Quadruple-precision solution of genome-scale models of Metabolism and macromolecular Expression

Ding Ma∗, Laurence Yang†, Ronan M. T. Fleming‡, Ines Thiele‡, Bernhard O. Palsson§, & Michael A. Saunders∗¶

qLP20 June 2, 2016

Constraint-Based Reconstruction and Analysis (COBRA) is currently the only methodology that permits integrated modeling of Metabolism and macromolecular Expression (ME) at genome-scale. Linear optimization can compute steady-state flux solutions to ME models, but flux values are spread over many orders of magnitude. Standard double-precision solvers may return inaccurate solutions or report that no solution exists. ME models currently have 70,000 constraints and variables and will grow larger, so that exact simplex solvers are not practical. We have developed a quadruple-precision version of our linear and nonlinear optimizer MINOS, and a solution procedure (DQQ) involving Double and Quad MINOS that balances efficiency and reliability for ME models. Efficient double-precision optimizers already enabled exponential growth in biological applications of metabolic models. Combined use of Double and Quad solvers now promises extensive use of linear, nonlinear, genome-scale, and multiscale ME models.

∗Dept of Management Science and Engineering, Stanford University, Stanford, CA, USA.
†Dept of Bioengineering, University of California, San Diego, CA, USA.
‡Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Luxembourg.
§Novo Nordisk Foundation Center for Biosustainability, The Technical University of Denmark, 2970 Hørsholm, Denmark.
¶Correspondence should be addressed to M.A.S. (email: saunders@stanford.edu).
Constraint-Based Reconstruction and Analysis (COBRA) [40] has been applied successfully to predict phenotypes for a range of genome-scale biochemical processes. The popularity of COBRA is partly due to the efficiency of the underlying optimization algorithms, permitting genome-scale modeling at a particular timescale using readily available open source software [43, 53] and industrial quality optimization algorithms [20, 22, 31]. A widespread application of COBRA is the modeling of steady states in genome-scale Metabolic models (M models). COBRA has also been used to model steady states in macromolecular Expression networks (E models), which stoichiometrically represent the transcription, translation, post-transcriptional modification and formation of all protein complexes required for macromolecular biosynthesis and metabolic reaction catalysis [49, 46]. COBRA of metabolic networks or expression networks depends on numerical optimization algorithms to compute solutions to certain model equations, or to determine that no solution exists. Our purpose is to discuss available options and to demonstrate an approach that is reliable and practical for increasingly large networks.

We previously demonstrated that COBRA can be used to stoichiometrically couple metabolic and macromolecular expression networks with single nucleotide resolution at genome-scale [48, 28]. The corresponding Metabolic and macromolecular Expression models (ME models) explicitly represent catalysis by macromolecules, and in turn, metabolites are substrates in macromolecular synthesis reactions (Figure 1). These reconstructions lead to the first multi-timescale and genome-scale stoichiometric models, as they account for multiple cellular functions operating on widely different timescales and typically account for about 40 percent of a prokaryote’s open reading frames. A typical M model might be represented by 1000 reactions generated by hand [8]. In contrast, ME models can have more than 50,000 reactions, most of which have been generated algorithmically from template reactions (defined in the literature) and omics data [48, 28].

ME models have opened a whole new vista for predictive mechanistic modeling of cellular processes, but their size and multiscale nature pose a challenge to standard optimization solvers. Typical net metabolic reaction rates are 6 orders of magnitude faster than macromolecular synthesis reaction rates (millimole/gDW vs nanomole/gDW, gDW = gram dry weight), and the number of metabolic moieties in a macromolecule can be many orders of magnitude larger than in a typical metabolite. The combined effect is that the corresponding ME models have biochemically significant digits over many orders of magnitude. For example, Flux Balance Analysis (FBA) [39] is a COBRA method that predicts steady-state reaction rates (fluxes) in biological networks via linear optimization (commonly called linear programming or LP, but nowadays denoted LO). When FBA is augmented with coupling constraints [47] that constrain the ratio between catalytic usage of a molecule and synthesis of the same molecule, the corresponding LO problem is multiscale in the sense that both data values and solution values have greatly varying magnitudes. For a typical ME model, input data values (objective, stoichiometric or coupling coefficients, or bounds) differ by 6 orders of magnitude, and biochemically meaningful solution values can be as large as $10^8$ or as small as $10^{-10}$.

Linear optimization solvers usually apply scaling techniques [13, 51] to problems that are not already well scaled. The scaled problem typically solves more efficiently and accurately, but the solver must then unscale the solution, and this may generate significant primal or dual infeasibilities in the original problem (the constraints or optimality conditions may not be accurately satisfied).

We previously implemented a lifting approach [45] to alleviate this difficulty with multiscale FBA problems. Lifting involves the introduction of auxiliary variables in a manner that seeks to turn a poorly
scaled problem into a less poorly scaled problem in a higher dimension. For example, one can decompose

\[ A + 10,000B \rightarrow C + D \]  

(1)

into two reactions using a dummy molecule \( \hat{B} \) to yield

\[ 100B \rightarrow \hat{B} \]  

(2)

\[ A + 100\hat{B} \rightarrow C + D. \]  

(3)

A large matrix entry is reduced, as well as the ratio in magnitude of the largest and smallest meaningful solution values. This approach has permitted standard solvers (with scaling) to find more accurate solutions to FBA problems because the unscaling process is less damaging.

Another approach to increasing the precision of FBA is to use an exact solver. We previously used the exact simplex solver QSopt\_ex \(^{11,41}\) for FBA of a ME model of *Thermotoga maritima* \(^{28}\) (model TMA\_ME) representing a reaction matrix with about 18,000 metabolites and reactions. The solution time with the exact solver was about two weeks, compared to a few minutes for a standard double-precision solver.

QSopt\_ex has since been applied to a collection of 98 metabolic models by Chindelvitch et al. \(^{2}\), who question the validity of solutions from standard solvers. Most of the 98 models have less than 1000 metabolites and reactions. QSopt\_ex required about a day to solve all models \(^{2}\), compared to a few seconds in total for a double-precision solver.

Note that exact solvers compute exact solutions to LO problems involving rational data. Although stoichiometric coefficients for chemical reactions are in principal integers, most genome-scale metabolic models have non-integer coefficients where the stoichiometry is known to only a few digits, e.g., a coefficient in a biomass reaction. Such a stoichiometric coefficient should not be considered exact data (to be converted into a rational number for use with an exact solver) because an exact solver may erroneously conclude that the model admits no bounded steady-state flux. Some sort of feasibility tolerance is necessary.

To advance COBRA for increasingly large biochemical networks, there is a need for solvers that can obtain solutions more efficiently than currently possible with exact simplex solvers. Model solutions must be sufficiently accurate to ensure biological fidelity, and (ideally) solvers should accommodate smooth nonlinearities in COBRA problems, as already demonstrated in Yang et al. \(^{50}\).

Let Single, Double, and Quad denote the main floating-point options, with about 7, 16, and 34 digits of precision respectively. For many years, scientific computation has advanced in two complementary ways: improved algorithms and improved hardware. Typically the desire is for greater speed, but to improve the reliability of algorithms for multiscale modeling problems we must look back as well as forward (Table 1). Kahan \(^{24}\) notes that early C compilers generated Double instructions for all floating-point computation even for program variables stored in single precision (item 1b in Table 1). Thus for a brief period, C programs were serendipitously more reliable than typical Fortran programs of the time, which followed item 1a. (For Single variables \( a \) and \( b \), Fortran compilers would use Single arithmetic to evaluate the basic expressions \( a \pm b, a\times b, a/b \), whereas C compilers would transfer \( a \) and \( b \) to longer registers and operate on them using Double arithmetic.) Most often, the C compiler’s extra precision was not needed, but occasionally it did make a critical difference. Kahan calls this the humane approach to debugging complex numerical software.

Since then, compilers have typically followed the 1a approach, evaluating expressions using the same arithmetic as the variables’ data type (items 2a and 2b). Most scientific codes follow 2b: Double variables and Double arithmetic throughout (16 significant digits stored in 64-bit words). The floating-point hardware often has slightly extended precision (80-bit registers), but there is no Double/Quad combination (no item 2c) to give double-precision codes a serendipitous gain in reliability. Nowadays, Quad hardware is available to some extent (e.g., via microcoding on IBM mainframes), but for the foreseeable future it will be simulated on most machines by much slower software \(^{24}\). As a result, items 3a and 3b remain dominant and most practitioners would not think of adopting option 3c. Nevertheless, for multiscale COBRA problems, history is pointing the way. It has long been unthinkable to implement a large-scale optimizer using Single arithmetic. The benefits of using Double everywhere are uncountable. We believe the time has come to produce Quad versions of key sparse-matrix packages and large-scale optimization solvers (item 3c).

Here, we report the development and biological application of Quad MINOS, a quadruple-precision version of our general-purpose, industrial-strength linear and nonlinear optimization solver MINOS \(^{34,35}\). We also developed a Double-Quad-Quad MINOS (DQQ-MINOS) procedure that combines the use of Double and Quad simplex solvers in order to achieve

| Decade | Variables | Computation |
|--------|-----------|-------------|
| 1a     | 1970 Fortran | Single    | Single |
| 1b     | 1970 C     | Single     | Double  |
| 2a     | 1980       | Single     | Double  |
| 2b     | 1980       | Double     | Double  |
| 3a     | 2010       | Single     | Double  |
| 3b     | 2010       | Double     | Double  |
| 3c     | 2010       | Quad       | Quad    |
a balance between efficiency in computation and accuracy of the solution. We extensively tested this DQQ-MINOS procedure for FBA of 83 genome-scale metabolic network models (M models) obtained from the UCSD Systems Biology repository [32, 33] and 78 from the BiGG database [25]. We also tested the DQQ-MINOS procedure for FBA of ME models of Thermotoga maritima [28] (about 18,000 metabolites and reactions) and E. coli K12 MG1655 [35] (about 70,000 metabolites and reactions). For M models, we find that Double MINOS alone is sufficient to obtain non-zero steady-state solutions that satisfy feasibility and optimality conditions with a tolerance of $10^{-7}$. For ME models, application of our DQQ procedure resulted in non-zero steady-state solutions that satisfy feasibility and optimality conditions with a tolerance of $10^{-20}$. The largest model, a lifted version of the E. coli ME model, required 4.5 hours on an Apple iMac (Intel i7 ≈ 3GHz). Solution with an exact solver would take many weeks.

Gleixner et al. [18] describe a further alternative to an exact solver. This “zoom” or “iterative refinement” procedure depends on a floating-point solver. For difficult cases this could be Quad MINOS, as discussed later.

**Results**

**Quad MINOS: A general-purpose Quad optimization solver.** On today’s machines, Double is implemented in hardware, while Quad (if available) is typically implemented in a software library, such as libquadmath [14, 15]. Since release 4.6 of the GCC C and Fortran compilers (2011 and later), Quad has been available via the long double and real(16) data types. Thus, we developed a Quad version of our Fortran 77 linear and nonlinear optimization solver MINOS [34, 35] using the gfortran compiler (GNU Fortran 5.2.0). Nonlinear ME models have been treated by Quad MINOS in Yang et al. [50]. Here, we apply only the LO part of MINOS, which uses a stable implementation of the primal simplex method. We discuss the simplex method in terms of the FBA problem \( \text{(8)} \) with empty \( Cv \leq d \).

For Double MINOS, floating-point variables are declared real(8) (≈ 16 digits). For Quad MINOS, they are real(16) (≈ 34 digits). The data are stored in Quad even though they are not known to that precision. This allows operations such as \( Sv \) and \( S^Tv \) to be carried out directly on the elements of \( S \) and the Quad vectors \( v, y \). If \( S \) were stored in Double, such products would require each entry \( S_{ij} \) to be converted from Double to Quad at runtime many times.

The simplex solver in MINOS includes geometric mean scaling of the data [13], the EXPAND anti-degeneracy procedure [17], 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 [29]. Cold starts use a Crash procedure to find a triangular initial basis matrix. Basis files are used to preserve solutions between runs and to enable warm starts.

Metabolic models with Quad solvers admit biomass synthesis. COBRA models of metabolic networks assume the existence of at least one steady-state flux vector that satisfies the imposed constraints and admits a non-zero optimal objective. Where the objective is to maximize a biomass synthesis reaction, the corresponding FBA problem should admit a nonzero biomass synthesis rate. It is established practice to solve monoscale metabolic FBA problems with Double solvers, so one may ask: do biomass synthesis predictions from metabolic models hold when higher precision solvers are applied to the same FBA problem? We tested 78 M models derived from the BiGG database [25] using Double and Quad solvers. We downloaded these models in the JSON format and parsed them using the JSON reader in cobrapy [7]. The models were not modified after loading, so all constraints, bounds, and objective coefficients were used as in the original files. All models were feasible using both Double and Quad, and all but five models had an optimal objective value greater than zero. Of these five models, four simply had all-zero objective coefficients, while the remaining (RECON1) model maximized a single reaction (S6T14g) but its optimal value was zero. The maximum difference in objective value between Double and Quad was \( 2.6 \times 10^{-12} \). The additional precision provided by Quad MINOS enabled us to conclude efficiently and effectively that the 78 metabolic models could be solved reliably using Double solvers. This conclusion is consistent with previous findings by Ebrahim et al. [6].

**Efficient combination of Double and Quad.** To achieve reliability and efficiency, we developed the following 3-step procedure (Figure 2).
Algorithm 1: DQQ procedure

**Data:** Linear program \[ S = (B, N) \]

**Result:** Flux vector \( \nu^* \), Basis partition \( \mathbf{S} = (B, N) \), one of three states:
- optimal, infeasible, or unbounded
  (possible if infinite bounds exist)

**Step D:** use Double MINOS with scaling;
1. Scale columns and rows of \( S \);
2. repeat
   - Find a nonsingular basis matrix \( B \) from the columns of \( S = (B, N) \);
   - Find \( v = (v_B, v_N) \) satisfying \( Su = Bu_B + Nu_N = 0 \);
   - Partition \( c \) accordingly as \( (c_B, c_N) \);
   - Solve \( B^T y = c_B \);
   - \( z_N \leftarrow c_N - N^T y \);
   - \( \Delta = (1 + \|y\|_\infty)\delta_2 \);
3. Save \( B \):
   - **Step Q1:** use Quad MINOS with scaling;
     - Start with the saved \( B \) from Step D to run lines 1–3 to find a new \( B \);
   - **Step Q2:** use Quad MINOS without scaling;
     - Start with the saved \( B \) from Step Q1 to run lines 2–3 to reach a final \( B \);

Table 2: MINOS runtime options (defaults and those selected for each step of the DQQ procedure).

| Data          | Default | Step D | Step Q1 | Step Q2 |
|---------------|---------|--------|---------|---------|
| Precision     |         |        |         |         |
| Scale option  | 2       | 2      | 2       | 0       |
| Feasibility tol \( \delta_1 \) | 1e-6 | 1e-7 | 1e-15 | 1e-15 |
| Optimality tol \( \delta_2 \) | 1e-6 | 1e-7 | 1e-15 | 1e-15 |
| Expand frequency | 100000 | 100000 | 100000 | 100000 |
| LU Factor tol | 100.0 | 1.9 | 10.0 | 5.0 |
| LU Update tol | 10.0 | 1.9 | 10.0 | 5.0 |

**Algorithm 1** is described further in Algorithm 2 where \( \delta_1 \) and \( \delta_2 \) are Feasibility and Optimality tolerances. MINOS terminates part 2 when the bounds on (scaled) variables are satisfied to within \( \delta_1 \) and when \( z_j/(1 + \|y\|_\infty) \) has the correct sign to within \( \delta_2 \). Table 2 shows the default runtime options for Double MINOS and the options chosen for each step of DQQ.

Steps D and Q1 are usually sufficient. In case Q1 is interrupted, Q2 provides some insurance and ensures that the tolerances \( \delta_1 \) and \( \delta_2 \) are imposed upon the original problem (not the scaled problem). For conventional Double solvers, it is reasonable to set tolerances in the range \( 10^{-6} \) to \( 10^{-8} \). For Quad MINOS, we set \( \delta_1 = \delta_2 = 10^{-15} \) to be sure of capturing fluxes \( v_j \) as small as \( O(10^{-10}) \).

In the following, we apply procedure DQQ to 14 challenging linear models arising in biology, economics, and image reconstruction.

**Small M models.** Of the 98 metabolic network models in the UCSD Systems Biology repository [52], A. Ebrahim was able to parse 83 models [4] and compute solutions with a range of solvers [5]. We constructed MPS files for the 83 models [3] and solved them via DQQ. Most models have less than 1000 metabolites and reactions. A histogram for the DQQ solve times is in Figure 3. Almost all models solved in less than 0.08 seconds, and many in less than 0.01 seconds. The total time was less than 3 seconds. In contrast, the exact arithmetic solver needs a day [2].

**Large ME models.** The results of DQQ on three large ME models TMA_ME, GlcAerWT, and GlcAlif are shown in Tables 3 and 4 including problem dimensions \( (m, n) \), number of nonzero entries \( \text{nnz}(S) \), norms of the optimal primal and dual solution vectors \( (x^*, y^*) \), number of iterations, runtime, objective value, primal and dual infeasibility after each step (Pinf and Dinf), and total solve time for each model. (The constraints \( (8b) − (8d) \) are satisfied to within Pinf, and \( z_j/(1 + \|y^*\|_\infty) \) has the correct sign to within Dinf.)

**TMA_ME** developed by Lerman et al. [28] has some large matrix entries \( |S_{ij}| \) and many small solution values \( v_j \) that are meaningful to systems biologists. For example, transcription and translation rates can have values \( O(10^{-10}) \) or less, which is much smaller than metabolic reactions. These small values are linked to large matrix entries arising from build-
ing large macromolecules from smaller constituents [45]. The ME part of the model also contains small matrix entries. For instance, enzyme levels are estimated in ME models by dividing certain metabolic fluxes by “effective rate constants.” Because these constants are typically large (e.g., 234,000 h⁻¹), the matrix entries (the inverse of the rate constants) become small. In step D, almost all iterations went on finding a feasible solution, and the objective then had the correct order of magnitude (but only one correct digit).

This was the first ME model that we used for Quad experiments. The data M, c, ℓ, u in [3] came as a Matlab structure with c_j = 0, ℓ_j = 0, u_j = 1000 for most j, except four variables had smaller upper bounds, the last variable had moderate positive bounds, and 64 variables were fixed at zero. The objective was to maximize flux v_17533. We output the data to a plain text file. Most entries of M were integers (represented exactly), but about 5000 S_j values were of the form 8.037943687315e-01 or 3.488862383191e-06 with 13 significant digits. The text data was read into Double and Quad versions of a prototype Fortran 90 implementation of SQOPT [16].

For the present work, 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 3.48887e-06 and in total about 5000 S_j values had O(10⁻⁶) relative perturbations of this kind. This was a fortuitous limitation for the ME models. 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 10⁶ or even 10¹² (as estimated by LU_SOL [24] each time SQOPT or MINOS factorizes the current basis B). However, in comparing Quad SQOPT and Quad MINOS with SoPlex [55, 44] and the exact simplex solver QSopt_ex [14], we observe that the final objective values for TMA_ME in Matlab data reported by QSopt_ex and Quad SQOPT match in every digit (Table 5). Moreover, the objective value achieved by Quad MINOS on the perturbed data in MPS format agrees to 5 digits of the results from the exact solver QSopt_ex on the “accurate” data. These results show the robustness of the TMA_ME model and our 34-digit Quad solvers.

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 on slightly different data. Some example solution values are given in Table 6.

Among all j for which max(v_j, w_j) > δ_1 = 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

### Table 3: Three large ME biochemical network models TMA_ME, GlcAerWT, GlcAlift [28, 45]. Dimensions of m x n constraint matrices S, size of the largest optimal primal and dual variables z*, y*, number of iterations and runtimes in seconds for each step, and the total runtime of each model.

| ME model | TMA_ME | GlcAerWT | GlcAlift |
|----------|--------|----------|----------|
| m        | 18210  | 68300    | 69529    |
| n        | 17535  | 76664    | 78793    |
| nz(S)    | 336302 | 926357   | 928815   |
| max | 2.1e+04 | 8.0e-05  | 2.0e-05  |
| ∥v∥_∞   | 5.9e+00 | 6.3e-07  | 6.3e-07  |
| ∥y∥_∞   | 1.1e+00 | 2.4e-07  | 2.4e-07  |
| D itns  | 21026  | 47718    | 93857    |
| D time  | 350.9  | 10567.8  | 15913.7  |
| Q1 itns | 597    | 4287     | 1631     |
| Q1 time | 29.0   | 1958.9   | 277.3    |
| Q2 itns | 0      | 4        | 1        |
| Q2 time | 5.4    | 72.1     | 44.0     |
| Total time | 385  | 12599    | 16235    |

### Table 4: Three large ME biochemical network models TMA_ME, GlcAerWT, GlcAlift [28, 45]. Optimal objective value of each step, Pinf and Dinf = final maximum primal and dual infeasibilities (log_{10} values tabulated, except − means 0). Bold figures show the final (step Q2) Pinf and Dinf.

| ME model | Step | Objective | Pinf | Dinf |
|----------|------|-----------|------|------|
| TMA_ME   | D    | 8.3789966820e-07 | −06  | −05  |
|         | Q1   | 8.7036313583e-07 | −25  | −32  |
|         | Q2   | 8.7036313586e-07 | −26  | −32  |
| GlcAerWT | D    | 7.0382449681e+05 | −07  | −26  |
|         | Q1   | 7.0382449681e+05 | −07  | −26  |
|         | Q2   | 7.0382449681e+05 | −07  | −26  |
| GlcAlift | D    | 5.3319574961e+05 | −03  | −01  |
|         | Q1   | 7.0434008750e+05 | −08  | −22  |
|         | Q2   | 7.0434008750e+05 | −18  | −23  |

### Table 5: TMA_ME model. Robustness of objective values computed by four high-accuracy solvers for two slightly different versions of the problem with 13-digit and 6-digit data (from Matlab and MPS data respectively).

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

### Table 6: TMA_ME model. Robustness of small solution values v_j and w_j computed by Quad MINOS for two slightly different versions of the problem. The values come from Matlab and MPS data respectively.

| j  | 107 | 201 | 302 |
|----|-----|-----|-----|
| v_j | 2.336815e-06 | 8.703646e-07 | 1.454536e-11 |
| w_j | 2.336823e-06 | 8.703632e-07 | 1.454540e-11 |
Table 7: TMA_ME model. The values of 9 fluxes $v_j, w_j$ computed by Quad MINOS for two slightly different versions of the problem, revealing robustness of all 9 solution pairs. These values have 1 digit of agreement. Almost all 17535 pairs of values agree to 5 or more digits.

| $j$ | $v_j$  | $w_j$  | Relative difference |
|-----|-------|-------|----------------------|
| 16383 | 6.07e-07 | 2.04e-06 | 0.70 |
| 16459 | 1.71e-06 | 2.18e-06 | 0.22 |
| 16483 | 2.47e-06 | 5.99e-07 | 0.76 |
| 16730 | 1.44e-06 | 7.87e-07 | 0.46 |
| 17461 | 1.71e-06 | 2.18e-06 | 0.22 |
| 17462 | 2.47e-06 | 5.99e-07 | 0.76 |
| 17478 | 6.07e-07 | 2.04e-06 | 0.70 |
| 17507 | 1.44e-06 | 7.87e-07 | 0.46 |
| 17517 | 8.70e-07 | 2.97e-06 | 0.71 |

was zero and there are alternative solutions with the same objective value. The remaining 9 variables had $v_j, w_j$ values shown in Table 7.

We see that the values are small (the same magnitude as the data perturbation) but for each of the nine pairs there is about 1 digit of agreement. We could naturally expect thousands of small solution pairs to differ more, yet for almost all 17535 pairs there are at least 5 digits of agreement.

Note that the efficiency advantage of our Quad solvers is evident: 385 seconds of solution time for DQQ (Total time in Table 3) compared to 2 weeks using exact arithmetic [28].

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

GlcAerWT is a ME model from the detailed study by Thiele et al. [48]. After 33,000 iterations, Double MINOS began to report singularities following updates to the basis LU factors (71 times during the next 15,000 iterations). After 47,718 iterations (D itns in Table 3), step D terminated with maximum primal and dual infeasibilities $O(10^{-4})$ and $O(1)$ (Pinf and Dinf in Table 4). These were small enough to be classified “Optimal”, but we see that the final objective value $-6.7687e-05$ had no correct digits compared to $-7.0382e-05$ in steps Q1 and Q2. For this large model, step Q1 is important. It required significant work: 4,287 iterations costing 1958.9 seconds (Q1 itns and time in Table 3). Step Q2 quickly confirmed the final objective value with high accuracy. This, the largest ME model so far, solved in 12,599 seconds (3.5 hours) compared to an expected time of many weeks for an exact solver.

GlcAlift is motivated by the difficulties with solving TMA_ME and GlcAerWT in Double arithmetic. The lifting technique of [45] (see 1–3 above) was applied to GlcAerWT to reduce some of the large matrix values. For constraints of the form (7) with large values of $\sigma_{\text{max}}$ such as 10,000, lifting introduces variables and constraints of the form $v_1 \leq 100s_1$, $s_1 \leq 100v_2$. The aim of lifting is to remove the need for scaling (and hence the difficulties with unscaling), but with DQQ we do scale in step D because steps Q1 and Q2 follow. Our experience is that lifting improves accuracy for Double solvers but substantially increases the simplex iterations. On GlcAlift, Double MINOS again reported frequent singularities follow basis updates (235 times starting near iteration 40,000). It took 93,857 iterations (D itns in Table 3), twice as many as GlcAerWT, with only a slight improvement in max{Pinf,Dinf} (Table 4). There is still no agreement with the final objective $-7.0434008750e-05$ in steps Q1 and Q2, and the total solve time increased (4.5 hours), mostly in step D.

The objective function for both GlcA models is to maximize variable $v_{100069}$. The fact that the step D objective values have no correct digits illustrates the challenge these models present. Starting from the basis that the Double solver reaches, steps Q1 and Q2 are accurate and efficient. Theoretically, the Q2 objectives for GlcAerWT and GlcAlift should agree, but the limited data precision in MPS files could explain why there is just 3-digit agreement. The Tomlab interface [50] used by Thiele et al. [48] allowed full double-precision data for both problems, and CPLEX obtained a more accurate solution for GlcAlift in [48] than Double MINOS did here.

On the NEOS server [36], Gurobi was unable to solve GlcAerWT with default parameters (numeric error after nearly 600,000 iterations). It performed considerably better on GlcAlift (about 46,000 iterations) but terminated with a warning of unscaled primal/dual residuals 1.07 and 1.22e-06.

Further tests of the DQQ procedure on challenging LO problems are reported below. As for the ME models, the simplex method in Double MINOS usually gives a good starting point for the same simplex method in Quad MINOS. Hence, much of the work can be performed efficiently with conventional 16-digit floating-point hardware to obtain near-optimal solutions. For Quad MINOS, 34-digit floating-point operations are implemented in the compiler’s Quad math library via software (on today’s machines). Each simplex iteration is therefore considerably slower, but the reward is extremely high accuracy. Of significant interest is that Quad MINOS almost invariably achieves far more accurate solutions than requested (see bold figures in Tables 4 and 9). This is a favorable empirical finding.

Discussion
Although most coefficients in stoichiometric matrices are integers (stored exactly in floating-point), the col-
that shrunken volume contains no data. "We surmise that Kahan has anticipated our observed situation, but: "Arithmetic precision is usually extrav-

determine the data's and the desired result's. Often accurate enough if it is somewhat more than twice as 
greater than 6 digits of accuracy. Scaling the column by 

Exact solvers are based on rational arithmetic. There has been considerable work on their implement-

While today’s advanced LO solvers, such as CPLEX, Gurobi, Mosek, and Xpress [22, 41, 20, 9], are effective on a wide range of large and challenging linear (and mixed integer) optimization models, the study by Thiele et al. [18] emphasizes the need for improved reliability in solving FBA and ME models in systems biology. Our DQQ procedure has demonstrated that warm starts with Quad solvers are likely to be acceptably efficient, and that the accuracy achieved exceeds requirements by a very safe margin. Kahan [24] 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” and that “default evaluation in Quad is the humane option.” The “humane” approach of Kahan—use of Quad solvers—is certainly more efficient than applying exact simplex solvers.

An intriguing question remains concerning the bold figures in Tables 4 and 9. The primal and dual so-

where scalars $\alpha, \beta$ are proportional to the smallest molecular mass considered non-zero and the largest molecular mass allowed (e.g., $\alpha = 10^{-4}, \beta = 10^4$). Note that problem (4) involves $S^T$ and is larger than the FBA problem (8) itself. We could not design consistent FBA models in this way unless we were sure of being able to solve (4) effectively. Our work here offers assurance of such capability.

We believe that quadruple-precision solutions are now practical for multiscale applications such as FBA and flux variability analysis (FVA) computations for ME models in systems biology [10, 33, 48, 19, 47], and that they justify increased confidence as systems biologists build ever-larger models to explore new hypotheses about metabolism and macromolecular synthesis. Our DQQ procedure 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 [3] has derived an approach to analog filter design that requires a Quad linear or nonlinear solver to deal with a wide range of frequencies (which must be raised to high powers). This application, like ME models with nonlinear con-

Methods
Multiscale constraint-based modeling. Consider a network of biochemical reactions, represented by a sto-

| $S_{ij}$ | represents the stoichiometry of molecular species $i$ participating as a substrate (negative) or product (positive) in reaction $j$. The evolution of
molecular species concentrations with respect to time \( t \) is given by the ordinary differential equation
\[
\frac{dc(t)}{dt} := Sf(c(t)),
\]
where \( c(t) \in \mathbb{R}_{\geq 0}^n \) is a vector of time-dependent concentrations and \( v(c(t)) : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}^n \) is a nonlinear function of concentrations, with a form that depends on the kinetic mechanism of each reaction.

If one assumes that species concentrations are time-invariant, then the set of all steady-state reaction rates, satisfying \( Sf(c) = 0 \), may be approximated by the linear steady-state constraint \( Sv = 0 \), where \( v \in \mathbb{R}^n \) is a vector of reaction fluxes. Thermodynamic principles and experimental data can also be used to specify lower and upper bound constraints on reaction fluxes \( \ell \leq v \leq u \). Biochemical relationships between the rates of macromolecular synthesis and utilization can be approximated by coupling of the corresponding reaction fluxes \( c \), e.g., pyruvate kinase reaction flux and the synthesis flux of pyruvate kinase in a ME model \( 43 \). Flux coupling can be represented by bounding the ratio between two reaction fluxes with two coupling coefficients
\[
\sigma_{\min} \leq \frac{v_i}{v_j} \leq \sigma_{\max},
\]
where \( v_i \) and \( v_j \) are a pair of non-negative fluxes. This nonlinear constraint can be reformulated into a pair of linear coupling constraints
\[
\sigma_{\min} v_j \leq v_i, \quad v_i \leq \sigma_{\max} v_j,
\]
or more generally a set of linear inequalities \( Cv \leq d \). In addition to the aforementioned physicochemical and biochemical constraints, one may hypothesize a biologically motivated objective. For example, in modeling a growing cell, one may hypothesize that the objective is to maximize the rate of a biomass synthesis reaction. Typically, a biomass synthesis reaction is created with experimentally determined stoichiometric coefficients, each of which represents the relative composition of a cellular biomass constituent. Optimization of a linear combination of reaction fluxes \( c^Tv \) subject to the aforementioned constraints results in the following linear optimization problem:
\[
\begin{align*}
\text{max} & \quad c^Tv \\
\text{s.t.} & \quad Sv = 0, \\
& \quad Cv \leq d, \\
& \quad \ell \leq v \leq u.
\end{align*}
\]
Flux balance analysis of a ME model with coupling constraints results in an ill-scaled instance of this mathematical optimization problem because the stoichiometric coefficients and coupling coefficients vary over many orders of magnitude.

**MINOS implementation.** MINOS \([34, 35]\) is a linear and nonlinear optimization solver implemented in Fortran 77 to solve problems of the form
\[
\text{min} \quad c^Tv + \varphi(v) \quad \text{s.t.} \quad \ell \leq \begin{pmatrix} v \\ Sv \\ f(v) \end{pmatrix} \leq u,
\]
where \( \varphi(v) \) is a smooth nonlinear function and \( f(v) \) is a vector of smooth nonlinear functions. The matrix \( S \) and the Jacobian of \( f(v) \) are assumed to be sparse.

Let Single, Double, and Quad denote the main floating-point formats defined in the 2008 IEEE 754 standard \([23]\) 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 for multiscale problems. This is the reason for our work. Since release 4.6 of the GCC C and Fortran compilers \([14]\), Quad has been available via the long double and real(16) data types. Thus, we have made a Quad version of Double MINOS using the GNU gfortran compiler.

On today’s machines, Double is implemented in hardware, while Quad (if available) is typically implemented in a software library, in this case GCC libquadmath \([15]\). Our aim is to explore combined use of the Double and Quad MINOS simplex solvers for the solution of large multiscale linear programs. We seek greater efficiency than is normally possible with exact simplex solvers.

For Double MINOS, floating-point variables are declared \texttt{real(8)} \((\approx 16\) digits). For Quad MINOS, they are \texttt{real(16)} \((\approx 34\) digits). The data \( S, c, \ell, u \) are stored in Quad even though they are not known to that precision. This allows operations such as \( Sv \) and \( S^Ty \) to be carried out directly on the elements of \( S \) and the Quad vectors \( v, y \). If \( S \) were stored in Double, such products would require each entry \( S_{ij} \) to be converted from Double to Quad at runtime (many times).

The primal simplex solver in MINOS includes geometric mean scaling \([13]\), the EXPAND anti-degeneracy procedure \([17]\), 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 \([29]\). Cold starts use a Crash procedure to find a triangular initial basis matrix. Basis files are used to preserve solutions between runs and to enable warm starts.

Scaling is commonly applied to linear programs to make the scaled data and solution values closer to 1. Feasibility and optimality tolerances can be chosen more easily for the scaled problem, and LU factors of the basis matrix are more likely to be sparse. For geometric mean scaling, several passes are made through the columns and rows of \( S \) to compute a scale factor for each column and row. A difficulty is that the scaled problem may solve to within specified feasibility and optimality tolerances, but when the solution is unscaled it may lie significantly outside the original (unscaled) bounds.

EXPAND tries to accommodate consecutive “degenerate” simplex iterations that make no improvement to the objective function. The problem bounds are effectively expanded a tiny amount each iteration to permit nonzero improvement. Convergence is usually achieved but is not theoretically guaranteed \([21]\). Progress sometimes stalls for long sequences of iterations.

LUSOL bounds the subdiagonals of \( L \) when the current basis matrix \( B \) is factorized as \( P_1BP_2 = LU \) with some permutations \( P_1, P_2 \). It also bounds off-diagonal elements of elementary triangular factors \( L_j \) that update \( L \) in product form each simplex iteration. (The diagonals of \( L \) and each \( L_j \) are implicitly 1.) Maximum numerical stability would be achieved by setting the LU Factor and Update tolerances to be near 1.0, but larger values are typically chosen to balance stability with sparsity.

**Further tests of DQQ.** We report results from the primal simplex solver in Double MINOS and Quad MINOS...
on two sets of challenging LO problems shown in Table 8, and eight Mészáros problematics problems [30]. Dimensions of $m \times n$ constraint matrices $S$, and size of the largest optimal primal and dual variables $x^*$, $y^*$.

| model  | $m$ | $n$ | nz(S) | max | $\|x^*\|_\infty$ | $\|y^*\|_\infty$ |
|--------|-----|-----|-------|-----|-------------|-------------|
| 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 | 5105  | 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 | 111074| 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   |

Line 1 for pilot shows that Double MINOS with cold start and scaling (step D) required 16060 simplex iterations and 9 CPU seconds. The unscaled primal solution $x$ satisfied the constraints in $S$ 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 (step Q1) needed only 29 iterations and 0.3 seconds to obtain a very accurate solution.

Line 3 for pilot shows that in the “insurance” step (Q2), Quad MINOS warm-starting again but with no scaling gave an equally good solution (maximum infeasibilities 0.0 and $O(10^{-32})$) and was not really needed.

The final Double and Quad objective values differ in the 4th significant digit, as suggested by removal of step D’s $O(10^{-3})$ dual infeasibility.

Results for the other pilot problems are analogous.

The Mészáros problematics problems. Our DQQ procedure was initially developed for this set of difficult LO problems collected by Mészáros [30], who names them problematics and notes that “modeling mistakes made these problems “crazy,” but they are excellent examples to test numerical robustness of a solver.” They were provided as MPS files by Ed Klotz [26]. The first two problems have unusually large entries in the constraint matrix $S$. The step D objective value for de063155 has at best 1 digit of precision, and is quite erroneous for de063157. Nevertheless, the step Q1 and Q2 solutions are seen to be highly accurate (small Pinf and Dinf values) when the solution norms are taken into account.

The gen* problems come from image reconstruction, with no large entries in $S$, $v$, and highly degenerate primal solutions $v$. (In steps D and Q1 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, step Q1 gave an almost feasible initial solution (253 basic variables outside their bounds by more than $10^{-15}$ with a sum of infeasibilities of only $O(10^{-8})$), yet over 200,000 iterations were needed in step Q1 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 [11]), and/or steepestd-edge pricing [12], to reduce significantly the total number of iterations. Problems gen1 and gen4 show that step Q2 is sometimes needed to achieve high accuracy.
Problem 130 behaved similarly (80% degenerate iterations in steps D and Q1). The tiny objective value is essentially zero, so we can’t expect the Q1 and Q2 objectives to agree in their leading digits. The 500,000 step Q1 iterations were inadvertently limited to that number, but step Q2 did not have much further to go.

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.

Table 7 shows that Quad MINOS almost invariably achieves far more accurate solutions than requested, in the sense that the maximum primal and dual infeasibilities are almost always far smaller than $10^{-15}$. Thus our procedure for handling the problematic problems seemed appropriate for the systems biology M and ME models. Like the gen* problems, the ME models showed 40–60% degenerate iterations in step D, but fortunately not so many total iterations in step Q1 (see Table 3). This is important for FVA and for ME with nonlinear constraints, where there are many warm starts.

**ME models (FBA with coupling constraints).**

As coupling constraints are often functions of the organism’s growth rate $\mu$, O’Brien et al. [38] consider growth-rate optimization nonlinearly with the single $\mu$ as the objective in (8a) instead of via a linear biomass objective function. Nonlinear constraints of the form

$$v_i \geq \mu \sum_j \frac{v_j}{k_{i,j}}$$

are added to (8b), where $v_i, v_j, \mu$ are all variables, and $k_{i,j}$ is an effective rate constant. Constraints (10) are linear if $\mu$ is fixed at a specific value $\mu_k$. O’Brien et al. [38] 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 problem feasible. Thus, the procedure requires reliable solution of a sequence of related LO problems.

**Flux Variability Analysis (FVA).**

After FBA [8a], FVA examines how far a particular flux $v_j$ can vary within the feasible region without changing the optimal objective significantly (if $\gamma \approx 1$):

$$\begin{align*}
\max \text{ or } \min_{v_j} & \quad v_j \\
\text{s.t.} & \quad Sv = 0, \\
& \quad \epsilon^T v \geq \gamma Z_0, \\
& \quad l \leq v \leq u,
\end{align*}$$

where $0 < \gamma < 1$. Potentially 2n LO problems (11) are solved if all reactions are of interest, with warm starts being used when $j$ is increased to $j + 1$ [20].

For such a sequence of related problems, warm-starting each problem in Quad would be simplest (calling a single solver), but warm-starting in Double and then in Quad could sometimes be more efficient.

**Iterative refinement.**

For the biology models, our aim is to satisfy Feasibility and Optimality tolerances of $10^{-15}$ (close to Double precision). It is reasonable to suppose that this could be achieved within a Double simplex solver by implementing iterative refinement (Wilkinson [21]) for every linear system involving the basis matrix $B$ or $B^T$. This is a more sparing use of Quad precision. For example, each time the current $B$ is factorized directly (typically sparse LU factorization every 100 iterations), the constraints $Sv = 0$ can be satisfied more accurately by computing the primal residual $r = 0 - Sv$ from the current solution $v$, solving $B^T \delta v = r$, and updating $v_B \leftarrow v_B + \delta v_B$. In general, the new $v$ will not be significantly more accurate unless $r$ is computed in Quad. (If $B$ is nearly singular, more than one refinement may be needed.) Similarly for solving $B^T y = c_B$ after refactorization, and for two systems of the form $Bp = a$ and $B^T y = c_B$ each iteration of the simplex method.

By analogy with DQQ, we implemented the following procedure within a test version of Double MINOS. Note that “iterative refinement” in steps R1, R2 means a single refinement for each $B$ or $B^T$ system, with residuals $-S v$, $a - Bp$, $c_B - B^T y$ computed in Quad as just described.

**DRR procedure.**

**Step D** (Cold start with scaling): Apply Double MINOS with moderately strict options. Save the final basis.

**Step R1** (Warm start with refinement and scaling): Start Double MINOS from the saved basis with stricter tolerances and iterative refinement. Save the final basis.

**Step R2** (Warm start with refinement but no scaling): Start Double MINOS from the second saved basis without scaling but with stricter LU tolerances and iterative refinement.

Step D is the same as for DQQ (with no refinement). The runtime options for each step are the same as for DQQ in Table 2 except in steps R1, R2 the tolerances $1e-15$ were relaxed to $1e-9$.

In Table 10 we see that this simplified (cheap) form of iterative refinement is only partially successful, with step R2 achieving only 4, 3, and 2 correct digits in the final objective. For GlcAerWT, steps R1 and R2 encountered frequent near-singularities in the LU factors of $B$
(requiring excessive refactorizations and alteration of $B$), and in step R2, the single refinement could not always achieve full Double precision accuracy for each system. Additional refinements would improve the final $\text{Pinf}$ and $\text{Dinf}$, but would not reduce the excessive factorizations. We conclude that on the bigger ME problems, a Double solver is on the brink of failure even with the aid of conventional (Wilkinson-type) iterative refinement of each system involving $B$ and $B^T$.

**The zoom strategy.** A step toward warm-starting interior methods for optimization was proposed by Saunders and Tenenblat [12] to take advantage of the fact that a low-accuracy solution $(x_1, y_1)$ for a general problem

$$\min \ c^T x \ \text{st} \ Ax = b, \ \ell \leq x \leq u$$

can be obtained relatively cheaply when an iterative solver for linear systems is used to compute each search direction. (The iterative solver must work harder as the interior method approaches a solution.) If $(x_1, y_1)$ has at least some correct digits, the primal residual $r_1 = b - Ax_1$ will be somewhat small ($||r_1|| = O(1/\sigma)$ for some $\sigma \gg 1$) and the dual residual $d_1 = c - A^T y_1$ will be comparably small in the elements associated with the final $B$. If we define

$$b_2 = \sigma r_1, \quad c_2 = \sigma d_1,$$

$$\ell_2 = \sigma (\ell - x_1), \quad u_2 = \sigma (u - x_1)$$

and note that the problem is equivalent to

$$\min \ c_2^T x - y_2^T (Ax - b) \ \text{st} \ Ax = b, \ \ell \leq x \leq u$$

with dual variable $y - y_1$, we see that $x_2$ solves

$$\min \ c_2^T x_2 \ \text{st} \ Ax_2 = b_2, \ \ell_2 \leq x_2 \leq u_2$$

with dual variable $y_2$. Importantly, with $\sigma$ chosen carefully we expect $(x_2, y_2)$ in this “zoomed in” problem to be of order 1. Hence we can solve the problem with the same solver as before (as solvers use absolute tolerances and assume that $A$ and the solution are of order 1). If the computed $(x_2, y_2)$ has at least some digits of accuracy, the correction $x_2 \leftarrow x_1 + \frac{1}{\sigma} x_2, y_2 \leftarrow y_1 + \frac{1}{\sigma} y_2$ will be more accurate than before. The process can be repeated.

**Multiple zooms with the simplex method.** The zoom strategy has been developed extremely carefully and comprehensively by Gleixner et al. [18] for solving LO problems by the simplex method. With repeated zooms (named refinement rounds in [18]), the residuals $(r_1, d_1)$ must be computed with increasingly high precision. Subject to the expense of using rational arithmetic for this purpose, [18] gives extensive results for over 1000 challenging problems and shows that exceptional accuracy can be obtained in reasonable time: only 3 or 4 refinements to achieve $10^{-50}$ precision, and less than 20 refinements to achieve $10^{-99}$. The SoPlex solver [55] [44] is used for each refinement round with feasibility and optimality tolerances set to $10^{-9}$.

**The floating-point solver.** In [18] the authors recognize that much depends on the robustness of the simplex solver used for the original problem and each refinement. The potential difficulties are the same as in each step of our DRR procedure, where Double MINOS is on the brink of failure on the Glc* problems. A practical answer is to use a more accurate floating-point solver such as Quad MINOS for all problems in the refinement process (as anticipated in [18]).

**DQQ serves the current purpose.** In the context of ME models whose non-integer data is accurate to only 4 or 5 digits, we don’t need $10^{-50}$ precision. Tables 4 and 9 show that our DQQ procedure achieves more accuracy than necessary on all tested examples. For models where the Double solver is expected to encounter difficulty, step D can use a reasonable iteration limit. Step Q1 will perform more of the total work with greatly improved reliability.

**Data and software availability.** Double and Quad Fortran 77 implementations of MINOS are included within The COBRA Toolbox [43]. MPS or JSON files for all models discussed are available from [33]. Python code for running Double and Quad MINOS on the BiGG JSON files is also available from [33].

**References**

[1] D. L. Applegate, W. Cook, S. Dash, and D. G. Espinoza. Exact solutions to linear programming problems. Operations Res. Lett., 35:693–699, 2007.

[2] L. Chindelevitch, J. Trigg, A. Regev, and B. Berger. An exact arithmetic toolbox for a consistent and reproducible structural analysis of metabolic network models. Nat. Comm., 5(4803):9 pp, 2014.

[3] J. Dattorro. Convex Optimization & Euclidean Distance Geometry. Meboo Publishing USA, 2015. v2015.09.29, pp 208–214.

[4] A. Ebrahim. Generation of 83 models from UCSD repository. https://github.com/opencobra/m_model_collection/blob/master/load_models.ipynb 2015.

[5] A. Ebrahim. Solution of 83 models from UCSD repository. https://github.com/opencobra/m_model_collection/blob/master/exact_solving_models.ipynb 2015.

[6] Ali Ebrahim, Eivind Almaas, Eugen Bauer, Aarash Bordbar, Anthony P Burgard, Roger L Chang, Andreas Dräger, Iman Famili, Adam M Feist, Ronan MT Fleming, et al. Do genome-scale models need exact solvers or clearer standards? Molecular Systems Biology, 11:831, 2015.

[7] Ali Ebrahim, Joshua A Lerman, Bernhard O Palsson, and Daniel R Hyduke. COBRApy: Constraints-Based Reconstruction and Analysis for Python. BMC Syst Biol, 7:74, 2013.

[8] A. M. Feist, C. S. Henry, J. L. Reed, M. Krummenacker, A. R. Joyce, P. D. Karp, L. J. Broadbelt, V. Hatzimanikatis, and B. O. Palsson. A genome-scale metabolic reconstruction for Escherichia coli K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. Mol Sys Bio, 3(1):e121, 2007.

[9] FICO Xpress Optimization Suite. http://www.fico.com/en/products/fico-xpress-optimization-suite/ 2015.

[10] R. M. T Fleming, N. Vlassis, I. Thiele, and M. A. Saunders. Conditions for duality between fluxes and concentrations in biochemical networks. J. Theoretical Biology, submitted, 2015. http://arxiv.org/abs/1512.02690

[11] R. Fletcher. On Wolfe’s method for resolving degeneracy in linearly constrained optimization. SIAM J. Optim., 24(3):1122–1137, 2014.
[12] J. J. Forrest and D. Goldfarb. Steepest-edge simplex algorithms for linear programming. *Math. Program.*, 57:341–374, 1992.

[13] R. Fourer. Solving staircase linear programs by the simplex method, I: Inversion. *Math. Program.*, 23:274–313, 1982.

[14] GCC, the GNU Compiler Collection. [https://gcc.gnu.org/](https://gcc.gnu.org/) 2016.

[15] The GCC Quad-precision math library application programming interface (API). [http://gcc.gnu.org/onlinedocs/libquadmath/](http://gcc.gnu.org/onlinedocs/libquadmath/) 2016.

[16] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. SIGEST article.

[17] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. A practical anti-cycling procedure for linear and nonlinear programming. *Math. Program.*, 45:437–474, 1989.

[18] A. M. Gleixner, D. E. Steffy, and K. Wolter. Iterative refinement for linear programming. *ZIB Report 15-20*, 2015.

[19] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An algorithm for sparse nonlinear optimization. *Math. Program. Study*, 16:84–117, 1982.

[20] B. A. Murtagh and M. A. Saunders. Large-scale linearly constrained optimization. *Math. Program.*, 14:41–72, 1978.

[21] B. A. Murtagh and M. A. Saunders. A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. *Math. Program. Study*, 16:84–117, 1982.

[22] NEOS server for optimization. [http://www.neos-server.org/neos/](http://www.neos-server.org/neos/) 2014.

[23] Netlib collection of LP problems in MPS format. [http://www.netlib.org/lp/data](http://www.netlib.org/lp/data) 1988.

[24] E. J. O’Brien, J. A. Lerman, R. L. Chang, D. R. Hyduke, and B. O. Palsson. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Molecular Systems Biology*, 9(693):13 pp., 2013.

[25] Jeffrey D. Orth, Ines Thiele, and Bernhard O. Palsson. What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248, 2010.

[26] M. A. Saunders and L. Tenenblat. The zoom strategy for accelerating and warm-starting interior methods. Talk at INFORMS Annual Meeting, Pittsburgh, PA, USA, November 2006.

[27] J. Schellenberger, R. Que, R. M. T. Fleming, I. Thiele, J. D. Orth, A. M. Feist, D. C. Zielinski, A. Bordbar, N. E. Lewis, S. Rahmanian, et al. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature Protocols*, 6(9):1290–1307, 2011.

[28] Soplex: The sequential object-oriented simplex solver. [http://soplex.zib.de](http://soplex.zib.de) 2011.

[29] Multiscale systems biology collaboration. [http://stanford.edu/group/SOL/multiscale/models.html](http://stanford.edu/group/SOL/multiscale/models.html) 2015.

[30] B. A. Murtagh and M. A. Saunders. Large-scale linearly constrained optimization. *Math. Program.*, 14:41–72, 1978.

[31] B. O. Palsson. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, NY, 2006.

[32] QSopt_ex: A simplex solver for computing exact rational solutions to LP problems. [http://www.math.uwaterloo.ca/~bico/qsopt/index.html](http://www.math.uwaterloo.ca/~bico/qsopt/index.html) 2008.

[33] M. A. Saunders and L. Tenenblat. The zoom strategy for accelerating and warm-starting interior methods. Talk at INFORMS Annual Meeting, Pittsburgh, PA, USA, November 2006.

[34] J. Schellenberger, R. Que, R. M. T. Fleming, I. Thiele, J. D. Orth, A. M. Feist, D. C. Zielinski, A. Bordbar, N. E. Lewis, S. Rahmanian, et al. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature Protocols*, 6(9):1290–1307, 2011.

[35] Soplex: The sequential object-oriented simplex solver. [http://soplex.zib.de](http://soplex.zib.de) 2011.

[36] Y. Sun, R. M. T. Fleming, I. Thiele, and B. O. Palsson. A systems biology approach to the evolution of codon use pattern. *PLoS ONE*, 7(9):e1000312, 2012.

[37] LUSOL sparse LU factorization package. [http://stanford.edu/group/SOL/software/lusol](http://stanford.edu/group/SOL/software/lusol) 2013.

[38] C. Mészáros. A collection of challenging LP problems. [http://www.sztaki.hu/~meszaros/public_fpt/lptestset/problematic](http://www.sztaki.hu/~meszaros/public_fpt/lptestset/problematic) 2004.

[39] MOSEK Optimization Software. [http://www.mosek.com/](http://www.mosek.com/) 2014.

[40] Input format for LP data. [http://lp.solve.sourceforge.net/5.5/mps-format.htm](http://lp.solve.sourceforge.net/5.5/mps-format.htm) 1996.
Acknowledgements

We thank Jon Dattorro, Philip Gill, Ed Klotz, Joshua Lerman, Edward O’Brien, Yuekai Sun, and Elizabeth Wong for much help during the course of this work. 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_ex. 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 problems and lengthy discussions of their properties.

This work was supported by the National Institute of General Medical Sciences of the National Institutes of Health [award U01GM102098] and the US Department of Energy [award de-sc0010429]. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

Author contributions

R.F. and M.S. conceived this study. D.M. and M.S. developed the DQQ procedure and designed the manuscript. M.S. developed the Double and Quad MINOS solvers. L.Y. implemented Python interfaces and verified the solvers on linear and nonlinear ME models. R.F. implemented Matlab interfaces within the COBRA toolbox. I.T. highlighted the impact of coupling constraints in ME models and built the largest example, GlcAerWT. All authors read and revised the manuscript.

Additional information

Competing financial interests: The authors declare no competing financial interests.