HEAT AND MASS BALANCE USING CONSTRAINT EQUATIONS, A SPREADSHEET, AND THE NEWTON-RAPHSON TECHNIQUE

R C LOUBSER

Sugar Milling Research Institute, University of KwaZulu-Natal, Durban, 4041, South Africa.
E-mail: rloubser@smri.org

Abstract

To work out a balance problem in a factory, two simple rules are applied. Firstly, a steady state condition applies. This gives rise to the second rule: what goes in must come out. Based on these rules, the equations for heat and mass balances can be derived. There are several packages available to solve these rules or constraints. The choices include Simulink and Sugars. The use of these packages would require the ownership of a software licence. Often it is not cost effective to purchase the licence for a single project. This leaves the choice of solving the problem using a spreadsheet.

Once the equations are expressed in terms of brix, fibre, etc, the equations become non-linear, and linear methods of solution such as Gauss-Jordan row reduction are no longer possible. Traditionally, the equations are then manipulated to isolate terms and thereby extract a solution. This approach fails when there are numerous return streams. In this case, the iteration facility of the spreadsheet is used in an attempt to resolve the values that could not be solved explicitly.

This paper describes a technique where the constraint equations are entered into the spreadsheet for each point of mixing or separation in the system. The Jacobian matrix can then be constructed using some elementary rules. Thereafter, the power of the spreadsheet matrix functions can be employed to iterate simply to a solution using the Newton-Raphson (or any other applicable) technique. This process eliminates the need to manipulate the constraints to isolate variables, and ensures that the iteration can be handled in an orderly manner.

Keywords: modelling, simulation, spreadsheet, mass balance, heat balance, flowsheeting

Introduction

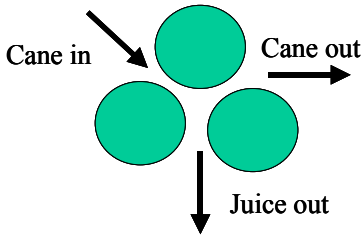
There are many software products available for solving heat and mass balance problems. A package such as Sugars (www.sugarsonline.com) has been used extensively and effectively for modelling sugar factories. A useful model was constructed for the Malelane and Komati mills and is reported by Stolz and Weiss (1997). Another package that is available for use is Simulink (Peacock, 2002). To use these powerful tools, a licence for the specialised software package will be required. Often the size of the project does not warrant the expenditure.

Another alternative that was used, particularly before large, flexible computing power was readily available, was to develop purpose built computer programs (Hoekstra, 1981, 1983, 1985; Guthrie, 1972). These programs were generally written for specific applications. They made optimum use of the computing facilities that were available at the time. Any significant changes in the problem posed would lead to recoding of parts of the program. This meant that

access to the source code and compiler or interpreter was required. In addition to this, certain programming skills were also necessary.

Spreadsheets have become powerful tools for solving problems. They are used to do everything from producing financial statements to designing machinery. Mass balance calculations have also been performed by numerous people, for example Lionnet and Achary (2001), Wienese (1992) and Getaz (1990). Since the component quantities such as fibre and brix are expressed as a percentage of total mass, the equations are non-linear. They therefore cannot be solved by the linear method of assembling the equations and solving them using a form of Gauss row reduction or matrix inversion. The resulting equations are usually manipulated so that the unknowns can be isolated and therefore formulas are derived for entry into the spreadsheet. Problems arise where complex flow and counter flow patterns exist, such as mud recycle or vapour bleeding. In these cases, it is often necessary to resort to iterative techniques. The equations are manipulated so that an unknown parameter is expressed in terms of a function, which includes a reference to the parameter itself, either directly or indirectly. The spreadsheet program's loop calculation or iteration facility is then used to seek the value of the unknown parameter. It may be necessary to introduce factors to limit the oscillations in the result and promote convergence to the value sought.

Another technique that is available is to use the optimiser in the spreadsheet (Hubbard and Love, 1998). The program varies several values to achieve a specific result on a single variable. Again, it is often necessary to use factors and weightings to avoid instabilities in the calculation which may result in the solution diverging.



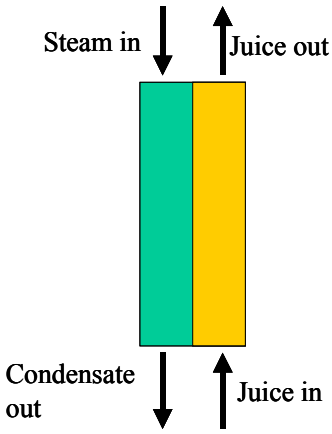


Figure 1(a) Mill balance. **(b) Heater balance.**

Instead of using the built-in solver/optimiser, a solution scheme can be coded into the spreadsheet. This gives the user nearly full control over how the problem is solved by the spreadsheet program. Although the mathematical approach is essentially the same as that used by Hoekstra (1983), the use of a spreadsheet makes the technique available to anyone with a spreadsheet package on their computer. The technique discussed here is the Newton-Raphson method. Although it requires a small amount of mathematical skill, a few simple rules can be applied to generate the model.

Constraints and balances

The principle of a material or energy balance is that what flows into the system must flow out, albeit in an altered from. In other words, the net flow into a system must be zero. Consider the overall mass flow of a mill shown in Figure 1(a).

Balancing the mass entering and leaving gives:

$$Mc_{in} = Mb_{out} + Mj_{out}$$

where:

c refers to cane
b refers to bagasse
j refers to juice

Alternatively, calculating the net mass flow into the mill using outflow as negative gives:

$$M_{cane\ in} - M_{cane\ out} - M_{juice\ out} = 0$$

which is in the form of a constraint equation.

Figure 1(b) shows a schematic of a juice heater. Balancing the heat flow gives:

$$M_{st\ in} \times h_{st\ in} - M_{cnd\ out} \times h_{cnd\ out} + M_{jce\ in} \times h_{jce\ in} - M_{jce\ out} \times h_{jce\ out} = 0$$

Two other constraint equations arise from the mass balance:

$$M_{st\ in} - M_{cnd\ out} = 0$$
$$M_{jce\ in} - M_{jce\ out} = 0$$

All these constraint equations form a set of *n* equations of the form:

$$\phi_i(q_1, q_2, q_3, ...q_n) = 0 \quad i = 1..n \tag{1}$$

Once all the constraint equations have been defined, values for the unknowns, q_i, need to be found that satisfy all the constraints. Several possible algorithms exist, but the one tested for this study was the Newton-Raphson approach.

For the system of equations to be completely defined there must be one equation for each unknown and the equations must be independent from each other, otherwise the least squares method described by Hoekstra (1983) would have to be implemented.

Satisfying the constraints

Starting with the *n* equations it can be seen that the functions, ϕ_i, will have a value of zero only at specific values of the variables, q_i. At all other values of q_i the functions will have non-zero values. The task of the solution strategy is to adjust the values of the variables in such a manner as to reduce the values of all the functions to zero. One such strategy or algorithm is the Newton-Raphson method.

Seeking the root of a function

The root of a function is the point where the function is zero. The method of finding this zero point is best understood considering a single degree-of-freedom function as shown in Figure 2. First a guess is made for the value of the root; say 8. At this value the function has a y-value of 4.2 and a slope of 2.3. Taking a step of distance δ, which can be calculated from

the y-value and slope, will lead to a better estimate of the position where the function cuts the x-axis. This process can be repeated until the result is sufficiently close to the actual value for practical use.

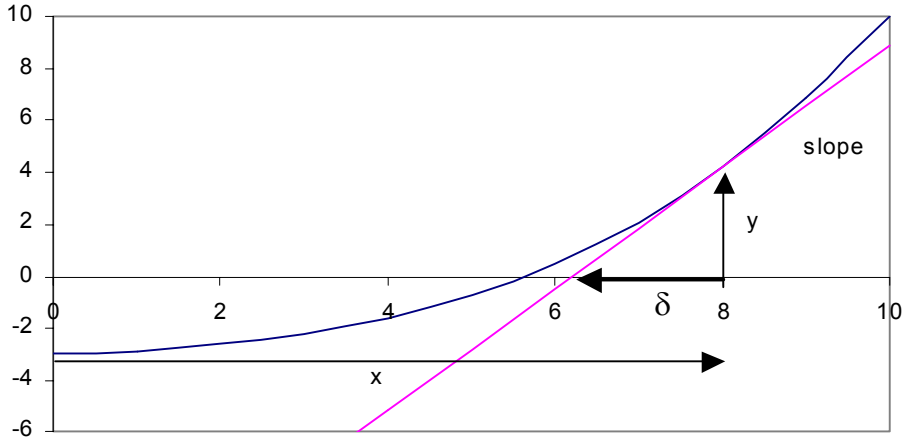


Figure 2. Single degree-of-freedom function.

For a more rigorous explanation, the Taylor's series expansion needs to be used.

The value of a function in the neighbourhood of a known value can be calculated using the known value and a series of derivatives and the change in the independent variable.

$$\phi(x+\delta) = \phi(x) + \frac{\partial\phi(x)}{\partial x}\delta + \frac{\partial^2\phi(x)}{\partial x^2}\frac{\delta^2}{2!} + R$$

Since the objective is to find the root of the function, ϕ, the left hand side becomes zero. If the step size, δ, is small then the higher order terms can be neglected. The equation then becomes:

$$0 = \phi(x) + \frac{\partial\phi(x)}{\partial x}\delta$$

This equation can be rearranged in terms of δ:

$$\delta = -\phi(x) \times \left(\frac{\partial\phi(x)}{\partial x}\right)^{-1}$$

This is the same equation as would be obtained with the geometric approach.

The Taylor's series can be extended to the multi-degree-of-freedom system that arises from applying constraint equations to the mass or heat balance equations associated with the series of pieces of equipment. The difference is that there are *n* equations and unknowns rather than just one, so a convenient matrix and vector notation must be used instead of the single function. The Taylor's series expansion, neglecting higher order terms and equating to zero, then becomes:

$$\underline{0} = \underline{\phi}(\underline{q}) + \mathbf{J}\underline{\delta}$$

Instead of having a single derivative, a matrix of derivatives, known as the Jacobian, is required. This matrix has the form:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \phi_1}{\partial q_1} & \frac{\partial \phi_1}{\partial q_2} & \Lambda & \frac{\partial \phi_1}{\partial q_n} \\ \frac{\partial \phi_2}{\partial q_1} & \frac{\partial \phi_2}{\partial q_2} & \Lambda & \frac{\partial \phi_2}{\partial q_n} \\ \mathrm{M} & \mathrm{M} & \mathrm{O} & \mathrm{M} \\ \frac{\partial \phi_n}{\partial q_1} & \frac{\partial \phi_n}{\partial q_2} & \Lambda & \frac{\partial \phi_n}{\partial q_n} \end{bmatrix}$$

Although this matrix may appear rather complex, a few simple rules can be used to construct it. These will be discussed later.

In a similar manner in which the single degree-of-freedom equation was rearranged, the multi-degree-of-freedom equation may be rearranged to give the step size required to improve the estimate of the point where the constraint equations are satisfied.

$$\underline{\delta} = -\mathbf{J}^{-1}\underline{\phi}$$

In this case the power –1 refers to the matrix inverse rather than division, as was the case with the single degree-of-freedom calculation. The matrix inverse is available as a function in most modern spreadsheets.

Calculating the Jacobian

The expression $\partial\phi_1/\partial q_1$ means the slope of the function ϕ_1 resulting from varying q_1 while keeping all the other variables constant. Since the other variables are kept constant, the function can be shown as a graph similar to that shown in Figure 3.

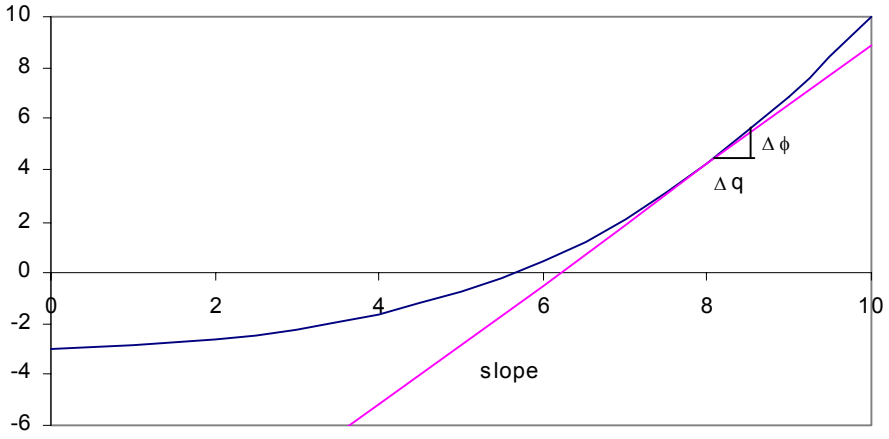


Figure 3. Determining slope of constraint function.

The slope of the graph can be worked out by evaluating the value of the function at a particular value of q_1 and at another point a small distance, Δq_1, away. The slope is then approximated by the ratio of the change in the value of the function, $\Delta\phi_1$, and the change Δq_1.

That is:

$$\frac{\partial \phi_1}{\partial q_1} \approx \frac{\Delta \phi_1}{\Delta q_1}$$

To illustrate this, consider the constraint equation for the brix balance across the k'th mill. Let this give rise to the l'th constraint equation:

$$\phi_l \equiv Mc_k \times Bc_k - Mb_k \times Bb_k - Mj_k \times Bj_k = 0$$

Consider, for example, the brix in bagasse term, Bb_k. To follow the rule, a small change in the Bb_k term, ΔBb_k must be added to the constraint equation. The difference between this value $\phi_l(Bb_k+\Delta Bb_k)$ and $\phi_l(Bb_k)$ must be divided by the step ΔBb_k.

$$\frac{\partial \phi_l}{\partial Bb_k} = \frac{(Mc_k \times Bc_k - Mb_k \times (Bb_k + \Delta Bb_k) - Mj_k \times Bj_k) - (Mc_k \times Bc_k - Mb_k \times Bb_k - Mj_k \times Bj_k)}{\Delta Bb_k}$$
$$= -Mb_k$$

The value for q_i in the Jacobian is therefore the coefficient of q_i. Using this rule, the terms for the k'th mill and the l'th constraint equation can be entered into the Jacobian

	…	Mc_k	Bc_k	…	Mb_k	Bb_k	…	Mj_k	Bj_k	…
ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ
$\phi_l \equiv$	…	Bc_k	Mc_k	…	$-Bb_k$	$-Mb_k$	…	$-Bj_k$	$-Mj_k$	…
ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ	ℵ

Appendix A shows the application of this to a five mill tandem.

The above rule only holds for a situation where the function of q_i is linear. This will be the case for most balance equations. If non-linear conditions exist, such as a log or a power relationship, then calculus may be used to calculate the derivatives. Alternatively, a small value can be given to Δq_i and then the ratio $\Delta \phi_i / \Delta q_i$ explicitly calculated.

Solution procedure

The solution procedure is best summarised using a flow diagram, which is shown in Figure 4. First the known and initial estimate values need to be defined. The error in the constraint equations can then be calculated and the Jacobian matrix can be defined. From this, the inverse of the Jacobian can be calculated by using the spreadsheet function. The calculation of the step size and updating the size of step to the next estimate can be calculated. The process is repeated until the step size is small.

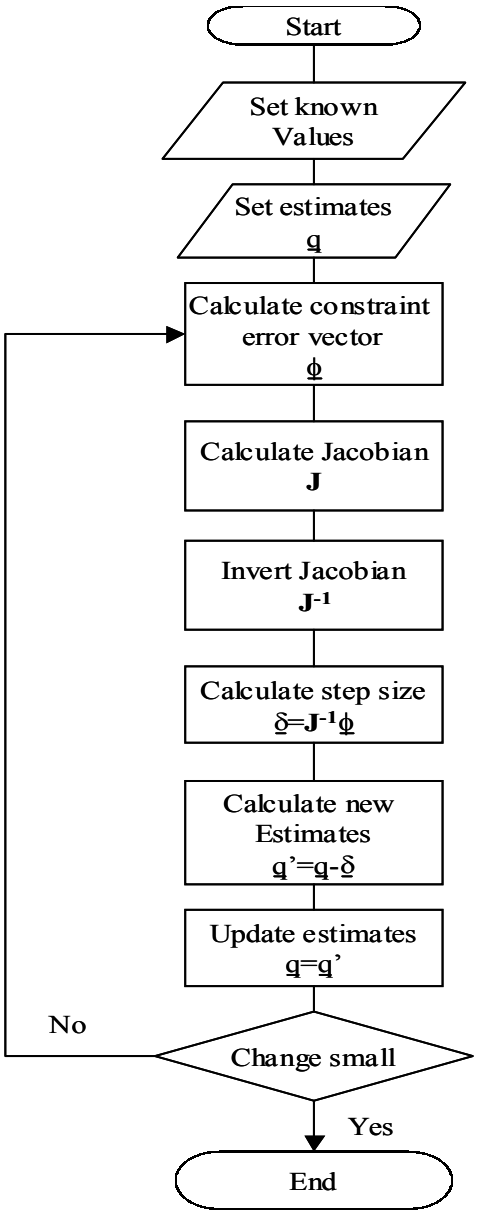


Figure 4. Flow diagram.

The implementation of the process in a spreadsheet is shown in Figure 5. The known values, such as mass, fibre and brix entering the system, were entered in an area of the spreadsheet. An area was designated where the estimates of the unknown values were stored. This area was first initialised using initial estimates of the values. The estimates together with the known values were used to calculate the error or ϕ vector. The Jacobian depends on the known values and the estimates. Spreadsheet functions were used to calculate the inverse of the Jacobian. The inverse of the Jacobian could then be multiplied by the ϕ vector the give the step size or δ vector. New estimates of the unknowns could be calculated by adding the step size to the current estimates. The current estimates were then assigned to be equal to the new estimates. This process was repeated using the spreadsheet iteration facility until the size of any change was small enough to satisfy the convergence criterion.

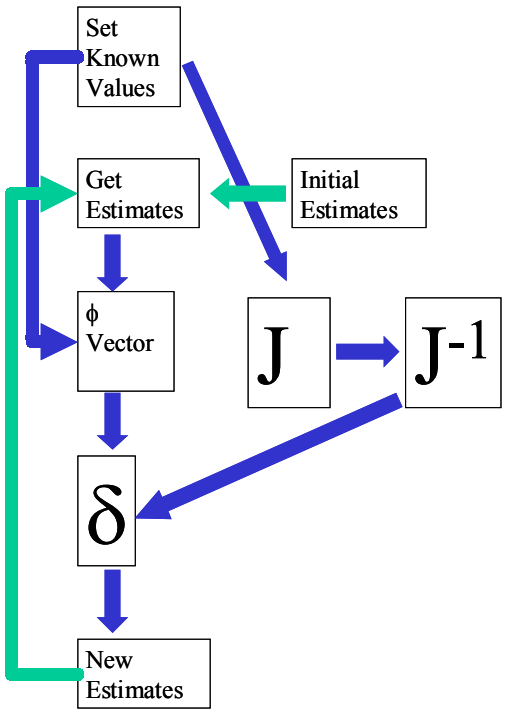


Figure 5. Spreadsheet structure.

Advantages and disadvantages

The main advantage is that a spreadsheet is a tool that is readily available. The flowsheeting and specialised programs available allow the user to construct models without requiring knowledge of the mathematics behind the solution algorithm. With the Newton-Raphson approach, the user has the advantage of having full control over the equations that are used to model elements of the system. Although the technique is mathematical, the spreadsheet provides the matrix algebra required to solve the problem.

Using the Newton-Raphson technique reduces the amount of algebra that must be performed to formulate the problem for solution by the spreadsheet. It is only necessary to derive the constraint or balance equations around individual equipment and points of mixing or separation rather than attempting to explicitly solve the non-linear set of equations that arise when the interaction between equipment and return streams is analysed. The iteration is performed in an orderly manner ensuring convergence of the solution process. The formulation of the constraint equations in the spreadsheet avoids the need for special dampers and weighting factors that would have to be introduced, usually by trial and error, to prevent divergence of the solution when the internal solver is used to solve for arbitrary variables.

Conclusion

The Newton-Raphson technique, as implemented on a spreadsheet was discussed. This technique can be used for solving heat and mass balance problems. Although the technique requires more mathematics from the user than flowsheeting and purpose written programs, it is an alternative technique that may be implemented where conventional algebraic manipulation of the balance equations leads to difficulties such as divergence where iteration is required to address circular reference problems. Unlike the conventional algebraic technique, which manipulates several values to optimise a single value, the Newton-Raphson technique manipulates values to reduce the value of several variables to zero simultaneously. This gives rise to a scheme that is more likely to converge.

The technique limits the algebraic manipulation to a node-by-node balance and no manipulation is required to accommodate inter-node flow relationships.

Nomenclature

M	=	mass flow
h	=	enthalpy
n	=	number of equations
i	=	equation number
ϕ	=	constraint function
q	=	unknown value
R	=	higher order terms
$\mathbf{J}$	=	Jacobian matrix
$\underline{\phi}$	=	vector of values of constraint functions
Mc	=	mass of cane entering mill
Bc	=	brix of cane entering mill
Mb	=	mass of bagasse leaving mill
Bb	=	brix of bagasse leaving mill
Mj	=	mass of juice leaving mill
Bj	=	brix of juice leaving mill

Matrices are represented by **bold** type and vectors are underlined.

REFERENCES

Getaz MA (1990). Malelane mass balances. Technical Note 26/90, Sugar Milling Research Institute, University of KwaZulu-Natal, Durban, South Africa. 22 pp.

Guthrie AM (1972). Sugar factory material balance calculations with the aid of a digital computer. *Proc S Afr Sug Technol Ass* 46: 110-115.

Hoekstra RG (1981). A computer program for simulating and evaluating multiple effect evaporators in the sugar industry. *Proc S Afr Sug Technol Ass* 55: 43-50.

Hoekstra RG (1983). A flexible computer program for four-component balances in sugar industry boiling houses. *Int Sug J* 85: 227-232 and 262-265.

Hoekstra RG (1985). Program for simulating and evaluating a continuous A-sugar pan. *Proc S Afr Sug Technol Ass* 59: 48-57.

Hubbard G and Love DJ (1998). Reconciliation of process flow rates for steady state mass balance on centrifugal. *Proc S Afr Sug Technol Ass* 72: 290-299.

Lionnet GRE and Achary M (2001). Mass balance calculations with mud routing to the extraction plant. Technical Report No. 1857, Sugar Milling Research Institute, University of KwaZulu-Natal, Durban, South Africa. 9 pp.

Peacock SD (2002). The use of Simulink for process modelling in the sugar industry. *Proc S Afr Sug Technol Ass* 76: 444-455.

Stolz N and Weiss W (1997). Simulation of Malelane and Komati mills with SUGARS™ simulation software. *Proc S Afr Sug Technol Ass* 71: 184-188.

Wienese A (1992). Simunye extraction model. Technical Note 47/92, Sugar Milling Research Institute, University of KwaZulu-Natal, Durban, South Africa. 13 pp.

www.sugarsonline.com, Sugars International, Englewood, USA. (accessed 13 Jan 2004).

APPENDIX A

Five mill mass balance example

As an example, a five-mill tandem, shown in Figure 6, was modelled.

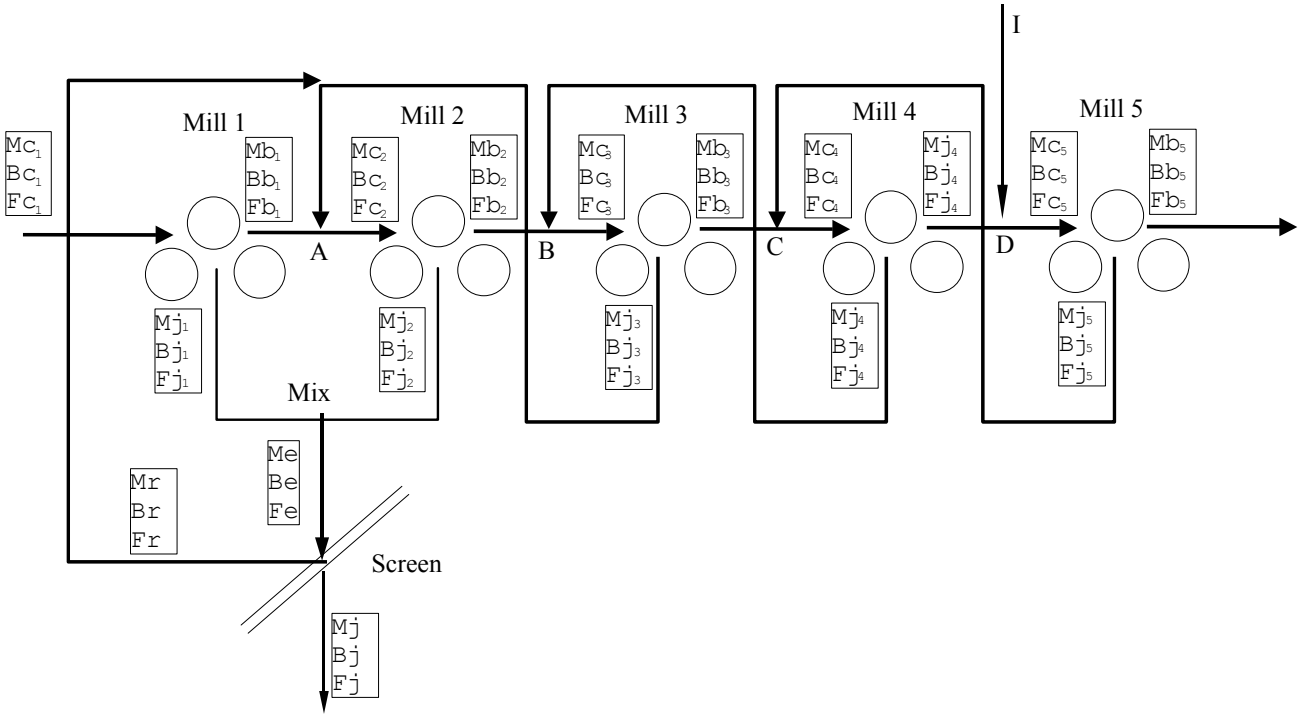


Figure 6. Five mill tandem.

The following parameters were assumed to be available:

Mc1, Bc1, Fc1, Bb1, Bj1, Fj1, Bb2, Bj2, Fj2, Bb3, Bj3, Fj3, Bb4, Bj4, Fj4, Bb5, Bj5, Fj5, Bj, Fj, I

Where:

M is mass
B is Brix
F is fibre
c is cane
b is bagasse
j is juice
e is extract
r is cush-cush return

and the number represents the mill number.

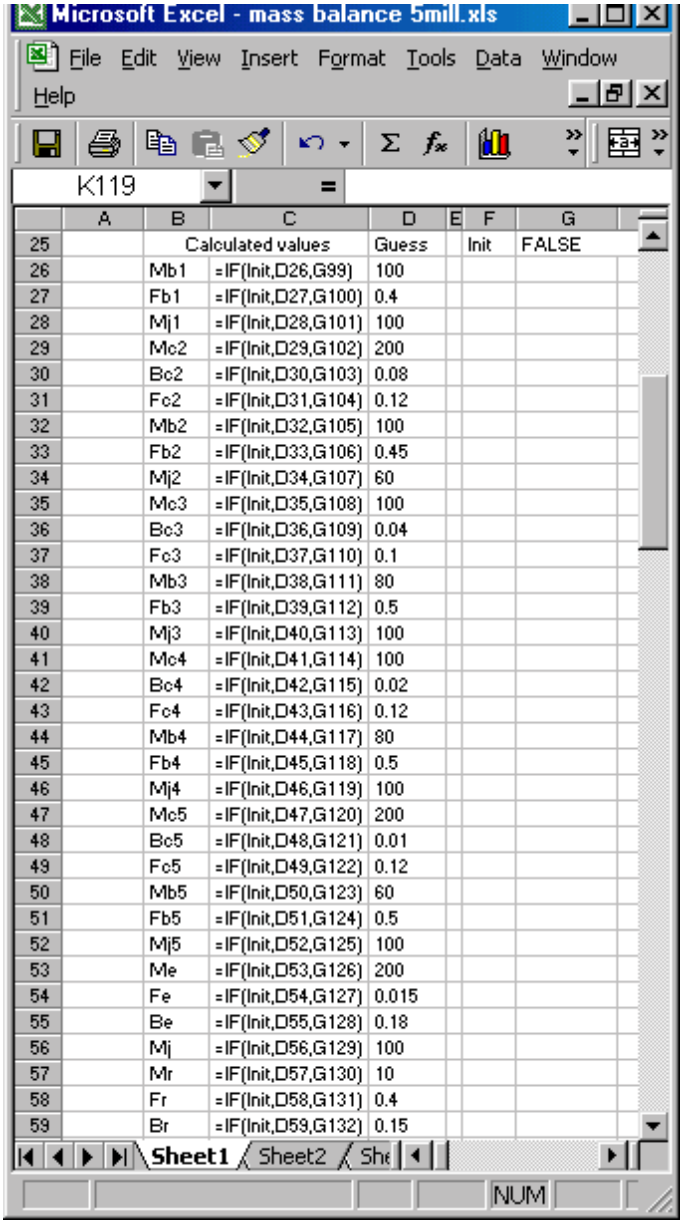
I is imbibition per cent fibre.

It is recommended that automatic calculation is turned off while the spreadsheet is constructed (*Tools-Options-Calculation-Manual-OK*). This problem was solved using Excel. The first step is to assemble the unknowns. For simplicity of understanding, names were given to the cells so that the formulas could be written in the same way as the equations would be written *(<select cell>-Insert-Name-Define-<cell_name>*). Note, however, the

introduction of the underbar (_) to avoid conflict with cell co-ordinate references. The Newton-Raphson technique requires an initial estimate of the unknown values. An 'if statement', shown in Figure 7, was used to either select the estimate stored in D26 to D59 or the result of the calculation stored in G99 to G132, shown in Figure 9, depending of the value of an 'init' cell. If 'init' was 'TRUE' then initial estimates would be copied to C25 to C59.

The constraint equations were developed using the material balances for overall mass, brix and fibre. This left the equation set under defined by one equation. To resolve this an assumption was made that the brix to water ratio of the juice arriving at the screen was the same as that of the juice leaving the screen. The resulting equation was manipulated to avoid division by unknown quantities. The vector was given the name phi.

The constraint equations and the Jacobian are shown in Figure 8. Note that the Jacobian has the same number of rows as columns.



	A	B	C	D	E	F	G
25		Calculated values		Guess		Init	FALSE
26		Mb1	=IF(Init,D26,G99)	100			
27		Fb1	=IF(Init,D27,G100)	0.4			
28		Mj1	=IF(Init,D28,G101)	100			
29		Mc2	=IF(Init,D29,G102)	200			
30		Bc2	=IF(Init,D30,G103)	0.08			
31		Fc2	=IF(Init,D31,G104)	0.12			
32		Mb2	=IF(Init,D32,G105)	100			
33		Fb2	=IF(Init,D33,G106)	0.45			
34		Mj2	=IF(Init,D34,G107)	60			
35		Mc3	=IF(Init,D35,G108)	100			
36		Bc3	=IF(Init,D36,G109)	0.04			
37		Fc3	=IF(Init,D37,G110)	0.1			
38		Mb3	=IF(Init,D38,G111)	80			
39		Fb3	=IF(Init,D39,G112)	0.5			
40		Mj3	=IF(Init,D40,G113)	100			
41		Mc4	=IF(Init,D41,G114)	100			
42		Bc4	=IF(Init,D42,G115)	0.02			
43		Fc4	=IF(Init,D43,G116)	0.12			
44		Mb4	=IF(Init,D44,G117)	80			
45		Fb4	=IF(Init,D45,G118)	0.5			
46		Mj4	=IF(Init,D46,G119)	100			
47		Mc5	=IF(Init,D47,G120)	200			
48		Bc5	=IF(Init,D48,G121)	0.01			
49		Fc5	=IF(Init,D49,G122)	0.12			
50		Mb5	=IF(Init,D50,G123)	60			
51		Fb5	=IF(Init,D51,G124)	0.5			
52		Mj5	=IF(Init,D52,G125)	100			
53		Me	=IF(Init,D53,G126)	200			
54		Fe	=IF(Init,D54,G127)	0.015			
55		Be	=IF(Init,D55,G128)	0.18			
56		Mj	=IF(Init,D56,G129)	100			
57		Mr	=IF(Init,D57,G130)	10			
58		Fr	=IF(Init,D58,G131)	0.4			
59		Br	=IF(Init,D59,G132)	0.15			

Figure 7. Calculated values.

Jacobian

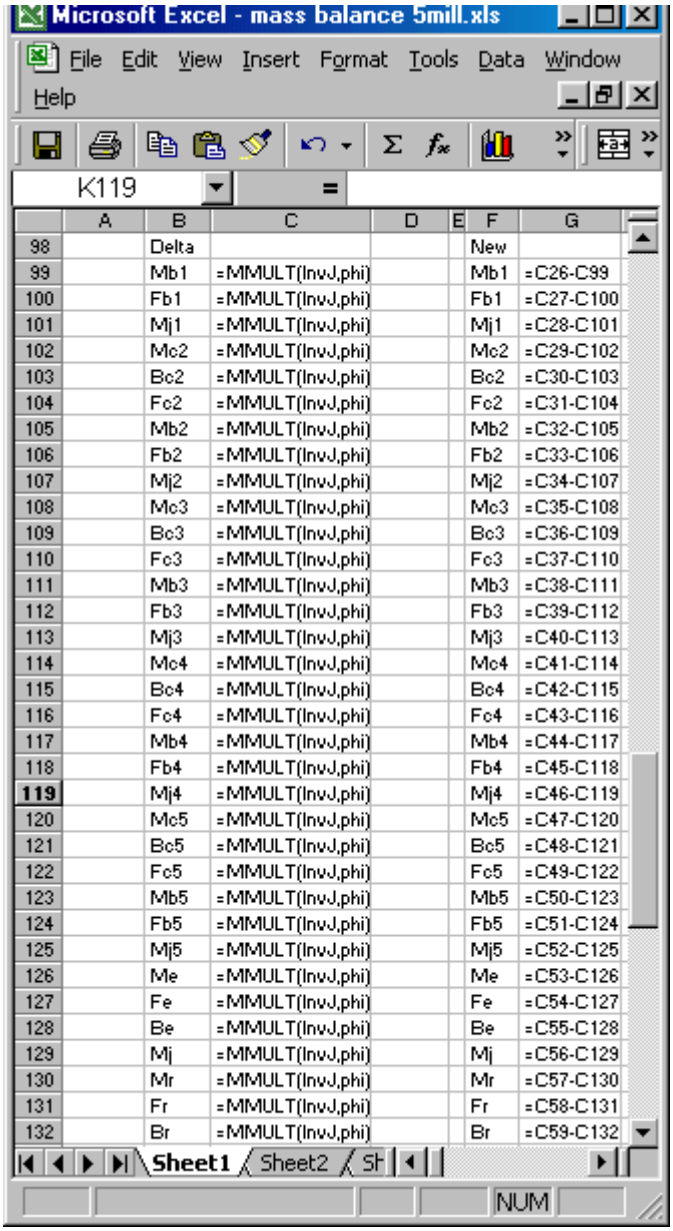
	Mb1	Fb1	Mj1	Mc2	Bc2	Fc2	Mb2	Fb2	Mj2	Mc3	Bc3	Fc3	Mb3	Fb3	Mj3	Mc4	Bc4	Fc4	Mb4	Fb4	Mj4	Mc5	Bc5	Fc5	Mb5	Fb5	Mj5	Me	Fe	Be	Mj	Mr	Fr	Br
1	-1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	=-F_b1	=-M_b1	=-F_j1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	=-B_b1	0	=-B_j1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	-1	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	=F_c2	0	=M_c2	=-F_b2	=-M_b2	=-F_j2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	=B_c2	=M_c2	0	=-B_b2	0	=-B_j2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	=F_c3	0	=M_c3	=-F_b3	0	-1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	=B_c3	=M_c3	0	=-B_b3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	=F_c4	0	=M_c4	-1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=B_c4	=M_c4	0	=-B_b4	0	=-B_j4	0	=-M_c5	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	=F_c5	0	=M_c5	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=B_b4	0	=B_j4	=B_c5	=M_c5	0	=-M_c5	0	=-B_j5	=-B_e	0	=-M_e	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=M_c5	=B_c5	=M_c5	0	=-B_b5	=M_b5	=F_e	=M_e	0	=-M_e	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	=B_c5	=M_c5	0	=-B_b5	=M_b5	=F_j5	=M_e	0	=-M_e	0	0	0
16	1	0	0	-1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=B_j5	0	=M_e	0	1	0	0
17	=F_b1	=M_b1	0	=F_c2	0	=M_c2	0	0	0	0	0	0	=F_b3	0	0	0	0	0	=F_b4	=M_b4	0	=F_c5	0	=M_c5	0	0	0	0	=F_e	=M_e	0	=F_r	=M_r	0
18	=B_b1	0	=B_j1	=B_c2	=M_c2	0	=M_b2	0	=M_j2	0	0	0	=B_b3	0	=M_j3	=B_c4	=M_c4	0	=B_b4	0	=M_j4	=B_c5	=M_c5	0	0	0	0	=B_e	0	=M_e	=B_r	0	=M_r
19	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	=F_b2	=M_b2	0	=F_c3	0	=M_c3	0	0	0	0	0	0	=F_j4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	=B_b2	0	0	=B_c3	=M_c3	0	0	0	0	0	0	0	=B_j4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	=F_b3	=M_b3	0	=F_c4	0	=M_c4	0	0	0	0	0	0	0	0	=F_j5	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	=B_b3	0	0	=B_c4	=M_c4	0	0	0	0	0	0	0	0	0	=B_j5	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=F_b4	=M_b4	0	=F_c5	0	=M_c5	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=B_b4	0	=B_j4	=B_c5	=-M_c5	0	0	0	0	-1	0	0	0	0	0	0
28	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=F_e	=-M_e	0	0	0	0	0
29	0	0	=F_j1	0	0	0	0	0	=F_j2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=B_e	0	=-M_e	0	0	0	0
30	0	0	=B_j1	0	0	0	0	0	=B_j2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=M_e	0	=-M_e	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	-1	0	0	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=F_e	=M_e	0	=F_j	=F_r	=M_r	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=B_e	0	=M_e	=B_j	=B_r	0	=M_r
34	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	=B_j	=1-F_j	0	0	0	0

Constraints

mill 1	1	=M_c1-M_b1-M_j1
	2	=M_c1*F_c1-M_b1*F_b1-M_j1*F_j1
	3	=M_c1*B_c1-M_b1*B_b1-M_j1*B_j1
mill 2	4	=M_c2-M_b2-M_j2
	5	=M_c2*F_c2-M_b2*F_b2-M_j2*F_j2
	6	=M_c2*B_c2-M_b2*B_b2-M_j2*B_j2
mill 3	7	=M_c3-M_b3-M_j3
	8	=M_c3*F_c3-M_b3*F_b3-M_j3*F_j3
	9	=M_c3*B_c3-M_b3*B_b3-M_j3*B_j3
mill 4	10	=M_c4-M_b4-M_j4
	11	=M_c4*F_c4-M_b4*F_b4-M_j4*F_j4
	12	=M_c4*B_c4-M_b4*B_b4-M_j4*B_j4
mill 5	13	=M_c5-M_b5-M_j5
	14	=M_c5*F_c5-M_b5*F_b5-M_j5*F_j5
	15	=M_c5*B_c5-M_b5*B_b5-M_j5*B_j5
A	16	=M_b1+M_j3-M_c2+M_r
	17	=M_b1*F_b1+M_j3*F_j3-N_c2*F_c2+M_r*F_r
	18	=M_b1*B_b1+M_j3*B_j3-M_c2*B_c2+M_r*B_r
B	19	=M_b2+M_j4-M_c3
	20	=M_b2*F_b2+M_j4*F_j4-N_c3*F_c3
	21	=M_b2*B_b2+M_j4*B_j4-M_c3*B_c3
C	22	=M_b3+M_j5-M_c4
	23	=M_b3*F_b3+M_j5*F_j5-N_c4*F_c4
	24	=M_b3*B_b3+M_j5*B_j5-M_c4*B_c4
D	25	=M_b4+I*M_c1*F_c1-M_c5
	26	=M_b4*F_b4-M_c5*F_c5
	27	=M_b4*B_b4-M_c5*B_c5
Mix	28	=M_j1+M_j2-M_e
	29	=M_j1*F_j1+M_j2*F_j2-M_e*F_e
	30	=M_j1*B_j1+M_j2*B_j2-M_e*B_e
Screen	31	=M_e-M_j-M_r
	32	=M_e*F_e-M_j*F_j-M_r*F_r
	33	=M_e*B_e-M_j*B_j-M_r*B_r
	34	=B_e-B_j-B_e*F_j+F_e*B_j

Figure 8. Constraints and Jacobian.

The inverse of the Jacobian was calculated *(<[1]mark area same shape as Jacobian>-fx-Math and trig-MINVERSE-OK-<Mark Jacobian>-ctrl OK[2])*. The result was labelled invJ.



	A	B	C	D	E	F	G
98		Delta				New	
99		Mb1	=MMULT(InvJ,phi)			Mb1	=C26-C99
100		Fb1	=MMULT(InvJ,phi)			Fb1	=C27-C100
101		Mj1	=MMULT(InvJ,phi)			Mj1	=C28-C101
102		Mc2	=MMULT(InvJ,phi)			Mc2	=C29-C102
103		Bc2	=MMULT(InvJ,phi)			Bc2	=C30-C103
104		Fc2	=MMULT(InvJ,phi)			Fc2	=C31-C104
105		Mb2	=MMULT(InvJ,phi)			Mb2	=C32-C105
106		Fb2	=MMULT(InvJ,phi)			Fb2	=C33-C106
107		Mj2	=MMULT(InvJ,phi)			Mj2	=C34-C107
108		Mc3	=MMULT(InvJ,phi)			Mc3	=C35-C108
109		Bc3	=MMULT(InvJ,phi)			Bc3	=C36-C109
110		Fc3	=MMULT(InvJ,phi)			Fc3	=C37-C110
111		Mb3	=MMULT(InvJ,phi)			Mb3	=C38-C111
112		Fb3	=MMULT(InvJ,phi)			Fb3	=C39-C112
113		Mj3	=MMULT(InvJ,phi)			Mj3	=C40-C113
114		Mc4	=MMULT(InvJ,phi)			Mc4	=C41-C114
115		Bc4	=MMULT(InvJ,phi)			Bc4	=C42-C115
116		Fc4	=MMULT(InvJ,phi)			Fc4	=C43-C116
117		Mb4	=MMULT(InvJ,phi)			Mb4	=C44-C117
118		Fb4	=MMULT(InvJ,phi)			Fb4	=C45-C118
119		Mj4	=MMULT(InvJ,phi)			Mj4	=C46-C119
120		Mc5	=MMULT(InvJ,phi)			Mc5	=C47-C120
121		Bc5	=MMULT(InvJ,phi)			Bc5	=C48-C121
122		Fc5	=MMULT(InvJ,phi)			Fc5	=C49-C122
123		Mb5	=MMULT(InvJ,phi)			Mb5	=C50-C123
124		Fb5	=MMULT(InvJ,phi)			Fb5	=C51-C124
125		Mj5	=MMULT(InvJ,phi)			Mj5	=C52-C125
126		Me	=MMULT(InvJ,phi)			Me	=C53-C126
127		Fe	=MMULT(InvJ,phi)			Fe	=C54-C127
128		Be	=MMULT(InvJ,phi)			Be	=C55-C128
129		Mj	=MMULT(InvJ,phi)			Mj	=C56-C129
130		Mr	=MMULT(InvJ,phi)			Mr	=C57-C130
131		Fr	=MMULT(InvJ,phi)			Fr	=C58-C131
132		Br	=MMULT(InvJ,phi)			Br	=C59-C132

Figure 9. δ and new estimate calculation.

The inverse of the Jacobian was multiplied by the constraint error vector to give step size *(<mark destination column vector with same number of rows as constraints>-fx-Math and trig-MMULT-OK-array 1-<mark inverse Jacobian>-array 2-<mark constraint vector>-ctrl OK).*

The result, in C99 to C132, was subtracted from the original estimate, in C26 to C59, and stored as a new value.

The new values that resulted were carried back to the solution vector via the 'if' statements

[1] Action description contained in < >
[2] Hold control key and click OK

shown in Figure 7. Initially, values had to be inserted into the solution vector. Setting the 'init' cell to 'TRUE' and pressing F9 to calculate did this. Iteration had to be enabled to calculate the answer and the criterion for convergence set to a maximum step size of 0.000001 (*Tools-Options-Calculation-<check iteration>-Max change-0.000001*). The value of the 'init' cell was set to 'FALSE' and F9 pressed to commence calculation of the solution. The calculation time was too short to measure on a 500 MHz P3 computer.