# NOTE 224 – AIPS++ Least Squares background

Wim Brouw

22 January 1999

## Contents



## 1 Introduction

Trying to find a more stable non-linear least squares method (*LSQ*) than the one currently used in Newstar, I found another few areas that could be added or improved in the existing routines. In the end I produced a set of routines to handle linear and non-linear cases, both with or without SVD, and constraint equations.

In addition the present document describes the usage and background of the different routines.

Least squares solutions in Aips++ have the following general background:

1. all are based on creating normal equations from an, initially unknown, number of condition equations
2. the actual *LSQ*-object is a symmetric (or Hermitian) matrix
3. matrix inversions are done using in-place Cholesky methods where possible: it is for standard use the fastest, least memory consuming and a stable method. In cases where the inverse matrix is needed, the method uses resources of the same order as full LU-decomposition (only if explicit constraint equations are present)
4. input/output to the *LSQ*-object is done in any precision, internally all calculations are done in double precision
5. complex numbers are assumed to be contiguous pairs of real numbers with the real part being the first

The following terminology is used throughout:

$n$ number of unknowns to be solved

$m$ number of simultaneous knowns

$N$ number of condition equations

$\chi^2$ merit function

$z^*$ complex conjugate of $z$

$z_\Re$ real part of $z$

$z_\Im$ imaginary part of $z$

$\mathbf{x}$ column vector of unknowns $x_0, \ldots, x_{n-1}$

$\mathbf{a}_i$ vector of factors $a_{0,i}, \ldots, a_{n-1,i}$ in condition equation $i$ ( $i = 0, \ldots, N-1$ )

$\mathbf{C}$ the $N \times n$ array of condition equations

$\mathbf{A}$ $n \times n$ array: normal equations matrix

$\mathbf{L}$ $n$ column vector: right-hand side normal equations

$y_i$ model value for condition equation $i$ ( $i = 0, \ldots, N-1$ )

$l_i$ measured value for condition equation $i$ ( $i = 0, \ldots, N-1$ )

$\sigma_i$ standard deviation for condition equation $i$ ( $i = 0, \ldots, N-1$ )

$w_i$ weight for condition equation $i$ ($w_i = 1/\sigma_i^2$)

$[ab]$ shorthand for $\sum_{i=0}^{N-1} w_i a_i b_i$

The condition equations can, with the above definitions, be written in one of the following ways:

$$\mathbf{a}_i \cdot \mathbf{x} = l_i \qquad\qquad i = 0, \ldots, N-1 \tag{1}$$

or as:

$$\sum_{k=0}^{n-1} a_{ik} x_k = l_i \qquad\qquad i = 0, \ldots, N-1 \tag{2}$$

or as:

$$\mathbf{C} \cdot \mathbf{x} = \mathbf{l} \tag{3}$$

The normal matrix is defined as:

$$\mathbf{A} = \mathbf{C}^T \cdot \mathbf{Q}^{-1} \cdot \mathbf{C} \tag{4}$$

resulting in the normal equations:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{L} \tag{5}$$

where $\mathbf{L} = \mathbf{C^T} \cdot \mathbf{Q}^{-1} \cdot \mathbf{l}$ and $\mathbf{Q}$ is the covariance matrix of the observations. Here only covariance matrices with the same value in each column $i$ (the 'weight' $w_i$) are considered.

# 2 Linear equations

## 2.1 Real

The merit function we want to minimise is:

$$\chi^2 = \sum_{i=0}^{N-1} \left[ \frac{l_i - y_i}{\sigma_i} \right]^2 \tag{6}$$

For the minimum of $\chi^2$ holds:

$$\frac{\partial \chi^2}{\partial x_k} = 0 \qquad\qquad k = 0, \ldots, n-1 \tag{7}$$

which leads to the following set of normal equations:

$$\sum_{i=0}^{N-1} \frac{[l_i - y_i]}{\sigma_i^2} \frac{\partial y_i}{\partial x_k} = 0 \qquad\qquad k = 0, \ldots, n-1 \tag{8}$$

or in matrix form:

$$\begin{pmatrix} [a_0 a_0] & [a_0 a_1] & \cdots & [a_0 a_{n-1}] \\ [a_1 a_0] & [a_1 a_1] & \cdots & [a_1 a_{n-1}] \\ \vdots & \vdots & \ddots & \vdots \\ [a_{n-1} a_0] & [a_{n-1} a_1] & \cdots & [a_{n-1} a_{n-1}] \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} [a_0 l] \\ [a_1 l] \\ \vdots \\ [a_{n-1} l] \end{pmatrix} \tag{9}$$

or:

$$\mathbf{A} \cdot \mathbf{x} = \mathbf{L} \tag{10}$$

This symmetric set of equations can be solved, if the matrix is positive definite, i.e. if there are no dependencies between the columns. The solution is given by:

$$\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{L} \tag{11}$$

### 2.1.1 Errors

After solution for the unknowns $\mathbf{x}$, $\chi^2$ as defined in (6) can be directly calculated if we rewrite it as:

$$\chi^2 = [ll] - 2 \sum_{k=0}^{n-1} [a_k l] x_k + \sum_{i=0}^{N-1} w_i y_i^2 \tag{12}$$

Noting that $y_i = \sum_{k=0}^{n-1} a_k x_k$ and using equation (9) to note that $[a_i l] = \sum_{k=0}^{n-1} [a_i a_k] x_k$, we can rewrite (12) as:

$$\chi^2 = [ll] - \sum_{k=0}^{n-1} x_k [a_k l] \tag{13}$$

The $\chi^2$ could be used to assess the goodness of fit if the actual $\sigma_i$'s were known, and the errors are normal distributed. In general the actual values of $\sigma_i$ are not known, and often the distribution is not normal (.e.g. if we solve for logarithmic values). An estimate of the standard deviation can be made by:

$$\sigma^2 = \frac{\left[(l_i - y_i)^2\right]}{N - n} \tag{14}$$

which can be estimated by:

$$\sigma_o^2 = \frac{\chi^2}{N-n} \tag{15}$$

to give 'an error per observation'. The 'error per unit weight', or the standard deviation, can be expressed as:

$$\sigma_w^2 = \frac{\chi^2}{[1]} \frac{N}{N-n} \tag{16}$$

The uncertainty in the solution $x_i$ can be expressed as:

$$\sigma^2(x_i) = \sum_{k=1}^{N-1} \sigma_k^2 \left( \frac{\partial x_i}{\partial y_k} \right)^2 \tag{17}$$

Since

$$x_i = \sum_{k=0}^{n-1} \left( \mathbf{A}^{-1} \right)_{ik} [a_k l] \tag{18}$$

we have:

$$\frac{\partial x_i}{\partial y_k} = \sum_{j=0}^{n-1} \left( \mathbf{A}^{-1} \right)_{ij} \frac{a_{kj}}{\sigma_k^2} \tag{19}$$

leading to:

$$\sigma^2(x_i) = \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} \left( \mathbf{A}^{-1} \right)_{ik} \left( \mathbf{A}^{-1} \right)_{il} \left[ \sum_{j=0}^{N-1} \frac{a_{kj} a_{lj}}{\sigma_j^2} \right] \tag{20}$$

Doing the sums, this equation reduces to:

$$\sigma^2(x_i) = \left( \mathbf{A}^{-1} \right)_{ii} \tag{21}$$

If the $\sigma_i$ was not known originally, the estimate of the standard uncertainties in the unknowns $\mathbf{x}$ are:

$$\sigma(x_i) = \sigma_o \sqrt{\left( \mathbf{A}^{-1} \right)_{ii}} \tag{22}$$

If the uncertainties in the unknowns are wanted, the inverse matrix $\mathbf{A}^{-1}$ is calculated by backsubstitution of the identity matrix, and the uncertainties by (22).

## 2.2 Complex

The merit function we want to minimise in this case is:

$$\chi^2 = \sum_{i=0}^{N-1} \left[\frac{l_i - y_i}{\sigma_i}\right] \left[\frac{l_i - y_i}{\sigma_i}\right]^* \tag{23}$$

Differentiating $\chi^2$ leads to the following set of normal equations:

$$\begin{pmatrix} [a_0^* a_0] & [a_0^* a_1] & \cdots & [a_0^* a_{n-1}] \\ [a_1^* a_0] & [a_1^* a_1] & \cdots & [a_1^* a_{n-1}] \\ \vdots & \vdots & \ddots & \vdots \\ [a_{n-1}^* a_0] & [a_{n-1}^* a_1] & \cdots & [a_{n-1}^* a_{n-1}] \end{pmatrix} \cdot \mathbf{x} = \begin{pmatrix} [a_0^* l] \\ [a_1^* l] \\ \vdots \\ [a_{n-1}^* l] \end{pmatrix} \tag{24}$$

The normal matrix is Hermitian. It can be solved by Choleski methods. However, internally the matrix is converted to a real form. Although this has an, in general, small memory penalty, it has no influence on CPU time, and makes it possible to use the same routines for complex and real solutions. The conversion to real is done by splitting each element $A_{ij}$ of the normal matrix into $A_{\Re ij} + \imath A_{\Im ij}$ and replacing it by:

$$A_{ij} = \begin{pmatrix} A_{ij\Re} & -A_{ij\Im} \\ A_{ij\Im} & A_{ij\Re} \end{pmatrix} \tag{25}$$

and simular replacements for the vector elements $x_i$ and $L_i$ as:

$$x_i = \begin{pmatrix} x_{i\Re} \\ x_{i\Im} \end{pmatrix} \tag{26}$$

and:

$$L_i = \begin{pmatrix} L_{i\Re} \\ L_{i\Im} \end{pmatrix} \tag{27}$$

Another reason for solving real rather than complex equations is given in the next section.

### 2.2.1 Separable complex

In cases where both the unknowns $\mathbf{x}$ and their complex conjugates $\mathbf{x}^*$ appear in the condition equations, differentiating the merit function (23) will not produce a symmetric or Hermitian normal matrix, since there exists no linear relation between $x_i$ and $x_i^*$. We could, of course, solve for $2n$ complex

equations with added constraints that the sum of even and odd unknowns must be real, and their difference imaginary, but this will lead to $4n$ complex equations.

If, however, we consider each complex unknown as two real unknowns (i.e. $x_{i\Re}$ and $x_{i\Im}$) then differentiating (23) produces the following symmetric set of $2n$ real equations:

$$\begin{pmatrix} [a_0{}^*a_0]_\Re & [a_1{}^*a_0]_\Im & [a_2{}^*a_0]_\Re & \cdots & [a_{2n-1}{}^*a_0]_\Im \\ -[a_0{}^*a_1]_\Im & [a_1{}^*a_1]_\Re & -[a_2{}^*a_1]_\Im & \cdots & [a_{2n-1}{}^*a_1]_\Re \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -[a_0{}^*a_{2n-1}]_\Im & [a_1{}^*a_{2n-1}]_\Re & -[a_2{}^*a_{2n-1}]_\Im & \cdots & [a_{2n-1}{}^*a_{2n-1}]_\Re \end{pmatrix} \cdot \begin{pmatrix} x_{0\Re} \\ x_{0\Im} \\ x_{1\Re} \\ \vdots \\ x_{n-1\Im} \end{pmatrix} = \begin{pmatrix} [a_0{}^*l]_\Re \\ [a_1{}^*l]_\Im \\ [a_2{}^*l]_\Re \\ \vdots \\ [a_{2n-1}{}^*l]_\Im \end{pmatrix} \tag{28}$$

A number of special cases can be distinguished.

In the 'normal' complex case (previous section), $a_{2i} \equiv a_{2i+1}$, and (28) reduces to (25).

If all $a_{2i}$ and $a_{2i+1}$ are real, but specified, e.g., as a complex number, (28) deteriorates into:

$$\begin{pmatrix} [a_{0\Re}a_{0\Re}] & 0 & [a_{0\Re}a_{1\Re}] & \cdots & 0 \\ 0 & [a_{0\Im}a_{0\Im}] & 0 & \cdots & [a_{0\Im}a_{n-1\Im}] \\ [a_{1\Re}a_{0\Re}] & 0 & [a_{1\Re}a_{1\Re}] & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & [a_{n-1\Im}a_{0\Im}] & 0 & \cdots & [a_{n-1\Im}a_{n-1\Im}] \end{pmatrix} \cdot \begin{pmatrix} x_{0\Re} \\ x_{0\Im} \\ x_{1\Re} \\ \vdots \\ x_{n-1\Im} \end{pmatrix} = \begin{pmatrix} [a_{0\Re}l_\Re] \\ [a_{0\Im}l_\Im] \\ [a_{1\Re}l_\Re] \\ \vdots \\ [a_{n-1\Im}l_\Im] \end{pmatrix} \tag{29}$$

Note that in this case there is a true separation of the real and imaginary parts of the different unknowns, and two separate real solutions with each $n$ unknowns will produce exactly the same result.

The full and special cases are all catered for in the aips++ routines.

### 2.2.2 Errors

Using arguments comparable to those in (2.1.1), we can get (13) for the complex case as:

$$\chi^2 = [ll^*] - \sum_{k=0}^{n-1} \left( x_{k\Re}[a_{2k}^* l]_{\Re} + x_{k\Im}[a_{2k+1}^* l]_{\Im} \right) \tag{30}$$

with the $a$'s and $x$'s as defined in (29).

## 3 Dependant linear equations

If there are not enough independent condition equations, the normal matrix **A** cannot be inverted, and a call to WNMLTN will fail with a *.false.* return value.

The equations could still be solved if some additional 'constraint' equations would be introduced. In the more complex cases the precise, let alone the best, form for these additional equations is difficult to determine (e.g. the redundancy situation in Westerbork).

A method known as 'Singular value decomposition' (*SVD*) can be used to obtain the minimal set of orthogonal equations that have to be added to solve the *LSQ* problem. Several implementations exist in the literature.

In general we can distinguish three types of constrained equations:

- the minimum number of sufficient constraint equations are known to be able to solve a system of equations
- constraints are used to add additional information (e.g. the sum of angles of a triangle)
- no actual constraint equations are known

All three cases are handled in the LSQ package, the first two in the same way.

The general constraint situation arises from the use of Lagrange multiplicators. Assume that in addition to the condition equations, with measured values, we have a set of $p$ rigorous equations:

$$\phi_i(\mathbf{x}) = 0 \qquad i = 0, \ldots, p-1 \tag{31}$$

We must therefore make $\chi^2$ minimal, subject to the set of (31), or:

$$\sum_{i=0}^{n-1} \frac{\partial \chi^2}{\partial x_i} dx_i = 0 \tag{32}$$

subject to the conditions:

$$\sum_{i=0}^{n-1} \frac{\partial \phi_k}{\partial x_i} dx_i = 0 \qquad k = 0, \ldots, p-1 \tag{33}$$

which leads to a set of $n + p$ equations:

$$\frac{\partial \chi^2}{\partial x_i} + \sum_{k=0}^{p-1} \lambda_k \frac{\partial \phi_k}{\partial x_i} = 0 \qquad i = 0, \ldots, n-1 \tag{34}$$

together with the (31).

Note that I have chosen for having constraint equations linear in the unknowns, with a zero value. In cases where this is not adequate (e.g. $x + y + z = 360$) a simple linear transformation will suffice to make it e.g. $x' + y' + z' = 0$.

Defining the second term in (34) as $B_{ik}$, we can write our expanded set of normal equations as:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{L} \\ 0 \end{pmatrix} \tag{35}$$

### 3.1 Unknown constraints

The routines to handle this situation use a method based on "Pseudo-inverse berekening en Cholesky factorisatie", Hans van der Marel, 27 maart 1990, Faculty of Geodesy, Delft University. I will briefly describe Hans' paper, together with the additions I have made.

In the case we are considering, the normal equation $\mathbf{A}$ has not full column rank. Therefore, there exists, if the rank defect is $p$, an $n \times p$ matrix $\mathbf{G}$, a basis for the null-space of $\mathbf{A}$, with $\mathbf{AG} = 0$. If we assume that $\mathbf{B}$ has just sufficient constraint equations to solve the rank defect, the inverse of the matrix in (35) is:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{\#} & \mathbf{G}\left(\mathbf{B}^T\mathbf{G}\right)^{-1} \\ \left(\mathbf{G}^T\mathbf{B}\right)^{-1}\mathbf{G}^T & 0 \end{pmatrix} \tag{36}$$

with $\mathbf{A}^{\#}$ the pseudo-inverse of $\mathbf{A}$. A number of relations hold for $\mathbf{A}^{\#}$, the most important for us that $\mathbf{B}^T\mathbf{A}^{\#} = 0$ and $\mathbf{A}^{\#}\mathbf{B} = 0$, or $\mathbf{B}$ is a base for the null-space of $\mathbf{A}^{\#}$. For a mininorm solution we can choose $\mathbf{B} = \mathbf{G}$. In that case $\mathbf{A}^{\#}$ is the Moore-Penrose inverse, and $\mathbf{B}^T\mathbf{G}$ is regular and symmetric.

We can now proceed in the following way:

1. Do a Cholesky factorisation of the normal equations $\mathbf{A}$ until the remaining columns of $\mathbf{A}$ are dependent. Pivoting along the diagonal of $\mathbf{A}$ occurs to make sure that $\mathbf{A}$ can be partitioned in an independent and a dependent part with rank defect $n_2$. Dependency is determined by looking at the angle between a column and the space formed by the already determined independent space (the so-called 'collinearity number').

   If $n_1 = n - n_2$, we can say that after $n_1$ Choleski steps, we have:

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} U_{11}^T & 0 \\ U_{12}^T & I \end{pmatrix} \cdot \begin{pmatrix} U_{11} & U_{12} \\ 0 & \overline{A}_{22} \end{pmatrix} \tag{37}$$

   with $U_{11}$ the Cholesky factor of $A_{11}$, $U_{12} = {U_{11}^T}^{-1} A_{12}$ and $\overline{A}_{22} = A_{22} - U_{12}^T U_{12}$.

   The collinearity angle $\delta$ can be determined for the current column $i$ by: $\sin^2 \delta = u_{ii}^2 / a_{ii}$.
2. Determine a $G_1$ replacing the (rectangular) $U_{12}$ from $U_{11} G_1 = -U_{12}$.
3. Determine a (symmetric) $G_2$ replacing $A_{22}$ from the rank basis $I + G_1^T G_1 = (G_2^T - 1)(G_2 - 1)$
4. determine constraint equations $G = (G_1 G_2, G_2)^T$.
5. Solve for the $n_1$ independent unknowns $\mathbf{x}_1$ using $A_{11}$ and $L_1$ by back substitution
6. Solve constraint equations for the remaining $n_2$ unknowns, using $G_2$ by back substitution: $\mathbf{x}_2 = -G_2^{-1} G_1^T \mathbf{x}_1$
7. Make Baarda's S-transform of independent $n_1$ unknowns: $\mathbf{x}_1 = \mathbf{x}_1 + G_1 \mathbf{x}_2$.

Setting $\mathbf{F} = A_1^{-1} L_1$, $\mathbf{D} = -G_2^{-1} G_1^T$ and $\mathbf{E} = \begin{pmatrix} \mathbf{I}_{n_1} + G_1 \mathbf{D} \\ \mathbf{D} \end{pmatrix}$, we can write the above steps as:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{I}_{n_1} + G_1 \mathbf{D} \\ \mathbf{D} \end{pmatrix} \begin{pmatrix} A_1^{-1} L_1 \end{pmatrix} = \mathbf{E} \cdot \mathbf{F} \tag{38}$$

### 3.1.1 Errors

Errors are determined in the same way as described in (2.1.1), where it should be noted that (13) can either be used summing over $n_1$ variables and using the first guess of $\mathbf{x}_1$, or over $n$ and using the final $\mathbf{x}$. Internally the

first option is used. The uncertainties in $\mathbf{x}$ are determined by calculating the covariance matrix. Using similar arithmetic as in (2.1.1), it can be shown that:

$$\mathbf{A}^{-1} = \mathbf{E} \cdot \mathbf{a}_1^{-1} \cdot \mathbf{E}^T \tag{39}$$

$\mathbf{A}^{-1}$ is calculated by first solving $\mathbf{H} = \mathbf{E}\left(\mathbf{A}_1^{-1} \cdot \mathbf{I}_{n_1}\right)$ and then $\mathbf{A}^{-1} = \mathbf{E} \cdot \mathbf{H}^T$.

### 3.2 Known constraints

In the case of known constraints, hence when $\mathbf{B}$ is known in (35), this equation can be solved. Cholesky decomposition does not work in this case, and a, transparent, Crout LU decomposition is done to determine the (symmetric) covariance (i.e. inverse) matrix of the left hand side of (35). .

#### 3.2.1 Errors

Errors are determined as explained in (2.1.1). Since all constraint equations are $\equiv 0$, the sum in (13) is taken over $n$, not over $n + p$ if $p$ are the number of constraint equations.

## 4 Non-linear equations

In the case of non-linear condition equations, no simple solution to minimise the merit function (e.g. (6)) exists. A simple solution is to take the condition equation, make a first order Taylor expansion around an estimated $\hat{\mathbf{x}}$; solve for $d\mathbf{x}$, and iterate till solution found.

However, better and more stable methods exist (e.g. steepest-descend) for some circumstances. A method that seems to be quite stable and using both steepest-descent and Taylor expansion, is the method by Levenberg-Marquardt (see e.g. "Numerical recipes", W.H. Press *et al.*, Cambridge University Press).

If we have an estimate for $\mathbf{x}$, we can find a better one by:

$$\hat{\mathbf{x}}_{next} = \hat{\mathbf{x}} + \mathbf{H}^{-1} \cdot \left[-\nabla\chi^2(\hat{\mathbf{x}})\right] \tag{40}$$

where $\mathbf{H}$ is the Hessian matrix of $\chi^2$. If our approximation is not good enough, we could use the steepest-descent by calculating:

$$\hat{\mathbf{x}}_{next} = \hat{\mathbf{x}} - constant \cdot \left[-\nabla\chi^2(\hat{\mathbf{x}})\right] \tag{41}$$

The Hessian matrix $\mathbf{H}$ has as elements:

$$H_{ij} = \frac{\partial^2 \chi^2}{\partial x_i \partial x_j} = 2 \sum_{k=0}^{N-1} \frac{1}{\sigma_i^2} \left[ \frac{\partial y_k}{\partial x_i} \frac{\partial y_k}{\partial x_j} - (l_i - y_i) \frac{\partial^2 y_i}{\partial x_i \partial x_j} \right] \tag{42}$$

By dropping the second term in (42), and multiplying each $H_{ii}$ term with $(1 + \lambda)$, and defining the first derivative of $\chi^2$ as $\mathbf{b}$, we can combine the equations (40), (41) into:

$$\mathbf{H} \cdot d\mathbf{x} = \mathbf{b} \tag{43}$$

which can be solved as standard normal equations.

Choosing $\lambda$ is the crux of the matter, and where to stop iterating. A start value of $\lambda = 0.001$ is used in the following routines. The method looks at the $\chi^2$ for $\hat{\mathbf{x}} + d\mathbf{x}$. If it has increased over the value of $\chi^2$ for $\hat{\mathbf{x}}$, $\hat{\mathbf{x}}$ is unchanged, and $\lambda = 10\lambda$. If there is a decrease; a new value for $\hat{\mathbf{x}}$ is used, and $\lambda = \lambda/10$. Iteration can be stopped if the fractional decrease in $\chi^2$ is small (say $< 0.001$); never if there is an increase in $\chi^2$.

### 4.1 Errors

Errors are calculated as described in (2.1.1). The errors returned by WNMLSN are, of course, only approximations, since the original equations are non-linear, but give a good impression of the residuals. The covariance matrix should be calculated by doing a final linear run on the residuals, and solve the equations.

## 5 Summary

An *LSQ*-object takes a total space of:

$$\begin{aligned} 9 &+ \text{(administration)} \\ (n+p)(n+p+1)/2 &+ \text{(normal array)} \\ 4m &+ \text{(error calculations)} \\ m(m+p) &+ \text{(known sides)} \\ (n+p+1)/2 &+ \text{(pivot area)} \\ (n+p) & \quad \text{(solution) 8 byte words} \end{aligned} \tag{44}$$

where $n$ is the number of unknowns ($2n$ if complex); $m$ the number of simultaneously to be solved equations; $p$ the number of external constraint equations. In the non-linear case two *LSQ*-objects are used. In cases where

the inverted normal array is calculated (known constraints) a temporary storage of $n^2$ 8-byte words is used.

The following calls are available:

1. To be given

*wnb, July 30, 2015*