Should Pruning be a Pre-processor of any Linear System?

Syamal K. Sen, Suja Ramakrishnan, Ravi P. Agarwal[1]
Department of Mathematical Sciences, Florida Institute of Technology, 150 West University Boulevard, Melbourne, FL 32901-6975, United States

and

Gholam Ali Shaykhian
National Aeronautics and Space Administration (NASA), Technical Integration Office (IT-G), Information Technology (IT) Directorate, Kennedy Space Center, FL 32899, United States

Abstract There are many real-world problems whose mathematical models turn out to be linear systems $Ax = b$, where A is an $m \times n$ matrix. Each equation of the linear system is an information. An information, in a physical problem, such as 4 mangoes, 6 bananas, and 5 oranges cost \$10, is mathematically modeled as an equation $4x_1 + 6x_2 + 5x_3 = 10$, where x_1, x_2, x_3 are each cost of one mango, that of one banana, and that of one orange, respectively. All the information put together in a specified context, constitutes the physical problem and need not be all distinct. Some of these could be redundant, which cannot be readily identified by inspection. The resulting mathematical model will thus have equations corresponding to this redundant information and hence are linearly dependent and thus superfluous. Consequently, these equations once identified should be better pruned in the process of solving the system. The benefits are (i) less computation and hence less error and consequently a better quality of solution and (ii) reduced storage requirements. In literature, the pruning concept is not in vogue so far although it is most desirable. It is assumed that at least one information, i.e. one equation is known to be correct and which will be our first equation.

In a numerical linear system, the system could be slightly inconsistent or inconsistent of varying degree. If the system is too inconsistent, then we should fall back on to the physical problem (PP), check the correctness of the PP derived from the material universe, modify it, if necessary, and then check the corresponding mathematical model (MM) and correct it. In nature/material universe, inconsistency is completely non-existent. If the MM becomes inconsistent, it could be due to error introduced by the concerned measuring device and/or due to assumptions made on the PP to obtain an MM which is relatively easily solvable or simply due to human error. No measuring device can usually measure a quantity with an accuracy greater that 0.005% or, equivalently with a relative error less than 0.005%. Hence measurement error is unavoidable in a numerical linear system when the quantities are continuous (or even discrete with

[1] Corresponding author
E-mail addresses: sksen@fit.edu (S.K. Sen), sramakrishna2009@my.fit.edu (S. Ramakrishnan), agarwal@fit.edu (R. P. Agarwal), ali.shaykhian@nasa.gov (G.A. Shaykhian)

extremely large number). Assumptions, though not desirable, are usually made when we find the problem sufficiently difficult to be solved within the available means/tools/resources and hence distort the PP and the corresponding MM. The error thus introduced in the system could (not always necessarily though) make the system somewhat inconsistent. If the inconsistency (contradiction) is too much then one should definitely not proceed to solve the system in terms of getting a least-squares solution or the minimum-norm least-squares solution. All these solutions will be invariably of no real-world use. If, on the other hand, inconsistency is reasonably low, i.e. the system is near-consistent or, equivalently, has near-linearly-dependent rows, then the foregoing solutions are useful. Pruning in such a near-consistent system should be performed based on the desired accuracy and on the definition of near-linear dependence. In this article, we discuss pruning over various kinds of linear systems and strongly suggest its use as a pre-processor or as a part of an algorithm. Ideally pruning should (i) be a part of the solution process (algorithm) of the system, (ii) reduce both computational error and complexity of the process, and (iii) take into account the *numerical zero* defined in the context. These are precisely what we achieve through our proposed $O(mn^2)$ algorithm presented in Matlab, that uses a subprogram of solving a single linear equation and that has embedded in it the pruning.

Keywords: Inconsistency; Linearly dependent rows; Linear system; Linsolver; Pruning-based Matlab implementation

1. Introduction

To err is human while *Not to err is computer* could be two proverbs – the first one was existing from time immemorial and will continue to exist eternally. The second one is of recent origin and is understood with respect to a modern digital computer of late twentieth century and later. Here "err" implies mistake. Any living being – a human being including a superman/woman as well as an animal or even an insect – are prone to commit mistake while a computer does not. There exists no living being on earth or possibly in the universe, who can say that he does not commit a mistake! Under these circumstances there is no need to have redundant information to remain in a computing system so that an error (mistake) can be detected by this redundant information. Redundancy in some physical systems (and consequently in their corresponding mathematical models) could be helpful in detecting errors and correcting them [1-3]. In the case of linear systems, redundant information is not only of not any use but also does contribute to unnecessary computations and consequent additional error injection in the solution using a computer [also refer 4-9].

A linear equation in a system is a mathematical model of an information. For instance, the mathematical model of the information "3 mangoes, 4 oranges, and 5 bananas cost \$ 6" could be written as $3x_1 + 4x_2 + 5x_3 = 6$, where x_i, $i = 1,2,3$ are the unit cost (in dollar) of one mango, one orange, and one banana, respectively. Similarly, the mathematical models of the information "2 mangoes, 2 orange, and 4 bananas cost \$ 4" and "5 mangos, 6 oranges, and 9 bananas cost \$ 10" are $2x_1 + 2x_2 + 4x_3 = 4$ and $5x_1 + 6x_2 + 9x_3 = 10$, respectively. Not all these three information are distinct. It is not, in general, possible to detect by inspection or by any simple trick whether the given information are distinct or not. An information may be generated/derived from other specified information in a system of information or, equivalently, in a system of linear equations. The derived information will not be considered distinct since it does not add to the system any new information. Such a derived information or, equivalently, a linear equation is redundant and is called linearly dependent on the other equation(s). This linearly dependent equation is of no use so far as the/a solution of the system is concerned and hence should be pruned/weed out. Such a pruning is not done before or in the process of solving a linear system and consequently more storage locations more computations, and more errors result. However, there are many linear systems occurring in a mathematical model, which could have all the linear equations linearly independent. For example, a partial differential equation or a system of partial differential equations – a mathematical model of a physical (real-world) problem, when approximated by a finite-difference scheme for a numerical solution, could produce a tri-diagonal system of linear equations, all of which are linearly independent [10, 11]. In such a situation, the question of pruning does not, in general, arise. Even if the system is subjected to pruning, no pruning will take place. On the other hand, there are plenty of real-world mathematical models, each of which is a linear system consisting of redundant (linearly dependent) equations. In the area of operations research, specifically, in the problems of least-squares curve fitting and in linear programs consisting of a system of linear equality[2] constraints (linear equations) we might encounter many linearly dependent (redundant) equations in the concerned system. We will consider here numerically linearly dependent equations rather than mathematically linearly dependent equations. Mathematically linearly dependent equations will always be invariably numerically linearly dependent while the converse is not true. These are connected to mathematical zero and numerical zero. While the mathematical zero is the absolute zero, a numerical zero is not. A numerical zero is any quantity which is less than the minimum permitted relative error (in magnitude). For instance, if the quantity of higher order accuracy is Q and that of lower order accuracy is Q', then the relative error in the quantity Q' is $|Q-Q'|/|Q|$. If the minimum permitted relative error in Q' is 0.5×10^{-4}, then any value less than $0.5\times10^{-4}|Q|$ in magnitude will be considered a numerical zero. We will be concerned here with the pruning of the numerically linearly dependent equations from the system as these do not add numerically any new information to the system. Such a pruning will always remove mathematically linearly dependent equations from the system since mathematically linearly dependent equations constitute a proper subset of numerically linearly dependent equations. In fact, before solving a linear system, i.e., computing a numerical solution of the consistent system or obtaining a least-squares solution (may not be unique) of the inconsistent system or determining the minimum-norm least-squares solution (unique) of the consistent/inconsistent system, we could use pruning as a pre-processor or we could

[2] So far as the linear inequality constraints are concerned, we usually add (subtract) slack (surplus) variables to convert inequalities to equations and then deal with the system of equations rather than inequalities directly. This is mainly because we have a rich information base for the theory of equations unlike that for the theory of inequalities [12]. The resulting equations here will be all linearly independent and hence the question of pruning does not arise.

use pruning as an integral part of the solution process. In both the cases, the computational complexity is $O(n^3)$.

In section 2, we provide a physically concise Linsolver algorithm with pruning as its integral part of the solution process along with a Matlab – a highest level programming language – code. This algorithm produces in addition to the solution vector x of the given linear system $Ax = b$, the rank of the $m \times n$ matrix A, the orthogonal projection operator $P = I - A^+ A$, where A^+ is the Moore-Penrose inverse of the matrix A. It also detects numerical/mathematical inconsistency. The Matlab program can be easily followed just like the algorithm written in a pseudo-code (in English). So we have omitted presenting the pseudo-code version of the algorithm. This Matlab code (actual program) has additional advantage that allows the reader to use the code readily by simply copying, pasting, and execute the code/program with specified inputs A, b. All that is required for the reader to have the Matlab software implemented on his laptop/desktop. Section 3 consists of numerical examples while section 4 comprises conclusions.

2. Linsolver and Its Pruning-based Matlab Implementation

We consider the linear system $Ax = b$, where A is an $m \times n$ matrix and b is an $m \times 1$ vector. Let I be an $n \times n$ identity (unit) matrix. We provide below an algorithm called Linsolver [12] which is used as the starting point for a Matlab implementation of the algorithm with pruning as its in-built feature.

Linsolver Let a_i^t be the $i-th$ row of A. Then a_i is the column vector. The following algorithm – a modified version of the algorithms in Lord et al. [15, 16] – finds a particular solution x. It obtains as a by-product the orthogonal projection operator $P = I - A^+ A$ [17-23], where A^+ is the minimum-norm least-squares inverse of A and the rank r of A.

The Linsolver

```
$P = I$; $x = 0$; $r = 0$; for $i = 1$ to $m$ do begin eqn($a_i, b_i, P, x$); $r = r + c$ end
procedure eqn($a, b, P, x$); (*obtains a solution of the equation $a^t x = b$ *)
begin $c = 0$; $u = Pa$; $v = \|u\|^2$; inconsistency $= b - a^t x$; if $\neq 0$ then
   begin $P = P - uu^t / v$; $x = x +$ inconsistency $\times u / v$; $c = 1$ end else
   if inconsistency $\neq 0$ then $Ax = b$ is inconsistent and exit
end; (*The foregoing $\times$ is simply multiplication of vector $u$ by a scalar.*)
```

Complexity of Linsolver The computational complexity of the procedure *eqn* is $O(n^2)$. Since the number of applications of *eqn* 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 k significant digit accuracy, before it is implemented on a digital computer. Let $\leftarrow$ denote *'is to be replaced by'.*

$$v \neq 0 \leftarrow v \geq 0.5\times10^{-k}\bar{a} \quad and$$

$$inconsistency \neq 0 \leftarrow inconsistency \geq 0.5\times10^{-k}\bar{b}$$

where $\bar{a} = (\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|)/(mn)$, $\bar{b} = \sum_{i=1}^{m}|b_i|/m$. If we desire k decimal digit accuracy then $\bar{a}$ and $\bar{b}$ should be removed or, equivalently, $\bar{a} = \bar{b} = 1$ should be taken. It can be seen that $\log_{10}$(1/relative error) gives the number of significant digits up to which the result/quantity is correct while $\log_{10}$(1/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.

The following Matlab program which is self-explanatory (i) computes the orthogonal projection operator $P = I - A^{+}A$ of the matrix A, where A^{+} is the Moore-Penrose inverse of the rectangular matrix A, (ii) produces a solution to the system if the system is consistent, (iii) prunes the redundant (linearly dependent) rows of the system if the system is consistent, depicts the rank of the matrix A, (iv) displays the pruned system, where the pruned matrix is evidently full-row rank and the system is consistent, (v) provides a solution to the system, and (vi) also produces a nontrivial solution of the system if the system is homogeneous, i.e., if $Ax = 0$ (null column vector).

```
function pruningbasedlinsolver2(A,b);
B=A; c=b; rrow=0;
[m n]=size(A); p=0;
x=zeros(n,1); r=0; P=eye(n); k=1;
abar=sum(sum(abs(A)))/(m*n); bbar=sum(abs(b))/m;
for i = 1:m
    a=A(i,:)'; brow=b(i); u=P*a; v=(norm(u))^2; inconsistency = brow-
a'*x;
    if abs(v) >= 0.00005*abar   %Permits 4 significant digit accuracy
        P=P-u*u'/v; x = x + inconsistency*u/v; r=r+1;
    else
            if abs(inconsistency) > 0.00005*bbar
            disp('Linear system Ax=b is inconsistent.'); p=1;
            break;
        else
            % Store indices of redundant rows in a vector.
            redrow(k)=i; k=k+1; rrow=1;
        end
    end
end
disp('The orthogonal projection operator P=I-A^+A is'); P
%If the system is homogeneous, i.e. if Ax=0 (always consistent), then
% a solution is x=Pz, where z is an arbitrary column vector.
if bbar==0
    z=rand(n,1);disp('A nontrivial solution to Ax=0 is');P*z
end
% Prune the redundant rows of the augmented matrix (A, b) of Ax=b.
```

```
if rrow==1
c(redrow)=[]; B(redrow,:)=[];
end
% Display the results.
if p ~= 1
    S=size(B);
    if S(1)<=m
        disp('A solution to the system is '); disp(x);
        disp('The rank of the matrix A is '); disp(r);
        disp('The pruned system Bx=c has B and c as');B, c
    else
        disp('The solution to the system is '); disp(x);
    end
end
```

3. Numerical Examples

We illustrate the foregoing algorithm in section 2 by considering the following numerical examples.

Example 1 (A 5×4 consistent matrix equation)

```
>> A=[1 2 3 4;2 4 6 8; 3 6 9 12;1 1 1 1; 1 -2 1 -3], b=[10
20 30 4 -3]',pruningbasedlinsolver2(A,b)

A =

     1     2     3     4
     2     4     6     8
     3     6     9    12
     1     1     1     1
     1    -2     1    -3


b =

    10
    20
    30
     4
    -3

The orthogonal projection operator P=I-A^+A is

P =

    0.2816   -0.3218   -0.2011    0.2414
   -0.3218    0.3678    0.2299   -0.2759
   -0.2011    0.2299    0.1437   -0.1724
    0.2414   -0.2759   -0.1724    0.2069

A solution to the system is
```

```
    1.0000
    1.0000
    1.0000
    1.0000

The rank of the matrix A is
     3

The pruned system Bx=c has B and c as

B =
     1     2     3     4
     1     1     1     1
     1    -2     1    -3


c =
    10
     4
    -3
```

Example 2 (A 5×4 inconsistent matrix equation)

```
>> A=[1 2 3 4;2 4 6 8; 3 6 9 12;1 1 1 1; 1 -1 1 -1], b=[10 20 20 4
0]',pruningbasedlinsolver2(A,b)

A =
     1     2     3     4
     2     4     6     8
     3     6     9    12
     1     1     1     1
     1    -1     1    -1

b =
  10
  20
  20
   4
   0

Linear system Ax=b is inconsistent (no solution).

The orthogonal projection operator P=I-A^+A is

P =
   0.9667  -0.0667  -0.1000  -0.1333
  -0.0667   0.8667  -0.2000  -0.2667
```

```
   -0.1000  -0.2000   0.7000  -0.4000
   -0.1333  -0.2667  -0.4000   0.4667
```

Example 3 (An 8×5 random linear system (overdetermined))

>> A=rand(8,5), b=sum(A')',pruningbasedlinsolver2(A,b)

```
A =
    0.4018    0.9027    0.7803    0.5752    0.6491
    0.0760    0.9448    0.3897    0.0598    0.7317
    0.2399    0.4909    0.2417    0.2348    0.6477
    0.1233    0.4893    0.4039    0.3532    0.4509
    0.1839    0.3377    0.0965    0.8212    0.5470
    0.2400    0.9001    0.1320    0.0154    0.2963
    0.4173    0.3692    0.9421    0.0430    0.7447
    0.0497    0.1112    0.9561    0.1690    0.1890

b =
    3.3091
    2.2020
    1.8550
    1.8206
    1.9863
    1.5837
    2.5163
    1.4749
```

The orthogonal projection operator P=I-A^+A is

```
P =
  1.0e-014 *
    0.0912   -0.0163    0.1846   -0.0090   -0.0291
   -0.0163   -0.0999    0.1277    0.1166    0.0083
    0.1846    0.1277   -0.1110   -0.1499   -0.0971
   -0.0090    0.1166   -0.1499    0.1277   -0.0916
   -0.0291    0.0083   -0.0971   -0.0916    0.0791
```

A solution to the system is

```
    1.0000
    1.0000
    1.0000
    1.0000
    1.0000
```

The rank of the matrix A is   5

The pruned system Bx=c has B and c as

```
B =
   0.4018   0.9027   0.7803   0.5752   0.6491
   0.0760   0.9448   0.3897   0.0598   0.7317
   0.2399   0.4909   0.2417   0.2348   0.6477
   0.1233   0.4893   0.4039   0.3532   0.4509
   0.1839   0.3377   0.0965   0.8212   0.5470

c =
   3.3091
   2.2020
   1.8550
   1.8206
   1.9863
```

Example 4 (Another 8×5 over-determined random system)

```
>> A=rand(8,5), b=sum(A')',pruningbasedlinsolver2(A,b)

A =
   0.4018   0.9027   0.7803   0.5752   0.6491
   0.0760   0.9448   0.3897   0.0598   0.7317
   0.2399   0.4909   0.2417   0.2348   0.6477
   0.1233   0.4893   0.4039   0.3532   0.4509
   0.1839   0.3377   0.0965   0.8212   0.5470
   0.2400   0.9001   0.1320   0.0154   0.2963
   0.4173   0.3692   0.9421   0.0430   0.7447
   0.0497   0.1112   0.9561   0.1690   0.1890

b =
   3.3091
   2.2020
   1.8550
   1.8206
   1.9863
   1.5837
   2.5163
   1.4749

The orthogonal projection operator P=I-A^+A is

P =
  1.0e-014 *
   0.0912  -0.0163   0.1846  -0.0090  -0.0291
```

```
 -0.0163  -0.0999   0.1277   0.1166   0.0083
  0.1846   0.1277  -0.1110  -0.1499  -0.0971
 -0.0090   0.1166  -0.1499   0.1277  -0.0916
 -0.0291   0.0083  -0.0971  -0.0916   0.0791
```

A solution to the system is

```
  1.0000
  1.0000
  1.0000
  1.0000
  1.0000
```

The rank of the matrix A is 5

The pruned system Bx=c has B and c as

```
B =
  0.4018   0.9027   0.7803   0.5752   0.6491
  0.0760   0.9448   0.3897   0.0598   0.7317
  0.2399   0.4909   0.2417   0.2348   0.6477
  0.1233   0.4893   0.4039   0.3532   0.4509
  0.1839   0.3377   0.0965   0.8212   0.5470

c =
  3.3091
  2.2020
  1.8550
  1.8206
  1.9863
```

Example 5 (A 5×8 under-determined random system)

>> A=rand(5,8), b=sum(A')',pruningbasedlinsolver2(A,b)

```
A =
  0.6868   0.0811   0.4468   0.7948   0.3507   0.5870   0.8443   0.4357
  0.1835   0.9294   0.3063   0.6443   0.9390   0.2077   0.1948   0.3111
  0.3685   0.7757   0.5085   0.3786   0.8759   0.3012   0.2259   0.9234
  0.6256   0.4868   0.5108   0.8116   0.5502   0.4709   0.1707   0.4302
  0.7802   0.4359   0.8176   0.5328   0.6225   0.2305   0.2277   0.1848

b =
  4.2273
  3.7162
  4.3578
```

```
    4.0568
    3.8320
```

The orthogonal projection operator P=I-A^+A is

```
P =
    0.4883    0.1331   -0.4478   -0.1349    0.0336   -0.1089    0.0187   -0.0057
    0.1331    0.4814   -0.0104   -0.1338   -0.4370    0.0617    0.1338   -0.0353
   -0.4478   -0.0104    0.4404    0.1181   -0.1531    0.0893    0.0183    0.0082
   -0.1349   -0.1338    0.1181    0.2611   -0.0251   -0.3530    0.0017    0.1327
    0.0336   -0.4370   -0.1531   -0.0251    0.5126    0.1035   -0.1472   -0.0357
   -0.1089    0.0617    0.0893   -0.3530    0.1035    0.6940   -0.0404   -0.2275
    0.0187    0.1338    0.0183    0.0017   -0.1472   -0.0404    0.0440    0.0114
   -0.0057   -0.0353    0.0082    0.1327   -0.0357   -0.2275    0.0114    0.0783
```

A solution *(minimum norm least squares or simply minimum norm here)* to the system is

```
    1.0236
    0.8065
    0.9371
    1.1332
    1.1484
    0.7813
    0.9597
    1.0736
```

The rank of the matrix A is 5

The pruned system Bx=c has B and c as

```
B =
    0.6868    0.0811    0.4468    0.7948    0.3507    0.5870    0.8443    0.4357
    0.1835    0.9294    0.3063    0.6443    0.9390    0.2077    0.1948    0.3111
    0.3685    0.7757    0.5085    0.3786    0.8759    0.3012    0.2259    0.9234
    0.6256    0.4868    0.5108    0.8116    0.5502    0.4709    0.1707    0.4302
    0.7802    0.4359    0.8176    0.5328    0.6225    0.2305    0.2277    0.1848
```

```
c =
    4.2273
    3.7162
    4.3578
    4.0568
    3.8320
```

Example 6 (A 5×5 consistent linear system with rank 2)

```
>> A=[1 2 3 4 5; 2 4 6 8 10; 3 6 9 12 15; 1 1 1 1 1; 2 2 2 2 2]; b=sum(A');
pruningbasedlinsolver2(A,b)

The orthogonal projection operator P=I-A^+A is

P =
   0.4000  -0.4000  -0.2000  -0.0000   0.2000
  -0.4000   0.7000  -0.2000  -0.1000  -0.0000
  -0.2000  -0.2000   0.8000  -0.2000  -0.2000
  -0.0000  -0.1000  -0.2000   0.7000  -0.4000
   0.2000  -0.0000  -0.2000  -0.4000   0.4000

A solution to the system is

   1.0000
   1.0000
   1.0000
   1.0000
   1.0000

The rank of the matrix A is 2

The pruned system Bx=c has B and c as

B =
   1   2   3   4   5
   1   1   1   1   1

c =
   15
    5
```

4. Conclusions

Assumption We have seen that each information is represented as an equation. It is necessary for us to be sure about the correctness of one information, i.e. one equation. Once we identify such an equation, we put it as the first equation of the linear system. In a physical problem, obtaining such a correct equation out of a large number of equations is not difficult. This will allow us to detect both numerical inconsistency (contradiction) as well as numerical redundancy (linear dependence) appropriately.

Mathematical linear dependence versus numerical linear dependence The linear dependence of some rows of the augmented matrix (A, b) of the system $Ax = b$ on other rows includes both the mathematical (exact) linear dependence as well as numerical linear dependence. The mathematical linear dependence is exact/error-free and is implicitly based on infinite precision computation. Numerical linear dependence, on the other hand, is connected to relative error and is based on a finite precision computation. The Matlab program presented here takes care of both. By appropriately changing the factor 0.5×10^{-4} in the program we will achieve (higher or lower) accuracy as required subject, however, to the specified precision of computation. Here we have 15 digit accuracy in Matlab's standard precision. For higher accuracy, the Matlab *vpa* (variable precision arithmetic) command may be used/explored.

Least squares/Minimum norm least-squares solution of the linear system The linear system $Ax=b$, under- or over-determined , consistent or not, its least-squares solution can be obtained from solving the ever-consistent system $A'x=b'$, where $A'=A^tA, b'=A^tb$. The system $A'x=b'$ should be subjected to pruning if more accuracy, less storage, and possibly less computational resources are desired. Some systems, however, can be readily/a priori recognized to be the ones where there exist no redundant rows. In such a case, pruning is not to be used since it will be unnecessary though completely harmless.

Numerical zero versus mathematical zero A mathematically redundant (linearly dependent) row differs from a numerically redundant row in that the former one is exactly redundant and carries absolutely no new information for the system and hence should always be pruned as soon as it is detected. The difference is essentially due to the difference between a numerical zero and the mathematical zero. The mathematical zero is unique while a numerical zero is not. There are infinite possible numerical zeros which always include the mathematical zero and is any value less than $0.5\times10^{-k}\,|Q|$ [13] in magnitude, where k is a positive integer and Q is the quantity of higher order accuracy. k could be taken as, say, 4 or larger depending on the relative accuracy desired. The higher the accuracy desired, the higher will be the value of the positive integer k. The proposed pruning-based Linsolver algorithm weeds out both numerically as well as mathematically redundant rows from the given system

Rank and orthogonal projection operator: What they are and what their use is The algorithm produces a solution along with rank (number of non-redundant or, equivalently linearly independent rows) of the system, and the orthogonal projection operator $P=I-A^{+}A$ often required not only for solving a linear program $Max\ c^t x\ subject\ to\ Ax=b, x\geq 0$ using an interior-point method [14] but also for obtaining a non-trivial solution $x=Pz$, where z is an arbitrary non-zero column vector, of a homogeneous linear system $Ax=0$. The rank of the system conceptually/physically is a measure of the information content of the system. The higher the rank of the system is, the more is the information content of the system. Here the information content is measured in terms of the number of linearly independent (non-redundant) rows of the augmented system (A,b), where each row of the augmented matrix, that represents a linear equation is essentially an information. An information or, equivalently, an equation that could be generated from the existing equations is always redundant and will be of no use for any purpose so far as the system is concerned.

References

1. http://en.wikipedia.org/wiki/Error_detection_and_correction
2. K. S. Andrews, D. Divasalar, S. Dolinar, J. Hamkins, C.R. Jones, and F. Pollara, *The Development of Turbo and LDPC Codes for Deep-Space Applications, Proceedings of the IEEE*, **95**, No. 11, Nov. 2007.
3. W. Huffman, V. Pless, *Fundamentals of error-correcting codes*, Cambridge University Press, ISBN 9780521782807, 2003.
4. S.K. Sen, Near-singular/ill-conditioned singular systems: Nclinsolver versus Matlab solvers, Chapter 17 of the book *Advances in Mathematical Problems in Engineering Aerospace and Sciences (ed. Dr. Seenith Sivasundaram)*, Ooh Publishing, United Kingdom, pp. 183-200, 2008. Also includes, as an appendix, the article "Dr. Lak.: The man I know of".
5. S.K. Sen, R.P. Agarwal, and G.A. Shaykhian, Ill- Versus Well-conditioned Singular Linear Systems: Scope of Randomized Algorithms, *J. Appl. Math. & Informatics*, **27**, No. 3-4, pp. 621-638. Website: http://www.kcam.biz
6. S.K. Sen and Sagar Sen, Linear systems: Relook, concise algorithms, and Matlab programs, *National Journal of Jyoti Research Academy*, **1**, 1, 1-8, 2007.
7. S.K. Sen, Open problems in computational linear algebra, *Nonlinear Analysis* **63**(2005), 926-934 (Available online at www.sciencedirect.com).
8. S.K. Sen and Sagar Sen, $O(n^3)$ g-inversion-free noniterative near-consistent linear system solver for minimum-norm least-squares and nonnegative solutions, *J. Computational Methods in Sciences and Engineering*, **6,** Nos. 1,4, pp. 71-85, 2006.
9. V. Lakshmikantham, S.K. Sen, and S. Sivasundaram, Concise row-pruning algorithm to invert a matrix, *Applied Mathematics and Computation*, **60,** 1994, 17-24.
10. E.V. Krishnamurthy and S.K. Sen, *Numerical Algorithms: Computations in Science and Engineering*, Affiliated East-West Press, New Delhi, 2007.
11. G. D. Smith, *Numerical Solution of Partial Differential Equations,* Oxford University Press, Oxford, 1965.
12. S.K. Sen, H. Agarwal, and S. Sen, Chemical equation balancing: An integer programming approach, *Mathematical and Computer Modelling*, **44**, 2006, 678-691.
13. V. Lakshmikantham and S.K. Sen, *Computational Error and Complexity in Science and Engineering*, Elsevier, Amsterdam, 2005.
14. S.K. Sen, S. Ramakrishnan, and R.P. Agarwal, Solving Linear Program as Linear System in Polynomial-time, *Mathematical and Computer Modelling*, **53**, 2011, 1056-1073.
15. E.A. Lord, V. Ch. Venkaiah and S.K. Sen, A concise algorithm to solve under-/over-determined linear systems, *SIMULATION*, **54**, 239-240, (1990).
16. E.A. Lord, V. Ch. Venkaiah and S.K. Sen, A shrinking polytope method for linear programming, *Neural, Parallel & Scientific Computations*, **4**, 325-340, (1996).
17. E.V. Krishnamurthy and S.K. Sen, *Numerical Algorithms: Computations in Science and Engineering*, Affiliated East-West Press, New Delhi, (2001).

18. C.R. Rao and S.K. Mitra, *Generalized Inverse of Matrices and Its Application*, Wiley, New York, (1971).
19. V. Lakshmikantham, S.K. Sen and G.W. Howell, Vectors versus matrices: p-inversion, cryptographic applications, and vector implementation, *Neural, Parallel & Scientific Computations*, **4**, 129-140, (1996).
20. R. Penrose, A generalized inverse for matrices, *Proc. Chemb. Phil. Soc.*, **51**, 406-413, (1955).
21. E.H. Moore, On the reciprocal of the general algebraic matrix (abs.), *Bull. Amer. Math. Soc.,* **26**, 394-395, (1920).
22. S.K. Sen and E.V. Krishnamurthy, Rank-augmented LU-algorithm for computing generalized matrix inverses, *IEEE Trans. Computers*, **C-23**, 199-201, (1974).
23. S.K. Sen and S.S. Prabhu, Optimal iterative schemes for computing Moore-Penrose matrix inverse, *Int. J. Systems. Sci.*, **8**, 748-753, (1976).