Solving multiscale linear programs using the simplex method in quadruple precision

Ding Ma and Michael A. Saunders

1 Introduction

We consider the solution of large, multiscale linear programs (LPs) of the form

$$\min_x \ c^T x \ \text{ s.t. } \ \ell \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u, \tag{1}$$

where A is a sparse matrix whose entries, like the variables in x, may be of widely varying magnitude. Such problems arise in systems biology in the modeling of biochemical reaction networks, notably Metabolic Expression (ME) models [36, 20]. Reliable solution methods are of such importance to systems biologists that *exact simplex solvers* have been employed [20], even though the typical solution time for an exact solver is measured in weeks for genome-scale models (compared to minutes for a conventional solver using double-precision floating-point arithmetic).

Exact solvers are based on rational arithmetic. There has been considerable work on their implementation and application to important problems [18, 1, 2, 32]. The use of quadruple-precision floating-point has also been mentioned in passing [18, 1].

Let Single, Double, and Quad denote the main floating-point options, with about 7, 16, and 34 digits of precision respectively. Single is not useful in the present context, and Double may not ensure adequate accuracy. This is the reason for our work. On today's machines, Double is implemented in hardware, while Quad (if available) is typically implemented in a software library such as libquadmath [8]. Fortunately, the GCC Fortran compiler now makes Quad available via the real(16) data type. We have therefore been able to make a Quad version of the Fortran 77 linear and nonlinear optimization solver MINOS [26, 27] using the gfortran compiler. Our aim is to explore combined use of the Double and Quad MINOS simplex solvers for

Ding Ma
Dept of MS&E, Stanford University, Stanford, CA, USA e-mail: `dingma@stanford.edu`

Michael Saunders
Dept of MS&E, Stanford University, Stanford, CA, USA, e-mail: `saunders@stanford.edu`

Draft of August 24, 2014.
To appear in M. Al-Baali, L. Grandinetti, and A. Purnama (eds.),
Recent Developments in Numerical Analysis and Optimization,
Springer Proceedings in Mathematics and Statistics.

the solution of large multiscale linear programs. We seek greater efficiency than is normally possible with exact simplex solvers.

Kahan [16] notes that "*carrying somewhat more precision in the arithmetic than twice the precision carried in the data and available for the result will vastly reduce embarrassment due to roundoff-induced anomalies*." He further notes that Quad precision is unlikely to be adopted widely in the foreseeable future because of the cost in CPU time and memory (especially cache) relative to Double, but in terms of finding ways to avoid unexpected total loss of accuracy, "*default evaluation in Quad is the humane option*."

We apply the "humane" approach to difficult LP problems by using the Double simplex solver first, saving the final solution, and warm-starting Quad simplex from that point. For a sequence of related problems, warm-starting each problem in Quad is simplest, but warm-starting in Double and then in Quad may be more efficient.

2 Motivating Applications

The Constraint-Based Reconstruction and Analysis (COBRA) approach [31, 33] has been successfully applied to biological processes such as metabolism and macromolecular synthesis, which when integrated result in inherently multiscale models. In the COBRA approach, a biochemical network is represented by a stoichiometric matrix S with m rows corresponding to metabolites and n columns representing reactions. Mathematically, S is part of the ordinary differential equation that governs the time-evolution of concentrations in the network:

$$\frac{d}{dt}x(t) = Sv(t), \tag{2}$$

where $x(t) \in \mathbf{R}^m$ is a vector of time-dependent concentrations and $v(t) \in \mathbf{R}^n$ is a vector of reaction fluxes. With c^Tv being a biologically motivated objective function (such as maximizing the growth rate at steady state), the constraint-based approach constructs the following LP:

$$\begin{aligned} \max_{v}\ \ & c^T v && \text{(3a)}\\ \text{s.t.}\ \ & Sv = 0, && \text{(3b)}\\ & l \le v \le u, && \text{(3c)} \end{aligned}$$

where growth is defined as the biosynthetic requirements of experimentally determined biomass composition, and biomass generation is a set of reaction fluxes linked in appropriate ratios [31].

The following applications have motivated our work.

Flux Balance Analysis (FBA) FBA is a mathematical and computational approach widely used for studying biochemical reaction networks [31, 30]. The biochemical networks reconstructed in FBA with a linear objective function are essentially LPs as in (3), where the fluxes in vector v may have widely varying values in the range 0–100 say, with small values such as $v_j = 10^{-10}$ being meaningful. With the increasingly large, sparse, and multiscale nature of biochemical networks, a Quad solver has become more necessary, practical, and even efficient.

ME models (FBA with coupling constraints) FBA has been used by Thiele et al. [36] for the first integrated stoichiometric multiscale model of metabolism and macromolecular synthesis for *Escherichia coli* K12 MG1655. The model modifies (3) by adding constraints that couple enzyme synthesis and catalysis reactions to (3b). Coupling constraints of the form

$$c_{\min} \le \frac{v_i}{v_j} \le c_{\max} \tag{4}$$

become linear constraints

$$c_{\min} v_j \le v_i, \quad v_i \le c_{\max} v_j \tag{5}$$

for various pairs of fluxes v_i, v_j. They are linear approximations of nonlinear constraints and make S in (3b) even less well-scaled because of large variations in reaction rates. Quad precision is evidently more appealing in this case.

ME models with nonlinear constraints As coupling constraints are often functions of the organism's growth rate μ, O'Brien et al. [29] consider growth-rate optimization nonlinearly with the single μ as the objective in (3a) instead of via a linear biomass objective function. Nonlinear constraints of the form

$$\frac{v_i}{v_j} \le \mu \tag{6}$$

represented as

$$v_i \le \mu v_j \tag{7}$$

are added to (3b), where v_i, v_j, μ are all variables. Constraints (7) are linear if μ is fixed at a specific value μ_k. O'Brien et al. [29] employ a binary search on a discrete set of values within an interval $[\mu_{\min}, \mu_{\max}]$ to find the largest $\mu_k \equiv \mu^*$ that keeps the associated linear program feasible. Thus, the procedure requires reliable solution of a sequence of related LPs.

Flux Variability Analysis (FVA) After FBA (3) returns an optimal objective value $c^T v^* = Z_0$ (3a), FVA examines how far a particular flux v_j can vary within the feasible region without changing the optimal objective significantly (if $\gamma \approx 1$):

$$\begin{array}{rl} \max\limits_v \text{ or } \min\limits_v & v_j \\ \text{s.t.} & Sv = 0, \\ & c^T v \ge \gamma Z_0, \\ & l \le v \le u, \end{array} \tag{8}$$

where $0 < \gamma < 1$. Potentially $2n$ LPs (8) are solved if all reactions are of interest, with warm starts being used when j increases to $j+1$ [12].

Other challenging LPs A set of difficult LP problems has been collected by Mészáros [22], who names them *problematic* and notes that "*modeling mistakes made these problems "crazy," but they are excellent examples to test numerical robustness of a solver*." Our procedure for handling these *problematic* problems seems appropriate for the systems biology models as well.

3 Algorithm and Implementation

The primal simplex solver in MINOS includes geometric-mean scaling of the constraint matrix, the EXPAND anti-degeneracy procedure [10], and partial pricing (but no steepest-edge pricing, which would generally reduce total iterations and time). Basis LU factorizations and updates are handled by LUSOL [21]. Cold starts use a Crash procedure to find a triangular initial basis. Basis files are used to preserve solutions between runs.

For Double MINOS, floating-point variables are declared double precision (≈ 15 digits). For Quad MINOS, they are real(16) (≈ 34 digits). The LP data A, b, c, S, ℓ, u are stored in Quad even though they are not known to that precision. This allows operations such as Ax and $A^T y$ to be carried out directly on the elements of A and the Quad vectors x, y. If A were stored in Double, such products would require each entry A_{ij} to be converted from Double to Quad at runtime.

To achieve reliability on the Mészáros problems, we developed the following 3-step procedure for solving challenging examples of problems (1)–(8):

Step 1 (Cold start in Double with scaling) Apply Double MINOS with somewhat strict options. Save a final basis file.
Step 2 (Warm start in Quad with scaling) Start Quad MINOS from the saved file with stricter Feasibility and Optimality tolerances. Save a final basis file.
Step 3 (Warm start in Quad without scaling) Start Quad MINOS from the second saved file with no scaling but stricter LU tolerances.

Steps 1 and 2 are "obvious" and should usually be sufficient. In case Step 2 is interrupted, Step 3 provides some insurance and ensures that the Feasibility and Optimality tolerances are imposed upon the original problem (not the scaled problem).

Table 1 MINOS runtime options (defaults and those selected for Steps 1–3).

	Default	Step1	Step2	Step3
	Double	Double	Quad	Quad
Scale option	2	2	2	0
Feasibility tol	1e-6	1e-7	1e-15	1e-15
Optimality tol	1e-6	1e-7	1e-15	1e-15
LU Factor tol	100.0	10.0	10.0	5.0
LU Update tol	10.0	10.0	10.0	5.0
Expand frequency	10000	100000	100000	100000

Table 1 shows the default runtime options for Double MINOS and the options chosen for Steps 1–3 above. The Feasibility tolerance δ_1 is applied in absolute form. Thus, a (possibly scaled) solution v is considered feasible for problem (3) if $\ell - \delta_1 \leq v \leq u + \delta_1$. The Optimality tolerance δ_2 is applied in a relative way. If the current basic solution is of the form $Sv \equiv Bv_B + Nv_N = b$ and if $B^T y = c_B$ for the nonsingular basis matrix B, the current v is considered optimal if $z \equiv c - A^T y$ has the correct sign to within the tolerance $(1 + \|y\|_\infty)\delta_2$.

For conventional Double solvers it is reasonable to set δ_1 and δ_2 in the range 10^{-6} to 10^{-8}. For Quad MINOS we set $\delta_1 = \delta_2 = 10^{-15}$ to be sure of capturing accurately any fluxes v_j as small as $O(10^{-10})$.

4 Numerical Results

We report results from Double and Quad versions of the primal simplex solver in MINOS. All runs were on a 2.93 GHz Apple iMac with quad-core Intel i7, using the gfortran compiler with -O flag. Double MINOS uses 64-bit hardware floating-point throughout. Quad MINOS uses 128-bit software floating-point throughout via gfortran's libquadmath library.

We applied our 3-step procedure to three sets of LP problems. Table 2 lists the problem dimensions and the norms of the optimal primal and dual solution vectors x^*, y^*. Table 3 summarizes the results of Steps 1–3 for each problem.

All problems were input from files in the classical MPS format of commercial mathematical programming systems [24] with 12-character fields for all data values. This was a fortuitous limitation for the ME models, as we mention below. The MPS files for these 14 LP models are downloadable from [25].

Table 2 Three pilot models from Netlib [28], eight Mészáros *problematic* LPs [22], and three ME biochemical network models [19, 36, 35]. Dimensions of $m \times n$ constraint matrices A (= S for the ME models), and size of the largest optimal primal and dual variables x^*, y^*.

model	m	n	nnz(A)	max $\lvert A_{ij}\rvert$	$\|x^*\|_\infty$	$\|y^*\|_\infty$
pilot4	411	1000	5145	2.8e+04	9.6e+04	2.7e+02
pilot	1442	3652	43220	1.5e+02	4.1e+03	2.0e+02
pilot87	2031	4883	73804	1.0e+03	2.4e+04	1.1e+01
de063155	853	1488	5405	8.3e+11	3.1e+13	6.2e+04
de063157	937	1488	5551	2.3e+18	2.3e+17	6.2e+04
de080285	937	1488	5471	9.7e+02	1.1e+02	2.6e+01
gen1	770	2560	64621	1.0e+00	3.0e+00	1.0e+00
gen2	1122	3264	84095	1.0e+00	3.3e+00	1.0e+00
gen4	1538	4297	110174	1.0e+00	3.0e+00	1.0e+00
l30	2702	15380	64790	1.8e+00	1.0e+09	4.2e+00
iprob	3002	3001	12000	9.9e+03	3.1e+02	1.1e+00
TMA_ME	18210	17535	336302	2.1e+04	5.9e+00	1.1e+00
GlcAerWT	68300	76664	926357	8.0e+05	6.3e+07	2.4e+07
GlcAlift	69529	77893	928815	2.6e+05	6.3e+07	2.4e+07

Table 3 Iterations and runtimes in seconds for Step 1 (Double MINOS) and Steps 2–3 (Quad MINOS). Pinf and Dinf = final maximum primal and dual infeasibilities ($\log_{10}$ values tabulated). Problem iprob is infeasible. Bold figures show Pinf and Dinf at the end of Step 3. Pinf $= 10^{-99}$ means Pinf $= 0$. Note that Pinf$/\|x^*\|_\infty$ and Dinf$/\|y^*\|_\infty$ are all $O(10^{-30})$ or smaller, even though only $O(10^{-15})$ was requested. This is an unexpectedly favorable empirical finding.

model	Itns	Times	Final objective	Pinf	Dinf
pilot4	1571	0.1	−2.5811392602e+03	−05	−13
	6	0.0	−2.5811392589e+03	−39	−31
	0	0.0	−2.5811392589e+03	**−99**	**−30**
pilot	16060	5.7	−5.5739887685e+02	−06	−03
	29	0.7	−5.5748972928e+02	−99	−27
	0	0.2	−5.5748972928e+02	**−99**	**−32**
pilot87	19340	15.1	3.0171038489e+02	−09	−06
	32	2.2	3.0171034733e+02	−99	−33
	0	1.2	3.0171034733e+02	**−99**	**−33**
de063155	921	0.0	1.8968704286e+10	−13	+03
	78	0.1	9.8830944565e+09	−99	−17
	0	0.0	9.8830944565e+09	**−99**	**−24**
de063157	488	0.0	1.4561118445e+11	+20	+18
	476	0.5	2.1528501109e+07	−27	−12
	0	0.0	2.1528501109e+07	**−99**	**−12**
de080285	418	0.0	1.4495817688e+01	−09	−02
	132	0.1	1.3924732864e+01	−35	−32
	0	0.0	1.3924732864e+01	**−99**	**−32**
gen1	369502	205.3	−1.6903658594e−08	−06	−12
	246428	9331.3	1.2935699163e−06	−12	−31
	2394	81.6	1.2953925804e−06	**−45**	**−30**
gen2	44073	60.0	3.2927907828e+00	−04	−11
	1599	359.9	3.2927907840e+00	−99	−29
	0	10.4	3.2927907840e+00	**−99**	**−32**
gen4	45369	212.4	1.5793970394e−07	−06	−10
	53849	14812.5	2.8932268196e−06	−12	−30
	37	10.4	2.8933064888e−06	**−54**	**−30**
l30	1229326	876.7	9.5266141574e−01	−10	−09
	275287	7507.1	−7.5190273434e−26	−25	−32
	0	0.2	−4.2586876849e−24	**−24**	**−33**
iprob	1087	0.2	2.6891551285e+03	+02	−11
	0	0.0	2.6891551285e+03	+02	−31
	0	0.0	2.6891551285e+03	+02	**−28**
TMA_ME	12225	37.1	8.0051076669e−07	−06	−05
	685	61.5	8.7036315385e−07	−24	−30
	0	6.7	8.7036315385e−07	**−99**	**−31**
GlcAerWT	62856	9707.3	−2.4489880182e+04	+04	−05
	5580	3995.6	−7.0382449681e+05	−07	−26
	4	60.1	−7.0382449681e+05	**−19**	**−21**
GlcAlift	134693	14552.8	−5.1613878666e+05	−03	−01
	3258	1067.1	−7.0434008750e+05	−09	−26
	2	48.1	−7.0434008750e+05	**−20**	**−22**

The pilot problems

These are economic models developed by Prof George Dantzig's group in the Systems Optimization Laboratory at Stanford University during the 1980s. They are available from Netlib [28]. For the middle example (pilot), MINOS required about 24 hours on a DEC MicroVAX II during 1987, and did not perform reliably until the EXPAND anti-degeneracy procedure was developed.

Line 1 for pilot in Table 3 shows that Double MINOS with cold start and scaling required 16060 primal simplex iterations and 5.7 CPU seconds. The final unscaled primal solution x satisfied the bounds ℓ and u in (1) to within $O(10^{-6})$, and the dual solution y satisfied the optimality conditions to within $O(10^{-3})$.

Line 2 for pilot shows that Quad MINOS starting from that point with scaling needed only 29 iterations and 0.7 seconds to obtain a very accurate solution (where $\text{Pinf} = 10^{-99}$ means that the maximum primal infeasibility was 0.0).

Line 3 for pilot shows that in the "insurance" step, Quad MINOS warm-starting again but with no scaling gave a full quad-precision solution at almost no cost: maximum infeasibilities 0.0 and $O(10^{-32})$. The final Double and Quad objective values differ in the 4th significant digit, as suggested by removal of Step 1's $O(10^{-3})$ dual infeasibility.

Results for the bigger problem pilot87 are analogous.

The Mészáros problems

The *problematic* LPs were provided as MPS files by Ed Klotz [17]. The first two problems have unusually large entries in the constraint matrix A. The Step 1 Double MINOS solution has at best 1 digit of precision in the objective value for de063155, and is quite erroneous for de063157. Nevertheless, the Step 2 and 3 Quad solutions are seen to be highly accurate when the solution norms are taken into account.

The gen* problems come from image reconstruction, with no large entries in A, x, y, but highly degenerate primal solutions x. (In both Steps 1 and 2 for gen1, 60% of the iterations made no improvement to the objective, and the final solution has 30% of the basic variables on their lower bound.) For gen1, warm-starting Quad MINOS from the Step 1 basis gave an almost feasible initial solution (266 basic variables outside their bounds by more than 10^{-15} with a sum of infeasibilities of only $O(10^{-8})$), yet nearly 250,000 iterations were needed in Step 2 to reach optimality. These examples show that Quad precision does not remove the need for a more rigorous anti-degeneracy procedure (such as Wolfe's method as advocated by Fletcher [6]), and/or steepest-edge pricing [7], to reduce significantly the total number of iterations.

Problem l30 behaved similarly (80% degenerate iterations in Steps 1–2). The tiny objective value is essentially zero, so we can't expect the Step 2 and 3 objectives to agree in their leading digits.

Problem iprob is an artificial one that was intended to be feasible with a very ill-conditioned optimal basis, but the MPS file provided to us contained low-precision data (many entries like 0.604 or 0.0422). Our Double and Quad runs agree that the problem is infeasible. This is an example of Quad removing some doubt that was inevitable with just Double.

The systems biology ME problems

Like the gen* problems, the ME models showed 40–60% degenerate iterations in Step 1, but fortunately not so many total iterations in Step 2. This is important for FVA and for ME with nonlinear constraints, where there are many warm starts.

Problem TMA_ME developed by Lerman [19] has some large matrix entries $|S_{ij}|$ and many small solution values v_j that are meaningful to systems biologists. The ME part of the model also contains small matrix entries. In Step 1, almost all iterations went on finding a feasible solution, and the objective then had the correct order of magnitude.

This was the first ME model that we used for Quad experiments (in April 2012). The data S, c, ℓ, u in (3) came as a Matlab structure with $c_j = 0$, $l_j = 0$, $u_j = 1000$ for most j, except $c_{17533} = 1$ (meaning maximize flux v_{17533}), four variables had smaller positive upper bounds, the last variable had moderate positive bounds, and 64 variables were fixed at zero. We output the data to a plain text file. Most entries of S are integers (represented exactly), but about 5000 S_{ij} values are of the form 8.037943687315e-01 or 3.488862338191e-06 with 13 significant digits. The text data was read into Double and Quad versions of a prototype Fortran 90 implementation of SQOPT [11].

For the present paper, we used the same Matlab data to generate an MPS file for input into MINOS. Since this is limited to 6 significant digits, the values in the preceding paragraph were rounded to 8.03794e-01 and 3.48886e-06 and in total about 5000 S_{ij} values had $O(10^{-6})$ relative perturbations of this kind. We have been concerned that such data perturbations could alter the FBA solution greatly because the final basis matrices could have condition number as large as 10^6 or even 10^{12} (as estimated by LUSOL). In comparing Quad SQOPT on Matlab data with Quad MINOS on MPS data, we fortunately observe that the final objective values for TMA_ME agree to 5 digits and match the results from SoPlex [34] and the exact simplex solver QSopt_ex [32], as reported by Lerman [19]:

	Optimal objective	
SoPlex 80bit	8.703671403e-07	Matlab data
QSopt_ex	8.703646169e-07	Matlab data
Quad SQOPT	8.703646169e-07	Matlab data
Quad MINOS	8.703631539e-07	MPS data

More importantly, for the most part *even small solution values* are perturbed in only the 5th or 6th significant digit. Let v and w be the solutions obtained by the two Quad solvers on slightly different data. Some example solution values follow:

j	107	201	302	
Quad SQOPT v_j	2.336815e-06	8.703646e-07	1.454536e-11	Matlab data
Quad MINOS w_j	2.336823e-06	8.703632e-07	1.454540e-11	MPS data

Among all j for which $\max(v_j, w_j) > 10^{-15}$ (the feasibility tolerance), the largest relative difference $|v_j - w_j|/\max(v_j, w_j)$ was less than 10^{-5} for all but 31 variables.

For 22 of these pairs, either v_j or w_j was primal or dual degenerate (meaning one of them was zero and there are alternative solutions with the same objective value). The remaining 9 variables had these values:

j	v_j	w_j	Relative difference
16383	6.0731e-07	2.0374e-06	0.70
16459	1.7090e-06	2.1778e-06	0.22
16483	2.4675e-06	5.9936e-07	0.76
16730	1.4432e-06	7.8685e-07	0.46
17461	1.7090e-06	2.1778e-06	0.22
17462	2.4675e-06	5.9936e-07	0.76
17478	6.0731e-07	2.0374e-06	0.70
17507	1.4432e-06	7.8685e-07	0.46
17517	8.7036e-07	2.9740e-06	0.71

We see that the v_j, w_j values are quite small (the same magnitude as the data perturbation), and for each of the 9 pairs there is about 1 digit of agreement. In general we could expect thousands of small solution pairs to differ this much, yet for almost *all* 17535 pairs, there are at least 5 digits of agreement.

These observations about two forms of problem TMA_ME are welcome empirical evidence of the robustness of this particular multiscale model. Quad solvers can help evaluate the robustness of future (increasingly large) models of metabolic networks by enabling similar comparison of high-accuracy solutions for slightly different problems.

Problem GlcAerWT is an ME model from the detailed study by Thiele et al. [36]. Difficulties with solving TMA_ME and GlcAerWT led to the lifting technique of Sun et al. [35] (and to problem GlcAlift).

After 33000 iterations on GlcAerWT, Double MINOS began to report singularities every 50–100 iterations following updates to the basis LU factors. After another 30000 iterations, MINOS terminated Step 1 with maximum infeasibility $O(10^4)$. Step 2 required significant work to achieve a reasonably accurate solution. Step 3 quickly confirmed the final objective value with high accuracy considering the $O(10^7)$ primal and dual solutions norms.

Problem GlcAlift is a reformulated version of GlcAerWT in which some large matrix entries $c_{\max}$ in (5) have been reduced via the lifting technique [35]. In Step 1, Double MINOS again reported frequent singularities and required twice as many iterations as GlcAerWT, but a near-optimal solution was found (allowing for the primal and dual solution norms of $O(10^7)$), and Steps 2–3 were more efficient and accurate. The objective function for both GlcA models is to maximize variable v_{60069}. The fact that the Step 1 objective values have no correct digits illustrates the challenge these models present and emphasizes the benefits that Quad precision offers. Theoretically the optimal objectives for GlcAerWt and GlcAlift should agree. We assume that the limited data precision in the MPS files is responsible for only 3-digit agreement. Fortunately the Tomlab interface used by Thiele et al. [36] allows full double-precision data [37]. We can do the same for MINOS, as we did for SQOPT.

5 Discussion

While today's advanced LP solvers such as CPLEX, Gurobi, Mosek, and Xpress [3, 23, 13, 5] are effective on a wide range of large and challenging linear optimization models, the study by Thiele et al. [36] emphasizes the need for improved reliability in solving FBA and ME models in systems biology. Fortunately, reformulation [35] and careful use of the commercial solvers CPLEX and Gurobi permitted successful analysis in Thiele et al. [36] of the GlcAerWT and GlcAlift models discussed here, and we encourage this approach for Step 1 of our proposed 3-step procedure (section 3). The bulk of the work in solving multiscale LP problems can still be performed there by conventional Double solvers (possibly including Barrier solvers with simplex "cross-over" to provide a basis).

Our aim has been to demonstrate that the Step 2 and 3 warm starts with Quad solvers will be acceptably efficient, and that the accuracy achieved exceeds requirements by a very safe margin. The "humane" approach of Kahan [16]—use of Quad LP solvers—is certainly more efficient than applying exact simplex solvers, even though the latter have proved their value in several applications [18, 1, 2, 20].

An intriguing question remains concerning the bold figures in Table 3. The primal and dual solutions obtained with Quad precision are *substantially more accurate than the* 10^{-15} *requested*. The same has been true for all of the classic set of Netlib problems [28] that we have run. Kahan [16] explains that "perturbations get amplified by singularities near the data." He describes a "pejorative surface" of data points where singularity exists, and expects loss of accuracy as data approaches the surface. The volume surrounding the pejorative surface is the danger zone, but: "*Arithmetic precision is usually extravagant enough if it is somewhat more than twice as [great] as the data's and the desired result's. Often that shrunken volume contains no data*." We can surmise that Kahan has anticipated our observed situation, wherein LP problems defined with double-precision data appear unlikely to be too ill-conditioned for a Quad solver.

We believe that quadruple-precision solutions are now practical for multiscale LP applications such as FBA and FVA models in systems biology [31, 30, 36, 12], and that they justify increased confidence as systems biologists build ever-larger models to explore new hypotheses about metabolism and macromolecular synthesis. Our 3-step procedure of section 3 allows combined use of Double and Quad solvers and should lead to solutions of exceptional accuracy in other areas of computational science involving multiscale optimization problems. For example, Dattorro [4] has derived an approach to analog filter design that requires a Quad LP or nonlinear solver to deal with a wide range of frequencies (which must be raised to the fourth power). We look forward to implementing this approach, as well as treating the nonlinear constraints (7) directly to take advantage of the nonlinear algorithms in Quad MINOS.

Funding

This work was supported by the National Institute of General Medical Sciences of the National Institutes of Health [award U01GM102098]. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

Acknowledgements We thank Ronan Fleming, Philip Gill, Ed Klotz, Joshua Lerman, Edward O'Brien, Yuekai Sun, Ines Thiele, and Elizabeth Wong for much help during the course of this work and for valuable comments on the manuscript. Joshua Lerman at UC San Diego provided the model named TMA_ME here (originally model_final_build_unscaled.mat) and advised us of the final objective values obtained by SoPlex and QSopt_exact. Ines Thiele at the University of Luxembourg provided the GlcAerWT model and insight into the ME coupling constraints. Yuekai Sun at Stanford University created the reformulated version named GlcAlift here. Ed Klotz of IBM in Carson City NV provided MPS files for the Mészáros *problematic* LPs and lengthy discussions of their properties. A referee also provided valuable feedback.

References

1. Applegate, D. L., Cook, W., Dash, S., and Espinoza, D. G. (2007). Exact solutions to linear programming problems. *Operations Res. Lett.*, **35**, 693–699.
2. Cook, W., Dash, S., and Espinoza, D. G. (2007). Exact solutions to linear programming problems. *Operations Research Letters*.
3. CPLEX (2014). IBM ILOG CPLEX optimizer. `http://www.ibm.com/software/commerce/optimization/cplex-optimizer/`.
4. Dattorro, J. (2014). Private communication, Stanford University, Stanford, CA, USA.
5. FICO Xpress Optimization Suite. `http://www.fico.com/en/products/fico-xpress-optimization-suite/`.
6. Fletcher, R. (2014). On Wolfe's method for resolving degeneracy in linearly constrained optimization. *SIAM J. Optim.* **24**(3), 1122–1137.
7. Forrest, J. J. and Goldfarb, D. (1992). Steepest-edge simplex algorithms for linear programming. *Math. Program.*, **57**, 341–374.
8. GCC libquadmath (2014). The GCC Quad-precision math library application programming interface (API). `http://gcc.gnu.org/onlinedocs/libquadmath/`.
9. Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1987). Maintaining LU factors of a general sparse matrix. *Linear Algebra and its Applications*, **88/89**, 239–270.
10. Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1989). A practical anti-cycling procedure for linear and nonlinear programming. *Math. Program.*, **45**, 437–474.
11. Gill, P. E., Murray, W., and Saunders, M. A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, **47**(1), 99–131. SIGEST article.
12. Gudmundsson, S. and Thiele, I. (2010). Computationally efficient flux variability analysis. *BMC Bioinformatics*, **11**(489), 3 pp.
13. Gurobi (2014). Gurobi optimization system for linear and integer programming. http://www.gurobi.com.
14. Hall, J. A. J. and McKinnon, K. I. M. (2004). The simplest examples where the simplex method cycles and conditions where EXPAND fails to prevent cycling. *Math. Progam., Ser. B*, **100**, 133–150.
15. IEEE standard for floating-point arithmetic (2008). IEEE Std 754-2008. IEEE Computer Society.

16. Kahan, W. (2011). Desperately needed remedies for the undebuggability of large floating-point computations in science and engineering. IFIP/SIAM/NIST Working Conference on Uncertainty Quantification in Scientific Computing, Boulder CO, `http://www.eecs.berkeley.edu/~wkahan/Boulder.pdf`.
17. Klotz, E. (2014). Private communication, IBM, Carson City, NV, USA.
18. Koch, T. (2004). The final NETLIB-LP results. *Operations Research Letters*, **32**, 138–142.
19. Lerman, J. A. (2012). Private communication, Department of Bioengineering, University of California San Diego, CA, USA.
20. Lerman, J. A., Hyduke, D. R., Latif, H., Portnoy, V. A., Lewis, N. E., Orth, J. D., Schrimpe-Rutledge, A. C., Smith, R. D., Adkins, J. N., Zengler, K., and Palsson, B. Ø. (2012). In silico method for modelling metabolism and gene product expression at genome scale. *Nature Communications*, **3**(929), 10 pp.
21. LUSOL (2013). Sparse LU factorization package. `http://web.stanford.edu/group/SOL/software/lusol`.
22. Mészáros, C. (2004). A collection of challenging LP problems. `http://www.sztaki.hu/~meszaros/public_ftp/lptestset/problematic`.
23. MOSEK (2014). MOSEK Optimization Software. `http://www.mosek.com/`.
24. MPS (1960). Input format for LP data. `http://lpsolve.sourceforge.net/5.5/mps-format.htm`.
25. MPS files (2014). Data files in MPS format for models in Table 2. `http://web.stanford.edu/group/SOL/multiscale/models.html`.
26. Murtagh, B. A. and Saunders, M. A. (1978). Large-scale linearly constrained optimization. *Math. Program.*, **14**, 41–72.
27. Murtagh, B. A. and Saunders, M. A. (1982). A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. *Math. Program. Study*, **16**, 84–117.
28. Netlib (1988). Netlib collection of LP problems in MPS format. `http://www.netlib.org/lp/data`.
29. O'Brien, E. J., Lerman, J. A., Chang, R. L., Hyduke, D. R., and Palsson, B. Ø. (2013). Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Molecular Systems Biology*, **9**(693), 13 pp.
30. Orth, J. D., Thiele, I., and Palsson, B. Ø. (2010). What is flux balance analysis? *Nature Biotechnology*, **28**(3), 245–248.
31. Palsson, B. Ø. (2006). *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, NY.
32. QSopt_ex (2008). Qsopt_ex: A simplex solver for computing exact rational solutions to LP problems. `http://www.math.uwaterloo.ca/~bico/qsopt/index.html`.
33. Schellenberger, J., Que, R., Fleming, R. M. T., Thiele, I., Orth, J. D., Feist, A. M., Zielinski, D. C., Bordbar, A., Lewis, N. E., Rahmanian, S., *et al.* (2011). Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature Protocols*, **6**(9), 1290–1307.
34. SoPlex (1996). Soplex: The sequential object-oriented simplex solver. `http://soplex.zib.de`.
35. Sun, Y., Fleming, R. M. T., Thiele, I., and Saunders, M. A. (2013). Robust flux balance analysis of multiscale biochemical reaction networks. *BMC Bioinformatics*, **14**(240).
36. Thiele, I., Fleming, R. M. T., Que, R., Bordbar, A., Diep, D., and Palsson, B. Ø. (2012). Multiscale modeling of metabolism and macromolecular synthesis in *E. coli* and its application to the evolution of codon usage. *PLOS ONE*, **7**(9), 18 pp.
37. TOMLAB (2014). Optimization environment for MATLAB. `http://tomopt.com`.
38. Vanderbei, R. (2007). *Linear Programming: Foundations and Extensions*. Springer, 3 edition.