4. General Algebraic Equations

Many technical problems require the solution of polynomial equations, such as

$$x^4 + 3x^3 - 6x^2 + x - 3 = 0$$

or transcendental equations, such as

$$e^x - x^2 = 0$$

or simultaneous equations which do not lend themselves to simple solution by substitution, such as

$$3x^2 + 2y^2 - 2 = 0$$
$$x^2 - z/y = 0$$
$$y - z^2 - 2 = 0$$

A *general algebraic equation* is an equation of the form

$$g(x) = 0$$

which may be linear or nonlinear and which may contain transcendental functions. A system of general algebraic equations is a set of simultaneous equations of the form

$$g_1(x_1, x_2, \ldots, x_m) = 0$$
$$g_2(x_1, x_2, \ldots, x_m) = 0$$
$$\vdots$$
$$g_n(x_1, x_2, \ldots, x_m) = 0$$

or in vector notation

$$\bar{g}(\bar{x}) = 0$$

The roots of these equations, i.e., the values of x which satisfy them, may be determined by executing a statement having the typical form

FIND $x_1, x_2, \ldots, x_m$; **in** p; **BY** *solver* **TO MATCH** $g_1, g_2, \ldots, g_n$

where p is the MODEL which specifies the constraints $\overline{g(x)}$ which must be "matched to zero."

Example

The three simultaneous equations cited above may be solved by a program containing the following statements:

```
FIND X,Y,Z; IN THREE(X,Y,Z,GX,GY,GZ); BY AJAX;
     TO MATCH G1,G2,G3
```

```
MODEL THREE (X,Y,Z,G1,G2,G3)
    G1 = 3*X**2 + 2*Y**2 -2
    G2 = X**2 -Z/Y
    G3 = Y - Z**2 - 2
END
```

4.1. Implicit Equations

In any system of equations, some equations may be reducible to the form

$$x_k = f_k (x_1, ..., x_{k-1}, x_{k+1}, ..., x_m)$$

For example, in the simultaneous equations cited above, the equation

$$y - z^2 - 2 = 0$$

may be reduced to

$$y = z^2 + 2$$

Equations of this form are called explicit equations, since one of the unknowns may be computed directly in terms of the remaining unknowns of the simultaneous equation system. The remaining equations which are not reducible to this form are called *implicit equations*.

Whenever equations of a simultaneous system are separated into an explicit set and an implicit set, the "unknowns" computed by the explicit equations are no longer independent variables of the system. These unknowns and their associated constraints should be eliminated from the FIND statement, and there is no need to provide an initial estimate of their values. If no estimate is supplied, however, they must be computed in the model before they are used.

Example

The three simultaneous equations cited in the above example may be reduced to an explicit equation and two implicit equations and solved as follows:

```
    :
FIND X,Z; IN TWO (X,Z,G1,G2); BY AJAX; TO MATCH G1,G2
    :
MODEL TWO (X,Z,G1,G2)
    Y = Z**2 + 2
    G1 = 3*X**2 + 2*Y**2 -2
    G2 = X**2 - Z/Y
END
```

It is usually desirable to separate a set of simultaneous equations into an explicit set and an implicit set in the above manner in order to reduce the computer's solution task. However, if the algebraic reduction is difficult, much time may be saved by solving the equations in their original form.

4.2. AJAX - Newton-Gauss Equations Solver

AJAX is the nominal solver for general algebraic equations. It is invoked by a FIND statement which has the general form

FIND *unknowns*; **IN** *model-call*; **BY AJAX** {(*controller*)};
{{**WITH**}**LOWER**{**S**} *bottoms*;} {{**WITH**|**AND**} **UPPER**{**S**} *tops*;}
{**REPORTING** *names*} {**WITH FLAG** *signal* ;}
TO MATCH *constraints*

The *model-call* symbol is the model name optionally followed by an argument list as in the right-hand-side of a CALL statement. This form is not general, but is limited to certain solvers, including AJAX, MARS, HERA and THOR.

The optional clauses may appear in any order.

The meaning of the clauses in the FIND statement is defined in Section 1.3 and Section 2.2.2[1].

Examples

(1) **FIND X; IN POLYNOM; BY AJAX; TO MATCH F**

(2) **FIND X1, X2, X3; IN EQNS(SETUP);**
WITH LOWERS X1L,X2L,X3L; AND UPPERS X1H,X2H,X3H;
REPORTING BRAV, DIV, EXPONENT;
TO MATCH C

(3) **FIND VECTORX IN MANYEQS; BY AJAX(ACON);**
WITH FLAG TEST TO MATCH VECTORG

The FIND operator initiates an iterative calculation during which the gradients with respect to the *unknowns* are automatically evaluated for use by AJAX. Thus, the *unknowns* become the *independent variables* while the solution process is underway. During some of the iterations, first partial derivatives with respect to these unknowns are evaluated for all variables which depend on them. In particular, AJAX makes use of the Jacobian Matrix, which contains the gradients of the constraints with respect to

1 The flag variable signal is used ot signal the condition of solution as explained in section 2.2.5

the unknowns as its rows, i.e.,

$$J = \left[\frac{\partial \bar{g}}{\partial \bar{x}}\right] = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} \cdots \frac{\partial g_1}{\partial x_m} \\ \vdots \quad \vdots \\ \frac{\partial g_n}{\partial x_1} \cdots \frac{\partial g_n}{\partial x_m} \end{bmatrix}$$

Solution Process - AJAX solves the general algebraic equations by successive approximation of the unknowns. The model procedure is executed iteratively until the constraints are matched to zero within a specified tolerance. During iteration *i* of this process, the *ith* approximation $\bar{x}_{(i)}$ of the unknown vector $\bar{x}$ is computed according to the damped Newton formula

$$\bar{x}_{(i)} = \bar{x}_{(i-1)} - \beta J_{(i-1)}^{-1} \bar{g}_{(i-1)}$$

where $J_{(i-1)}$ is the Jacobian matrix evaluated during the *(i-1)th* iteration, and β is a damping factor (nominally unity - corresponding to the undamped Newton method).

AJAX will solve systems of equations which fall within any of the three classes of general algebraic equations:

(1) *Deterined systems* - equal number of equations and unknowns

(2) *Overdetermined systems* - more equations than unknowns

(3) *Underdetermined systems* - fewer equations than unknowns

Determined Systems - If the number of equations *n* is equal to the number of independent variables *m*, then the problem is said to be determined. In this case, the matrix J^{-1} is the true inverse of the Jacobian matrix J[2].

If the equations have more than one solution, as is usually true of nonlinear equations, the particular solution obtained will be the one nearest to the initial approximation of $x_p,\ldots,x_m$ (the values of these variables upon initial execution of the model procedure).

Overdetermined Systems - If the number of equations *n* is greater than the number of independent variables *m*, then the system is said to be overdetermined. In a true sense, no solution to this system exists. However, AJAX assumes that the n equations represent redundant approximations to m equations and solves the system by the

---

2 In the actual computation of AJAX, the inverse is not determined *per se*, rather $\Delta\bar{x} = \bar{x}_{(i)} - \bar{x}_{(i-1)}$ is computed by a process known as sequential orthogonalization. The net effect is the same as if the inverse were determined, but is more efficient.

method of least squares. Again, the particular solution obtained will be the one nearest to the initial approximation $x_p,...,x_m$. In this case, the matrix J^{-1} is the *pseudo-inverse*[3] of the Jacobian matrix.

Underdetermined or Ill-Conditioned Systems - If the number of equations n is fewer than the number of independent variables m, then the system is said to be *underdetermined.* The equivalent situation can also occur when $n = m$ if the equations are unstable (ill-conditioned). Loosely stated, a system of equations is numerically unstable if small changes in the coefficients of the equations produce large changes in the solution. Apparent changes in coefficient values could arise due to the precision of matrix inversion. For example, the coefficients of the following sets of linear equations differ by only a small amount, yet the solutions are entirely different:

	(a)	(b)
Equations:	x - y = 1	x - y = 1
	x - 1.00001y = 0	x - .9999y = 0
Solutions:	(100001, 100000)	(-99999, -100000)

In either case, underdetermined or ill-conditioned, an infinite number of solutions exist. However, by minimizing the Euclidean norm

$$||\bar{x}|| = \sqrt{x_1^2 + ... + x_m^2}$$

AJAX will find one of a finite number of smallest least squares solutions to the nonlinear equations. The particular solution obtained will be the one nearest to the initial approximation of $x_p,...,x_m$. In this case, the matrix J^{-1} is the pseudo-inverse of the Jacobian matrix J.

In essence the underdetermined case is a minimization problem in which the objective function

$$\sum_{i=1}^{m} x_i^2 + \sum_{j=1}^{n} g_j^2$$

is to be minimized.

Damping - In cases where the Newton method would normally diverge or oscillate (in the case of a periodic function), damping is applied in AJAX. Convergence problems arise when one or more components of the gradient of a constraint is small relative to

3 The pseudo-inverse of a matrix A is a matrix which has all of the relevant properties of a true inverse when used in place of a true inverse, even though A is singular or nonsquare, and the true inverse of A does not exist.

the constraint. The result can be inordinately large changes in the independent variables, thereby (possibly) moving too far away from the solution point to achieve convergence. Damping enforces convergence to a *stationary point* [4] on the surface of the squared norm of the constraints

$$||\bar{g}^2|| = g_1^2 + ... + g_n^2$$

This convergence is a necessary but insufficient condition for a solution to the constraint equations (all solutions are stationary points, but not all stationary points are solutions).

Damping is applied in a sub-iteration procedure, which computes the damping fraction

$$\beta = 1/2^n$$

to be applied to the Newton step if necessary. The exponent n is the smallest value that satisfies the damping inequality

$$2^n(1 - ||\bar{g}_{(i)}||^2/||\bar{g}_{(i-1)}||^2) \geq \text{DAMP}$$

where DAMP is a control variable whose nominal value is 0.2. At the end of each AJAX iteration, a Newton step, $\Delta x_{(i-1)} = -J_{(i-1)}^{-1}\bar{g}_{(i-1)}$ is computed and saved, and a *tentative* $\bar{x}_{(i)} = \bar{x}_{(i-1)} + \Delta\bar{x}_{(i-1)}$ is computed. Then, at the beginning of the next iteration, a sub-iteration of the model is executed[5] to compute the constraint vector $\bar{g}_{(i)}$ from the tentative $\bar{x}_{(i)}$. If the damping inequality is satisfied then the tentative step is accepted and the model is executed with derivative evaluation active to compute $J_{(i)}$ and complete the iteration. If the inequality is not satisfied, n is increased by one and the resulting β is used to halve the previous step resulting in a new $\bar{x}_{(i)} = \bar{x}_{(i-1)} + \beta\Delta\bar{x}_{(i-1)}$. This value is used to repeat the sub-iteration process until the damping inequality is satisfied or until $\beta < 10^{-6}$.

Damping is a very effective control procedure which should normally never be disabled (DAMP = 0) although smaller values of DAMP may be indicated for very flat constraint surfaces.

Limiting Stepping - An alternative control procedure may be employed to limit independent variable stepping, when the initial estimate is known to be good. This procedure is imposed by the control variable VLIMIT which prescribes the maximum change of any independent variable during an iteration. Thus it is only useful when all of the independent variables are of the same order of magnitude. The nominal value of VLIMIT (10^{50}) effectively disables this control.

---

4 A stationary point is a point where the gradient is zero.

5 Partial derivative evaluation is not active during the subiteration execution of the model, since the Jacobian computed in the last iteration still specifies the Newton search direction.

Linear Equations - In the case where the equations to be solved are linear, there is only one solution for each of the determined, overdetermined, or underdetermined/ill-conditioned cases. The linear solution is achieved in a single execution of the model block by AJAX.

Nesting - A FIND operation employing AJAX may appear in a model which itself is the operand of a FIND statement. In such a case, AJAX is said to be nested within the solver of the outer FIND operation. The solver of the outer FIND operation may also be AJAX, in which case AJAX is said to be self-nested.

AJAX may be self-nested to any level, or may be nested within other solvers to any level. This principle is illustrated in Section 5.5, pertaining to the solution of implicit differential equations, and in Section 6.3.1, pertaining to optimization with equality constraints.

Control Variables - The control variables for AJAX include the following[6]:

Variable	Description	Preset Value
DETAIL	**Detailed iteration report option**	**0**
	Value *Option*	
	0 No detailed report	
	1 Detailed report for every iteration	
	n Detailed report for iterations initial,n,2n,...,last	
SUMMARY	**Solution summary report option**	**1**
	Value *Option*	
	0 No summary report	
	1 Generate solution summary	
CONVERGE	**Convergence option**	**1**
	Value *Option*	
	1 Satisfy constraints convergence or unknowns convergence	
	2 Satisfy constraints convergence and unknowns convergence	
REMAX	**Maximum number of allowed iterations**	**20**
ZERO	**Zero tolerance for constraints convergence**	**10^{-6}**

6 These control variables also apply for the solver MARS (Section 4.3)

Variable	Description	Preset Value
DELTA	**Tolerance on change in unknowns for unknowns convergence**	10^{-7}
BREAKIN	**Interactive breakpoint option** *Value* *Option* 0 No breakpoints n Breakpoints after every nth iteration	**0**
VLIMIT	**Limit on the maximum change of the independent variables during a single iteration**	10^{50}
DAMP	**Criteria for imposing damping**	**0.2**
DETOUT	**Selection of detailed output medium** *Value* *Option* 0 Output to PRINTER (>0) Output to SCROLL (<0) Output to CONSOLE	**+1**
SUMOUT	**Selection of summary output medium** *Value* *Option* 0 Output to PRINTER (>0) Output to SCROLL (<0) Output to CONSOLE	**-1**
TAU	**Maximum relative error in the constraints being matched, i.e. max (error-G)/G.** TAU must be positive. TAU permits the specification of the approximate error in the values computed in the calculus model, for example errors due to measurement uncertainty in a data fitting problem. This value is used to hold computational error to a point where it is negligible in comparison to the a priori error. The value is also used to determine the rank of the problem, i.e. the larger the value of TAU the more likely that the problem is rank deficient (ill-conditioned or singular). Obviously, an unreasonably small value for TAU may obscure the true nature of the problem and at the same time provoke unnecessarily extended computation.	**0.001**

Variable	Description	Preset Value
PREDAMP	**Arbitrary damping factor to be applied to the first iteration. (0 < PREDAMP ≤ 1)** PREDAMP is used only when damping is applied (control variable DAMP nonzero). The damping scheme involves a comparison of successive estimates of the norm of the constraints and hence cannot be applied to the initial estimate. Thus, there is some possibility of an inordinately large Newton step on the first iteration. When this is anticipated, perhaps from a previous attempt at a solution, PREDAMP may be used to limit the first step.	1

Convergence Criteria - The convergence criteria for AJAX consist of either or both of the criteria *constraints convergence* or *unknowns convergence* as specified by the control variable **CONVERGE**. *Constraints convergence* is satisfied if

$$\max |g_1 \ldots g_n|$$

where $g_1 \ldots g_n$ are the constraint variables to be matched to zero. *Unknowns convergence* is satisfied if

$$\max |\Delta x_1/x_1 \ldots \Delta x_m/x_m| < \textbf{DELTA}$$

where $x_1, \ldots, x_m$ are the unknowns of the FIND statement and $\Delta x_1, \ldots, \Delta x_m$ are the changes in these unknowns from the previous iteration.

Printed Reports - AJAX automatically accumulates, formats, and issues printed reports whenever specified by the control variables **SUMMARY** and **DETAIL**. To illustrate the formats of these reports, consider the computed solution of the system of nonlinear equation

$$y_1 = 12.5 - 3x_1x_2 - x_3 = 0$$

$$y_2 = 3.317 - \sin(x_1) - \exp(x_2) = 0$$

$$y_3 = 1.609 - x_2 \ln(x_3) = 0$$

with initial guesses $x_1 = x_2 = x_3 = 2$.

The FC program for this problem is shown in Figure 4-1. In this program, the statement DETAIL = 1 in the controller SET will generate a detailed iteration report for every iteration. Also, because of the preset value of the control variable SUMMARY (= 1), the solution summary report is generated after the FIND operation is complete.

```
PROBLEM SIMUL
  DIMENSION X(3),Y(3)
  DATA X/2,2,2/
  FIND X; IN EQS(X,Y); BY AJAX(SET); TO MATCH Y
END
MODEL EQS(X,Y)
  DIMENSION X(*),Y(*)
  Y(1) = 12.5 - 3*X(1)*X(2) - X(3)
  Y(2) = 3.317 - SIN(X(1)) - EXP(X(2))
  Y(3) = 1.609 - X(2)*LOG(X(3))
END
 CONTROLLER SET(AJAX)
  DETAIL=1
END
```

FIG. 4-1 Simultaneous Equations Solving Program

Detailed Iteration Report - An example of the detailed iteration report is given in Figure 4-2. While the information provided in this report is not essential to effective solving of equations, it is sometimes helpful in understanding peculiarities of particular problems.

```
----AJAX ITERATION 2 INVOKED AT SIMUL[3] FOR MODEL EQS

    JACOBIAN MATRIX [DG(I)/DX(J)]
                  (   1)          (   2)          (   3)
    (   1)   -3.212651E+00  -7.885323E+00  -1.000000E+00
    (   2)    8.712017E-01  -2.917956E+00   0.
    (   3)    0.            -1.330697E+00  -2.830629E-01

    JACOBIAN (SCALED)
                  (   1)          (   2)          (   3)
    (   1)   -3.747326E-01  -9.197662E-01  -1.166428E-01
    (   2)    2.860867E-01  -9.582037E-01   0.
    (   3)    0.            -9.781209E-01  -2.080372E-01

    RANK ..... 3

    UNKNOWNS - X
      2.628441E+00   1.070884E+00   3.783681E+00

    DELTA-X  (NEWTON STEP FOR NEXT ITERATION)
     -6.671173E-02  -5.140623E-02   8.917324E-01

    ABSOLUTE DELTA-X / X
      2.538072E-02   4.800356E-02   2.356785E-01

    CONSTRAINTS - (||G|| = 3.410349E-01)
      2.720562E-01  -9.188176E-02   1.839781E-01

    CONVERGENCE CONDITION
       UNKNOWNS NOT CONVERGED
       CONSTRAINTS UNSATISFIED
       SPECIFIED CRITERIA UNSATISFIED

----END AJAX ITERATION 2
```

FIG. 4-2 Example of Detailed Iteration Report

The Jacobian matrix is printed as first computed during the iteration, and as scaled[7] for solution. The zeros in J_{23} and J_{31} indicate that the constraint y_2 is not dependent upon x_3 and the constraint y_3 is not dependent upon x_1. These facts are also apparent from the equations themselves, but might not be so if the equations were slightly more complex.

The rank of the Jacobian matrix, as determined by AJAX, is also printed. If this number is less than the number of rows or columns, then the Jacobian matrix is singular[8]. The remaining information of the detailed print gives the current results of the unknowns, their changes and fractional changes from the previous iteration, and the values of the constraints corresponding to the previous unknowns.

Solution Summary Report - An example of the solution summary report is given in Figure 4-2. Most of the time this report gives an adequate record of progress toward a solution. The names of the unknowns and constraints are printed, and their successive values are printed for each iteration until convergence is achieved. The REPORTING phrase may be used to add variables which are neither independent variables nor constraints to the report so that their successive values may be traced.

Limitations of AJAX - The Newton procedure employed by AJAX presumes that the constraint functions being solved have continuous first derivatives with respect to the unknowns. If this does not hold in some region, then AJAX will encounter difficulty in that region and may not converge to a solution.

Convergence using Newton's method is sometime quite sensitive to the initial guesses provided for the unknowns. If these values are very far from a solution, then convergence may not be achieved, or the solution obtained may not be physically realistic. In such cases it is desirable to experiment with several starting points until a realistic solution is obtained. If this kind of experimentation fails to produce useful results, then the next recourse is to change the formulation in order to simplify it numerically.

4.3. MARS - Newton-Householder Equations Solver

MARS provides an alternative to AJAX for problems involving the solution of systems of algebraic equations. Both AJAX and MARS employ a damped Newton iterative scheme and hence both automatically invoke the evaluation of first partial derivatives in the calculus model.

---

7 Scaling is achieved by dividing each element by the magnitude of the largest gradient vector (row).

8 If the matrix is not square, yet the rank is less than the minimum of the number of rows or columns, then the square submatrix partition is also singular.

```
----AJAX SUMMARY, INVOKED AT SIMUL[3] FOR MODEL EQS
    CONVERGENCE CONDITION
       UNKNOWNS CONVERGED
       CONSTRAINTS SATISFIED
       ALL SPECIFIED CRITERIA SATISFIED

  LOOP NUMBER .........   [INITIAL]          1              2
  UNKNOWNS
            X (  1)       2.000000E+00  2.294485E+00  2.628441E+00
            X (  2)       2.000000E+00  1.342432E+00  1.070884E+00
            X (  3)       2.000000E+00  2.678497E+00  3.783681E+00
  CONSTRAINTS
            Y (  1)      -1.500000E+00  5.809312E-01  2.720562E-01
            Y (  2)      -4.981354E+00 -1.260712E+00 -9.188176E-02
            Y (  3)       2.227056E-01  2.863608E+01  1.839781E-01

  LOOP NUMBER .........   [INITIAL]          3              4
  UNKNOWNS
            X (  1)       2.000000E+00  2.561729E+00  2.506922E+00
            X (  2)       2.000000E+00  1.019477E+00  1.001978E+00
            X (  3)       2.000000E+00  4.675413E+00  4.967233E+00
  CONSTRAINTS
            Y (  1)      -1.500000E+00 -1.028819E-02 -2.877234E-03
            Y (  2)      -4.981354E+00 -2.655155E-03 -4.237323E-04
            Y (  3)       2.227056E-01  3.664223E-02  2.966356E-03

  LOOP NUMBER .........   [INITIAL]          5              6
  UNKNOWNS
            X (  1)       2.000000E+00  2.500841E+00  2.500787E+00
            X (  2)       2.000000E+00  1.000336E+00  1.000323E+00
            X (  3)       2.000000E+00  4.994990E+00  4.995219E+00
  CONSTRAINTS
            Y (  1)      -1.500000E+00 -2.996827E-05 -2.094964E-09
            Y (  2)      -4.981354E+00  7.323269E-06  6.219238E-10
            Y (  3)       2.227056E-01  2.473896E-05  1.643819E-09

----END OF LOOP SUMMARY
```

FIG. 4-3 Example of Solution Summary Report

The range of application of the two solvers is roughly identical. Both are suitable for under-, over-, and exactly-determined problems and both may be used when ill-conditioning of singularities reduce the rank of the problem. The principal distinction arises in the very common area of overdetermined systems, examples of which include regression analysis, parameter estimation, and process identification, or more generally, least-squares optimization. AJAX addresses such problems by forming the normal equations and then solving the resulting square system by taking the pseudo-inverse through an orthogonalization method. Conversely, MARS directly solves the rectangular system by a QR-decomposition technique employing householder elementary transformations. This method involves rather more computation than AJAX, although there is a compensating increase in precision.

The major benefits of MARS are twofold. First, it is a more stable computational scheme and is less impacted by the errors arising in large and/or poorly scaled problems. More importantly, it is far less subject to difficulty when the problem is ill-conditioned.

Since the condition number of the normal system is the square of that of the basic problem, overdetermined systems are often ill-conditioned when the method of AJAX is used. Another area of advantage to MARS involves boundary value problems in which the equations are very sensitive to the boundary conditions.

The FIND statements for MARS and AJAX are identical and both employ the same control variables.

A point to note in using MARS is that the influence of control variable TAU is somewhat different from the case of AJAX due to their different algorithms. Effectively, MARS takes a less conservative view of computational error, as is consistent with its relative stability. Thus, for the same value of TAU, MARS may converge more rapidly than AJAX but with a larger solution norm. When necessary, tighter solutions may be obtained by reducing DELTA and/or TAU. For overdetermined systems, control variable ZERO has little influence since constraint convergence is rarely possible.

4.4. Inequality Constraints

In applications of solvers MARS and AJAX, it is frequently necessary to bound the allowable values of the independent variables, both in the final solution and during the iterative solution process. This need most commonly arises in least-squares optimization problems, i.e. data fitting and parameter estimation, and for these, one recourse is to use one of the constrained optimization solvers, such as JUPITER or THOR. When the required bounds are constant, however, the simpler and more efficient methods of MARS and AJAX may be used to specify constant inequality constraints.

Some typical uses for this feature are:

- A regression problem in which the curve must have a positive ordinate intercept;
- The solution of a set of equations which are mathematically invalid for some values of the unknowns, e.g. because of negative square roots, etc.;
- The solution of implicit equations with multiple roots, for which the root closest to the initial guess is required.

Constant inequality constraints are specified by UPPERS and LOWERS clauses in the FIND statement.

Example

The following program will fit a cubic to a set of 20 data points and require that the curve must be non-negative at $x = 0$.

```
      PROBLEM FIT
        DIMENSION X(20), Y(20), DIFF(20)
C     INPUT DATA AND INITIAL CONDITIONS FOR THE UNKNOWN COEFFICIENTS
C     OF THE CURVE, I.E. A,B,C,D SUCH THAT Y=A*X**3+B*X**2+C*X+D
C     SET UP LOWER LIMITS ALO,BLO,CLO,DLO
        DATA/ALO,BLO,CLO,DLO/3*-1E50,0/
        READ *,X,Y
C     COMPUTE AND OUTPUT THE SOLUTION
        FIND A,B,C,D IN CUBIC(A,B,C,D,X,Y,DIFF); BY MARS;
~         WITH LOWERS ALO,BLO,CLO,DLO;
~         TO MATCH DIFF
      END

      MODEL CUBIC(A,B,C,D,X,Y,DIFF)
        DIMENSION X(*),Y(*),DIFF(*)
        DO 10 I=1,20
          DIFF(I)=Y(I)-((A*X(I)+B)*X(I)+C)*X(I)+D
10      CONTINUE
      END
```

Note the way in which the limits were specified. When any unknown is limited, limits must be set for all (The limits are associated with the unknowns in left-to-right sequence.) In this case, the limits on A, B, and C were set essentially at "infinity". An alternate, and more certain, approach would be to specify upper limits also and to develop the limits for the unconstrained unknowns A, B, and C.

4.5. Systems of Inequalities

The use of slack variables is a common technique in linear programming for reducing an arbitrary linear program to standard form. That is, slack variables are used to convert inequality constraints into equality constraints.

To see how this approach works, consider the following linear inequality

$$\sum_{i=1}^{n} a_i x_i \geq 0$$

Since the left hand side is greater than or equal to zero there exists a nonnegative quantity, z, such that

$$\sum_{i=1}^{n} a_i x_i - z = 0$$

The nonnegative quantity z is called a slack variable. It is required to be nonnegative only by convention and practice; however, by adhering to this, much confusion over choice of signs can be avoided.

The same technique also works in nonlinear programming; in fact, its application is even more straightforward than in the linear case. Obviously, in the latter case, the variable z was required to enter linearly; otherwise the linearity of the original problem would be lost. Hence, in the linear case, an additional constraint $z \geq 0$, must always be

explicitly adjoined to the problem. In the nonlinear case, the slack variable need not be linear (although it may be if so desired). In particular, if the slack variables enter the inequalities as squared terms then these terms are automatically nonnegative and the additional constraints, required in the linear program, are not present in the nonlinear case.

Consider the set of general algebraic equations of the form

$$h_1(x_1, x_2, \ldots, x_m) = 0$$
$$h_2(x_1, x_2, \ldots, x_m) = 0$$
$$\vdots$$
$$h_j(x_1, x_2, \ldots, x_m) = 0$$

and the set of inequalities

$$g_1(x_1, x_2, \ldots, x_m) \geq 0$$
$$g_2(x_1, x_2, \ldots, x_m) \geq 0$$
$$\vdots$$
$$g_k(x_1, x_2, \ldots, x_m) \geq 0$$

Subtracting l slack variables brings the system of inequalities into the form

$$g_1(x_1, x_2, \ldots, x_m) - z_1 = 0$$
$$g_2(x_1, x_2, \ldots, x_m) - z_2 = 0$$
$$\vdots$$
$$g_k(x_1, x_2, \ldots, x_m) - z_k = 0$$

Thus the original system of j equalities and k inequalities in m unknowns has been transformed into a system of j + k equalities in m + k unknowns. The advantage gained from this transformation is that pseudo-inverse methods can be used to directly solve the transformed system. In FC, this can be done quite easily employing the solver, AJAX. The transformed system is typically non-square, however, since AJAX can solve over- or underdetermined systems this poses no additional difficulties.

Finding Feasible Solutions - Slack variable techniques can also be used to achieve initial feasibility (or maintain it) in inequality constrained nonlinear programs. Such applications generally lead to an underdetermined system of equations. It was pointed out earlier that underdetermined systems have no unique solution. Thus, the solution to a set of inequalities is merely the nearest of a set of points in a feasible subspace of $(x_1,\ldots,x_n)$ bounded by the inequality constraints. However, finding any point in this subspace, i.e., any feasible point, is an important initial step in constrained optimization, or may be employed as a substep of an unconstrained optimization search in order to solve a constrained optimization problem.

The following approach can be used to solve systems of inequalities of the above form.

```
GLOBAL ALL
PROBLEM CONSTRAINTS
  DYNAMIC X,Z,G
  READ *,N,M : ALLOT X(N),Z(M),G(M)
  <Z>=1
  FIND X,Z; IN SLACK; BY AJAX; TO MATCH G
END

MODEL SLACK
  DYNAMIC X,Z,G
  CALL INEQLS
  <ZSQ>=<Z>*<Z>
  <G>=<G>+<ZSQ>
END

MODEL INEQLS
  DYNAMIC X,G
  G(1)=... -Z(1)
    :
  G(M)=... -Z(M)
END
```

In this code, the slack variables are initialized to unity. This is not necessary in general, but for certain functions, and particular choices of initial guesses for the original n variables, the resulting Jacobian matrix may be exactly the zero matrix unless the slack variables are given nonzero initial values.

A further consideration in the use of slack variables is that it is often necessary to change the convergence criteria of AJAX, especially for very large problems. Experience in using this technique has shown that both the relative change in the function (i.e., the values of the constraints) and in the independent variables must be less than the required tolerance (the control variable ZERO). The default criterion for AJAX is that only one or the other of these be less than zero. If this is permitted, it is possible for the change in the independent variables to become less than the tolerance even though the constraints are not satisfied. Thus, it is advisable to set CONVERGE = 2 in a CONTROLLER block for AJAX.

4.6. Multiple Roots of Nonlinear Equations

Another use of the slack variable technique is to permit the application of AJAX to the problem of finding multiple roots of nonlinear systems. One of the main difficulties of this problem is that of avoiding a root already found, during the search for the next root. In principle, this can be accomplished quite easily. The trick is simply to enclose each root, as it is found, in an n-sphere, and then constrain further searches to regions outside the n-sphere (or for more than two roots outside the union of n-spheres). The resulting constraints are necessarily inequalities, having the form

$$\sum_{i=1}^{n} (x_{0i} - x_i)^2 \geq r^2$$

where

$x_0 = (x_{01}, ..., x_{0n})$ is the known root

$x = (x_1, ..., x_n)$ is an arbitrary point, and

r is the radius of the n-sphere

The inequality can be put into the form of an equality constraint by adding the slack variable z:

$$r^2 - \sum_{i=1}^{n} (x_{0i} - x_i)^2 + z^2 = 0 \qquad (1)$$

If the original system consisted of the equations

$$\begin{aligned} g_1(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ g_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \qquad (2)$$

The new system, used to find the second root, is

$$\begin{aligned} g_1(x_1, x_2, \ldots, x_n) &= 0 \\ &\vdots \\ g_n(x_1, x_2, \ldots, x_n) &= 0 \\ g_{n+1}(x_1, x_2, \ldots, x_{n+1}) &= 0 \end{aligned} \qquad (3)$$

This system is still a determined system, just as the original system was.

The following code may be used to solve a system of the form (2), by automatically converting it to the form (3), as successive roots are found.

```
      PROBLEM ROOTS
        COMMON/CONS/KROOT,N,RADIUS,X0
        DYNAMIC X,G,X0
        READ *,N,RADIUS,NUMROOTS
        ALLOT X(N),G(N)
        READ *,(X(J),J=1,N)
        KROOT=0
10      FIND X; IN FN(X,G); BY AJAX; TO MATCH G
        IF(KROOT.LT.NUMROOTS-1) THEN
          KROOT=KROOT+1
          ALLOT X(N+KROOT),G(N+KROOT),X0(KROOT,N)
          DO 20 J=1,N
            X0(KROOT,J)=X(J)
            X(J)=X0(KROOT,J)+2*RADIUS
20        CONTINUE
          X(N+KROOT)=1
          GOTO 10
        ELSE
          STOP
        ENDIF
      END
```

```
      MODEL FN(X,G)
        COMMON /CONS/KROOT,N,RADIUS,X0
        DYNAMIC X0,X,G
        IF(KROOT.GT.0) CALL CONSTRAINTS(X,G)
        G(1)= ...
          :
        G(N)= ...
      END

      MODEL CONSTRAINTS(X,G)
        COMMON /CONS/KROOT,N,RADIUS,X0
        DYNAMIC X0,X,G
        DO 10 I=1,KROOT
          G(N+I)=RADIUS**2+X(N+I)**2
          DO 10 J=1,N
            G(N+I)=G(N+I)-(X0(I,J)-X(J))**2
10      CONTINUE
      END
```