.1
FIT1P	1677	627	-10	5.7E-02	1102	-1	-1	-1	1000001	696.6
FIT2D	10500	25	0	1.0E-05	53323	67687	82051	96415	110781	249.2
FIT2P	13525	3000	-11	3.6E-02	1260	-1	-1	-1	362659	1800.0
FORPLAN	421	161	-10	7.4E-01	-1	-1	-1	-1	1000001	115.7
GANGES	1681	1309	0	1.0E-05	2239	2870	4646	6432	8216	4.2
GFRD-PNC	1092	616	-10	7.5E+01	-1	-1	-1	-1	1000001	113.9
GOFFIN	51	50	0	0.0E+00	1	1	1	1	1	0.0
GREENBEA	5405	2392	-11	1.0E-02	516325	772125	-1	-1	777032	1800.0
GREENBEB	5405	2392	-11	1.2E+00	-1	-1	-1	-1	733355	1800.0
GROW15	645	300	0	9.9E-06	29	51	73	95	116	0.0
GROW22	946	440	0	9.4E-06	30	47	69	91	113	0.0
GROW7	301	140	0	9.8E-06	33	55	77	98	120	0.0
ISRAEL	142	174	-10	2.2E+02	-1	-1	-1	-1	1000001	183.2
KB2	41	43	0	1.0E-05	33765	67219	100675	134131	167587	4.2
LINSPANH	97	33	0	9.6E-06	83	98	119	141	163	0.0
LOTFI	308	153	0	1.0E-05	58774	81006	103240	125472	147704	9.0
MAKELA4	21	40	0	0.0E+00	1	1	1	1	1	0.0
MAROS-R7	9408	3136	0	9.7E-06	116	138	159	181	203	6.3
MAROS	1443	846	-10	2.9E+02	-1	-1	-1	-1	1000001	785.8
MODSZK1	1620	687	0	1.0E-05	109	131	153	175	196	0.0
MPSBCD03	5940	5414	-11	3.1E+02	-1	-1	-1	-1	72513	1800.0
NESM	2923	662	0	1.0E-05	137310	176354	227116	291220	356948	321.7
OET1	3	1002	0	9.9E-06	45	67	89	111	132	0.0
OET3	4	1002	0	9.4E-06	44	65	87	109	131	0.0
PEROLD	1376	625	-10	3.3E+02	-1	-1	-1	-1	1000001	663.0
PILOT4	1000	410	-10	2.0E+00	-1	-1	-1	-1	1000001	439.4
PILOT87	4883	2030	-11	1.1E+01	-1	-1	-1	-1	59040	1800.0
PILOT-JA	1988	940	-11	1.2E+01	-1	-1	-1	-1	384324	1800.0
PILOTNOV	2172	975	-10	4.0E+00	-1	-1	-1	-1	1000001	1539.2
PILOT	3652	1441	-11	4.1E+00	-1	-1	-1	-1	145775	1800.0
PILOT-WE	2789	722	-10	2.0E+01	-1	-1	-1	-1	1000001	684.0
PT	2	501	0	9.9E-06	31	53	75	97	118	0.0
QAP12	8856	3192	0	9.8E-06	51	73	95	116	138	22.5
QAP15	22275	6330	0	9.6E-06	56	78	99	121	143	126.5
QAP8	1632	912	0	1.0E-05	44	66	88	110	131	0.8
QPBD_OUT	263	211	-10	6.1E-04	51286	175544	822366	-1	1000001	96.5
READING2	6003	4000	0	1.0E-05	53	141	1363	3543	5723	3.6
RECIPELP	180	91	0	1.0E-05	72459	223959	411043	602947	795035	105.8
S277-280	4	4	0	9.3E-06	34	56	78	100	122	0.0
SC105	103	105	0	9.1E-06	28	50	72	94	116	0.0
SC205	203	205	0	9.5E-06	39	60	82	104	126	0.0
SC50A	48	50	0	9.6E-06	35	57	78	100	122	0.0
SC50B	48	50	0	9.5E-06	32	49	71	93	115	0.0
SCAGR25	500	471	0	1.0E-05	5541	7615	9691	11765	13841	1.3
SCAGR7	140	129	0	9.8E-06	607	799	991	1183	1377	0.0
SCFXM1	457	330	0	1.0E-05	181043	266701	352359	438017	523675	62.7
SCFXM2	914	660	-10	2.0E-03	648749	855421	-1	-1	1000001	256.5
SCFXM3	1371	990	-10	2.9E-03	680703	887719	-1	-1	1000001	397.2
SCORPION	358	388	0	9.8E-06	49	71	93	114	136	0.0
SCRS8	1169	490	-10	1.4E-02	522104	-1	-1	-1	1000001	155.2
SCSD1	760	77	0	9.9E-06	15	37	59	81	102	0.0
SCSD6	1350	147	0	9.9E-06	30	52	74	96	117	0.0
SCSD8	2750	397	0	9.9E-06	56	78	100	122	143	0.0
SCTAP1	480	300	0	1.0E-05	12168	18042	23926	29810	35694	3.2
SCTAP2	1880	1090	0	1.0E-05	12200	17632	23068	28508	33946	12.8
SCTAP3	2480	1480	0	1.0E-05	12126	17528	22938	28348	33758	16.8
SEBA	1028	515	-10	3.4E+00	-1	-1	-1	-1	1000001	166.6
SHARE1B	225	117	-10	4.3E+00	-1	-1	-1	-1	1000001	51.3

Table A.2: Complete results EEPM (continued)

					iterations until error <					
name	n	m	status	error	0.1	0.01	0.001	0.0001	0.00001	CPU time
SHARE2B	79	96	0	1.0E-05	34048	65612	97176	128740	160302	6.3
SHELL	1775	536	-10	2.1E+22	-1	-1	-1	-1	1000001	426.2
SHIP04L	2118	402	0	1.0E-05	14693	30743	46793	62843	78893	15.5
SHIP04S	1458	402	0	1.0E-05	14569	30839	47117	63395	79673	11.2
SHIP08L	4283	778	0	1.0E-05	2443	6233	10029	13825	17621	7.4
SHIP08S	2387	778	0	1.0E-05	2479	6227	10145	14061	17979	4.2
SHIP12L	5427	1151	0	1.0E-05	7819	12583	17351	22117	26883	14.6
SHIP12S	2763	1151	0	1.0E-05	8043	12953	17865	22777	27689	7.8
SIERRA	2036	1227	-10	2.2E+01	-1	-1	-1	-1	1000001	565.0
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	9.3E-06	27	41	63	85	107	0.0
SIPOW4	4	2000	0	9.3E-06	28	42	64	86	108	0.0
SSEBLIN	194	72	0	9.8E-06	355	429	503	577	651	0.0
STAIR	467	356	0	9.9E-06	1601	2263	2923	3583	4245	1.1
STANDATA	1075	359	0	1.0E-05	405430	534472	663704	793010	922314	113.9
STANDGUB	1184	361	0	1.0E-05	405440	534482	663720	793024	922328	116.8
STANDMPS	1075	467	0	1.0E-05	170843	231489	292135	352781	413427	61.0
STOCFOR1	111	117	0	9.5E-06	73	86	108	130	152	0.0
STOCFOR2	2031	2157	0	9.9E-06	287	449	609	771	931	0.6
STOCFOR3	15695	16675	0	1.0E-05	2209	3091	3973	4855	5737	48.9
TESTDECK	14	15	0	9.7E-06	35	57	79	100	122	0.0
TFI2	3	101	0	9.6E-06	31	53	74	96	118	0.0
TRUSS	8806	1000	0	9.4E-06	89	111	133	155	177	0.2
TUFF	587	333	-10	1.2E+01	-1	-1	-1	-1	1000001	250.4
VTP-BASE	203	198	-10	3.9E+01	-1	-1	-1	-1	1000001	76.7
WOOD1P	2594	244	0	1.0E-05	72	14942	40796	66650	92502	265.9
WOODW	8405	1098	0	1.0E-05	4520	11642	20802	30168	39536	56.9