CHALMERS | UNIVERSITY OF GOTHENBURG

PREPRINT 2014:24

On some numerical methods for solving strongly overdetermined systems of linear equations

IVAR GUSTAFSSON

Department of Mathematical Sciences
Division of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg Sweden 2014

Preprint 2014:24

On some numerical methods for solving strongly overdetermined systems of linear equations

Ivar Gustafsson

Department of Mathematical Sciences
Division of Mathematics
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg, Sweden
Gothenburg, December 2014

Preprint 2014:24
ISSN 1652-9715

Matematiska vetenskaper
Göteborg 2014

On some numerical methods for solving strongly overdetermined systems of linear equations

Ivar Gustafsson[1]

[1]Department of Mathematical Sciences, Division of Mathematics, Chalmers University of Technology, Gothenburg, Sweden, email: ivar@chalmers.se

1 Abstract

In the present paper, we study overdetermined linear systems arising by sampling values in a large number of experiments. The recieved Big Data could contain hundreds of columns and millions of rows. It is not possible to apply the ordinary Multiple Linear Regression (MLR) method on such a matrix since it is too large for our computers to handle in one piece.

Recently, a method called Multiple Bilinear Analysis (MBA) was developed, see Reference [1]. In the MBA method, one of the variables (columns) at a time is regressed with the right side. One very important conclusion drawn in Reference [1] is that it is fairly easy to pin-point the most important variables by the MBA method on Big Data.

In the present study we are concerned about the fit ability of the different methods. It then turnes out that, for problems that are not so strongly overdetermined as the above example, a method based on one or a few iterations of the preconditioned conjugate gradient (PCG) method is a strong alternative to the MBA method.

In the present article the new ideas, MBA and PCG, are compared to the classical MLR method for different sizes of the matrices, in particular for strongly overdetermined systems. The importance of centering the data is stressed and different amount of noise in the data is considered.

Also, rank deficient and almost rank deficient problems are studied. Among all solutions to a rank deficient overdetermined linear system, the PCG method computes the solution with smallest norm, like the pseudoinverse method does, see Reference [2].

Some conclusions made from the first study i Reference [1] and the present investigation are:

1. Normal MLR is not possible on Big Data.

2. It is fairly easy to pin-point the most important variables by MBA on Big Data

3. It is important to center the data in order for the MBA method to have a good fit ability.

4. Selection of the columns (variables) like in the MBA method is a good idea.

5. For strongly overdetermined and not very illconditioned systems, the MBA method, with centering of data, is close to the MLR method when the fit ability is considered.

6. For not so extremly overdetermined systems, a couple of PCG iterations surpass the MBA method regarding fit ability.

7. For all studied overdetermined systems, the PCG method turnes out to be a close approximation to the MLR method.

8. Good fit ability can be obtained by using not all the given columns in the matrix but a certain amount of them, say 50-70 %.

9. The PCG method computes the minimal norm solution to rank deficient problems.

2 Introduction

Datasets, particularly time dependent datasets, rapidly grow very large. This is an identified problem in Big Data. New methods are needed, methods that are not overwhelmed by gathering data like the standard MLR method. Three methods, the MLR method, the MBA method, and the PCG method, are compared with respect to the following properties:

1. The fit ability, measured by the value R2, dependent on the number of rows and columns and the fraction of noise.

2. The feasibility to apply the methods on an SQL database in a reasonable time (a week or so).

3. The behavior of the methods for illconditioned, rank deficient and almost rank deficient problems.

We study three different methods:

1. QR-factorization (backslash in MATLAB). Except from rounding errors and the number of operations needed, this method is equivalent to Gaussian elimination of the corresponding system of normal equations i.e. the MLR method.

2. The method of Multiple Bilinear Analysis (MBA), see Reference [1]. This method is equivalent to one step of the classical Jacobi method, starting from a zero vector, see Reference [2].

3. The preconditioned Conjugate Gradient method, see Reference [2]. For a strongly overdetermined system just one or a few iterations are needed for reasonably good solution.

3 The methods used

Let the overdetermined system of linear equations be

$$Ax = b, \quad A \in R^{m \times n}, m > n, \ b \in R^m, \ x \in R^n \tag{1}$$

3.1 The QR method

The first method computes the least squares solution to (1) by QR-factorization i.e.

$$A = QR, \quad Rx = Q^T b \tag{2}$$

Here, Q is an orthogonal matrix and R is an upper triangulari matrix, so the system $Rx = A^T b$ is easily solved by backward substitution. The solution obtained by (2) is, except from rounding errors and the amount of floating point operations needed, equivalent to the system of normal equations, that is $x = (A^T A)^{-1} A^T b$, but here computed in the numerically more stable way given by (2).

The QR-factorization involved in (1) is preferably made by Householder transformations, see Reference [1]. The computational complexity, the number of floating point operations, for this method is then of order $2mn^2$ (counting both additions and multiplications). This is the reason why this method is not useful for large m and n, compared to the methods to follow.

3.2 The Multiple Bilinear Analysis (MBA) method

In the second method, the corresponding system of normal equations to (1) i.e $A^TAx = A^Tb$ is considered. An approximation to the solution x is computed by $\hat{x} = D^{-1}(A^Tb)$, where D is the diagonal of the matrix A^TA. This technique is in fact equivalent to performing one iteration of the classical Jacobi method on the system of normal equations, starting from the zero vector. We do not consider doing more than one iteration since the Jacobi method does not in general converge for our type of problems. The requirement for convergence is for instance diagonally dominance of the system and this condition is not satisfied here in general. For completeness we give the Jacobi method in MATLAB code:

```
$function\ x = Jacobinorm(A, D, b, start, iter)$
$x = start;$
$for\ i = 1 : iter,$
  $r = A' * (b - A * x);$
  $x = x + r./diag(D);$
$end$
```

Recall that one iteration, starting with the zero vector, gives $x = D^{-1}(A^Tb)$. Let a_j be the j'th column of A, then the solution component $x_j = a_j^Tb/a_j^Ta_j$. Notice that $x_j/\sqrt{b^Tb}$ equals the cosine of the angle between a_j and b. For more details on this method, see Reference [1].

We also note that the computational complexity of this method is of order $4mn$ (counting both additions and multiplications), for computing $D = diag(A^TA)$ and then A^Tb.

3.3 The preconditioned Conjugate Gradient (PCG) method

The third method is the preconditioned conjugate gradient method for solving the normal equation system $A^TAx = A^Tb$. We use the matrix $D = diag(A^TA)$ for preconditioning. This technique is usually called diagonal scaling preconditioning. This method is well known but for completeness we give the algorithm here in MATLAB code:

```
$function\ x = pcgnorm(A, D, b, start, iter)$
$x = start;$
$r = A' * (b - A * x);$
$q = r./diag(D);$
$rgnorm = r' * q;$
$for\ i = 1 : iter,$
  $aq = A' * (A * q);$
  $alpha = rgnorm/(q' * aq);$
  $x = x + alpha * q;$
  $r = r - alpha * aq;$
  $br = r./diag(D);$
  $rnorm = r' * br;$
  $beta = rnorm/rgnorm;$
  $q = br + beta * q;$
  $rgnorm = rnorm;$
$end$
```

We comment that performing just one iteration of the preconditioned conjugate gradient method is equivalent to one iteration with the preconditioned steepest descent method. An advantage with the preconditioned conjugate gradient method, compared to the MBA method, is that we may perform several iterations in order to compute a more accurate approximation to the true least squares solution. In fact this method converges for all systems (1) with matrices A of full rank, see Reference [2]. Indeed, we find in the numerical experiments that this method converges also for rank deficient problems. This observation is outside the theory in Reference [2] and has to be studied more thoroughly from a theoretical point of view in the future.

The computational complexity for one iteration of the preconditioned steepest descent method, when starting with the zero vector, is of order $6mn$, for computing $D = diag(A^TA)$, $r = A^Tb$ and Aq, where $q = D^{-1}r$. Observe that (in the first iteration) α (*alpha* in the code above) may be computed by

$$\alpha = \frac{r^Tq}{q^TA^TAq} = \frac{r^Tq}{(Aq)^T(Aq)}. \tag{3}$$

For each new iteration in the preconditioned conjugate gradient method the computational cost is of order $4mn$ for computing $aq = A^T(Aq)$.

4 Numerical experiments

The computational tests are based on four different kinds of data i.e. columns of the matrix A. In all cases entries of the matrix is at first computed as equally distributed random numbers in the interval (0, 1) and a vector x is computed with normally distributed random numbers with expectation 0 and standard deviation, std=1, and finally a vector y is computed by $y = Ax$.

The columns of A and y are adjusted in four different ways, such that they become:
Case 1: not centered, std=1
Case 2: centered, std=1
Case 3: centered, various std, the first column with std=1 until the n'th column with std=n.
Case 4: centered, various std, the first four columns almost the same, the numbers differ only by normally distributed numbers with std=0.001.
Case 5: centered, various std, the first four columns exactly the same, the rank of the matrix is then $n - 3$.

For completeness the different cases are given in MATLAB code, As is the adjusted A and y is adjusted similarly:

$A = rand(m, n);$
$x = randn(n, 1);$
$y = A * x;$
$As = A./(ones(m, 1) * std(A));\ \%Case1$
$As = (A - (ones(m, 1) * mean(A)))./(ones(m, 1) * std(A));\ \%Case2$
$As = (A - (ones(m, 1) * mean(A))). * (ones(m, 1) * [1 : n]);\ \%Case3$
$for\ k = 2 : 4,\ As(:, k) = As(:, 1) + 0.001 * randn(m, 1);\ end\ \%Case4$
$for\ k = 2 : 4,\ As(:, k) = As(:, 1);\ end\ \%Case5$

Guided by the nice results regarding the MBA method in Reference [1], the columns of the matrix are sorted according to the size of the scalar product of each column and the righthand side i.e in MATLAB code:

$c = As' * ys;$
$[s, I] = sort(abs(c),' descend');$
$As = As(:, I);$

The possible presens of different amount of noise in the right hand side, the vector y, in the different cases, is modeled in MATLAB by

$ys = ys * (1 - noise) + randn(m, 1) * std(ys) * noise;\ \%Amount\ of\ noise$

The amount of noise used in the present experiments is 0, 0.5 or 1.

The experiments are performed on three different sizes of matrices with different degree of overdetermination, number of rows (always) $m = 600$ and number of columns $n = 200, n = 60$ or $n = 10$. The results are presented in Figure 1 up to Figure 10 and is commented on in the section to follow. In each figure the coefficients of determination, R^2, see Reference [3], are given for different numbers of columns in the computed order as described above. The plotted result are everages of 100 runs with different random values of the kinds presented above.

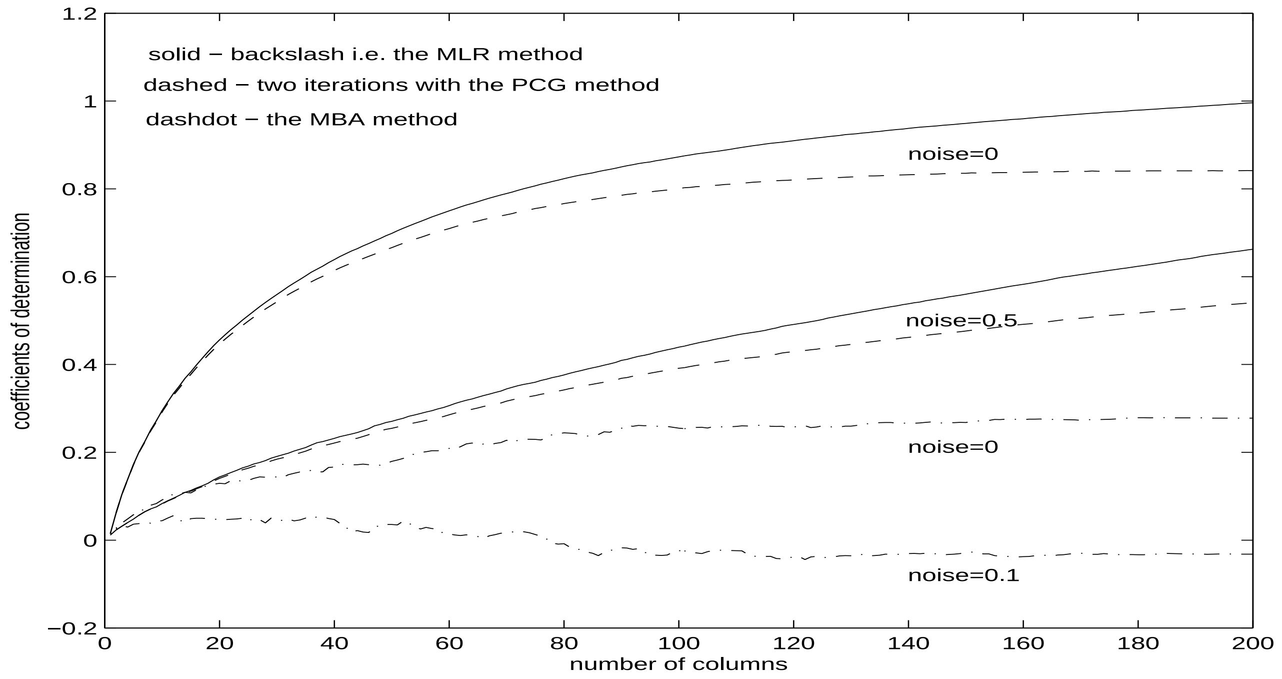

Figure 1: Coefficients of determination, R^2, for three different solution methods, 600 rows, 200 columns, various amount of noise, not centered data, standard deviation one

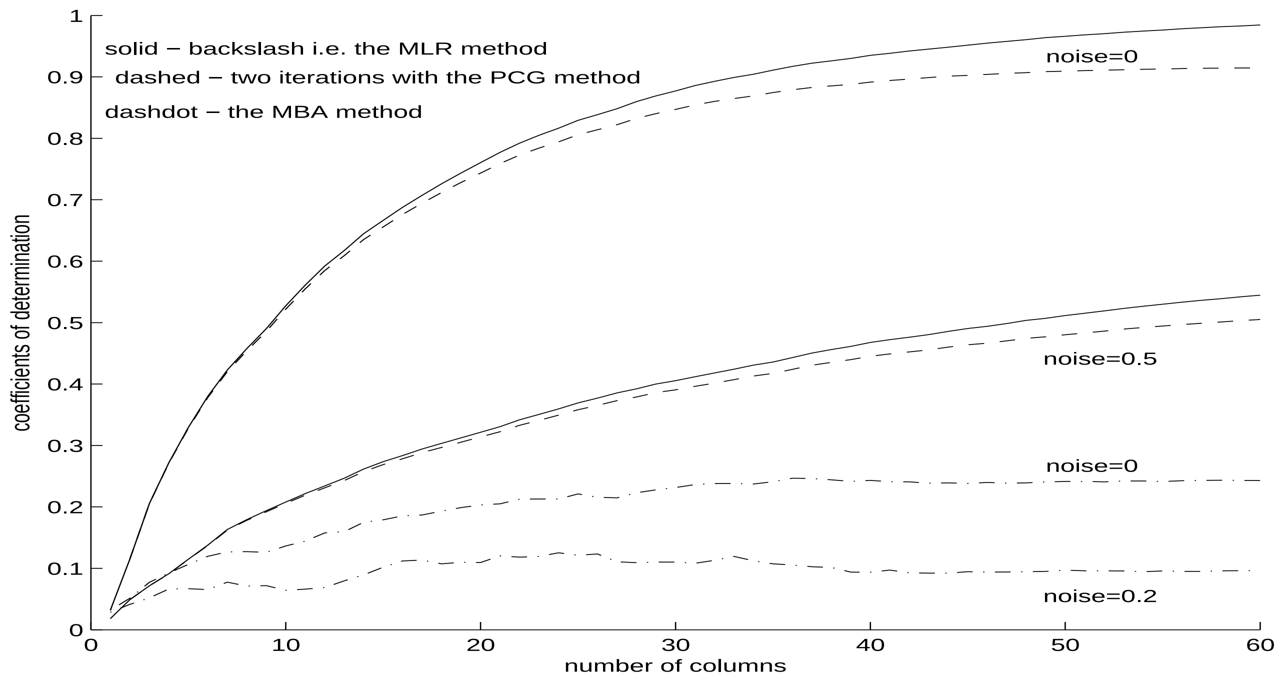

Figure 2: Coefficients of determination, R^2, for three different solution methods, 600 rows, 60 columns, various amount of noise, not centered data, standard deviation one

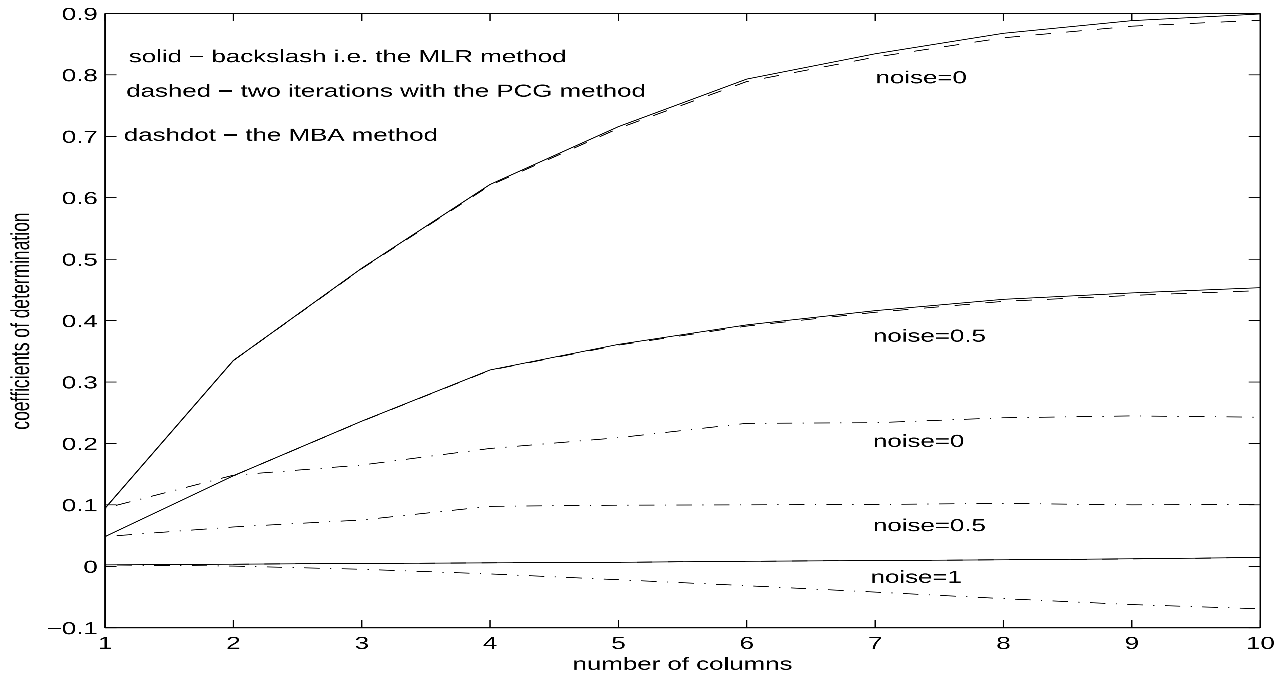

Figure 3: Coefficients of determination, R^2, for three different solution methods, 600 rows, 10 columns, various amount of noise, not centered data, standard deviation one

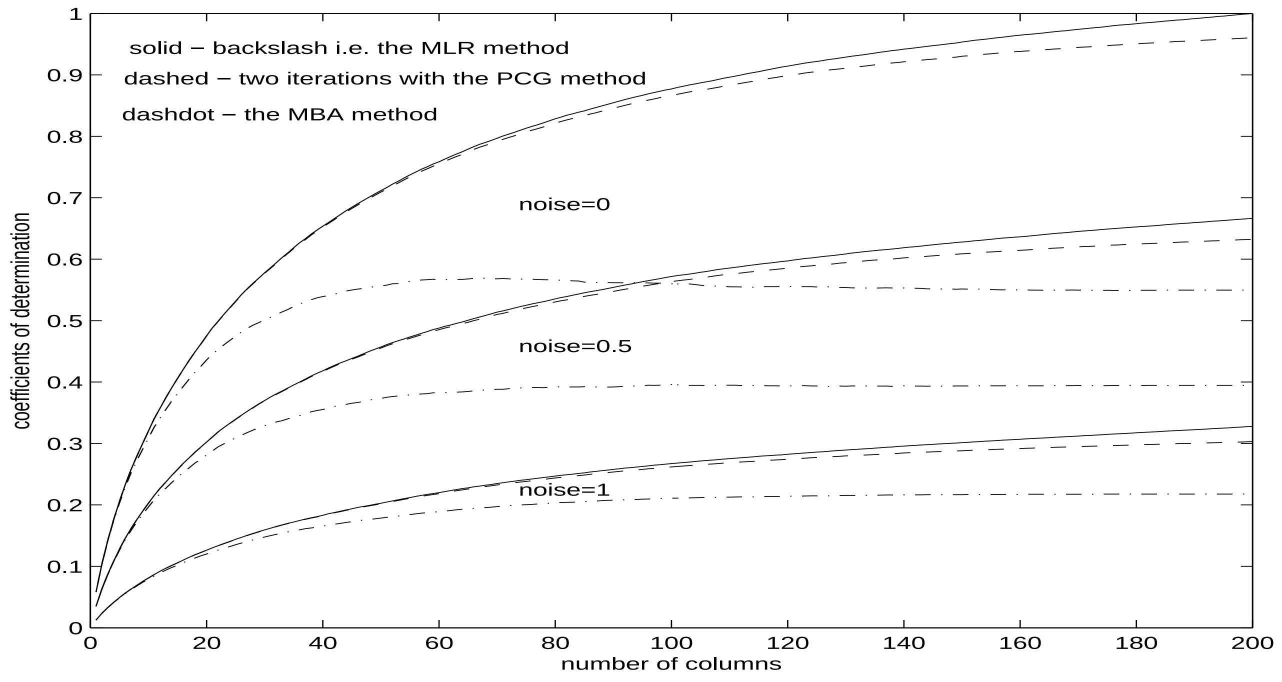

Figure 4: Coefficients of determination, R^2, for three different solution methods, 600 rows, 200 columns, various amount of noise; centered data, standard deviation one

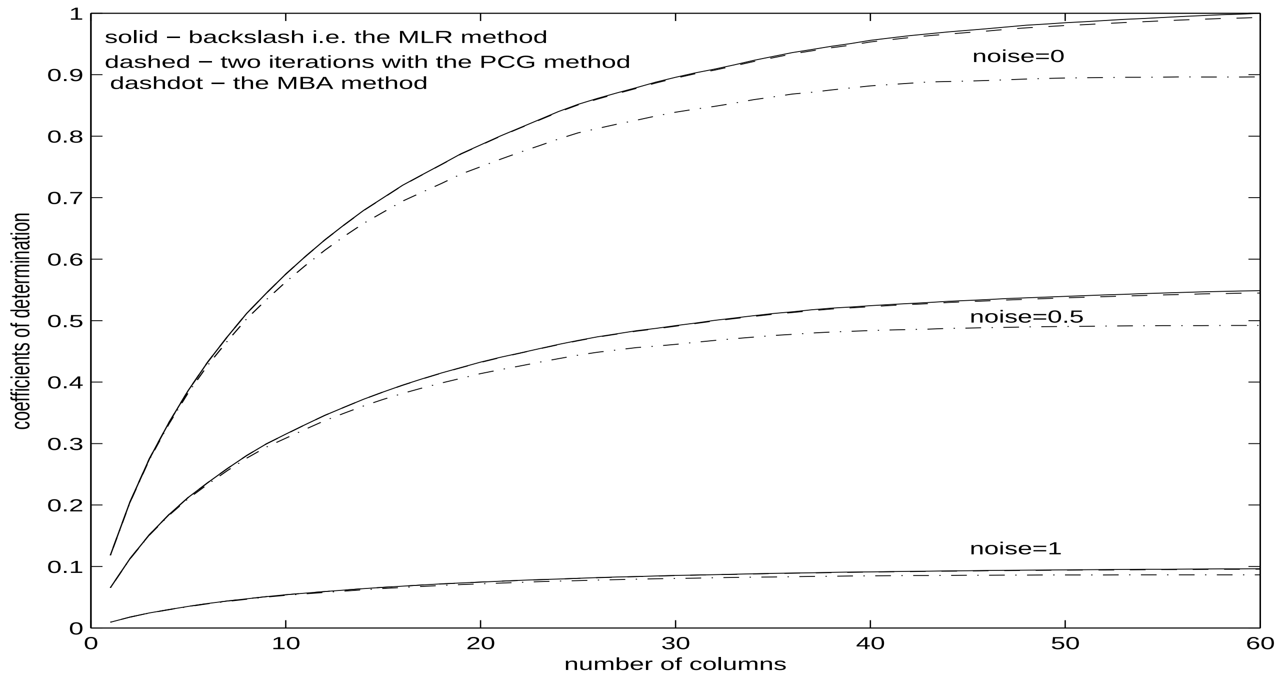

Figure 5: Coefficients of determination, R^2, for three different solution methods, 600 rows, 60 columns, various amount of noise, centered data, standard deviation one

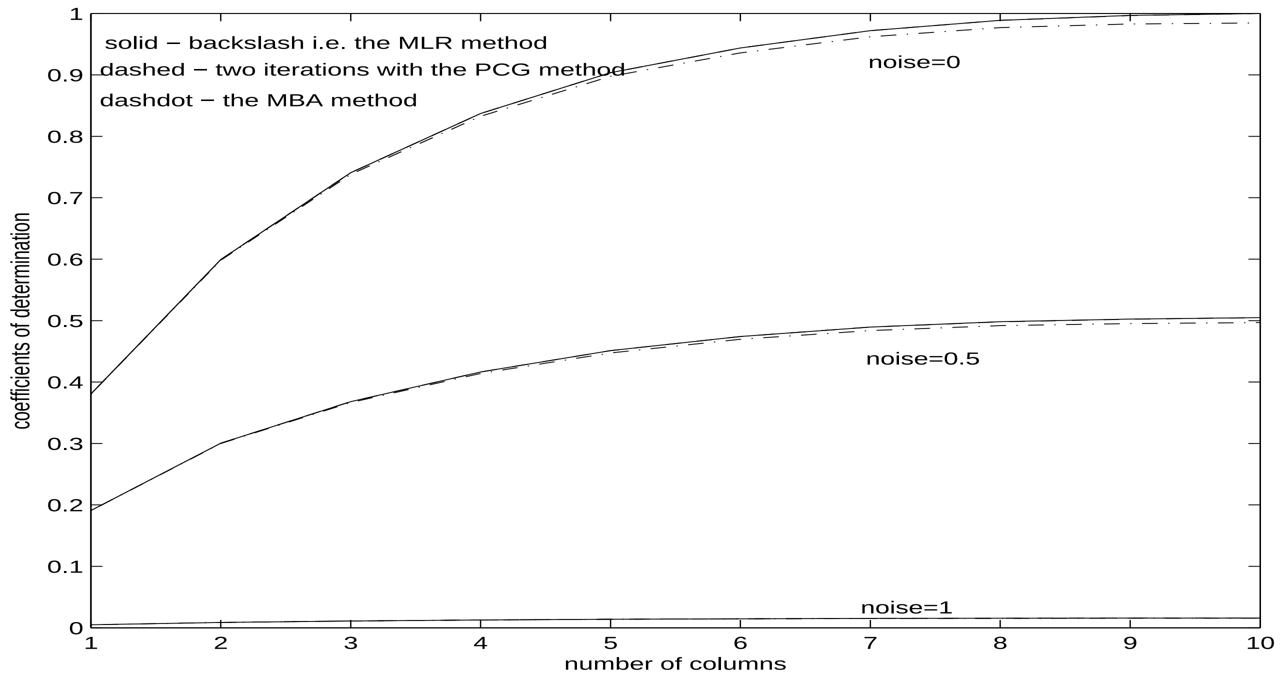

Figure 6: Coefficients of determination, R^2, for three different solution methods, 600 rows, 10 columns, various amount of noise, centered data, standard deviation one

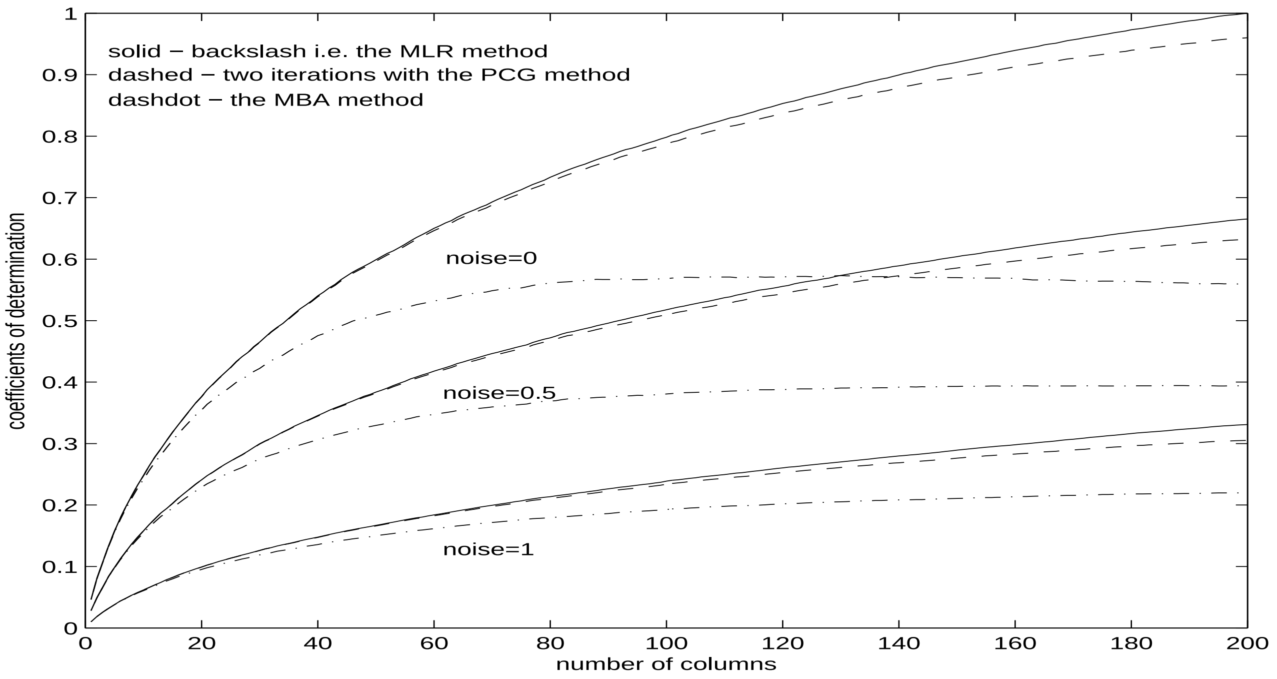

Figure 7: Coefficients of determination, R^2, for three different solution methods, 600 rows, 200 columns, various amount of noise, centered data, varying standard deviation

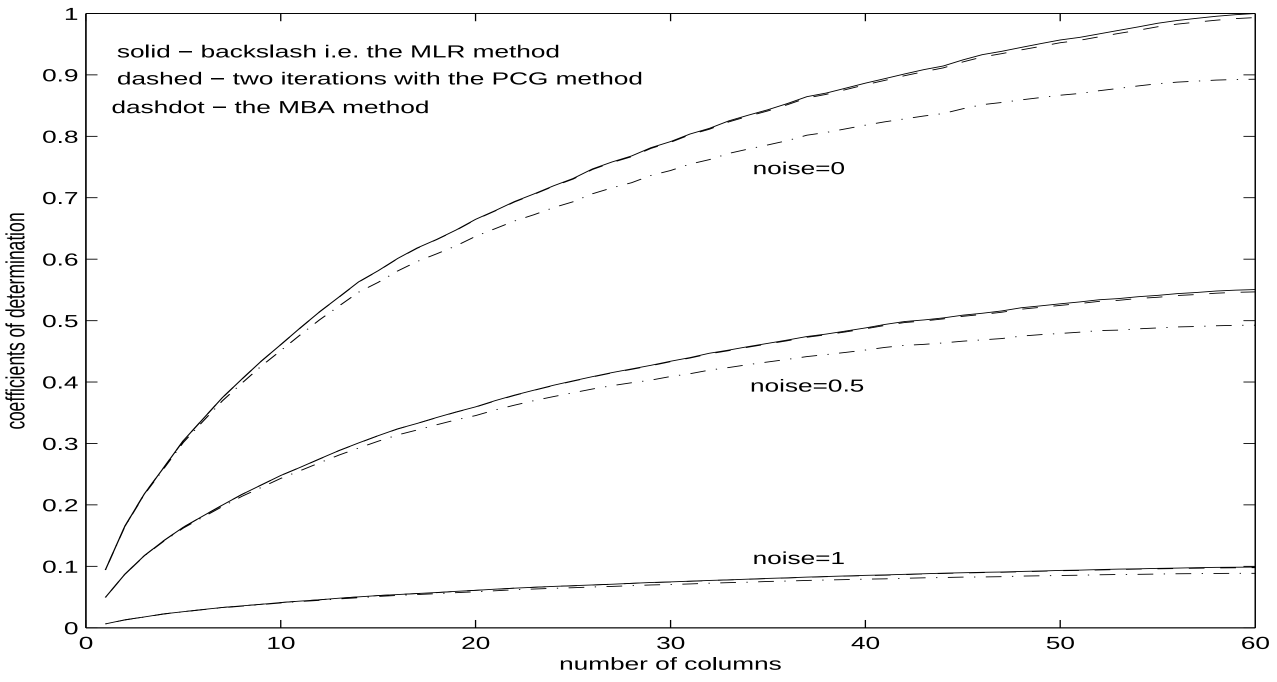

Figure 8: Coefficients of determination, R^2, for three different solution methods, 600 rows, 60 columns, various amount of noise, centered data, varying standard deviation

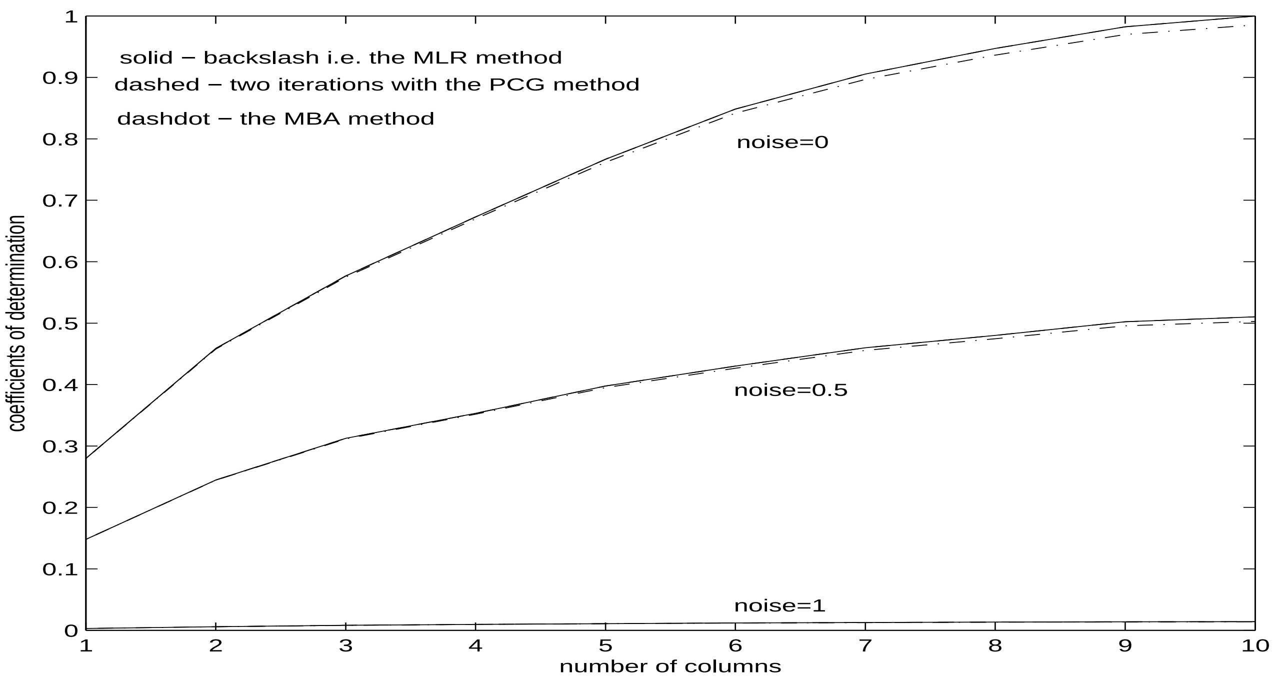

Figure 9: Coefficients of determination, R^2, for three different solution methods, 600 rows, 10 columns, various amount of noise, centered data, varying standard deviation

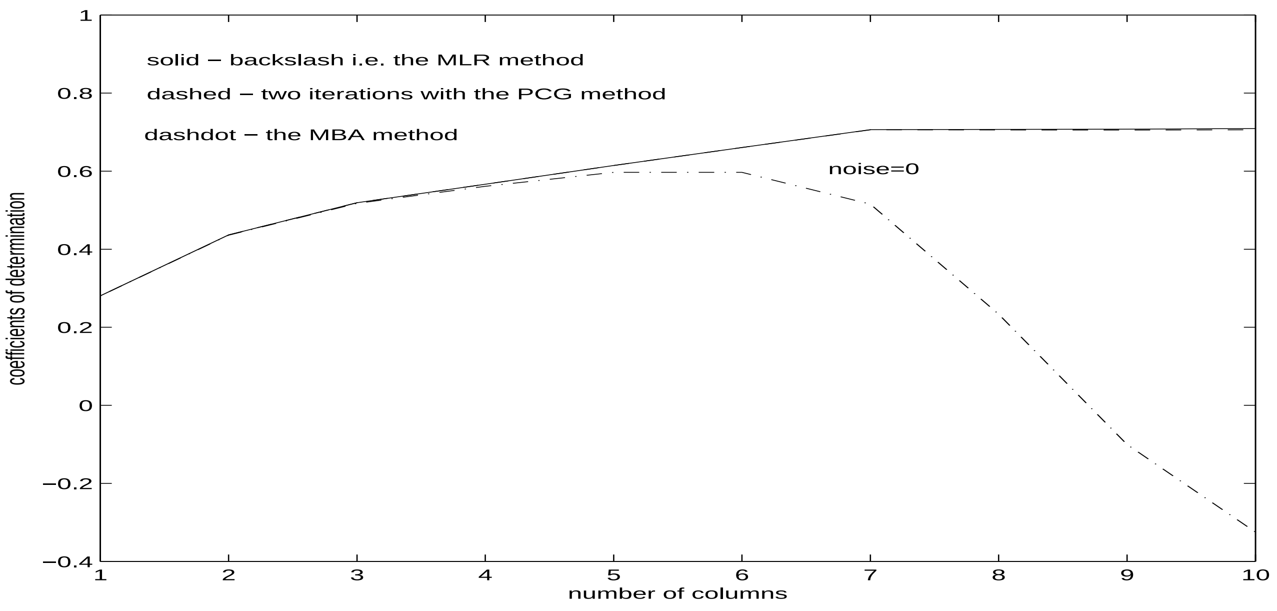

Figure 10: Coefficients of determination, R^2, for three different solution methods, 600 rows, 10 columns, without noise, centered data, varying standard deviation, almost rank defficient system, rank ≈ 7

5 Discussion and conclusion

At first we observe that in all cases, the MLR method, computed by backslash i.e QR-facorization, gives the largest coefficient of determination, R^2. However, for large matrices this method is too expensive, the computational cost, the number of floating points operations, is of order $2mn^2$ while the MBA method and the PCG method with two iterations requires approximately $4mn$ and $10mn$ operations, respectively.

From the experiments it becomes clear that the idea in Reference [1] of selecting the columns (variables) according to the size of the scalar product of each column and the righthand side is very powerful also when the fit ability, the value of the coefficient of determination, R^2, is conserned.

From Figure 1 up to Figure 3, it can be seen that the MBA method is no good for not centered data (Case 1). The PCG method with two iterations, however, comes very close to the MLR method, in particular for strongly overdetermined systems.

By Figure 4 up to Figure 6, it becomes clear that for centered data, standard deviation one, and not very ill-conditioned systems (Case 2), the MBA method approaches the MLR method as the system becomes more overdetermined.

With respect to fit ability, the PCG method with two iterations is a good approximation to the MLA method in all cases studied. As for the solution of the overdetermined systems involved, the solutions obtained by these two methods are also very close, if the system has full rank.

It can be realised from Figure 7 up to Figure 9 that no change to speak of is observed if the columns have various standard deviations, Case 3 compared to Case 2, as long as the data is centered. The results of the methods look pretty much the same. Considering the number of columns needed for a certain level of the coefficients of determination, R^2, however, there is an observable difference. For instance, detailed studies of Figure 4 and Figure 7 unveil that for $n = 200$, about 140 columns are needed for $R^2 = 0.9$ in Case 3 (varying standard deviation), compared to about 100 columns in Case 2 (standard deviation one). This can be explaned by the fact that the condition number of the matrix in Case 3 is much larger than in Case 2. In the present example the (average) condition number is about 80 times larger in Case 3 (compared to Case 2). The MBA method completely breaks down for almost rank deficient systems (Case 4), as can be seen in Figure 10, where $n = 10$ and the rank is almost only 7. Remember that the MBA method could still be very good for selection of the most important variables, as pointed out in Reference [1]. The PCG method with two iterations, on the other hand, works very well in such situations, giving results very close to the MLA method for strongly overdetermined systems. Notice that in the present case, the condition number of the matrix is fairly large. For $n = 200$ the average condition number was about $1.7 \cdot 10^5$ and for $n = 10$, the result presented in Figure 10, the average condition number was about $5.9 \cdot 10^3$. Similar results are obtained in Case 5, when the system is exactly rank deficient with rank equal $n - 3$. The actual figure was almost identical to Figure 10 and is therefore omitted.

As a very nice side result, in the present investigations, it is observed that the PCG method gives the smallest norm solution to a rank deficient system (Case 5). This solution is actually the same as is given by the pseudoinverse method, see Reference [2]. Futhermore, the approximations computed by few iterations of the PCG method always have smaller norms than the true minimum norm solution (obtained after fully converged PCG method or by the pseudoinverse method). These latter observations motivate further theoretical studies regarding the PCG method for rank deficient problems.

6 Acknowledgement

The author thanks Sven Ahlinder at Volvo GTT Advance Technology for pointing out the very problem in Big Data and for helpful guidance during the development work contained in the current paper.

7 References

[1] Ahlinder, Gustafsson. A real life engineering problem in Big Data solved by Pearson's correlation coefficient.
[2] Demmel. Applied Numerical Linear Algebra, Society for Industrial and Applied Mathematics, 1997.
[3] Box, Hunter, Hunter. Statistics for Experimenters ISBN 0-417-09315-7 1978 John Wiley & Sons, Inc.