Available online at www.sciencedirect.com
SCIENCE DIRECT®
Mathematical and Computer Modelling 44 (2006) 678–691
MATHEMATICAL AND COMPUTER MODELLING
www.elsevier.com/locate/mcm

Chemical equation balancing: An integer programming approach

S.K. Sen[a,*], Hans Agarwal[b], Sagar Sen[c]

[a] *Department of Mathematical Sciences, Florida Institute of Technology, 150 West University Boulevard, Melbourne, FL 32901-6975, USA*
[b] *Energy Research Center, Lehigh University, 117ATLSS Drive, Bethlehem, PA 18018, USA*
[c] *Modeling, Simulation and Design Laboratory, School of Computer Science, McGill University, McConnell Engineering Building, Room 202, 3480 University Street, Montreal, Quebec, Canada H3A 2A7*

Received 2 February 2006; accepted 6 February 2006

Abstract

Presented here is an integer linear program (ILP) formulation for automatic balancing of a chemical equation. Also described is a integer nonlinear programming (INP) algorithm for balancing. This special algorithm is polynomial time $O(n^3)$, unlike the ILP approach, and uses the widely available conventional floating-point arithmetic, obviating the need for both rational arithmetic and multiple modulus residue arithmetic. The rational arithmetic is unsuitable due to intermediate number growth, while the residue arithmetic suffers from the lack of a priori knowledge of the set of prime bases that avoids a possible failure due to division by zero. Further, unlike the floating point arithmetic, both arithmetics are not built-in/standard and hence additional programming effort is needed. The INP algorithm has been tested on several typical chemical equations and found to be very successful for most problems in our extensive balancing experiments. This algorithm also has the capability to determine the feasibility of a new chemical reaction and, if it is feasible, then it will balance the equation and also provide the information if two or more linearly independent balancings exist through the rank information. Any general method to solve the ILP is fail-proof, but it is not polynomial time. Since we have not encountered truly large chemical equations having, say, 1000 products and reactants in a real-world situation, a non-polynomial ILP solver is also useful. A justification for the objective functions for ILP and INP algorithms, each of which produces a unique solution, is provided.

Keywords: Balancing chemical equations; Floating-point arithmetic; Integer program; Linear system; Minimum-norm least-squares solution; NP-complete problems

1. Introduction

Anybody having chemistry as a subject is bound to come across balancing chemical equations. They would be doing this balancing by (human) inspection/trial-and-error. In fact, quite often the chemistry students memorize important chemical equations for the purpose of performing well in a time-bound test/examination. This was certainly the situation in pre-computer days. With the advent of widely used programmable pocket calculators, students should

* Corresponding author. Tel.: +1 321 674 7714; fax: +1 321 674 7412.
E-mail addresses: sksen@fit.edu (S.K. Sen), haa6@lehigh.edu (H. Agarwal), sagar.sen@cs.mcgill.ca (S. Sen).

0895-7177/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2006.02.004

be able to balance a chemical equation in seconds using a calculator with the appropriate software, as they can find the value of trigonometric functions in a fraction of a second in their day-to-day life as well as in an examination. The days of log, trigonometric, and statistical tables are practically over. Of course, though statistical tables are still used to some extent, all the important distribution functions can be computed readily in most calculators by pressing just a couple of keys. The student's dependence on the calculator has grown so much that, without it, it is often difficult for most of them to obtain the (positive) square-root of a given decimal number, as the deterministic method for square-rooting seems to have been forgotten/not learnt.

A linear program *Maximize* $c^t x$ *subject to* $Ax = b, x \geq 0$ has the first polynomial time $O(n^{3.5})$ interior-point algorithm [1] based on a projective transformation for its solution in the real number model. The symbol t denotes the transpose. Since then, several polynomial-time algorithms have been developed [2]. This is certainly an important development after nearly three decades of the extensive use of the famous simplex algorithm (and its variations) developed by Dantzig [3] during the early 1950s. The simplex algorithm is exponential time (slow) in the worst case, although for most real-world problems it behaves more like a polynomial-time (fast) algorithm and is still being widely used along with polynomial-time algorithms.

An integer linear program (ILP) *Maximize* $c^t x$ *subject to* $Ax = b, x \geq 0$ *and integer* is, on the other hand, one of a class of hardest possible problems (some of them solvable sometimes quickly by inspired guessing). Many important combinatorial problems, such as the traveling salesman problem (TSP), are equivalent to ILPs. In other words, ILP is NP-complete [4]. Technically, the NP-complete problem is the one of determining whether an integer vector x exists, subject to the constraints and making a linear function of n variables greater than a predetermined constant. There are several efficient solution methods for *particular cases* of ILPs. But it is unlikely that there is an efficient solution method for all ILPs, since this would imply NP = P, a conjecture widely believed to be false. In fact, we have not yet been able to prove or disprove the impossibility of the existence of an efficient algorithm for all ILPs. We do not yet have any revelation (or conscious feeling) of a possible efficient algorithm for all ILPs. It is worth mentioning that we had the conscious feeling that a real number (the totality of a rational and an irrational number) is non-countably infinite from time immemorial. A mathematical proof, however, eluded mathematicians for centuries until Cantor's famous diagonal argument [4–7] came into existence in the nineteenth century. This (solving a general ILP in polynomial time) is indeed an open problem of immense importance, and may some day be either solved or shown to be unsolvable. It is probably not out of place to mention that a couple of important open problems, such as that of developing an efficient algorithm for linear programs and that for testing for primality of a number [1,8], have been solved in the recent past. These successes have a positive psychological effect among mathematicians/scientists pursuing vitally important open problems with renewed vigor.

A *chemical reaction* is a process in which one set of chemicals is changed into a new set of chemicals. The initial chemicals are called the *reactants.* The newly formed chemicals are called the *products*. The products are new chemicals with properties that are different from those of the reactants. A chemical reaction involves the breaking of one set of chemical bonds in the reactants and the formation of a new set of bonds in the products.

The branch of chemical mathematics called *Stoichiometry* deals with the weight relations determined by chemical equations and formulas. Accordingly the balancing of equations is important in this area. Since a chemical reaction (when it occurs/is feasible) is a natural process, the consequent equation is always consistent [4]. Hence we must have a non-trivial solution and we should be able to obtain it assuming its existence, since such an assumption is absolutely valid and does not introduce any error. However, if the reaction is infeasible, then the only solution is a trivial one, i.e., the coefficients of the reactants and the products will be simply zero (implying the impossibility of the reaction). Most methods for balancing are inspection/trial-and-error based and utilize a set of rules to find the change in oxidation number or to balance the gain in or loss of electrons or to balance the half-reactions [9–11].

Trial-and-error-free deterministic methods based on either computing an exact g-inverse [12] or Hermite normal form [13,14] have been developed [9,15,16]. These methods, using multiple modulus residue arithmetic or rational arithmetic, solve exactly non-trivial unconstrained equation balancing problem involving a linear homogeneous algebraic system. Both these arithmetics are unconventional, very much unlike the widely used floating point arithmetic, and need significant additional programming effort. While the residue arithmetic suffers from the lack of a priori knowledge of the set of prime bases, which avoids a possible failure due to a division by zero, the rational arithmetic is undesirable due to intermediate number growth. The deterministic approach is important, since it enables us to classify the chemical reactions as:

(i) *infeasible* when the reaction matrix (the coefficient matrix of the linear homogeneous system) is non-singular (and hence invariably square, as a non-square matrix is never non-singular) or, equivalently, the number of its rows = the number of its columns = its rank or, equivalently, its nullity is zero;
(ii) *unique* (within relative proportions) when the nullity of the reaction matrix is one; or
(iii) *non-unique* when its nullity is two or more.

In Section 2, we propose an integer linear program (ILP) formulation for balancing the chemical equation through its matrix representation. This ILP is NP-complete and hence intractable when the problem is physically large. We are not first going to solve the ILP algorithm and certainly we will not use it when the number of reactants and products is very large. In practice, i.e., in the real-world environment, we are least likely to encounter very large ILP models that are intractable on a personal computer. In the background of the ILP formulation, it will be easy to appreciate the corresponding integer nonlinear program (INP) formulation for the equation. The INP formulation and the concerned solution in polynomial time using just the readily available floating point arithmetic are also included in Section 2. Section 3 presents a modified version of the new algorithm [17] which is physically concise, $O(mn^2)$, and computes the rank and the solution vector of the rectangular system. The modification is oriented towards balancing chemical equations. Typical examples, which include, in addition to some of our own selected examples, those considered by Krishnamurthy [9] and Blakley [15] as well as those in Dickson [11], are presented in Section 4. These examples will, we believe, enable the reader to get the full flavor of the proposed polynomial-time INP solver as well as the non-polynomial-time ILP solver in the background of the well-thought-out examples selected by Krishnamurthy [9] and Blakley [15] computing an exact generalized matrix inverse[1]/Hermite normal form. Section 5 consists of conclusions.

2. Integer linear/nonlinear program formulation

For ease of understanding, consider a specific skeletal chemical equation

$$x_1\mathrm{KNO_3} + x_2\mathrm{C} \rightarrow x_3\mathrm{K_2CO_3} + x_4\mathrm{CO} + x_5\mathrm{N_2}, \tag{1}$$

where the left-hand side of the arrow consists of compounds/elements called reactants while the right-hand side comprises compounds/elements called the products. $x_i\ i = 1(1)5$ are unknown coefficients of the reactants and the products. These coefficients x_i (≥ 1) are all integers to be determined and satisfy three basic principles, viz.,

(i) the law of conservation of mass,
(ii) the law of conservation of atoms,
(iii) the time-independence of Eq. (1)—an assumption usually valid for stable/non-sensitive reactions.

The arrow[2] '$\rightarrow$' here implies that, after completion of the reaction, the mathematical *equal to* '=' depicts equality in the number of atoms on both sides of the arrow. It may be seen that the basic principles define a closed input–output static model [18].

The chemical Eq. (1) can be formulated as a matrix equation. Let there be m distinct elements and n distinct reactants and products in a reaction. Form the $m \times n$ matrix A, called the reaction matrix, whose (i, j)th element a_{ij} is the number of atoms of type i in each compound/element (reactant or product). Set a_{ij} positive or negative according to whether it corresponds to a reactant or a product. Hence Eq. (1) is formulated as the homogeneous equation

$$Ax = 0 \text{ (null column vector)}, \tag{2}$$

where

$$A = \begin{array}{c} \\ \mathrm{K} \\ \mathrm{N} \\ \mathrm{O} \\ \mathrm{C} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathrm{KNO_3} & \mathrm{C} & \mathrm{K_2CO_3} & \mathrm{CO} & \mathrm{N_2} \end{array} \\ \begin{bmatrix} 1 & 0 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 \\ 3 & 0 & -3 & -1 & 0 \\ 0 & 1 & -1 & -1 & 0 \end{bmatrix} \end{array}.$$

[1] A generalized inverse (g-inverse) X of a matrix A is define as $AXA = A$ and is denoted by $X = A^-$ [14,19]. For a singular square/non-square rectangular matrix A, there are infinite possible g-inverses, while the minimum norm least-squares g-inverse A^+ that satisfies the four conditions $AA^+A = A$, $A^+AA^+ = A^+$, $(A^+A)^t = (A^+A)$, and $(AA^+)^t = AA^+$ is unique.

[2] Here, the arrow will also imply the direction of the reaction. The mathematical *equal to* in chemical equations may sometimes be used to imply reversible reaction. However, the context tells us precisely the meaning of the symbol.

Table 1
Possible cases of balancing chemical equations

$m \geq n = r$	$k = n - r = 0$, i.e., 0-parameter family of balancing or trivial solution	$x = 0$, i.e. reaction is infeasible
$m \geq n, r = n - 1$	$k = n - r = 1$, i.e., 1-parameter family of balancing or unique[a] solution (within relative proportions)	$x > 0$, i.e., reaction is feasible and is unique
$m \geq n, r < n - 1$	$k = n - r > 1$, i.e., k-parameter family of balancing or k (>1) linearly independent solutions (within relative proportions)	$x > 0$, i.e., reaction is feasible and is non-unique
$m < n,\ r \leq m$	$k = n - r \geq 1$, i.e., k-parameter family of balancing or k (≥ 1) linearly independent solutions (within relative proportions)	$x > 0$, i.e., reaction is feasible and is non-unique if $k > 1$

[a] By the term "unique", we imply that all the infinite possible derived solutions are linearly dependent.

The foregoing matrix A is actually the 4×5 numerical matrix excluding the notations for chemical compounds and elements. For ease of forming the matrix A, the foregoing notations have been used. In this formulation we have used the convention that the elements/compounds (i and j) are taken in the order in which they occur in Eq. (1). Accordingly, the elements of the foregoing 4×5 numerical matrix A are written without the row/column labels. Any other convention can be used, and that will be equally valid so far as the solution is concerned. It may be seen that the row permutation in A may produce different orthogonal projection operators $P = I - A^+A$, where I is the unit matrix of order n and A^+ is the minimum-norm least-squares (or the Moore–Penrose or the pseudo- or the p-) inverse of A [14,19–23].

The general solution of the homogeneous (always consistent) equation $Ax = 0$, where A is an $m \times n$ matrix of rank r, is

$$x = Pz \tag{3}$$

where z is the arbitrary column vector. It is easy to prove that the rank of P is $k = n - r$, the nullity of A. The different cases in Table 1 are possible.

In all the foregoing cases, the integer linear program (ILP) formulation provides a unique solution, i.e., a unique valid balancing. In the real-world situation, we do not have two or more different balancings under identical thermodynamical, pressure, and time conditions, although the foregoing mathematical model may produce two or more linearly independent balancings/solutions. To achieve the true (real) unique balancing, it is necessary to include additional information (equations/inequations) based on properties other than just the law of conservation of atoms. The problem of deriving additional equations/inequations/objective functions for the unique solution needs to be explored further within the limits of all natural laws (not all are known to us).

Let e denote the column vector of order n having all elements 1, A be an $m \times n$ matrix, $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^t$ be a column vector. The ILP and INP formulations are then as follows.

The ILP. The integer linear programming problem is then

$$\text{Minimize} \quad f = \sum_{i=1}^{n} x_i \quad \text{subject to } Ax = 0 \text{ (null column vector)}, x \geq e \text{ and integer.} \tag{4}$$

The INP. The integer nonlinear programming problem, on the other hand, is

$$\text{Minimize} \quad g = x^t x \quad \text{subject to } Ax = 0, x \geq e \text{ and integer.} \tag{5}$$

Although the foregoing two formulations differ only in their objective functions, the solution produced by both could be the same when the orthogonal projection operator $P = I - A^+A$ obtained without computing the p-inverse A^+ using the physically concise algorithm by Sen and Sen [17] has elements that are all non-negative. This non-negativity condition is sufficient but not necessary. It may be seen that P is always symmetric and idempotent (i.e., $P = P^2 = P^3 = \cdots = P^n$).

Allowing Z to be an $m \times n$ null matrix, z to be a null column vector of order m, observing that I is the unit matrix of order n and e is the column vector of order n having all elements 1, and defining

$$R = \begin{bmatrix} A & Z \\ I & -I \end{bmatrix}, \quad b = \begin{bmatrix} z \\ e \end{bmatrix}, \tag{6}$$

the INP (5) yields the INP in the form of *non-homogeneous* equations:

$$\text{Minimize} \quad h = y^t y \quad \text{subject to } Ry = b, \text{ with first } n \text{ components of } y \text{ positive integer} \tag{7}$$

where

$$y_i = \begin{cases} x_i & i = 1(1)n \\ \text{any value}, & i = n+1(1)2n. \end{cases} \tag{8}$$

While solving the INP (7) using the algorithm *cheminpsolver* (presented later), we do not insist in any restriction on the sign as well as the value of the variables $y_{n+1}, y_{n+2}, \ldots, y_{2n}$. In fact, the values of these variables are of no concern to us, as long as we get a positive integer for y_i $i = 1(1)n$, that satisfies Eq. (7). However, it can be seen that, when we get a feasible solution for a feasible reaction, we have $y_i - y_{n+i} = 1$, $i = 1(1)n$.

3. Solving INP using a concise linear system solver

We solve the INP using *cheminpsolver* based on the following new polynomial-time $O(mn^2)$ concise algorithm called LINSOLVER [17].

LINSOLVER. This is a physically concise mathematical algorithm for consistent linear system $A\mathbf{x} = \mathbf{b}$, where A is an $m \times n$ matrix. Let $\mathbf{a}_i^t$ be the i-th row of A. Then $\mathbf{a}_i$ is the column vector, i.e., the i-th row (of A) written as the column. The following algorithm, which is a modified version of the algorithms in Lord et al. [24,25] and which is better from comprehension point of view, finds a particular solution $\mathbf{x}$. It obtains, as a by-product, the orthogonal projection operator $P = I - A^+A$ [14,19,21–23,26,27], without explicitly computing the minimum-norm least-squares inverse $A^+ = A_{mt}^-$. Also, in-built in the algorithm is the computation of the rank r of A.

(* LINSOLVER *)
$P = I$; $\mathbf{x} = \mathbf{0}$; $r = 0$; *for* $i := 1$ *to* m *do begin* ROW ($\mathbf{a}_i$, b_i, P, $\mathbf{x}$); $r := r + c$ *end*
procedure ROW(**a**, b, P, **x**); (* **solves a single equation** $a^t\mathbf{x} = b$*)
begin
$c := 0$; $\mathbf{u} := P\mathbf{a}$; $v := \|\mathbf{u}\|^2$; $s := b - \mathbf{a}^t\mathbf{x}$; *if* $v \neq 0$ *then begin* $P := P - \mathbf{u}\mathbf{u}^t/v$; $\mathbf{x} := \mathbf{x} + s\mathbf{u}/v$; $c := 1$
end else
if $s \neq 0$ *then* $A\mathbf{x} = \mathbf{b}$ is inconsistent and terminate
end;

Complexity of LINSOLVER. The computational complexity of the procedure ROW is $O(n^2)$. Since the number of applications of ROW cannot exceed m, LINSOLVER is $O(mn^2)$.

Computational LINSOLVER. Since the numerical zero in a floating-point arithmetic is not the same as the mathematical zero [4], the algorithm LINSOLVER needs the following modifications, if we need four significant digit accuracy, before it is implemented on a digital computer. Let $\leftarrow$ denote '*is to be replaced by*':

- $v \neq 0 \leftarrow v \geq 0.5 \times 10^{-4}$ abar
- $s \neq 0 \leftarrow s \geq 0.5 \times 10^{-4}$ bbar

where abar $= \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|/(mn)$, bbar $= \sum_{i=1}^{m}|b_i|/m$. If we desire four decimal digit accuracy, then abar and bbar should be removed or, equivalently, abar $=$ bbar $= 1$ should be taken. It can be seen that $\log_{10}(1/\text{relative error})$ gives the number of significant digits up to which the result/quantity is correct, while $\log_{10}(1/\text{absolute error})$ provides the number of decimal digits up to which the quantity is correct. The later one, however, is not useful in most applications.

CHEMINPSOLVER. This algorithm is a generalized version of LINSOLVER and takes care of near-consistent/inconsistent linear systems. If there is human error in entering data or in formulation, the system could

show inconsistency, implying that a human error has been committed. Observe that the natural balancing is always consistent, and so should be the corresponding mathematical model, viz., the chemical equation. The justification of the generalized version will be found in Sen and Sen [17]. In the following Matlab program, the $m \times n$ input matrix A and the $m \times 1$ input vector $\mathbf{b}$ (which has all m elements 0) correspond to the $m \times n$ matrix A and the $m \times 1$ vector b of INP (7), respectively,

```
function [ ]=cheminpsolver(A,b,m,n);
'The matrix A and vector b of the system Ax=b are'
A,b
A=[A zeros(m,n);eye(n) -eye(n)]; b=[zeros(m,1);ones(n,1)];
%Foregoing line replaces original A and b by expanded A and b and these new A
%and b become R and b of eqns. (6) resulting in the non-homogeneous eqn. Ry=b. in (7)
m=m+n; n=2*n;

P=eye(n); sd=0;
x(1:n)=0; x=x';
delb(1:m)=0; delb=delb'; bo=b;
r=0; abar=0; for i=1:m, for j=1:n, abar=abar+abs(A(i,j)); end; end; abar=abar/(m*n);
bbar=0; for i=1:m, bbar=bbar+abs(b(i)); end; bbar=bbar/m;
for i=1:m
u=P*A(i,:)'; v=norm(u)^2; s=b(i)-A(i,:)*x; c=0;
if v<=.00005*abar & abs(s)>=.00005*bbar
 delb(i)=-s; sd=-s; b(i)=b(i)+delb(i); s=0;
elseif v<=.00005*abar & abs(s)<=.00005*bbar
delb(i)=0;
end
if v>=.00005*abar, x=x+u*s/v; P=P-u*u'/v; c=1; delb(i)=0; end
r=r+c;
end
if abs(sd)>.00005*(abar+bbar)*0.5, 'The system Ax=b is inconsistent.', end;

inci=norm(delb)/norm([A,b]);
err=norm(bo-A*x)/norm(x);
'The projection operator P = (I – A^+A) is', P
'The rank of the matrix A is', r
'The inconsistency index is', inci
%'Modification in vector b, i.e., Db is', delb
%'Vector b of the nearest consistent system is', b
%'Solution vector of the consistent/nearest consistent system is', x
%'Error in the solution vector x is', err
if norm(P)<.00005, 'The reaction is infeasible (coeffs. in reaction are all 0)', return; end;
n=n/2; xx=x(1:n); 'Actual sol.',xx, xx=xx/min(xx); xxx=xx,
for k=1:40
 xxx=k*xx,xxx=xxx-10^-10*ones(n,1);
rr=sum(ceil(xxx)-xxx)>.00005 & sum(xxx-floor(xxx))>.00005 & k<=40,
if rr==0, 'Value of rr=', rr, 'Value of k =', k, 'The coefficients are', xxx, break, end;
end;xxx1=xxx+10^-10*ones(n,1);
if k < 40,
'The integer coefficients of the reactants and products are', xxx1,
'The smallest multiplying factor k used for the foregoing coefficients is', k,
else 'The integer solution could not be obtained in this procedure.'; end;
if min(xxx1)<0, 'Since a coefficient is negative, cheminpsolver failed; solve the ILP.', end;
```

if rr==1, 'cheminpsolver has failed, solve the ILP using any ILP solver.', end;

The *cheminpsolver* produces the solution x, where $\|x\| = (x^t x)^{1/2}$ (where x = the first n elements of y) is the minimum out of all possible x. Whenever *cheminpsolver* produces a positive integer solution for the INP model, this solution could be the same as that due to solving the ILP by any method such as the branch and bound technique and the cutting plane method. Since *cheminpsolver* for the INP model is polynomial-time, we first attempt *cheminpsolver*. If it fails, then we solve the ILP model using, say, the software LINDO [28]. Our extensive experiment on a large number of chemical equations shows that *cheminpsolver* succeeds in producing the required solution in most balancings and hence it has distinct scope as a polynomial algorithm, unlike the solvers for ILP, which is exponential. The ILP solver can still be used in practice, as most real-world chemical equations are not considered to be large from the point of view of current computer capability.

4. Examples

We have significantly benefited from the previous authors, viz., Krishnamurthy [9] and Blakley [15], providing typical examples to illustrate their methods, viz., those for computing an error-free generalized matrix inverse and an HNF. Here we have used all these typical examples in addition to others available in Dickson [11]. The purpose of selecting these typical examples is for the reader to check the scope of the proposed INP/ILP solver in the background of the solvers presented by the foregoing authors.

Example 1. As an illustration for *cheminpsolver*, consider the skeletal chemical equation Eq. (1), viz.,

$$x_1\mathrm{KNO_3} + x_2\mathrm{C} \rightarrow x_3\mathrm{K_2CO_3} + x_4\mathrm{CO} + x_5\mathrm{N_2},$$

where the $m \times n$ input matrix A, $m \times 1$ input vector z, m, and n are

$$A = \begin{bmatrix} 1 & 0 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 \\ 3 & 0 & -3 & -1 & 0 \\ 0 & 1 & -1 & -1 & 0 \end{bmatrix}, \quad z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad m = 4, n = 5.$$

The Matlab command is then

≫ **cheminpsolver(A, z, m, n)**

The corresponding non-homogeneous system produced by *cheminpsolver* is $Ry = b$, where R, b are given by Eq. (6):

$$R = \begin{bmatrix} A & Z \\ I & -I \end{bmatrix} = \begin{bmatrix} 1 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & -3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} z \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

Using the Matlab program *cheminpsolver*, we obtain, besides rank, the inconsistency index (0 here) and the orthogonal projection operator P, the required integer coefficients of the reactants and products. These are

2.0000, 4.0000, 1.0000, 3.0000, 1.0000.

The smallest multiplying factor k used to multiply the original solution produced by LINSOLVER to obtain the foregoing integer coefficients is $k = 1$. Hence, observing that only the extensively available built-in floating-point arithmetic is used, the balanced equation is:

$$2\mathrm{KNO_3} + 4\mathrm{C} \rightarrow \mathrm{K_2CO_3} + 3\mathrm{CO} + \mathrm{N_2}.$$

Example 2. Consider the skeletal chemical equation [15]

$$x_1H_3PO_4 + x_2MgSiO_3 + x_3CF_2Cl_2 + x_4NaAlF_4 + x_5KI + x_6PbCrO_4 + x_7FeSO_4 + x_8BrCl + x_9Ca(CN)_2 + x_{10}SO_2 + x_{11}H_2 \rightarrow x_{12}PI_3 + x_{13}MgCO_3 + x_{14}Na_2SiO_3 + x_{15}PbBr_2 + x_{16}CrCl_3 + x_{17}KAl(OH)_4 + x_{18}Fe(SCN)_3 + x_{19}CaF_2 + x_{20}H_2O.$$

Omitting the square brackets, the 19×20 input matrix A is given by $A =$

$$\begin{array}{cccccccccccccccccccc}
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & -4 & 0 & 0 & -2 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 3 & 0 & 0 & 0 & 4 & 4 & 0 & 0 & 2 & 0 & 0 & -3 & -3 & 0 & 0 & -4 & 0 & 0 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -3 & 0 & 0 \\
0 & 0 & 2 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0
\end{array}$$

while the 19×1 input vector z is $z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t$, $m = 19$, $n = 20$. The Matlab command is then

$\gg$ **cheminpsolver(A, z, m, n)**

The required integer coefficients of the reactants and products are (omitting trivial zeros after the decimal point),

2, 3, 3, 6, 6, 6, 10, 12, 15, 20, 88, 2, 3, 3, 6, 6, 6, 10, 15, 79.

The smallest multiplying factor k used to multiply the original solution produced by LINSOLVER to obtain the foregoing integer coefficients is $k = 2$. Hence the balanced chemical equation is

$$2H_3PO_4 + 3MgSiO_3 + 3CF_2Cl_2 + 6NaAlF_4 + 6KI + 6PbCrO_4 + 10FeSO_4 + 12BrCl + 15Ca(CN)_2 + 20SO_2 + 88H_2 \rightarrow 2PI_3 + 3MgCO_3 + 3Na_2SiO_3 + 6PbBr_2 + 6CrCl_3 + 6KAl(OH)_4 + 10Fe(SCN)_3 + 15CaF_2 + 79H_2O.$$

Examples 3–7. We have considered the following chemical equations [9,14] in their skeletal form and solved them using *cheminpsolver*, where the smallest multiplying factor used to multiply the original solution produced by LINSOLVER to obtain the foregoing integer coefficients is k:

$$Al + 4HNO_3 \rightarrow Al(NO_3)_3 + NO + 2H_2O, \quad k = 1$$

$$2NO_2 + HClO + H_2O \rightarrow 2HNO_3 + HCl, \quad k = 1$$

$$3As_2S_3 + 4H_2O + 28HNO_3 \rightarrow 28NO + 6H_3AsO_4 + 9H_2SO_4, \quad k = 3$$

$$KClO_3 + 6HCl \rightarrow KCl + 3Cl_2 + 3H_2O, \quad k = 1$$

$0NO_2 + 0HClO \rightarrow 0HNO_3 + 0HCl$, infeasible since reaction matrix is non-singular (rank $r = m = n = 4$).

Example 8 (*Cheminpsolver Failed; ILP Model Solved Using LINDO*). The balanced form of the skeletal equation is

$$2KClO_3 + 4HCl \rightarrow 2KCl + 2ClO_2 + Cl_2 + 2H_2O.$$

This is an example [9,14] where LINDO [28] for solving the linear program/integer linear program has been used. In fact, the ILP model is more general than the INP model presented here. The inputs of LINDO, which here uses the branch and bound technique [28], are (written in three columns to conserve space, although in actual LINDO inputs all the entries will be in a single column):

min	**x1>=1**	**gin x1**
x1+x2+x3+x4+x5+x6	**x2>=1**	**gin x2**
st	**x3>=1**	**gin x3**
x1-x3=0	**x4>=1**	**gin x4**
x1+x2-x3-x4-2x5=0	**x5>=1**	**gin x5**
3x1-2x4-x6=0	**x6>=1**	**gin x6**
x2-2x6=0	**end**	

The LINDO solution, after clicking 'solve' twice', is $[2\,4\,2\,2\,1\,2]^t$. It is observed that several elements of the projection operator P, while using *cheminpsolver*, are negative, which are undesirable, although *cheminpsolver* has been found to solve all balancing problems with $P \geq 0$ (*null matrix*) and even many others with some element(s) of P negative. In the foregoing example, if we replace the original non-integral non-degenerate solution produced by LINSOLVER (embedded inside *cheminpsolver*) by the integral part only, then the required solution is reached. However, we are yet to justify such a solution (probably valid) mathematically. In our extensive experiments on chemical equation balancing, this is one of the few examples where the ILP solution procedure has been used.

Example 9. We illustrate the two-parameter family of balancing, i.e., two linearly independent solutions considering the example, viz., the skeletal chemical equation $x_1CO + x_2CO_2 + x_3H_2 \rightarrow x_4CH_4 + x_5H_2O$ [15]. The inputs, i.e., the reaction matrix A, right-hand side vector z, and the orders m and n of A, are

$$A = \begin{bmatrix} 1 & 1 & 0 & -1 & 0 \\ 1 & 2 & 0 & 0 & -1 \\ 0 & 0 & 2 & -4 & -2 \end{bmatrix}, \quad z = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad m = 3, n = 5.$$

The matrix A has rank 3 produced by LINSOLVER of *cheminpsolver*. This rank implies, for this 3×5 matrix, that the nullity is $n - r = 5 - 3 = 2$. The Matlab command

$\gg$ **cheminpsolver(A, z, m, n)**

then produces the solution $\begin{bmatrix} 2 & 1 & 10 & 3 & 4 \end{bmatrix}^t$. Hence, a balanced equation is

$$2CO + CO_2 + 10H_2 \rightarrow 3CH_4 + 4H_2O.$$

Another linearly independent solution can be obtained by allowing $x_1 = 1, x_2 = 1$ and then solving the resulting non-homogeneous equation, observing that the coefficient matrix is non-singular:

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ 2 & -4 & -2 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} -2 \\ -3 \\ 0 \end{bmatrix}$$

hence we get $x_3 = 7, x_4 = 2, x_5 = 3$. Thus the required solution is $\begin{bmatrix} 1 & 1 & 7 & 2 & 3 \end{bmatrix}^t$. Since the solutions $\begin{bmatrix} 2 & 1 & 10 & 3 & 4 \end{bmatrix}^t$ and $\begin{bmatrix} 1 & 1 & 7 & 2 & 3 \end{bmatrix}^t$ are linearly independent and the nullity is 2, we do not have any other solution that will be linearly independent. For example, the solutions [15] $\begin{bmatrix} 1 & 2 & 11 & 3 & 5 \end{bmatrix}^t$ and $\begin{bmatrix} 3 & 1 & 13 & 4 & 5 \end{bmatrix}^t$ are linearly dependent on the foregoing two linearly independent solutions. Thus, these two linearly dependent solutions correspond to the chemical equations

$$CO + 2CO_2 + 11H_2 \rightarrow 3CH_4 + 5H_2O, \qquad 3CO + CO_2 + 13H_2 \rightarrow 4CH_4 + 5H_2O.$$

Only the two linearly independent solutions are the most important ones. The question that crops up is: having obtained one solution, how do we obtain a valid linearly independent solution? There is no deterministic way of answering this question. In fact, we should attempt to impose additional equations/inequations/objective functions to obtain the appropriate mathematical model that produces the unique solution compatible with the real-world situation. The real-world balancing will produce only one (not multiple) balancing under precise thermodynamical, pressure, and time conditions. Further, it may be seen that, for the foregoing example, LINDO produces the integer solution $\begin{bmatrix}1 & 1 & 7 & 2 & 3\end{bmatrix}^t$, which is different from the solution obtained from *cheminpsolver*.

Examples 10–11. The following examples [15] have been balanced using *cheminpsolver*, where k is the smallest multiplying factor, as mentioned earlier:

$$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O, \qquad k = 1$$

$$0(NH_4)_2SO_4 + 0NH_4OH \rightarrow 0SO_2 \text{ and } 0(NH_4)_2SO_4 \rightarrow 0NH_4OH + 0SO_2, \text{ infeasible.}$$

Example 12–13. *Cheminpsolver* failed for the following examples [15], while LINDO used the ILP model, where the LINDO inputs are (written in three columns to conserve space, although in actual LINDO inputs all the entries will be in a single column):

min	**2x1+3x2+2x3+2x4+2x5**	**gin x1**
x1+x2+x3+x4+x5+x6+x	**-1x6-11x7=0**	**gin x2**
7	**x1>=1**	**gin x3**
st	**x2>=1**	**gin x4**
2x1+3x2+6x3+5x4+9x5	**x3>=1**	**gin x5**
+0x6-50x7=0	**x4>=1**	**gin x6**
5x1+7x2+14x3+9x4+11	**x5>=1**	**gin x7**
x5-2x6-73x7=0	**x6>=1**	
1x1+1x2+4x3+1x4+1x5	**x7>=1**	
+0x6-15x7=0	**end**	

and produced the correct balancing, viz.,

$$C_2H_5NO_2 + C_3H_7NO_3 + 2C_6H_{14}N_4O_2 + 3C_5H_9NO_2 + 2C_9H_{11}NO_2 \rightarrow 8H_2O + C_{50}H_{73}N_{15}O_{11}.$$

Similarly, for the next example below, LINDO produced the correct balancing:

$$2C_5H_9NO_2 + 2C_5H_{11}NO_2 + 2C_6H_{13}NO_2 + 2C_9H_{11}NO_2 + 2C_5H_{12}N_2O_2 \rightarrow C_{60}H_{92}N_{12}O_{10} + 10H_2O.$$

Some important chemical aspects and nomenclature are provided by Blakley [15] in connection with examples 2, and 9–13. These may be informative and of use for the reader.

Examples 14–39. Below, we have included 26 examples chosen arbitrarily from "Problems and Questions" of Chapter 7 of the book by Dickson [11] out of several unbalanced chemical equations to conserve space. Although all these examples are small, these form an excellent base to readily see/check not only the solution on any PC having Matlab software but also to observe linearly dependent rows of the reaction matrix and its effect (if any) on the solution. Further, all the following 26 examples occur frequently in an undergraduate/graduate chemistry text-book. The concerned students may avoid memorizing the balancing or determining it by trial-and-error/inspection just by using a programmable calculator with the requisite software and the program '*cheminpsolver*' and that for solving the ILP. We have experimented extensively on many chemical equations and have found that $O(n^3)$ *cheminpsolver* could balance almost all of them, implying that our first choice is *cheminpsolver* and the next choice (if necessary) is any ILP solver. k is the smallest multiplying factor to produce an integer solution (coefficients) from the original fractional

solution, obviating the need to use non-standard arithmetic, as discussed in the introduction.

$2Mg + O_2 \rightarrow 2MgO, \quad k = 1$
$4Fe + 3O_2 \rightarrow 2Fe_2O_3, \quad k = 2$
$2KClO_3 \rightarrow 2KCl + 3O_2, \quad k = 2$
$4Na + O_2 \rightarrow 2Na_2O, \quad k = 1$
$2Sb + 3Cl_2 \rightarrow 2SbCl_3, \quad k = 2$
$C_2H_4 + 3O_2 \rightarrow 2CO_2 + 2H_2O, \quad k = 1$
$2KNO_3 + 4C \rightarrow K_2CO_3 + 3CO + N_2, \quad k = 1$
$3Mg + N_2 \rightarrow Mg_3N_2, \quad k = 1$

$2HgO \rightarrow 2Hg + O_2, \quad k = 1$
$3Fe + 4H_2O \rightarrow Fe_3O_4 + 4H_2, \quad k = 1$
$2P + 3S \rightarrow P_2S_3, \quad k = 1$
$Pb_3O_4 \rightarrow 3Pb + 2O_2, \quad k = 1$
$2P + 3H_2 \rightarrow 2PH_3, \quad k = 2$
$4NH_3 + 5O_2 \rightarrow 4NO + 6H_2O, \quad k = 4.$

$4Al + 3O_2 \rightarrow 2Al_2O_3, \quad k = 2$
$N_2 + 3H_2 \rightarrow 2NH_3, \quad k = 1$
$3C + CaO \rightarrow CaC_2 + CO, \quad k = 1$
$2C_3H_6 + 9O_2 \rightarrow 6CO_2 + 6H_2O, \quad k = 2$
$6Ca + 2NH_3 \rightarrow 3CaH_2 + Ca_3N_2, k = 1$
$4P + 5O_2 \rightarrow P_4O_{10}, \quad k = 1$
$2Ca + O_2 \rightarrow 2CaO, \quad k = 1$

$2ZnS + 3O_2 \rightarrow 2ZnO + 2SO_2, \quad k = 2$
$2CaCO_3 + 5C \rightarrow 2CaC_2 + 3CO_2, \quad k = 2$
$CaCN_2 + 3H_2O \rightarrow CaCO_3 + 2NH_3, \quad k = 1$
$FeCrO_4 + 4C \rightarrow Cr + Fe + 4CO, \quad k = 1$
$3BaO_2 + 2Al \rightarrow 3Ba + Al_2O_3, \quad k = 1$

5. Conclusions

A priori knowledge of basic variables in ILP/INP and existence always of non-degenerate integer solutions. It is assumed that *we know all the reactants and all the products in a feasible chemical reaction*. Hence we have both the ILP and the INP non-degenerate. The non-degeneracy here implies that the basic variables are always known a priori with positive values, unlike in the case of a general ILP or an LP (linear program). For a general ILP, it is impossible to determine the integer solution from the solution of the corresponding LP. While an LP can be solved in polynomial time [1], a general ILP cannot be. Further, having known the basic variables of the corresponding LP, it is also impossible to know the basic variables of a general ILP, since these variables may differ significantly. Also, there will always exist a non-degenerate integer solution (in a feasible reaction) with each element whose minimum value cannot be less than 1.

ILP solver, though exponential, may be used. The computational complexity of any method for a general ILP is exponential. It is not possible to predict the upper limit of the number of iterations in some method, say the cutting plane method (Gomory method of integer forms), for a given ILP irrespective of whether it is small or large when the real number model is used. For some other method, say the branch and bound technique, it is possible to predict the upper bound of the number of iterations, but then this number grows exponentially. However, we expect the time complexity for most real-world chemical equations to be reasonably small on a personal computer, specifically due to the very nature of the chemical equation. In fact, almost all chemical equations known to us so far do not produce a large linear homogeneous system. Hence, the cutting plane method in a finite precision environment (i.e., in a digital computer) or a branch and bound approach, though exponential, may be used in practice to achieve a balancing.

CHEMINPSOLVER here is polynomial and can be used for lots of balancing. If the orthogonal projection operator[3] $P = I - A^+A \geq 0$ (*null square matrix*), i.e., each element of P is non-negative, then the INP algorithm which is $O(n^3)$, where n = the number of reactants and products, can be used to obtain a feasible solution for balancing; this balancing, unlike an ILP-solving or an HNF-computing using the rational arithmetic or computing an exact generalized matrix inverse using the residue arithmetic, is achieved just using the widely used floating-point arithmetic. Neither the rational arithmetic nor the residue arithmetic are desired. This is because the rational arithmetic suffers from intermediate number growth and is not built-in like the floating-point arithmetic in standard digital computers, while the multiple modulus residue arithmetic in some prime base may fail as a result of a *division by zero* and is also not built-in. Further, both the arithmetics, specifically the residue arithmetic, need considerable programming effort, which is discouraging, in general. We have considered almost all the typical examples cited by

[3] Let A be a singular square or a non-square rectangular matrix which is of concern to us in chemical equation balancing. The orthogonal projection operator $P = I - A^+A$ is unique since A^+ is unique, unlike the Hermite canonical form [14,19] $H = I - A^-A$ used by Krishnamurthy [9], because the generalized inverse A^- is non-unique (unless the matrix A is non-singular).

Krishnamurthy [9] as well as Blakley [15], in addition to a number of problems taken from Dickson [11]. Except for just a couple of examples, as discussed in Section 4, all the examples worked perfectly in *cheminpsolver*. Those for which *cheminpsolver* failed have been solved by the ILP formulation using LINDO, which has used the branch and bound technique. Since a very large number of chemical equations in the real world are small for current personal computers, a general ILP solver, say the branch and bound method, though exponential, is tractable and can be used in practice effectively to achieve balancing.

Use of INP/ILP model in economic equilibrium. Both the INP and the ILP models presented here can be used to study the economic equilibrium that has been studied using the closed input–output Liontief static model [18].

Non-unique reaction and difficulty in mathematical modeling. When the mathematical model based on the law of conservation of atoms produces non-unique balancing, one could use thermodynamical criteria to select one of the valid reactions [29]. This then would correspond to *constrained optimization and/or inclusion of additional equations/inequalities*. Often this could result in a general mixed integer nonlinear programming problem, provided that the complex mathematical model, if necessary, takes into account thermodynamical, pressure, and other environmental parameters and truly represents the physical problem. In unstable/sensitive reactions, which are usually time-dependent, devising the appropriate mathematical model that is reasonably solvable is difficult and could be impossible.

Singularity of the reaction matrix A and effect of slight change in A. As the elements of the generalized inverse of a matrix are not necessarily continuous functions of the elements of the original matrix, the requirement that A is singular (square/non-square rectangular) for a non-trivial chemical reaction to be valid is consistent with the known fact that even small changes in A must be accompanied by abnormally large and discontinuous changes in the chemical reaction.

Role of linear/nonlinear objective functions. There are two reasons for choosing the nonlinear objective function "Minimize $h = y^t y = \sum_{i=1}^{n} y_i^2$" in INP formulation.

(i) We already have several polynomial-time algorithms (mentioned earlier) specifically to solve the system with this objective function. This objective function implies minimum norm (i.e., $\|y\| = (y^t y)^{1/2}$ is minimum) solution of the consistent linear system. The least-squares aspect of the solution is non-existent, since the system is not inconsistent. However, due to human/modeling error, a small inconsistency may be introduced; this near-consistency/inconsistency is easily detected by *cheminpsolver*. Such a detection facility is useful, as the foregoing error cannot be ruled out. Interestingly, if we replace the nonlinear objective function by the linear objective function "Minimize $\sum_{i=1}^{n} y_i$", include non-negativity and integrality conditions, and retain the same linear system, then the problem becomes NP-complete and is not amenable to any polynomial algorithm known so far.

(ii) The outcome of any natural process that always obeys all the laws of nature perfectly, i.e., which never breaks any law of nature (known or unknown to us) is the result of minimization. No roundabout or unnecessary steps/activities can be a part of any natural process that is perfectly optimized (minimized) [4]. While we do not know the exact nature of the objective function, we have found that a simple linear objective function, or even a simple nonlinear objective function as used in ILP and INP, respectively, produces the required balancing.

Integer solution using Matlab ceil and floor operations with a small number. To obtain the integer solution out of the (15 digit) floating-point solution in Matlab, we have used $ceil(x - 10^{-10})$, $floor(x - 10^{-10})$ operations. The subtraction of a small number 10^{-10} or even 10^{-5} (compared to the value of x, which is of the order of 1 here) is necessary to get the desired ceil and floor values. The number 10^{-10} is, however, added to the computed x to obtain the integer value. If a small number is not used, then we could end up with no, or wrong, integer solutions. This is an important programming technique needed for the required integer solution.

Chemical equation has no information about type and rate of reaction and state of reactants. A chemical reaction involves the breaking and forming of bonds to form new arrangements of combined atoms. Chemical bonds involve energy. The *activation energy* of a reaction is the minimum amount of potential energy that colliding molecules must have to successfully give products. The reactants have a total chemical energy different to the products. Some chemical reactions are called *exothermic reactions*, such as the reaction of sodium and chlorine, which release 411 kJ of energy as heat for each mole of common salt $2Na(s) + Cl_2(g) \rightarrow NaCl(s)$. The symbols s and g denote solids and gases, respectively. Others, on the other hand, are called *endothermic reactions*, such as photosynthesis, which absorbs 15 MJ of heat energy from the surroundings for each kg of glucose ($C_6H_{12}O_6$):

$$\text{sunlight} + 6CO_2(g) + H_2O(l) \rightarrow C_6H_{12}O_6(aq) + 6O_2(g).$$

or, equivalently,

$$6CO_2(g) + H_2O(l) \xrightarrow{\text{heat}} C_6H_{12}O_6(aq) + 6O_2(g).$$

The symbols l and aq denote liquids and aqueous solution, respectively. The speed at which a reaction takes place can be expressed as the *rate of the reaction.* An increase in the temperature and/or the use of an appropriate catalyst may enhance the rate of reaction. The hydrogen gas (H_2) and the nitrogen gas (N_2) can react to form ammonia (NH_3). When these gases are mixed at room temperature (20 °C), the rate of reaction is so slow that no detectable reaction occurs. Even with a catalyst, say iron (Fe), the reaction at room temperature is very slow. The reaction system has to be heated to 425 °C (also using Fe as a catalyst) to increase the rate to obtain significant amounts of ammonia in a reasonable time:

$$N_2(g) + 3H_2(g) \xrightarrow{425\,^\circ\text{C, Fe}} 2NH_3(g).$$

A catalyst (as opposed to an inhibitor) is a chemical that increases the rate of a chemical reaction without being chemically changed. For example, hydrogen peroxide (H_2O_2) in an aqueous solution will decompose very slowly at room temperature. In the presence of the catalyst manganese oxide (MnO_2), the (H_2O_2) decomposes rapidly at room temperature:

$$2H_2O_2(aq) \xrightarrow{MnO_2} 2H_2O(l) + O_2(g).$$

For gaseous reactants, concentration can be increased by increasing the pressure. For example, ethylene (C_2H_4) and steam (H_2O) react to produce ethyl alcohol (C_2H_5OH). To raise the rate of reaction to an industrially productive level, the reaction is carried out in a closed system at 400 °C and at a very high pressure. It is thus necessary to specify the environmental conditions of the reaction explicitly:

$$C_2H_4(g) + H_2O(g) \xrightarrow{400\,^\circ\text{C, pressure}} C_2H_5OH(g).$$

While just the chemical equation reflects the law of conservation of atoms, it does not provide any information whether (i) the reaction concerned was endothermic or exothermic, (ii) the state of reactants and products concerned were gaseous, liquid, and/or solid, and (iii) the speed of reaction concerned, which depends on temperature, pressure, and/or catalyst(s), was slow or fast. Nevertheless, balancing a chemical equation is important, in that it describes the chemical reaction in a more succinct and useful way.

References

[1] N. Karmarkar, A new polynomial-time algorithm in linear programming, Combinatorics 4 (1984) 373–395.
[2] K.G. Murty, S.Y. Chang, The steepest descent gravitational method for linear programming, Discrete Appl. Math. 25 (1989) 211–239.
[3] G.B. Dantzig, Linear Programming and Extension, Princeton University Press, Princeton, 1963.
[4] V. Lakshmikantham, S.K. Sen, Computational Error and Complexity in Science and Engineering, Elsevier, Amsterdam, 2005.
[5] R. Courant, H. Robbins, What is Mathematics? An Elementary Approach to Ideas and Methods, 2nd ed., Oxford University Press, Oxford, England, 1996, pp. 81–83.
[6] P. Hoffman, The Man Who Loved Only Numbers: The Story of Paul Erdos and the Search for Mathematical Truth, Hyperion, New York, 1998, pp. 220–223.
[7] R. Penrose, The Emperor's New Mind: Concerning Computers, Minds and the Laws of Physics, Oxford University Press, Oxford, England, 1989, pp. 84–85.
[8] M. Agarwal, N. Kayal, N. Saxena, PRIMES in P, Indian Institute of Technology, Kanpur, 2002. Unpublished manuscript.
[9] E.V. Krishnamurthy, Generalized matrix inverse approach to automatic balancing of chemical equation, Int. J. Math. Educ. Sci. Technol. 9 (1978) 323–328.
[10] D.H. Andrews, R.J. Kokes, Fundamental Chemistry, Wiley, New York, 1963.
[11] T.R. Dickson, Introduction to Chemistry, 5th ed., Wiley, New York, 1987 (Chapter 7).
[12] T.M. Rao, K. Subramanian, E.V. Krishnamurthy, Residue arithmetic algorithms for exact computation of generalized inverse of matrix, SIAM J. Numer. Anal. 13 (1976) 155–171.
[13] J.T. Moore, Elements of Linear Algebra and Matrix Theory, McGraw-Hill, New York, 1968.
[14] E.V. Krishnamurthy, S.K. Sen, Numerical Algorithms: Computations in Science and Engineering, Affiliated East-West Press, New Delhi, 2001.
[15] G.R. Blakley, Chemical equation balancing, J. Chem. Educ. 59 (1982) 728–734.

[16] D.D. Kumar, Computer applications in balancing chemical equations, J. Sci. Educ. Technol. 10 (2001) 347–350.
[17] S.K. Sen, S. Sen, $O(n^3)$ g-inversion-free noniterative near-consistent linear system solver for minimum-norm least-squares and nonnegative solutions, J. Comput. Methods Sci. Eng. (2005) (in press).
[18] R.E. Kuenne, The Theory of General Economic Equilibrium, Princeton University Press, Princeton, 1963.
[19] C.R. Rao, S.K. Mitra, Generalized Inverse of Matrices and Its Application, Wiley, New York, 1971.
[20] G. Golub, W. Kahan, Calculating the singular values and the pseudo-inverse of a matrix, SIAM J. Numer. Anal. B-2 (1965) 205–224.
[21] V. Lakshmikantham, S.K. Sen, G.W. Howell, Vectors versus matrices: p-inversion, cryptographic applications, and vector implementation, Neural Parallel Sci. Comput. 4 (1996) 129–140.
[22] R. Penrose, A generalized inverse for matrices, Proc. Chemb. Phil. Soc. 51 (1955) 406–413.
[23] E.H. Moore, On the reciprocal of the general algebraic matrix (abs.), Bull. Amer. Math. Soc. 26 (1920) 394–395.
[24] E.A. Lord, V.Ch. Venkaiah, S.K. Sen, A concise algorithm to solve under-/over-determined linear systems, Simulation 54 (1990) 239–240.
[25] E.A. Lord, V.Ch. Venkaiah, S.K. Sen, A shrinking polytope method for linear programming, Neural Parallel Sci. Comput. 4 (1996) 325–340.
[26] S.K. Sen, E.V. Krishnamurthy, Rank-augmented LU-algorithm for computing generalized matrix inverses, IEEE Trans. Comput. C-23 (1974) 199–201.
[27] S.K. Sen, S.S. Prabhu, Optimal iterative schemes for computing Moore–Penrose matrix inverse, Internat. J. Systems Sci. 8 (1976) 748–753.
[28] W.L. Winston, Operations Research: Applications and Algorithms, 4th ed., Thomson, Belmont, CA, 2004.
[29] V.F. Zeggeren, S.H. Storey, The Computation of Chemical Equilibria, Cambridge University Press, London, 1970.