RAL-TR-96-013

A projection method for the solution of rectangular systems

Jesús Cardenal[†], Iain S. Duff[‡], and José M. Jiménez[§]

ABSTRACT

We present a general method for the linear least-squares solution of overdetermined and underdetermined systems. The method is particularly efficient when the coefficient matrix is quasi-square, that is when the number of rows and number of columns is almost the same. The numerical methods proposed in the literature for linear least-squares problems and minimum-norm solutions do not generally take account of this special characteristic. The proposed method is based on an **LU** factorization of the original quasi-square matrix **A**, assuming that **A** has full rank. In the overdetermined case, the **LU** factors are used to compute a basis for the null space of $\mathbf{A}^{\mathrm{T}}$. The right-hand side vector **b** is then projected onto this subspace and the least-squares solution is obtained from the solution of this reduced problem. In the case of underdetermined systems, the desired solution is again obtained through the solution of a reduced system. The use of this method may lead to important savings in computational time for both dense and sparse matrices. Some practical examples that illustrate the use of the method are included.

Keywords: multibody systems, Gaussian elimination, least-squares, overdetermined and underdetermined systems, sparsity, minimum-norm solution, reduced gradient method, null-space method.

AMS(MOS) subject classifications: 65F05, 65F50, 65F20.

[†] University of La Coruña, Mendizábal, s/n. Esteiro, 15403 Ferrol (La Coruña), Spain.
Phone: + 34 81 354130 Fax: +34 81 354463 e-mail: jcarde@cdf.udc.es
[‡] Also at CERFACS, 42 Ave G Coriolis, 31057 Toulouse Cedex, France.
Phone: +44 1235 445803 Fax: +44 1235 446626 e-mail: isd@letterbox.rl.ac.uk
[§] CEIT and University of Navarre, P. Manuel de Lardizábal, 15, 20009 San Sebastián, Spain.
Phone: +34 43 212800 Fax: +34 43 213076 e-mail: jmjimenez@ceit.es

Current reports available by anonymous ftp from seamus.cc.rl.ac.uk (internet 130.246.8.32) in the directory "pub/reports". This report is in file cadjRAL96013.ps.gz.

Computing and Information Systems Department
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX
February 26, 1996.

Contents

1 Introduction.

We consider the solution of the linear least-squares problem

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 , \tag{1.1}$$

and of the problem

$$\min \|\mathbf{x}\|_2 \quad \text{subject to} \quad \mathbf{A}^{\mathrm{T}}\mathbf{x} = \mathbf{b}, \tag{1.2}$$

where $\mathbf{A}$ is an $m \times n$ matrix ($m \geq n$). In many engineering applications, $\mathbf{A}$ is sparse and is nearly square; that is the number of rows, m, is almost equal to the number of columns, n. We call such a system *quasi-square*. The numerical methods proposed in the literature for least-squares and minimum-norm problems do not generally take account of this special characteristic.

A similar problem arises when the solution of a set of nonlinear equations is sought. Most of the methods for finding a solution to nonlinear systems of equations are based on solving a sequence of linear systems of equations. The Jacobian is usually the coefficient matrix of these linear systems and may become rank deficient during the iterative process, for example if the current iteration point is far from the desired root. In general, we would hope that the linear equation solver will detect this rank deficiency, which is one reason why orthogonalization methods are so popular. However, in many cases including the application examined in this paper, it is sufficient to use Gaussian elimination techniques and rank deficiency is not a problem as the system becomes underdetermined but consistent. We show how the **LU** factorization of $\mathbf{A}$ can be used in this case also.

This paper presents a general method for the solution of both linear least-squares and minimum-norm problems. The motivation for this work comes from the study of a particular problem that arises in mechanical engineering, namely the computer simulation of the kinematics of multibody systems. Although we test our methods on this class of problems, it is important to point out that the proposed methods are completely general and may be successfully applied in many other fields.

Section 2 describes the most relevant characteristics of the computer simulation of the kinematic behaviour of multibody systems. The description includes a short review of the difficulties that arise when applying classical methods to solve linear least-squares and minimum-norm problems.

Section 3 derives the proposed method for solving the systems (1.1) and (1.2). Although the method is completely general, it is particularly well suited to quasi-square systems. This method is based on an **LU** factorization of the original sparse matrix $\mathbf{A}$. In the case of the least-squares problems, the $\mathbf{L}$ factor is used to construct a basis for the null space of $\mathbf{A}^{\mathrm{T}}$. The projection of the right-hand side vector $\mathbf{b}$ onto that subspace is computed and then the linear least-squares problem is solved directly. The minimum-norm solution (1.2) again uses the same **LU** factorization of the original sparse matrix $\mathbf{A}$. We again use the factor $\mathbf{L}$ to define a basis for the null space of $\mathbf{A}^{\mathrm{T}}$. Once this basis is evaluated, the minimum-norm solution of the underdetermined system is easily found. The method is known in optimization circles as a reduced-gradient or null-space method and systems which are quasi-square are encountered in the solution of constrained optimization problems using sequential quadratic programming methods (for example, Biegler, Nocedal & Schmidt (1995)). However, we explore the implementation and use of the **LU** factorization in greater detail and additionally indicate its use in the solution of the minimum-norm problem.

Section 4 presents a short description of problems on which our method is tested and compared. In Section 4.1, we compare our algorithm with implementations of well known

methods (normal equations and the method of Peters and Wilkinson (1970)) in the case when $\mathbf{A}$ is considered as a dense matrix. We compare the methods using theoretical counts of the number of arithmetic operations and compare their execution times on our test matrices using implementations based on LAPACK routines. Computation times required to solve the same test problems using routines for sparse matrices are given in Section 4.2. In this section, we compare normal equations, augmented systems, and the proposed method, implemented using different sparse solvers.

Finally, some concluding remarks are presented in Section 5.

2 Description of the problem.

As already mentioned, the motivation for this work is the study of the computer-aided simulation of the kinematic behaviour of multibody systems.

A *multibody system* is any mechanical system composed of several rigid or flexible parts or bodies linked by means of kinematic pairs or joints. Typical examples of such systems are robot arms, satellites, automobiles, or even the human body. They are also referred to as mechanisms.

The mathematical description of such a system usually leads to a set of nonlinear algebraic equations expressed in terms of the coordinates used to describe the mechanical system. These equations may be written as

$$\mathbf{\Phi}(\mathbf{q}) = \mathbf{0}, \tag{2.1}$$

where $\mathbf{q}$ is the vector of dependent coordinates that define the position of each body of the mechanism.

Very often the set of equations that describe the mechanical system includes more equations than needed; that is, there are more equations than unknowns. The reason for this may be either the particular kinematic structure of the mechanism being simulated or the way in which the computer model is obtained. However, equation (2.1) describes a real physical device and therefore a solution must exist. In other words, the set of nonlinear equations is consistent and has at least one solution that satisfies all the equations.

To solve the nonlinear problem given by equation (2.1), a Newton-Raphson iterative procedure of the form

$$\begin{aligned} \left(\mathbf{\Phi_q}\right)_i \Delta\mathbf{q}_i &= -\mathbf{\Phi}_i, \\ \mathbf{q}_{i+1} &= \mathbf{q}_i + \Delta\mathbf{q}_i, \end{aligned} \tag{2.2}$$

is performed until convergence is reached. This procedure must be repeated several times for each different position of the mechanical system during the simulation. Equation (2.2) is a linearization of equation (2.1). As we said, the system (2.1) may contain redundant equations but it is compatible for each particular configuration of the mechanical system. However, during the Newton-Raphson iteration, the vector of coordinates $\mathbf{q}_i$ does not satisfy equation (2.1) and the system (2.2) is overdetermined. We thus are obliged to solve (2.2) in the least-squares sense to enable us to continue with the iterative procedure. This may be expressed as

$$\min_{\Delta\mathbf{q}_i} \left\| \mathbf{\Phi}_i + \left(\mathbf{\Phi_q}\right)_i \Delta\mathbf{q}_i \right\|_2, \tag{2.3}$$

or, in the notation we have been using previously,

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 . \tag{2.4}$$

During the iterative procedure, the rank of the matrix usually equals the number of unknowns. However, in some cases, this matrix experiences a sudden loss of rank. Mathematically, this problem may arise, for example, when the current iteration point is far from the desired root. Physically, a singular Jacobian matrix indicates that the mechanical system has reached a "singular position". Park & Haug (1988) distinguished two different kinds of singularities: *lock-up configurations* and *bifurcation points*. In a lock-up configuration, the set of degrees of freedom does not define the motion of the mechanical system, which is in some sense "locked". On the contrary, a bifurcation point implies that, at least theoretically, the mechanism can proceed along different paths. To overcome this problem, different solutions have been proposed (for example, Nakamura & Hanafusa (1986), Angeles, Anderson, Cyril & Chen (1988)). The simplest method of preventing the loss of rank in the Jacobian causing a failure of the analysis consists in seeking a minimum-norm solution. This solution, $\overline{\mathbf{x}}$, must satisfy

$$\overline{\mathbf{x}} = \operatorname{argmin} \|\mathbf{x}\|_2 , \tag{2.5}$$

subject to the constraints given by equation (2.2). We note that, in this case, the physical system dictates that system (2.2) will be consistent and underdetermined.

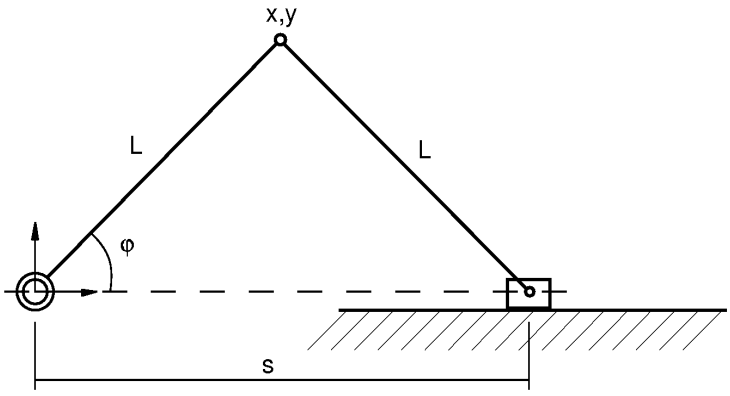

Figure 2.1: A slider-crank mechanism.

In order to clarify how equation (2.2) can represent both an overdetermined and an underdetermined linear system of equations, a very simple example is presented.

Figure 2.1 represents a slider-crank mechanism with one degree of freedom. Both the slider and the crank have the same length L. To define the position of the system, we use the Cartesian coordinates of point (x, y), the distance s between the slider and the origin O, and the angle φ. The system of constraint equations, derived from these links, is given by

$$\begin{aligned} x^2 + y^2 - L^2 &= 0 \\ (x - s)^2 + y^2 - L^2 &= 0 \\ x - L\cos\varphi &= 0 \\ y - L\sin\varphi &= 0 \\ x - f(t) &= 0 \end{aligned} \tag{2.6}$$

Therefore, we have 5 nonlinear equations relating 4 unknowns. The last equation in (2.6) drives the degree of freedom of the system. The third and fourth equations in (2.6) define the same constraint, hence one of them could be eliminated. However, it is common practice, when dealing with angles as dependent coordinates, to use both relations because the sine function measures every angle well except those near $\pi/2$ and $3\pi/2$, and the same problem arises with the cosine near 0 and π. The kinematic analysis of the system depicted in Figure 2.1 involves the solution of system (2.6) for each position. Most of the time our system is an overdetermined one, with the coefficient matrix

$$\mathbf{\Phi_q} = \begin{bmatrix} 2x & 2y & 0 & 0 \\ 2(x-s) & 2y & -2(x-s) & 0 \\ 1 & 0 & 0 & L\sin\varphi \\ 0 & 1 & 0 & -L\cos\varphi \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{2.7}$$

However, when the mechanism reaches the position shown in Figure 2.2, in which $x = 0$, $y = L$, $s = 0$ and $\varphi = \pi/2$, the Jacobian becomes

$$\mathbf{\Phi_q} = \begin{bmatrix} 0 & 2L & 0 & 0 \\ 0 & 2L & 0 & 0 \\ 1 & 0 & 0 & L \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \tag{2.8}$$

and is clearly rank deficient. Equations 1, 3, and 5 are consistent and so we effectively have a system of three equations with four unknowns. Thus, the kinematic analysis of one system can lead to both an overdetermined system or, less commonly, to an underdetermined one.

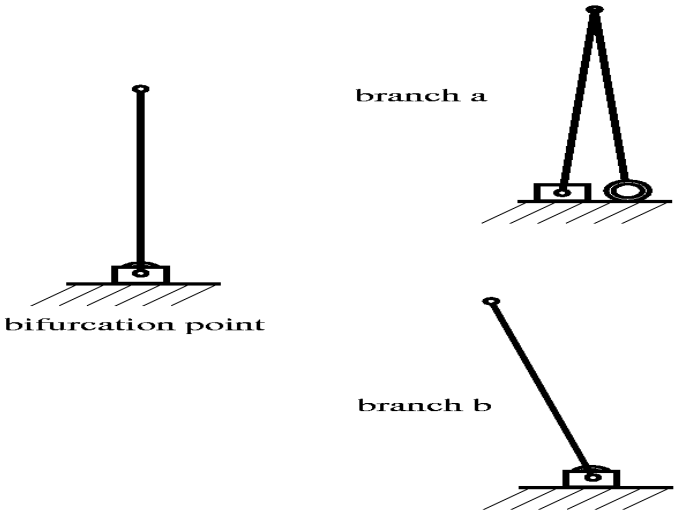

Figure 2.2: Singular position with a single bifurcation point.

A detailed description of the problems that arise in computer-aided simulation of multibody systems may be found in García de Jalón & Bayo (1994). In this work, the method developed for the solution of the linear least-squares system (2.4) or for the minimum-norm solution of equation (2.5) takes into account the particular characteristics of the kinematic simulation process and its numerical aspects.

Before presenting the general description of our proposed method, we give a brief explanation of the specific characteristics of the simulation process.

Designers and mechanical engineers require programs to assist them in the complex task of the kinematic design of multibody systems. These programs must provide interactive response, allowing the user to control interactively the degrees of freedom of the mechanism being analysed. Also, it is very convenient for these programs to include a graphic postprocessor that generates realistic images to show the evolution of the system on the computer screen. These requirements limit the computing platform to workstations with advanced 3D graphics capabilities. These computers are usually single processor machines, and hence parallel implementations are not of interest. Interactive response means that the program should provide between 5 and 15 frames per second. Each frame requires the solution of the set of nonlinear equations (2.1); that is, a linear least-squares problem must be solved several times for each frame. For those reasons, our aim is to look for the fastest method for the solution of systems (2.4) and (2.5). We also note that the solution is not required to high accuracy and numerical conditioning is not generally a problem in this application, so it is not necessary to choose the most robust numerical technique.

On the other hand, the characteristics of the matrix $\mathbf{A}$ in (2.4) and (2.5) must be considered. Typically, the kinematic simulation leads to small or medium sized quasi-square systems (see statistics on typical problems in Table 4.1). The number of rows is usually between 5% and 10% more than the number of columns. When $\mathbf{A}$ suffers a loss of rank, the underdetermined linear system (2.2) has only a few more unknowns than independent equations. In Section 4, Table 4.1 shows the characteristics of some matrices corresponding to typical multibody systems applications. These matrices are used in the numerical experiments both for least-squares and for minimum-norm solutions (in the latter case, the transpose of the matrices are used). The method selected for the solution of (2.4) or (2.5) must take into account the above mentioned requirements and characteristics. Because of the interactive response requirement and the selected computer platform, methods based on orthogonalization techniques are not of interest because of their high computational cost, despite their numerical stability (Duff & Reid 1976). It is, however, possible that sparse orthogonalization methods using multifrontal codes like those of Puglisi (1993) and Matstoms (1994) might be competitive although we have not considered them here.

Methods based on Gaussian elimination are examined. Usually, $\mathbf{A}$ is well conditioned and the normal equations method may be successfully applied. In addition, $\mathbf{A}$ is usually so small that the augmented matrix approach may be used without storage problems. This method is competitive with that of normal equations if sparse solvers are used (Duff & Reid 1976) and has better numerical behaviour. A third alternative that can be considered is the method of Peters & Wilkinson (1970). This method takes advantage of the sparsity of the original matrix $\mathbf{A}$ and also has good numerical behaviour, but it requires more computational effort than the normal equations method.

Tewarson & Jen (1988) proposed a method for the solution of full and rank deficient sets of linear equations that shows some similarities to the approach followed in the present paper. That method is based on $\mathbf{LU}$ elimination of the original matrix with scaled partial pivoting. In the rank deficient case, an orthogonal triangularization procedure is performed on the remaining columns of the matrix. Although this method could be extended to the case of overdetermined systems, the use of an orthogonalization technique makes it more expensive in terms of computational effort than the method proposed here.

The method of Peters and Wilkinson has a similar starting point to the method presented in this paper. Our method also takes advantage of the sparsity of the original

matrix $\mathbf{A}$ and performs a sparse $\mathbf{LU}$ factorization on it. Then, instead of using the $\mathbf{LU}$ factors to construct the reduced normal equations, it uses the $\mathbf{L}$ factor to build a basis for the null space of $\mathbf{A}^{\mathrm{T}}$ and looks for both the projection of the right-hand side $\mathbf{b}$ onto the subspace spanned by the columns of matrix $\mathbf{A}$ (which is the least-squares solution) and its projection onto the computed basis of the null space of $\mathbf{A}^{\mathrm{T}}$. As will be seen in the following sections, for quasi-square matrices, this method leads to important savings in terms of floating-point operations.

With respect to the practical implementation, we note that the size of the matrices involved in the computation is typically a few hundred rows and columns. For this reason, the use of efficient solvers for dense matrices may be considered as an alternative to sparse solvers, which are more complicated to implement and use. We show that, when dealing with these small matrices (like those appearing in multibody simulation problems), the solver implementation using LAPACK library routines (Anderson, Bai, Bischof, Demmel, Dongarra, DuCroz, Greenbaum, Hammarling, McKenney, Ostrouchov & Sorensen 1995), whose computational efficiency is based on the use of BLAS routines (Dongarra, Du Croz, Duff & Hammarling 1990), is not so efficient as the sparse solvers, even in the case of very small matrices. This point is investigated in this paper and some comparative results are included in Section 4.

3 Proposed method.

We now present a unified approach to describing our algorithm for solving both the overdetermined and the underdetermined case. An important aspect is that we wish to use the same $\mathbf{LU}$ factorization of $\mathbf{A}$ in each case.

For both the least-squares problem

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 , \tag{3.1}$$

or the minimum-norm solution to the underdetermined system

$$\min \|\mathbf{x}\|_2 \quad \text{subject to} \quad \mathbf{A}^{\mathrm{T}}\mathbf{x} = \mathbf{b}, \tag{3.2}$$

where $\mathbf{A}$ is an $m \times n$ matrix ($m \geq n$), we use the augmented system formulation

$$\begin{bmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} . \tag{3.3}$$

For the least-squares problem

$$\mathbf{c}_1 = \mathbf{b} \quad \text{and} \quad \mathbf{c}_2 = \mathbf{0},$$

and the solution is given by $\mathbf{y}_2$.

For the minimum-norm solution

$$\mathbf{c}_1 = \mathbf{0} \quad \text{and} \quad \mathbf{c}_2 = \mathbf{b},$$

and the solution is given by $\mathbf{y}_1$.

In both cases, we will use an $\mathbf{LU}$ factorization of $\mathbf{A}$, so that, if

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \end{bmatrix},$$

where $\mathbf{A}_1$ is an $n \times n$ nonsingular matrix. The factorization can be written

$$\mathbf{A} = \begin{bmatrix} \mathbf{L}_1 \\ \mathbf{L}_2 \end{bmatrix} \mathbf{U}, \tag{3.4}$$

where $\mathbf{A}_1 = \mathbf{L}_1\mathbf{U}$. Of course, it is the $\mathbf{LU}$ factorization of $\mathbf{A}$ that determines the partitioning. When performing this factorization we will use standard threshold pivoting (Duff, Erisman & Reid 1986) to limit growth in the factors of $\mathbf{L}$. Note that it is actually a row permutation of the matrix $\mathbf{A}$ that is factorized but we have omitted the permutation matrix from (3.4) for the sake of clarity.

If we partition the coefficient matrix of the augmented system (3.3) accordingly, we get the matrix

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{A}_1 \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_2 \\ \mathbf{A}_1^{\mathrm{T}} & \mathbf{A}_2^{\mathrm{T}} & \mathbf{0} \end{bmatrix}. \tag{3.5}$$

In both overdetermined and underdetermined cases, we will solve the problem using the block factorization of a permutation of (3.5) given by

$$\begin{bmatrix} \mathbf{A}_1^{\mathrm{T}} & \mathbf{0} & \mathbf{A}_2^{\mathrm{T}} \\ \mathbf{I} & \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1^{\mathrm{T}} & & \\ \mathbf{I} & \mathbf{I} & \\ & \mathbf{J} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & & \mathbf{J}^{\mathrm{T}} \\ & \mathbf{A}_1 & -\mathbf{J}^{\mathrm{T}} \\ & & \mathbf{S} \end{bmatrix}, \tag{3.6}$$

where $\mathbf{J} = \mathbf{A}_2\mathbf{A}_1^{-1} = \mathbf{L}_2\mathbf{L}_1^{-1}$ and $\mathbf{S} = \mathbf{I} + \mathbf{J}\mathbf{J}^{\mathrm{T}}$.

We show, in Sections 3.1 and 3.2, how this factorization can be used to solve the systems (3.1) and (3.2) respectively.

3.1 Overdetermined systems.

In the case of the least-squares problem (3.1), if we partition and reorder the system according to the matrices in (3.6) we obtain

$$\begin{bmatrix} \mathbf{A}_1^{\mathrm{T}} & \mathbf{0} & \mathbf{A}_2^{\mathrm{T}} \\ \mathbf{I} & \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{x} \\ \mathbf{r}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix},$$

then the use of the factorization (3.6) implies that we first solve the system

$$\mathbf{S}\mathbf{r}_2 = \mathbf{b}_2 - \mathbf{J}\mathbf{b}_1, \tag{3.7}$$

with $\mathbf{S}$ and $\mathbf{J}$ defined as above.

The solution to the least squares-problem is then found by using the factors of $\mathbf{A}_1$ to solve the equation

$$\mathbf{A}_1\mathbf{x} = \mathbf{b}_1 + \mathbf{J}^{\mathrm{T}}\mathbf{r}_2. \tag{3.8}$$

Thus we can solve the least-squares problem by solving a reduced problem of dimension $m-n$ (which we have assumed small) and a set of linear equations of order n for which we have computed the $\mathbf{LU}$ factorization. Furthermore, we can control the stability through pivoting in the factorization (3.4).

We note that the proposed method has the same first step as the method of Peters and Wilkinson, namely the $\mathbf{LU}$ factorization of the original sparse matrix $\mathbf{A}$. Both avoid some of the problems of ill-conditioning by pivoting in the factorization (3.4) to keep entries in $\begin{bmatrix} \mathbf{L}_1 \\ \mathbf{L}_2 \end{bmatrix}$ small.

3.2 Underdetermined systems.

In the case of the minimum-norm problem (3.2), after partitioning and permuting the system becomes

$$\begin{bmatrix} \mathbf{A}_1^{\mathrm{T}} & \mathbf{0} & \mathbf{A}_2^{\mathrm{T}} \\ \mathbf{I} & \mathbf{A}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{r} \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

Then the use of the factorization (3.6) results in the solution of the system

$$\mathbf{S}\mathbf{x}_2 = \mathbf{J}(\mathbf{A}_1^{-\mathrm{T}}\mathbf{b}), \tag{3.9}$$

with $\mathbf{S}$ and $\mathbf{J}$ defined as above.

The solution to the underdetermined system is then given by

$$\mathbf{x}_1 = \mathbf{A}_1^{-\mathrm{T}}\mathbf{b} - \mathbf{J}^{\mathrm{T}}\mathbf{x}_2. \tag{3.10}$$

Thus we can solve the underdetermined system by solving a reduced problem of dimension $m - n$ (which we have assumed small) and a set of linear equations of order n for which we have computed the $\mathbf{LU}$ factorization. Furthermore, we can again control the stability through pivoting in the factorization (3.4).

4 Comparative results.

This section compares the numerical efficiency of the proposed method with other well known least-squares methods based on Gaussian elimination. This comparison is performed at both theoretical and experimental levels. Theoretical comparisons are performed using floating-point operation counts for dense matrices. These results are supported by the CPU times required for the solution of some test problems, both considering the matrices as dense and sparse. In the case of dense matrix computations, these practical implementations have been carried out using the LAPACK library (Anderson et al. 1995). The implementation for sparse matrix computations has been carried out using several sparse matrix factorization routines from the Harwell Subroutine Library (Anon 1993).

Every test matrix considered in this paper comes from real applications of the kinematic simulation of multibody systems; all of them are Jacobian matrices corresponding to different real mechanical systems. These test problems are briefly introduced in Table 4.1, which summarizes their main characteristics. As can be seen, they are all small sparse quasi-square matrices. Matrix Test 1 is the Jacobian matrix corresponding to a six degrees of freedom Stewart platform. This kind of device is used, for example, to construct flight simulators. Matrix Test 2 represents the complete kinematic model of a car, including both front and rear suspension as well as the steering system. Matrix Test 3 corresponds to a complex kinematic model of the human body with up to 45 degrees of freedom. Matrix Test 4 comes from the kinematic model of a deployable antenna, and matrix Test 5 represents the kinematic model of the same antenna and the satellite to which it is attached.

The results presented in this section will indicate the most suitable method for the kind of problems considered in this paper. In addition, they will also show that, even for small matrices, like those arising in multibody simulation, the use of sparse solvers leads to important savings in computation times.

Table 4.1: Characteristics of test matrices.

	Name	# rows	# cols	# entries	Ratio m/n
6 dof. platform	Test 1	117	105	615	1.11
Car model	Test 2	170	161	1069	1.06
Human body model	Test 3	181	180	1200	1.01
Deployable antenna	Test 4	385	361	1661	1.07
Antenna + satellite	Test 5	574	526	3365	1.09

4.1 Dense matrices

This section compares the normal equations method, the method of Peter and Wilkinson, and the proposed method. The number of floating-point operations needed to obtain the solution is evaluated for every step of each algorithm. In this calculation, it is assumed that the coefficient matrix is dense. It is also assumed that additions and multiplications are equivalent single floating-point operations in terms of CPU requirements and only the higher order terms are included. The operation counts are obtained for the solution of overdetermined systems. Results for the case of underdetermined systems may be obtained following a similar procedure and are not included here.

For the normal equations method, the steps performed by the algorithm can be written as shown in Table 4.2.

Table 4.2: Normal equations method.

Step	Matrix computation	Computational cost
1) Products	$\mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{b}$	mn^2
2) Factorization	$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}$	$n^3/3$
3) Solve	$\mathbf{L}\mathbf{D}\mathbf{L}^{\mathrm{T}}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$	$2n^2$

The method of Peters and Wilkinson is slightly more complicated. This method first requires the $\mathbf{LU}$ factorization of the original matrix $\mathbf{A}$. Afterwards, the normal equations are constructed and then the resulting set of linear equations is solved. Table 4.3 shows the sequence of operations required by this method.

Table 4.3: Method of Peters and Wilkinson.

Step	Matrix computation	Computational cost
1) Factorization	$\mathbf{A} = \mathbf{L}\mathbf{U}$	$mn^2 - n^3/3$
2) Products	$\mathbf{L}^{\mathrm{T}}\mathbf{L}$ and $\mathbf{L}^{\mathrm{T}}\mathbf{b}$	$mn^2 - 2n^3/3$
3) Factorization	$\mathbf{L}^{\mathrm{T}}\mathbf{L} = \tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^{\mathrm{T}}$	$n^3/3$
4) Solve	$\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^{\mathrm{T}}\mathbf{U}\mathbf{x} = \mathbf{L}^{\mathrm{T}}\mathbf{b}$	$3n^2$

The method proposed in this paper again begins with the factorization of the original matrix $\mathbf{A}$. Once the $\mathbf{LU}$ factors have been computed, the matrix $\mathbf{J}$ has to be obtained. The evaluation of such a matrix only requires a few forward reductions and back substitutions using the blocks $\mathbf{L}_1$ and $\mathbf{L}_2$ of the factor $\mathbf{L}$. The number of right-hand side vectors to be solved equals the number of redundant rows in the original matrix $\mathbf{A}$; that is, $(m - n)$.

Having calculated the matrix $\mathbf{J}$, the next step requires the solution of equations (3.7) and (3.8). In equation (3.7) the matrix $\mathbf{I} + \mathbf{J}\mathbf{J}^{\mathrm{T}}$ is calculated and factorized. The algorithm corresponding to the proposed method may be summarized in five steps, as shown in Table 4.4.

Table 4.4: Proposed method.

Step	Matrix computation	Computational cost
1) Factorization	$\mathbf{A} = \mathbf{L}\mathbf{U}$	$mn^2 - n^3/3$
2) Solve	$\mathbf{J} = \mathbf{L}_2\mathbf{L}_1^{-1}$	$(m-n)n^2$
3) Products	$\mathbf{S} = (\mathbf{I} + \mathbf{J}\mathbf{J}^{\mathrm{T}})$ and $\mathbf{J}\mathbf{b}_1$	$(m-n)^2 n$
4) Solve	$\mathbf{S}\mathbf{r}_2 = \mathbf{b}_2 - \mathbf{J}\mathbf{b}_1$	$(m-n)^3/3$
5) Solve	$\mathbf{L}_1\mathbf{U}\mathbf{x} = \mathbf{b}_1 + \mathbf{J}^{\mathrm{T}}\mathbf{r}_2$	$2mn$

Table 4.5 summarizes the number of floating-point operations for the three different methods. From this table, we see that, if the difference $(m-n)$ is small, the last term of the computational cost of the proposed method may be neglected. In that case, the number of arithmetic operations required by our method is even lower than the number required by normal equations.

Table 4.5: Total number of floating-point operations.

Method	Computational cost
Normal equations	$mn^2 + n^3/3$
Peters and Wilkinson	$2mn^2 - 2n^3/3$
Proposed method	$mn^2 - n^3/3 + (m-n)(mn + (m-n)^2/3)$

All these algorithms are compared using the expressions for the number of floating-point operations given in Tables 4.2 to 4.4. Those comparisons are shown graphically in Figure 4.3 for different combinations of m and n. The method of the normal equations has been taken as a reference. Thus each plot shows the ratio between the computational effort required by either the method of Peters and Wilkinson or the method proposed in this paper and the number of operations required by normal equations. These ratios are shown for different values of the ratio m/n where m is the number of rows and n the number of columns of the original matrix $\mathbf{A}$. It may be seen that the number of operations required for the proposed method, when $m/n = 1.05, 1.10, 1.15$, is less than that required for the normal equations method, while the method of Peters and Wilkinson is more expensive than normal equations in every case considered.

Table 4.6: Least-squares. CPU time in milliseconds using LAPACK routines.

	Code	*Test 1*	*Test 2*	*Test 3*	*Test 4*	*Test 5*
Normal equations	LAPACK	57.41	203.26	238.61	1914.05	5990.06
Proposed Method	LAPACK	29.83	47.71	56.36	391.71	2124.20

From the graphs in Figure 4.3, it can be concluded that an implementation of the proposed method should be even faster than normal equations for the case of dense quasi-

square matrices. Also, from these graphical results, it is clear that the method of Peters and Wilkinson will not be competitive with normal equations nor with the proposed method. For this reason, we have not implemented the method of Peters and Wilkinson.

For dense matrices, both the proposed and the normal equations methods have been implemented using LAPACK routines. Table 4.6 shows the CPU times needed to solve the test problems that were summarized in Table 4.1. Times are in milliseconds and have been obtained using a single processor of a Silicon Graphics workstation (an ONYX with a MIPS R4400 processor running at 150 MHz). It is seen how the proposed method leads to significant savings in CPU times as forecast by the operation count study.

The implementation of these methods follows essentially the guidelines indicated in Tables 4.2 and 4.4, respectively. The times reported in Table 4.6 are the total CPU times required to perform all the operations needed for each particular method.

In Table 4.7, we show the CPU times required to solve the same test problems when the minimum-norm solution is required. As we see, the results in this case are very similar to those for the least-squares problem.

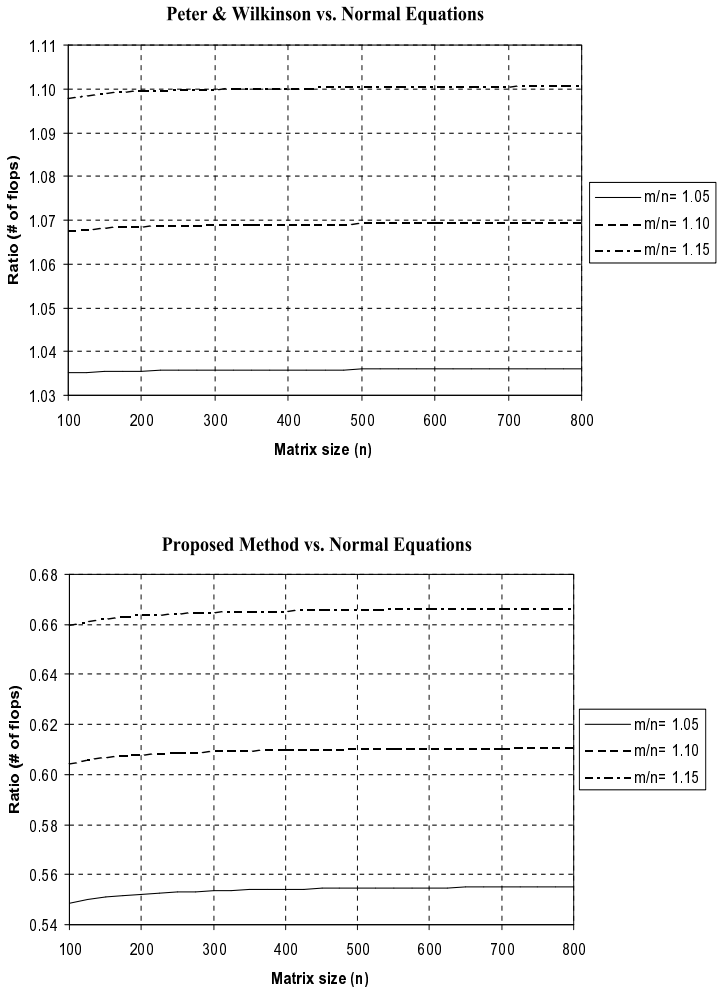

Figure 4.3: Comparison between the number of operations required in various methods.

Table 4.7: Minimum-norm. CPU time in milliseconds using LAPACK routines.

	Code	*Test 1*	*Test 2*	*Test 3*	*Test 4*	*Test 5*
Normal equations	LAPACK	61.24	195.07	252.17	2045.03	6413.86
Proposed Method	LAPACK	29.79	49.40	58.13	385.31	2120.17

4.2 Sparse matrices.

In the previous section, a theoretical and practical comparison between three different methods for dense matrices was presented. In this section, different implementations of the proposed method, the method of normal equations, and the augmented systems method will be compared in terms of CPU time to solve the same test problems. All the implementations in this section are based on Harwell Subroutine Library routines for sparse matrices.

Two different implementations have been carried out for the normal equations method using MA27 and MA47 routines; the augmented system method has been implemented using MA27, MA47 and MA48 routines and, finally, the MA48 package has been modified to implement the method described in this paper. CPU times measured when solving the problems in Table 4.1 can be seen in Table 4.8.

Table 4.8: Least-squares. CPU time in milliseconds using HARWELL routines.

	Code	*Test 1*	*Test 2*	*Test 3*	*Test 4*	*Test 5*
Normal equations	MA27	12.13	32.31	36.99	100.25	288.55
	MA47	15.29	37.85	42.03	117.00	317.12
Augmented system	MA27	7.31	15.59	13.13	43.44	120.70
	MA47	38.98	29.15	26.93	81.33	1961.45
	MA48	6.59	10.09	7.07	17.36	58.17
Proposed method	MA48	4.53	7.25	4.09	21.07	67.65

When solving nonlinear systems of equations, like those appearing in the kinematic simulation of multibody systems, the pattern of the coefficient matrix remains unchanged during the whole iterative process. Therefore the symbolic factorization of the matrix needs to be performed only once. For this reason, the times in Table 4.8 are those corresponding to the numerical solution for a single iteration, assuming that the symbolic factorization has been obtained previously.

Looking at the results in Table 4.8, the improvement obtained by those methods whose implementation is based on the MA48 package is very significant. In the case of the augmented systems, this implementation is more efficient than that based on the MA27 or MA47 routines, although those routines deal with symmetric problems while MA48 was designed for general unsymmetric rectangular matrices. One of the reasons for this *unexpected* behaviour may be the way the matrix is reordered. The MA48 package reorders the augmented matrix to a block triangular form (BTF). This appears to be a good strategy for matrices arising in multibody simulations.

We see that the sparse codes comfortably outperform LAPACK codes even on the small problems. The last row in Table 4.8 shows that the proposed method is the most efficient when dealing with small matrices. When matrices are moderate in size (Tests

4 and 5), the profiling analysis of the solution process indicates that most of the time is spent in computing the matrix $\mathbf{J}$. In those cases, this computation takes about 70% of the time required to solve the whole problem, while the numerical factorization of the original rectangular matrix takes only about 25% of the total time. Consequently, special attention must be paid to the optimization of the forward reduction and back substitution routines required to evaluate the matrix $\mathbf{J}$. This is thus the key step for improving the performance of the implementation, especially if the matrix is large. A BTF reordering strategy for rectangular matrices would lead to a faster execution time for the proposed method (this feature is not available in the current implementation of the method based on the MA48 package). The remaining steps, like the computation and factorization of the matrix $\mathbf{I} + \mathbf{J}\mathbf{J}^{\mathrm{T}}$, are not very expensive. Remember that the order of matrix $\mathbf{J}$ is $(m-n) \times n$ and can be treated as sparse. Moreover, $\mathbf{I} + \mathbf{J}\mathbf{J}^{\mathrm{T}}$ is an $(m-n)$ square, symmetric and positive definite matrix. Its factorization using LAPACK routines requires low computational effort, provided the original matrix $\mathbf{A}$ is quasi-square.

Table 4.9: Minimum-norm. CPU time in milliseconds using HARWELL routines.

	Code	*Test 1*	*Test 2*	*Test 3*	*Test 4*	*Test 5*
Normal equations	MA27	16.18	42.43	49.20	136.15	398.82
	MA47	19.25	47.93	54.00	152.07	427.23
Augmented system	MA27	7.45	16.09	13.40	43.73	120.64
	MA47	39.03	29.31	26.02	80.28	1700.56
	MA48	6.76	10.71	7.65	17.82	60.12
Proposed method	MA48	4.53	7.22	4.11	20.86	66.61

Table 4.9 shows the CPU times required to solve the same test problems using sparse solvers when the minimum-norm solution is required. Comments made for the least-squares problem are also valid in this case.

It is important to point out that problems arising in the kinematic simulation of multibody systems rarely involve more than two hundred equations in two hundred unknowns. Bigger cases, like those corresponding to test 4 and 5 are not at all frequent in practical applications. Therefore, the proposed method is a very attractive alternative to classical ones.

5 Conclusions.

In this paper we have presented a new method for the solution of linear least-squares and minimum-norm problems. The method is general and can be applied successfully to problems arising in different fields. Moreover, it has been shown that it is especially well suited for problems where the coefficient matrix is nearly square, like those appearing in the field of the kinematic simulation of multibody systems.

The proposed method for least-squares solution belongs to the category of elimination methods. Its numerical efficiency has been compared to the efficiency of other well known elimination methods for both dense and sparse matrices. Some comparative results in terms of CPU times required for the solution of practical test cases have been presented.

The method is also particularly well suited for problems where a sudden loss of rank of the original coefficient matrix can appear. In these cases it is not necessary to solve the problem from scratch. The initial factorization can be used to obtain the desired solution.

As a final remark, it should be noted that the method developed satisfies the constraints imposed by the kinematics of multibody systems. This method may solve the set of nonlinear equations that describes the mechanism many times per second. Therefore, the simulation program will give an interactive response even if the computer platform is a low cost workstation.

Acknowledgments.

The partial support of this work by CERFACS (RECITE programme) is gratefully acknowledged. The authors would like to thank Nick Gould of Rutherford Appleton Laboratory for his helpful remarks on an earlier draft and Mike Saunders of Stanford University for his comments on the text and for suggesting the matrix factorization (3.6). Their comments helped shorten the description of our method.

References

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., DuCroz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. & Sorensen, D. (1995), *LAPACK Users' Guide, second edition*, SIAM Press.

Angeles, J., Anderson, K., Cyril, X. & Chen, B. (1988), 'Inverse kinematic solutions with singularity robustness for robot manipulator control', *Journal of Dynamic Systems, Measurement and Control* **110**, 246–254.

Anon (1993), *Harwell Subroutine Library. A Catalogue of Subroutines (Release 11)*, Theoretical Studies Department, AEA Industrial Technology.

Biegler, L. T., Nocedal, J. & Schmidt, C. (1995), 'A reduced Hessian method for large-scale constrained optimization', *SIAM J. Optimization* **5**(2), 314–347.

Dongarra, J. J., Du Croz, J., Duff, I. S. & Hammarling, S. (1990), 'A set of Level 3 Basic Linear Algebra Subprograms.', *ACM Transactions on Mathematical Software* **16**, 1–17.

Duff, I. S. & Reid, J. K. (1976), 'A comparison of some methods for the solution of sparse overdetermined systems of linear equations', *J. Inst. Maths. Applics.* **17**, 267–280.

Duff, I. S., Erisman, A. M. & Reid, J. K. (1986), *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, England.

García de Jalón, J. & Bayo, E. (1994), *Kinematic and Dynamic Simulation of Multibody Systems. The Real Time Challenge*, Springer-Verlag.

Matstoms, P. (1994), **QR** Factorization with Applications to Linear Least Squares Problems, PhD thesis, Linköping University.

Nakamura, Y. & Hanafusa, H. (1986), 'Inverse kinematic solutions with singularity robustness for robot manipulator control', *Journal of Dynamic Systems, Measurement and Control* **108**, 163–171.

Park, T. & Haug, E. J. (1988), 'Ill-conditioned equations in kinematics and dynamics of machines sparsity patterns', *Int Journal of Numerical Methods in Engineering* **26**, 217–230.

Peters, G. & Wilkinson, J. H. (1970), 'The least-squares problem and pseudoinverses', *Computer J.* **13**, 309–316.

Puglisi, C. (1993), **QR** Factorization of Large Sparse Overdetermined and Square Matrices using the Multifrontal Method in a Multiprocessor Environment, PhD thesis, CERFACS. Technical Report TH/PA/93/33.

Tewarson, R. P. & Jen, J. F. (1988), 'Solution of full and rank deficient linear equations', *Communications in Applied Numerical Methods* **4**, 255–261.