SIF File Documentation

Prani Nalluri — SULI Intern

July 2021

1 Introduction

The SIF file format relies on constructed sets of elements and groups to create a function. These files are formatted with specific sections that each hold different types of information about the final function. Moreover, the specific row and column location of each piece of data in the file affects how the data relates to the final function.

2 Data Fields

Each line of a SIF file is divided into six data fields. Each field has a set number of characters as follows:

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
Characters 1-3	4-13	14-23	24-35	36-45	46-end

2.1 Exceptions

If a line begins with an asterisk $*$, the entire line is considered a comment for the user. None of the information in that line is necessary for determining the final function expression.

Similarly, if a $ character appears anywhere in a line, all characters after the $ are considered a comment and are not relevant to determining the final function.

3 Macroscopic Format

Each SIF file had three large sections - the Standard Data Input section, the Standard Element Type Input section, and the Standard Group Type Input section.

The Standard Data Input section contains information about how the "elements" and "groups" of a nonlinear program are related. The objective and each constraint of a nonlinear program will have multiple "groups" that are assigned to it. Each "group" will have multiple "elements", "parameters", and "constants" assigned

to it. Each "element" has various "internal variables", "element variables", and "parameters" assigned to it. Once all "elements", "parameters", and "constants" are assigned to "groups", all the "groups" assigned to a given constraint/objective are added together to create the constraint/objective function.

The Standard Element Type section provides information about the internal structure of each "element" defined in the Standard Data Input section. In the Standard Data Input section, various "internal variables", "element variables", and "parameters" are assigned to a given "element". The Standard Element Type section describes how these parts of an "element" are arithmetically related to one another.

The Standard Group Type section provides information about the internal structure of each "group" defined in the Standard Data Input section. In the Standard Data Input section, various "internal variables", "elements", "parameters", and "constants" are assigned to a given "group". The "elements" and "constants" are added and designated with a "group variable" name - the name itself should have been specified in the Standard Data Input section. The Standard Group Type section describes how the "group variable" and any "parameters" assigned to the group are arithmetically related.

4 Standard Data Input Format

4.1 Section Headers

The data is each SIF file is partitioned under different headers. Certain headers are mandatory, while others are optional. Some headers can also be synonymous with one another. Below, is compiled a list of all possible headers for the Standard Data Input section of the SIF file:

Section Header	**Synonyms**	**Required**
NAME	none	mandatory
GROUPS	ROWS, CONSTRAINTS	mandatory
VARIABLES	COLUMNS	mandatory
CONSTANTS	RHS, RHS'	optional
RANGES	none	optional
BOUNDS	none	optional
START POINT	none	optional
QUADRATIC	QUADS, QUADOBJ, QSECTION, HESSIAN	optional
ELEMENT TYPE	none	optional
ELEMENT USES	none	optional
GROUP TYPE	none	optional
GROUP USES	none	optional
ENDATA	none	mandatory

4.2 NAME Section

This section will have only one line, which reads "NAME prob-name", where prob-name is the name of the nonlinear program for which the SIF contains data. This also marks the beginning of the Standard Data Input section.

4.3 ENDATA Section

This section will have only one line, which reads "ENDATA". This line marks the end of the Standard Data Input section.

4.4 Integers and Real Parameters

Between the NAME and ENDATA sections, numerical values for integer and real parameters can be defined. (Note: These parameters are not necessarily the "parameters" assigned to groups and elements in the Standard Data Input section. The syntax for these assignments is different, and will be discussed later.)

Various commands in field 1 will determine the arithmetic that must be done to define certain variables. The table below compiles all these commands:

Command (Field 1)	**Arithmetic**
IE	Set field 2 = field 4
IR	Set field 2 = field 3
IA	Set field 2 = field 3+field 4
IS	Set field 2 = field 4-field 3
IM	Set field 2 = (field 3)*(field 4)
ID	Set field 2 = (field 4)/(field 3)
I=	Set field 2 = field 3
I+	Set field 2 = field 3+field 5
I-	Set field 2 = field 3-field 5
I*	Set field 2 = (field 3)*(field 5)
I/	Set field 2 = (field 3)/(field 5)
RE	Set field 2 = field 4
RI	Set field 2 = field 3
RA	Set field 2 = field 4 + field 3
RS	Set field 2 = field 4-field 3
RM	Set field 2 = (field 3)*(field 4)
RD	Set field 2 = (field 4)/(field 3)
RF	Set field 2 = evaluate function in field 3 at the value in field 4
R=	Set field 2 = field 3
R+	Set field 2 = field 3+field 5
R-	Set field 2 = field 3-field 5
R*	Set field 2 = (field 3)*(field 5)
R/	Set field 2 = (field 3)/(field 5)
R(	Set field 2 = evaluate function in field 3 at the value in field 5

Note that an array of parameters can also be defined at any point in the Standard Data Input section of a SIF file. These can occur within a do-loop (essentially a for-loop). Commands used when defining arrays of parameters are included below:

Command (Field 1)	Arithmetic
AE	Set field 2 = field 4
AI	Set field 2 = field 3
AA	Set field 2 = field 4 + field 3
AS	Set field 2 = field 4-field 3
AM	Set field 2 = (field 3)*(field 4)
AD	Set field 2 = (field 4)/(field 3)
AF	Set field 2 = evaluate function in field 3 at the value in field 4
A=	Set field 2 = field 3
A+	Set field 2 = field 3+field 5
A-	Set field 2 = field 3-field 5
A*	Set field 2 = (field 3)*(field 5)
A/	Set field 2 = (field 3)/(field 5)
A(	Set field 2 = evaluate function in field 3 at the value in field 5

4.5 Do-Loops

Do-loops are in essence for-loops. A do-loop can occur anywhere in a GROUPS, VARIABLES, RANGES, BOUNDS, START POINT, QUADRATIC, ELEMENT USES, or GROUP USES section. Do-loops have various commands that can be used to initialize, end, and nest them. These commands are compiled in a list below:

Command (Field 1)	Description
DO	Initializes do-loop to iterate through index (field 2) from start value (field 3) to end value (field 5)
DI	Set step size (field 3) for the do-loop iterating though the index in field 2
OD	End do-loop that is iterating on index in field 2
ND	End all do-loops that are currently iterating, regardless of index

4.6 GROUPS Section

The GROUPS section defines the groups that associated with the various constraints and objective function of the nonlinear program. Various commands can be used to define groups, define the objective or type of constraint they're associated with, assign variables to them, and scale the groups by coefficients. Below are the commands that can be used within a GROUPS section:

For N, E, XE, XN commands, if field 3 contains a variable, assign the variable in field 3 to the group in field 2. If field 4 is not empty, assign (value in field 4) $*$ (variable in field 3) to the group in field 2.

For ZE and ZN commands, if field 3 contains a variable, assign the variable in field 3 to the group in field 2. If field 5 is not empty, assign (value in field 5) $*$ (variable in field 3) to the group in field 2.

If field 3 contains the string "'SCALE'", for N, E, XE, XN commands, if field 3 contains the string "'SCALE'", multiply the group in field 2 by the inverse of the value in field 4. For ZN, ZE commands, multiply the group in field 2 by the inverse of the value in field 5.

Command (Field 1)	Description
N	Group (name in field 2) part of an objective function
E	Group (name in field 2) is a normal group – it is part of an objective function for the unconstrained case,
XN	an array of N-type groups are being defined
ZN	an array of N-type groups are being defined
XE	an array of E-type groups are being defined
ZE	an array of E-type groups are being defined
DE	linearly combine two groups to form one E-type group
DN	linearly combine two groups to form one N-type group

For DE, DN commands, the following formula determines the expression for the newly defined group:

$$\text{new group} = (\text{value in field 4}) * (\text{group in field 3}) + (\text{value in field 6}) * (\text{group in field 5})$$

4.7 VARIABLES Section

For lines in the VARIABLES section, the command in field 1 can be blank or read "X" or "Z". This section is meant to define variables for the nonlinear program. The data in field 3 of a particular line is what determines how a variable relates to the nonlinear program (NP).

For lines with an "X" or blank in field 1, if field 3 is:

- empty, the line is simply meant to define a variable for the NP
- is the name of a group, the variable in field 2 is assigned to the group in field 3. If field 4 is not empty, the variable in field 2 multiplied by the value in field 4 is assigned to the group in field 3.
- reads "'SCALE'", multiply the variable in field 2 by the inverse of the value in field 4 whenever the variable in field 2 is used in the NP
- reads "'INTEGER'", the variable in field 2 can only take on integer values
- reads "'ZERO-ONE'", the variable in field 2 can only take on zero or one as values

Note: If fields 5 and 6 are not empty, they are to be interpreted the same way as fields 3 and 4.

For lines with an "Z" in field 1, if field 3 is:

- empty, the line is simply meant to define a variable for the NP
- is the name of a group, the variable in field 2 is assigned to the group in field 3. If field 5 is not empty, the variable in field 2 multiplied by the value in field 5 is assigned to the group in field 3.
- reads "'SCALE'", multiply the variable in field 2 by the inverse of the value in field 5 whenever the variable in field 2 is used in the NP
- reads "'INTEGER'", the variable in field 2 can only take on integer values
- reads "'ZERO-ONE'", the variable in field 2 can only take on zero or one as values

4.8 CONSTANTS Section

In the CONSTANTS section, the command in field 1 can be blank, contain an "X", or contain a "Z". The data in field 3 is key to determine how a particular constant relates to the nonlinear program.

If field 3 contains the string "'DEFAULT'", then every group is assigned a constant that is the opposite sign of the constant defined in field 4 (if field 1 contains an "X") or field 5 (if field 1 contains an "Z"). If we have a group that is associated with a constraint, rather than an objective function, we make the value in field 4 or 5 the right-hand side of the constraint, rather than assigning a constant.

If field 3 contains the name of a group, then the group is assigned a constant that is the opposite sign of the constant defined in field 4 (if field 1 contains an "X") or field 5 (if field 1 contains an "Z"). If we have a group that is associated with a constraint, rather than an objective function, we make the value in field 4 or 5 the right-hand side of the constraint, rather than assigning a constant.

Note that in both of the above cases, if field 1 contains an "X" and fields 5 and 6 are not empty, fields 5 and 6 operate the same way fields 3 and 4 would.

4.9 QUADRATIC Section

For any line in the QUADRATIC section, field 1 can be blank or contain an "X" or "Z". Any line in this section signifies that an element of the following form should be added to the objective function. Note that the "coefficient" referenced below defaults to one, unless otherwise specified.

$$\frac{1}{2}(coefficient)(variable\,1)(variable\,2)$$

If field 1 is blank or contains an "X", fields 2 and 3 contain the variables involved in creating this element. If field 4 it not empty, it will contain the coefficient by which the element must be multiplied. If field 5 is not empty, then another element with variables in fields 2 and 5 can be added onto the objective. If field 6 is not empty, it contains the coefficient for this second element.

If field 1 contains a "Z", then the variables are located in fields 2 and 3. If field 5 is not empty, it contains the coefficient for the element.

4.10 ELEMENT TYPE Section

The ELEMENT TYPE section defines the components necessary to construct each type of element in the nonlinear program. Three types of components can be built: element parameters, internal variables, and element variables. These components will be symbolically assigned to element types in this section. In the ELEMENT USES section, elements will be assigned an element type from the ELEMENT TYPE section, and will be given actual values and variables for the symbolic components that correspond to that element type.

An element parameter is a constant value that is arithmetically included in the formula for the element to which it is assigned. An element variable is a symbolic variable that is a placeholder for an actual variable from the VARIABLES section. An internal variable is a component that is equivalent to a linear combination

of element variables. An element parameter, element variable, and internal variable can be assigned with the respective commands EP, EV, and IV in field 1.

The EV command assigns the element variable in field 3 to the element type in field 2. If field 5 is not empty, it is also assigned to the element type in field 2 as an element variable. Similarly, the IV command assigns internal variables in fields 3 (and potentially 5) to the element type in field 2. The EP command assigns element parameters in fields 3 (and potentially 5) to the element type in field 2.

4.11 ELEMENT USES Section

The ELEMENT USES section defines elements that will be used in various groups in the nonlinear program. In this section, each element will be given a name, type, and actual values for its element parameters, element variables, and internal variables.

Various commands in field 1 can be used to fully define these elements. A list of the commands is compiled below:

Command (Field 1)	**Description**
T	element (field 2) is assigned type (field 3)
XT	element (field 2) is assigned type (field 3) (note: element is indexed out of an array)
V	element (field 2) has element variable (field 3) set equal to variable from VARIABLES section (field 5)
ZV	element (field 2) has element variable (field 3) set equal to variable from VARIABLES section (field 5) (note: element or variable is indexed out of an array)
P	element (field 2) has element parameter (field 3) set equal to an actual value (field 4) (note: field 5,6 can operate like field 3,4 if not empty)
XP	element (field 2) has element parameter (field 3) set equal to an actual value (field 4) (note: field 5,6 can operate like field 3,4 if not empty) (note: element or parameter is indexed out of an array)
ZP	element (field 2) has element parameter (field 3) set equal to an actual value (field 5) (note: element or parameter is indexed out of an array)

As an exception, a line with "T" or "XT" in the first field contains the string "'DEFAULT'" in field 2, every element defined in the ELEMENT USES section automatically has the type referenced in field 3. For an element to not have this default type, another line with "T" or "XT" in field 1 must set the given element to a different element type after the "default element" line.

4.12 GROUP TYPE Section

The GROUP TYPE section sets up the various group types that will be used through the nonlinear program. Two types of components can be built: group parameters and group variables. These components will be symbolically assigned to group types in this section. In the GROUP USES section, groups will be assigned a group type from the GROUP TYPE section, and will be given actual values and expressions for the symbolic

components that correspond to that group type. Note that a group variable traditionally is the sum of all the elements and constants assigned to a group.

The two commands in field 1 that are specific to the GROUP TYPE section are GV and GP. GV indicates that the group type assigned in field 2 has a group variable specified in field 3. GP indicates that the group type in field 2 has the group parameters assigned in field 3 (and possibly field 5, if not empty).

4.13 GROUP USES Section

The GROUP USES section defines elements that will be used in the nonlinear program. In this section, each element will be given a name, type, values for group parameters, and assigned elements.

Various commands in field 1 will give ensure these assignments are made. These commands are compiled below:

Command (Field 1)	**Description**
T	group (field 2) is assigned type (field 3)
XT	group (field 2) is assigned type (field 3) (note: group is indexed out of an array)
E	group (field 2) contains element (field 3) w/ coefficient of 1 (or of field 4, if not empty) (note: field 5,6 can operate like field 3,4 if not empty)
XE	group (field 2) contains element (field 3) w/ coefficient of 1 (or of field 4, if not empty) (note: field 5,6 can operate like field 3,4 if not empty) (note: element or group is indexed out of an array)
ZE	group (field 2) contains element (field 3) w/ coefficient of 1 (or of field 5, if not empty) (note: element or group is indexed out of an array)
P	group (field 2) has group parameter (field 3) set equal to an actual value (field 4) (note: field 5,6 can operate like field 3,4 if not empty)
XP	group (field 2) has group parameter (field 3) set equal to an actual value (field 4) (note: field 5,6 can operate like field 3,4 if not empty) (note: group or parameter is indexed out of an array)
ZP	group (field 2) has group parameter (field 3) set equal to an actual value (field 5) (note: group or parameter is indexed out of an array)

5 Standard Element Type Format

The Standard Element Type format is used to define formulas for each element type declared ELEMENT TYPE subsection of the Standard Input Data section. Note that if no element types were declared, then the entire Standard Element Type section can be omitted.

5.1 Data Fields

Each line within the Standard Element Type section is divided into either four or six data fields. If a line has six data fields, the fields are split up in the same manner as in the Standard Input Data section. If a line does not have six data fields, fields 4-6 are merged into a single field, referenced as field 7. Each field has a set number of characters as follows:

Field 1	Field 2	Field 3	Field 7
Characters 1-3	4-13	14-23	24-end

Note that the only case in which a line in the Standard Element Type section would not have a 4-field structure is when the command in field 1 is an "R".

5.2 Section Headers

The Standard Element Type format has various possible subsections. A list is compiled below:

Section Header	**Required**
ELEMENTS	mandatory (if there are element types declared in the ELEMENT TYPE subsection of the Standard Input Data section)
TEMPORARIES	optional
GLOBALS	optional
INDIVIDUALS	mandatory (if there are element types declared in the ELEMENT TYPE subsection of the Standard Input Data section)
ENDATA	mandatory (if there are element types declared in the ELEMENT TYPE subsection of the Standard Input Data section)

5.3 ELEMENTS Section

Indicates beginning of Standard Element Type section.

5.4 TEMPORARIES Section

The TEMPORARIES section contains a set of parameters that will be used in the GLOBALS and INDIVIDUALS sections to define formulas for an element, its gradient, and its Hessian. These parameters are strings that will be set equal to numerical expressions in the GLOBALS and INDIVIDUALS sections. Note that parameters in the Standard Element Type can be numerical values of function expressions (e.g. sin(3)).

For each line in this section, field 1 specifies the type of parameter to be defined, while field 2 gives the parameter a name. A list of possible types of parameters are included below:
Note that an intrinsic function is built into Fortran (e.g. sin(x), log(3)), while and external function is not.

Command (Field 1)	Description
I	integer type parameter
R	real type parameter
L	logical type parameter
M	intrinsic function parameter
F	external function parameter

When porting test functions in SIF files to other languages, one must be mindful of the translation between in-built Fortran functions and in-built functions in other languages.

5.5 GLOBALS Section

The GLOBALS section also announces parameters that are to be used later in the INDIVIDUALS section. However, the numerical expressions or values these parameters are set equal to are listed in the GLOBALS section itself. Various commands in field 1 determine the assignment of expressions and values to these parameters. They are compiled below:

Command (Field 1)	Description
A	parameter (field 2) equals expression (field 7)
A+	description (field 7) adds onto description in line above
I	parameter (field 2) equals expression (field 7) if the string in field 3 has value true (more information below)
I+	description (field 7) adds onto description in line above
E	parameter (field 2) equals expression (field 7) if the string in field 3 has value false (more information below)
E+	description (field 7) adds onto description in line above

For lines that begin with either "I" or "E" in field 1, whether the parameter in field 2 takes on the value in field 7 is dependent on whether the string in field 3 has a value of true or false, respectively. This string in field 3 will have been previously defined either in the GLOBALS or INDIVIDUALS section, on a line with the command "A" in field 1. The string will have been named in field 2, with a statement in field 7 that can either be true or false for a given element.

5.6 INDIVIDUALS Section

The INDIVIDUALS section describes the formulas for each element, its gradient, and its Hessian. There are various commands in field one that accomplish these tasks. A list is compiled below:
To elaborate, if the command in field 1 of a line is "R", then the internal variable in field 2 is determined as follows:

$$(coefficient\,1)(variable\,1) + (coefficient\,2)(variable\,2)$$

Coefficients 1,2 default to one if they are not defined in fields 4 and 6, respectively. Variables 1,2 default to zero, unless they are defined in fields 3 and 5, respectively.

Command (Field 1)	Description
T	indicates that element type in field 2 will be defined below
R	internal variable (field 2) is defined as a linear combination of coefficients and variables in fields 3-6 (this is further described below)
A	parameter (field 2) equals expression (field 7)
A+	description (field 7) adds onto description in line above
I	parameter (field 2) equals expression (field 7) if the string in field 3 has value true (more information below)
I+	description (field 7) adds onto description in line above
E	parameter (field 2) equals expression (field 7) if the string in field 3 has value false (more information below)
E+	description (field 7) adds onto description in line above
F	field 7 is the function expression for the element
F+	description (field 7) adds onto description in line above
G	field 7 is the gradient of the element with respect to the variable in field 2
G+	description (field 7) adds onto description in line above
H	field 7 is a nonzero component of the Hessian matrix for the nonlinear element with respect to the variables in fields 2,3
H+	description (field 7) adds onto description in line above

For lines that begin with either "I" or "E" in field 1, whether the parameter in field 2 takes on the value in field 7 is dependent on whether the string in field 3 has a value of true or false, respectively. This string in field 3 will have been previously defined either in the GLOBALS or INDIVIDUALS section, on a line with the command "A" in field 1. The string will have been named in field 2, with a statement in field 7 that can either be true or false for a given element.

Note that for each element type elaborated upon in the Standard Element Type section, the following four field 1 commands must appear: "T", "F", "G", "H". The "T" and "F" commands can only appear once per element type, while the "G" and "H" commands can appear multiple times. Each time a new element type is elaborated upon, the a line beginning with "T" in field 1 and the element type in field 2 indicates the beginning of the description for that element type. When the next line beginning with "T" occurs, the description of the previous element type is considered complete.

5.7 ENDATA Section

The ENDATA section signifies the end of the Standard Element Type section.

6 Standard Group Type Format

The Standard Group Type format is used to define formulas for each group type declared GROUP TYPE subsection of the Standard Input Data section. Note that if no group types were declared, then the entire Standard Group Type section can be omitted.

6.1 Data Fields

Each line within the Standard Group Type section is divided into either four or six data fields. A line is split into four or six fields in the same manner as in the Standard Element Type section. The exceptions when a line is split six fields, rather than only four, are the same as in the Standard Element Type section.

6.2 Section Headers

The Standard Group Type format has various possible subsections. A list is compiled below:

Section Header	**Required**
GROUPS	mandatory
TEMPORARIES	optional
GLOBALS	optional
INDIVIDUALS	mandatory (if there are group types declared in the GROUP TYPE subsection of the Standard Input Data section)
ENDATA	mandatory (if there are group types declared in the GROUP TYPE subsection of the Standard Input Data section)

6.3 GROUPS Section

Indicates beginning of Standard Group Type section.

6.4 TEMPORARIES Section

The TEMPORARIES section is structured just as it is in the Standard Element Type section, and follows the same set of rules.

6.5 GLOBALS Section

The GLOBALS section is structured just as it is in the Standard Element Type section, and follows the same set of rules.

6.6 INDIVIDUALS Section

The INDIVIDUALS section describes the formulas for each group and its first and second derivative. There are various commands in field one that accomplish these tasks. A list is compiled below:

Command (Field 1)	Description
T	indicates that group type in field 2 will be defined below
A	parameter (field 2) equals expression (field 7)
A+	description (field 7) adds onto description in line above
I	parameter (field 2) equals expression (field 7) if the string in field 3 has value true (more information below)
I+	description (field 7) adds onto description in line above
E	parameter (field 2) equals expression (field 7) if the string in field 3 has value false (more information below)
E+	description (field 7) adds onto description in line above
F	field 7 is the function expression for the group
F+	description (field 7) adds onto description in line above
G	field 7 is the first derivative of the group with respect to the group variable
G+	description (field 7) adds onto description in line above
H	field 7 is the first derivative of the group with respect to the group variable
H+	description (field 7) adds onto description in line above

For lines that begin with either "I" or "E" in field 1, whether the parameter in field 2 takes on the value in field 7 is dependent on whether the string in field 3 has a value of true or false, respectively. This string in field 3 will have been previously defined either in the GLOBALS or INDIVIDUALS section, on a line with the command "A" in field 1. The string will have been named in field 2, with a statement in field 7 that can either be true or false for a given element.

Note that for each group type elaborated upon in the Standard Group Type section, the following four field 1 commands must appear: "T", "F", "G", "H". The "T" and "F" commands can only appear once per group type, while the "G" and "H" commands can appear multiple times. Each time a new group type is elaborated upon, the a line beginning with "T" in field 1 and the group type in field 2 indicates the beginning of the description for that group type. When the next line beginning with "T" occurs, the description of the previous group type is considered complete.

6.7 ENDATA Section

The ENDATA section signifies the end of the Standard Group Type section.

7 An Example: GENROSE.SIF

To demonstrate an application of the above process, we will decode the GENROSE.SIF file. The entire file is included below.

```
1
2  **************************
```

```
* SET UP THE INITIAL DATA *
**************************

NAME          GENROSE

*    Problem :
*    ---------

*    The generalized Rosenbrock function.

*    Source: problem 5 in
*    S. Nash,
*    "Newton-type minimization via the Lanczos process",
*    SIAM J. Num. Anal. 21, 1984, 770-788.

*    SIF input: Nick Gould, Oct 1992.
*               minor correction by Ph. Shott, Jan 1995.

*    classification SUR2-AN-V-0

*    Number of variables

 IE N                     5              $-PARAMETER
*IE N                     10             $-PARAMETER
*IE N                     100            $-PARAMETER
*IE N                     500            $-PARAMETER

*    other parameter definitions

 IE 1                     1
 IE 2                     2
 IA N-1         N         -1
 IA N+1         N         1
 RI RN+1        N+1

VARIABLES

 DO I           1                        N
 X  X(I)
 ND

GROUPS

 N  OBJ

 DO I           2                        N
 XN Q(I)        'SCALE'   0.01
 XN Q(I)        X(I)      1.0
 XN L(I)        X(I)      1.0
 ND

CONSTANTS

    GENROSE     OBJ       -1.0

 DO I           2                        N
 X  GENROSE     L(I)      1.0
 ND

BOUNDS

 FR GENROSE     'DEFAULT'

START POINT

*  start with X(I) = I/N+1.

 DO I           1                        N

 RI RI          I
 R/ T           RI                       RN+1
 ZV GENROSE     X(I)                     T

 ND

ELEMENT TYPE

 EV MSQR        V

ELEMENT USES

```

```
 XT  'DEFAULT'  MSQR

 DO  I          2                          N
 IA  I-1        I           -1
 ZV  Q(I)       V                          X(I-1)
 ND

GROUP TYPE

 GV  L2         GVAR

GROUP USES

 XT  'DEFAULT'  L2

 DO  I          2                          N
 XE  Q(I)       Q(I)
 ND

OBJECT BOUND

 LO  GENROSE                 1.0

*    Solution

*LO  SOLTN                   1.0

ENDATA

***********************
* SET UP THE FUNCTION *
* AND RANGE ROUTINES  *
***********************

ELEMENTS       GENROSE

INDIVIDUALS

 T   MSQR
 F                           - V ** 2
 G   V                       - 2.0D+0 * V
 H   V          V            - 2.0D+0


ENDATA

**********************
* SET UP THE GROUPS  *
* ROUTINE            *
**********************

GROUPS         GENROSE

INDIVIDUALS

 T   L2
 F                           GVAR * GVAR
 G                           GVAR + GVAR
 H                           2.0

ENDATA
```

We will now deconstruct the file, section by section. The first section we will work with is the section above the VARIABLES section. Many of the parameters for the file are defined here.

```

***************************
* SET UP THE INITIAL DATA *
***************************

NAME           GENROSE

*    Problem :
*    ---------

*    The generalized Rosenbrock function.

*    Source: problem 5 in
*    S. Nash,
```

```
15  *    "Newton-type minimization via the Lanczos process",
16  *    SIAM J. Num. Anal. 21, 1984, 770-788.
17
18  *    SIF input: Nick Gould, Oct 1992.
19  *               minor correction by Ph. Shott, Jan 1995.
20
21  *    classification SUR2-AN-V-0
22
23  *    Number of variables
24
25   IE N                     5              $-PARAMETER
26  *IE N                     10             $-PARAMETER
27  *IE N                     100            $-PARAMETER
28  *IE N                     500            $-PARAMETER
29
30  *    other parameter definitions
31
32   IE 1                     1
33   IE 2                     2
34   IA N-1         N         -1
35   IA N+1         N         1
36   RI RN+1        N+1
```

Note that every line beginning with a '*' is a comment. Our first uncommented line is line 6, the NAME section. Here we learn that the name of our function is GENROSE. The next uncommented line is line 25, which contains the "IE" command in field 1. This command indicates that the parameter "N" is equal to 5. The section of line 25 that occurs after the $ is considered a comment.

Our next uncommented lines are line 32 and 33, which contain the command "IE" in field 1. These lines indicate that the values in field 2 equal those in field 4, meaning "1"=1 and "2"=2. Lines 34 and 35 contain the command "IA" in field 1, meaning "N-1"="N"+"-1" and "N+1"="N"+"1". Line 36 contains "RI" in field 1, indicating that "RN+1"="N+1".

The next section we will include is the VARIABLES section.

```
1
2   VARIABLES
3
4    DO I           1                        N
5    X  X(I)
6    ND
```

The VARIABLES tag in line 2 indicates the VARIABLES section is beginning. The "DO" command in line 4 indicates that a for-loop is beginning, and will iterate from I=1 to I=N. Line 5 indicates that elements of an array of variables are being created as we iterate through the do-loop. For each I, a variable X(I) is created. The "ND" in line 6 indicates that the do-loop ends at this line, once the iterating is complete.

The next section we will include is the GROUPS section.

```
1   GROUPS
2
3    N   OBJ
4
5    DO I           2                        N
6    XN Q(I)        'SCALE'    0.01
7    XN Q(I)        X(I)       1.0
8    XN L(I)        X(I)       1.0
9    ND
```

The GROUPS tag in line 1 indicates the GROUPS section is beginning. The "N" in field 1 of line 3 indicates a group related to the objective function is being created. The "OBJ" in field 2 of line 3 is the name of the objective function group. In line 5, the "DO" indicates a do-loops is being created. This loop iterates from I=2 to I=N. The "XN" in line 6 indicates that an array of objective function groups is being created. For each I, a group Q(I) is created. The "'SCALE'" in field 3 of line 6 indicates each group Q(I) will be scaled by the inverse of the value in field 4, resulting in a scale factor of 100. Line 7 indicates that for each I, the group Q(I) will be assigned variable X(I) with coefficient 1.0. Line 8 contains the command "XN",

indicating for each I, a group L(I) is created. Each group L(I) will contain a variable X(I) with coefficient 1.0. Line 9 contains "ND", indicating the of the do-loop. As a rule, all the groups used later in the file are defined in the GROUPS section.

The next section we will include is the CONSTANTS section.

```
1  CONSTANTS
2
3      GENROSE   OBJ          -1.0
4
5   DO I         2                        N
6   X  GENROSE   L(I)         1.0
7   ND
```

The CONSTANTS tag in line 1 indicates the CONSTANTS section is beginning. Line 3 contains and empty field 1, but references a group name in field 3. This indicates that the group OBJ will be assigned a constant that is the opposite of the value in field 4. Thus, the OBJ group has a constant -1.0. The string "GENROSE" in field 2 of line 3, simply means that the name for the constant is "GENROSE". Field 2 could contain any string, but the name in the string will not affect the final function in any way. Line 5 contains "DO", indicating the beginning of a do-loop from I=2 to I=N. Line 6 contains an "X", indicating that an array is being indexed through. For each I, the group L(I) is assigned a constant -1.0. Once again, the string in field 2 does not affect the final function at all.

The next sections we will include are the ELEMENT TYPE and ELEMENT USES sections.

```
1  ELEMENT TYPE
2
3   EV MSQR        V
4
5  ELEMENT USES
6
7   XT 'DEFAULT' MSQR
8
9   DO I           2                        N
10  IA I-1         I          -1
11  ZV Q(I)        V                        X(I-1)
12  ND
```

The ELEMENT TYPE tag in line 1 indicates the ELEMENT TYPE section is beginning. "EV" in line 3 indicates an element variable is being assigned to an element type. Field 2 contains the element type in question, MSQR, and field 3 contains the element variable, V, being assigned to the element type.

The ELEMENT USES tag in line 5 indicates the ELEMENT USES section is beginning. The "XT" command in line 7 indicates an array of elements are having their type defined. The "'DEFAULT'" string in field 2 indicates the default type for all elements declared in this section will be of type MSQR (field 3). For an element to have a different element type, another "XT" or "T" command would have to redefine its type later in the ELEMENT USES section. Line 9 contains "DO", indicating the start of a do-loop from I=2 to I=N. Line 10 indicates "I-1"="I"+"-1". "ZV" in line 11 indicates that an array is being indexed out of. For every I, the group Q(I) has the variable X(I-1) assigned for the element variable V. The "ND" in line 12 indicates the termination of the do-loop.

The next sections we will include are the GROUP TYPE and GROUP USES sections.

```
1  GROUP TYPE
2
3   GV L2          GVAR
4
5  GROUP USES
6
7   XT 'DEFAULT' L2
8
9   DO I           2                        N
10  XE Q(I)        Q(I)
11  ND
```

The GROUP TYPE tag in line 1 indicates the GROUP TYPE section is beginning. The "GV" command in line 3 indicates that a group variable is being assigned to a particular group type. Group type L2 is being assigned group variable GVAR.

The GROUP USES tag in line 5 indicates the GROUP USES section is beginning. The "XT" command in line 7 indicates an array of groups are having their type defined. The "'DEFAULT'" string in field 2 indicates the default type for all groups declared in this section will be of type L2 (field 3). For an element to have a different element type, another "XT" or "T" command would have to redefine its type later in the GROUP USES section. Line 9 contains "DO", indicating the start of a do-loop from I=2 to I=N. The "XE" in line 10 indicates that an array fo elements is being indexed out of. For each I, the group Q(I) is assigned element Q(I). Note that the group Q(I) and element Q(I) are distinct. An element being assigned to a group need not necessarily have the same name. The "ND" in line 11 marks the end of the do-loop.

The next section we will include is the ENDATA section.

```
1  ENDATA
```

The ENDATA tag indicates the end of the Standard Data Input section.

The next section we will include is the Standard Element Type section.

```
1  ************************
2  * SET UP THE FUNCTION  *
3  * AND RANGE ROUTINES   *
4  ************************
5
6  ELEMENTS        GENROSE
7
8  INDIVIDUALS
9
10  T  MSQR
11  F                          -  V ** 2
12  G  V                       -  2.0D+0 * V
13  H  V            V          -  2.0D+0
14
15
16 ENDATA
```

Note that all lines beginning with a '*' are comments. The ELEMENTS tag in line 6 indicates the Standard Element Type section has begun. The second string in line 6 has no bearing on the value of the final function. Note that no TEMPORARIES or GLOBALS section is defined for this SIF file - this is acceptable, since these sections are optional.

The INDIVIDUALS tag in line 8 indicates the INDIVIDUALS section of the Standard Element Type section has begun. The "T" command in line 10 indicates that we are now elaborating upon element type MSQR (field 2). The "F" command in line 11 indicates that we are defining the formula for the element type MSQR in field 7. This formula reads "$-V^2$". Note that these formulas are always written in Fortran syntax, meaning that certain sequences of characters, such as "**" or "2.0D+0", may have arithmetic meanings that are not intuitive. For example, "**" indicates an exponent, and "2.0D+0" is a form of scientific notification equalling "$2.0 * (10^0) = 2$".

The command "G" in line 12 gives the gradient of the element MSQR with respect to element variable V as "-2V". The command "H" in line 13 gives the nonzero component of the Hessian matrix for MSQR with respect to element variables V and V as "-2". The ENDATA tag in line 16 concludes the Standard Element Type section.

The next section we will include is the Standard Group Type section.

```
1  ********************
2  * SET UP THE GROUPS *
3  * ROUTINE           *
4  ********************
```

```
5
6  GROUPS            GENROSE
7
8  INDIVIDUALS
9
10   T   L2
11   F                          GVAR * GVAR
12   G                          GVAR + GVAR
13   H                          2.0
14
15 ENDATA
```

Note that all lines beginning with a '*' are comments. The GROUPS tag in line 6 indicates the Standard Element Type section has begun. The second string in line 6 has no bearing on the value of the final function. Note that no TEMPORARIES or GLOBALS section is defined for this SIF file - this is acceptable, since these sections are optional.

The INDIVIDUALS tag in line 8 indicates the INDIVIDUALS section of the Standard Group Type section has begun. The "T" command in line 10 indicates that we are now elaborating upon group type L2 (field 2). The "F" command in line 11 indicates that we are defining the formula for the group type L2 in field 7. This formula reads "GVAR*GVAR". Note that these formulas are always written in Fortran syntax, meaning that certain sequences of characters may have arithmetic meanings that are not intuitive. The command "G" in line 12 gives the first derivative of the group type L2 with respect to group variable gvar as "GVAR+GVAR". The command "H" in line 13 gives the second derivative for the group type L2 with respect to group variable GVAR as "2.0". The ENDATA tag in line 15 concludes the Standard Element Type section.

To construct our final final function, note that our groups and elements have been assigned components as indicated below:

Group Name	**Q(I) – I=2:N**	**L(I) – I=2:N**	**OBJ**
Group Coefficients	100		
Group Type	L2	L2	L2
Group Variable	GVAR	GVAR	GVAR
Group Type Formula	GVAR*GVAR	GVAR*GVAR	GVAR*GVAR
Constants		-1	-1
Elements	Q(I)		
Linear Variables	X(I)	X(I)	

Element Name	**Q(I) – I=2:N**
Element Type	MSQR
Element Variables	V = X(I-1)
Element Type Formula	$-V^2$
Internal Variables	
Element Parameters	

By convention, we use element type formulas to construct elements. Note that all our elements are of type MSQR. Thus for I=2 to I=N, each element Q(I) = -X(I-1)^2.

By convention, the group variable for a particular group is defined as the sum of all the constants and elements assigned to the group. For I=2 to I=N, each group Q(I) has GVAR = Q(I) + X(I), where Q(I) is an element; thus, each group Q(I) has GVAR = -X(I-1)^2 + X(I). For I=2 to I=N, each group L(I) has GVAR = -1 + X(I). The group OBJ has GVAR = -1.

Next, the group type formula must be applied to each group. Note that all our groups are of type L2. For I=2 to I=N, each group Q(I) = $(-X(I-1)^2+X(I))^2$. For I=2 to I=N, each group L(I) = $(-1+X(I))^2$. The final OBJ group is OBJ = $(-1)^2$.

Since all our groups were defined on lines with "N" or "-N" in the field 1 command, they are all N-type groups. Thus, all our groups are meant for the objective function. Therefore, we will add up all of our groups to find the final objective function. The final objective function reads as follows:

$$objective = 1 + \sum_{i=2}^{N} \left(100(-x_{i-1}^2 + x_i)^2 + (x_i - 1)^2\right)$$