SAS/OR® 15.1 User's Guide
Mathematical Programming
The Nonlinear Programming Solver

This document is an individual chapter from *SAS/OR® 15.1 User's Guide: Mathematical Programming*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2018. *SAS/OR® 15.1 User's Guide: Mathematical Programming*. Cary, NC: SAS Institute Inc.

SAS/OR® 15.1 User's Guide: Mathematical Programming

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

November 2018

Chapter 11
The Nonlinear Programming Solver

Contents

Overview: NLP Solver **531**
Getting Started: NLP Solver **533**
Syntax: NLP Solver **545**
  Functional Summary 545
  NLP Solver Options 546
Details: NLP Solver **553**
  Basic Definitions and Notation 553
  Constrained Optimization 554
  Interior Point Algorithm 555
  Interior Point Direct Algorithm 557
  Active-Set Method 559
  Multistart 561
  Covariance Matrix 562
  Iteration Log for the Local Solver 565
  Iteration Log for Multistart 565
  Solver Termination Criterion 566
  Solver Termination Messages 567
  Macro Variable _OROPTMODEL_ 567
Examples: NLP Solver **570**
  Example 11.1: Solving Highly Nonlinear Optimization Problems 570
  Example 11.2: Solving Unconstrained and Bound-Constrained Optimization Problems 572
  Example 11.3: Solving Equality-Constrained Problems 574
  Example 11.4: Solving NLP Problems with Range Constraints 576
  Example 11.5: Solving Large-Scale NLP Problems 579
  Example 11.6: Solving NLP Problems That Have Several Local Minima 581
  Example 11.7: Maximum Likelihood Weibull Estimation 587
  Example 11.8: Finding an Irreducible Infeasible Set 589
References **594**

Overview: NLP Solver

The sparse nonlinear programming (NLP) solver is a component of the OPTMODEL procedure that can solve optimization problems containing both nonlinear equality and inequality constraints. The general nonlinear optimization problem can be defined as

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & h_i(x) = 0, i \in \mathcal{E} = \{1, 2, \dots, p\} \\ & g_i(x) \geq 0, i \in \mathcal{I} = \{1, 2, \dots, q\} \\ & l \leq x \leq u \end{array}$$

where $x \in \mathbb{R}^n$ is the vector of the decision variables; $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function; $h : \mathbb{R}^n \mapsto \mathbb{R}^p$ is the vector of equality constraints—that is, $h = (h_1, \dots, h_p)$; $g : \mathbb{R}^n \mapsto \mathbb{R}^q$ is the vector of inequality constraints—that is, $g = (g_1, \dots, g_q)$; and $l, u \in \mathbb{R}^n$ are the vectors of the lower and upper bounds, respectively, on the decision variables.

It is assumed that the functions f, h_i, and g_i are twice continuously differentiable. Any point that satisfies the constraints of the NLP problem is called a *feasible point*, and the set of all those points forms the feasible region of the NLP problem—that is, $\mathcal{F} = \{x \in \mathbb{R}^n : h(x) = 0, g(x) \geq 0, l \leq x \leq u\}$.

The NLP problem can have a unique minimum or many different minima, depending on the type of functions involved. If the objective function is convex, the equality constraint functions are linear, and the inequality constraint functions are concave, then the NLP problem is called a convex program and has a unique minimum. All other types of NLP problems are called nonconvex and can contain more than one minimum, usually called *local minima*. The solution that achieves the lowest objective value of all local minima is called the *global minimum* or *global solution* of the NLP problem. The NLP solver can find the unique minimum of convex programs and a local minimum of a general NLP problem. In addition, the solver is equipped with specific options that enable it to locate the global minimum or a good approximation of it, for those problems that contain many local minima.

The NLP solver implements the following primal-dual methods for finding a local minimum:

- interior point trust-region line-search algorithm
- interior point line-search algorithm, which uses direct linear algebra
- active-set trust-region line-search algorithm

These three methods can solve small-, medium-, and large-scale optimization problems efficiently and robustly, and they use exact first and second derivatives to calculate search directions. The memory requirements of all algorithms are reduced dramatically because only nonzero elements of matrices are stored. Convergence of all algorithms is achieved either (1) by using a trust-region line-search framework that guides the iterations toward the optimal solution or (2) by simply performing a backtracking line search. For the first convergence technique, if a trust-region subproblem fails to provide a suitable step of improvement, a line search is then used to fine-tune the trust-region radius and ensure sufficient decrease in objective function and constraint violations. The second convergence technique is a backtracking line search on a primal-dual merit function.

The NLP solver implements two primal-dual interior point algorithms. Both methods use barrier functions to ensure that the algorithm remains feasible with respect to the bound constraints. Interior point methods are extremely useful when the optimization problem contains many inequality constraints and you suspect that most of these constraints will be satisfied as strict inequalities at the optimal solution.

The active-set technique implements an active-set algorithm in which only the inequality constraints that are satisfied as equalities, together with the original equality constraints, are considered. Once that set of constraints is identified, active-set algorithms typically converge faster than interior point algorithms. They converge faster because the size and the complexity of the original optimization problem can be reduced if only few constraints need to be considered.

For optimization problems that contain many local optima, the NLP solver can be run in multistart mode. If the multistart mode is specified, the solver samples the feasible region and generates a number of starting points. Then the local solvers can be called from each of those starting points to converge to different local optima. The local minimum with the smallest objective value is then reported back to the user as the optimal solution.

The NLP solver implements many powerful features that are obtained from recent research in the field of nonlinear optimization algorithms (Akrotirianakis and Rustem 2005; Armand, Gilbert, and Jan-Jégou 2002; Armand and Omheni 2017a, b; Erway, Gill, and Griffin 2007; Forsgren and Gill 1998; Vanderbei 1999; Wächter and Biegler 2006; Yamashita 1998). The term *primal-dual* means that the algorithm iteratively generates better approximations of the decision variables x (usually called *primal* variables) in addition to the dual variables (also referred to as Lagrange multipliers). At every iteration, the algorithm uses a modified Newton's method to solve a system of nonlinear equations. The modifications made to Newton's method are implicitly controlled by the current trust-region radius. The solution of that system provides the direction and the steps along which the next approximation of the local minimum is searched. The active-set algorithm ensures that the primal iterations are always within their bounds—that is, $l \leq x^k \leq u$, for every iteration k. However, the interior approach relaxes this condition by using slack variables, and intermediate iterations might be infeasible.

Finally, for parameter estimation problems such as least squares, maximum likelihood, or Bayesian estimation problems, the NLP solver can calculate the covariance matrix after it successfully obtains parameter estimates.

Getting Started: NLP Solver

The NLP solver consists of three techniques that can solve a wide class of optimization problems efficiently and robustly. This section presents three examples that introduce the three techniques of NLP. The examples also introduce basic features of the modeling language of PROC OPTMODEL that is used to define the optimization problem.

The NLP solver can be invoked using the SOLVE statement,

SOLVE WITH NLP < / *options* > **;**

where *options* specify the technique name, termination criteria, and how to display the results in the iteration log. For a detailed description of the *options*, see the section "NLP Solver Options" on page 546.

A Simple Problem

Consider the following simple example of a nonlinear optimization problem:

$$\begin{array}{ll} \text{minimize} & f(x) = (x_1 + 3x_2 + x_3)^2 + 4(x_1 - x_2)^2 \\ \text{subject to} & x_1 + x_2 + x_3 = 1 \\ & 6x_2 + 4x_3 - x_1^3 - 3 \geq 0 \\ & x_i \geq 0, i = 1, 2, 3 \end{array}$$

The problem consists of a quadratic objective function, a linear equality constraint, and a nonlinear inequality constraint. The goal is to find a local minimum, starting from the point $x^0 = (0.1, 0.7, 0.2)$. You can use the following call to PROC OPTMODEL to find a local minimum:

```
proc optmodel;
   var x{1..3} >= 0;
   minimize f = (x[1] + 3*x[2] + x[3])**2 + 4*(x[1] - x[2])**2;

   con constr1: sum{i in 1..3}x[i] = 1;
   con constr2: 6*x[2] + 4*x[3] - x[1]**3 - 3 >= 0;

   /* starting point */
   x[1] = 0.1;
   x[2] = 0.7;
   x[3] = 0.2;

   solve with NLP;
   print x;
quit;
```

Because no options have been specified, the default solver (INTERIORPOINT) is used to solve the problem. The SAS output displays a detailed summary of the problem along with the status of the solver at termination, the total number of iterations required, and the value of the objective function at the best feasible solution that was found. The summaries and the returned solution are shown in Figure 11.1.

Figure 11.1 Problem Summary, Solution Summary, and the Returned Solution

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Quadratic
Number of Variables	3
Bounded Above	0
Bounded Below	3
Bounded Below and Above	0
Free	0
Fixed	0
Number of Constraints	2
Linear LE (<=)	0
Linear EQ (=)	1
Linear GE (>=)	0
Linear Range	0
Nonlinear LE (<=)	0
Nonlinear EQ (=)	0
Nonlinear GE (>=)	1
Nonlinear Range	0

Figure 11.1 *continued*

Solution Summary	
Solver	NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Best Feasible
Objective Value	1.0000158715
Optimality Error	0.1041603358
Infeasibility	2.4921244E-8
Iterations	5
Presolve Time	0.00
Solution Time	0.01

[1]	x
1	0.0000162497
2	0.0000039553
3	0.9999798200

The SAS log shown in Figure 11.2 displays a brief summary of the problem being solved, followed by the iterations that are generated by the solver.

Figure 11.2 Progress of the Algorithm as Shown in the Log

```
NOTE: Problem generation will use 16 threads.
NOTE: The problem has 3 variables (0 free, 0 fixed).
NOTE: The problem has 1 linear constraints (0 LE, 1 EQ, 0 GE, 0 range).
NOTE: The problem has 3 linear constraint coefficients.
NOTE: The problem has 1 nonlinear constraints (0 LE, 0 EQ, 1 GE, 0 range).
NOTE: The OPTMODEL presolver removed 0 variables, 0 linear constraints, and 0
      nonlinear constraints.
NOTE: Using analytic derivatives for objective.
NOTE: Using analytic derivatives for nonlinear constraints.
NOTE: The NLP solver is called.
NOTE: The Interior Point algorithm is used.
                        Objective                              Optimality
           Iter             Value     Infeasibility                 Error
              0        7.20000000                 0            6.40213404
              1        1.22115550        0.00042385            0.00500000
              2        1.00188693        0.00003290            0.00480263
              3        1.00275609        0.00002123            0.00005000
              4        1.00001702   0.0000000252254            0.00187172
              5        1.00001738   0.0000000250883       0.0000005000000
NOTE: Optimal.
NOTE: Objective = 1.000017384.
NOTE: Objective of the best feasible solution found = 1.0000158715.
NOTE: The best feasible solution found is returned.
NOTE: To return the local optimal solution found, set the SOLTYPE= option to 0.
```

A Least Squares Problem

Although the NLP solver does not implement techniques that are specialized for least squares problems, this example illustrates how the NLP solver can solve least squares problems by using general nonlinear optimization techniques.

The Bard function, $f(x)$, (Moré, Garbow, and Hillstrom 1981) is a least squares problem with $n = 3$ parameters and $m = 15$ functions f_i:

$$f(x) = \frac{1}{2}\sum_{i=1}^{15} f_i^2(x), \quad x = (x_1, x_2, x_3)$$

where the functions are defined as

$$f_i(x) = y_i - \left(x_1 + \frac{u_i}{v_i x_2 + w_i x_3} \right)$$

with $u_i = i$, $v_i = 16 - i$, $w_i = \min(u_i, v_i)$, and

$$y = (0.14, 0.18, 0.22, 0.25, 0.29, 0.32, 0.35, 0.39, 0.37, 0.58, 0.73, 0.96, 1.34, 2.10, 4.39)$$

Starting from $x^0 = (1, 1, 1)$, you can reach the minimum at the point $(0.08, 1.13, 2.34)$, with corresponding objective value $f(x^*)$ = 4.107E–3. You can use the following SAS code to formulate and solve this problem:

```
proc optmodel;
   set S = 1..15;
   number u{i in S} = i;
   number v{i in S} = 16 - i;
   number w{i in S} = min(u[i], v[i]);
   number y{S} = [ .14 .18 .22 .25 .29 .32 .35 .39 .37 .58
                   .73 .96 1.34 2.10 4.39 ];
   var x{1..3} init 1;
   min f = 0.5 * sum{i in S} ( y[i] -
              ( x[1] + u[i]/(v[i] * x[2] + w[i] * x[3]) )
           )^2;
   solve with nlp / algorithm=ipdirect;
   print x;
quit;
```

The output that summarizes the problem characteristics and the solution that the IPDIRECT solver obtains are displayed in Figure 11.3.

Figure 11.3 Problem Summary, Solution Summary, and Returned Solution

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Nonlinear
Number of Variables	3
Bounded Above	0
Bounded Below	0
Bounded Below and Above	0
Free	3
Fixed	0
Number of Constraints	0

Solution Summary	
Solver	NLP
Algorithm	Interior Point Direct
Objective Function	f
Solution Status	Optimal
Objective Value	0.0041074387
Optimality Error	3.1901663E-8
Infeasibility	0
Iterations	7
Presolve Time	0.00
Solution Time	0.00

[1]	x
1	0.082411
2	1.133036
3	2.343696

The SAS log shown in Figure 11.4 displays a brief summary of the problem being solved, followed by the iterations that are generated by the solver.

Figure 11.4 Progress of the Algorithm as Shown in the Log

```
NOTE: Problem generation will use 16 threads.
NOTE: The problem has 3 variables (3 free, 0 fixed).
NOTE: The problem has 0 linear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The problem has 0 nonlinear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The OPTMODEL presolver removed 0 variables, 0 linear constraints, and 0
      nonlinear constraints.
NOTE: Using analytic derivatives for objective.
NOTE: The NLP solver is called.
NOTE: The experimental Interior Point Direct algorithm is used.
                       Objective                          Optimality
           Iter            Value     Infeasibility             Error
              0      20.84084793                 0       25.93561876
              1       2.25897658                 0        6.07201621
              2       0.64185534                 0        2.33605324
              3       0.11641732                 0        0.76836910
              4       0.01174864                 0        0.17396096
              5       0.00416426                 0        0.01403811
              6       0.00410744                 0        0.00022897
              7       0.00410744                 0   0.0000000319017
NOTE: Optimal.
NOTE: Objective = 0.0041074387.
```

A Larger Optimization Problem

Consider the following larger optimization problem:

$$
\begin{array}{ll}
\text{minimize} & f(x) = \sum_{i=1}^{1000} x_i y_i + \frac{1}{2}\sum_{j=1}^{5} z_j^2 \\
\text{subject to} & x_k + y_k + \sum_{j=1}^{5} z_j = 5, \text{ for } k = 1, 2, \ldots, 1000 \\
& \sum_{i=1}^{1000} (x_i + y_i) + \sum_{j=1}^{5} z_j \geq 6 \\
& -1 \leq x_i \leq 1, i = 1, 2, \ldots, 1000 \\
& -1 \leq y_i \leq 1, i = 1, 2, \ldots, 1000 \\
& 0 \leq z_i \leq 2, i = 1, 2, \ldots, 5
\end{array}
$$

The problem consists of a quadratic objective function, 1,000 linear equality constraints, and a linear inequality constraint. There are also 2,005 variables. The goal is to find a local minimum by using the ACTIVESET technique. This can be accomplished by issuing the following call to PROC OPTMODEL:

```
proc optmodel;
   number n = 1000;
   number b = 5;
   var x{1..n} >= -1 <= 1 init  0.99;
   var y{1..n} >= -1 <= 1 init -0.99;
   var z{1..b} >=  0 <= 2 init  0.5;
   minimize f = sum {i in 1..n} x[i] * y[i] + sum {j in 1..b} 0.5 * z[j]^2;
   con cons1{k in 1..n}: x[k] + y[k] + sum {j in 1..b} z[j] = b;
   con cons2: sum {i in 1..n} (x[i] + y[i]) + sum {j in 1..b} z[j] >= b + 1;
```

```
      solve with NLP / algorithm=activeset logfreq=10;
   quit;
```

The SAS output displays a detailed summary of the problem along with the status of the solver at termination, the total number of iterations required, and the value of the objective function at the local minimum. The summaries are shown in Figure 11.5.

Figure 11.5 Problem Summary and Solution Summary

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Quadratic
Number of Variables	2005
Bounded Above	0
Bounded Below	0
Bounded Below and Above	2005
Free	0
Fixed	0
Number of Constraints	1001
Linear LE (<=)	0
Linear EQ (=)	1000
Linear GE (>=)	1
Linear Range	0

Solution Summary	
Solver	NLP
Algorithm	Active Set
Objective Function	f
Solution Status	Optimal
Objective Value	-996.4999999
Optimality Error	3.9546178E-7
Infeasibility	8.8315E-9
Iterations	10
Presolve Time	0.00
Solution Time	0.18

The SAS log shown in Figure 11.6 displays a brief summary of the problem that is being solved, followed by the iterations that are generated by the solver.

Figure 11.6 Progress of the Algorithm as Shown in the Log

```
NOTE: Problem generation will use 16 threads.
NOTE: The problem has 2005 variables (0 free, 0 fixed).
NOTE: The problem has 1001 linear constraints (0 LE, 1000 EQ, 1 GE, 0 range).
NOTE: The problem has 9005 linear constraint coefficients.
NOTE: The problem has 0 nonlinear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The OPTMODEL presolver removed 0 variables, 0 linear constraints, and 0
      nonlinear constraints.
NOTE: Using analytic derivatives for objective.
NOTE: Using 2 threads for nonlinear evaluation.
NOTE: The NLP solver is called.
NOTE: The Active Set algorithm is used.
                        Objective                                Optimality
           Iter             Value     Infeasibility                   Error
              0     -979.47500000        3.50000000              0.14142857
             10     -996.49999991   0.0000000088312      0.0000003954618
NOTE: Optimal.
NOTE: Objective = -996.4999999.
```

An Optimization Problem with Many Local Minima

Consider the following optimization problem:

$$\begin{array}{lcl} \text{minimize } f(x) & = & e^{\sin(50x)} + \sin(60e^{y}) + \sin(70\sin(x)) + \sin(\sin(80y)) \\ & & -\sin(10(x+y)) + (x^2+y^2)/4 \\ \text{subject to} & & -1 \le x \le 1 \\ & & -1 \le y \le 1 \end{array}$$

The objective function is highly nonlinear and contains many local minima. The NLP solver provides you with the option of searching the feasible region and identifying local minima of better quality. This is achieved by writing the following SAS program:

```
proc optmodel;
  var x >= -1 <= 1;
  var y >= -1 <= 1;
  min f = exp(sin(50*x)) + sin(60*exp(y)) + sin(70*sin(x)) + sin(sin(80*y))
          - sin(10*(x+y)) + (x^2+y^2)/4;
  solve with nlp / multistart=(maxstarts=30) seed=94245;
quit;
```

The MULTISTART=() option is specified, which directs the algorithm to start the local solver from many different starting points. The SAS log is shown in Figure 11.7.

Figure 11.7 Progress of the Algorithm as Shown in the Log

```
NOTE: Problem generation will use 16 threads.
NOTE: The problem has 2 variables (0 free, 0 fixed).
NOTE: The problem has 0 linear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The problem has 0 nonlinear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The OPTMODEL presolver removed 0 variables, 0 linear constraints, and 0
      nonlinear constraints.
NOTE: Using analytic derivatives for objective.
NOTE: The NLP solver is called.
NOTE: The Interior Point algorithm is used.
NOTE: The MULTISTART option is enabled.
NOTE: The deterministic parallel mode is enabled.
NOTE: The Multistart algorithm is executing in single-machine mode.
NOTE: The Multistart algorithm is using up to 16 threads.
NOTE: Random number seed 94245 is used.
                    Best       Local  Optimality    Infeasi-  Local  Local
      Start    Objective   Objective       Error      bility  Iters  Status
          1   -2.0415721  -2.0415721        5E-7           0      3  Optimal
          2   -2.0415721  -1.9515416        5E-7           0      5  Optimal
          3    -2.311349   -2.311349        5E-7           0      3  Optimal
          4   -2.6355024  -2.6355024        5E-7           0      4  Optimal
          5   -2.6355024  -2.3850402        5E-7           0      3  Optimal
          6   -2.6355024  -1.3696639        5E-7           0      5  Optimal
          7   -2.7358255  -2.7358255        5E-7           0      3  Optimal
          8   -2.7358255  -1.5435267        5E-7           0      4  Optimal
          9   -2.7358255  -1.9746745        5E-7           0      4  Optimal
         10   -2.7358255  -1.6502615        5E-7           0      3  Optimal
         11   -2.7358255  -1.8895721        5E-7           0      3  Optimal
         12   -2.7358255   -2.233731        5E-7           0      5  Optimal
         13   -2.7358255  -1.5173191        5E-7           0      3  Optimal
         14   -2.7358255  -1.0893788        5E-7           0      5  Optimal
         15   -2.7358255  -1.9014544  6.62491E-7  6.62491E-7      3  Optimal
         16   -2.7358255  0.06801522        5E-9           0      4  Optimal
         17   -2.7358255  -1.3833451        5E-7           0      4  Optimal
         18   -2.7358255  -2.2203943        5E-7           0      4  Optimal
         19   -2.7358255  -2.5334314        5E-7           0      4  Optimal
         20   -2.7358255  -1.3316067        5E-7           0      3  Optimal
         21   -2.7358255  -2.1510179        5E-7           0      3  Optimal
         22   -2.7358255  -1.4915074        5E-7           0      4  Optimal
         23   -3.1440794  -3.1440794        5E-7           0      3  Optimal
         24   -3.1440794  -2.0402058        5E-7           0      4  Optimal
         25   -3.1440794  -3.0626257        5E-7           0      4  Optimal
         26   -3.1440794  -1.1233222        5E-7           0      3  Optimal
         27   -3.1440794  -2.6634391        5E-7           0      5  Optimal
         28   -3.1440794  -1.9508557        5E-7           0      4  Optimal
         29   -3.1440794  -1.9921598        5E-7           0      3  Optimal
         30   -3.3068686  -3.3068686        5E-7           0      4  Optimal
NOTE: The Multistart algorithm generated 640 sample points.
NOTE: 30 distinct local optima were found.
NOTE: The best objective value found by local solver = -3.306868647.
```

Figure 11.7 *continued*

```
NOTE: The solution found by local solver with objective = -3.306868647 was
      returned.
```

The SAS log presents additional information when the MULTISTART=() option is specified. The first column counts the number of restarts of the local solver. The second column records the best local optimum that has been found so far, and the third through sixth columns record the local optimum to which the solver has converged. The final column records the status of the local solver at every iteration.

The SAS output is shown in Figure 11.8.

Figure 11.8 Problem Summary and Solution Summary

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Nonlinear
Number of Variables	2
Bounded Above	0
Bounded Below	0
Bounded Below and Above	2
Free	0
Fixed	0
Number of Constraints	0

Solution Summary	
Solver	Multistart NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Optimal
Objective Value	-3.306868647
Number of Starts	30
Number of Sample Points	640
Number of Distinct Optima	30
Random Seed Used	94245
Optimality Error	5E-7
Infeasibility	0
Presolve Time	0.00
Solution Time	0.06

A Least Squares Estimation Problem for a Regression Model

The following data are used to build a regression model:

```
data samples;
   input x1 x2 y;
   datalines;
 4  8   43.71
62  5  351.29
81 62 2878.91
85 75 3591.59
65 54 2058.71
96 84 4487.87
98 29 1773.52
36 33  767.57
30 91 1637.66
 3 59  215.28
62 57 2067.42
11 48  394.11
66 21  932.84
68 24 1069.21
95 30 1770.78
34 14  368.51
86 81 3902.27
37 49 1115.67
46 80 2136.92
87 72 3537.84
;
```

Suppose you want to compute the parameters in your regression model based on the preceding data, and the model is

$$L(a, b, c) = a * x1 + b * x2 + c * x1 * x2$$

where a, b, c are the parameters that need to be found.

The following PROC OPTMODEL call specifies the least squares problem for the regression model:

```
/* Reqression model with interactive term: y = a*x1 + b*x2 + c*x1*x2 */
proc optmodel;
   set obs;
   num x1{obs}, x2{obs}, y{obs};
   num mycov{i in 1.._nvar_, j in 1..i};
   var a, b, c;
   read data samples into obs=[_n_] x1 x2 y;
   impvar Err{i in obs} = y[i] - (a*x1[i]+b*x2[i]+c*x1[i]*x2[i]);
   min f = sum{i in obs} Err[i]^2;
   solve with nlp/covest=(cov=5 covout=mycov);
   print mycov;
   print a b c;
quit;
```

The solution is displayed in Figure 11.9.

Figure 11.9 Least Squares Problem Estimation Results

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Quadratic
Number of Variables	3
Bounded Above	0
Bounded Below	0
Bounded Below and Above	0
Free	3
Fixed	0
Number of Constraints	0

Solution Summary	
Solver	NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Optimal
Objective Value	7.1862967833
Optimality Error	8.638471E-11
Infeasibility	0
Iterations	18
Presolve Time	0.00
Solution Time	0.01

mycov

	1	**2**	**3**
1	0.0000047825		
2	-.0000000996	0.0000032426	
3	-.0000000676	-.0000000442	0.0000000017

a	**b**	**c**
3.0113	2.0033	0.4998

Syntax: NLP Solver

The following PROC OPTMODEL statement is available for the NLP solver:

SOLVE WITH NLP < / *options* > ;

Functional Summary

Table 11.1 summarizes the options that can be used with the SOLVE WITH NLP statement.

Table 11.1 Options for the NLP Solver

Description	Option
Covariance Matrix Options and Suboptions	
Requests that the NLP solver compute a covariance matrix	COVEST=()
Specifies an absolute singularity criterion for matrix inversion	ASINGULAR=
Specifies the type of covariance matrix	COV=
Specifies the name of the output covariance matrix	COVOUT=
Specifies the tolerance for deciding whether a matrix is singular	COVSING=
Specifies a relative singularity criterion for matrix inversion	MSINGULAR=
Specifies a number for calculating the divisor for the covariance matrix when VARDEF=DF	NDF=
Specifies a number for calculating the scale factor for the covariance matrix	NTERMS=
Specifies a scalar factor for computing the covariance matrix	SIGSQ=
Specifies the divisor for calculating the covariance matrix	VARDEF=
Miscellaneous Option	
Specifies the seed to use to generate random numbers	SEED=
Multistart Options	
Directs the local solver to start from multiple initial points	MULTISTART=()
Specifies the maximum range of values that each variable can take during the sampling process	BNDRANGE=
Specifies the tolerance for local optima to be considered distinct	DISTTOL=
Specifies the amount of printing solution progress in multistart mode	LOGLEVEL=
Specifies the time limit in multistart mode	MAXTIME=
Specifies the maximum number of starting points to be used by the multistart algorithm	MAXSTARTS=
Optimization Option	
Specifies the optimization technique	ALGORITHM=

Table 11.1 Options for the NLP Solver (continued)

Description	Option
Output Options	
Specifies the frequency of printing solution progress (local solvers)	LOGFREQ=
Specifies the allowable types of output solution	SOLTYPE=
Solver Options	
Specifies the feasibility tolerance	FEASTOL=
Specifies the type of Hessian used by the solver	HESSTYPE=
Enables or disables IIS detection with respect to linear constraints and variable bounds	IIS=
Specifies the maximum number of iterations	MAXITER=
Specifies the time limit for the optimization process	MAXTIME=
Specifies the upper limit on the objective	OBJLIMIT=
Specifies the convergence tolerance	OPTTOL=
Specifies whether the solver is allowed to shift the user-supplied initial point to be interior to the bounds	PRESERVEINIT=
Specifies units of CPU time or real time	TIMETYPE=

NLP Solver Options

This section describes the options that are recognized by the NLP solver. These options can be specified after a forward slash (/) in the SOLVE statement, provided that the NLP solver is explicitly specified using a WITH clause.

Covariance Matrix Options

COVEST=(*suboptions*)

requests that the NLP solver produce a covariance matrix. When this option is applied, the following PROC OPTMODEL options are automatically set: PRESOLVER=NONE and SOLTYPE=0. For more information, see the section "Covariance Matrix" on page 562.

You can specify the following *suboptions*:

ASINGULAR=*asing*

specifies an absolute singularity criterion for measuring the singularity of the Hessian and crossproduct Jacobian and their projected forms, which might have to be inverted to compute the covariance matrix. The value of *asing* can be any number between the machine precision and the largest positive number representable in your operating environment. The default is the square root of the machine precision. For more information, see the section "Covariance Matrix" on page 562.

COV=*number* | *string*

specifies one of six formulas for computing the covariance matrix. The formula that is used depends on the type of objective (MIN or LSQ) that is specified. Table 11.2 describes the valid values for this option and their corresponding formulas, where *nterms* is the value of the

NTERMS= option and MIN, LSQ, and other symbols are defined in the section “Covariance Matrix” on page 562.

Table 11.2 Values of COV= Option

number	*string*	**MIN Objective**	**LSQ Objective**
1	M	$(\mathit{nterms}/d)G^{-1}JJ(f)G^{-1}$	$(\mathit{nterms}/d)G^{-1}VG^{-1}$
2	H	$(\mathit{nterms}/d)G^{-1}$	$\sigma^2 G^{-1}$
3	J	$(1/d)W^{-1}$	$\sigma^2 JJ(f)^{-1}$
4	B	$(1/d)G^{-1}WG^{-1}$	$\sigma^2 G^{-1}JJ(f)G^{-1}$
5	E	$(\mathit{nterms}/d)JJ(f)^{-1}$	$(1/d)V^{-1}$
6	U	$(\mathit{nterms}/d)W^{-1}JJ(f)W^{-1}$	$(\mathit{nterms}/d)JJ(f)^{-1}VJJ(f)^{-1}$

For MAX type problems, the covariance matrix is converted to MIN type by using negative Hessian, Jacobian, and function values in the computation. For more information, see the section “Covariance Matrix” on page 562.

By default, COV=2.

COVOUT=*parameter*

specifies the name of the parameter that contains the output covariance matrix. Because a covariance matrix is symmetric, you should declare the covariance matrix as either a lower-triangular matrix or a square matrix with indexes starting from 1. For example:

```
num mycov{i in 1..N, j in 1..i};   /* a  lower triangular matrix */
```

or

```
num mycov{i in 1..N, j in 1..N};   /* a square matrix */
```

where N is the number of variables.

Depending on the type of output covariance matrix, the solver updates either the lower-triangular matrix or the full square matrix. If you declare the covariance matrix as neither a lower-triangular matrix nor a square matrix, or if the indexes do not start from 1, the NLP solver issues an error message. You can use the CREATE DATA statement to output the results to a SAS data set. For more information, see the section “Covariance Matrix” on page 562.

COVSING=*covsing*

specifies a threshold, $\mathit{covsing} > 0$, that determines whether to consider the eigenvalues of a matrix to be 0. The value of *covsing* can be any number between the machine precision and the largest positive number representable in your operating environment. The default is set internally by the algorithm. For more information, see the section “Covariance Matrix” on page 562.

MSINGULAR=*msing*

specifies a relative singularity criterion *msing* > 0 for measuring the singularity of the Hessian and crossproduct Jacobian and their projected forms. The value of *msing* can be any number between machine precision and the largest positive number representable in your operating environment. The default is 1E–12. For more information, see the section "Covariance Matrix" on page 562.

NDF=*ndf*

specifies a number to be used in calculating the divisor d, which is used in calculating the covariance matrix when VARDEF=DF. The value of *ndf* can be any positive integer up to the largest four-byte signed integer, which is $2^{31} - 1$. The default is the number of optimization variables in the objective function. For more information, see the section "Covariance Matrix" on page 562.

NTERMS=*nterms*

specifies a number to be used in calculating the scale factor for the covariance matrix, as shown in Table 11.2. The value of *nterms* can be any positive integer up to the largest four-byte signed integer, which is $2^{31} - 1$. The default is the number of nonconstant terms in the objective function. For more information, see the section "Covariance Matrix" on page 562.

SIGSQ=*sq*

specifies a real scalar factor, *sq* > 0, for computing the covariance matrix. The value of *sq* can be any number between the machine precision and the largest positive number representable in your operating environment. For more information, see the section "Covariance Matrix" on page 562.

VARDEF=DF | N

controls how the divisor d is calculated. This divisor is used in calculating the covariance matrix and approximate standard errors. The value of d also depends on the values of the NDF= and NTERMS= options, *ndf* and *nterms*, respectively, as follows:

$$d = \begin{cases} \max(1, \textit{nterms} - \textit{ndf}) & \text{for VARDEF=DF} \\ \textit{nterms} & \text{for VARDEF=N} \end{cases}$$

By default, VARDEF=DF if the SIGSQ= option is not specified; otherwise, by default VARDEF=N. For more information, see the section "Covariance Matrix" on page 562.

Miscellaneous Option

SEED=*N*

specifies a positive integer to be used as the seed for generating random number sequences. You can use this option to replicate results from different runs.

Multistart Options

MULTISTART=(*suboptions*)

MS=(*suboptions*)

enables multistart mode. In this mode, the local solver solves the problem from multiple starting points, possibly finding a better local minimum as a result. This option is disabled by default. For more information about multistart mode, see the section "Multistart" on page 561.

You can specify the following *suboptions*:

BNDRANGE=*M*

defines the range from which each variable can take values during the sampling process. This option affects only the sampling process that determines starting points for the local solver. It does not affect the bounds of the original nonlinear optimization problem. More specifically, if the *i*th variable x_i has lower and upper bounds ℓ_i and u_i, respectively (that is, $\ell_i \le x_i \le u_i$), then an initial point is generated by a sampling process as follows:

For each sample point x, the *i*th coordinate x_i is generated so that the following bounds hold, where x_i^0 is the default starting point or a specified starting point:

$$\begin{array}{ll} l_i \le x_i \le u_i & \text{if } l_i \text{ and } u_i \text{ are both finite} \\ l_i \le x_i \le l_i + M & \text{if only } l_i \text{ is finite} \\ u_i - M \le x_i \le u_i & \text{if only } u_i \text{ is finite} \\ x_i^0 - M/2 \le x_i \le x_i^0 + M/2 & \text{otherwise} \end{array}$$

The default value is 200 in a shared-memory computing environment and 1,000 in a distributed computing environment.

DISTTOL=ϵ

defines the tolerance by which two optimal points are considered distinct. Optimal points are considered distinct if the Euclidean distance between them is at least ϵ. The default is 1.0E–6.

LOGLEVEL=*number*

PRINTLEVEL=*number*

defines the amount of information that the multistart algorithm displays in the SAS log. Table 11.3 describes the valid values of this suboption.

Table 11.3 Values for LOGLEVEL= Suboption

number	**Description**
0	Turns off all solver-related messages to SAS log
1	Displays multistart summary information when the algorithm terminates
2	Displays multistart iteration log and summary information when the algorithm terminates
3	Displays the same information as LOGLEVEL=2 and might display additional information

By default, LOGLEVEL=2.

MAXTIME=*T*

defines the maximum allowable time *T* (in seconds) for the NLP solver to locate the best local optimum in multistart mode. The value of the TIMETYPE= option determines the type of units that are used. The time that is specified by the MAXTIME= suboption is checked only once after the completion of the local solver. Because the local solver might be called many times, the maximum time that is specified for multistart is recommended to be greater than the maximum time specified for the local solver. If you do not specify this option, the multistart algorithm does not stop based on the amount of time elapsed.

MAXSTARTS=*N*

defines the maximum number of starting points to be used for local optimization. That is, there will be no more than *N* local optimization calls in the multistart algorithm. You can specify *N* to be any nonnegative integer. When *N* = 0, the algorithm uses the default value of this option. In a shared-memory computing environment, the default value is 100. In a distributed computing environment, the default value is a number proportional to the number of threads across all the grid nodes (usually more than 100).

Optimization Options

ALGORITHM=*keyword*

TECHNIQUE=*keyword*

TECH=*keyword*

SOLVER=*keyword*

specifies the optimization technique to be used to solve the problem. The following *keywords* are valid:

INTERIORPOINT

uses a primal-dual interior point method. This technique is recommended for both small- and large-scale nonlinear optimization problems. This is the preferred solver if the problem includes a large number of inactive constraints.

IPDIRECT (experimental)

uses a primal-dual interior point augmented Lagrangian method. The use of direct factorizations and other enhancements can reduce both the number of iterations and the CPU time for many problem types.

ACTIVESET

uses a primal-dual active-set method. This technique is recommended for both small- and large-scale nonlinear optimization problems. This is the preferred solver if the problem includes only bound constraints or if the optimal active set can be quickly determined by the solver.

CONCURRENT

runs the INTERIORPOINT and ACTIVESET techniques in parallel, with one thread using the INTERIORPOINT technique and the other thread using the ACTIVESET technique. The solution is returned by the first method that terminates.

The default is INTERIORPOINT.

Output Options

LOGFREQ=*N*

PRINTFREQ=*N*

specifies how often the iterations are displayed in the SAS log. *N* should be an integer between zero and the largest four-byte, signed integer, which is $2^{31} - 1$. If $N \geq 1$, the solver prints only those iterations that are a multiple of *N*. If $N = 0$, no iteration is displayed in the log. The default value is 1.

SOLTYPE=0 | 1

specifies the type of solution to return:

0	returns a locally optimal solution, provided that the solver locates one.
1	returns the best feasible solution found, provided that its objective value is better than that of the locally optimal solution found. You cannot specify this value if you call the experimental IPDIRECT solver (that is, if ALGORITHM=IPDIRECT).

By default, SOLTYPE=1.

Solver Options

FEASTOL=ϵ

defines the feasible tolerance. The solver exits if the constraint violation is less than ϵ and the scaled optimality conditions are less than the value of the OPTTOL= option. By default, FEASTOL=1E–6.

HESSTYPE=FULL | PRODUCT

specifies the type of Hessian for the solver to use:

FULL	uses a full Hessian. In this case, the algorithm can create a better preconditioner to solve the problem in less CPU time.
PRODUCT	uses only Hessian-vector products, not the full Hessian. When the solver uses only Hessian-vector products to find a search direction, it usually uses much less memory, especially when the problem is large and the Hessian is not sparse. You cannot specify this option when you specify ALGORITHM=IPDIRECT because the experimental IPDIRECT solver needs the full expression of the Hessian matrix.

By default, SOLTYPE=FULL.

IIS=*number* | *string*

specifies whether the NLP solver attempts to identify a set of linear constraints and variables that form an irreducible infeasible set (IIS). Table 11.4 describes the valid values of the IIS= option.

Table 11.4 Values for IIS= Option

number	*string*	**Description**
0	OFF	Disables IIS detection
1	ON	Enables IIS detection. All other NLP solver options are ignored except the following: FEASTOL=, LOGFREQ=, LOGLEVEL=, MAX-ITER=, MAXTIME=, and TIMETYPE=.

By default, IIS=OFF.

The NLP solver ignores nonlinear constraints, if any, and invokes the LP solver's algorithm to attempt to identify an IIS. If an IIS is found, information about the infeasibilities can be found in the .status suffix values of the constraints and variables. For more information about the IIS= option, see the section "Irreducible Infeasible Set" on page 275 of Chapter 7, "The Linear Programming Solver." Also see Example 11.8 for an example that demonstrates the use of the IIS= option of the NLP solver.

MAXITER=*N*

specifies that the solver take at most *N* major iterations to determine an optimum of the NLP problem. The value of *N* is an integer between zero and the largest four-byte, signed integer, which is $2^{31} - 1$. A major iteration in NLP consists of finding a descent direction and a step size along which the next approximation of the optimum resides. The default is 5,000 iterations.

MAXTIME=*t*

specifies an upper limit of *t* units of time for the optimization process, including problem generation time and solution time. The value of the TIMETYPE= option determines the type of units used. If you do not specify the MAXTIME= option, the solver does not stop based on the amount of time elapsed. The value of *t* can be any positive number; the default value is the positive number that has the largest absolute value that can be represented in your operating environment.

OBJLIMIT=*M*

specifies an upper limit on the magnitude of the objective value. For a minimization problem, the algorithm terminates when the objective value becomes less than –*M*; for a maximization problem, the algorithm stops when the objective value exceeds *M*. The algorithm stopping implies that either the problem is unbounded or the algorithm diverges. If optimization were allowed to continue, numerical difficulty might be encountered. The default is *M*=1E+20. The minimum acceptable value of *M* is 1E+8. If the specified value of *M* is less than 1E+8, the value is reset to the default value 1E+20.

OPTTOL=ϵ

RELOPTTOL=ϵ

defines the measure by which you can decide whether the current iterate is an acceptable approximation of a local minimum. The value of this option is a positive real number. The NLP solver determines that the current iterate is a local minimum when the norm of the scaled vector of the optimality conditions is less than ϵ and the true constraint violation is less than FEASTOL. The default is ϵ=1E–6.

PRESERVEINIT=*number* | *string*

specifies whether the solver is allowed to shift the user-supplied initial point to be interior to the bounds. Table 11.5 describes the valid values of the PRESERVEINIT= option.

Table 11.5 Values for PRESERVEINIT= Option

number	*string*	**Description**
0	OFF	Shifts the user-supplied initial point to be interior to the bounds
1	ON	Uses the user-supplied initial point without shifting

This option can be used only when calling the INTERIORPOINT algorithm. By default, PRESERVEINIT=OFF.

TIMETYPE=*number* | *string*

specifies the units of time used by the MAXTIME= option and reported by the PRESOLVE_TIME and SOLUTION_TIME terms in the _OROPTMODEL_ macro variable. Table 11.6 describes the valid values of the TIMETYPE= option.

Table 11.6 Values for TIMETYPE= Option

number	*string*	**Description**
0	CPU	Specifies units of CPU time
1	REAL	Specifies units of real time

The "Optimization Statistics" table, an output of PROC OPTMODEL if you specify PRINTLEVEL=2 in the PROC OPTMODEL statement, also includes the same time units for Presolver Time and Solver Time. The other times (such as Problem Generation Time) in the "Optimization Statistics" table are also in the same units.

The default value of the TIMETYPE= option depends on various options. When the solver is used with distributed or multithreaded processing, then by default TIMETYPE= REAL. Otherwise, by default TIMETYPE= CPU. Table 11.7 describes the detailed logic for determining the default; the first context in the table that applies determines the default value.

Table 11.7 Default Value for TIMETYPE= Option

Context	**Default**
Solver is invoked in an OPTMODEL COFOR loop	REAL
NODES= value is nonzero for multistart mode	REAL
NTHREADS= value is greater than 1	REAL
NTHREADS= 1	CPU

Details: NLP Solver

This section presents a brief discussion about the algorithmic details of the NLP solver. First, the notation is defined. Next, an introduction to the fundamental ideas in constrained optimization is presented; the main point of the second section is to present the necessary and sufficient optimality conditions, which play a central role in all optimization algorithms. The section concludes with a general overview of primal-dual interior point and active-set algorithms for nonlinear optimization. A detailed treatment of the preceding topics can be found in Nocedal and Wright (1999), Wright (1997), and Forsgren, Gill, and Wright (2002).

Basic Definitions and Notation

The gradient of a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the vector of all the first partial derivatives of f and is denoted by

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)^{\mathrm{T}}$$

where the superscript T denotes the transpose of a vector.

The Hessian matrix of f, denoted by $\nabla^2 f(x)$, or simply by $H(x)$, is an $n \times n$ symmetric matrix whose (i, j) element is the second partial derivative of $f(x)$ with respect to x_i and x_j. That is, $H_{i,j}(x) = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$.

Consider the vector function, $c : \mathbb{R}^n \mapsto \mathbb{R}^{p+q}$, whose first p elements are the equality constraint functions $h_i(x), i = 1, 2, \ldots, p$, and whose last q elements are the inequality constraint functions $g_i(x), i = 1, 2, \ldots, q$. That is,

$$c(x) = (h(x) : g(x))^{\mathrm{T}} = (h_1(x), \ldots, h_p(x) : g_1(x), \ldots, g_q(x))^{\mathrm{T}}$$

The $(p+q) \times n$ matrix whose ith row is the gradient of the ith element of $c(x)$ is called the Jacobian matrix of $c(x)$ (or simply the Jacobian of the NLP problem) and is denoted by $J(x)$. You can also use $J_h(x)$ to denote the $p \times n$ Jacobian matrix of the equality constraints and use $J_g(x)$ to denote the $q \times n$ Jacobian matrix of the inequality constraints.

Constrained Optimization

A function that plays a pivotal role in establishing conditions that characterize a local minimum of an NLP problem is the Lagrangian function $\mathcal{L}(x, y, z)$, which is defined as

$$\mathcal{L}(x, y, z) = f(x) - \sum_{i \in \mathcal{E}} y_i h_i(x) - \sum_{i \in \mathcal{I}} z_i g_i(x)$$

Note that the Lagrangian function can be seen as a linear combination of the objective and constraint functions. The coefficients of the constraints, $y_i, i \in \mathcal{E}$, and $z_i, i \in \mathcal{I}$, are called the Lagrange multipliers or dual variables. At a feasible point $\hat{x}$, an inequality constraint is called active if it is satisfied as an equality—that is, $g_i(\hat{x}) = 0$. The set of active constraints at a feasible point $\hat{x}$ is then defined as the union of the index set of the equality constraints, $\mathcal{E}$, and the indices of those inequality constraints that are active at $\hat{x}$; that is,

$$\mathcal{A}(\hat{x}) = \mathcal{E} \cup \{i \in \mathcal{I} : g_i(\hat{x}) = 0\}$$

An important condition that is assumed to hold in the majority of optimization algorithms is the so-called linear independence constraint qualification (LICQ). The LICQ states that at any feasible point $\hat{x}$, the gradients of all the active constraints are linearly independent. The main purpose of the LICQ is to ensure that the set of constraints is well-defined in a way that there are no redundant constraints or in a way that there are no constraints defined such that their gradients are always equal to zero.

The First-Order Necessary Optimality Conditions

If x^* is a local minimum of the NLP problem and the LICQ holds at x^*, then there are vectors of Lagrange multipliers y^* and z^*, with components $y_i^*, i \in \mathcal{E}$, and $z_i^*, i \in \mathcal{I}$, respectively, such that the following conditions are satisfied:

$$\begin{array}{rcll} \nabla_x \mathcal{L}(x^*, y^*, z^*) & = & 0 & \\ h_i(x^*) & = & 0, & i \in \mathcal{E} \\ g_i(x^*) & \geq & 0, & i \in \mathcal{I} \\ z_i^* & \geq & 0, & i \in \mathcal{I} \\ z_i^* g_i(x^*) & = & 0, & i \in \mathcal{I} \end{array}$$

where $\nabla_x \mathcal{L}(x^*, y^*, z^*)$ is the gradient of the Lagrangian function with respect to x, defined as

$$\nabla_x \mathcal{L}(x^*, y^*, z^*) = \nabla f(x) - \sum_{i \in \mathcal{E}} y_i \nabla h_i(x) - \sum_{i \in \mathcal{I}} z_i \nabla g_i(x)$$

The preceding conditions are often called the *Karush-Kuhn-Tucker (KKT) conditions*. The last group of equations ($z_i g_i(x) = 0, i \in I$) is called the complementarity condition. Its main aim is to try to force the Lagrange multipliers, z_i^*, of the inactive inequalities (that is, those inequalities with $g_i(x^*) > 0$) to zero.

The KKT conditions describe the way the first derivatives of the objective and constraints are related at a local minimum x^*. However, they are not enough to fully characterize a local minimum. The second-order optimality conditions attempt to fulfill this aim by examining the curvature of the Hessian matrix of the Lagrangian function at a point that satisfies the KKT conditions.

The Second-Order Necessary Optimality Condition

Let x^* be a local minimum of the NLP problem, and let y^* and z^* be the corresponding Lagrange multipliers that satisfy the first-order optimality conditions. Then $d^{\mathrm{T}} \nabla_x^2 \mathcal{L}(x^*, y^*, z^*) d \geq 0$ for all nonzero vectors d that satisfy the following conditions:

1. $\nabla h_i^{\mathrm{T}}(x^*) d = 0, \forall i \in \mathcal{E}$
2. $\nabla g_i^{\mathrm{T}}(x^*) d = 0, \forall i \in \mathcal{A}(x^*) \cap \mathcal{I}$, such that $z_i^* > 0$
3. $\nabla g_i^{\mathrm{T}}(x^*) d \geq 0, \forall i \in \mathcal{A}(x^*) \cap \mathcal{I}$, such that $z_i^* = 0$

The second-order necessary optimality condition states that, at a local minimum, the curvature of the Lagrangian function along the directions that satisfy the preceding conditions must be nonnegative.

Interior Point Algorithm

Primal-dual interior point methods can be classified into two categories: feasible and infeasible. The first category requires that the starting point and all subsequent iterations of the algorithm strictly satisfy all the inequality constraints. The second category relaxes those requirements and allows the iterations to violate some or all of the inequality constraints during the course of the minimization procedure. The NLP solver implements an infeasible algorithm; this section concentrates on that type of algorithm.

To make the notation less cluttered and the fundamentals of interior point methods easier to understand, consider without loss of generality the following simpler NLP problem:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \geq 0, i \in \mathcal{I} = \{1, 2, \ldots, q\} \end{array}$$

Note that the equality and bound constraints have been omitted from the preceding problem. Initially, slack variables are added to the inequality constraints, giving rise to the problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) - s_i = 0, i \in \mathcal{I} \\ & s \geq 0 \end{array}$$

where $s = (s_1, \ldots, s_q)^{\mathrm{T}}$ is the vector of slack variables, which are required to be nonnegative. Next, all the nonnegativity constraints on the slack variables are eliminated by being incorporated into the objective function, by means of a logarithmic function. This gives rise to the equality-constrained NLP problem

$$\begin{array}{ll} \text{minimize} & B(x, s) = f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) \\ \text{subject to} & g_i(x) - s_i = 0, i \in \mathcal{I} \end{array}$$

where μ is a positive parameter. The nonnegativity constraints on the slack variables are implicitly enforced by the logarithmic functions, since the logarithmic function prohibits s from taking zero or negative values.

Next, the equality constraints can be absorbed by using a quadratic penalty function to obtain the following:

$$\text{minimize} \quad \mathcal{M}(x,s) = f(x) + \frac{1}{2\mu}\|g(x) - s\|_2^2 - \mu \textstyle\sum_{i \in \mathcal{I}} \ln(s_i)$$

The preceding unconstrained problem is often called the *penalty-barrier subproblem*. Depending on the size of the parameter μ, a local minimum of the barrier problem provides an approximation to the local minimum of the original NLP problem. The smaller the size of μ, the better the approximation becomes. Infeasible primal-dual interior point algorithms repeatedly solve the penalty-barrier problem for different values of μ that progressively go to zero, in order to get as close as possible to a local minimum of the original NLP problem.

An unconstrained minimizer of the penalty-barrier problem must satisfy the equations

$$\begin{array}{rcl} \nabla f(x) - J(x)^{\mathrm{T}} z & = & 0 \\ z - \mu S^{-1} e & = & 0 \end{array}$$

where $z = -(g(x) - s)/\mu$, $J(x)$ is the Jacobian matrix of the vector function $g(x)$, S is the diagonal matrix whose elements are the elements of the vector s (that is, $S = \operatorname{diag}\{s_1, \ldots, s_q\}$), and e is a vector of all ones. Multiplying the second equation by S and adding the definition of z as a third equation produces the following equivalent nonlinear system:

$$F^{\mu}(x,s,z) = \begin{pmatrix} \nabla f(x) - J(x)^{\mathrm{T}} z \\ Sz - e \\ g(x) - s + \mu z \end{pmatrix} = 0$$

Note that if $\mu = 0$, the preceding conditions represent the optimality conditions of the original optimization problem, after adding slack variables. One of the main aims of the algorithm is to gradually reduce the value of μ to zero, so that it converges to a local optimum of the original NLP problem. The rate by which μ approaches zero affects the overall efficiency of the algorithm. Algorithms that treat z as an additional variable are considered primal-dual, while those that enforce the definition of $z = -(g(x) - s)/\mu$ at each iteration are consider purely primal approaches.

At iteration k, the infeasible primal-dual interior point algorithm approximately solves the preceding system by using Newton's method. The Newton system is

$$\begin{bmatrix} H_{\mathcal{L}}(x^k, z^k) & 0 & -J(x^k)^{\mathrm{T}} \\ 0 & Z_k & S_k \\ J(x^k) & -I & \mu I \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta s^k \\ \Delta z^k \end{bmatrix} = - \begin{bmatrix} \nabla_x f(x^k) - J(x^k)^{\mathrm{T}} z \\ -\mu e + S_k z^k \\ g(x^k) - s^k + \mu z^k \end{bmatrix}$$

where $H_{\mathcal{L}}$ is the Hessian matrix of the Lagrangian function $\mathcal{L}(x,z) = f(x) - z^{\mathrm{T}} g(x)$ of the original NLP problem; that is,

$$H_{\mathcal{L}}(x,z) = \nabla^2 f(x) - \sum_{i \in \mathcal{I}} z_i \nabla^2 g_i(x)$$

The solution $(\Delta x^k, \Delta s^k, \Delta z^k)$ of the Newton system provides a direction to move from the current iteration (x^k, s^k, z^k) to the next,

$$(x^{k+1}, s^{k+1}, z^{k+1}) = (x^k, s^k, z^k) + \alpha(\Delta x^k, \Delta s^k, \Delta z^k)$$

where α is the step length along the Newton direction. The step length is determined through a line-search procedure that ensures sufficient decrease of a merit function based on the augmented Lagrangian function of the barrier problem. The role of the merit function and the line-search procedure is to ensure that the objective and the infeasibility reduce sufficiently at every iteration and that the iterations approach a local minimum of the original NLP problem.

Interior Point Direct Algorithm

This primal-dual algorithm combines interior point techniques for handling the inequality constraints and an augmented Lagrangian method for the equalities. It uses tools that have been developed in recent research papers (Armand and Omheni 2017a, b). For the sake of simplicity, this section uses the slack formulation that is introduced in the previous section,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) - s_i = 0, i \in \mathcal{I} \\ & s \geq 0 \end{array}$$

where $s = (s_1, \ldots, s_q)^{\mathrm{T}}$ is the vector of slack variables, which are required to be nonnegative. To deal with the nonnegative lower bound on s, the logarithmic term $-\mu \sum_{i \in \mathcal{I}} \ln(s_i)$ is added to the objective function, giving the barrier problem,

$$\begin{array}{ll} \text{minimize} & B(x,s) = f(x) - \mu \sum_{i \in \mathcal{I}} \ln(s_i) \\ \text{subject to} & g_i(x) - s_i = 0, i \in \mathcal{I} \end{array}$$

where $\mu > 0$ is the barrier parameter. The nonlinear equality constraints are handled using an augmented Lagrangian term. The resulting unconstrained formulation is

$$\text{minimize} \quad \mathcal{M}(x,s) = f(x) + \lambda^{\top}(g(x) - s) + \frac{1}{2\sigma}\|g(x) - s\|^2 - \mu \sum_{i \in \mathcal{I}} \ln(s_i)$$

where $\lambda \in \mathbb{R}^q$ is an estimate of the Lagrange multiplier that is associated with equality constraints and where $\sigma > 0$ is the penalty parameter. The first-order optimality conditions of this unconstrained problem are given by

$$\begin{array}{rcl} \nabla f(x) + J(x)^{\mathrm{T}}(\lambda + \frac{1}{\sigma}(g(x) - s)) & = & 0 \\ -\lambda - \frac{1}{\sigma}(g(x) - s) - \mu S^{-1}e & = & 0 \end{array}$$

By introducing the variables $y = \lambda + \frac{1}{\sigma}(g(x) - s)$ and $z = \mu S^{-1}e$, this system of equations can be rewritten in the following form:

$$\begin{array}{rcl} \nabla f(x) + J(x)^{\mathrm{T}} y & = & 0 \\ -y - z & = & 0 \\ g(x) - s & = & \sigma(y - \lambda) \\ SZe & = & \mu e \end{array}$$

These optimality conditions can be interpreted as perturbed optimality conditions for the original slack formulation problem. That is, if $\sigma = 0$ or if $\lambda = y$ and $\mu = 0$, then the preceding conditions reduce to the standard first-order optimality conditions. The algorithm is designed to encourage λ to naturally converge to y; thus, the penalty term σ might then remain bounded, resulting in better conditioning and fewer iterations overall.

The basic idea of the algorithm is to apply Newton's method to the perturbed optimality conditions system while preserving strict feasibility with respect to s and z, driving the barrier term, μ, to 0, and driving the penalty term, σ, to 0 while periodically updating the Lagrange multiplier estimate, λ. These update rules ensure convergence to an optimal solution of the original problem. The algorithm has two types of iterations: *outer* and *inner*. At an outer iteration k, the algorithm computes a search direction by using the Newton equations

$$\begin{bmatrix} \nabla_x^2 \mathcal{L}(x^k, y^k, z^k) & 0 & J(x^k)^\top & 0 \\ 0 & 0 & -I & -I \\ J(x^k) & -I & -\sigma I & 0 \\ 0 & Z_k & 0 & S_k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta s^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = - \begin{bmatrix} \nabla f(x^k) + J(x^k)^\top y^k \\ -y^k - z^k \\ g(x^k) - s^k + \sigma(\lambda - y^k) \\ S_k Z_k e - \mu e \end{bmatrix}$$

where $\mathcal{L}$ denotes the Lagrangian function,

$$\mathcal{L}(x, y, z) = f(x) + y^\top (g(x) - s) - z^\top s$$

A solution to the Newton equations is obtained by solving the equivalent reduced symmetric linear system:

$$\begin{bmatrix} \nabla_x^2 \mathcal{L}(x^k, y^k, z^k) & 0 & J(x^k)^\top \\ 0 & S_k{}^{-1} Z_k & -I \\ J(x^k) & -I & -\sigma I \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta s^k \\ \Delta y^k \end{bmatrix} = - \begin{bmatrix} \nabla f(x^k) + J(x^k)^\top y^k \\ -y^k - \mu S_k{}^{-1} e \\ g(x^k) - s^k + \sigma(\lambda - y^k) \end{bmatrix}$$

The search direction Δz^k is then recovered by the following equation:

$$\Delta z^k = \mu S_k{}^{-1} e - z^k - S_k{}^{-1} Z_k \Delta s^k$$

For nonconvex problems, an inertia-correcting diagonal modification can be used to ensure downhill search directions, resulting in the following modified KKT matrix:

$$\begin{bmatrix} \nabla_x^2 \mathcal{L}(x^k, y^k, z^k) + \theta I & 0 & J(x^k)^\top \\ 0 & S_k{}^{-1} Z_k + \theta I & -I \\ J(x^k) & -I & -\sigma I \end{bmatrix}$$

The regularization parameter, $\theta \geq 0$, is updated to make sure that the KKT matrix has exactly $n + q$ positive, q negative, and no zero eigenvalues. During inner iterations, the inertia control mechanism guarantees that the solution of the linear system is a descent direction for some primal-dual merit function. This mechanism is also applied along outer iterations to avoid convergence to a stationary point that would not be a minimum. The natural regularization that is introduced by σ in the lower right block of the KKT matrix plays an important role whenever degenerate problems are solved for which the constraint Jacobian, $J(x^k)$, is rank deficient.

The resolution of the linear system is followed by the computation of the step sizes α^P and $\alpha^D \in (0, 1]$ by applying the fraction to the boundary rule in order to maintain strict feasibility of the iterates with respect to the bound constraints. The trial new iterate is then defined as

$$(x_+^k, s_+^k, y_+^k) = (x^k, s^k, y^k) + \alpha^P (\Delta x^k, \Delta s^k, \Delta y^k) \quad \text{and} \quad z_+^k = z^k + \alpha^D \Delta z^k$$

Note that this algorithm allows the use of different step sizes for the dual variable z from the variables x, s, and y. If the norm of the perturbed KKT conditions is deemed sufficiently small, then the trial point is accepted as a new iterate; that is,

$$(x^{k+1}, s^{k+1}, y^{k+1}, z^{k+1}) = (x_+^k, s_+^k, y_+^k, z_+^k)$$

Otherwise, a sequence of inner iterations is applied to find a new iterate $(x^{k+1}, s^{k+1}, y^{k+1}, z^{k+1})$ that decreases the perturbed KKT conditions. The inner iterations correspond to a backtracking line-search algorithm that is applied to a primal-dual merit function. Depending on the problem type, the step length might be determined through a filter technique.

Before starting a new outer iteration, the penalty parameter and the estimate of the Lagrange multiplier are updated depending on the reduction of the constraint violation, as in classical augmented Lagrangian. The barrier parameter is decreased as soon as the KKT conditions of the barrier problem are satisfied to some tolerance.

The main differences between the two interior point methods are:

- The IPDIRECT solver uses direct linear algebra to compute the solution of the linear system, whereas the INTERIORPOINT solver uses an iterative approach. Unless your problem is very large and sparse, it is suggested that you try the direct approach first. The direct approach has been empirically demonstrated to improve robustness while reducing CPU time, function evaluations, and iterations.
- To handle challenges such as nonconvexity and singular systems, safeguards and globalization strategies have been added to both the INTERIORPOINT and IPDIRECT solvers to ensure robust convergence on a wide range of challenging optimization problems. However, the IPDIRECT solver has been aggressively streamlined to minimize the negative impact that these safeguard strategies might have on asymptotic convergence rates.

Active-Set Method

Active-set methods differ from interior point methods in that no barrier term is used to ensure that the algorithm remains interior with respect to the inequality constraints. Instead, attempts are made to learn the true active set. For simplicity, use the same initial slack formulation used by the interior point method description,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) - s_i = 0, i \in \mathcal{I} \\ & s \geq 0 \end{array}$$

where $s = (s_1, \ldots, s_q)^{\mathrm{T}}$ is the vector of slack variables, which are required to be nonnegative. Begin by absorbing the equality constraints as before into a penalty function, but keep the slack bound constraints explicitly:

$$\begin{array}{ll} \text{minimize} & \mathcal{M}(x, s) = f(x) + \dfrac{1}{2\mu}\|g(x) - s\|_2^2 \\ \text{subject to} & s \geq 0 \end{array}$$

where μ is a positive parameter. Given a solution pair $(x(\mu), s(\mu))$ for the preceding problem, you can define the active-set projection matrix P as follows:

$$P_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } s_i(\mu) = 0 \\ 0 & \text{otherwise.} \end{cases}$$

Then $(x(\mu), s(\mu))$ is also a solution of the equality constraint subproblem:

$$\begin{array}{ll} \text{minimize} & \mathcal{M}(x,s) = f(x) + \frac{1}{2\mu}\|g(x) - s\|_2^2 \\ \text{subject to} & Ps = 0. \end{array}$$

The minimizer of the preceding subproblem must be a stationary point of the Lagrangian function

$$\mathcal{L}^{\mu}(x,s,z) = f(x) + \frac{1}{2\mu}\|g(x) - s\|_2^2 - z^T Ps$$

which gives the optimality equations

$$\begin{array}{rclcl} \nabla_x \mathcal{L}^{\mu}(x,s,z) & = & \nabla f(x) - J(x)^T y & = & 0 \\ \nabla_s \mathcal{L}^{\mu}(x,s,z) & = & y - P^T z & = & 0 \\ & = & Ps & = & 0 \end{array}$$

where $y = -(g(x) - s)/\mu$. Using the second equation, you can simplify the preceding equations to get the following optimality conditions for the bound-constrained penalty subproblem:

$$\begin{array}{lcl} \nabla f(x) - J(x)^T P^T z & = & 0 \\ P(g(x) - s) + \mu z & = & 0 \\ Ps & = & 0 \end{array}$$

Using the third equation directly, you can reduce the system further to

$$\begin{array}{lcl} \nabla f(x) - J(x)^T P^T z & = & 0 \\ Pg(x) + \mu z & = & 0 \end{array}$$

At iteration k, the primal-dual active-set algorithm approximately solves the preceding system by using Newton's method. The Newton system is

$$\begin{bmatrix} H_{\mathcal{L}}(x^k, z^k) & -J_{\mathcal{A}}^{\mathrm{T}} \\ J_{\mathcal{A}} & \mu I \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta z^k \end{bmatrix} = - \begin{bmatrix} \nabla_x f(x^k) - J_{\mathcal{A}}^{\mathrm{T}} z \\ Pg(x^k) + \mu z^k \end{bmatrix}$$

where $J_{\mathcal{A}} = PJ(x^k)$ and $H_{\mathcal{L}}$ denotes the Hessian of the Lagrangian function $f(x) - z^{\mathrm{T}} Pg(x)$. The solution $(\Delta x^k, \Delta z^k)$ of the Newton system provides a direction to move from the current iteration (x^k, s^k, z^k) to the next,

$$(x^{k+1}, z^{k+1}) = (x^k, z^k) + \alpha(\Delta x^k, \Delta z^k)$$

where α is the step length along the Newton direction. The corresponding slack variable update s^{k+1} is defined as the solution to the following subproblem whose solution can be computed analytically:

$$\begin{array}{ll} \text{minimize} & \mathcal{M}(x^{k+1}, s) = f(x) + \frac{1}{2\mu}\|g(x^{k+1}) - s\|_2^2 \\ \text{subject to} & s \geq 0 \end{array}$$

The step length α is then determined in a similar manner to the preceding interior point approach. At each iteration, the definition of the active-set projection matrix P is updated with respect to the new value of the constraint function $g(x^{k+1})$. For large-scale NLP, the computational bottleneck typically arises in seeking to solve the Newton system. Thus active-set methods can achieve substantial computational savings when the size of $J_{\mathcal{A}}$ is much smaller than $J(x)$; however, convergence can be slow if the active-set estimate changes combinatorially. Further, the active-set algorithm is often the superior algorithm when only bound constraints are present. In practice, both the interior point and active-set approach incorporate more sophisticated merit functions than those described in the preceding sections; however, their description is beyond the scope of this document. See Gill and Robinson (2010) for further reading.

Multistart

Frequently, nonlinear optimization problems contain many local minima because the objective or the constraints are nonconvex functions. The quality of different local minima is measured by the objective value achieved at those points. For example, if x_1^* and x_2^* are two distinct local minima and $f(x_1^*) \leq f(x_2^*)$, then x_1^* is said to be of better quality than x_2^*. The NLP solver provides a mechanism that can locate local minima of better quality by starting the local solver multiple times from different initial points. By doing so, the local solver can converge to different local minima. The local minimum with the lowest objective value is then reported back to the user.

The multistart feature consists of two phases. In the first phase, the entire feasible region is explored by generating sample points from a uniform distribution. The aim of this phase is to place at least one sample point in the region of attraction of every local minimum. Here the region of attraction of a local minimum is defined as the set of feasible points that, when used as starting points, enable a local solver to converge to that local minimum.

During the second phase, a subset of the sample points generated in the first phase is chosen by applying a clustering technique. The goal of the clustering technique is to group the initial sample points around the local minima and allow only a single local optimization to start from each cluster or group. The clustering technique aims to reduce computation time by sparing the work of unnecessarily starting multiple local optimizations within the region of attraction of the same local minimum.

The number of starting points is critical to the time spent by the solver to find a good local minimum. You can specify the maximum number of starting points by using the MAXSTARTS= suboption. If this option is not specified, the solver determines the minimum number of starting points that can provide reasonable evidence that a good local minimum will be found.

Many optimization problems contain variables with infinite upper or lower bounds. These variables can cause the sampling procedure to generate points that are not useful for locating different local minima. The efficiency of the sampling procedure can be increased by reducing the range of these variables by using the BNDRANGE= suboption. This option forces the sampling procedure to generate points that are in a smaller interval, thereby increasing the efficiency of the solver to converge to a local optimum.

The multistart feature is compatible with the PERFORMANCE statement in the OPTMODEL procedure. See Chapter 4, “Shared Concepts and Topics,” for more information about the PERFORMANCE statement. The multistart feature currently supports only the DETERMINISTIC value for the PARALLELMODE= option in the PERFORMANCE statement. To ensure reproducible results, specify a nonzero value for the SEED= option.

Accessing the Starting Point That Leads to the Best Local Optimum

The starting point that leads to the best local optimum can be accessed by using the .msinit suffix in PROC OPTMODEL. In some cases, the knowledge of that starting point might be useful. For example, you can run the local solver again but this time providing as initial point the one that is stored in .msinit. This way the multistart explores a different part of the feasible region and might discover a local optimum of better quality than those found in previous runs. The use of the suffix .msinit is demonstrated in Example 11.6. For more information about suffixes in PROC OPTMODEL, see “Suffixes” on page 135 in Chapter 5, “The OPTMODEL Procedure.”

Covariance Matrix

You must specify the COVEST=() option to compute an approximate covariance matrix for the parameter estimates under asymptotic theory for least squares, maximum likelihood, or Bayesian estimation, with or without corrections for degrees of freedom as specified in the VARDEF= option.

The standard form of this class of the problems is one of following:

- least squares (LSQ): $\min f(x) = s \sum_{i=1}^{m} f_i^2(x)$
- minimum or maximum (MIN/MAX): $\text{opt } f(x) = s \sum_{i=1}^{m} f_i(x)$

For example, two groups of six different forms of covariance matrices (and therefore approximate standard errors) can be computed corresponding to the following two situations, where TERMS is an index set.

- LSQ: The objective function consists solely of a positively scaled sum of squared terms, which means that least squares estimates are being computed:

  $$\min f(x) = s \sum_{(i,j)\in \text{TERMS}} f_{ij}^2(x)$$

  where $s > 0$.

- MIN or MAX: The MIN or MAX declaration is specified, and the objective is not in least squares form. Together, these characteristics mean that maximum likelihood or Bayesian estimates are being computed:

  $$\text{opt } f(x) = s \sum_{(i,j)\in \text{TERMS}} f_{ij}(x)$$

  where opt is either $\min$ or $\max$ and s is arbitrary.

In the preceding section, TERMS is used to denote an arbitrary index set. For example, if your problem is

$$\min z = 0.5 \sum_{i\in\mathcal{I}} (g_1^2[i] + g_2^2[i])$$

then TERMS = $\{(i,j) : i \in \mathcal{I} \text{ and } j \in \{1,2\}\}$, where $\mathcal{I}$ is the index set of input data. The following rules apply when you specify your objective function:

- The terms $f_{ij}(x)$ can be either IMPVAR expressions or constant expressions (expressions that do not depend on variables). The i and j values can be partitioned among observation and function indices as needed. Any number of indices can be used, including non-array indices to implicit variables.
- The nonconstant terms are defined by using the IMPVAR declaration. Each nonconstant IMPVAR element can be referenced at most once in the objective.
- The objective consists of a scaled sum of terms (or squared terms for least squares). The scaling, shown as s in the preceding equations, consists of outer multiplication or division by constants of the unscaled sum of terms (or squared terms for least squares). The unary $+$ or $-$ operators can also be used for scaling.

- Least squares objectives require the scaling to be positive ($s > 0$). The individual f_{ij} values are scaled by $\sqrt{2s}$ by PROC OPTMODEL.
- Objectives that are not least squares allow arbitrary scaling. The scale value is distributed to the f_{ij} values.
- The summation of terms (or squared terms for least squares) is constructed with the binary $+$, SUM, and IF-THEN-ELSE operators (where IF-THEN-ELSE must have a first operand that does not depend on variables). The operands can be terms or a summation of terms (or squared terms for least squares).
- A squared term is specified as **`term^2`** or **`term**2`**.
- The default value of the NTERMS= option is determined by counting the nonconstant terms. The constant terms do not contribute to the covariance matrix.

The following PROC OPTMODEL statements demonstrate these rules:

```
var x{VARS};
impvar g{OBS} = ...;  /* expression involves x */
impvar h{OBS} = ...;  /* expression involves x */

/* This objective is okay. */
min z1 = sum{i in OBS} (g[i] + h[i]);

/* This objective is okay. */
min z2 = 0.5*sum{i in OBS} (g[i]^2 + h[i]^2);

/* This objective is okay. It demonstrates multiple levels of scaling. */
min z3 = 3*(sum{i in OBS} (g[i]^2 + h[i]^2))/2;

/* This objective is okay. */
min z4 = (sum{i in OBS} (g[i]^2 + h[i]^2))/2;
```

Note that the following statements are not accepted:

```
/* This objective causes an error because individual scaling is not allowed. */
/* (division applies to inner term)                                          */
min z5 = sum{i in OBS} (g[i]^2 + h[i]^2)/2;

/* This objective causes an error because individual scaling is not allowed. */
min z6 = sum{i in OBS} 0.5*g[i]^2;

/* This objective causes an error because the element g[1] is repeated. */
min z7 = g[1] + sum{i in OBS} g[i];
```

The covariance matrix is always positive semidefinite. For MAX type problems, the covariance matrix is converted to MIN type by using negative Hessian, Jacobian, and function values in the computation. You can use the following options to check for a rank deficiency of the covariance matrix:

- The ASINGULAR= and MSINGULAR= options enable you to set two singularity criteria for the inversion of the matrix A that is needed to compute the covariance matrix, when A is either the Hessian or one of the crossproduct Jacobian matrices. The singularity criterion that is used for the inversion is

$$|d_{j,j}| \leq \max(\textit{asing}, \textit{msing} \times \max(|A_{1,1}|, \ldots, |A_{n,n}|))$$

where $d_{j,j}$ is the diagonal pivot of the matrix A, and *asing* and *msing* are the specified values of the ASINGULAR= and MSINGULAR= options, respectively.

- If the matrix A is found to be singular, the NLP solver computes a generalized inverse that satisfies Moore-Penrose conditions. The generalized inverse is computed using the computationally expensive but numerically stable eigenvalue decomposition, $A = Z\Lambda Z^T$, where Z is the orthogonal matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$. The generalized inverse of A is set to

  $$A^- = Z\Lambda^- Z^T$$

  where the diagonal matrix $\Lambda^- = \mathrm{diag}(\lambda_1^-, \ldots, \lambda_n^-)$ is defined as follows, where *covsing* is the specified value of the COVSING= option:

  $$\lambda_i^- = \begin{cases} 1/\lambda_i & \text{if } |\lambda_i| > \textit{covsing} \\ 0 & \text{if } |\lambda_i| \le \textit{covsing} \end{cases}$$

  If the COVSING= option is not specified, then the default is $\max(\textit{asing}, \textit{msing} \times \max(|A_{1,1}|, \ldots, |A_{n,n}|))$, where *asing* and *msing* are the specified values of the ASINGULAR= and MSINGULAR= options, respectively.

For problems of the MIN or LSQ type, the matrices that are used to compute the covariance matrix are

$$G = \nabla^2 f(x)$$

$$J(f) = (\nabla f_1, \ldots, \nabla f_m) = \left(\frac{\partial f_i}{\partial x_j}\right)$$

$$JJ(f) = J(f)^T J(f)$$

$$V = J(f)^T \mathrm{diag}(f_i^2) J(f)$$

$$W = J(f)^T (\mathrm{diag}(f_i))^{-1} J(f)$$

where f_i is defined in the standard form of the covariance matrix problem. Note that when some f_i are 0, $(\mathrm{diag}(f_i))^{-1}$ is computed as a generalized inverse.

For unconstrained minimization, the formulas of the six types of covariance matrices are given in Table 11.2. The value of d in the table depends on the VARDEF= option and the values of the NDF= and NTERMS= options, *ndf* and *nterms*, respectively, as follows:

$$d = \begin{cases} \max(1, \textit{nterms} - \textit{ndf}) & \text{for VARDEF=DF} \\ \textit{nterms} & \text{for VARDEF=N} \end{cases}$$

The value of σ^2 depends on the specification of the SIGSQ= option and on the value of d,

$$\sigma^2 = \begin{cases} \textit{sq} \times \textit{nterms}/d & \text{if SIGSQ=}\textit{sq}\text{ is specified} \\ 2f(x^*)/d & \text{if SIGSQ= is not specified} \end{cases}$$

where $f(x^*)$ is the value of the objective function at the optimal solution x^*.

Because of the analytic definition, in exact arithmetic the covariance matrix is positive semidefinite at the solution. A warning message is issued if numerical computation does not result in a positive semidefinite matrix. This can happen because round-off error is introduced or the incorrect type of covariance matrix for a specified problem is selected.

Iteration Log for the Local Solver

The iteration log for the local solver provides detailed information about progress towards a locally optimal solution. This log appears when multistart mode is disabled.

The following information is displayed in the log:

Iter	indicates the iteration number.
Objective Value	indicates the objective function value.
Infeasibility	indicates the maximum value out of all constraint violations.
Optimality Error	indicates the relative optimality error (see the section "Solver Termination Criterion" on page 566).

Iteration Log for Multistart

When the MULTISTART=() option is specified, the iteration log differs from that of the local solver. More specifically, when a value of 2 is specified for the LOGLEVEL= suboption, the following information is displayed in the log:

Start — indicates the index number of each local optimization run. The following indicator can appear beside this number to provide additional information about the run:

*	indicates the local optimization started from a user-supplied point.

Best Objective	indicates the value of the objective function at the best local solution found so far.
Local Objective	indicates the value of the objective function obtained at the solution returned by the local solver.
Infeasibility	indicates the infeasibility error at the solution returned by the local solver.
Optimality Error	indicates the optimality error at the solution returned by the local solver.
Local Iters	indicates the number of iterations taken by the local solver.
Local Status	indicates the solution status of the local solver. Several different values can appear in this column:

OPTIMAL	indicates that the local solver found a locally optimal solution.
BESTFEASIBLE	indicates that the local solver returned the best feasible point found. See the SOLTYPE= option for more information.
INFEASIBLE	indicates that the local solver converged to a point that might be infeasible.
LOCALINFEAS	indicates that the local solver converged to a point of minimal local infeasibility.

UNBOUNDED	indicates that the local solver determined that the problem is unbounded.
ITERLIMIT	indicates that the local solver reached the maximum number of iterations and could not find a locally optimal solution.
TIMELIMIT	indicates that the local solver reached the maximum allowable time and could not find a locally optimal solution.
ABORTED	indicates that the local solver terminated due to a user interrupt.
FUNEVALERR	indicates that the local solver encountered a function evaluation error.
NUMERICERR	indicates that the local solver encountered a numerical error other than a function evaluation error.
INTERNALERR	indicates that the local solver encountered a solver system error.
OUTMEMORY	indicates that the local solver ran out of memory.
FAILED	indicates a general failure of the local solver in the absence of any other error.

Solver Termination Criterion

Because badly scaled problems can lead to slow convergence, the NLP solver dynamically rescales both the objective and constraint functions adaptively as needed. The optimality conditions are always stated with respect to the rescaled NLP. However, because typically you are most interested in the constraint violation of the original NLP, and not the internal scaled variant, you always work with respect to the true constraint violation. Thus, the solver terminates when both of the following conditions are true:

- The norm of the optimality conditions of the scaled NLP is less than the user-defined or default tolerance (OPTTOL= option).
- The norm of the constraint violation of the original NLP is less than the user-defined feasibility tolerance (FEASTOL= option).

More specifically, if

$$F(x, s, z) = (\nabla_x f(x) - J(x)^{\mathrm{T}} z, \quad Sz, \quad g(x) - s)^{\mathrm{T}}$$

is the vector of the optimality conditions of the rescaled NLP problem, then the solver terminates when

$$\| \ F(x, s, z) \ \| \leq \mathrm{OPTTOL}(1 + \|(x, s)\|)$$

and the maximum constraint violation is less than FEASTOL.

Solver Termination Messages

Upon termination, the solver produces the following messages in the log:

Optimal
The solver has successfully found a local solution to the optimization problem.

Conditionally optimal solution found
The solver is sufficiently close to a local solution, but it has difficulty in completely satisfying the user-defined optimality tolerance. This can happen when the line search finds very small steps that result in very slight progress of the algorithm. It can also happen when the prespecified tolerance is too strict for the optimization problem at hand.

Maximum number of iterations reached
The solver could not find a local optimum in the prespecified number of iterations.

Maximum specified time reached
The solver could not find a local optimum in the prespecified maximum real time for the optimization process.

Did not converge
The solver could not satisfy the optimality conditions and failed to converge.

Problem might be unbounded
The objective function takes arbitrarily large values, and the optimality conditions fail to be satisfied. This can happen when the problem is unconstrained or when the problem is constrained and the feasible region is not bounded.

Problem might be infeasible
The solver cannot identify a point in the feasible region.

Problem is infeasible
The solver detects that the problem is infeasible.

Out of memory
The problem is so large that the solver requires more memory to solve the problem.

Problem solved by the OPTMODEL presolver
The problem was solved by the OPTMODEL presolver.

Macro Variable _OROPTMODEL_

The OPTMODEL procedure always creates and initializes a SAS macro variable called _OROPTMODEL_, which contains a character string. After each PROC OPTMODEL run, you can examine this macro variable by specifying `%put &_OROPTMODEL_;` and check the execution of the most recently invoked solver from the value of the macro variable. After the NLP solver is called, the various terms of the variable are interpreted as follows:

STATUS

indicates the solver status at termination. It can take one of the following values:

OK	The solver terminated normally.
SYNTAX_ERROR	The use of syntax is incorrect.
DATA_ERROR	The input data are inconsistent.
OUT_OF_MEMORY	Insufficient memory was allocated to the procedure.
IO_ERROR	A problem in reading or writing of data has occurred.
SEMANTIC_ERROR	An evaluation error, such as an invalid operand type, has occurred.
ERROR	The status cannot be classified into any of the preceding categories.

ALGORITHM

indicates the algorithm that produced the solution data in the macro variable. This term only appears when STATUS=OK. It can take one of the following values:

IP	The interior point algorithm produced the solution data.
IPDIR	The experimental interior point direct algorithm produced the solution data.
AS	The active-set algorithm produced the solution data.

When running algorithms concurrently (ALGORITHM=CONCURRENT), this term indicates which algorithm was the first to terminate.

SOLUTION_STATUS

indicates the solution status at termination. It can take one of the following values:

OPTIMAL	The solution is optimal.
CONDITIONAL_OPTIMAL	The optimality of the solution cannot be proven.
BEST_FEASIBLE	The solution returned is the best feasible solution. This solution status indicates that the algorithm has converged to a local optimum but a feasible (not locally optimal) solution with a better objective value has been found and hence is returned.
INFEASIBLE	The problem is infeasible.
UNBOUNDED	The problem might be unbounded.
INFEASIBLE_OR_UNBOUNDED	The problem is infeasible or unbounded.
BAD_PROBLEM_TYPE	The problem type is not supported by the solver.
ITERATION_LIMIT_REACHED	The maximum allowable number of iterations has been reached.
TIME_LIMIT_REACHED	The solver reached its execution time limit.
FAILED	The solver failed to converge, possibly due to numerical issues.

OBJECTIVE
indicates the objective value that is obtained by the solver at termination.

NUMSTARTS
indicates the number of starting points. This term appears only in multistart mode.

SAMPLE_POINTS
indicates the number of points that are evaluated in the sampling phase. This term appears only in multistart mode.

DISTINCT_OPTIMA
indicates the number of distinct local optima that the solver finds. This term appears only in multistart mode.

SEED
indicates the seed value that is used to initialize the solver. This term appears only in multistart mode.

INFEASIBILITY
indicates the level of infeasibility of the constraints at the solution.

OPTIMALITY_ERROR
indicates the norm of the optimality conditions at the solution. See the section "Solver Termination Criterion" on page 566 for details.

ITERATIONS
indicates the number of iterations required to solve the problem.

PRESOLVE_TIME
indicates the real time taken for preprocessing (seconds).

SOLUTION_TIME
indicates the real time taken by the solver to perform iterations for solving the problem (seconds).

NOTE: The time that is reported in PRESOLVE_TIME and SOLUTION_TIME is either CPU time or real time. The type is determined by the TIMETYPE= option.

Examples: NLP Solver

Example 11.1: Solving Highly Nonlinear Optimization Problems

This example demonstrates the use of the NLP solver to solve the following highly nonlinear optimization problem, which appears in Hock and Schittkowski (1981):

$$\begin{array}{ll}
\text{minimize} & f(x) = 0.4(x_1/x_7)^{0.67} + 0.4(x_2/x_8)^{0.67} + 10 - x_1 - x_2 \\
\text{subject to} & 1 - 0.0588x_5x_7 - 0.1x_1 \geq 0 \\
& 1 - 0.0588x_6x_8 - 0.1x_1 - 0.1x_2 \geq 0 \\
& 1 - 4x_3/x_5 - 2/(x_3^{0.71}x_5) - 0.0588x_7/x_3^{1.3} \geq 0 \\
& 1 - 4x_4/x_6 - 2/(x_4^{0.71}x_6) - 0.0588x_8/x_4^{1.3} \geq 0 \\
& 0.1 \leq f(x) \leq 4.2 \\
& 0.1 \leq x_i \leq 10, i = 1, 2, \ldots, 8
\end{array}$$

The initial point used is $x^0 = (6, 3, 0.4, 0.2, 6, 6, 1, 0.5)$. You can call the NLP solver within PROC OPTMODEL to solve the problem by writing the following SAS statements:

```
proc optmodel;
   var x{1..8} >= 0.1 <= 10;

   min f = 0.4*(x[1]/x[7])^0.67 + 0.4*(x[2]/x[8])^0.67 + 10 - x[1] - x[2];

   con c1: 1 - 0.0588*x[5]*x[7] - 0.1*x[1] >= 0;
   con c2: 1 - 0.0588*x[6]*x[8] - 0.1*x[1] - 0.1*x[2] >= 0;
   con c3: 1 - 4*x[3]/x[5] - 2/(x[3]^0.71*x[5]) - 0.0588*x[7]/x[3]^1.3 >= 0;
   con c4: 1 - 4*x[4]/x[6] - 2/(x[4]^0.71*x[6]) - 0.0588*x[8]/x[4]^1.3 >= 0;
   con c5: 0.1 <= f <= 4.2;

   /* starting point */
   x[1] = 6;
   x[2] = 3;
   x[3] = 0.4;
   x[4] = 0.2;
   x[5] = 6;
   x[6] = 6;
   x[7] = 1;
   x[8] = 0.5;

   solve with nlp / algorithm=activeset;
   print x;
 quit;
```

The summaries and the solution are shown in Output 11.1.1.

Output 11.1.1 Summaries and the Returned Solution

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Nonlinear
Number of Variables	8
Bounded Above	0
Bounded Below	0
Bounded Below and Above	8
Free	0
Fixed	0
Number of Constraints	5
Linear LE (<=)	0
Linear EQ (=)	0
Linear GE (>=)	0
Linear Range	0
Nonlinear LE (<=)	0
Nonlinear EQ (=)	0
Nonlinear GE (>=)	4
Nonlinear Range	1

Solution Summary	
Solver	NLP
Algorithm	Active Set
Objective Function	f
Solution Status	Best Feasible
Objective Value	3.9511579677
Optimality Error	0.0001050714
Infeasibility	7.7302351E-7
Iterations	26
Presolve Time	0.00
Solution Time	0.01

[1]	x
1	6.46332
2	2.23453
3	0.66746
4	0.59582
5	5.93298
6	5.52723
7	1.01379
8	0.40066

Example 11.2: Solving Unconstrained and Bound-Constrained Optimization Problems

Although the NLP techniques are suited for solving generally constrained nonlinear optimization problems, these techniques can also be used to solve unconstrained and bound-constrained problems efficiently. This example considers the relatively large nonlinear optimization problems

$$\text{minimize } f(x) = \sum_{i=1}^{n-1}(-4x_i + 3.0) + \sum_{i=1}^{n-1}(x_i^2 + x_n^2)^2$$

and

$$\begin{array}{ll} \text{minimize} & f(x) = \sum_{i=1}^{n-1} \cos(-.5x_{i+1} - x_i^2) \\ \text{subject to} & 1 \le x_i \le 2,\ i = 1, \ldots, n \end{array}$$

with $n = 100{,}000$. These problems are unconstrained and bound-constrained, respectively.

For large-scale problems, the default memory limit might be too small, which can lead to out-of-memory status. To prevent this occurrence, it is recommended that you set a larger memory size. See the section "Memory Limit" on page 21 for more information.

To solve the first problem, you can write the following statements:

```
proc optmodel;
   number N=100000;
   var x{1..N} init 1.0;

   minimize f = sum {i in 1..N - 1} (-4 * x[i] + 3.0)  +
                sum {i in 1..N - 1} (x[i]^2 + x[N]^2)^2;

   solve with nlp;
quit;
```

The problem and solution summaries are shown in Output 11.2.1.

Output 11.2.1 Problem Summary and Solution Summary

```
            The OPTMODEL Procedure

              Problem Summary
Objective Sense                  Minimization
Objective Function                          f
Objective Type                      Nonlinear

Number of Variables                    100000
Bounded Above                               0
Bounded Below                               0
Bounded Below and Above                     0
Free                                   100000
Fixed                                       0

Number of Constraints                       0

              Solution Summary
Solver                             NLP
Algorithm               Interior Point
Objective Function                   f
Solution Status                Optimal
Objective Value                      0

Optimality Error          1.001286E-14
Infeasibility                        0

Iterations                          16
Presolve Time                     0.01
Solution Time                     2.12
```

To solve the second problem, you can write the following statements (here the active-set method is specifically selected):

```
proc optmodel;
   number N=100000;
   var x{1..N} >= 1 <= 2;

   minimize f = sum {i in 1..N - 1} cos(-0.5*x[i+1] - x[i]^2);

   solve with nlp / algorithm=activeset;
quit;
```

The problem and solution summaries are shown in Output 11.2.2.

Output 11.2.2 Problem Summary and Solution Summary

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Nonlinear
Number of Variables	100000
Bounded Above	0
Bounded Below	0
Bounded Below and Above	100000
Free	0
Fixed	0
Number of Constraints	0

Solution Summary	
Solver	NLP
Algorithm	Active Set
Objective Function	f
Solution Status	Optimal
Objective Value	-99999
Optimality Error	4.3487696E-9
Infeasibility	0
Iterations	5
Presolve Time	0.01
Solution Time	2.32

Example 11.3: Solving Equality-Constrained Problems

In real applications, many problems are formulated as nonlinear optimization problems that have only equality constraints. The problems of this type are efficiently handled by the NLP solver. Consider the following equality-constrained problem, taken from Hock and Schittkowski (1981):

$$\begin{array}{ll} \text{minimize} & f(x) = \prod_{i=1}^{5} x_i \\ \text{subject to} & \sum_{i=1}^{5} x_i^2 = 10 \\ & x_2 x_3 - 5 x_4 x_5 = 0 \\ & x_1^3 + x_2^3 = -1 \end{array}$$

An initial point is given at $x^0 = (-2, 1.5, 2, -1, -1)$. You can use the following SAS code to formulate and solve this problem:

```
proc optmodel;
   var x{1..5};

   minimize obj = prod {i in 1..5} x[i];

   con constr1: sum {i in 1..5} x[i]^2 = 10;
   con constr2: x[2] * x[3] - 5 * x[4] * x[5] = 0;
   con constr3: x[1]^3 + x[2]^3 = -1;

   /* starting point */
   x[1] = -2;
   x[2] =  1.5;
   x[3] =  2;
   x[4] = -1;
   x[5] = -1;

   solve with nlp / algorithm=ipdirect;
   print x;
quit;
```

To solve this equality-constrained problem, the experimental interior point direct algorithm (IPDIRECT) is invoked. The problem summary, solution summary, and the optimal solution are shown in Output 11.3.1.

Output 11.3.1 Summaries and the Optimal Solution

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	obj
Objective Type	Nonlinear
Number of Variables	5
Bounded Above	0
Bounded Below	0
Bounded Below and Above	0
Free	5
Fixed	0
Number of Constraints	3
Linear LE (<=)	0
Linear EQ (=)	0
Linear GE (>=)	0
Linear Range	0
Nonlinear LE (<=)	0
Nonlinear EQ (=)	3
Nonlinear GE (>=)	0
Nonlinear Range	0

Output 11.3.1 *continued*

```
            Solution Summary
Solver                              NLP
Algorithm          Interior Point Direct
Objective Function                  obj
Solution Status                 Optimal
Objective Value            -2.919700415

Optimality Error           4.8986188E-9
Infeasibility              4.8986188E-9

Iterations                            3
Presolve Time                      0.00
Solution Time                      0.00
```

```
[1]         x
  1  -1.71714
  2   1.59571
  3   1.82725
  4  -0.76364
  5  -0.76364
```

Example 11.4: Solving NLP Problems with Range Constraints

Some constraints have both lower and upper bounds (that is, $a \le g(x) \le b$). These constraints are called *range constraints*. The NLP solver can handle range constraints in an efficient way. Consider the following NLP problem, taken from Hock and Schittkowski (1981),

$$\begin{array}{ll} \text{minimize} & f(x) = 5.35(x_3)^2 + 0.83x_1x_5 + 37.29x_1 - 40792.141 \\ \text{subject to} & 0 \le a_1 + a_2x_2x_5 + a_3x_1x_4 - a_4x_3x_5 \le 92 \\ & 0 \le a_5 + a_6x_2x_5 + a_7x_1x_2 + a_8x_3^2 - 90 \le 20 \\ & 0 \le a_9 + a_{10}x_3x_5 + a_{11}x_1x_3 + a_{12}x_3x_4 - 20 \le 5 \\ & 78 \le x_1 \le 102 \\ & 33 \le x_2 \le 45 \\ & 27 \le x_i \le 45, \quad i = 3,4,5 \end{array}$$

where the values of the parameters $a_i, i = 1, 2, \ldots, 12$, are shown in Table 11.8.

Table 11.8 Data for Example 4

i	a_i	**i**	a_i	**i**	a_i
1	85.334407	5	80.51249	9	9.300961
2	0.0056858	6	0.0071317	10	0.0047026
3	0.0006262	7	0.0029955	11	0.0012547
4	0.0022053	8	0.0021813	12	0.0019085

The initial point used is $x^0 = (78, 33, 27, 27, 27)$. You can call the NLP solver within PROC OPTMODEL to solve this problem by writing the following statements:

```
proc optmodel;
   number l {1..5} = [78 33 27 27 27];
   number u {1..5} = [102 45 45 45 45];

   number a {1..12} =
      [85.334407 0.0056858 0.0006262 0.0022053
      80.51249 0.0071317 0.0029955 0.0021813
      9.300961 0.0047026 0.0012547 0.0019085];

   var x {j in 1..5} >= l[j] <= u[j];

   minimize f = 5.35*x[3]^2 + 0.83*x[1]*x[5] + 37.29*x[1]
                  - 40792.141;

   con constr1:
      0 <= a[1] + a[2]*x[2]*x[5] + a[3]*x[1]*x[4] -
         a[4]*x[3]*x[5] <= 92;
   con constr2:
      0 <= a[5] + a[6]*x[2]*x[5] + a[7]*x[1]*x[2] +
         a[8]*x[3]^2 - 90 <= 20;
   con constr3:
      0 <= a[9] + a[10]*x[3]*x[5] + a[11]*x[1]*x[3] +
         a[12]*x[3]*x[4] - 20 <= 5;

   x[1] = 78;
   x[2] = 33;
   x[3] = 27;
   x[4] = 27;
   x[5] = 27;

   solve with nlp / algorithm=activeset;
   print x;
quit;
```

The summaries and solution are shown in Output 11.4.1.

Output 11.4.1 Summaries and the Optimal Solution

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Quadratic
Number of Variables	5
Bounded Above	0
Bounded Below	0
Bounded Below and Above	5
Free	0
Fixed	0
Number of Constraints	3
Linear LE (<=)	0
Linear EQ (=)	0
Linear GE (>=)	0
Linear Range	0
Nonlinear LE (<=)	0
Nonlinear EQ (=)	0
Nonlinear GE (>=)	0
Nonlinear Range	3

Solution Summary	
Solver	NLP
Algorithm	Active Set
Objective Function	f
Solution Status	Best Feasible
Objective Value	-30689.17757
Optimality Error	6.4824579E-6
Infeasibility	3.036293E-9
Iterations	45
Presolve Time	0.00
Solution Time	0.01

[1]	x
1	78.000
2	33.000
3	29.995
4	45.000
5	36.776

Example 11.5: Solving Large-Scale NLP Problems

The following example is a selected large-scale problem from the CUTEr test set (Gould, Orban, and Toint 2003) that has 20,400 variables, 20,400 lower bounds, and 9,996 linear equality constraints. This problem was selected to provide an idea of the size of problem that the NLP solver is capable of solving. In general, the maximum size of nonlinear optimization problems that can be solved with the NLP solver is controlled less by the number of variables and more by the density of the first and second derivatives of the nonlinear objective and constraint functions.

For large-scale problems, the default memory limit might be too small, which can lead to out-of-memory status. To prevent this occurrence, it is recommended that you set a larger memory size. See the section "Memory Limit" on page 21 for more information.

```
proc optmodel;
   num nx = 100;
   num ny = 100;

   var x {1..nx, 0..ny+1} >= 0;
   var y {0..nx+1, 1..ny} >= 0;

   min f = (
        sum {i in 1..nx-1, j in 1..ny-1} (x[i,j] - 1)^2
      + sum {i in 1..nx-1, j in 1..ny-1} (y[i,j] - 1)^2
      + sum {i in 1..nx-1} (x[i,ny] - 1)^2
      + sum {j in 1..ny-1} (y[nx,j] - 1)^2
      ) / 2;

   con con1 {i in 2..nx-1, j in 2..ny-1}:
      (x[i,j] - x[i-1,j]) + (y[i,j] - y[i,j-1]) = 1;
   con con2 {i in 2..nx-1}:
      x[i,0] + (x[i,1] - x[i-1,1]) + y[i,1] = 1;
   con con3 {i in 2..nx-1}:
      x[i,ny+1] + (x[i,ny] - x[i-1,ny]) - y[i,ny-1] = 1;
   con con4 {j in 2..ny-1}:
      y[0,j] + (y[1,j] - y[1,j-1]) + x[1,j] = 1;
   con con5 {j in 2..ny-1}:
      y[nx+1,j] + (y[nx,j] - y[nx,j-1]) - x[nx-1,j] = 1;

   for {i in 1..nx-1} x[i,ny].lb = 1;
   for {j in 1..ny-1} y[nx,j].lb = 1;

   solve with nlp;
quit;
```

The problem and solution summaries are shown in Output 11.5.1.

Output 11.5.1 Problem Summary and Solution Summary

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Quadratic
Number of Variables	20400
Bounded Above	0
Bounded Below	20400
Bounded Below and Above	0
Free	0
Fixed	0
Number of Constraints	9996
Linear LE (<=)	0
Linear EQ (=)	9996
Linear GE (>=)	0
Linear Range	0

Solution Summary	
Solver	NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Optimal
Objective Value	6237012.1174
Optimality Error	6.8107185E-7
Infeasibility	6.8107185E-7
Iterations	6
Presolve Time	0.01
Solution Time	6.23

Example 11.6: Solving NLP Problems That Have Several Local Minima

Some NLP problems contain many local minima. By default, the NLP solver converges to a single local minimum. However, the NLP solver can search the feasible region for other local minima. After it completes the search, it returns the point where the objective function achieves its minimum value. (This point might not be a local minimum; see the SOLTYPE= option for more details.) Consider the following example, taken from Hock and Schittkowski (1981):

$$
\begin{array}{ll}
\text{minimize} & f(x) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \\
\text{subject to} & x_1 + x_2^2 + x_3^3 = 2 + 3\sqrt{2} \\
 & x_2 + x_4 - x_3^2 = -2 + 2\sqrt{2} \\
 & x_1 x_5 = 2 \\
 & -5 \le x_i \le 5, i = 1, \ldots, 5
\end{array}
$$

The following statements call the NLP solver to search the feasible region for different local minima. The PERFORMANCE statement requests that the multistart algorithm use up to four threads. The SEED= option is specified for reproducibility, but it is not required in running the multistart algorithm.

```
proc optmodel;
   var x{i in 1..5} >= -5 <= 5 init -2;

   min f=(x[1] - 1)^2 + (x[1] - x[2])^2 + (x[2] - x[3])^3 +
         (x[3] - x[4])^4 + (x[4] - x[5])^4;

   con g1: x[1] + x[2]^2 + x[3]^3 =  2 + 3*sqrt(2);
   con g2: x[2] + x[4]   - x[3]^2 = -2 + 2*sqrt(2);
   con g3: x[1]*x[5] = 2;

   performance nthreads=4;
   solve with nlp/multistart=(maxstarts=10) seed=1234;
   print x.msinit x;
quit;
```

The PRINT statement prints the starting point (x.msinit) that led to the best local solution and the best local solution (x) that the NLP solver found in multistart mode. The SAS log is shown in Output 11.6.1.

Output 11.6.1 Progress of the Algorithm as Shown in the Log

```
NOTE: Problem generation will use 4 threads.
NOTE: The problem has 5 variables (0 free, 0 fixed).
NOTE: The problem has 0 linear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The problem has 3 nonlinear constraints (0 LE, 3 EQ, 0 GE, 0 range).
NOTE: The OPTMODEL presolver removed 0 variables, 0 linear constraints, and 0
      nonlinear constraints.
NOTE: Using analytic derivatives for objective.
NOTE: Using analytic derivatives for nonlinear constraints.
NOTE: The NLP solver is called.
NOTE: The Interior Point algorithm is used.
NOTE: The MULTISTART option is enabled.
NOTE: The deterministic parallel mode is enabled.
NOTE: The Multistart algorithm is executing in single-machine mode.
NOTE: The Multistart algorithm is using up to 4 threads.
NOTE: Random number seed 1234 is used.
                     Best       Local  Optimality    Infeasi-  Local  Local
      Start     Objective   Objective       Error      bility  Iters  Status
          1    607.035801  607.035801  8.62585E-7  8.62585E-7      7  Optimal
          2    52.9025715  52.9025715  1.19566E-7  9.15988E-8      7  Optimal
          3 *  52.9025715  607.034952  9.03873E-7  9.03873E-7      7  Optimal
          4    52.9025715  607.035512  2.81991E-7  5.88394E-9      7  Optimal
          5    52.9025715  52.9025732   2.3055E-7  6.49818E-8     12  Optimal
          6    27.8719049  27.8719049  7.45206E-7  5.07182E-7      5  Optimal
          7    27.8719049  64.8739987  3.12846E-7  3.12846E-7      7  Optimal
          8    0.02931084  0.02931084        5E-7  2.61833E-7      8  Optimal
          9    0.02931084  27.8719053  5.77935E-7  3.72218E-7      7  Optimal
         10    0.02931084  0.02931087  5.72626E-7  5.72626E-7      7  Optimal
NOTE: The Multistart algorithm generated 800 sample points.
NOTE: 5 distinct local optima were found.
NOTE: The best objective value found by local solver = 0.0293108361.
NOTE: The solution found by local solver with objective = 0.0293108361 was
      returned.
```

The first column in the log indicates the index of the current starting point. An additional indicator (*) can appear after the index to provide more information about the optimization run that started from the current point. For more information, see the section "Iteration Log for Multistart" on page 565. The second column records the best objective that has been found so far. Columns 3 to 6 report the objective value, optimality error, infeasibility, and number of iterations that the local solver returned when it was started from the current starting point. Finally, the last column records the status of the local solver—namely, whether it was able to converge to a local optimum from the current starting point.

The summaries and solution are shown in Output 11.6.2. Note that the best local solution was found by starting the local solver from a point at x.msinit.

Output 11.6.2 Summaries and the Optimal Solution

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Nonlinear
Number of Variables	5
Bounded Above	0
Bounded Below	0
Bounded Below and Above	5
Free	0
Fixed	0
Number of Constraints	3
Linear LE (<=)	0
Linear EQ (=)	0
Linear GE (>=)	0
Linear Range	0
Nonlinear LE (<=)	0
Nonlinear EQ (=)	3
Nonlinear GE (>=)	0
Nonlinear Range	0

Solution Summary	
Solver	Multistart NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Optimal
Objective Value	0.0293108361
Number of Starts	10
Number of Sample Points	800
Number of Distinct Optima	5
Random Seed Used	1234
Optimality Error	5E-7
Infeasibility	2.6183296E-7
Presolve Time	0.00
Solution Time	0.04

[1]	x.MSINIT	x
1	-0.23856	1.1166
2	2.93912	1.2204
3	1.08486	1.5378
4	-1.11094	1.9728
5	1.80540	1.7911

Alternatively, the following SAS statements show how you can add the NODES= option in the PERFORMANCE statement to run this example in distributed mode.

NOTE: SAS High-Performance Optimization software must be installed before you can invoke the MULTISTART option in distributed mode.

```
proc optmodel;
   var x{i in 1..5} >= -5 <= 5 init -2;

   min f=(x[1] - 1)^2 + (x[1] - x[2])^2 + (x[2] - x[3])^3 +
         (x[3] - x[4])^4 + (x[4] - x[5])^4;

   con g1: x[1] + x[2]^2 + x[3]^3 =  2 + 3*sqrt(2);
   con g2: x[2] + x[4]   - x[3]^2 = -2 + 2*sqrt(2);
   con g3: x[1]*x[5] = 2;

   performance nodes=4 nthreads=4;
   solve with nlp/multistart=(maxstarts=10) seed=1234;
   print x;
quit;
```

The SAS log is displayed in Output 11.6.3.

Output 11.6.3 Progress of the Algorithm as Shown in the Log

```
NOTE: Problem generation will use 16 threads.
NOTE: The problem has 5 variables (0 free, 0 fixed).
NOTE: The problem has 0 linear constraints (0 LE, 0 EQ, 0 GE, 0 range).
NOTE: The problem has 3 nonlinear constraints (0 LE, 3 EQ, 0 GE, 0 range).
NOTE: The OPTMODEL presolver removed 0 variables, 0 linear constraints, and 0
      nonlinear constraints.
NOTE: Using analytic derivatives for objective.
NOTE: Using analytic derivatives for nonlinear constraints.
NOTE: The NLP solver is called.
NOTE: The Interior Point algorithm is used.
NOTE: The MULTISTART option is enabled.
NOTE: The Multistart algorithm is executing in the distributed computing
      environment with 4 worker nodes.
NOTE: The Multistart algorithm is using up to 4 threads.
NOTE: Random number seed 1234 is used.
                   Best       Local  Optimality    Infeasi-  Local  Local
      Start    Objective   Objective       Error      bility  Iters  Status
          1   607.035871  607.035871  8.43836E-7  8.43836E-7      8  Optimal
          2   52.9025792  52.9025792  2.39856E-7  2.84012E-8      7  Optimal
          3   52.9025792  607.035521  4.17297E-7  4.17297E-7      8  Optimal
          4   52.9025015  52.9025015  9.03254E-7  9.03254E-7      5  Optimal
          5   52.9025015  52.9025794  6.69221E-8  3.26907E-9      8  Optimal
          6   52.9025015  64.8739968  2.99586E-7  1.19421E-7      8  Optimal
          7   0.02931083  0.02931083  2.53217E-7  3.54155E-9     10  Optimal
          8   0.02931083  52.9025785  4.76385E-8  1.40303E-8      7  Optimal
          9   0.02931083  52.9026071  3.70891E-7  3.70891E-7     11  Optimal
         10   0.02931083  52.9026134  4.08103E-7  4.08103E-7      8  Optimal
NOTE: The Multistart algorithm generated 1600 sample points.
NOTE: 7 distinct local optima were found.
NOTE: The best objective value found by local solver = 0.0293108314.
NOTE: The solution found by local solver with objective = 0.0293108314 was
      returned.
```

Output 11.6.4 shows the summaries and solution. Note that the “Performance Information” table shows that four computing nodes with four threads on each node are used in distributed mode.

Output 11.6.4 Summaries and the Optimal Solution

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Minimization
Objective Function	f
Objective Type	Nonlinear
Number of Variables	5
Bounded Above	0
Bounded Below	0
Bounded Below and Above	5
Free	0
Fixed	0
Number of Constraints	3
Linear LE (<=)	0
Linear EQ (=)	0
Linear GE (>=)	0
Linear Range	0
Nonlinear LE (<=)	0
Nonlinear EQ (=)	3
Nonlinear GE (>=)	0
Nonlinear Range	0

Solution Summary	
Solver	Multistart NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Optimal
Objective Value	0.0293108314
Number of Starts	10
Number of Sample Points	1600
Number of Distinct Optima	7
Random Seed Used	1234
Optimality Error	2.5321667E-7
Infeasibility	3.5415495E-9
Presolve Time	0.00
Solution Time	1.64

[1]	x
1	1.1167
2	1.2204
3	1.5378
4	1.9727
5	1.7911

Output 11.6.4 *continued*

Performance Information

Host Node	<< your grid host >>
Execution Mode	Distributed
Number of Compute Nodes	4
Number of Threads per Node	4

Example 11.7: Maximum Likelihood Weibull Estimation

The following data are taken from Lawless (1982, p. 193) and represent the number of days that it took rats that were painted with a carcinogen to develop carcinoma. The last two observations are censored data from a group of 19 rats.

```
data pike;
   input days cens @@;
   datalines;
143  0  164  0  188  0  188  0
190  0  192  0  206  0  209  0
213  0  216  0  220  0  227  0
230  0  234  0  246  0  265  0
304  0  216  1  244  1
;
```

Suppose you want to compute the maximum likelihood estimates of the scale parameter σ (α in Lawless), the shape parameter c (β in Lawless), and the location parameter θ (μ in Lawless). The observed likelihood function of the three-parameter Weibull transformation (Lawless 1982, p. 191) is

$$L(\theta,\sigma,c) = \frac{c^m}{\sigma^m}\prod_{i\in D}\left(\frac{t_i-\theta}{\sigma}\right)^{c-1}\prod_{i=1}^{n}\exp\left(-\left(\frac{t_i-\theta}{\sigma}\right)^c\right)$$

where n is the number of individuals involved in the experiment, D is the set of individuals whose lifetimes are observed, $m = |D|$, and t_i is defined by the data set. Then the log-likelihood function is

$$l(\theta,\sigma,c) = m\log c - mc\log\sigma + (c-1)\sum_{i\in D}\log(t_i-\theta) - \sum_{i=1}^{n}\left(\frac{t_i-\theta}{\sigma}\right)^c$$

For $c < 1$, the logarithmic terms become infinite as $\theta \uparrow \min_{i\in D}(t_i)$. That is, $l(\theta,\sigma,c)$ is unbounded. Thus our interest is restricted to c values greater than or equal to 1. Further, for the logarithmic terms to be defined, it is required that $\sigma > 0$ and $\theta < \min_{i\in D}(t_i)$.

The following PROC OPTMODEL call specifies the maximization of the log-likelihood function for the three-parameter Weibull estimation:

```
proc optmodel;
   set OBS;
   num days {OBS};
   num cens {OBS};
   read data pike into OBS=[_N_] days cens;
   var sig   >= 1.0e-6 init 10;
   var c     >= 1.0e-6 init 10;
   var theta >= 0 <= min {i in OBS: cens[i] = 0} days[i] init 10;

   impvar fi {i in OBS} =
      (if cens[i] = 0 then
         log(c) - c * log(sig) + (c - 1) * log(days[i] - theta)
      )
    - ((days[i] - theta) / sig)^c;
   max logf = sum {i in OBS} fi[i];

   set VARS = 1.._NVAR_;
   num mycov {i in VARS, j in 1..i};
   solve with NLP / covest=(cov=2 covout=mycov);
   print sig c theta;
   print mycov;
   create data covdata from [i j]={i in VARS, j in 1..i}
      var_i=_VAR_[i].name var_j=_VAR_[j].name mycov;
   num std_error {i in VARS} = sqrt(mycov[i,i]);
   num t_stat    {i in VARS} = _VAR_[i].sol / std_error[i];
   num p_value   {i in VARS} = 2 * (1 - cdf('T', t_stat[i], card(OBS)));
   print _VAR_.name _VAR_ std_error t_stat p_value;
quit;
```

The solution is displayed in Output 11.7.1. The solution that the NLP solver obtains closely matches the local maximum $\theta^* = 122$, $\sigma^* = 108.4$, and $c^* = 2.712$ that are given in Lawless (1982, p. 193). Note that the result from the NLP solver is platform-dependent and might not match this result.

Output 11.7.1 Three-Parameter Weibull Estimation Results

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Maximization
Objective Function	logf
Objective Type	Nonlinear
Number of Variables	3
Bounded Above	0
Bounded Below	2
Bounded Below and Above	1
Free	0
Fixed	0
Number of Constraints	0

Output 11.7.1 *continued*

Solution Summary	
Solver	NLP
Algorithm	Interior Point
Objective Function	logf
Solution Status	Optimal
Objective Value	-87.32424712
Optimality Error	5E-7
Infeasibility	0
Iterations	24
Presolve Time	0.00
Solution Time	0.03

sig	c	theta
108.38	2.7115	122.03

mycov

	1	2	3
1	1259.9683		
2	35.5374	1.3312	
3	-1056.9857	-31.6629	977.6164

[1]	_VAR_.NAME	_VAR_	std_error	t_stat	p_value
1	sig	108.3827	35.4960	3.0534	0.00653971
2	c	2.7115	1.1538	2.3501	0.02972368
3	theta	122.0259	31.2669	3.9027	0.00095684

Example 11.8: Finding an Irreducible Infeasible Set

This example demonstrates the use of the IIS= option to locate an irreducible infeasible set. Suppose you have the following nonlinear programming problem:

$$\begin{array}{lrcccccl}
\text{minimize} & & x_1^4 & + & x_2^4 & + & x_3^4 & \\
\text{subject to} & & x_1 & + & x_2 & & & \geq 10 \quad (\text{c1}) \\
& & x_1 & & & + & x_3 & \leq 4 \quad (\text{c2}) \\
& 4 \leq & & & x_2 & + & x_3 & \leq 5 \quad (\text{c3}) \\
& & x_1^2 & & & + & x_3 & \leq 5 \quad (\text{c4}) \\
& & & & & x_1, & x_2 & \geq 0 \\
& & & & 0 & \leq & x_3 & \leq 3
\end{array}$$

It is easy to verify that the following three linear constraints and one variable bound form an IIS for this problem:

$$\begin{array}{rcrcrcrl}
x_1 & + & x_2 & & & \geq & 10 & \text{(c1)} \\
x_1 & & & + & x_3 & \leq & 4 & \text{(c2)} \\
& & x_2 & + & x_3 & \leq & 5 & \text{(c3)} \\
& & & & x_3 & \geq & 0 &
\end{array}$$

You can formulate the problem and call the NLP solver by using the following statements:

```
proc optmodel;
   /* declare variables */
   var x{1..3} >= 0;

   /* upper bound on variable x[3] */
   x[3].ub = 3;

   /* objective function */
   min f = x[1]^4 + x[2]^4 + x[3]^4;

   /* constraints */
   con c1: x[1] + x[2] >= 10;
   con c2: x[1] + x[3] <= 4;
   con c3: 4 <= x[2] + x[3] <= 5;
   con c4: x[1]^2 + x[3] <= 5;

   solve with nlp / iis = on;

   print x.status;
   print c1.status c2.status c3.status;

   expand / IIS;
quit;
```

The SAS log output is shown in Output 11.8.1. Note that the PROC OPTMODEL presolver is disabled because the IIS= option is enabled. Also, a warning message is displayed to alert the user that the nonlinear constraints are ignored for the purpose of detecting an IIS.

Output 11.8.1 Finding an IIS: Original Problem

```
NOTE: The OPTMODEL presolver is disabled when the IIS= option is enabled.
NOTE: Problem generation will use 16 threads.
NOTE: The problem has 3 variables (0 free, 0 fixed).
NOTE: The problem has 3 linear constraints (1 LE, 0 EQ, 1 GE, 1 range).
NOTE: The problem has 6 linear constraint coefficients.
NOTE: The problem has 1 nonlinear constraints (1 LE, 0 EQ, 0 GE, 0 range).
WARNING: The nonlinear constraints are ignored because the IIS= option is
         enabled.
NOTE: The NLP solver is called.
NOTE: The LP solver is called.
NOTE: The IIS= option is enabled.
                           Objective
      Phase Iteration        Value         Time
       P 1          1    6.000000E+00         0
       P 1          3    9.998343E-01         0
NOTE: Applying the IIS sensitivity filter.
NOTE: The sensitivity filter removed 1 constraints and 3 variable bounds.
NOTE: Applying the IIS deletion filter.
NOTE: Processing constraints.
      Processed     Removed      Time
              0           0         0
              1           0         0
              2           0         0
              3           0         0
NOTE: Processing variable bounds.
      Processed     Removed      Time
              0           0         0
              1           0         0
              2           0         0
              3           0         0
NOTE: The deletion filter removed 0 constraints and 0 variable bounds.
NOTE: The IIS= option found this problem to be infeasible.
NOTE: The IIS= option found an irreducible infeasible set with 1 variables and
      3 constraints.
NOTE: The IIS solve time is 0.00 seconds.
```

There are two ways to display the rows and columns that are included in the IIS. One way is to use the PRINT statement to print the value of the .status suffix for each variable and constraint. The more straightforward approach is to use the EXPAND statement with the IIS option.

The “Solution Summary” table and the output of both approaches appear in Output 11.8.2.

Output 11.8.2 Solution Summary and PRINT Statement Output

The OPTMODEL Procedure

Solution Summary	
Solver	NLP
Algorithm	IIS
Objective Function	f
Solution Status	Infeasible
Iterations	10
Iterations2	0
Presolve Time	0.00
Solution Time	0.00

[1]	x.STATUS
1	
2	
3	I_L

c1.STATUS	c2.STATUS	c3.STATUS
I_L	I_U	I_U

The “Solution Summary” table shows that the problem is infeasible. As you can see, the lower bound of variable x_3, the lower bound of constraint c1, and the upper bounds of constraints c2 and c3 form an IIS.

Making any of the components in the preceding IIS nonbinding removes the infeasibility from the IIS. Because there could be multiple IISs, you would want to remove the infeasibility from the preceding IIS and call the NLP solver with the IIS= option enabled again to see whether there is any other IIS. The following statements show how to modify the original PROC OPTMODEL statements to set the upper bound of constraint c3 to infinity, represented by CONSTANT(’BIG’), and invoke the NLP IIS detection:

```
   /* relax upper bound on constraint c3 */
   c3.ub = constant('BIG');

   solve with nlp / iis = on;

   print x.status;
   print c1.status c2.status c3.status;
```

The SAS log output for the modified problem is shown in Output 11.8.3.

Output 11.8.3 Finding an IIS: Modified Problem

```
NOTE: The OPTMODEL presolver is disabled when the IIS= option is enabled.
NOTE: Problem generation will use 16 threads.
NOTE: The problem has 3 variables (0 free, 0 fixed).
NOTE: The problem has 3 linear constraints (1 LE, 0 EQ, 2 GE, 0 range).
NOTE: The problem has 6 linear constraint coefficients.
NOTE: The problem has 1 nonlinear constraints (1 LE, 0 EQ, 0 GE, 0 range).
WARNING: The nonlinear constraints are ignored because the IIS= option is
         enabled.
NOTE: The NLP solver is called.
NOTE: The LP solver is called.
NOTE: The IIS= option is enabled.
                           Objective
      Phase Iteration        Value         Time
       P 1          1    1.400000E+01         0
       P 1          2    0.000000E+00         0
NOTE: The IIS= option found this problem to be feasible.
NOTE: The IIS solve time is 0.00 seconds.
```

The “Solution Summary” table and the output of the PRINT statements appear in Output 11.8.4. As you can see, both the variable status and constraint status tables are empty. There is no other IIS, and the problem becomes feasible.

Output 11.8.4 Solution Summary and PRINT Statement Output

The OPTMODEL Procedure

Solution Summary	
Solver	NLP
Algorithm	IIS
Objective Function	f
Solution Status	Feasible
Iterations	2
Iterations2	0
Presolve Time	0.00
Solution Time	0.00

[1]	x.STATUS
1	
2	
3	

c1.STATUS	c2.STATUS	c3.STATUS

References

Akrotirianakis, I., and Rustem, B. (2005). "Globally Convergent Interior-Point Algorithm for Nonlinear Programming." *Journal of Optimization Theory and Applications* 125:497–521.

Armand, P., Gilbert, J. C., and Jan-Jégou, S. (2002). "A BFGS-IP Algorithm for Solving Strongly Convex Optimization Problems with Feasibility Enforced by an Exact Penalty Approach." *Mathematical Programming* 92:393–424.

Armand, P., and Omheni, R. (2017a). "A Globally and Quadratically Convergent Primal-Dual Augmented Lagrangian Algorithm for Equality Constrained Optimization." *Optimization Methods and Software* 32:1–21.

Armand, P., and Omheni, R. (2017b). "A Mixed Logarithmic Barrier-Augmented Lagrangian Method for Nonlinear Optimization." *Journal of Optimization Theory and Applications* 173:523–547.

Erway, J., Gill, P. E., and Griffin, J. D. (2007). "Iterative Solution of Augmented Systems Arising in Interior Point Methods." *SIAM Journal on Optimization* 18:666–690.

Forsgren, A., and Gill, P. E. (1998). "Primal-Dual Interior Methods for Nonconvex Nonlinear Programming." *SIAM Journal on Optimization* 8:1132–1152.

Forsgren, A., Gill, P. E., and Wright, M. H. (2002). "Interior Methods for Nonlinear Optimization." *SIAM Review* 44:525–597.

Gill, P. E., and Robinson, D. P. (2010). "A Primal-Dual Augmented Lagrangian." *Computational Optimization and Applications* 47:1–25.

Gould, N. I. M., Orban, D., and Toint, P. L. (2003). "CUTEr and SifDec: A Constrained and Unconstrained Testing Environment, Revised." *ACM Transactions on Mathematical Software* 29:373–394.

Hock, W., and Schittkowski, K. (1981). *Test Examples for Nonlinear Programming Codes*. Vol. 187 of Lecture Notes in Economics and Mathematical Systems. Berlin: Springer-Verlag.

Lawless, J. F. (1982). *Statistical Methods and Methods for Lifetime Data*. New York: John Wiley & Sons.

Moré, J. J., Garbow, B. S., and Hillstrom, K. E. (1981). "Testing Unconstrained Optimization Software." *ACM Transactions on Mathematical Software* 7:17–41.

Nocedal, J., and Wright, S. J. (1999). *Numerical Optimization*. New York: Springer-Verlag.

Vanderbei, R. J. (1999). "LOQO: An Interior Point Code for Quadratic Programming." *Optimization Methods and Software* 11:451–484.

Wächter, A., and Biegler, L. T. (2006). "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming." *Mathematical Programming* 106:25–57.

Wright, S. J. (1997). *Primal-Dual Interior-Point Methods*. Philadelphia: SIAM.

Yamashita, H. (1998). "A Globally Convergent Primal-Dual Interior Point Method for Constrained Optimization." *Optimization Methods and Software* 10:443–469.

Subject Index

active-set method
    overview, 559
active-set primal-dual algorithm, 550

constrained optimization
    overview, 554
covariance matrix, 546

interior point algorithm
    overview, 555
interior point direct algorithm
    overview, 557
interior point direct primal-dual algorithm, 550
interior point primal-dual algorithm, 550

macro variable
    OROPTMODEL, 567
Moore-Penrose conditions, 564

NLP solver
    covariance matrix, 546, 563
    eigenvalue tolerance, 547
    singularity criterion, 546, 548
    solver termination messages, 567
NLP solver examples
    finding an irreducible infeasible set, 589
    maximum likelihood Weibull estimation, 587
    solving equality-constrained problems, 574
    solving highly nonlinear optimization problems, 570
    solving large-scale NLP problems, 579
    solving NLP problems that have several local minima, 581
    solving NLP problems with range constraints, 576
    solving unconstrained and bound-constrained optimization problems, 572

OPTMODEL procedure, NLP solver
    details, 553
    functional summary, 545
    introductory examples, 533
    macro variable _OROPTMODEL_, 567
    solver, 550
    technique, 550
OROPTMODEL macro variable, 567

singularity, 563
    absolute singularity criterion, 546
    relative singularity criterion, 548

Syntax Index

ALGORITHM= option
  SOLVE WITH NLP statement, 550
ASINGULAR= option
  SOLVE WITH NLP statement, 546, 563

COV= option
  SOLVE WITH NLP statement, 546
COVEST=() option
  SOLVE WITH NLP statement, 562
COVEST=() option
  SOLVE WITH NLP statement, 546
COVOUT= option
  SOLVE WITH NLP statement, 547
COVSING= option
  SOLVE WITH NLP statement, 547, 564

FEASTOL= option
  SOLVE WITH NLP statement, 551

HESSTYPE= option
  SOLVE WITH NLP statement, 551

IIS= option
  SOLVE WITH NLP statement, 551

LOGFREQ= option
  SOLVE WITH NLP statement, 550

MAXITER= option
  SOLVE WITH NLP statement, 552
MAXTIME= option
  SOLVE WITH NLP statement, 552
MSINGULAR= option
  SOLVE WITH NLP statement, 548, 563
BNDRANGE= suboption
  SOLVE WITH NLP statement, 549
MS=() option
  SOLVE WITH NLP statement, 548
MULTISTART=() option
  SOLVE WITH NLP statement, 548
DISTTOL= suboption
  SOLVE WITH NLP statement, 549
LOGLEVEL= suboption
  SOLVE WITH NLP statement, 549
MAXSTARTS= suboption
  SOLVE WITH NLP statement, 550
MAXTIME= suboption
  SOLVE WITH NLP statement, 549
PRINTLEVEL= suboption
  SOLVE WITH NLP statement, 549

NDF= suboption
  SOLVE WITH NLP statement, 548
NTERMS= option
  SOLVE WITH NLP statement, 548

OBJLIMIT= option
  SOLVE WITH NLP statement, 552
OPTMODEL procedure, NLP solver
  syntax, 545
OPTTOL= option
  SOLVE WITH NLP statement, 552

PRESERVEINIT= option
  SOLVE WITH NLP statement, 552
PRINTFREQ= option
  SOLVE WITH NLP statement, 550

SEED= option
  SOLVE WITH NLP statement, 548
SIGSQ= option
  SOLVE WITH NLP statement, 548, 564
SOLTYPE= option
  SOLVE WITH NLP statement, 551
SOLVE WITH NLP statement
  ALGORITHM= option, 550
  ASINGULAR= option, 546
  COV= option, 546
  COVEST=() option, 546
  COVOUT= option, 547
  COVSING= option, 547
  FEASTOL= option, 551
  HESSTYPE= option, 551
  IIS= option, 551
  LOGFREQ= option, 550
  MAXITER= option, 552
  MAXTIME= option, 552
  MSINGULAR= option, 548
  BNDRANGE= suboption, 549
  MS=() option, 548
  MULTISTART=() option, 548
  DISTTOL= suboption, 549
  LOGLEVEL= suboption, 549
  MAXSTARTS= suboption, 550
  MAXTIME= suboption, 549
  PRINTLEVEL= suboption, 549
  NDF= suboption, 548
  NTERMS= option, 548

OBJLIMIT= option, 552
OPTTOL= option, 552
PRESERVEINIT= option, 552
PRINTFREQ= option, 550
SEED= option, 548
SIGSQ= option, 548
SOLTYPE= option, 551
SOLVER= option, 550
TECH= option, 550
TECHNIQUE= option, 550
TIMETYPE= option, 552
VARDEF= option, 548

SOLVER= option
SOLVE WITH NLP statement, 550

TECH= option
SOLVE WITH NLP statement, 550

TECHNIQUE= option
SOLVE WITH NLP statement, 550

TIMETYPE= option
SOLVE WITH NLP statement, 552

VARDEF= option
SOLVE WITH NLP statement, 548, 562