BIOINFORMATICS

Vol. 00 no. 00 2015
Pages 1–6

Conservation analysis of genome-scale biochemical networks

Nick W. Henderson[1], San Kim[1], Ding Ma[2], Michael A. Saunders[2,*], Yuekai Sun[1], Ronan M. T. Fleming[3], and Ines Thiele[3]

[1]Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, USA
[2]Department of Management Science and Engineering, Stanford University, Stanford, CA, USA
[3]Luxembourg Centre for Systems Biomedicine, University of Luxembourg, Campus Belval, Luxembourg

DRAFT of Tuesday 19th January, 2016

Associate Editor: Dr Xxx Xxxxx

ABSTRACT
Motivation: Conservation analysis is a crucial preliminary step in the analysis of biochemical networks. Emerging genome-scale models of biochemical networks capture the multiscale nature of biological systems and require specialized sparse-matrix algorithms.
Results: We propose methods for conservation analysis of genome-scale biochemical networks based on rank-revealing sparse matrix factorizations. We describe implementations powered by SuiteSparseQR (a sparse QR factorization package) and LUSOL (a sparse LU factorization and update package). We demonstrate the performance of our implementations on genome-scale models from the BiGG database and a considerably larger model (62000×76000).
Availability and Implementation: SuiteSparseQR is available from Davis (SPQR, 2013). It is implemented in C++ with interfaces to MATLAB, C, and C++. LUSOL is available from the Stanford Systems Optimization Laboratory (LUSOL, 2013). There are separate versions in Fortran 77 and Fortran 90, and a MATLAB interface.
Contact: Michael Saunders: saunders@stanford.edu

1 INTRODUCTION

The activity within a biochemical network naturally conserves certain molecular subgroups. Each conserved subgroup contains several molecular species, and the total mass of the species is conserved as the species move around closed loops in the network. The subgroups are called *conserved moieties* or simply *moieties*. The canonical example is the adenine nucleotide moiety (ADP, ATP, AMP). Other examples include NAD/NADH, CoA/Acetyl-CoA, and phosphorylated/unphosphorylated protein. The total amount of each moiety is determined by the initial conditions.

Correct determination of conservation relations is crucial to the analysis of metabolic networks and is an area of active research (e.g., Sauro and Ingalls, 2004; Vallabhajosyula *et al.*, 2006; Terzer *et al.*, 2009; Schryer *et al.*, 2011). Conservation analysis is a crucial first step in the analysis of metabolic networks for evaluating drug targets. Traditional drugs kill pathogens by disrupting metabolite concentrations or reaction fluxes to an extent harmful to the organism. Conservation laws limit the extent to which a drug can affect the concentration of a metabolite. We refer to Bakker *et al.*, 1999, 2000 and Cornish-Bowden and Eisenthal, 2000; Cornish-Bowden and Hofmeyr, 2002 for details about evaluating drug targets. Conservation analysis is also a preliminary step in analyzing the transient behavior of biochemical networks. Many techniques for studying transient behavior, such as implicit time-stepping methods, require the associated Jacobian to have full rank (Reder, 1988). Conserved moieties create rank-deficiencies in the stoichiometric and Jacobian matrices. The rank must be identified, along with sets of independent rows and columns. Sometimes a secondary benefit is a reduction in model size.

2 SYSTEM AND METHODS

The time-evolution of concentrations in a biochemical network is governed by a system of ordinary differential equations:

$$\frac{d}{dt}x(t) = Sv(t), \tag{2.1}$$

where $x(t) \in \mathbf{R}^m$ is a vector of time-dependent concentrations, $S \in \mathbf{R}^{m\times n}$ is the stoichiometric matrix (with rows and columns corresponding to molecular species and chemical reactions), and $v(t) \in \mathbf{R}^n$ is a vector of reaction fluxes. Certain quantities are typically conserved during time-evolution of the network. These conservation relations are manifested as linear dependencies among the rows of S. The purpose of conservation analysis includes the following aims:

A1 Partition S into *independent and dependent rows*. For some permutation P_{row} and with $\operatorname{rank}(S) = \operatorname{rank}(S_{\text{ind}}) \equiv r$, we want to express (2.1) as

$$\frac{d}{dt}P_{\text{row}}x(t) = P_{\text{row}}Sv(t) \quad\equiv\quad \frac{d}{dt}\begin{pmatrix} x_{\text{ind}}(t) \\ x_{\text{dep}}(t)\end{pmatrix} = \begin{pmatrix} S_{\text{ind}} \\ S_{\text{dep}}\end{pmatrix}v(t).$$

A2 Find a well-conditioned *null-space matrix* Z that spans the null space of S^T (thus $S^TZ = 0$). If $z = Zw \in \mathcal{N}(S^T)$, the quantity $z^Tx(t)$ remains constant:

$$\frac{d}{dt}z^Tx(t) = z^T\frac{d}{dt}x(t) = z^TSv(t) = 0.$$

A3 Compute a *link matrix* N that describes the relations among the species: $S = NS_{\text{ind}}$.

A4 Compute a nonsingular *reduced Jacobian* for analyzing transient behavior in bifurcation analysis, frequency analysis, metabolic control analysis, etc. (Reder, 1988).

We focus on aims A1–A3. Note that most species in a biochemical network are involved in just a few reactions, and most reactions involve just a few species, so that most rows and columns of S are *sparse*. For efficiency we

*To whom correspondence should be addressed.

must take advantage of the sparsity of S, while being conscious of a few rather dense rows or columns. Also note that the nullspace and link matrices are likely to be dense and therefore should not be obtained explicitly. Instead we seek linear operator forms for Z and N that allow efficient computation of matrix-vector products Zv, Z^Tw, Nx, N^Ty.

Further computational details about conservation analysis are given by Sauro and Ingalls (2004); Vallabhajosyula *et al.* (2006); Schryer *et al.* (2011). Our aim is to improve upon the numerical tools proposed there.

3 ALGORITHM AND IMPLEMENTATION

We discuss the main matrix factorizations in turn.

SVD For dense matrices S, the most reliable numerical method for estimating $r \equiv \operatorname{rank}(S)$ and obtaining a nullspace matrix Z is the singular value decomposition (SVD) (Golub and Van Loan, 2013). For $S \in \mathbf{R}^{m\times n}$ (with $m < n$ in our case) it has the form

$$S = UDV^T \equiv \begin{pmatrix} U_1 & U_2 \end{pmatrix} \begin{pmatrix} D_1 & \\ & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} = U_1 D_1 V_1^T, \qquad (3.1)$$

where $U \in \mathbf{R}^{m\times m}$ and $V \in \mathbf{R}^{n\times n}$ are orthogonal and $D \in \mathbf{R}^{m\times n}$ is diagonal with nonnegative nonincreasing diagonals. The columns of U and V are left and right singular vectors, and the diagonals of D are singular values. All of U_1, D_1, V_1 have r columns.

A1 Surprisingly, SVD does not suggest a permutation P_{row} to partition the rows of S into S_{ind} and S_{dep}.

A2 Since $U^TS = DV^T$ and thus $S^TU = VD^T = V_1 \begin{pmatrix} D_1 & 0 \end{pmatrix}$, the matrix $Z \equiv U_2$ (columns of U associated with zero singular values) satisfies $S^TZ = 0$ as required. This Z (having orthonormal columns) is perfectly conditioned.

A3 Since A1 is not possible with SVD, nor is A3. If the rows of S_{ind} were known by other means and we had $S_{\text{ind}} = UDV^T$ (of full rank) , the link matrix would be $N = SVD^{-1}U^T$, as indicated by Sauro and Ingalls, 2004, §4.5.1.

Although SVD is the most reliable factorization for determining r, Z, and possibly N, the density of U and V makes SVD computationally expensive for sparse S and thus unsuitable for conservation analysis of genome-scale networks.

Householder QR QR factorization with column interchanges (Golub and Van Loan, 2013) is almost as reliable as SVD for estimating rank and obtaining a nullspace matrix Z. It has the form

$$SP_{\text{col}} = QR \equiv \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix} = Q_1 \begin{pmatrix} R_1 & R_2 \end{pmatrix}, \qquad (3.2)$$

where $P_{\text{col}} \in \mathbf{R}^{n\times n}$ is a permutation, $Q \in \mathbf{R}^{m\times m}$ has orthonormal columns, $R \in \mathbf{R}^{m\times n}$ is upper trapezoidal, and $R_1 \in \mathbf{R}^{r\times r}$ is upper triangular with nonzero diagonals.

A1 As with SVD, these QR factors do not partition S into independent and dependent rows.

A2 In the dense case, Q is formed as a product of Householder transformations and P_{col} is chosen dynamically to maximize the absolute value of the next diagonal of R_1 at each stage (Golub and Van Loan, 2013). Since $Q^TSP_{\text{col}} = R$, the matrix $Z = Q_2$ satisfies $Z^TSP_{\text{col}} = 0$ and hence $S^TZ = 0$ as required. With Q in product form, we can treat Z as the matrix operator $Z = Q\begin{pmatrix} 0 \\ I \end{pmatrix}$. It is perfectly conditioned as for SVD.

A3 Since A1 is not possible with QR, nor is A3.

For large sparse S, Davis has recently shown that *sparse* rank-revealing Householder QR can often be computed efficiently (Davis, 2011). In his SuiteSparseQR package (SPQR, 2013), Davis chooses P_{col} in advance to promote sparsity in the QR factors. As each diagonal of R is formed, if it would be essentially zero, the corresponding column of R is effectively permuted to the end and P_{col} is revised with no impact on the sparsity of Q and R. If $\operatorname{rank}(S)$ is well-defined (as it typically is for stoichiometric S), SuiteSparseQR is likely to find it correctly as $r = \operatorname{rank}(R)$. With Q and hence Z stored in sparse product form (and Z having perfect condition), this promises to be a powerful new tool for conservation analysis, at least for Aim A2.

Householder QR on S^T Transposing S has little effect on SVD or on rank estimation, but may greatly affect the sparsity of QR or LU factors. The analogue of (3.2) is

$$S^TP = Q_1 \begin{pmatrix} R_1 & R_2 \end{pmatrix}, \qquad (3.3)$$

where $r = \operatorname{rank}(S) = \operatorname{rank}(R_1)$. We define

$$Z \equiv P \begin{pmatrix} -R_1^{-1}R_2 \\ I \end{pmatrix}, \qquad N \equiv \begin{pmatrix} I \\ R_2^TR_1^{-T} \end{pmatrix}.$$

A1 Transposing (3.3) gives $P^TS \equiv \begin{pmatrix} S_{\text{ind}} \\ S_{\text{dep}} \end{pmatrix} = \begin{pmatrix} R_1^T \\ R_2^T \end{pmatrix} Q_1^T$, so that P^T partitions S as required.

A2 From (3.3), Z satisfies $S^TZ = 0$. A product $p = Zv$ can be computed by solving $R_1t = R_2v$ by back-substitution and forming $p = \begin{pmatrix} -t \\ v \end{pmatrix}$.

A3 From A1 we have $S_{\text{dep}} = R_2^TQ_1^T = R_2^TR_1^{-T}S_{\text{ind}}$. The link matrix N satisfies $S = NS_{\text{ind}}$ as required. Products with N are similar to those with Z.

Dense Householder QR on S^T was advocated by Vallabhajosyula *et al.* (2006) for moderate-sized networks. In that case, P can be chosen carefully in (3.3) to maximize each diagonal of R_1, and Z should be reasonably well-conditioned. Note: there is no need for Gauss-Jordan reduction on R_1!

For SuiteSparseQR on S^T, Z might not be so well-conditioned because P is chosen to promote sparsity, not to maximize $\operatorname{diag}(R_1)$.

Sparse LU For cases where S contains some rather dense rows or columns, we propose to use the Fortran package LUSOL (Gill *et al.*, 1987, 2005) to compute triangular factors of the form

$$P_{\text{row}}SP_{\text{col}} = LDU \equiv \begin{pmatrix} L_1 & \\ L_2 & I \end{pmatrix} \begin{pmatrix} D_1 & \\ & 0 \end{pmatrix} \begin{pmatrix} U_1 & U_2 \\ & I \end{pmatrix}, \qquad (3.4)$$

where $L \in \mathbf{R}^{m\times m}$ and $U \in \mathbf{R}^{n\times n}$ are lower and upper triangular with unit diagonals, $D \in \mathbf{R}^{m\times n}$ is diagonal, and L_1, D_1, U_1 all have r rows and columns. The permutations P_{row}, P_{col} are chosen so that at each stage, the next column of L and the next row of U are sparse (if possible) but the next diagonal of D is not too small. LUSOL has three *pivot strategies* for striking this balance between sparsity and stability. At each stage, the next pivot element δ is chosen to be a nonzero S_{ij} in the current (modified) S such that row i and column j are reasonably sparse and δ is reasonably large compared to certain other nonzeros:

- **TPP** (*threshold partial pivoting*) compares δ with nonzeros in its own column;
- **TRP** (*threshold rook pivoting*) compares δ with nonzeros in its own column and its own row;
- **TCP** (*threshold complete pivoting*) compares δ with all remaining nonzeros.

For some *stability tolerance* $\tau > 1$, the net effect is that in producing LDU factors, TPP maintains $|L_{ij}| \le \tau$ (only), while TRP and TCP also maintain $|U_{ij}| \le \tau$.

In all cases, L tends to be well-conditioned if τ is rather close to 1 (say 1.5 or 1.1). For TRP and TCP, U also tends to be well-conditioned when τ is close to 1, and the condition and rank of D then reflects the condition and rank of S. The value $\tau = 1$ would generally provide maximum rank-revealing capability, but setting $\tau > 1$ provides some flexibility to retain sparsity.

Define

$$Z \equiv L^{-T} P_{\text{row}}^T \begin{pmatrix} 0 \\ I \end{pmatrix}, \qquad N \equiv \begin{pmatrix} I \\ L_2 L_1^{-1} \end{pmatrix}. \tag{3.5}$$

A1 P_{row} partitions the rows of S as required.

A2 In (3.4), the last $m - r$ rows of $L^{-1} P_{\text{row}} S$ are zero. Hence, Z satisfies $S^T Z = 0$. Vectors of the form $w = Zv$ and $s = Z^T t$ can be obtained by solving well-conditioned triangular systems involving L^T and L respectively.

A3 The link matrix N satisfies $S = N S_{\text{ind}}$ as required. A product $y = Nx$ can be obtained by solving $Lw \equiv L\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} x \\ 0 \end{pmatrix}$ and setting $y_1 = x$ and $y_2 = -w_2$.

If many vectors Zv, $Z^T t$ are needed and if Q from SuiteSparseQR is too dense, LUSOL provides another important tool for conservation analysis. A MATLAB interface to LUSOL is available (Henderson, 2013).

Since LUSOL's rook pivoting stability test is less demanding than for complete pivoting, TRP is likely to give sparser LU factors than TCP, especially if S contains some large entries. TRP provides a practical compromise between TPP and TCP when rank-estimation is required. We note that TPP is sometimes much faster, and may be sufficiently rank-revealing for typical stoichiometric S, especially if we scale S first.

Sparse LU on S^T LDU factors of S^T are already evident in (3.4) as $S^T = U^T D L^T$, but their properties depend on the pivot strategy and they may be more sparse than the factors of S. For clarity we write the analogue of (3.4) as

$$P_1 S^T P_2 = LDU \equiv \begin{pmatrix} L_1 & \\ L_2 & I \end{pmatrix} \begin{pmatrix} D_1 & \\ & 0 \end{pmatrix} \begin{pmatrix} U_1 & U_2 \\ & I \end{pmatrix} \tag{3.6}$$

and define

$$Z \equiv P_2 U^{-1} \begin{pmatrix} 0 \\ I \end{pmatrix}, \qquad N \equiv \begin{pmatrix} I \\ U_2^T U_1^{-T} \end{pmatrix}. \tag{3.7}$$

A1 P_2^T partitions the rows of S as required.

A2 The operator Z satisfies $S^T Z = 0$ and will be well-conditioned if we use TRP or TCP.

A3 The link matrix N satisfies $S = N S_{\text{ind}}$ as required. A product $y = Nx$ can be obtained by solving $U^T w \equiv U^T \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} x \\ 0 \end{pmatrix}$ and setting $y_1 = x$ and $y_2 = -w_2$.

4 MORE ABOUT LUSOL

4.1 Factor, Solve, Update

LUSOL is a set of procedures for computing and updating LU factors of a general sparse matrix $A \in \mathbf{R}^{m \times n}$. The design allows A to be square or rectangular and possibly rank-deficient. The main functions follow.

- **Factor** finds row and column permutation matrices P_{row} and P_{col} and factors L and U such that $A = LU$, where $P_{\text{row}} L P_{\text{row}}^T$ is lower triangular with unit diagonals and bounded subdiagonals (this is "L" in (3.4)), and $P_{\text{row}} U P_{\text{col}}$ is upper triangular (this is "DU" in (3.4)). The main steps are summarized in Algorithm 1.

Algorithm 1 Sparse LU factorization of A

```
for $k = 1$ to $\min(m,n)$ do
    Choose a pivot $\delta \equiv A_{ij}$ in a sparse row $i$ and column $j$
        subject to $|A_{ij}|$ being suitably large according to
        one of the strategies TPP, TRP, TCP
    Record row $i$ and column $j$ in $P_{\text{row}}$ and $P_{\text{col}}$ respectively
    $l \leftarrow A_{\cdot j}/\delta, \quad u^T \leftarrow A_{i\cdot}$
    $L \leftarrow \begin{bmatrix} L & l \end{bmatrix}, \quad U \leftarrow \begin{bmatrix} U \\ u^T \end{bmatrix}, \quad A \leftarrow A - lu^T$
end for
```

Note: If A contains p distinct columns of $\pm I$, their single nonzeros are chosen first. The top p rows of U are of the form $\begin{pmatrix} I_p & U_p \end{pmatrix}$ regardless of any large nonzeros in U_p.

- **Solve** various systems using the LU factors (compute x from given b):

$$Lx = b \qquad Ux = b \qquad Ax = b$$
$$L^T x = b \qquad U^T x = b \qquad A^T x = b.$$

- **Replace column** Update the LU factors and permutations when a column of A is replaced. This is the most common update required.
- **Other updates** Update the LU factors and permutations when a row or column is added or deleted (thus changing the size of A), when a row is replaced, or when a rank-one matrix vw^T is added to A. These updates are available in the Fortran version of LUSOL, but have not been incorporated into the Matlab interface.

The original Factor procedures with TPP pivot strategy are described by Gill *et al.*, 1987. The rank-revealing TRP and TCP strategies are more recent (Gill *et al.*, 2005). The Bartels-Golub update for column-replacement follows the sparse implementation of Reid, 1982 and involves a "forward sweep" of eliminations. The other updates are implemented similarly (some requiring a backward sweep).

Scaling In Algorithm 1, acceptable pivot elements δ will be chosen sooner if A is previously *scaled* to ensure that the largest nonzeros in each row and column are of order 1. (That is, $A \leftarrow RAC$, where R and C are positive-definite diagonal scaling matrices.) The LU factors will be more sparse and rank detection more reliable. We provide a Matlab routine `gmscale.m` based on the geometric-mean scaling method of Fourer (1982). A Fortran version is used in SNOPT (Gill *et al.*, 2005).

4.2 Nullspace operations with lusolZ

Aim A2 involves a nullspace operator Z that can be obtained from (3.5) or (3.7). In both cases, $S^T Z = 0$. Users may need products of the form $w = Zv$ and/or $s = Z^T t$. These operations are implemented in some MATLAB functions that we refer to as lusolZ. A given matrix S is scaled to satisfy $RAC = S$, where R and C are diagonal matrices of row and column scales respectively. LUSOL is used to factorize either A or A^T, depending on a variable `trans` being 0 or 1 respectively. Users may write code of the following form:

```
trans = 1;  % trans = 0; might be more efficient
options       = lusol();
options.pivot = 'TPP';
options.Ltol1 = 2.0;
Z = lusolZ( S,trans,options );
w = lusolZv( Z,v );    % w = Z*v
s = lusolZt( Z,t );    % s = Z'*t
```

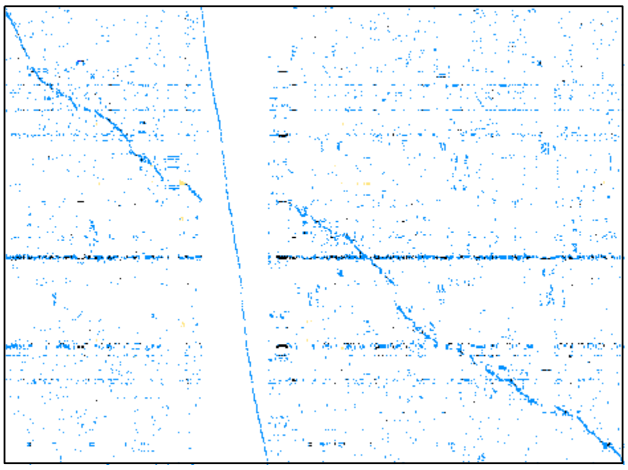

Fig. 1. Recon1: 2766 × 3742 with 14300 nonzeros.

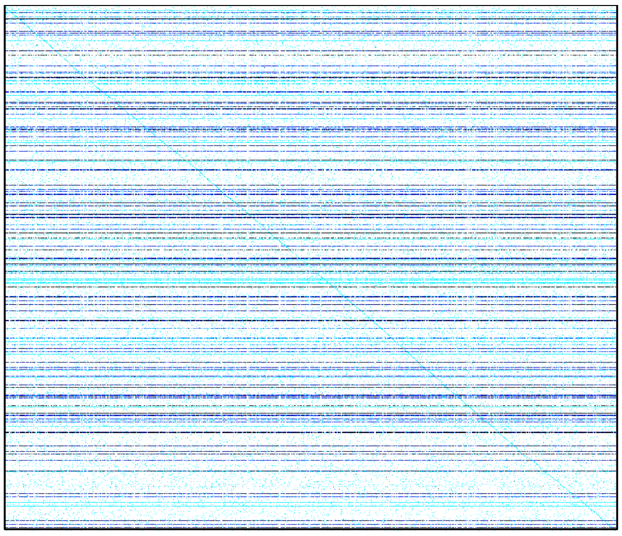

Fig. 2. Th_MA: 15024 × 17582 with 326035 nonzeros.

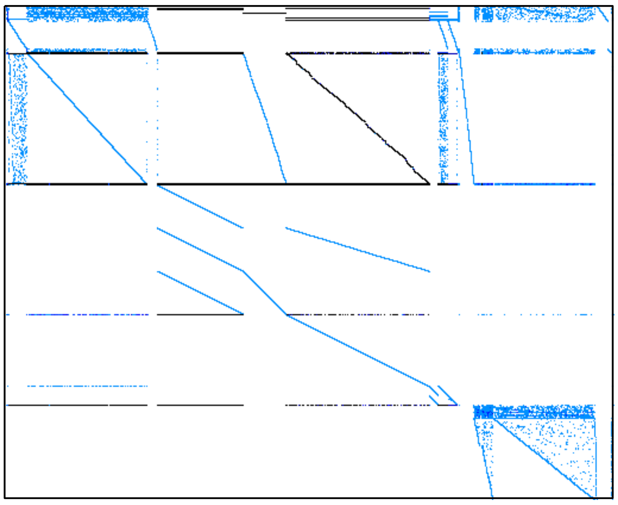

Fig. 3. GlcAerWT: 62212 × 76664 with 913967 nonzeros.

5 NUMERICAL RESULTS

5.1 Factor

We compare sparse QR and LU factors of both S and S^T on three genome-scale models: Recon1, the largest of 9 similar models in the BiGG database (Schellenberger *et al.*, 2010), Th_MA, an ME model from Lerman *et al.* (2012), and GlcAerWT, a larger ME model from Thiele *et al.* (2012). Table 1 lists the model names and dimensions. Figures 1–3 are `cspy` plots of the three stoichiometric matrices S.

Table 2 shows the performance of SPQR, the sparse QR routine from SuiteSparseQR (SPQR, 2013).

Tables 3 and 4 give results for LUSOL with Threshold Partial Pivoting and Threshold Rook Pivoting respectively. Tables 5 and 6 give similar results for LUSOL with S scaled beforehand.

Table 1. Dimensions of $m \times n$ stoichiometric matrices S.

model	m	n	rank(S)	nnz(S)
Recon1	2766	3742	2674	14300
Th_MA	15024	17582	14983	326035
GlcAerWT	62212	76664	62182	913967

Table 2. SPQR nonzeros and factorization times (seconds) and SVD times on S and S^T. See (3.2)–(3.3).

model		nnz(Q)	nnz(R)	SPQR	SVD
Recon1	S	2750	21093	0.1	17.5
Th_MA		844096	10595016	2.5	11hrs
GlcAerWT		1287	916600	0.2	∞
Recon1	S^T	107935	36929	0.1	17.2
Th_MA		624640	605888	0.7	11hrs
GlcAerWT		3573696	4038988	2.7	∞

Table 3. LUSOL nonzeros and factorization times on S and S^T. See (3.4)–(3.6). Threshold Partial Pivoting with $\tau = 2.0$.

model	TPP	nnz(L)	nnz(U)	LUSOL
Recon1	S	721	13585	0.1
Th_MA		7779	324483	0.2
GlcAerWT		533	913781	0.4
Recon1	S^T	9304	7813	0.2
Th_MA		81506	268938	2.7
GlcAerWT		337433	703619	126.7

Table 4. LUSOL nonzeros and factorization times on S and S^T. See (3.4)–(3.6). Threshold Rook Pivoting with $\tau = 2.0$.

model	TRP	nnz(L)	nnz(U)	LUSOL
Recon1	S	4280	16463	0.1
Th_MA		30962	346122	4.1
GlcAerWT		635571	1810491	186.2
Recon1	S^T	12832	7421	0.3
Th_MA		501198	358601	37.8
GlcAerWT		1996892	709448	586.0

Table 5. LUSOL nonzeros and factorization times on scaled S and S^T. See (3.4)–(3.6). Threshold Partial Pivoting with $\tau = 2.0$.

model	TPP	nnz(L)	nnz(U)	LUSOL
Recon1	S	712	13598	0.0
Th_MA	scaled	4043	327461	0.4
GlcAerWT		534	913883	0.5
Recon1	S^T	9797	4612	0.2
Th_MA	scaled	130976	218256	1.1
GlcAerWT		820879	307625	36.9

Table 6. LUSOL nonzeros and factorization times on scaled S and S^T. See (3.4)–(3.6). Threshold Rook Pivoting with $\tau = 2.0$.

model	TRP	nnz(L)	nnz(U)	LUSOL
Recon1	S	867	13604	0.0
Th_MA	scaled	41142	369394	7.2
GlcAerWT		333844	1431318	147.8
Recon1	S^T	9897	4567	0.0
Th_MA	scaled	342958	64891	4.5
GlcAerWT		1181198	301319	122.0

5.2 Nullspace operations with lusolZ

We also compare the average errors, $\|S^T Zv\|_\infty$ and $\|Z^T Sv\|_\infty$, and the average times for computing Zv and $Z^T t$ (where $t = Sv$) using $S = LU$ or $S^T = LU$ on the genome-scale models in BiGG and on the larger model from Thiele. In each case, 50 random vectors v were chosen with uniform random $v_i \in (-0.5, 0.5)$. From the definition of Z and t, we expect the errors to be small. A 2012 Apple MacBook Pro laptop (2.5 GHz Intel Core i5) was used for the Matlab experiments, and times are measured in seconds.

Table 7. Average error ($\|S^T Zv\|_\infty$) and computing time (Zv) of lusolZv with scaled S and S^T. Threshold Partial Pivoting with $\tau = 2.0$.

model	TPP	error	time
Recon1	$S = LU$	3.3e-16	0.0016
Th_MA	scaled	4.0e-16	0.0040
GlcAerWT		2.3e-17	0.0105
Recon1	$S^T = LU$	5.4e-16	0.0020
Th_MA	scaled	3.5e-15	0.2247
GlcAerWT		4.3e-17	0.0460

Table 8. Average error ($\|Z^T Sv\|_\infty$) and computing time ($Z^T t$) of lusolZt with scaled S and S^T. Threshold Partial Pivoting with $\tau = 2.0$.

model	TPP	error	time
Recon1	$S = LU$	1.4e-15	0.0014
Th_MA	scaled	1.8e-14	0.0039
GlcAerWT		3.5e-17	0.0133
Recon1	$S^T = LU$	1.8e-15	0.0007
Th_MA	scaled	1.1e-13	0.0087
GlcAerWT		6.7e-17	0.0264

6 DISCUSSION

The Matlab software is available from lusolZ (2016).

FUNDING

This work was supported by the Office of Naval Research [award N00014-11-1-0067] and 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 Joshua Lerman at UC San Diego for providing the stoichiometric matrix named Th_MA here.

Conflicts of interest: None declared.

REFERENCES

Bakker, B. M., Michels, P. A. M., Opperdoes, F. R., and Westerhoff, H. V. (1999). What controls glycolysis in bloodstream form Trypanosoma brucei. *J. Biol. Chem.*, **274**(21), 14551–14559.

Bakker, B. M., Westerhoff, H. V., Opperdoes, F. R., and Michels, P. A. M. (2000). Metabolic control analysis of glycolysis in trypanosomes as an approach to improve selectivity and effectiveness of drugs. *Mol. Biochem. Parasitol.*, **106**(1), 1–10.

Cornish-Bowden, A. and Eisenthal, R. (2000). Computer simulation as a tool for studying metabolism and drug design. In *Technological and Medical Implications of Metabolic Control Analysis*, pages 165–172. Springer.

Cornish-Bowden, A. and Hofmeyr, J.-H. (2002). The role of stoichiometric analysis in studies of metabolism: an example. *J. Theoret. Biol.*, **216**(2), 179–191.

Davis, T. A. (2011). Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization. *ACM Trans. Math. Softw.*, **38**(1), 8:1–8:22.

Fourer, R. (1982). Solving staircase linear programs by the simplex method. 1: Inversion. *Math. Program.*, **23**, 274–313.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1987). Maintaining LU factors of a general sparse matrix. *Linear Algebra Appl.*, **88**, 239–270.

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.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins University Press, 4 edition.

Henderson, N. W. (2013). Matlab interface to LUSOL. `https://github.com/nwh/lusol`.

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. O. (2012). In silico method for modelling metabolism and gene product expression at genome scale. *Nature Communications*, **3**(929), 10 pp.

LUSOL (2013). Sparse LU factorization package. `http://stanford.edu/group/SOL/software/lusol`.

lusolZ (2016). lusolZ: Nullspace of sparse matrix via LUSOL. `http://stanford.edu/group/SOL/software/lusolZ`.

Reder, C. (1988). Metabolic control theory: a structural approach. *J. Theoret. Biol.*, **135**(2), 175–201.

Reid, J. K. (1982). A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. *Math. Prog.*, **24**(1), 55–69.

Sauro, H. M. and Ingalls, B. (2004). Conservation analysis in biochemical networks: computational issues for software writers. *Biophys. Chem.*, **109**(1), 1–15.

Schellenberger, J., Park, J., Conrad, T. M., and Palsson, B. Ø. (2010). BiGG: a biochemical genetic and genomic knowledgebase of large-scale metabolic reconstructions. *BMC Bioinform.*, **11**(1), 213.

Schryer, D. W., Vendelin, M., and Peterson, P. (2011). Symbolic flux analysis for genome-scale metabolic networks. *BMC Systems Biology*, **5**(81), 13 pp.

SPQR (2013). Sparse QR factorization package. `http://www.cise.ufl.edu/research/sparse/SPQR/`.

Terzer, M., Maynard, N. D., Covert, M. W., and Stelling, J. (2009). Genome-scale metabolic networks. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, **1**(3), 285–297.

Thiele, I., Fleming, R. M. T., Que, R., Bordbar, A., Diep, D., and Palsson, B. O. (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.

Vallabhajosyula, R. R., Chickarmane, V., and Sauro, H. M. (2006). Conservation analysis of large biochemical networks. *Bioinform.*, **22**(3), 346–353.