SNOPT

Contents

Articles

References

SNOPT

TOMLAB /SNOPT (hereafter referred to as SNOPT) is a general-purpose system for constrained optimization. It minimizes a linear or nonlinear function subject to bounds on the variables and sparse linear or nonlinear constraints. It is suitable for large-scale linear and quadratic programming and for linearly constrained optimization, as well as for general nonlinear programs of the form (SparseNP)

$$\begin{aligned} &\min_x && f_0(x) \\ &\text{subject to} && l \leq \begin{pmatrix} x \\ f(x) \\ A_L x \end{pmatrix} \leq u \end{aligned}$$

where l and u are constant lower and upper bounds, $f_0(x)$ is a smooth scalar objective function, A_L is a sparse matrix, and $f(x)$ is a vector of smooth nonlinear constraint functions $\{f_i(x)\}$. An optional parameter maximize may specify that $f_0(x)$ should be maximized instead of minimized.

Ideally, the first derivatives (gradients) of $f_0(x)$ and $f_i(x)$ should be known and coded by the user.

Note that upper and lower bounds are specified for all variables and constraints. This form allows full generality in specifying various types of constraint. Special values are used to indicate absent bounds ($l_j = -\infty$ or $u_j = +\infty$ for appropriate j). Free variables and free constraints ("free rows") are ones that have both bounds infinite. Fixed variables and equality constraints have $l_j = u_j$.

Problem types

If $f_0(x)$ is linear and $f(x)$ is absent, SparseNP is a *linear program* (LP) and SNOPT applies the primal simplex method. Sparse basis factors are maintained by LUSOL as in MINOS.

If only the objective is nonlinear, the problem is *linearly constrained* (LC) and tends to solve more easily than the general case with nonlinear constraints (NC). For both cases, SNOPT applies a sparse sequential quadratic programming (SQP) method, using limited-memory quasi-Newton approximations to the Hessian of the Lagrangian. The merit function for steplength control is an augmented Lagrangian, as in the dense SQP solver NPSOL.

In general, SNOPT requires less matrix computation than NPSOL and fewer evaluations of the functions than the nonlinear algorithms in MINOS.

It is suitable for nonlinear problems with thousands of constraints and variables, and is efficient if many constraints and bounds are active at a solution. (Thus, ideally there should not be thousands of degrees of freedom.)

Description of the SQP method

- Description of the SQP method

Optional parameters

- Optional parameters

File Output

- File Output

SNOPT Description of the SQP method

This page is part of the SNOPT Manual. See SNOPT.

Here we summarize the main features of the SQP algorithm used in SNOPT and introduce some terminology used in the description of the subroutine and its arguments. The SQP algorithm is fully described by Gill, Murray and Saunders.

Constraints and slack variables

The upper and lower bounds on the m components of $f(x)$ and $AL\ x$ are said to define the *general constraints* of the problem. SNOPT converts the general constraints to equalities by introducing a set of *slack variables*. For example, the linear constraint $5 \leq 2x_1 + 3x_2 \leq +\infty$ is replaced by $2x_1 + 3x_2 - s_1 = 0$ together with the bounded slack $5 \leq s_1 \leq +\infty$. SparseNP can therefore be written in the equivalent form

$$\min_{x,s} \quad f_0(x)$$

$$\text{subject to} \quad \begin{pmatrix} f(x) \\ A_L x \end{pmatrix} - s = 0, l \leq \begin{pmatrix} x \\ s \end{pmatrix} \leq u.$$

The general constraints become the equalities $f(x) - s_N = 0$ and $A_L x - s_L = 0$, where s_L and s_N are known as the *linear* and *nonlinear* slacks.

Major iterations

The basic structure of the SQP algorithm involves *major* and *minor* iterations. The major iterations generate a sequence of iterates $\{x_k\}$ that satisfy the linear constraints and converge to a point that satisfies the first-order conditions for optimality. At each iterate a QP subproblem is used to generate a search direction towards the next iterate x_{k+1}. The constraints of the subproblem are formed from the linear constraints $A_L x - s_L = 0$ and the nonlinear constraint linearization

$$f(x_k) + f'(x_k)(x - x_k) - s_N = 0,$$

where $f'(xk)$ denotes the *Jacobian matrix*, whose elements are the first derivatives of $f(x)$ evaluated at x_k. The QP constraints therefore comprise the m linear constraints

$$\begin{aligned} f'(x_k)x - s_N &= f(x_k) + f(x_k)'x_k, \\ A_L X - s_L &= 0, \end{aligned}$$

where x and s are bounded above and below by u and l as before. If the $m \times n$ matrix A and m-vector b are defined as

$$A = \begin{pmatrix} f'(x_k) \\ A_L \end{pmatrix} \quad and \quad b = \begin{pmatrix} -f(x_k) + f'(x_k)x_k \\ 0 \end{pmatrix},$$

then the QP subproblem can be written as

$$\min_{x,s} \quad q(x) \quad \text{subject to} \quad Ax - s = b, \quad l \leq \begin{pmatrix} x \\ s \end{pmatrix} \leq u,$$

where $q(x)$ is a quadratic approximation to a modified Lagrangian function.

Minor iterations

Solving the QP subproblem is itself an iterative procedure. The iterations of the QP solver are the *minor* iterations of the SQP method. At each minor iteration, the constraints $Ax - s = b$ are (conceptually) partitioned into the form

$$Bx_B + Sx_S + Nx_N = b,$$

where the *basis matrix B* is square and nonsingular. The elements of x_B, x_S and x_N are called the *basic*, *superbasic* and *nonbasic* variables respectively; they are a permutation of the elements of x and s. At a QP solution, the basic and superbasic variables will lie somewhere between their bounds, while the nonbasic variables will normally be equal to one of their bounds. At each iteration, x_S is regarded as a set of independent variables that are free to move in any desired direction, namely one that will improve the value of the QP objective (or the sum of infeasibilities). The basic variables are then adjusted in order to ensure that (x, s) continues to satisfy $Ax - s = b$. The number of superbasic variables (n_S, say) therefore indicates the number of degrees of freedom remaining after the constraints have been satisfied. In broad terms, n_S is a measure of *how nonlinear* the problem is. In particular, n_S will always be zero for LP problems.

If it appears that no improvement can be made with the current definition of B, S and N, a nonbasic variable is selected to be added to S, and the process is repeated with the value of n_S increased by one. At all stages, if a basic or superbasic variables encounters one of its bounds, the variables is made nonbasic and the value of n_S is decreased by one.

Associated with each of the m equality constraints $Ax - s = b$ are the *dual variables* p. Similarly, each variable in (x, s) has an associated *reduced gradient* d_j. The reduced gradients for the variables x are the quantities $g - A^T\pi$, where g is the gradient of the QP objective, and the reduced gradients for the slacks are the dual variables p. The QP subproblem is optimal if $d_j = 0$ for all nonbasic variables at their lower bounds, $d_j = 0$ for all nonbasic variables at their upper bounds, and $d_j = 0$ for other variables, including superbasics. In practice, an *approximate* QP solution $(\hat{x}_k, \hat{t}_k, \hat{\pi}_k)$ is found by relaxing these conditions.

The merit function

After a QP subproblem has been solved, new estimates of the SparseNP solution are computed using a line search on the augmented Lagrangian merit function

$$f(x, s, \pi) = f_0(x) - \pi^T(f(x) - s_N) + (f(x) - S_N)^T D(f(x) - S_N)$$

where D is a diagonal matrix of penalty parameters ($D_{ii} >= 0$). If (x_k, s_k, p_k) denotes the current solution estimate and $(\hat{x}_k, \hat{t}_k, \hat{pi}_k)$ denotes the QP solution, the line search determines a step $\alpha_k (0 < \alpha_k \leq 1)$ such that the new point

$$\begin{pmatrix} x_{k+1} \\ s_{k+1} \\ \pi_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ s_k \\ \pi_k \end{pmatrix} + \alpha_k \begin{pmatrix} \hat{x}_k - x_k \\ \hat{s}_k - s_k \\ \hat{\pi}_k - \pi_k \end{pmatrix}$$

gives a *sufficient decrease* in the merit function. When necessary, the penalties in D are increased by the minimum-norm perturbation that ensures descent for M.

As in NPSOL, s_N is adjusted to minimize the merit function as a function of s prior to the solution of the QP subproblem.

Treatment of constraint infeasibilities

SNOPT makes explicit allowance for infeasible constraints. First, infeasible *linear* constraints are detected by solving the linear program (FLP)

$$\begin{aligned} &\min_{x,v,w} \quad e^T(v+w) \\ &\text{subject to} \quad l \le \begin{pmatrix} x \\ A_L x - v + w \end{pmatrix} \le u, \quad v \ge 0, \quad w \ge 0, \end{aligned}$$

where *e* is a vector of ones, and the nonlinear constraint bounds are temporarily excluded from *l* and *u*. This is equivalent to minimizing the sum of the general linear constraint violations subject to the bounds on *x*. (The sum is the £1 -norm of the linear constraint violations. In the linear programming literature, the approach is called *elastic programming*.)

The linear constraints are infeasible if the optimal solution of FLP has $v \neq 0$ or $w \neq 0$. SNOPT then terminates without computing the nonlinear functions.

Otherwise, all subsequent iterates satisfy the linear constraints. (Such a strategy allows linear constraints to be used to define a region in which the functions can be safely evaluated.) SNOPT proceeds to solve SparseNP as given, using search directions obtained from the sequence of QP subproblems.

If a QP subproblem proves to be infeasible or unbounded (or if the dual variables π for the nonlinear constraints become large), SNOPT enters "elastic" mode and thereafter solves the problem (NP(γ))

$$\begin{aligned} &\min_{x,v,w} \quad f_0(x) + \gamma e^T(v+w) \\ &\text{subject to} \quad l \le \begin{pmatrix} x \\ f(x) - v + w \\ A_L x \end{pmatrix} \le u \quad v \ge 0, \quad w \ge 0 \end{aligned}$$

where γ is a nonnegative parameter (the *elastic weight*), and $f_0(x) + \gamma e^T(v+w)$ is called a *composite objective* (the ℓ_1 penalty function for the nonlinear constraints).

The value of γ may increase automatically by multiples of 10 if the optimal *v* and *w* continue to be nonzero. If γ is sufficiently large, this is equivalent to minimizing the sum of the nonlinear constraint violations subject to the linear constraints and bounds. A similar ℓ_1 formulation of SparseNP is fundamental to the ℓ_1 QP algorithm of Fletcher. See also Conn.

The initial value of γ is controlled by the optional parameters Elastic mode and Elastic weight.

SNOPT Optional parameters

This page is part of the SNOPT Manual. See SNOPT.

The performance of each SNOPT interface is controlled by a number of parameters or "options". Each option has a default value that should be appropriate for most problems. The options are normally set in *optPar* or *Prob.SOL.optPar* before calling the solver. For special situations it is possible to specify non-standard values for some or all of the options. This options may be defined in a file called a *SPECS file*.

The SPECS file

The specs file contains a list of option definitions, using data in the following general form:

```
Begin options
   Iterations limit            500
   Minor feasibility tolerance      1.0e-7
   Solution                  Yes
End options
```

We call such data a SPECS file because it specifies various options. The file starts with the keyword Begin and ends with End. Each line specifies a single option in free format, using one or more items as follows:

1. A *keyword* (required for all options).
2. A *phrase* (one or more words) that qualifies the keyword (only for some options).
3. A *number* that specifies an integer or real value (only for some options). Such numbers may be up to 16 contiguous characters in Fortran 77's I, F, E or D formats, terminated by a space.

The items may be entered in upper or lower case or a mixture of both. Some of the keywords have synonyms, and certain abbreviations are allowed, as long as there is no ambiguity. Blank lines and comments may be used to improve readability. A comment begins with an asterisk (*), which may appear anywhere on a line. All subsequent characters on the line are ignored.

It may be useful to include a comment on the first (Begin) line of the file. This line is echoed to the SUMMARY file.

Most of the options described in the next section should be left at their default values for any given model. If experimentation is necessary, we recommend changing just one option at a time.

SPECS file checklist and defaults

The following example SPECS file shows all valid *keywords* and their *default values*. The keywords are grouped according to the function they perform.

Some of the default values depend on *E*, the relative precision of the machine being used. The values given here correspond to double-precision arithmetic on most current machines ($\epsilon \approx 2.22 \times 10^{-16}$). Similar values would apply to any machine having about 15 decimal digits of precision.

```
BEGIN checklist of SPECS file parameters and their default values
* Printing
  Major print level                1       * one-line major iteration log
  Minor print level                 1       * one-line minor iteration log
  Print file                       9       *
  Summary file                       6       * typically the screen
  Print frequency                  100       * minor iterations log on PRINT file
```

```
  Summary frequency                  100       * minor iterations log on SUMMARY file
  Solution                          Yes       * on the PRINT file
* Suppress options listing                    * default: options are listed
  System information                  No       * prints more system information

* Problem specification
  Minimize                              * (opposite of Maximize)
* Feasible point                        * (alternative to Max or Min)
  Infinite Bound size                 1.0e+20 *

* Convergence Tolerances
  Major feasibility tolerance            1.0e-6       * target nonlinear constraint violation
Major optimality tolerance             1.0e-6       * target complementarity gap
Minor feasibility tolerance            1.0e-6       * for satisfying the QP bounds

* Derivative checking
Verify level                          0       * cheap check on gradients
Start objective check at col            1       *
Stop objective check at col             n       *
Start constraint check at col           1       *
Stop constraint check at col            n       *

* Scaling
Scale option                          1       * linear constraints and variables
Scale tolerance                    0.9       *
* Scale Print                               * default: scales are not printed

* Other Tolerances
  Crash tolerance                   0.1       *
  Linesearch tolerance                   0.9       * smaller for more accurate search
  Pivot tolerance                   3.7e-11 * e^2/3

* QP subproblems
  QPSolver                           Cholesky * default
  Crash option                          3       * first basis is essentially triangular
  Elastic mode                          No       * until it seems necessary
  Elastic weight                    1.0e+4       * used only during elastic mode
  Iterations limit                    10000       * or 20m if that is more
  Partial price                     1       * 10 for large LPs

* SQP method
  Major iterations limit              1000       * or m if that is more
  Minor iterations limit              500       *
  Major step limit                    2.0       *
  Superbasics limit                    n+1
  Hessian dimension                    750       * or Superbasics limit if that is less
  Derivative level                    3       *
```

```
  Derivative linesearch                  *
* Nonderivative linesearch                  *
  Function precision                   3.0e-13 * ?0.8 (almost full accuracy)
  Difference interval                   5.5e-7       * (Function precision)^1/2
  Central difference interval             6.7e-5       * (Function precision)^1/3
  New superbasics limit             99       * controls early termination of QPs
  Objective row                   ObjRow       * row number of objective in F(x)
  Penalty parameter                   0.0       * initial penalty parameter
  Proximal point method             1       * satisfies linear constraints near x0
  Violation limit                   10.0       * unscaled constraint violation limit
  Unbounded step size                  1.0e+18 *
  Unbounded objective                  1.0e+15 *

* Hessian approximation
  Hessian                        full memory * default if n <= 75
  Hessian                      limited memory * default if n > 75
  Hessian frequency                  999999       * for full Hessian (never reset)
  Hessian updates                  20       * for limited memory Hessian
  Hessian flush                   999999       * no flushing

* Frequencies
  Check frequency                   60       * test row residuals ||Ax - s||
  Expand frequency                   10000       * for anti-cycling procedure
  Factorization frequency             50       * 100 for LPs
  Save frequency                   100       * save basis map

* LU options
  LU factor tolerance                  10.0       * limits size of multipliers in L
  LU update tolerance                  10.0       * the same during updates
  LU singularity tolerance             3.25e-11 *
  LU partial pivoting                       * default pivot strategy
  LU rook pivoting                        * use rook pivoting for the LU
  LU complete pivoting                       * use complete pivoting for the LU

* Partitions of cw, iw, rw
  Total character workspace             lencw       *
  Total integer workspace             leniw       *
  Total real workspace                   lenrw       *
  User character workspace             500       *
  User integer workspace             500       *
  User real workspace                  500       *

* Miscellaneous
  Debug level                         0       * for developers
  Timing level                         3       * prints cpu times
  End of SPECS file checklist
```

Description of the optional parameters

The following is an alphabetical list of the options that may appear in the SPECS file, and a description of their effect. In the description of the options we use the notation of the problem format SparseNP to refer to the objective and constraint functions.

```
Central difference interval r   Default = ε^1/3 ≈ 6.0e-6
```

When Derivative level < 3), the central-difference interval r is used near an optimal solution to obtain more accurate (but more expensive) estimates of gradients. Twice as many function evaluations are required compared to forward differencing. The interval used for the jth variable is $h_j = r(1 + |xj|)$. The resulting derivative estimates should be accurate to $O(r^2)$, unless the functions are badly scaled.

```
Check frequency  i  Default = 60
```

Every ith minor iteration after the most recent basis factorization, a numerical test is made to see if the current solution x satisfies the general linear constraints (including linearized nonlinear constraints, if any). The constraints are of the form $Ax - s = b$, where s is the set of slack variables. To perform the numerical test, the residual vector $r = b - Ax + s$ is computed. If the largest component of r is judged to be too large, the current basis is refactorized and the basic variables are recomputed to satisfy the general constraints more accurately.

Check frequency 1 is useful for debugging purposes, but otherwise this option should not be needed.

```
Crash option     i  Default = 3
Crash tolerance  r  Default = 0.1
```

Except on restarts, a CRASH procedure is used to select an initial basis from certain rows and columns of the constraint matrix (A - I). The Crash option i determines which rows and columns of A are eligible initially, and how many times CRASH is called. Columns of $-I$ are used to pad the basis where necessary.

i	Meaning
0	The initial basis contains only slack variables: $B = I$.
1	CRASH is called once, looking for a triangular basis in all rows and columns of the matrix A.
2	CRASH is called twice (if there are nonlinear constraints). The first call looks for a triangular basis in linear rows, and the iteration proceeds with simplex iterations until the linear constraints are satisfied. The Jacobian is then evaluated for the first major iteration and CRASH is called again to find a triangular basis in the nonlinear rows (retaining the current basis for linear rows).
3	CRASH is called up to three times (if there are nonlinear constraints). The first two calls treat *linear equalities* and *linear inequalities* separately. As before, the last call treats nonlinear rows before the first major iteration.

If i >= 1, certain slacks on inequality rows are selected for the basis first. (If i >= 2, numerical values are used to exclude slacks that are close to a bound.) CRASH then makes several passes through the columns of A, searching for a basis matrix that is essentially triangular. A column is assigned to "pivot" on a particular row if the column contains a suitably large element in a row that has not yet been assigned. (The pivot elements ultimately form the diagonals of the triangular basis.) For remaining unassigned rows, slack variables are inserted to complete the basis.

The Crash tolerance r allows the starting procedure CRASH to ignore certain "small" nonzeros in each column of A. If $a_{\max}$ is the largest element in column j, other nonzeros a_{ij} in the column are ignored if $|a_{ij}| = a_{\max} \times r$. (To be meaningful, r should be in the range $0 <= r < 1$.)

When $r > 0.0$, the basis obtained by CRASH may not be strictly triangular, but it is likely to be nonsingular and almost triangular. The intention is to obtain a starting basis containing more columns of A and fewer (arbitrary) slacks. A feasible solution may be reached sooner on some problems.

For example, suppose the first *m* columns of *A* are the matrix shown under LU factor tolerance; i.e., a tridiag- onal matrix with entries -1, 4, -1. To help CRASH choose all *m* columns for the initial basis, we would specify Crash tolerance *r* for some value of $r > 1/4$.

```
Derivative level  i  Default = 3
```

The keyword Derivative level specifies which nonlinear function gradients are known analytically and will be supplied to SNOPT. This is normally automatically set by TOMLAB.

i	Meaning
3	All objective and constraint gradients are known.
2	All constraint gradients are known, but some or all components of the objective gradient are unknown.
1	The objective gradient is known, but some or all of the constraint gradients are unknown.
0	Some components of the objective gradient are unknown and some of the constraint gradients are unknown.

The value $i = 3$ should be used whenever possible. It is the most reliable and will usually be the most efficient.

If $i = 0$ or 2, SNOPT will *estimate* the missing components of the objective gradient, using finite differences. However, it could increase the total run-time substantially, and there is less assurance that an acceptable solution will be located. If the nonlinear variables are not well scaled, it may be necessary to specify a nonstandard Difference interval (see below).

If $i = 0$ or 1, SNOPT will estimate missing elements of the Jacobian. For each column of the Jacobian, one call is needed to estimate all missing elements in that column, if any. If Jacobian = sparse and the sparsity pattern of the Jacobian happens to be

$$\begin{pmatrix} * & * & & * \\ & ? & ? & \\ * & & ? & \\ & * & & * \end{pmatrix}$$

where * indicates known gradients and ? indicates unknown elements, SNOPT will use one call to estimate the missing element in column 2, and another call to estimate both missing elements in column 3. No calls are needed for columns 1 and 4.

At times, central differences are used rather than forward differences.

```
Derivative linesearch    Default

Nonderivative linesearch
```

At each major iteration a line search is used to improve the merit function. A Derivative linesearch uses safeguarded cubic interpolation and requires both function and gradient values to compute estimates of the step a_k . If some analytic derivatives are not provided, or a Nonderivative linesearch is specified, SNOPT employs a line search based upon safeguarded quadratic interpolation, which does not require gradient evaluations.

A nonderivative line search can be slightly less robust on difficult problems, and it is recommended that the default be used if the functions and derivatives can be computed at approximately the same cost. If the gradients are very expensive relative to the functions, a nonderivative line search may give a significant decrease in computation time.

Comment: Derivative linesearch is only default if analytic derivatives are provided.

```
Difference interval  h1  Default = ε^1/2 ≈ 1.5e-8
```

This alters the interval h_1 that is used to estimate gradients by forward differences in the following circumstances:

- In the initial ("cheap") phase of verifying the problem derivatives.
- For verifying the problem derivatives.
- For estimating missing derivatives.

In all cases, a derivative with respect to x_j is estimated by perturbing that component of x to the value $x_j + h_1(1 + |x_j|)$, and then evaluating $f0(x)$ or $f(x)$ at the perturbed point. The resulting gradient estimates should be accurate to $O(h1)$ unless the functions are badly scaled. Judicious alteration of $h1$ may sometimes lead to greater accuracy.

```
Elastic mode No   Default

Elastic mode Yes
```

Normally SNOPT initiates elastic mode only when it seems necessary. Option Yes causes elastic mode to be entered from the beginning.

```
Elastic weight ω Default = 10^4
```

This keyword determines the initial weight γ associated with problem NP(γ).

At major iteration k, if elastic mode has not yet started, a scale factor $\sigma_k = 1 + ||g(x_k)||_\infty$is defined from the current objective gradient. Elastic mode is then started if the QP subproblem is infeasible, or the QP dual variables are larger in magnitude than $\sigma_k\omega$. The QP is re-solved in elastic mode with $\gamma = \sigma_k\omega$.
Thereafter, major iterations continue in elastic mode until they converge to a point that is optimal for problem NP(γ). If the point is feasible for SparseNP ($v = w = 0$), it is declared locally optimal. Otherwise, γ is increased by a factor of 10 and major iterations continue. If γ has already reached a maximum allowable value, SparseNP is declared locally infeasible.

```
Expand frequency i Default = 10000
```

This option is part of the EXPAND anti-cycling procedure designed to make progress even on highly degenerate problems.

For linear models, the strategy is to force a positive step at every iteration, at the expense of violating the bounds on the variables by a small amount. Suppose that the Minor feasibility tolerance is δ. Over a period of i iterations, the tolerance actually used by SNOPT increases from 0.5δ to δ(in steps of $0.5\delta/i$).

For nonlinear models, the same procedure is used for iterations in which there is only one superbasic variable. (Cycling can occur only when the current solution is at a vertex of the feasible region.) Thus, zero steps are allowed if there is more than one superbasic variable, but otherwise positive steps are enforced.

Increasing i helps reduce the number of slightly infeasible nonbasic variables (most of which are eliminated during a resetting procedure). However, it also diminishes the freedom to choose a large pivot element (see Pivot tolerance).

```
Factorization frequency k Default = 50
```

At most k basis changes will occur between factorizations of the basis matrix.

- With linear programs, the basis factors are usually updated every iteration. The default k is reasonable for typical problems. Higher values up to $k = 100$ (say) may be more efficient on problems that are extremely sparse and well scaled.
- When the objective function is nonlinear, fewer basis updates will occur as an optimum is approached. The number of iterations between basis factorizations will therefore increase. During these iterations a test is made regularly (according to the Check frequency) to ensure that the general constraints are satisfied. If necessary the basis will be refactorized before the limit of k updates is reached.

```
Feasibility tolerance           t  Default = 1.0e-6
see Minor feasibility tolerance
```

```
Feasibility point
see Minimize
```

```
Function precision  ε_R  Default = ε^0.8 ≈ 3.7e-11
see Minimize
```

The *relative function precision* ε_R is intended to be a measure of the relative accuracy with which the nonlinear functions can be computed. For example, if $f(x)$ is computed as 1000.56789 for some relevant x and if the first 6 significant digits are known to be correct, the appropriate value for *ER* would be 1.0e-6.

(Ideally the functions $f(x)$ or $F_i(x)$ should have magnitude of order 1. If all functions are substantially *less* than 1 in magnitude, *ER* should be the *absolute* precision. For example, if $f(x)$ = 1.23456789e-4 at some point and if the first 6 significant digits are known to be correct, the appropriate value for *ER* would be 1.0e-10.)

- The default value of *ER* is appropriate for simple analytic functions.
- In some cases the function values will be the result of extensive computation, possibly involving an iterative procedure that can provide rather few digits of precision at reasonable cost. Specifying an appropriate Function precision may lead to savings, by allowing the line search procedure to terminate when the difference between function values along the search direction becomes as small as the absolute error in the values.

```
Hessian dimension  r  Default = Superbasics limit or 750
```

This specifies that an $r \times r$ triangular matrix R is to be available for use by the Cholesky QP solver (to define the reduced Hessian according to $R^T R = Z^T H Z$).

```
Hessian full memory     Default = Full if n_1 <= 75
Hessian limited memory
```

These options select the method for storing and updating the approximate Hessian. (SNOPT uses a quasi-Newton approximation to the Hessian of the Lagrangian. A BFGS update is applied after each major iteration.)

If Hessian full memory is specified, the approximate Hessian is treated as a dense matrix and the BFGS updates are applied explicitly. This option is most efficient when the number of nonlinear variables n_1 is not too large (say, less than 75). In this case, the storage requirement is fixed and one can expect 1-step Q-superlinear convergence to the solution.

Hessian limited memory should be used on problems where n_1 is very large. In this case a limited-memory procedure is used to update a diagonal Hessian approximation H_r a limited number of times. (Updates are accumulated as a list of vector pairs. They are discarded at regular intervals after H_r has been reset to their diagonal.)

```
Hessian frequency  i  Default = 999999
```

If Hessian Full is selected and *i* BFGS updates have already been carried out, the Hessian approximation is reset to the identity matrix. (For certain problems, occasional resets may improve convergence, but in general they should not be necessary.)

Hessian Full memory and Hessian frequency = 20 have a similar effect to Hessian Limited memory and Hessian updates = 20 (except that the latter retains the current diagonal during resets).

```
Hessian updates  i  Default = 20
```

If Hessian Limited memory is selected and *i* BFGS updates have already been carried out, all but the diagonal elements of the accumulated updates are discarded and the updating process starts again.

Broadly speaking, the more updates stored, the better the quality of the approximate Hessian. However, the more vectors stored, the greater the cost of each QP iteration. The default value is likely to give a robust algorithm without significant expense, but faster convergence can sometimes be obtained with significantly fewer updates (e.g., *i* = 5).

```
Iterations limit  k  Default = max{10000, 20m}
```

This is the maximum number of minor iterations allowed (i.e., iterations of the simplex method or the QP algorithm), summed over all major iterations.

```
Infinite Bound size  r  Default = 1.0e+20
```

If *r* > 0, *r* defines the "infinite" bound BigBnd in the definition of the problem constraints. Any upper bound greater than or equal to BigBnd will be regarded as plus infinity (and similarly for a lower bound less than or equal to -BigBnd). If *r* <= 0, the default value is used.

```
Linesearch tolerance  t  Default = 0.9
```

This controls the accuracy with which a steplength will be located along the direction of search each iteration. At the start of each line search a target directional derivative for the merit function is identified. This parameter determines the accuracy to which this target value is approximated.

- *t* must be a real value in the range 0.0 <= *t* <= 1.0.
- The default value *t* = 0.9 requests just moderate accuracy in the line search.
- If the nonlinear functions are cheap to evaluate, a more accurate search may be appropriate; try *t* = 0.1, 0.01 or 0.001. The number of major iterations might decrease.
- If the nonlinear functions are expensive to evaluate, a less accurate search may be appropriate. *If all gradients are known*, try *t* = 0.99. (The number of major iterations might increase, but the total number of function evaluations may decrease enough to compensate.)
- If not all gradients are known, a moderately accurate search remains appropriate. Each search will require only 1-5 function values (typically), but many function calls will then be needed to estimate missing gradients for the next iteration.

```
Log frequency  k  Default = 100
```

see Print frequency

```
LU factor tolerance  r1  Default = 100.0 (LP) or 3.99 (NLP)
LU factor tolerance  r1  Default = 10.0 (LP) or 3.99 (NLP)
```

These tolerances affect the stability and sparsity of the basis factorization $B = LU$ during refactorization and updating, respectively. They must satisfy $r_1, r_2 >= 1.0$. The matrix L is a product of matrices of the form

$$\begin{pmatrix} 1 & \\ \mu & 1 \end{pmatrix},$$

where the multipliers μ satisfy $|\mu| <= r_i$. Smaller values of *ri* favor stability, while larger values favor sparsity. The default values usually strike a good compromise.

- For large and relatively dense problems, $r_1 = 5.0$ (say) may give a useful improvement in stability without impairing sparsity to a serious degree.

- For certain very regular structures (e.g., band matrices) it may be necessary to reduce r_1 and/or r_2 in order to achieve stability. For example, if the columns of *A* include a submatrix of the form

$$\begin{pmatrix} 4 & -1 & & & & \\ -1 & 4 & -1 & & & \\ & -1 & 4 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 4 & -1 \\ & & & & -1 & 4 \end{pmatrix},$$

both r_1 and r_2 should be in the range $1.0 \leq r_i < 4.0$.

```
LU Partial Pivoting  Default

LU Rook Pivoting

LU Complete Pivoting
```

The LU factorization implements a Markowitz-type search for a pivot that locally minimizes the fill-in subject to a threshold pivoting stability criterion. The default option is to use threshold partial pivoting. The options LU rook pivoting and LU complete pivoting are more expensive than partial pivoting but are more stable and better at revealing rank.

```
LU density tolerance      r1 Default = 0.5

LU singularity tolerance  r2 Default = ε^0.67 ≈ 3.25e-11
```

The density tolerance r_1 is used during LU factorization of the basis matrix. Columns of *L* and rows of *U* are formed one at a time, and the remaining rows and columns of the basis are altered appropriately. At any stage, if the density of the remaining matrix exceeds *r*1 , the Markowitz strategy for choosing pivots is terminated. The remaining matrix is factored by a dense LU procedure. Raising the density tolerance towards 1.0 may give slightly sparser LU factors, with a slight increase in factorization time.

The singularity tolerance r_2 helps guard against ill-conditioned basis matrices. When the basis is refactorized, the diagonal elements of *U* are tested as follows: if $|U_{jj}| \leq r_2$or $|U_{jj}| < r_2 \max_i |U_{ij}|$, the *j*th column of the basis is replaced by the corresponding slack variable. (This is most likely to occur after a restart, or at the start of a major iteration.)

In some cases, the Jacobian matrix may converge to values that make the basis exactly singular. (For example, a whole row of the Jacobian could be zero at an optimal solution.) Before exact singularity occurs, the basis could become very ill-conditioned and the optimization could progress very slowly (if at all). Setting a larger tolerance r_2 = 1.0e-5, say, may help cause a judicious change of basis.

```
Major feasibility tolerance εr Default = 1.0e-6
```

This specifies how accurately the nonlinear constraints should be satisfied. The default value of 1.0e-6 is appropriate when the linear and nonlinear constraints contain data to about that accuracy.

Let rowerr be the maximum nonlinear constraint violation, normalized by the size of the solution. It is required to satisfy

$$\text{rowerr} = \max_i \text{viol}_i / ||x|| \leq \epsilon_r,$$

where $vviol_i$is the violation of the *i*th nonlinear constraint (*i* = 1 : nnCon).

In the major iteration log, rowerr appears as the quantity labeled "Feasibl". If some of the problem functions are known to be of low accuracy, a larger Major feasibility tolerance may be appropriate.

```
Major optimality tolerance ε_d Default = 1.0e-6
```

This specifies the final accuracy of the dual variables. On successful termination, SNOPT will have computed a solution (*x, s,* π) such that

$$\text{maxComp} = \max_j \text{Comp}_j/||\pi|| \leq \epsilon_d,$$

where Comp_j is an estimate of the complementarity slackness for variable j ($j = 1 : n + m$). The values Comp_j are computed from the final QP solution using the reduced gradients $d_j = g_j - \pi^T a_j$ (where *gj* is the *j*th component of the objective gradient, *aj* is the associated column of the constraint matrix (*A* - *I*), and *&pi* is the set of QP dual variables):

$$\text{Comp}_j = \begin{array}{ll} d_j \min\{x_j - l_j, 1\} & if d_j \geq 0 \\ -d_j \min\{u_j - x_j, 1\} & if d_j < 0. \end{array}$$

In the major iteration log, maxComp appears as the quantity labeled "Optimal".

```
Major iterations limit k Default = max{1000, m}
```

This is the maximum number of major iterations allowed. It is intended to guard against an excessive number of linearizations of the constraints.

```
Major print level p Default = 00001
```

This controls the amount of output to the PRINT and SUMMARY files each major iteration. Major print level 1 gives normal output for linear and nonlinear problems, and Major print level 11 gives addition details of the Jacobian factorization that commences each major iteration.

In general, the value being specified may be thought of as a binary number of the form

```
Major print level JFDXbs
```

where each letter stands for a digit that is either 0 or 1 as follows:

s	a single line that gives a summary of each major iteration. (This entry in JFDXbs is not strictly binary since the summary line is printed whenever JFDXbs = 1).
b	BASIS statistics, i.e., information relating to the basis matrix whenever it is refactorized. (This output is always provided if JFDXbs = 10).
X	x_k, the nonlinear variables involved in the objective function or the constraints.
D	π_k, the dual variables for the nonlinear constraints.
F	$F(x_k)$, the values of the nonlinear constraint functions.
J	$J(x_k)$, the Jacobian matrix.

To obtain output of any items JFDXbs, set the corresponding digit to 1, otherwise to 0.

If J=1, the Jacobian matrix will be output column-wise at the start of each major iteration. Column *j* will be preceded by the value of the corresponding variable x_j and a key to indicate whether the variable is basic, superbasic or nonbasic. (Hence if J=1, there is no reason to specify X=1 unless the objective contains more nonlinear variables than the Jacobian.) A typical line of output is

```
3      1.250000D+01 BS      1      1.00000E+00      4      2.00000E+00
```

which would mean that x_3 is basic at value 12.5, and the third column of the Jacobian has elements of 1.0 and 2.0 in rows 1 and 4.

Major print level 0 suppresses most output, except for error messages.

```
Major step limit  r  Default = 2.0
```

This parameter limits the change in x during a line search. It applies to all nonlinear problems, once a "feasible solution" or "feasible subproblem" has been found.

1. A line search determines a step a over the range $0 < a = \beta$, where β is 1 if there are nonlinear constraints, or the step to the nearest upper or lower bound on x if all the constraints are linear. Normally, the first steplength tried is $a1 = \min(1, \beta)$.
2. In some cases, such as $f(x) = aebx$ or $f(x) = axb$, even a moderate change in the components of x can lead to floating-point overflow. The parameter r is therefore used to define a limit $\bar{\beta} = r(1 + x)/p$ (where p is the search direction), and the first evaluation of $f(x)$ is at the potentially smaller steplength $a1 = \min(1, \bar{\beta}, \beta)$.
3. Wherever possible, upper and lower bounds on x should be used to prevent evaluation of nonlinear functions at meaningless points. The Major step limit provides an additional safeguard. The default value $r = 2.0$ should not affect progress on well behaved problems, but setting $r = 0.1$ or 0.01 may be helpful when rapidly varying functions are present. A "good" starting point may be required. An important application is to the class of nonlinear least-squares problems.
4. In cases where several local optima exist, specifying a small value for r may help locate an optimum near the starting point.

```
Minimize        Default
Maximize
Feasible point
```

The keywords Minimize and Maximize specify the required direction of optimization. It applies to both linear and nonlinear terms in the objective.

The keyword feasible point means "Ignore the objective function" while finding a feasible point for the linear and nonlinear constraints. It can be used to check that the nonlinear constraints are feasible without altering the call to SNOPT.

```
Minor iterations limit  k  Default = 500
Maximize
Feasible point
```

If the number of minor iterations for the optimality phase of the QP subproblem exceeds k, then all nonbasic QP variables that have not yet moved are frozen at their current values and the reduced QP is solved to optimality.

Note that more than k minor iterations may be necessary to solve the reduced QP to optimality. These extra iterations are necessary to ensure that the terminated point gives a suitable direction for the line search.

In the major iteration log, a t at the end of a line indicates that the corresponding QP was artificially terminated using the limit k.

Note that Iterations limit defines an independent *absolute* limit on the *total* number of minor iterations (summed over all QP subproblems).

```
Minor feasibility tolerance  t  Default = 1.0e-6
```

SNOPT tries to ensure that all variables eventually satisfy their upper and lower bounds to within the tolerance t. This includes slack variables. Hence, general linear constraints should also be satisfied to within t.

Feasibility with respect to nonlinear constraints is judged by the Major feasibility tolerance (not by t).

- If the bounds and linear constraints cannot be satisfied to within *t*, the problem is declared *infeasible*. Let sInf be the corresponding sum of infeasibilities. If sInf is quite small, it may be appropriate to raise *t* by a factor of 10 or 100. Otherwise, some error in the data should be suspected.
- Nonlinear functions will be evaluated only at points that satisfy the bounds and linear constraints. If there are regions where a function is undefined, every attempt should be made to eliminate these regions from the problem.

For example, if $f(x) = \sqrt{x_1} + \log x_2$, it is essential to place lower bounds on both variables. If *t* = 1.0e-6, the bounds $x_1 \geq 10^{-5}$and $x_2 \geq 10^{-4}$might be appropriate. (The log singularity is more serious. In general, keep *x* as far away from singularities as possible.)

- If Scale option = 1, feasibility is defined in terms of the *scaled* problem (since it is then more likely to be meaningful).
- In reality, SNOPT uses *t* as a feasibility tolerance for satisfying the bounds on *x* and *s* in each QP subproblem. If the sum of infeasibilities cannot be reduced to zero, the QP subproblem is declared infeasible. SNOPT is then in *elastic mode* thereafter (with only the linearized nonlinear constraints defined to be elastic). See the Elastic options.

```
Minor print level  k  Default = 1
```

This controls the amount of output to the PRINT and SUMMARY files during solution of the QP subproblems. The value of *k* has the following effect:

0	No minor iteration output except error messages.
>= 1	A single line of output each minor iteration (controlled by Print frequency and Summary frequency).
>= 10	Basis factorization statistics generated during the periodic refactorization of the basis (see Factorization frequency). Statistics for the *first factorization* each major iteration are controlled by the Major print level.

```
New superbasics limit  i  Default = 99
```

This option causes early termination of the QP subproblems if the number of free variables has increased significantly since the first feasible point. If the number of new superbasics is greater than *i* the nonbasic variables that have not yet moved are frozen and the resulting smaller QP is solved to optimality.

In the major iteration log, a "T" at the end of a line indicates that the QP was terminated early in this way.

```
Partial price  i  Default = 10 (LP) or 1 (NLP)
```

This parameter is recommended for large problems that have significantly more variables than constraints. It reduces the work required for each "pricing" operation (when a nonbasic variable is selected to become superbasic).

- When *i* = 1, all columns of the constraint matrix *(A - I)* are searched.
- Otherwise, *A* and *I* are partitioned to give *i* roughly equal segments *Aj* , *Ij* (*j* = 1 to *i*). If the previous pricing search was successful on *Aj* , *Ij* , the next search begins on the segments *Aj*+1 , *Ij*+1 . (All subscripts here are modulo *i*.)
- If a reduced gradient is found that is larger than some dynamic tolerance, the variable with the largest such reduced gradient (of appropriate sign) is selected to become superbasic. If nothing is found, the search continues on the next segments *Aj*+2 , *Ij*+2 , and so on.
- Partial price *t* (or *t*/2 or *t*/3) may be appropriate for time-stage models having *t* time periods.

```
Pivot tolerance  r   Default = ε^(2/3) ≈ 3.7e-11
```

During solution of QP subproblems, the pivot tolerance is used to prevent columns entering the basis if they would cause the basis to become almost singular.

- When x changes to $x + \alpha p$ for some search direction p, a "ratio test" is used to determine which component of x reaches an upper or lower bound first. The corresponding element of p is called the pivot element.
- Elements of p are ignored (and therefore cannot be pivot elements) if they are smaller than the pivot tolerance r.
- It is common for two or more variables to reach a bound at essentially the same time. In such cases, the Minor Feasibility tolerance (say t) provides some freedom to maximize the pivot element and thereby improve numerical stability. Excessively small values of t should therefore not be specified.
- To a lesser extent, the Expand frequency (say f) also provides some freedom to maximize the pivot element. Excessively *large* values of f should therefore not be specified.

```
Proximal point method i Default = 1
```

i = 1 or 2 specifies minimization of $||x - x_0||_1$ or $\frac{1}{2}||x - x_0||_2^2$ when the starting point x_0 is changed to satisfy the linear constraints (where $x0$ refers to nonlinear variables).

```
QPSolver Cholesky Default

QPSolver CG       Default = 1

QPSolver QN       Default = 1
```

Specifies the algorithm used to solve the QP subproblem. QPSolver Cholesky uses the active-set QP method of SQOPT, which holds the full Cholesky factor R of the reduced Hessian Z^THZ. As the QP iterations proceed, the dimension of R changes as the number of superbasic variables changes. If it is necessary that the number of superbasic variables increases beyond the value of Hessian dimension, the reduced Hessian cannot be stored and the solver switches to QPSolver CG. The Cholesky solver is reactivated if the number of superbasics stabilizes at a value less than Hessian dimension.

QPSolver QN solves the QP subproblem using a quasi-Newton method similar to MINOS. In this case, R is the factor of a quasi-Newton approximate Hessian.

QPSolver CG uses an active-set method similar to QPSolver QN, but uses the conjugate-gradient method to solve all systems involving the reduced Hessian.

- The Cholesky QP solver is the most robust, but may require a significant amount of computation if the number of superbasics is large.
- The quasi-Newton QP solver does not require the computation of the R at the start of each QP and may be appropriate when the number of superbasics is large, but each QP subproblem requires relatively few minor iterations.
- The conjugate-gradient QP solver is appropriate for problems with large numbers of degrees of freedom.

Reduced Hessian dimension	i	Default = min{750, n1 + 1}
see Hessian dimension		
Scale option	i	Default = 2 (LP) or 1 (NLP)
Scale tolerance	r	Default = 0.9
Scale Print		

Three scale options are available as follows:

i	Meaning
0	No scaling. This is recommended if it is known that *x* and the constraint matrix (and Jacobian) never have very large elements (say, larger than 1000).
1	Linear constraints and variables are scaled by an iterative procedure that attempts to make the matrix coefficients as close as possible to 1.0 (see Fourer [11]). This will sometimes improve the performance of the solution procedures.
2	All constraints and variables are scaled by the iterative procedure. Also, an additional scaling is performed that takes into account columns of *(A - I)* that are fixed or have positive lower bounds or negative upper bounds. If nonlinear constraints are present, the scales depend on the Jacobian at the first point that satisfies the linear constraints. Scale option 2 should therefore be used only if (a) a good starting point is provided, and (b) the problem is not highly nonlinear.

Scale tolerance affects how many passes might be needed through the constraint matrix. On each pass, the scaling procedure computes the ratio of the largest and smallest nonzero coefficients in each column:

$$\rho_j = \max_i |a_{ij}| / \min_i |a_{ij}| \qquad (a_{ij} \neq 0).$$

If $\max_j \rho_j$ is less than *r* times its previous value, another scaling pass is performed to adjust the row and column scales. Raising *r* from 0.9 to 0.99 (say) usually increases the number of scaling passes through *A*. At most 10 passes are made.

Scale Print causes the row-scales *r*(*i*) and column-scales *c*(*j*) to be printed. The scaled matrix coefficients are $\bar{a}_{ij} = a_{ij}c(j)/r(i)$, and the scaled bounds on the variables and slacks are $\bar{l}_j = l_j/c(j), \bar{u}_j = u_j/c(j)$

```
QPSolver Cholesky  Default

QPSolver CG        Default = 1

QPSolver QN        Default = 1


Solution                                    Yes

Solution                                    No

Solution  If Optimal, Infeasible, or Unbounded
```

```
Start Objective Check at Column  k Default = 1
Start Constraint Check at Column k Default = 1
Stop Objective Check at Column   l Default = 'n'1
Stop Constraint Check at Column  l Default = 'n'1
```

If Verify level > 0, they may be used to abbreviate the verification of individual derivative elements. For example:

- If the first 100 objective gradients appeared to be correct in an earlier run, and if you have just found a bug in that ought to fix up the 101-th component, then you might as well specify Start Objective Check at Column 101. Similarly for columns of the Jacobian.
- If the first 100 variables occur nonlinearly in the constraints, and the remaining variables are nonlinear only in the objective, then one must set the first 100 components of g(*) to zero, but these hardly need to be verified. The above option would again be appropriate.

```
Summary file      f Default = 6
Summary frequency k Default = 100
```

If $f > 0$ and Minor print level > 0, a line of the QP iteration log will be output to file f every kth minor iteration.

```
Superbasics limit i Default = n1 + 1
```

This places a limit on the storage allocated for superbasic variables. Ideally, i should be set slightly larger than the "number of degrees of freedom" expected at an optimal solution.

For linear programs, an optimum is normally a basic solution with no degrees of freedom. (The number of variables lying strictly between their bounds is no more than m, the number of general constraints.) The default value of i is therefore 1.

For nonlinear problems, the number of degrees of freedom is often called the "number of independent variables".

Normally, i need not be greater than $n_1 + 1$, where n_1 is the number of nonlinear variables. For many problems, i may be considerably smaller than n_1. This will save storage if $n1$ is very large.

```
System Information No  Default
System Information Yes
```

This option allows the knowledgeable user to print some additional information on the progress of the major and minor iterations.

```
Timing level i Default = 3
```

$i = 0$ suppresses output of cpu times. (Intended for installations with dysfunctional timing routines.)

```
Unbounded objective value f_max Default = 1.0e+15
Unbounded step size       a_max Default = 1.0e+18
```

These parameters are intended to detect unboundedness in nonlinear problems. (They may not achieve that purpose!) During a line search, f_0 is evaluated at points of the form $x + \alpha p$, where x and p are fixed and α varies.

if $|f0|$ exceeds f_{max} or a exceeds a_{max}, iterations are terminated with the exit message Problem is unbounded (or badly scaled).

If singularities are present, unboundedness in $f_0(x)$ may be manifested by a floating-point overflow (during the evaluation of $f_0(x + \alpha p)$), before the test against $f_{\max}$ can be made.

Unboundedness in x is best avoided by placing finite upper and lower bounds on the variables.

```
Verify level l Default = 0
```

This option refers to finite-difference checks on the derivatives computed by the user-provided routines. Derivatives are checked at the first point that satisfies all bounds and linear constraints.

l	Meaning
0	Only a "cheap" test will be performed.
1	Individual gradients will be checked (with a more reliable test). A key of the form "OK" or "Bad?" indicates whether or not each component appears to be correct.
2	Individual columns of the problem Jacobian will be checked.
3	Options 2 and 1 will both occur (in that order).

Verify level 3 should be specified whenever a new function routine is being developed. The Start and Stop keywords may be used to limit the number of nonlinear variables checked. Missing derivatives are not checked, so they result in no overhead.

```
Violation limit t Default = 10
```

This keyword defines an absolute limit on the magnitude of the maximum constraint violation after the line search. On completion of the line search, the new iterate xk+1 satisfies the condition

$$v_i(x_{k+1}) \leq \tau \max\{1, v_i(x_0)\},$$

where x_0 is the point at which the nonlinear constraints are first evaluated and $v_i(x)$ is the ith nonlinear constraint violation $v_i(x) = \max(0, l_i - f_i(x), f_i(x) - u_i)$.

The effect of this violation limit is to restrict the iterates to lie in an *expanded* feasible region whose size depends on the magnitude of τ. This makes it possible to keep the iterates within a region where the objective is expected to be well-defined and bounded below. If the objective is bounded below for all values of the variables, then t may be any large positive value.

SNOPT File Output

This page is part of the SNOPT Manual. See SNOPT.

The files can be directed with the Print file and Summary file options (or suppressed).

The PRINT file

If Print file is set (not done through *optPar*, see the help for calling the solver), the following information is output to the PRINT file during the solution process. All printed lines are less than 131 characters.

- A listing of the SPECS file, if any.
- A listing of the options that were or could have been set in the SPECS file.
- An estimate of the working storage needed and the amount available.
- Some statistics about the problem being solved.
- The storage available for the *LU* factors of the basis matrix.
- A summary of the scaling procedure, if Scale option > 0.
- Notes about the initial basis resulting from a CRASH procedure or a BASIS file.
- The major iteration log.
- The minor iteration log.
- Basis factorization statistics.
- The EXIT condition and some statistics about the solution obtained.
- The printed solution, if requested.

The last five items are described in the following sections.

The major iteration log

If Major print level > 0, one line of information is output to the PRINT file every kth minor iteration, where k is the specified Print frequency (default $k = 1$).

Label	Description
Itns	The cumulative number of minor iterations.
Major	The current major iteration number.
Minors	is the number of iterations required by both the feasibility and optimality phases of the QP subproblem. Generally, Minors will be 1 in the later iterations, since theoretical analysis predicts that the correct active set will be identified near the solution (see §8.2).
Step	The step length a taken along the current search direction p. The variables x have just been changed to $x + ap$. On reasonably well-behaved problems, the unit step will be taken as the solution is approached.
nCon	The number of times user subroutines have been called to evaluate the nonlinear problem functions. Evaluations needed for the estimation of the derivatives by finite differences are not included. nCon is printed as a guide to the amount of work required for the line search.
Feasible	is the value of rowerr, the maximum component of the scaled nonlinear constraint residual (55). The solution is regarded as acceptably feasible if Feasible is less than the Major feasibility tolerance. In this case, the entry is contained in parenthesis. If the constraints are linear, all iterates are feasible and this entry is not printed.
Optimal	is the value of maxgap, the maximum complementarity gap (56). It is an estimate of the degree of nonoptimality of the reduced costs. Both Feasbl and Optimal are small in the neighborhood of a solution.

MeritFunction	is the value of the augmented Lagrangian merit function (see (53)). This function will decrease at each iteration unless it was necessary to increase the penalty parameters (see §8.2). As the solution is approached, Merit will converge to the value of the objective at the solution. In elastic mode, the merit function is a composite function involving the constraint violations weighted by the elastic weight. If the constraints are linear, this item is labeled Objective, the value of the objective function. It will decrease monotonically to its optimal value.
L+U	The number of nonzeros representing the basis factors L and U on completion of the QP subproblem. If nonlinear constraints are present, the basis factorization $B = LU$ is computed at the start of the first minor iteration. At this stage, LU = lenL + lenU, where lenL, the number of subdiagonal elements in the columns of a lower triangular matrix and lenU is the number of diagonal and superdiagonal elements in the rows of an upper-triangular matrix. As columns of B are replaced during the minor iterations, LU may fluctuate up or down but in general will tend to increase. As the solution is approached and the minor iterations decrease towards zero, LU will reflect the number of nonzeros in the LU factors at the start of the QP subproblem. If the constraints are linear, refactorization is subject only to the Factorize frequency, and LU will tend to increase between factorizations.
BSwap	The number of columns of the basis matrix B that were swapped with columns of S to improve the condition of B. The swaps are determined by an LU factorization of the rectangular matrix $BS = (B S)T$ with stability being favored more than sparsity.
nS	The current number of superbasic variables.
CondHz	An estimate of the condition number of RTR, an estimate of $Z TH Z$, the reduced Hessian of the La- grangian. It is the square of the ratio of the largest and smallest diagonals of the upper triangular matrix R (which is a lower bound on the condition number of $RT R$). Cond Hz gives a rough indica- tion of whether or not the optimization procedure is having difficulty. If E is the relative precision of the machine being used, the SQP algorithm will make slow progress if Cond Hz becomes as large as ε<syo>-1/2 ≈ 10^8 , and will probably fail to find a better solution if Cond Hz reaches $\varepsilon^{-3/4} \approx 10^{12}$. To guard against high values of Cond Hz, attention should be given to the scaling of the variables and the constraints. In some cases it may be necessary to add upper or lower bounds to certain variables to keep them a reasonable distance from singularities in the nonlinear functions or their derivatives.
Penalty	is the Euclidean norm of the vector of penalty parameters used in the augmented Lagrangian merit function (not printed if there are no nonlinear constraints.

The summary line may include additional code characters that indicate what happened during the course of the major iteration.

Code	Meaning
c	Central differences have been used to compute the unknown components of the objective and constraint gradients. A switch to central differences is made if either the line search gives a small step, or x is close to being optimal. In some cases, it may be necessary to re-solve the QP subproblem with the central-difference gradient and Jacobian.
d	During the line search it was necessary to decrease the step in order to obtain a maximum constraint violation conforming to the value of Violation limit.
l	The norm-wise change in the variables was limited by the value of the Major step limit. If this output occurs repeatedly during later iterations, it may be worthwhile increasing the value of Major step limit.
i	If SNOPT is not in elastic mode, an "i" signifies that the QP subproblem is infeasible. This event triggers the start of nonlinear elastic mode, which remains in effect for all subsequent iterations. Once in elastic mode, the QP subproblems are associated with the elastic problem NP(γ). If SNOPT is already in elastic mode, an "i" indicates that the minimizer of the elastic subproblem does not satisfy the linearized constraints. (In this case, a feasible point for the usual QP subproblem may or may not exist.)
M	An extra evaluation of the problem functions was needed to define an acceptable positive-definite quasi- Newton update to the Lagrangian Hessian. This modification is only done when there are nonlinear constraints.
m	This is the same as "M" except that it was also necessary to modify the update to include an augmented Lagrangian term.
n	No positive-definite BFGS update could be found. The approximate Hessian is unchanged from the previous iteration.

R	The approximate Hessian has been reset by discarding all but the diagonal elements. This reset will be forced periodically by the Hessian frequency and Hessian updates keywords. However, it may also be necessary to reset an ill-conditioned Hessian from time to time.
r	The approximate Hessian was reset after ten consecutive major iterations in which no BFGS update could be made. The diagonals of the approximate Hessian are retained if at least one update has been done since the last reset. Otherwise, the approximate Hessian is reset to the identity matrix.
s	A self-scaled BFGS update was performed. This update is always used when the Hessian approximation is diagonal, and hence always follows a Hessian reset.
t	The minor iterations were terminated because of the Minor iterations limit.
T	The minor iterations were terminated because of the New superbasics limit.
u	The QP subproblem was unbounded.
w	A weak solution of the QP subproblem was found.
z	The Superbasics limit was reached.

The minor iteration log

If Minor print level > 0, one line of information is output to the PRINT file every *k*th minor iteration, where *k* is the specified Minor print frequency (default $k = 1$). A heading is printed before the first such line following a basis factorization. The heading contains the items described below. In this description, a PRICE operation is defined to be the process by which a nonbasic variable is selected to become superbasic (in addition to those already in the superbasic set). The selected variable is denoted by jq. Variable jq often becomes basic immediately. Otherwise it remains superbasic, unless it reaches its opposite bound and returns to the nonbasic set.

If Partial price is in effect, variable jq is selected from A_{pp} or I_{pp}, the ppth segments of the constraint matrix (A - I).

Label	Description
Itn	The current iteration number.
RedCost,QPmult	This is the reduced cost (or reduced gradient) of the variable jq selected by PRICE at the start of the present iteration. Algebraically, dj is $dj = gj - pT\,aj$ for j = jq, where gj is the gradient of the current objective function, p is the vector of dual variables for the QP subproblem, and a_j is the jth column of (A - I). Note that dj is the 1-norm of the reduced-gradient vector at the start of the iteration, just after the PRICE operation.
LPstep,QPstep	The step length a taken along the current search direction p. The variables x have just been changed to $x + ap$. If a variable is made superbasic during the current iteration (+SBS > 0), Step will be the step to the nearest bound. During Phase 2, the step can be greater than one only if the reduced Hessian is not positive definite.
nInf	The number of infeasibilities *after* the present iteration. This number will not increase unless the iterations are in elastic mode.
SumInf	If nInf > 0, this is sInf, the sum of infeasibilities after the present iteration. It usually decreases at each nonzero Step, but if nInf decreases by 2 or more, SumInf may occasionally increase. In elastic mode, the heading is changed to Composite Obj, and the value printed decreases monotonically.
rgNorm	The norm of the reduced-gradient vector at the start of the iteration. (It is the norm of the vector with elements dj for variables j in the superbasic set.) During Phase 2 this norm will be approximately zero after a unit step. (The heading is not printed if the problem is linear.)
LPobjective,QPobjective,Elastic QPobj	The QP objective function after the present iteration. In elastic mode, the heading is changed to Elastic QPobj. In either case, the value printed decreases monotoni- cally.
The variable jq selected by PRICE to be added to the superbasic set.	

Pivot	If column a_q replaces the rth column of the basis B, Pivot is the rth element of a vector y satisfying $B_y = a_q$. Wherever possible, Step is chosen to avoid extremely small values of Pivot (since they cause the basis to be nearly singular). In rare cases, it may be necessary to increase the Pivot tolerance to exclude very small elements of y from consideration during the computation of Step.
L+U	The number of nonzeros representing the basis factors L and U . Immediately after a basis factorization $B = LU$, this is lenL+lenU, the number of subdiagonal elements in the columns of a lower triangular matrix and the number of diagonal and superdiagonal elements in the rows of an upper-triangular matrix. Further nonzeros are added to L when various columns of B are later replaced. As columns of B are replaced, the matrix U is maintained explicitly (in sparse form). The value of L will steadily increase, whereas the value of U may fluctuate up or down. Thus, in general, the value of L+U may fluctuate up or down; in general it will tend to increase.)
ncp	The number of compressions required to recover storage in the data structure for U . This includes the number of compressions needed during the previous basis factorization. Normally ncp should increase very slowly. If not, the amount of integer and real workspace available to SNOPT should be increased by a significant amount. As a suggestion, the work arrays iw(*) and rw(*) should be extended by L + U elements.
nS	The current number of superbasic variables. (The heading is not printed if the problem is linear.)
cond Hz	See the major iteration log. (The heading is not printed if the problem is linear.)

|+SBS

otherwise it has become nonbasic.

Basis factorization statistics

If Major print level >= 10, the following items are output to the PRINT file whenever the basis B or the rectangular matrix $B_S = (B\ S)^T$ is factorized before solution of the next QP subproblem.

Note that B_S may be factorized at the start of just some of the major iterations. It is immediately followed by a factorization of B itself.

Gaussian elimination is used to compute a sparse LU factorization of B or B_S , where PLP^T and PUQ are lower and upper triangular matrices for some permutation matrices P and Q. Stability is ensured as described under LU factor tolerance in SNOPT Optional parameters#Description of the optional parameters.

If Minor print level >= 10, the same items are printed during the QP solution whenever the current B is factorized.

Label	Description
Factorize	The number of factorizations since the start of the run.
Demand	A code giving the reason for the present factorization. **Code** / **Meaning** 0 — First LU factorization. 1 — The number of updates reached the Factorization Frequency. 2 — The nonzeros in the updated factors have increased significantly. 7 — Not enough storage to update factors. 10 — Row residuals too large (see the description of Check Frequency). 11 — Ill-conditioning has caused inconsistent results.
Itn	The current minor iteration number.
Nonlin	The number of nonlinear variables in the current basis B.
Linear	The number of linear variables in B.

<table>
<tr><td>Slacks</td><td>The number of slack variables in B.</td></tr>
<tr><td>B BR BS or</td><td>
<table>
<tr><td>BT factorize</td><td>The type of LU factorization.</td></tr>
<tr><td>B</td><td>Periodic factorization of the basis B.</td></tr>
<tr><td>BR</td><td>More careful rank-revealing factorization of B using threshold rook pivoting. This occurs mainly at the start, if the first basis factors seem singular or ill-conditioned. Followed by a normal B factorize.</td></tr>
<tr><td>BS</td><td>BS is factorized to choose a well-conditioned B from the current (B S). Followed by a normal B factorize.</td></tr>
<tr><td>BT</td><td>Same as BS except the current B is tried first and accepted if it appears to be not much more ill-conditioned than after the previous BS factorize.</td></tr>
</table>
</td></tr>
<tr><td>m</td><td>The number of rows in B or BS .</td></tr>
<tr><td>n</td><td>The number of columns in B or BS . Preceded by "=" or ">" respectively.</td></tr>
<tr><td>Elems</td><td>The number of nonzero elements in B or BS .</td></tr>
<tr><td>Amax</td><td>The largest nonzero in B or BS .</td></tr>
<tr><td>Density</td><td>The percentage nonzero density of B or BS .</td></tr>
<tr><td>Merit</td><td>The average Markowitz merit count for the elements chosen to be the diagonals of P U Q. Each merit count is defined to be (c - 1)(r - 1) where c and r are the number of nonzeros in the column and row containing the element at the time it is selected to be the next diagonal. Merit is the average of n such quantities. It gives an indication of how much work was required to preserve sparsity during the factorization.</td></tr>
<tr><td>lenL</td><td>The number of nonzeros in L.</td></tr>
<tr><td>Incres</td><td>The percentage increase in the number of nonzeros in L and U relative to the number of nonzeros in B or BS .</td></tr>
<tr><td>Utri</td><td>is the number of triangular rows of B or BS at the top of U .</td></tr>
<tr><td>lenU</td><td>The number of nonzeros in U .</td></tr>
<tr><td>Ltol</td><td>The maximum subdiagonal element allowed in L. This is the specified LU factor tolerance or a smaller value that is currently being used for greater stability.</td></tr>
<tr><td>Umax</td><td>The maximum nonzero element in U .</td></tr>
<tr><td>Ugrwth</td><td>The ratio Umax/Amax, which ideally should not be substantially larger than 10.0 or 100.0. If it is orders of magnitude larger, it may be advisable to reduce the LU factor tolerance to 5.0, 4.0, 3.0 or 2.0, say (but bigger than 1.0).
As long as Lmax is not large (say 10.0 or less), max{Amax, Umax} / DUmin gives an estimate of the condition number of B. If this is extremely large, the basis is nearly singular. Slacks are used to replace suspect columns of B and the modified basis is refactored.</td></tr>
<tr><td>Ltri</td><td>is the number of triangular columns of B or BS at the left of L.</td></tr>
<tr><td>dense1</td><td>is the number of columns remaining when the density of the basis matrix being factorized reached 0.3.</td></tr>
<tr><td>Lmax</td><td>The actual maximum subdiagonal element in L (bounded by Ltol).</td></tr>
<tr><td>Akmax</td><td>The largest nonzero generated at any stage of the LU factorization. (Values much larger than Amax indicate instability.)</td></tr>
<tr><td>growth</td><td>The ratio Akmax/Amax. Values much larger than 100 (say) indicate instability.</td></tr>
<tr><td>bump</td><td>is the size of the "bump" or block to be factorized nontrivially after the triangular rows and columns of B or BS have been removed.</td></tr>
<tr><td>dense2</td><td>is the number of columns remaining when the density of the basis matrix being factorized reached 0.6. (The Markowitz pivot strategy searches fewer columns at that stage.)</td></tr>
<tr><td>DUmax</td><td>The largest diagonal of P U Q.</td></tr>
<tr><td>DUmin</td><td>The smallest diagonal of P U Q.</td></tr>
<tr><td>condU</td><td>The ratio DUmax/DUmin, which estimates the condition number of U (and of B if Ltol is less than 100, say).</td></tr>
</table>

Cmpressns The number of times the data structure holding the partially factored matrix needed to be compressed to recover unused storage. Ideally this number should be zero. If it is more than 3 or 4, the amount of workspace available to SNOPT should be increased for efficiency.

Crash statistics

If Major print level = 10, the following items are output to the PRINT file. They refer to the number of columns that the CRASH procedure selects during several passes through A while searching for a triangular basis matrix.

Label	Description
Slacks	is the number of slacks selected initially.
Free cols	is the number of free columns in the basis, including those whose bounds are rather far apart.
Preferred	is the number of "preferred" columns in the basis (i.e., hs(j) = 3 for some $j = n$). It will be a subset of the columns for which hs(j) = 3 was specified.
Unit	is the number of unit columns in the basis.
Double	is the number of columns in the basis containing 2 nonzeros.
Triangle	is the number of triangular columns in the basis with 3 or more nonzeros.
Pad	is the number of slacks used to pad the basis (to make it a nonsingular triangle).

EXIT conditions

When any solver or auxiliary routine in the SNOPT package terminates, a message is printed that summarizes what happened during the run. The general form of the output message is:

```
SOLVER  EXIT ''e  ''-- exit  condition
SOLVER  INFO ''i ''-- informational message
```

where e is an integer that labels the particular *exit condition*, and i is one of several alternative *informational messages* that elaborate on the exit condition. For example, the solver may print the message:

```
SNOPTA  EXIT 20 -- the  problem  appears  to  be unbounded
SNOPTA  INFO 21 -- unbounded objective
```

Note that in this example, the exit condition gives a broad definition of what happened, while the informational message is more specific about the cause of the termination.

The number i associated with the informational message is the output value of the argument inform. Note that the number e associated with the exit condition may always be recovered from inform by stripping off the least significant decimal digit.

The list of possible exit conditions are:

0	Finished successfully
10	The problem appears to be infeasible
20	The problem appears to be unbounded
30	Resource limit error
40	Terminated after numerical difficulties
50	Error in the user-supplied functions
60	Undefined user-supplied functions
70	User requested termination
80	Insufficient storage allocated
90	Input arguments out of range
100	Finished successfully (associated with SNOPT auxiliary routines)
110	Errors while processing MPS data
120	Errors while estimating Jacobian structure
130	Errors while reading OPTIONS file
140	System error

The exit conditions 0-20 arise when a solution exists (though it may not be optimal).

Here we describe each message and suggest possible courses of action.

```
EXIT --  0   Finished successfully
INFO --  1   optimality  conditions satisfied
INFO --  2   feasible point found (from option Feasible point only)
INFO --  3   requested  accuracy could  not  be achieved
```

This message should be the cause of guarded optimism! It is certainly preferable to every other message, and we naturally want to believe what it says.

In all cases, a distinct level of caution is in order. For example, if the objective value is much better than expected, we may have obtained an optimal solution to the wrong problem! Almost any item of data could have that effect if it has the wrong value. Verifying that the problem has been defined correctly is one of the more difficult tasks for a model builder. It is good practice in the function subroutines to print any data that is input during the first entry.

If nonlinearities exist, one must always ask the question: could there be more than one local optimum? When the constraints are linear and the objective is known to be convex (e.g., a sum of squares) then all will be well if we are *minimizing* the objective: a local minimum is a global minimum in the sense that no other point has a lower function value. (However, many points could have the *same* objective value, particularly if the objective is largely linear.) Conversely, if we are *maximizing* a convex function, a local maximum cannot be expected to be global, unless there are sufficient constraints to confine the feasible region.

Similar statements could be made about nonlinear constraints defining convex or concave regions. However, the functions of a problem are more likely to be neither convex nor concave. Our advice is always to specify a starting point that is as good an estimate as possible, and to include reasonable upper and lower bounds on all variables, in order to confine the solution to the specific region of interest. We expect modelers to *know something about their problem*, and to make use of that knowledge as they themselves know best.

One other caution about "Optimality conditions satisfied". Some of the variables or slacks may lie outside their bounds more than desired, especially if scaling was requested. Some information concerning the run can be obtained from the short summary given at the end of the print and summary files. Here is an example.

```
SNOPTA  EXIT    0 -- finished successfully
SNOPTA  INFO    1 -- optimality  conditions satisfied,

Problem name                 Toy1
No. of iterations                7   Objective value     -1.0000000008E+00
No. of major iterations          7   Linear objective     0.0000000000E+00
Penalty parameter          3.253E-02 Nonlinear objective -1.0000000008E+00
No. of calls to funobj           9   No. of calls to funcon              9
No. of degenerate steps          0   Percentage                       0.00
Max x                      2 1.0E+00 Max pi                     1 1.2E-01
Max Primal infeas          0 0.0E+00 Max Dual infeas            2 8.0E-10
Nonlinear constraint violn   6.4E-09
```

Max Primal infeas refers to the largest bound infeasibility and which variable is involved. If it is too large, consider restarting with a smaller Minor feasibility tolerance (say 10 times smaller) and perhaps Scale option 0.

Similarly, Max Dual infeas indicates which variable is most likely to be at a non-optimal value. Broadly speaking, if

$$\text{Max Dual infeas}/\text{Max pi} = 10^{-d},$$

then the objective function would probably change in the dth significant digit if optimization could be continued. If d seems too large, consider restarting with a smaller Major optimality tolerance.

Finally, Nonlinear constraint violn shows the maximum infeasibility for nonlinear rows. If it seems too large, consider restarting with a smaller Major feasibility tolerance.

If the requested accuracy could not be achieved, a feasible solution has been found, but the requested accuracy in the dual infeasibilities could not be achieved. An abnormal termination has occurred, but SNOPT is within 10^{-2} of satisfying the Major optimality tolerance. Check that the Major optimality tolerance is not too small.

```
EXIT -- 10   The problem  appears  to  be infeasible
INFO -- 11   infeasible linear constraints
INFO -- 12   infeasible linear equalities
INFO -- 13   nonlinear infeasibilities minimized
INFO -- 14   infeasibilities minimized
```

This exit occurs if SNOPT unable to find a point that satisfies the constraints. When the constraints are linear, these message can probably be trusted. Feasibility is measured with respect to the upper and lower bounds on the variables and slacks. Among all the points satisfying the general constraints $Ax - s = 0$, there is apparently no point that satisfies the bounds on x and s. Violations as small as the Minor feasibility tolerance are ignored, but at least one component of x or s violates a bound by more than the tolerance.

When nonlinear constraints are present, infeasibility is *much* harder to recognize correctly. Even if a feasible solution exists, the current linearization of the constraints may not contain a feasible point. In an attempt to deal with this situation, when solving each QP subproblem, SNOPT is prepared to relax the bounds on the slacks associated with nonlinear rows.

If a QP subproblem proves to be infeasible or unbounded (or if the Lagrange multiplier estimates for the nonlinear constraints become large), SNOPT enters so-called "nonlinear elastic" mode. The subproblem includes the original QP objective and the sum of the infeasibilities-suitably weighted using the Elastic weight parameter. In elastic mode, some of the bounds on the nonlinear rows "elastic"-i.e., they are allowed to violate their specified bounds. Variables subject to elastic bounds are known as *elastic variables*. An elastic variable is free to violate one or both of its original upper or lower bounds. If the original problem has a feasible solution and the elastic weight is sufficiently large, a feasible point eventually will be obtained for the perturbed constraints, and optimization can continue on the

subproblem. If the nonlinear problem has no feasible solution, SNOPT will tend to determine a "good" infeasible point if the elastic weight is sufficiently large. (If the elastic weight were infinite, SNOPT would locally minimize the nonlinear constraint violations subject to the linear constraints and bounds.)

Unfortunately, even though SNOPT locally minimizes the nonlinear constraint violations, there may still exist other regions in which the nonlinear constraints are satisfied. Wherever possible, nonlinear constraints should be defined in such a way that feasible points are known to exist when the constraints are linearized.

```
EXIT -- 20      The problem  appears  to  be unbounded
INFO -- 21      unbounded objective
INFO -- 22      constraint violation limit reached
```

For linear problems, unboundedness is detected by the simplex method when a nonbasic variable can be increased or decreased by an arbitrary amount without causing a basic variable to violate a bound. A message prior to the EXIT message will give the index of the nonbasic variable. Consider adding an upper or lower bound to the variable. Also, examine the constraints that have nonzeros in the associated column, to see if they have been formulated as intended.

Very rarely, the scaling of the problem could be so poor that numerical error will give an erroneous indication of unboundedness. Consider using the Scale option.

For nonlinear problems, SNOPT monitors both the size of the current objective function and the size of the change in the variables at each step. If either of these is very large (as judged by the Unbounded parameters-see #Description of the optional parameters), the problem is terminated and declared unbounded. To avoid large function values, it may be necessary to impose bounds on some of the variables in order to keep them away from singularities in the nonlinear functions.

The second informational message indicates an abnormal termination while enforcing the limit on the constraint violations. This exit implies that the objective is not bounded below in the feasible region defined by expanding the bounds by the value of the Violation limit.

```
EXIT -- 30   Resource limit  error
INFO -- 31   iteration limit reached
INFO -- 32   major  iteration limit reached
INFO -- 33   the  superbasics limit is too  small
```

Either the Iterations limit or the Major iterations limit was exceeded before the required solution could be found. Check the iteration log to be sure that progress was being made. If so, restart the run using *WarmDefSOL* at the end of the run.

If the superbasics limit is too small, then the problem appears to be more nonlinear than anticipated. The current set of basic and superbasic variables have been optimized as much as possible and a PRICE operation is necessary to continue, but there are already Superbasics limit superbasics (and no room for any more).

```
EXIT -- 40      Terminated  after  numerical difficulties
INFO -- 41      current point cannot  be improved
INFO -- 42      singular  basis
INFO -- 43      cannot  satisfy the  general  constraints
INFO -- 44      ill-conditioned null-space basis
```

Several circumstances may lead to SNOPT not being able to improve on a non-optimal point.

1. Subroutines could be returning accurate function values but inaccurate gradients (or vice versa). This is the most likely cause. Study the comments given for INFO 51 and 52, and do your best to ensure that the coding is correct.
2. The function and gradient values could be consistent, but their precision could be too low. For example, accidental use of a real data type when double precision was intended would lead to a relative function precision

of about 10-6 instead of something like 10^{-15}. The default Major optimality tolerance of 10-6 would need to be raised to about 10^{-3} for optimality to be declared (at a rather suboptimal point). Of course, it is better to revise the function coding to obtain as much precision as economically possible.

3. If function values are obtained from an expensive iterative process, they may be accurate to rather few significant figures, and gradients will probably not be available. One should specify

```
Function precision         t
Major optimality tolerance √t
```

but even then, if t is as large as 10^{-5} or 10^{-6} (only 5 or 6 significant figures), the same exit condition may occur. At present the only remedy is to increase the accuracy of the function calculation.

Termination because of a singular basis is highly unlikely to occur. The first factorization attempt will have found the basis to be structurally or numerically singular. (Some diagonals of the triangular matrix U were respectively zero or smaller than a certain tolerance.) The associated variables are replaced by slacks and the modified basis is refactorized, but singularity persists. This must mean that the problem is badly scaled, or the LU factor tolerance is too much larger than 1.0.

If the general constraints cannot be satisfied, an LU factorization of the basis has just been obtained and used to recompute the basic variables x_B , given the present values of the superbasic and nonbasic variables. A step of "iterative refinement" has also been applied to increase the accuracy of x_B . However, a row check has revealed that the resulting solution does not satisfy the current constraints $Ax - s = 0$ sufficiently well.

This probably means that the current basis is very ill-conditioned. If there are some linear constraints and variables, try Scale option 1 if scaling has not yet been used.

For certain highly structured basis matrices (notably those with band structure), a systematic growth may occur in the factor U . Consult the description of Umax, Umin and Growth in #Basis factorization statistics, and set the LU factor tolerance to 2.0 (or possibly even smaller, but not less than 1.0).

```
EXIT -- 50   Error in the  user-supplied functions
INFO -- 51   incorrect objective derivatives
INFO -- 52   incorrect constraint derivatives
```

This exit implies that there may be errors in the subroutines that define the problem objective and constraints. If the objective derivatives appear to incorrect, a check has been made on some individual elements of the objective gradient array at the first point that satisfies the linear constraints. At least one component (G(k) or gObj(j)) is being set to a value that disagrees markedly with its associated forward-difference estimate $\partial f_0/\partial x_j$. (The relative difference between the computed and estimated values is 1.0 or more.) This exit is a safeguard, since SNOPT will usually fail to make progress when the computed gradients are seriously inaccurate. In the process it may expend considerable effort before terminating with INFO 41 above.

Check the function and gradient computation *very carefully*. A simple omission (such as forgetting to divide f_0 by 2) could explain everything. If f_0 or a component $\partial f_0/\partial x_j$is very large, then give serious thought to scaling the function or the nonlinear variables.

If you feel *certain* that the computed gObj(j) is correct (and that the forward-difference estimate is therefore wrong), you can specify Verify level 0 to prevent individual elements from being checked. However, the opti- mization procedure may have difficulty.

If some constraint derivatives appear to be incorrect, then at least one of the computed constraint derivatives is significantly different from an estimate obtained by forward-differencing the vector F (x). Follow the advice given above for the objective function, trying to ensure that the arrays F and G are being set correctly.

```
EXIT -- 60      Undefined  user-supplied functions
INFO -- 61      undefined  function at  the  first feasible point
INFO -- 62      undefined  function at  the  initial point
INFO -- 63      unable  to  proceed  into undefined  region
EXIT -- 70      User requested  termination
INFO -- 71      terminated during function evaluation
INFO -- 72      terminated during constraint  evaluation
INFO -- 73      terminated during objective evaluation
INFO -- 74      terminated from  monitor  routine
```

These exits occur when Status < -1 is set during some call to the user-defined routines. SNOPT assumes that you want the problem to be abandoned forthwith.

If the following exits occur during the *first* basis factorization, the primal and dual variables x and pi will have their original input values.

```
EXIT -- 80      Insufficient storage  allocated
INFO -- 81      work arrays   must have at  least 500 elements
INFO -- 82      not  enough character storage
INFO -- 83      not  enough integer  storage
INFO -- 84      not  enough real  storage
```

SNOPT cannot start to solve a problem unless the char, int and real work arrays are at least 500 elements.

If the main character, integer or real storage arrays cw(*), iw(*) and rw(*) are not large enough for the current problem, the routine declaring cw(*), iw and rw should be recompiled with a larger dimensions for those arrays. The new values should also be assigned to lencw, leniw and lenrw. An estimate of the additional storage required is given in messages preceding the exit.

If rw(*) is not large enough, be sure that the Hessian dimension is not unreasonably large.

```
EXIT -- 90      Input arguments out  of  range
INFO -- 91      invalid input argument
EXIT -- 140      System error
INFO -- 141      wrong number of  basic  variables
```

Solution output

At the end of a run, the final solution is output to the PRINT file in accordance with the Solution keyword. Some header information appears first to identify the problem and the final state of the optimization procedure. A ROWS section and a COLUMNS section then follow, giving one line of information for each row and column. The format used is similar to certain commercial systems, though there is no industry standard.

An example of the printed solution is given in #File Output. In general, numerical values are output with format f16.5.

The maximum record length is 111 characters, including the first (carriage-control) character.

To reduce clutter, a dot "." is printed for any numerical value that is exactly zero. The values ±1 are also printed specially as 1.0 and -1.0. Infinite bounds ($\pm 10^{20}$ or larger) are printed as None.

Note : If two problems are the same except that one minimizes an objective $f_0(x)$ and the other maximizes $-f_0(x)$, their solutions will be the same but the signs of the dual variables π_i and the reduced gradients d_j will be reversed.

The ROWS section

General linear constraints take the form $l = Ax = u$. The ith constraint is therefore of the form

$$\alpha \leq a^T x \leq \beta,$$

and the value of $a^T x$ is called the *row activity*. Internally, the linear constraints take the form $Ax - s = 0$, where the slack variables s should satisfy the bounds $l \leq s \leq u$. For the ith "row", it is the slack variable s_i that is directly available, and it is sometimes convenient to refer to its state. Slacks may be basic or nonbasic (but not superbasic).

Nonlinear constraints $a = f_i(x) + a^T x <= \beta$ are treated similarly, except that the row activity and degree of infeasibility are computed directly from $f_i(x) + a^T x$ rather than s_i.

<table>
<tr><th>Label</th><th>Description</th></tr>
<tr><td>Number</td><td>The value $n + i$. This is the internal number used to refer to the ith slack in the iteration log.</td></tr>
<tr><td>Row</td><td>The name of the ith row.</td></tr>
<tr><td>State</td><td>The state of the ith row relative to the bounds α and β. The various states possible are as follows.
<table>
<tr><td>LL</td><td>The row is at its lower limit, a. UL The row is at its upper limit, β.</td></tr>
<tr><td>EQ</td><td>The limits are the same ($a = \beta$).</td></tr>
<tr><td>BS</td><td>The constraint is not binding. si is basic.</td></tr>
</table>
A key is sometimes printed before the State to give some additional information about the state of the slack variable.
<table>
<tr><td>A</td><td>Alternative optimum possible. The slack is nonbasic, but its reduced gradient is essentially zero. This means that if the slack were allowed to start moving from its current value, there would be no change in the objective function. The values of the basic and superbasic variables might change, giving a genuine alternative solution. The values of the dual variables might also change.</td></tr>
<tr><td>D</td><td>Degenerate. The slack is basic, but it is equal to (or very close to) one of its bounds.</td></tr>
<tr><td>I</td><td>Infeasible. The slack is basic and is currently violating one of its bounds by more than the Feasibility tolerance.</td></tr>
<tr><td>N</td><td>Not precisely optimal. The slack is nonbasic. Its reduced gradient is larger than the Major optimality tolerance.
Note: If Scale option > 0, the tests for assigning A, D, I, N are made on the scaled problem because the keys are then more likely to be meaningful.</td></tr>
</table>
</td></tr>
<tr><td>Activity</td><td>The row value $a^T x$ (or $f_i(x) + a^T x$ for nonlinear rows).</td></tr>
<tr><td>Slack activity</td><td>The amount by which the row differs from its nearest bound. (For free rows, it is taken to be minus the Activity.)</td></tr>
<tr><td>Lower limit</td><td>α, the lower bound on the row.</td></tr>
<tr><td>Upper limit</td><td>β, the upper bound on the row.</td></tr>
<tr><td>Dual activity</td><td>The value of the dual variable pi , often called the shadow price (or simplex multiplier) for the ith constraint. The full vector p always satisfies $B^T \pi = g_B$, where B is the current basis matrix and gB contains the associated gradients for the current objective function.</td></tr>
<tr><td>I</td><td>The constraint number, i.</td></tr>
</table>

The COLUMNS section

Here we talk about the "column variables" x_j, $j = 1 : n$. We assume that a typical variable has bounds $a = x_j = \beta$.

<table>
<tr><th>Label</th><th>Description</th></tr>
<tr><td>Number</td><td>The column number, j. This is the internal number used to refer to xj in the iteration log.</td></tr>
<tr><td>Column</td><td>The name of x_j.</td></tr>
<tr><td>State</td><td>The state of x_j relative to the bounds α and β. The various states possible are as follows.
<table>
<tr><td>LL</td><td>x_j is nonbasic at its lower limit, α.</td></tr>
<tr><td>UL</td><td>x_j is nonbasic at its upper limit, β.</td></tr>
<tr><td>EQ</td><td>x_j is nonbasic and fixed at the value $\alpha = \beta$.</td></tr>
<tr><td>FR</td><td>x_j is nonbasic at some value strictly between its bounds: $a < x_j < \beta$.</td></tr>
<tr><td>BS</td><td>x_j is basic. Usually $\alpha < x_j < \beta$.</td></tr>
<tr><td>SBS</td><td>x_j is superbasic. Usually $\alpha < x_j < \beta$.</td></tr>
</table>
A key is sometimes printed before the State to give some additional information about the state of x_j.
<table>
<tr><td>A</td><td>Alternative optimum possible. The variable is nonbasic, but its reduced gradient is essentially zero.
This means that if x_j were allowed to start moving from its current value, there would be no change in the objective function. The values of the basic and superbasic variables might change, giving a genuine alternative solution. The values of the dual variables might also change.</td></tr>
<tr><td>D</td><td>Degenerate. x_j is basic, but it is equal to (or very close to) one of its bounds.</td></tr>
<tr><td>I</td><td>Infeasible. x_j is basic and is currently violating one of its bounds by more than the Feasibility tolerance.</td></tr>
<tr><td>N</td><td>Not precisely optimal. x_j is nonbasic. Its reduced gradient is larger than the Major optimality tolerance .
Note: If Scale option > 0, the tests for assigning A, D, I, N are made on the scaled problem because the keys are then more likely to be meaningful.</td></tr>
</table>
</td></tr>
<tr><td>Activity</td><td>The value of the variable x_j.</td></tr>
<tr><td>g_j, the jth component of the gradient of the (linear or nonlinear) objective function. (If any x_j is infeasible, g_j is the gradient of the sum of infeasibilities.)</td><td></td></tr>
<tr><td>Lower limit</td><td>α, the lower bound on x_j.</td></tr>
<tr><td>Upper limit</td><td>β, the upper bound on x_j.</td></tr>
<tr><td>Reduced gradnt</td><td>The reduced gradient $d_j = g_j - p^T a_j$, where a_j is the jth column of the constraint matrix (or the jth column of the Jacobian at the start of the final major iteration).</td></tr>
<tr><td>M+J</td><td>The value $m + j$.</td></tr>
</table>

Obj Gradient

The SUMMARY file

If Summary file > 0, the following information is output to the SUMMARY file. (It is a brief form of the PRINT file.) All output lines are less than 72 characters.

- The Begin line from the SPECS file, if any.
- The basis file loaded, if any.
- A brief Major iteration log.
- A brief Minor iteration log.
- The EXIT condition and a summary of the final solution.

The following SUMMARY file is from the example, using Major print level 1 and Minor print level 0.

```
==============================
S N O P T  7.1-1(2) (Jun 2004)
==============================

SNSPEC EXIT 100 -- finished successfully
SNSPEC INFO 101 -- OPTIONS file read



Nonlinear constraints       2     Linear constraints       0
Nonlinear variables         2     Linear variables         0
Jacobian  variables         2     Objective variables      2
Total constraints           2     Total variables          2




This is problem  Toy1
The user has defined       6   out of       6   first  derivatives


Major Minors     Step   nCon Feasible  Optimal  MeritFunction    nS Penalty
    0      2              1  4.1E-01  5.0E-01  1.0000000E+00     2           r
    1      2  1.0E+00      2 (0.0E+00) 3.1E-01 -4.1421356E-01     1         n rl
    2      1  1.0E+00      3  1.4E+00  1.5E-01 -9.3802987E-01     1         s
    3      1  1.0E+00      4  1.8E-01  3.3E-02 -9.6547072E-01     1 2.8E-03
    4      1  1.0E+00      5  3.4E-02  8.9E-03 -9.9129962E-01       2.8E-03
    5      0  1.0E+00      6  4.2E-02  4.8E-03 -1.0000531E+00       2.8E-03
    6      0  1.0E+00      7  1.9E-04  2.0E-05 -9.9999997E-01       3.3E-02
    7      0  1.0E+00      8 (3.7E-09)(4.0E-10)-1.0000000E+00       3.3E-02

SNOPTA EXIT   0 -- finished successfully
SNOPTA INFO   1 -- optimality conditions satisfied

Problem name                 Toy1
No. of iterations                  7   Objective value     -1.0000000008E+00
```

```
No. of major iterations              7   Linear objective     0.0000000000E+00
Penalty parameter           3.253E-02   Nonlinear objective -1.0000000008E+00
No. of calls to funobj               9   No. of calls to funcon               9
No. of degenerate steps              0   Percentage                        0.00
Max x                        2 1.0E+00   Max pi                       1 1.2E-01
Max Primal infeas            0 0.0E+00   Max Dual infeas              2 8.0E-10
Nonlinear constraint violn     6.4E-09



Solution printed on file  15

Finished problem Toy1

Time for MPS input                            0.00 seconds
Time for solving problem                      0.00 seconds
Time for solution output                      0.00 seconds
Time for constraint functions                 0.00 seconds
Time for objective function                   0.00 seconds

snOptA finished.
INFO  = 1
nInf  = 0
sInf  =  0.
Obj   = -1.
```

Article Sources and Contributors

SNOPT *Source*: http://tomwiki.com/index.php?oldid=2377 *Contributors*: Elias

SNOPT Description of the SQP method *Source*: http://tomwiki.com/index.php?oldid=2378 *Contributors*: Elias

SNOPT Optional parameters *Source*: http://tomwiki.com/index.php?oldid=2380 *Contributors*: Elias

SNOPT File Output *Source*: http://tomwiki.com/index.php?oldid=2574 *Contributors*: Elias