Neural, Parallel, and Scientific Computations 19 (2011) 211-228

PREPROCESSING IN MATLAB INCONSISTENT LINEAR SYSTEM FOR A MEANINGFUL LEAST SQUARES SOLUTION

SYAMAL K. SEN[1] and GHOLAM ALI SHAYKHIAN[2]

[1]*Department of Mathematical Sciences, Florida Institute of Technology, 150 West University Boulevard, Melbourne, FL 32901-6975, United States*
sksen@fit.edu

[2]*National Aeronautics and Space Administration (NASA), Technical Integration Office (IT-G), Information Technology (IT) Directorate, Kennedy Space Center, FL 32899, United States*
ali.shaykhian@nasa.gov

Abstract Mathematical models of many physical/statistical problems are systems of linear equations. Due to measurement and possible human errors/mistakes in modeling/data, as well as due to certain assumptions to reduce complexity, inconsistency (contradiction) is injected into the model, viz. the linear system. While any inconsistent system irrespective of the degree of inconsistency has always a least-squares solution, one needs to check whether an equation i.e. an information is too much inconsistent or, equivalently, too much contradictory. Such an equation will affect/distort the least-squares solution to such an extent that it becomes unacceptable/unfit to be used in a real-world application. We propose an algorithm, in Matlab, which (i) can detect and prune numerically redundant linear equations from the system, if necessary, as these do not add any new information to a non-least-squares model, although they do have significant impact in a least-squares model, (ii) detects contradictory linear equations along with a degree of contradiction (inconsistency index) and then (iii) obtain the minimum norm least-squares solution of the acceptably inconsistent reduced (pruned) linear system as well as that of non-reduced linear system without too contradictory equations. The resulting two solution vectors will be different in general and have important implication in a real-world environment. The algorithms presented in Matlab may reduce the computational and storage complexities and also may improve the accuracy of the solution. These also detect and provide the necessary warning if there exists a highly contradictory equation in the model. In addition, we suggest a thorough relook into the mathematical modeling to determine the reason why unacceptable contradiction has occurred thus prompting one to make necessary corrections/modifications to the models – both mathematical and, if necessary, physical. We will focus here mainly on the non-over-determined linear systems rather than over-determined systems which are often usually the case in a least-squares problem.

Keywords Minimum-norm least-squares solution; Non-over-determined systems; Pruning; Redundant linear equations; Too much inconsistent.

1. INTRODUCTION

There are many physical problems whose mathematical models turn out to be numerical linear systems $Ax = b$, with more equations than the number of variables (over-determined), or with less equations than the number of variables (underdetermined), or with the number of equations same as the number of variables. Specifically, statistical

Received June 1, 2011 1061-5369 $15.00 © Dynamic Publishers, Inc

problems giving rise to multivariate linear/multiple regression models are encountered in many real-world problems such as weather forecasting, psychological research, and business management. The systems will in general be inconsistent, i.e. the equations in a system will be, in general, contradictory to a varying extent. Consequently there will be no solution that will satisfy the linear system. If the system happens to be consistent (non-contradictory), then there will always be a solution which will satisfy all the equations in the system. If we attempt to find a least-squares solution x_l of such a consistent system, it will always be a true solution of the system and sum of the squares of the residuals $\| Ax_l - b \|^2$ will always be a numerical zero (defined in the context) [3].

A least-squares solution of any linear system $Ax = b$, consistent or not, always exist and can be readily computed just by computing the true solution of the ever consistent system $A^t Ax = A^t b$, where t denotes the transpose. A least-squares solution x_l is that solution for which the sum of the squares of the residuals viz. $\| Ax_l - b \|^2$ is the least, where $\| \quad \|$ denotes the Euclidean norm. This solution may not be unique. However, one of the possible least-squares solutions, viz. x_{ml} for which $\| Ax_{ml} - b \|^2$ and also $\|x_{ml}\|$ are both minimum out of all possible least-squares solutions x_l, is known as the minimum-norm least-squares solution (mls). The mls x_{ml} is always unique while the solution x_l may not be unique. When a least-squares solution of the system $Ax = b$ is not unique, the system will then always have infinity of least-squares solutions. Consider, for example, the system $Ax = b$, where

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ 15 \\ 23 \end{bmatrix}.$$

Using the Matlab commands (keyed in one line)

```
>> A=[1 2 3;4 5 6;7 8 9], b=[6 15 23]', B=A'*A, c=A'*b, xt=B\c,nt=norm(A*xt-b),
xmt=pinv(A)*b, nmt=norm(A*xmt-b), nxt=norm(xt), nxmt=norm(xmt)
```

we obtain a least-squares solution xt = [3.4091 -4.4848 3.9091]^t and the mls xmt =[0.6944 0.9444 1.1944]^t. Yet another least-squares solution is obtained using the Gauss reduction involving the following Matlab commands, which have used the foregoing matrix B and vector c,

```
>>E1=[1 0 0;-78/66 1 0;-90/66 0 1], B1=E1*B, c1=E1*c
>> E2=[1 0 0; 0 1 0; 0 -B1(3,2)/B1(2,2) 1]
>> B2=E2*B1, c2=E2*c1
>> x3=(c2(2)-B2(2,2))/B2(2,3)
>> x2=1, x1=(227-78*x2-90*x3)/66
```

This second least-squares solution then in an exact form is $x_t = [\frac{2}{3} \quad 1 \quad \frac{7}{6}]^t$. There are, in fact, infinity of least-squares solutions for the foregoing problem. The sum of the squares of the residuals viz. $\| Ax - b \|^2$ for all the foregoing three solutions, viz. $x_t = [\frac{2}{3} \quad 1 \quad \frac{7}{6}]^t$, xt = [3.4091 -4.4848 3.9091]^t, xmt =[0.6944 0.9444 1.1944]^t are numerically the same, viz. 0.4082 while their norms are different. These are 1.6750, 6.8569, and 1.6736, respectively.

Here we will be computing the numerical *mls* x_{ml} instead of the solution x_l. For any least-squares problem, computing x_{ml} instead of x_l is absolutely fine and is usable for any real-world application unless one has some other constraints in mind.

It may be seen that the number of solutions of any linear system could be either 0 or just 1 or infinite. It can never be just 2 or just 3 or just n, where n is a finite positive integer. For if there are two solutions then a linear combination of these two solutions is also a solution of the system $Ax = b$.

We present in section 2 the algorithms consisting of (i) pruning redundant (linearly dependent) equations, (ii) locating the equations, if any, that are unacceptably contradictory along with a computation of the inconsistency index *inci*, and (iii) computing the *mls* of the pruned system that does not have too much (unacceptable inconsistency) contradiction in the context. The pruned system, i.e. the system without linearly dependent rows, involves less storage and less computation resulting in less computational error as well as less space (storage) complexity. However, in the context of a least-squares solution for a least-squares model, pruning linearly dependent rows may alter the least-squares solution vector significantly to make the desired solution worse or better depending on the problem on hand. One may compute both the pruned solution and the un-pruned solution to correlate/interpret them in the physical/statistical problem under consideration. An un-pruned least-squares solution of the inconsistent system involving several linearly dependent rows of the augmented matrix $[A, b]$ is significant unlike a one for a fully consistent system. The algorithm detects excessive contradiction in an equation, if any, in the system and cautions the concerned researcher so that he falls back to the physical problem and the resulting mathematical model, finds out the reason for unacceptable inconsistency in the model, takes appropriate corrective measures, and then computes the required solution. It will definitely not serve any purpose to compute the *mls* (that can always be easily computed) of any system (having unacceptable inconsistency). In section 3, we provide numerical examples while in section 4 we include conclusions.

2. THE ALGORITHM

The algorithm is comprised of three sub-algorithms, viz. pruning sub-algorithm, over-inconsistency detection sub-algorithm, and *mls* solver sub-algorithm. We briefly describe these sub-algorithms with a numerical example. However, it is always possible as well as easy to combine these sub-algorithms into one algorithm and execute the algorithm.

Sub-algorithm 1 (Pruning redundant equations) Consider the linear system (generally inconsistent)

$$Ax = b, \qquad (1)$$

where A is a given numerical $m \times n$ matrix, b is a given numerical m-vector, and x is an n-vector to be computed in the least-squares sense. We assume or we know for certain that the very first equation of the system (usually over-determined) $Ax = b$ is correct. We first consider the augmented matrix $C = (A, b)$ which will be our coefficient matrix. The right-hand side column vector d is computed as the row sum of the matrix $C = c_{ij}$. That is, $d_i = \sum_{j=1}^{n+1} c_{ij}$, $i = 1(1)m$. Thus the system $Cy = d$ is always consistent. We now use the following Matlab program (self-explanatory) to sieve out the linearly dependent rows (if any) of C as these rows are not required for computing a solution vector. For the mathematics/justification of this program refer [1, 2]

```
function pruningldrows(C);
% C=Usually augmented matrix (A,b) of Ax=b
%Redundant (linearly dependent) rows of the C are pruned.
b=sum(C')'; A=C; redrow(1)=0;

disp('Given unpruned matrix is'); C
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(A(:,n-1)))/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
         % Store indices of redundant rows in a vector.
         redrow(k)=i; k=k+1; rrow=1;
      end
end; disp('Linearly dependent rows are'); redrow
if rrow==1
c(redrow)=[]; B(redrow,:)=[];
end
if p ~= 1
   S=size(B);
   if S(1)<=m
      disp('The rank of the matrix C or the pruned matrix B is '); disp(r);
      disp('The pruned matrix B is');B
   end
end
```

```
>> Ab =  1     2     3     4     5    15
         2     4     6     8    10    30
         3     6     9    12    15   400
         1     1     1     1     1    10
         2     2     2     2     2    10
         8    16    24    32    40   120
        11    23    35    47    59   175
```

where Ab= [A, b] is the augmented matrix where the first 5 columns constitute the coefficient matrix A while the last column of Ab is the right-hand side bb vector which is constructed by the row-sum of each row of the matrix A. Consequently, the system

$Abx = bb$ is always consistent. The Matlab command >> **pruningldrows([Ab bb])** produces pruned system as follows.

Given unpruned matrix is

C = [Ab bb]
```
      1      2      3      4      5     15    30
      2      4      6      8     10     30     60
      3      6      9     12     15    400    445
      1      1      1      1      1     10     15
      2      2      2      2      2     10     20
      8     16     24     32     40    120    240
     11     23     35     47     59    175    350
```

Linearly dependent rows are redrow = 2 5 6 7

The rank of the matrix C or the pruned matrix B is 3

The pruned matrix B is

B =
```
   1   2   3   4   5   15   30
   3   6   9  12  15  400  445
   1   1   1   1   1   10   15
```

Sub-algorithm 2 (Over-inconsistency detection) Having pruned linearly dependent (redundant) rows, the resulting system will be left with equations $Bx = c$ which could be (i) consistent or (ii) acceptably inconsistent or (iii) unacceptably (over-) inconsistent (in the context). In case (i), we compute the *mls* x_{ml} (unique) of $Bx = c$, which is a true solution of $Bx = c$, i.e. . $\| Bx_{ml} - c \|$ = 0 (numerical zero). In case (ii) also, we compute the *mls* x_{ml} of $Bx = c$, which is not a true solution but a solution such that the norm $\| Bx_{ml} - c \|$ i.e. the square-root of the sum of the squares of the residual, is a minimum and not a numerical zero. In case (iii), once we detect an abnormally (too) inconsistent equation or, equivalently, an *outlier* (analogous to a data point which is far away from other data points or a cluster of data points in statistics), we do not proceed to compute the *mls* of $Bx = c$ as it may not often be useful. Instead, we go back to the original physical (real-world) problem along with its corresponding mathematical model to determine the cause of such an unacceptably large contradiction (inconsistency) and take the necessary corrective measures before computing the *mls* of $Bx = c$. The following Matlab code (program) **nclinsolver(A, b)** which is self-explanatory detects the equations that cause over-inconsistency [2].

```
%Computational NC-LINSOLVER (Matlab program) The following program is self-explanatory.
%A general LINSOLVER program that includes near-consistent linear %system.
%Reference: S.K. Sen and Sagar Sen, Linear System: Relook, concise
%algorithms, and Matlab programs, Academic Studies – National Journal
%of Jyoti Research Academy, Feb., 2007, Vol. 1(1), pp. 1-8.
```

```
function[ ]=nclinsolver(A,b); [m, n]=size(A);
%NC-LINSOLVER: Near-consistent Linear System Solver
'The matrix  A  and vector  b  of the system  Ax=b  are', A,b,
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 nearest consistent system is', x,
'Error in the solution vector  x  is', err
```

The matrix AAA is our B matrix of the pruned system $Bx = b$ while the vector bbb is our vector b of the pruned system. The pruned system could be consistent or acceptably inconsistent or unacceptably inconsistent. We now detect the equation that causes the inconsistency, using the Matlab command >> **nclinsolver(AAA, bbb)**. The matrix AAA and vector bbb of the system AAAx=bbb are

```
AAA =
    1    2    3    4    5
    3    6    9   12   15
    1    1    1    1    1
```

bbb =[15 400 10]^t

The system Ax=b (i.e. AAAx=bbb) is inconsistent. The projection operator P = (I - A+A) (i.e. $P = (I - A^+A)$ needed for a solution $x_c = Pz$, of the homogeneous equation $Ax = 0$, where z is an arbitrary column vector and A^+ is the minimum-norm least squares (ml) inverse of the matrix A, also called the pseudo-inverse or the Moore-Penrose inverse of 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
```

The rank of the matrix A is r = 2. The inconsistency index is inci = 6.5950. Modification in vector b, i.e., Db is delb = [0 -355 0]^t. Vector b of the nearest consistent system is b = [15 45 10]^t. Solution vector of the nearest consistent system is x = [5.0000 3.5000 2.0000 0.5000 -1.0000]^t. Error in the solution vector x is err = 54.4545. delb in the foregoing solution shows that the second element, viz. -355 is unacceptably large. Ideally it should have been close to zero for an acceptable inconsistency. This implies that one needs to fall back to the original physical problem and its mathematical model to ascertain why such a large contradiction has occurred and to take necessary corrective measures. The inconsistency index inci as well as the error in the solution vector provide us enough information about the equation which has become too contradictory. One may recall that we assumed that the very first equation representing an information is correct. Or, in other words, we have already carefully checked from the physical model and its corresponding mathematical model that the first equation is correct or any of the equations of the system which is definitely known to be correct will be placed as our very first equation. This is because all the other equations, each one of which representing an information (an assertive sentence), are checked against the very first equation. A more desirable thing will be to assemble all those equations in the beginning (at the top) which have been checked and rechecked thoroughly and found to be definitely correct. In the foregoing system, clearly the second equation is the one which is too inconsistent and may be considered an outlier in a statistical sense.

However, if we have the same foregoing system with 40 instead of 400, then we will have inconsistency index inci, delb (to make the system consistent), as well as err (error in the solution vector x) as

inci = 0.0929, delb = [0 5 0]^t, and err = 0.7670,

respectively. In the given context, such a relatively small contradiction could be acceptable.

The solution vector of the nearest consistent system in the foregoing nclinsolver program has been shown so that one can compare the actual *mls* to be computed by the sub-algorithm 3 against it although it is not our focus.

Sub-algorithm 3 Minimum-norm least-squares solver (*ml* solver) Having confirmed that the system $Bx = c$ is an acceptably inconsistent system from the sub-algorithm 2, we simply compute the *mls* of $Bx = c$. While one can always compute simply a least-squares solution (need not be unique) of $Bx = c$ by any of the available numerical methods in literature [4-11], we compute the *mls* simply by using the one concise Matlab command >>**xc=pinv(B)*c**. This solution is also a least-squares solution and it serves our purpose perfectly well as long as our system is not too large. If the system is too large so that it is beyond the scope of the general Matlab command *pinv* from both storage point of view as well as from error point of view, we need to use a special software package suited for computing a least-squares solution of the system.

Our acceptably inconsistent system is AAAx=bbb, where

```
AAA =
     1     2     3     4     5
     3     6     9    12    15
     1     1     1     1     1
```

and bbb = $[15\ \ 40\ \ \ 10]^t$

>> **x=pinv(AAA)*bbb**

x = $[5.3000\ \ 3.6500\ \ 2.0000\ \ \ 0.3500\ \ -1.3000]^t$

One may compare the error erml due to this *mls* of the system AAAx=bbb with error errnc due to the solution of the foregoing nearest consistent system in Matlab as follows.

>> **errml=norm(AAA* [5.3000 3.6500 2.0000 0.3500 -1.3000]'-bbb)**

errml = 1.5811

>> **errnc=norm(AAA*[5.0000 3.5000 2.0000 0.5000 -1.0000]'-bbb)**

errnc = 5

It can be seen that error in the *mls* of the system is less possibly prompting us to infer that the inconsistency is acceptable. Although we do not have sufficient pressing reason to bother about nearest consistent system, we could see how much deviation in the right-hand side vector bbb is required for the original pruned inconsistent system to be consistent. If the numerical deviation is too much, then evidently the inconsistency is also too much and is unacceptable for a real-world implementation. Consider, for instance, the foregoing system with bbb=[15 400 10]’ instead of bbb=[15 40 10]’. Then we have

>> **bbb=[15 400 10]'**

bbb = $[15\ \ 400\ \ 10]^t$

>> **xml=pinv(AAA)*bbb**

xml = $[-16.3000\ \ -7.1500\ \ \ 2.0000\ \ 11.1500\ \ \ 20.3000]^t$

>> **errml1=norm(AAA* [5.3000 3.6500 2.0000 0.3500 -1.3000]'-bbb)**

errml1 = 359.5031

This error errml1 is sufficiently greater than the foregoing errnc. This indicates that the inconsistency is too large to be accepted. However, the acceptance of inconsistency very much depends on the context and is somewhat subjective.

Consider the following constructed (using Matlab rand command) over-determined system Ax=b.

>> **A=rand(12, 5), b=sum(A')'**

```
A =
   0.8147   0.9572   0.6787   0.6948   0.7094
   0.9058   0.4854   0.7577   0.3171   0.7547
   0.1270   0.8003   0.7431   0.9502   0.2760
   0.9134   0.1419   0.3922   0.0344   0.6797
   0.6324   0.4218   0.6555   0.4387   0.6551
   0.0975   0.9157   0.1712   0.3816   0.1626
   0.2785   0.7922   0.7060   0.7655   0.1190
   0.5469   0.9595   0.0318   0.7952   0.4984
   0.9575   0.6557   0.2769   0.1869   0.9597
   0.9649   0.0357   0.0462   0.4898   0.3404
   0.1576   0.8491   0.0971   0.4456   0.5853
   0.9706   0.9340   0.8235   0.6463   0.2238
```

b = [3.8548 3.2207 2.8966 2.1616 2.8034 1.7286 2.6613 2.8318 3.0368 1.8769 2.1347 3.5982]^t

The augmented matrix $C = [A \quad b]$ is obtained as follows.

```
>> C=[A b]
```

```
C =
   0.8147   0.9572   0.6787   0.6948   0.7094   3.8548
   0.9058   0.4854   0.7577   0.3171   0.7547   3.2207
   0.1270   0.8003   0.7431   0.9502   0.2760   2.8966
   0.9134   0.1419   0.3922   0.0344   0.6797   2.1616
   0.6324   0.4218   0.6555   0.4387   0.6551   2.8034
   0.0975   0.9157   0.1712   0.3816   0.1626   1.7286
   0.2785   0.7922   0.7060   0.7655   0.1190   2.6613
   0.5469   0.9595   0.0318   0.7952   0.4984   2.8318
   0.9575   0.6557   0.2769   0.1869   0.9597   3.0368
   0.9649   0.0357   0.0462   0.4898   0.3404   1.8769
   0.1576   0.8491   0.0971   0.4456   0.5853   2.1347
   0.9706   0.9340   0.8235   0.6463   0.2238   3.5982
```

The 13th, 14th, and 15th rows of C are constructed using the following three Matlab commands.

```
>> C(13,:)=2*C(12,:)+3*C(11,:)
>> C(14,:)=-1*C(10,:)+4*C(9,:)
>> C(15,:)=-1*C(8,:)+4*C(7,:)
```

Thus the 15×6 augmented matrix C is then

```
C =
   0.8147   0.9572   0.6787   0.6948   0.7094   3.8548
   0.9058   0.4854   0.7577   0.3171   0.7547   3.2207
   0.1270   0.8003   0.7431   0.9502   0.2760   2.8966
   0.9134   0.1419   0.3922   0.0344   0.6797   2.1616
   0.6324   0.4218   0.6555   0.4387   0.6551   2.8034
   0.0975   0.9157   0.1712   0.3816   0.1626   1.7286
   0.2785   0.7922   0.7060   0.7655   0.1190   2.6613
   0.5469   0.9595   0.0318   0.7952   0.4984   2.8318
   0.9575   0.6557   0.2769   0.1869   0.9597   3.0368
   0.9649   0.0357   0.0462   0.4898   0.3404   1.8769
   0.1576   0.8491   0.0971   0.4456   0.5853   2.1347
   0.9706   0.9340   0.8235   0.6463   0.2238   3.5982
   2.4140   4.4154   1.9383   2.6294   2.2034  13.6005
   2.8651   2.5873   1.0615   0.2577   3.4986  10.2702
   0.5671   2.2093   2.7924   2.2669  -0.0224   7.8133
```

We then inject inconsistency in 14th and 15th equations

```
>> C(14,6)=C(14,6)-1, C(15,6)=C(15,6)+2  (%Inconsistency is injected in 14th and 15th equations)
```

Thus the augmented matrix C of the inconsistent system becomes

```
C =
   0.8147   0.9572   0.6787   0.6948   0.7094   3.8548
   0.9058   0.4854   0.7577   0.3171   0.7547   3.2207
   0.1270   0.8003   0.7431   0.9502   0.2760   2.8966
   0.9134   0.1419   0.3922   0.0344   0.6797   2.1616
   0.6324   0.4218   0.6555   0.4387   0.6551   2.8034
   0.0975   0.9157   0.1712   0.3816   0.1626   1.7286
   0.2785   0.7922   0.7060   0.7655   0.1190   2.6613
   0.5469   0.9595   0.0318   0.7952   0.4984   2.8318
   0.9575   0.6557   0.2769   0.1869   0.9597   3.0368
   0.9649   0.0357   0.0462   0.4898   0.3404   1.8769
   0.1576   0.8491   0.0971   0.4456   0.5853   2.1347
   0.9706   0.9340   0.8235   0.6463   0.2238   3.5982
   2.4140   4.4154   1.9383   2.6294   2.2034  13.6005
   2.8651   2.5873   1.0615   0.2577   3.4986   9.2702
   0.5671   2.2093   2.7924   2.2669  -0.0224   9.8133
```

If we use the Matlab command >> pruningldrows(C) to prune linearly dependent rows then we have:

```
 Linearly dependent rows are

redrow =  6    7    8    9   10   11   12   13   15
```

The rank of the matrix C or the pruned matrix B is 6

The pruned matrix B is

```
B =
   0.8147    0.9572    0.6787    0.6948    0.7094    3.8548
   0.9058    0.4854    0.7577    0.3171    0.7547    3.2207
   0.1270    0.8003    0.7431    0.9502    0.2760    2.8966
   0.9134    0.1419    0.3922    0.0344    0.6797    2.1616
   0.6324    0.4218    0.6555    0.4387    0.6551    2.8034
   2.8651    2.5873    1.0615    0.2577    3.4986    9.2702
```

The rank of the augmented matrix is obtained using the Matlab command >> rank(C) as 6 (thus C is a matrix with full-column rank).

Thus the right-hand side vector bb of the pruned linear system Bx=bb is obtained using the command >> bb=B(:,6) as

bb = [3.8548   3.2207   2.8966   2.1616   2.8034   9.2702]^t

while the coefficient matrix B is sieved out using the command >> B= B(1:6, 1:5) as

```
B =
   0.8147    0.9572    0.6787    0.6948    0.7094
   0.9058    0.4854    0.7577    0.3171    0.7547
   0.1270    0.8003    0.7431    0.9502    0.2760
   0.9134    0.1419    0.3922    0.0344    0.6797
   0.6324    0.4218    0.6555    0.4387    0.6551
   2.8651    2.5873    1.0615    0.2577    3.4986
```

(The original augmented matrix B no longer exists.)

The Matlab program >> **nclinsolver(B, bb)** when executed produces the following results (omitting, however, the print output of the coefficient matrix B and the right-hand side vector bb).

The system Ax=b (i.e. Bx=bb) is inconsistent.

The projection operator P = (I - A+A) i.e. $P = (I - B^{+}B)$ is

```
P = 1.0e-014 *
   0.2554   -0.3381   -0.2512    0.3830   -0.0999
  -0.3381    0.0216   -0.1015   -0.0459    0.4447
  -0.2512   -0.1015   -0.1620    0.0737    0.4038
   0.3830   -0.0459    0.0737    0.0737   -0.4698
  -0.0999    0.4447    0.4038   -0.4698   -0.1887
```

which is a numerical null (zero) matrix.

The rank of the matrix A (i.e. B) is r = 5

The inconsistency index is inci = 0.0729

Modification in vector b, i.e., Db is delb = $[\,0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.9985]^t$

Vector b of the nearest consistent system is

b = $[3.8548 \quad 3.2207 \quad 2.8966 \quad 2.1616 \quad 2.8034 \quad 10.2687]^t$

Solution vector of the nearest consistent system is

x = $[1.0007 \quad 1.0000 \quad 1.0002 \quad 1.0001 \quad 0.9989]^t$

Error in the solution vector x is err = 0.4466

If we now consider the un-pruned system CCx=bc, where

bc = $[3.8548 \quad 3.2207 \quad 2.8966 \quad 2.1616 \quad 2.8034 \quad 1.7286 \quad 2.6613 \quad 2.8318 \quad 3.0368 \quad 1.8769 \quad 2.1347 \quad 3.5982 \quad 13.6005 \quad 9.2702 \quad 9.8133]^t$

and

```
CC =
   0.8147    0.9572    0.6787    0.6948    0.7094
   0.9058    0.4854    0.7577    0.3171    0.7547
   0.1270    0.8003    0.7431    0.9502    0.2760
   0.9134    0.1419    0.3922    0.0344    0.6797
   0.6324    0.4218    0.6555    0.4387    0.6551
   0.0975    0.9157    0.1712    0.3816    0.1626
   0.2785    0.7922    0.7060    0.7655    0.1190
   0.5469    0.9595    0.0318    0.7952    0.4984
   0.9575    0.6557    0.2769    0.1869    0.9597
   0.9649    0.0357    0.0462    0.4898    0.3404
   0.1576    0.8491    0.0971    0.4456    0.5853
   0.9706    0.9340    0.8235    0.6463    0.2238
   2.4140    4.4154    1.9383    2.6294    2.2034
   2.8651    2.5873    1.0615    0.2577    3.4986
   0.5671    2.2093    2.7924    2.2669   -0.0224
```

then we have the *mls* x1 of the pruned system Bx-bb and the *mls* x2 of the non-pruned system CCx=bc obtained using the commands

```
>> x1=pinv(B)*bb, x2=pinv(CC)*bc
```

as
x1 = $[1.4061 \quad 0.8277 \quad 1.0575 \quad 1.2195 \quad 0.4759]^t$
x2 = $[0.8656 \quad 0.8763 \quad 1.5941 \quad 1.1482 \quad 0.7234]^t$

Their respective norms are, using

>> **norm(x1), norm(x2),**

2.3439 and 2.4290, respectively.

>> **BB=[B;C(15,1:5)], bbb=[bb; C(15,6)]**

```
BB =
    0.8147    0.9572    0.6787    0.6948    0.7094
    0.9058    0.4854    0.7577    0.3171    0.7547
    0.1270    0.8003    0.7431    0.9502    0.2760
    0.9134    0.1419    0.3922    0.0344    0.6797
    0.6324    0.4218    0.6555    0.4387    0.6551
    2.8651    2.5873    1.0615    0.2577    3.4986
    0.5671    2.2093    2.7924    2.2669   -0.0224
```

bbb = $[3.8548 \quad 3.2207 \quad 2.8966 \quad 2.1616 \quad 2.8034 \quad 9.2702 \quad 9.8133]^t$

>> **x3=pinv(BB)*bbb**

x3 = $[1.3955 \quad 1.4626 \quad 1.8621 \quad 0.2521 \quad -0.1601]^t$

>> **norm(x3)**

```
    2.7647
```

>> **x1=pinv(B)*bb**

x1 = $[1.4061 \quad 0.8277 \quad 1.0575 \quad 1.2195 \quad 0.4759]^t$

>> **norm(x1)**

```
    2.3439
```

>> **CC**

```
CC =
    0.8147    0.9572    0.6787    0.6948    0.7094
    0.9058    0.4854    0.7577    0.3171    0.7547
    0.1270    0.8003    0.7431    0.9502    0.2760
    0.9134    0.1419    0.3922    0.0344    0.6797
    0.6324    0.4218    0.6555    0.4387    0.6551
    0.0975    0.9157    0.1712    0.3816    0.1626
    0.2785    0.7922    0.7060    0.7655    0.1190
    0.5469    0.9595    0.0318    0.7952    0.4984
    0.9575    0.6557    0.2769    0.1869    0.9597
    0.9649    0.0357    0.0462    0.4898    0.3404
```

```
   0.1576    0.8491    0.0971    0.4456    0.5853
   0.9706    0.9340    0.8235    0.6463    0.2238
   2.4140    4.4154    1.9383    2.6294    2.2034
   2.8651    2.5873    1.0615    0.2577    3.4986
   0.5671    2.2093    2.7924    2.2669   -0.0224
```

>> **bc**

bc = [3.8548 3.2207 2.8966 2.1616 2.8034 1.7286 2.6613 2.8318 3.0368 1.8769 2.1347 3.5982 13.6005 9.2702 9.8133]^t

>> **x2=pinv(CC)*bc**

x2 = [0.8656 0.8763 1.5941 1.1482 0.7234]^t

>> **norm(x2)**

```
   2.4290
```

3. NUMERICAL EXAMPLE

Consider the non-over-determined inconsistent system $Ax = b$ created by the following Matlab commands

>> **A=rand(4,6); A(5,:)=1.5*A(1,:)+2*A(2,:); A(6,:)=3*A(2,:)-1.7*A(3,:)+4*A(4,:), b=sum(A')'; b(5)=b(5)+ 2; b(6)=b(6)-1**

where the coefficient matrix A and the right-hand side column vector b are (correct up to 4 decimal digits)

```
A =
   0.6787    0.6555    0.2769    0.6948    0.4387    0.1869
   0.7577    0.1712    0.0462    0.3171    0.3816    0.4898
   0.7431    0.7060    0.0971    0.9502    0.7655    0.4456
   0.3922    0.0318    0.8235    0.0344    0.7952    0.6463
   2.5336    1.3256    0.5077    1.6764    1.4212    1.2598
   2.5788   -0.5594    3.2672   -0.5263    3.0241    3.2970
```

and

b = [2.9316 2.1635 3.7076 2.7235 10.7244 10.0815]^t,

respectively although the actual computation was done with 15 digits. If one tries to compute the rank of A with the foregoing 4-digit entry for each element then one will find that the rank of A as 6 instead of 4. This is due to rounding errors. However, if one retains all the 15 digits for each element of A and then compute the rank of A, then he will get the rank as 4 and the rank of the augmented matrix [A b] as 5.

The system $Ax = b$ is inconsistent. The 5th and 6th rows of the matrix A is linearly dependent on the foregoing four rows. The augmented matrix C obtained by the Matlab command >> **C=[A b]** is (correct up to 4 decimal digits)

```
C =
   0.6787   0.6555   0.2769   0.6948   0.4387   0.1869    2.9316
   0.7577   0.1712   0.0462   0.3171   0.3816   0.4898    2.1635
   0.7431   0.7060   0.0971   0.9502   0.7655   0.4456    3.7076
   0.3922   0.0318   0.8235   0.0344   0.7952   0.6463    2.7235
   2.5336   1.3256   0.5077   1.6764   1.4212   1.2598   10.7244
   2.5788  -0.5594   3.2672  -0.5263   3.0241   3.2970   10.0815
```

The linearly dependent (ld) row of C obtained using the command >> **pruningldrows(C)** is its 6th row. The rank of the matrix C or the pruned matrix B is 5 (The input data were kept correct up to 15 decimal digits and not up to 4 decimal digits in actual computation)

The right-hand side column vector bb is (correct up to 4 decimal digits) bb = [2.9316 2.1635 3.7076 2.7235 10.7244]^t. The pruned matrix B is (up to 4 decimal digits), using the command >> **BB=B(:,1:6)**,

```
BB =
   0.6787   0.6555   0.2769   0.6948   0.4387   0.1869
   0.7577   0.1712   0.0462   0.3171   0.3816   0.4898
   0.7431   0.7060   0.0971   0.9502   0.7655   0.4456
   0.3922   0.0318   0.8235   0.0344   0.7952   0.6463
   2.5336   1.3256   0.5077   1.6764   1.4212   1.2598
```

We now detect the equation(s), using the command >> **nclinsolver(BB,bb)**, that are inconsistent as follows.

The system Ax=b (i.e. BBx=bb) is inconsistent. The rank of the matrix A (i.e. BB) is r = 4. The inconsistency index is inci = 0.1746. Modification in vector b, i.e., Db is delb = [0 0 0 0 -2.0001]^t. The fifth element of delb depicts that the fifth equation is inconsistent. Vector b (i.e. bb) of the nearest consistent system is b = [2.9316 2.1635 3.7076 2.7235 8.7243]^t. Solution vector of the nearest consistent system is x = [1.1102 0.8803 0.9555 0.9595 1.2137 0.7351]^t. Error in the solution vector x is err = 0.8266.

The *mls* of the pruned system and that of un-pruned system along with their norms are given using the Matlab commands

>> **x1=pinv(BB)*bb, x1n=norm(x1), x2=pinv(A)*b, x2n=norm(x2)**

where x1 = 1.0e+003 * [3.8250 -6.4839 -0.5869 1.0113 6.5185 -9.3243]^t
is the *mls* of the pruned system and x1n = 1.3692e+004 is the norm of x1.

The *mls* of the un-pruned system is x2 = 1.0e+004 * [0.7794 2.2436 -1.5833 -3.4841 2.6687 -1.6636]^t and its norm is x2n = 5.4933e+004.

The square-roots of the sums of the squares of the residuals of the pruned and un-pruned systems are, using the commands >> **x1t=norm(BB*x1-bb), x2t=norm(A*x2-b),** x1t = 1.2841e-011 and x2t = 2.3662e-010, respectively. Here we see that the sum of the squares of the residuals for the pruned system is less than that of un-pruned system. In reality, it can be the other way also for some other least-squares problems. However, the foregoing inconsistency due to the fifth equation may be quite acceptable in most real-world computations.

4. **CONCLUSIONS**

Inconsistency for non-over-determined system $Ax = b$ is due to linear dependence of a row of the matrix A while the corresponding element of b on the right-hand side is not linearly dependent. In other words the row(s) of A are linearly dependent while the corresponding row(s) of the augmented matrix $[A, b]$ are not.

It is possible to know globally whether inconsistency is relatively large or not by computing the sum of squares of the residuals, viz. $\| Ax_{ml} - b \|$, where $\| \quad \|$ denotes the Euclidean norm and x_{ml} is the mls of the system $Ax = b$. This sum should be acceptably low in the context. But it is not possible to know the equations which have caused the inconsistency. In fact, it is more important to detect the equations which cause the excessive inconsistency and take necessary corrective measures before proceeding. Hence it is necessary to fall back on to the original physical/statistical problem and the corresponding mathematical model and find out the reason for over-inconsistency and correct the model accordingly before proceeding to compute the mls.

While for consistent systems, linearly dependent rows do not carry any new information and are hence redundant, for inconsistent systems, the linearly dependent rows do carry information about the physical problem and hence do not always deserve to be pruned (in many contexts). The mls for pruned and non-pruned inconsistent systems will be, in general, different. The sum of the squares of the residuals could be significantly more or significantly less than that for the non-pruned system. The foregoing numerical examples demonstrate this fact.

We have omitted the algorithms in mathematical form and provided them in Matlab codes (programs). One can readily translate these codes to algorithms in mathematical form if it is more convenient to appreciate the essence of the algorithms.

For pruning linearly dependent rows as well as for the detection of the equations that cause inconsistency, one may use Gauss reduction with partial pivoting. But this method involves row interchanges and also the need for keeping track of row numbers. These, specifically row interchanges, are not desirable in many situations. For instance, a situation where row interchanges disturb the structure of the matrix such as the one having one main diagonal with two diagonals above and two diagonals below. Of course, both pruning and detection of inconsistent equations assume that the first equation is definitely correct. This does not necessarily imply that some other subsequent linearly independent equation is also correct.

References

[1] S.K. Sen, S. Ramakrishnan, R.P.Agarwal, and G.A. Shaykhian, Should pruning be a pre-processor of any linear system? Journal of Applied Mathematics and Informatics (JAMI), (2011), to appear.

[2] 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.

[3] V. Lakshmikantham, S.K. Sen, Computational Error and Complexity in Science and Engineering, Elsevier, Amsterdam, 2005.

[4] C.R. Rao, S.K. Mitra, Generalized Inverse of Matrices and Its Application, Wiley, New York, 1971.

[5] G. Golub, W. Kahan, Calculating the singular values and the pseudo-inverse of a matrix, SIAM J. Numer. Anal. B-2 (1965) 205–224.

[6] E.V. Krishnamurthy, S.K. Sen, Numerical Algorithms: Computations in Science and Engineering, Affiliated East-West Press, New Delhi, 2001.

[7] S.K. Sen, E.V. Krishnamurthy, Rank-augmented LU-algorithm for computing generalized matrix inverses, IEEE Trans. Comput. C-23 (1974), 199-201.

[8] S.K. Sen, S.S. Prabhu, Optimal iterative schemes for computing Moore–Penrose matrix inverse, Internat. J. Systems Sci. 8 (1976) 748–753.

[9] E.A. Lord, V.Ch. Venkaiah, S.K. Sen, A concise algorithm to solve under-/over-determined linear systems, Simulation 54 (1990) 239–240.

[10] E.H. Moore, On the reciprocal of the general algebraic matrix (abs.), Bull. Amer. Math. Soc. 26 (1920) 394–395.

[11] R. Penrose, A generalized inverse for matrices, Proc. Chemb. Phil. Soc. 51 (1955) 406–413.