SENSITIVITY ANALYSIS WITH MINOS

A User's Guide

Stanford Business Software, Inc.

SENSITIVITY ANALYSIS WITH MINOS

A User's Guide

SENSITIVITY ANALYSIS WITH MINOS
A User's Guide

First Edition

Stanford Business Software, Inc.
2680 Bayshore Parkway, Suite 304
Mountain View, CA 94043

SENSITIVITY ANALYSIS WITH MINOS

This manual was typeset using LaTeX macros with Knuth's TeX typesetting system.

Publisher: Stanford Business Software, Inc.
2680 Bayshore Parkway, Suite 304
Mountain View, CA 94043

Contents

Chapter 1

INTRODUCTION

Costs and resource availabilities are often uncertain quantities in many applications. Thus, it is important to know the effect of changing a cost, a right-hand side, or even a constraint matrix coefficient on the solution to a linear program:

- How much can one decrease a cost and have a particular solution remain optimal?
- By how much can one change a right-hand side and have a particular basis remain feasible?

The value of the solution to a linear program is limited if the robustness of that solution is unknown.

SAMINOS, the Sensitivity Analysis package for use with MINOS (Version 5.4 or later), makes it easy for the MINOS user to create sensitivity analysis reports. SAMINOS allows the MINOS user to generate reports on costs, right-hand sides, and even constraint matrix coefficients without putting much more effort into input than would be required with MINOS. By default, when the sensitivity analysis option is invoked, ranges are provided on all costs, all right hand sides, and all nonzero elements of the coefficient matrix. However, the knowledgeable user can easily modify this as described in Appendix A.

It is assumed that you are already familiar with MINOS. We recommend that you have access to a copy of the MINOS User's Guide as you read this manual.

Chapter 2

SENSITIVITY ANALYSIS WITH SAMINOS

SAMINOS is a set of routines which, together with MINOS, allow you to obtain upper and lower bounds on the costs, right-hand sides, and matrix coefficients associated with a linear program.

In this chapter, we describe how to set up the SAMINOS executable program and the SAMINOS input files in order to perform sensitivity analysis on a linear program.

2.1 SETTING UP SAMINOS

REQUIREMENTS

To set up SAMINOS, you will need:

- a copy of the MINOS source code (Version 5.4 or later),
- a Fortran 77 compiler,
- the SAMINOS diskette, and,
- a computer with sufficient memory (see the MINOS User's Manual for details on memory requirements).

Note: If you decide to set up SAMINOS on a PC, we recommend that you use a Fortran 77 compiler that supports extended memory.

CREATING THE SAMINOS EXECUTABLE PROGRAM

In this section we describe the steps to create the SAMINOS executable program on your PC assuming that you have available the Lahey Fortran 77 F77L-EM/32 Version 5.01 compiler.[1]

1. Transfer all files from the SAMINOS diskette to the the directory where your MINOS files reside (we assume the directory is named MINOS).

   ```
   C:MINOS\> copy a:*.*
   ```

2. Because of the way MINOS is set up, you will have to make a small modification to the file MINOSL.FOR. Comment out the line

   ```
   entry matmod
   ```

   which is towards the end of MINOSL.FOR. Remember to save the file with this change. This modification is necessary because the sensitivity analysis routines are called through subroutine MATMOD, which resides in the file SAFUNS.FOR.

3. The batch file SAPCCOM.BAT is used to compile the source code for SAMINOS and link the resulting .OBJ files to create an executable SAMINOS.EXE. The batch file is invoked by typing

   ```
   C:\MINOS> sapccom
   ```

4. Two simple example problems are included on your SAMINOS diskette; EX1DIET, which is the diet problem T1DIET described in the MINOS User's Guide, and the problem EX2FURN, a furniture production problem taken from the literature. Each example uses two files:

   - a file with extension SPC, and
   - a file with extension MPS.

   To try SAMINOS on one of these problems, copy files as follows.

   ```
   C:\MINOS> copy EX1DIET.* MINOS.*
   ```

   or

[1] Should you decide to use a different compiler (or a different platform) please refer to your compiler manual and modify the files SAPCCOM.BAT and SAPCLP.LNK appropriately.

```
C:\MINOS> copy EX2FURN.* MINOS.*
```

5. Execute SAMINOS by issuing the following command.

```
C:\MINOS> SAMINOS
```

   If you try problem EX1DIET, you should get an objective function value of 92.5. If you try problem EX2FURN, you should get an objective function value of -18.67.

NOTES:

- The batch file SAPCCOM.BAT uses the default compiler options for the Lahey F77L-EM/32 Version 5.01 compiler. This results in the creation of files with a .SLD extension which are used by Lahey's symbolic online debugger. Add the /NS option to the batch file to suppress generation of these files.
- Lahey Fortran allows you to reduce the executable size of SAMINOS by putting the workspace array Z in a blank common block; the compiler will create a smaller executable that allocates memory for Z during execution rather. If you wish to make this change, you will need to modify the file MINOSL.FOR.
- If you plan to use a compiler other than Lahey F77L-EM/32, check to see if it supports extended memory. If your compiler supports extended memory, you should be to able compile and link by modifying the batch file SAPCCOM.BAT and link file SAPCLP.LNK. If your compiler does not support extended memory, you will need to study the source files and create overlays. The creation of overlays is done during link time for most compilers.

2.2 SAMINOS INPUT FILES

The PC version of SAMINOS uses the following files for input.

MINOS.SPC	SPECIFICATIONS File
MINOS.MPS	MPS File

Note that the required input files are exactly the same ones that you would use to solve a linear program with MINOS. However, on systems other than a PC, MINOS file names may automatically be supplied by the system; please refer to the MINOS 5.4 User's Guide and/or the files with extension .DOC supplied on your MINOS 5.4 diskette.

Because SAMINOS adds on to MINOS 5.4, SAMINOS can also read from or write to all the other files that MINOS 5.4 can access. If you want to use the information in these files, you may need to work with the specification file. See the MINOS 5.4 User's Guide for a description of these supplementary files.

SPECIFYING MINOS PARAMETERS FOR SAMINOS

The MINOS specifications file should be set up as if you were solving an ordinary linear program, with two important exceptions.

- *CYCLE LIMIT.* The number of linear programs solved by MINOS is controlled by the CYCLE LIMIT parameter found in the MINOS specifications file. At least one linear program must be solved if the sensitivity analysis routines are to be called. For this reason, include the line

  `CYCLE LIMIT` 2

- *WORKSPACE (USER).* You will need to reserve a portion of the MINOS workspace for use by the sensitivity analysis routines. Include the line

  `WORKSPACE (USER)` *maxw*

  where *maxw* is the number of spaces in the workspace array that you will allocate to the arrays needed for sensitivity analysis. You will need to set

  $$maxw \geq \frac{1}{2}(\max\ \{n, m, l\} + 1) + 2(n + m + l),$$

  where n is the number of variables, m is the number of constraints, and l is the number of matrix coefficient elements that you specify as input to the problem (as specified in the MINOS Users' Guide, this can be greater than the number of nonzeros). Typically, l will equal the number of non-zero elements in the constraint matrix.

An example of a specifications file is shown in Figure 2.1. This is the specifications file for the EX1DIET example included in the SAMINOS diskette.

SPECIFYING LINEAR PROBLEM DATA (MPS FILE)

As required by MINOS, all variables, and the constraints and bounds associated with the variables, are specified in the MPS data file. There are two exceptions you will need to keep in mind, however.

1. *Zero-valued Coefficient Matrix Element* If you would like the sensitivity analysis to report on a zero-valued element of the constraint matrix, be sure to list this element explicitly with a zero value in the COLUMNS section of your MPS file.

```
Begin ex1diet   (Diet LP problem)
   Minimize
   Rows                    20
   Columns                 30
   Elements               100
   Cycle Limit              2  * Set cycle limit to at least 2
   MPS file                10

   Iterations             100
   Print level              1  * OK for small problems
   Print frequency          1
   Summary frequency        1
   Workspace (user)      2500  * Allocate space for sensitivity analysis
End Diet Problem
```

Figure 2.1: A Specifications File

2. *Cannot Handle Range Constraints.* This version of SAMINOS is not equipped to handle ranged constraints, i.e., constraints of the form $b_i \leq a_i^T x \leq b_i + r$, where $r > 0$. You will need to break any such constraint up into two constraints, or SAMINOS will not report on this constraint. However, no actual error condition will result.

An example of an MPS file is shown in Figure 2.2. This is the MPS file for the EX1DIET problem included in the SAMINOS diskette.

2.3 SAMINOS OUTPUT

The solution to the linear program is written to the PRINT file used by MINOS 5.4, if a solution has been found; otherwise, the appropriate error messages are written to the PRINT file. SAMINOS generates three additional output files:

SAMINOS.COS	Cost Range File
SAMINOS.RHS	Right-Hand Side Range File
SAMINOS.MAT	Constraint Matrix Coefficient Range File

Each of the files is described in the following sub-sections.

```
NAME          EX1DIET
ROWS
 G  ENERGY
 G  PROTEIN
 G  CALCIUM
 N  COST
COLUMNS
    OATMEAL   ENERGY      110.0        PROTEIN       4.0
    OATMEAL   CALCIUM       2.0        COST          3.0
    CHICKEN   ENERGY      205.0        PROTEIN      32.0
    CHICKEN   CALCIUM      12.0        COST         24.0
    EGGS      ENERGY      160.0        PROTEIN      13.0
    EGGS      CALCIUM      54.0        COST         13.0
    MILK      ENERGY      160.0        PROTEIN       8.0
    MILK      CALCIUM     285.0        COST          9.0
    PIE       ENERGY      420.0        PROTEIN       4.0
    PIE       CALCIUM      22.0        COST         20.0
    PORKBEAN  ENERGY      260.0        PROTEIN      14.0
    PORKBEAN  CALCIUM      80.0        COST         19.0
RHS
    DEMANDS   ENERGY     2000.0        PROTEIN      55.0
    DEMANDS   CALCIUM     800.0
BOUNDS
 UP SERVINGS  OATMEAL       4.0
 UP SERVINGS  CHICKEN       3.0
 UP SERVINGS  EGGS          2.0
 UP SERVINGS  MILK          8.0
 UP SERVINGS  PIE           2.0
 UP SERVINGS  PORKBEAN      2.0
ENDATA
```

Figure 2.2: An MPS File

COST RANGE ANALYSIS

INDEX	NAME	STATUS	LOWER LIMIT	UPPER LIMIT
1	OATMEAL	UL	-INF	0.6187500E+01
2	CHICKEN	LL	0.1153125E+02	INF
3	EGGS	LL	0.9000000E+01	INF
4	MILK	BS	0.7619048E+01	0.1169231E+02
5	PIE	UL	-INF	0.2362500E+02
6	PORKBEAN	LL	0.1462500E+02	INF

Figure 2.3: SAMINOS.COS: A Cost Analysis File

2.3.1 COST RANGING REPORT (SAMINOS.COS)

SAMINOS.COS is a report of the sensitivity analysis results on the linear program objective cost coefficients. For each variable (not including slack variables implicit to the program), the report specifies a range over which each cost coefficient may be varied without changing the optimality of the current LP solution (provided all the other coefficients at their original values). Note that although the LP basic solution does not change if a cost coefficient is individually varied over the specified range, the objective function value itself will almost certainly change. It is straightforward to compute the new objective value; it can also be obtained by re-running MINOS.

Figure 2.3 displays the SAMINOS.COS file generated for the EX1DIET example provided with the SAMINOS diskette.

2.3.2 RIGHT HAND SIDE RANGING REPORT (SAMINOS.RHS)

SAMINOS.RHS is a report of the sensitivity analysis results on the linear program right-hand sides. For each constraint (not including the cost constraint, or any ranged constraints), the report specifies the range over which the right-hand size of that constraint may be varied without changing the feasibility of the current optimal LP basis. Note that although the basis does not change with a change in one of the right-hand sides, the values of the basic variables will almost certainly change, and hence so will the value of the objective function. To obtain the new values of the optimal basic feasible solution you can either re-run MINOS or SAMINOS with the new value of the right-hand side. If only the new objective value is desired, you can compute it using information from the current run by noting that the change to the optimal objective value is given by as a result of changing

RHS ANALYSIS

INDEX	NAME	STATUS	LOWER LIMIT	UPPER LIMIT
1	ENERGY	ACTIVE	0.1900000E+04	0.2560000E+04
2	PROTEIN	LOOSE	-INF	0.6000000E+02
3	CALCIUM	LOOSE	-INF	0.1334500E+04

Figure 2.4: SAMINOS.RHS: A Right-Hand Side Analysis File

the right hand side b_i is given by $\pi_i \Delta b_i$, where π_i is the simplex multiplier on constraint i and Δb_i is the change in value of the right hand side.

Figure 2.4 displays the SAMINOS.RHS file generated for the EX1DIET example provided with the SAMINOS diskette.

2.3.3 COEFFICIENT MATRIX RANGING REPORT (SAMINOS.MAT)

SAMINOS.MAT is a report of the sensitivity analysis results on the coefficient matrix elements of the linear program. Ranges on all matrix elements specified in the MPS input file will be reported, unless that element belongs to a ranged constraint. For each matrix coefficient element, SAMINOS.MAT lists a range (and occasionally, two ranges) of values that the matrix element may assume without affecting the feasibility or optimality of the current optimal basis. Note that although the basis does not change when we change a matrix element within a reported range, the values of the basic variables may change and the value of the objective function will almost definitely change. To obtain the values of the new optimal basic solution, to re-run MINOS (or SAMINOS) with the new value of the matrix element.

Figure 2.5 displays the SAMINOS.MAT file generated for the EX1DIET example provided with the SAMINOS diskette.

MATRIX RANGING ANALYSIS

ROW INDEX	ROW NAME	COL INDEX	COL NAME	LOWER LIMIT	UPPER LIMIT
1	ENERGY	1	OATMEAL	0.5333333E+02	0.1350000E+03
2	PROTEIN	1	OATMEAL	0.2750000E+01	INF
3	CALCIUM	1	OATMEAL	-0.1316250E+03	INF
1	ENERGY	2	CHICKEN	-INF	0.4266667E+03
2	PROTEIN	2	CHICKEN	-INF	INF
3	CALCIUM	2	CHICKEN	-INF	INF
1	ENERGY	3	EGGS	-INF	0.2311111E+03
2	PROTEIN	3	EGGS	-INF	INF
3	CALCIUM	3	EGGS	-INF	INF
1	ENERGY	4	MILK	0.1231579E+03	0.1858065E+03
2	PROTEIN	4	MILK	0.6888889E+01	INF
3	CALCIUM	4	MILK	0.1662222E+03	INF
1	ENERGY	5	PIE	0.3555556E+03	0.4700000E+03
2	PROTEIN	5	PIE	0.1500000E+01	INF
3	CALCIUM	5	PIE	-0.2452500E+03	INF
1	ENERGY	6	PORKBEAN	-INF	0.3377778E+03
2	PROTEIN	6	PORKBEAN	-INF	INF
3	CALCIUM	6	PORKBEAN	-INF	INF

Figure 2.5: SAMINOS.MAT: A Matrix Coefficient Analysis File

Chapter 3

SENSITIVITY ANALYSIS FORMULAS

In this chapter, we describe the sensitivity analysis formulas used by SAMINOS, and describe how to interpret the results. We will develop the formulas under the assumption that the problem at hand is

$$\begin{array}{lrcl} \text{Minimize} & c^T x & = & z \\ \text{Subject to} & Ax & = & b \\ & l \leq x \leq u & & \end{array}$$

Formulas will also be displayed for maximization problems; i.e., for those problems that maximize $c^T x$.

Throughout this chapter, we will use the problem EX1DIET as an example. The problem EX1DIET is Chvátal's diet problem as described in the MINOS User's Guide. The problem is expressed as follows.

$$\begin{array}{l} \text{Minimize} \\ \quad 3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6 = z \\ \text{Subject to} \\ 110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 - x_7 = 2000 \\ \quad 4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 - x_8 = 55 \\ \quad 2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 - x_9 = 800 \\ x_1 \leq 4,\ x_2 \leq 3,\ x_3 \leq 2,\ x_4 \leq 8,\ x_5 \leq 2,\ x_6 \leq 2 \\ x_i \geq 0,\ i = 1, \ldots, 9 \end{array}$$

The solution to this problem is

$$x^* = \begin{pmatrix} x_1^* \\ x_2^* \\ x_3^* \\ x_4^* \\ x_5^* \\ x_6^* \\ x_7^* \\ x_8^* \\ x_9^* \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \\ 0 \\ 4.5 \\ 2 \\ 0 \\ 0 \\ 5 \\ 534.5 \end{pmatrix}, \qquad z^* = 92.5.$$

The optimal optimal basis corresponds to the variables (x_4, x_8, x_9). The optimal dual variables for the three constraints are

$$\pi^* = \begin{pmatrix} \pi_1^* \\ \pi_2^* \\ \pi_3^* \end{pmatrix} = \begin{pmatrix} 9/160 \\ 0 \\ 0 \end{pmatrix}.$$

3.1 COST RANGING

Costs are often uncertain quantities in many applications. Thus, it is important to know the effect of a cost fluctuation on the optimality of an LP solution; note that regardless of how the cost fluctuates the feasibility of the linear program is not affected. The term "cost ranging" refers to the determination of an upper and lower bound on the cost c_j of a variable x_j such that the solution to the LP remains the same as long as no other parameters of the LP change, and as long as the cost remains within those bounds. There are two cases to consider: the effect of changing the cost of one of the nonbasic variables and the effect of changing the cost for one of the basic variables.

3.1.1 NONBASIC VARIABLES

Minimization Problem. Consider a minimization problem with an optimal solution x^* and an optimal dual solution π^*. We examine the nonbasic variable x_j with cost c_j.

For optimality to hold, the reduced cost $\bar{c}_j$ associated with x_j must be non-negative if x_j is at its lower bound l_j and non-positive if x_j is at its upper bound u_j:

$$\begin{aligned} \bar{c}_j &= c_j - A_{\bullet j}^T \pi^* \geq 0 \quad \text{if } x_j^* = l_j \\ \bar{c}_j &= c_j - A_{\bullet j}^T \pi^* \leq 0 \quad \text{if } x_j^* = u_j \end{aligned}$$

Thus, the range of costs c_j for the nonbasic variable x_j over which optimality of the basis (and hence the solution) is preserved is

$$c_j \geq A_{\bullet j}^T \pi^* \quad \text{if } x_j^* = l_j \tag{3.1}$$
$$c_j \leq A_{\bullet j}^T \pi^* \quad \text{if } x_j^* = u_j \tag{3.2}$$

For example, consider our sample problem EX1DIET and the variable x_1 with cost $c_1 = 3$. In the solution to this problem, x_1 is non-basic and is at its upper bound 4. It does not pay to reduce the value of x_1 as long as its reduced cost $\bar{c}_1 \leq 0$, or, from (3.2), as long c_1 satisfies:

$$c_1 \leq A_{\bullet 1}^T \pi^* = \begin{pmatrix} 110 & 4 & 2 \end{pmatrix} \begin{pmatrix} 0.05625 \\ 0 \\ 0 \end{pmatrix} = 6.1875.$$

If c_1 were to exceed 6.1875, then the Simplex Algorithm would attempt to lower the value of the variable x_1.

Maximization Problem. If the problem at hand is a maximization problem, then the optimality conditions on the reduced costs are reversed: the reduced cost $\bar{c}_j$ associated with x_j must be non-positive if x_j is at its lower bound l_j and non-negative if x_j is at its upper bound u_j. Hence, the optimality range for c_j would be

$$c_j \leq A_{\bullet j}^T \pi^* \quad \text{if } x_j^* = l_j \tag{3.3}$$
$$c_j \geq A_{\bullet j}^T \pi^* \quad \text{if } x_j^* = u_j \tag{3.4}$$

3.1.2 BASIC VARIABLES

Minimization Problem. Consider a minimization problem with an optimal solution x^* and an optimal dual solution π^*. We examine the kth basic variable x_{j_k} with cost c_{j_k}.

What is the range of costs over which c_{j_k} may vary without affecting the optimality of the current basis (and hence, the current solution)? These limits will depend on the effect of changing c_{j_k} on the reduced costs of the non-basic variables. For the current solution to remain optimal, the reduced cost $\bar{c}_i$ associated with non-basic variable x_i must be non-negative if x_i is at its lower bound and non-positive if x_i is at its upper bound.

Hence, the optimality range for c_{j_k} is

$$\left[c_{j_k} + \max_{i \in I} \frac{\bar{c}_i}{\bar{a}_{ki}}, c_{j_k} + \min_{i \in J} \frac{\bar{c}_i}{\bar{a}_{ki}} \right] \tag{3.5}$$

where

$$\begin{array}{rcl} I & = & \{i : x_i \text{ nonbasic}, \bar{a}_{ki} < 0, x_i^* = l_i \text{ or } \bar{a}_{ki} > 0, x_i^* = u_i\}, \\ J & = & \{i : x_i \text{ nonbasic}, \bar{a}_{ki} < 0, x_i^* = u_i \text{ or } \bar{a}_{ki} > 0, x_i^* = l_i\}, \\ \bar{a}_{ki} & = & \text{element } (k,i) \text{ of } B^{-1}A, \\ B^{-1}A & = & \text{constraint matrix expressed in terms of the optimal basis} \end{array}$$

As an example, consider the sample problem EX1DIET. The basic variable x_4 has an associated cost of $c_4 = 9$; by how much can we vary c_4 without affecting the optimality of the current solution? Because x_4 is the first basic variable, we will need to check the reduced costs $\bar{c}$ and the first row of $B^{-1}A$:

$$\bar{c} = \begin{pmatrix} -51/16 \\ 399/32 \\ 4 \\ 0 \\ -29/8 \\ 35/8 \\ 9/160 \\ 0 \\ 0 \end{pmatrix} \qquad B^{-1}A_{1\bullet} = \begin{pmatrix} 11/16 \\ 41/32 \\ 1 \\ 1 \\ 21/8 \\ 13/8 \\ -1/160 \\ 0 \\ 0 \end{pmatrix}.$$

Hence, from (3.5)

$$\begin{array}{rcl} c_4 & \geq & 9 + \max\ (\bar{c}_1/\bar{a}_{11}, \bar{c}_4/\bar{a}_{15}, \bar{c}_7/\bar{a}_{17}) \\ & = & 9 + \max\ (-51/11, -29/21, 9/-1) \\ & = & 160/21 \\ & \approx & 7.619 \end{array}$$

and

$$\begin{array}{rcl} c_4 & \leq & 9 + \min\ (\bar{c}_2/\bar{a}_{12}, \bar{c}_3/\bar{a}_{13}, \bar{c}_6/\bar{a}_{16}) \\ & = & 9 + \min\ (399/41, 4/1, 35/13) \\ & = & 152/13 \\ & \approx & 11.6923 \end{array}$$

Thus c_4 may take on any value between 7.619 and 11.6923 without affecting the optimality of the current solution.

The full cost analysis report for EX1DIET produced by SAMINOS is displayed in Figure 3.1. Because x_7, x_8, and x_9 are slack variables, they were not included in the the MPS input file. Hence, the cost ranges on the costs for variables x_7, x_8, and x_9 are not included in the cost report.

COST RANGE ANALYSIS

INDEX	NAME	STATUS	LOWER LIMIT	UPPER LIMIT
1	OATMEAL	UL	-INF	0.6187500E+01
2	CHICKEN	LL	0.1153125E+02	INF
3	EGGS	LL	0.9000000E+01	INF
4	MILK	BS	0.7619048E+01	0.1169231E+02
5	PIE	UL	-INF	0.2362500E+02
6	PORKBEAN	LL	0.1462500E+02	INF

Figure 3.1: Cost Range Analysis for EX1DIET

Maximization Problem. If the problem at hand is a maximization problem, the optimality conditions for reduced costs are reversed. In this case, the upper and lower bounds of optimality on c_{j_k} are

$$\left[c_{j_k} + \max_{i \in J} \frac{\bar{c}_i}{\bar{a}_{ki}}, c_{j_k} + \min_{i \in I} \frac{\bar{c}_i}{\bar{a}_{ki}}\right] \tag{3.6}$$

3.2 RIGHT HAND SIDE RANGING

The right-hand sides of LP constraints often represent resource availabilities. Since resource availabilities are frequently subject to adjustment, it is important to examine the effect of such changes on the LP solution. Although the values of the optimal basic variables of the LP will almost certainly change with a change in a right-hand side constraint, the optimal basis need not change. The problem at hand is to determine upper and lower bounds on a right-hand side b_r so that as long as no other LP parameters change, and as long as b_r remains within those bounds, the optimal basis does not change. Note that if a change in the right-hand side does not make the basis infeasible, the basis will remain optimal.

Consider a minimization problem with an optimal solution x^*, and suppose that we wish to replace the right-hand side of the rth constraint, b_r, with $b_r + \beta$; that is, we replace b with $b + \beta e_r$. The range of values β for which the current basis remains feasible and thus optimal is obtained by ensuring that $l_i \leq x_i \leq u_i$, if x_i is basic. If we let x_{B_i} denote the ith basic variable, we obtain the following bounds on $b_r(\beta) = b_r + \beta$:

$$b_r(\beta) \geq \quad b_r + \quad \max\left[\max_{i:B^{-1}_{ir}<0}\left(\frac{u_{B_i} - x^*_{B_i}}{B^{-1}_{ir}}\right), \max_{i:B^{-1}_{ir}>0}\left(\frac{l_{B_i} - x^*_{B_i}}{B^{-1}_{ir}}\right)\right] \tag{3.7}$$

RHS ANALYSIS

INDEX	NAME	STATUS	LOWER LIMIT	UPPER LIMIT
1	ENERGY	ACTIVE	0.1900000E+04	0.2560000E+04
2	PROTEIN	LOOSE	-INF	0.6000000E+02
3	CALCIUM	LOOSE	-INF	0.1334500E+04

Figure 3.2: Right-Hand Side Analysis for EX1DIET

$$b_r(\beta) \leq \quad b_r + \quad \min\left[\min_{i:B^{-1}_{ir}<0}\left(\frac{l_{B_i}-x^*_{B_i}}{B^{-1}_{ir}}\right), \min_{i:B^{-1}_{ir}>0}\left(\frac{u_{B_i}-x^*_{B_i}}{B^{-1}_{ir}}\right)\right] \tag{3.8}$$

As an example, consider the sample problem EX1DIET and the first right-hand side $b_1 = 2000$. We would like to determine the range of values that may b_1 assume without any change to the optimal basis which corresponds to the variables (x_4, x_8, x_9). Note that

$$B^{-1}_{\bullet 1} = \begin{pmatrix} 1/160 \\ 1/20 \\ 57/32 \end{pmatrix}, \qquad \begin{pmatrix} u_4 \\ u_8 \\ u_9 \end{pmatrix} = \begin{pmatrix} 8 \\ \infty \\ \infty \end{pmatrix}, \qquad \begin{pmatrix} l_4 \\ l_8 \\ l_9 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}.$$

Hence, from (3.7)

$$\begin{aligned} b_1 &\geq 2000 + \max\left((l_4 - x^*_4)/B^{-1}_{11}, (l_8 - x^*_8)/B^{-1}_{21}, (l_9 - x^*_9)/B^{-1}_{31}\right) \\ &= 2000 + \max\left(-720, -100, -17104/57\right) \\ &= 1900 \end{aligned}$$

and, from (3.8)

$$\begin{aligned} b_1 &\leq 2000 + \min\left((u_4 - x^*_4)/B^{-1}_{11}, (u_8 - x^*_8)/B^{-1}_{21}, (u_9 - x^*_9)/B^{-1}_{31}\right) \\ &= 2000 + \min\left(560, \infty, \infty\right) \\ &= 2560 \end{aligned}$$

That is, as long as $1900 \leq b_1 \leq 2560$, the current optimal basis stays feasible and optimal.

The full right-hand side analysis report for EX1DIET produced by SAMINOS is displayed in Figure 3.2.

3.3 COEFFICIENT MATRIX RANGING

Note that changing an element of the coefficient matrix will affect both feasibility and optimality of the current basis. Although the formulas are somewhat complex, they are somewhat easier to understand if it is understood that:

- Perturbing an element a_{kj} is equivalent to perturbing the variable cost c_j by a multiple of the optimal dual price π_k;
- Perturbing an element a_{kj} is equivalent to perturbing the right-hand side b_k by a multiple of the optimal x_j.

We will need to consider two cases: the effect of changes in a coefficient in a nonbasic column and the effect of changes in a coefficient in a basic column.

3.3.1 NONBASIC COLUMNS

Changing the coefficients associated with nonbasic columns can affect both the feasibility and optimality of the current basis, unless we are faced with the special case of having all nonbasic variables set to 0. The problem at hand is to determine upper and lower bounds on coefficient a_{kj}, where x_j is non-basic, so that as long as a_{kj} remains within those bounds, the current basis remains feasible and optimal.

OPTIMALITY BOUNDS

Suppose a_{kj} is replaced by $a_{kj}(\alpha) = a_{kj} + \alpha$. We would like to determine what upper and lower bounds we can place on α so that the optimality of the current basis is not affected. Note that the reduced cost $\bar{c}_j$ associated with x_j can be expressed as a function of α:

$$\bar{c}_j(\alpha) = c_j - (A_{\bullet j} + \alpha e_k)^T \pi^* = c_j - A_{\bullet j}^T \pi^* - \alpha e_k^T \pi^* = c_j - \alpha \pi_k^* - A_{\bullet j}^T \pi^* \tag{3.9}$$

This relation suggests that replacing a_{kj} with $a_{kj}(\alpha) = \alpha$ is equivalent to replacing c_j with $c_j - \alpha \pi_k^*$.

Minimization Problem. In Section 3.1.1, we derived upper and lower bounds on c_j. We now use this result to derive bounds on $a_{kj}(\alpha) = a_{kj} + \alpha$ as shown below.

$$a_{kj}(\alpha) \geq a_{kj} + \frac{\bar{c}_j}{\pi_k^*} \text{ if } \pi_k^* < 0 \text{ and } x_j^* = l_j \text{ or } \pi_k^* > 0 \text{ and } x_j^* = u_j \tag{3.10}$$

$$a_{kj}(\alpha) \leq a_{kj} + \frac{\bar{c}_j}{\pi_k^*} \text{ if } \pi_k^* < 0 \text{ and } x_j^* = u_j \text{ or } \pi_k^* > 0 \text{ and } x_j^* = l_j \tag{3.11}$$

As an example, consider the sample problem EX1DIET and the element a_{11}. Since x_1 is a non-basic variable at its upper bound, and since $\pi_1^* > 0$,

$$a_{11} \geq 110 + (51/16)/(-9/160) = 160/3 = 53.333 \tag{3.12}$$

Maximization Problem. If the problem at hand is a maximization problem, the upper and lower bounds on c_j are reversed. Hence, the upper and lower bounds on a_{kj} are also reversed:

$$a_{kj}(\alpha) \leq a_{kj} + \frac{\bar{c}_j}{\pi_k^*} \text{ if } \pi_k^* < 0 \text{ and } x_j^* = l_j \text{ or } \pi_k^* > 0 \text{ and } x_j^* = u_j \tag{3.13}$$

$$a_{kj}(\alpha) \geq a_{kj} + \frac{\bar{c}_j}{\pi_k^*} \text{ if } \pi_k^* < 0 \text{ and } x_j^* = u_j \text{ or } \pi_k^* > 0 \text{ and } x_j^* = l_j \tag{3.14}$$

FEASIBILITY BOUNDS

Note that a change in a non-basic column of A can affect the feasibility of the current basis, unless the associated non-basic variable is set to a bound of 0. We would like to determine what upper and lower bounds we can place on a fluctuation α in a_{kj} so that the feasibility of the current basis is not affected. If the current basis is to remain feasible, then $l_i \leq x_i \leq u_i$ must hold for each basic variable x_i. Note that

$$\begin{aligned} x_B(\alpha) &= B^{-1}b - B^{-1}(Nx_N - \alpha e_k x_j) \\ &= B^{-1}b - B^{-1}Nx_N - \alpha B^{-1}e_k x_j \\ &= B^{-1}(b - \alpha x_j e_k) - Nx_N \end{aligned} \tag{3.15}$$

The above relation suggests that replacing a_{kj} with $a_{kj}(\alpha) = a_{kj} + \alpha$ is equivalent to decreasing the right-hand side b_k by αx_j. In Section 3.2, we derived upper and lower bounds on b_k; we use these bounds to derive feasibility bounds on $a_{kj}(\alpha)$.

$$a_{kj}(\alpha) \geq a_{kj} + \max\left[\max_{i:B_{ir}^{-1}>0}\left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right), \max_{i:B_{ir}^{-1}<0}\left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right)\right] \text{ if } x_i^* > 0 \tag{3.16}$$

$$a_{kj}(\alpha) \geq a_{kj} + \max\left[\max_{i:B_{ir}^{-1}>0}\left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right), \max_{i:B_{ir}^{-1}<0}\left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right)\right] \text{ if } x_i^* < 0 \tag{3.17}$$

$$a_{kj}(\alpha) \leq a_{kj} + \min\left[\min_{i:B_{ir}^{-1}>0}\left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right), \min_{i:B_{ir}^{-1}<0}\left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right)\right] \text{ if } x_i^* > 0 \tag{3.18}$$

$$a_{kj}(\alpha) \leq a_{kj} + \min\left[\min_{i:B_{ir}^{-1}>0}\left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right), \min_{i:B_{ir}^{-1}<0}\left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}}\right)\right] \text{ if } x_i^* < 0 \tag{3.19}$$

As an example, consider the sample problem EX1DIET and the matrix element a_{11}. Since x_1 is non-basic and positive-valued, and we computed upper and lower bounds on b_1 as being 1900 and 2560 in Section 3.2, it is a simple matter to compute feasibility bounds for a_{11}:

$$\begin{aligned} a_{11} &\geq 110 + (2560 - 2000)/-4 \\ &= -30 \\ a_{11} &\leq 110 + (1900 - 2000)/-4 \\ &= 135 \end{aligned}$$

For the basis to remain feasible, $-30 \leq a_{11} \leq 135$ must hold. Since, from (3.12), we must have $a_{11} \geq 53.33$ for the basis to remain optimal, a_{11} must lie between 53.33 and 135 for the current basis to remain feasible and optimal.

3.3.2 BASIC COLUMNS

Derivation of feasibility and optimality bounds for a_{kj} when x_j is basic is somewhat more complicated than when x_j is non-basic. In fact, it turns out that the changes in x and $\bar{c}$ are *non-linear* in a change α in a_{kj}! What we do instead is derive upper and lower bounds on a variable $\phi(\alpha)$, where

$$\phi(\alpha) = \frac{1}{B^{-1}_{i_j k} + 1/\alpha}$$

and x_j is the i_jth basic variable (for details, see Dantzig & Thapa [1997]). Since ϕ increases when α increases, we can use upper and lower bounds on ϕ to derive upper and lower bounds on α.

As it turns out, adding α to a_{kj} is equivalent to

- Decreasing c_j by $\pi^*_k \phi(\alpha)$; and
- Decreasing b_k by $x^*_j \phi(\alpha)$

OPTIMALITY BOUNDS

The results from Section 3.1.2 provide optimality bounds for ϕ.

$$\phi(\alpha) \geq \max_{l \in I} \frac{\bar{c}_l}{-\pi^*_k \bar{a}_{i_j l}} \text{ if } \pi^*_k < 0 \tag{3.20}$$

$$\phi(\alpha) \geq \max_{l \in J} \frac{\bar{c}_l}{-\pi^*_k \bar{a}_{i_j l}} \text{ if } \pi^*_k > 0 \tag{3.21}$$

$$\phi(\alpha) \leq \min_{l \in J} \frac{\bar{c}_l}{-\pi^*_k \bar{a}_{i_j l}} \text{ if } \pi^*_k < 0 \tag{3.22}$$

$$\phi(\alpha) \leq \min_{l \in I} \frac{\bar{c}_l}{-\pi_k^* \bar{a}_{ijl}} \text{ if } \pi_k^* > 0 \tag{3.23}$$

where

$$I = \left\{ l : x_l \text{ nonbasic}, \bar{a}_{ijl} < 0, x_l^* = l_l \text{ or } \bar{a}_{ijl} > 0, x_l^* = u_l \right\},$$
$$J = \left\{ l : x_l \text{ nonbasic}, \bar{a}_{ijl} < 0, x_l^* = u_l \text{ or } \bar{a}_{ijl} > 0, x_l^* = l_l \right\}$$

FEASIBILITY BOUNDS

The results from Section 3.2 provide feasibility bounds for ϕ.

$$\phi(\alpha) \geq \max \left[\max_{i:B_{ir}^{-1}>0} \left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right), \max_{i:B_{ir}^{-1}<0} \left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right) \right] \text{ if } x_i^* > 0 \tag{3.24}$$

$$\phi(\alpha) \geq \max \left[\max_{i:B_{ir}^{-1}>0} \left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right), \max_{i:B_{ir}^{-1}<0} \left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right) \right] \text{ if } x_i^* < 0 \tag{3.25}$$

$$\phi(\alpha) \leq \min \left[\min_{i:B_{ir}^{-1}>0} \left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right), \min_{i:B_{ir}^{-1}<0} \left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right) \right] \text{ if } x_i^* > 0 \tag{3.26}$$

$$\phi(\alpha) \leq \min \left[\min_{i:B_{ir}^{-1}>0} \left(\frac{u_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right), \min_{i:B_{ir}^{-1}<0} \left(\frac{l_{B_i} - x_{B_i}^*}{-x_j^* B_{ir}^{-1}} \right) \right] \text{ if } x_i^* > 0 \tag{3.27}$$

EXAMPLE

As an example, consider the sample problem EX1DIET and the matrix element a_{14}, which is a coefficient for basic variable x_4. In Section 3.1.2, we derived bounds on c_4; we use these bounds and the fact that π_1^* equals 9/160 to derive optimality bounds on $\phi(\alpha)$.

$$\begin{aligned} (35/13)/(-9/160) &\leq \phi \leq (-29/21)/(-9/160) \\ -5600/117 &\leq \phi \leq 4640/189 \\ -47.8632 &\leq \phi \leq 24.5503 \end{aligned}$$

In Section 3.2, we derived bounds on b_1; we use these bounds and the fact that x_4^* equals 4.5 to derive feasibility bounds on $\phi(\alpha)$.

$$\begin{aligned} 560/-4.5 &\leq \phi \leq -100/-4.5 \\ -1120/9 &\leq \phi \leq 200/9 \\ -124.444 &\leq \phi \leq 22.2222 \end{aligned}$$

Hence, $\phi(\alpha)$ must fall between -47.8632 and 22.2222. This implies that

$$-36.8421 \leq \alpha \leq 25.8065$$

since

$$\begin{aligned} \phi(\alpha) &= 1/(B_{11}^{-1} + 1/\alpha) \\ &= 1/(1/160 + 1/\alpha) \\ &= \alpha/(160 + \alpha) \end{aligned}$$

We arrive at the following bounds for a_{14}.

$$123.579 \leq a_{14} \leq 185.8065$$

The full coefficient matrix analysis report for EX1DIET produced by SAMINOS are displayed in Figure 3.3. Note that x_7, x_8, and x_9 are slack variables, that were not included in the MPS input file. Hence, the matrix columns corresponding to x_7, x_8, and x_9 are not included in the matrix report.

MATRIX RANGING ANALYSIS

ROW INDEX	ROW NAME	COL INDEX	COL NAME	LOWER LIMIT	UPPER LIMIT
1	ENERGY	1	OATMEAL	0.5333333E+02	0.1350000E+03
2	PROTEIN	1	OATMEAL	0.2750000E+01	INF
3	CALCIUM	1	OATMEAL	-0.1316250E+03	INF
1	ENERGY	2	CHICKEN	-INF	0.4266667E+03
2	PROTEIN	2	CHICKEN	-INF	INF
3	CALCIUM	2	CHICKEN	-INF	INF
1	ENERGY	3	EGGS	-INF	0.2311111E+03
2	PROTEIN	3	EGGS	-INF	INF
3	CALCIUM	3	EGGS	-INF	INF
1	ENERGY	4	MILK	0.1231579E+03	0.1858065E+03
2	PROTEIN	4	MILK	0.6888889E+01	INF
3	CALCIUM	4	MILK	0.1662222E+03	INF
1	ENERGY	5	PIE	0.3555556E+03	0.4700000E+03
2	PROTEIN	5	PIE	0.1500000E+01	INF
3	CALCIUM	5	PIE	-0.2452500E+03	INF
1	ENERGY	6	PORKBEAN	-INF	0.3377778E+03
2	PROTEIN	6	PORKBEAN	-INF	INF
3	CALCIUM	6	PORKBEAN	-INF	INF

Figure 3.3: Constraint Matrix Analysis for EX1DIET

Appendix A

TECHNICAL DETAILS

This section provides a description of the technical details of SAMINOS. We will describe our implementation of the sensitivity analysis routines, discuss changes the user can make to the code, and provide a list of error codes.

A.1 IMPLEMENTATION DESCRIPTION

In this section, we describe the implementation of the sensitivity analysis formulas described in the previous chapter. There are five source files associated with SAMINOS:

- SAFUNS.FOR
- SAUTIL.FOR
- SACOSTRA.FOR
- SARHSRA.FOR
- SAMATRA.FOR

We will discuss the contents of each of these source files. Critical issues include memory allocation, subroutine design, and data structures.

A.1.1 SENSITIVITY ANALYSIS FUNCTIONS (SAFUNS.FOR)

SAMINOS differs from standard MINOS only in subroutine **MATMOD**. The purpose of **MATMOD** is to allow MINOS users to access the various MINOS data structures before or after MINOS solves a problem. Thus, MINOS is well-suited to solving for the quantities required for sensitivity analysis. The SAMINOS version of **MATMOD** was written so that

it calls other subroutines to perform the cost analysis, the right-hand side analysis, and the constraint matrix analysis. **MATMOD** is called by the MINOS routine MINOS3, which is found in the source file MI10*.FOR; the PC version of MINOS, for example, has a source file named MI10PC.FOR.

The file SAFUNS.FOR includes the SAMINOS version of **MATMOD**, which performs the following functions.

- Initialize the list of cost coefficients to analyze by calling subroutine **SACOAL**.
- Analyze the LP cost coefficients and generate a report by calling subroutine **SACOSA**.
- Initialize the list of right-hand sides to analyze by calling subroutine **SARHAL**.
- Analyze the LP right-hand sides and generate a report by calling subroutine **SARHSA**.
- Initialize the list of LP constraint matrix coefficients to analyze by calling subroutine **SAMAAL**.
- Analyze the LP constraint matrix coefficients and generate a report by calling subroutine **SAMATA**.

A.1.2 UTILITIES (SAUTIL.FOR)

SAUTIL.FOR is used for routines which are held in common with all of the sensitivity analysis routines, including: error handling, matrix and cost coefficient retrieval, tolerance computations, and memory allocation procedures. Memory allocation is an especially important issue in MINOS, and merits further discussion.

MINOS allocates memory by declaring one large double precision array Z and then setting pointers to different locations in Z for the various arrays needed to solve linear and nonlinear programs. Therefore, any memory needed for arrays to be used in sensitivity analysis must be pre-allocated. The length of the array Z is set in the program found in MINOSL.FOR.

MINOS allows its users to reserve memory for their own purposes at the beginning of Z through the SPECS file option WORKSPACE (USER). (See the MINOS User's Guide for more details on the SPECS file.) SAMINOS takes advantage of this in order to reserve memory for the lists of costs, right-hand sides, and matrix coefficients to report, and for the upper and lower bounds on these quantities.

- **SACOAL** is used to set up the list of variables whose cost coefficients should be reported in the sensitivity analysis. Currently, this list consists of all of the variables defined by the user in the MPS input file. Slack variables which are implicitly defined in MINOS are not reported on. **SACOAL** is called by **MATMOD**.

- **SARHAL** is used to set up the list of constraints whose right-hand sides are to be reported on in the sensitivity analysis. Currently, this list consists of all constraints except for the objective. It should be noted that although range constraints are included in this list, they are not reported on. **SARHAL** is called by **MATMOD**.

- **SAMAAL** is used to set up the list of constraint matrix coefficients to be reported on in the sensitivity analysis. Currently, this list consists of all matrix coefficients defined by the user in the MINOS MPS file. Note that matrix coefficients which belong to the objective or to range constraints will not be reported on. If the user wishes to determine the allowable range of a zero-valued element of the constraint matrix, the user should include this element explicitly with value zero in the COLUMNS section of the MPS input file. **SAMAAL** is called by **MATMOD**.

- **SASTO1** is used to allocate memory for the upper and lower bounds on costs, right-hand sides, and matrix coefficients. The workspace used begins at the end of the list of elements to be reported on, which is set up by **SACOAL**, **SARHAL**, or **SAMAAL**. If there isn't enough memory available, an error message will be issued from this routine, and the analysis will terminate. In this case, the user should reset the SPECS file option WORKSPACE (USER) with a larger number. In some cases, the user may have to edit MINOSL.FOR and increase the length of the workspace array Z. **SASTO1** is called from **SACOSA**, **SARHSA**, and **SAMATA** (see below).

What follows is a list of other subroutines found in SAUTIL.

- **SALPOP** is the routine used to determine whether the MINOS problem at hand is indeed a linear program which has been solved to optimality; otherwise, we cannot perform sensitivity analysis.

- **SAGMAT** and **SAGRHS** are used to retrieve constraint matrix and right-hand side elements from the MINOS data structures used to store them.

- **SAIRTY** is used to determine whether a particular constraint is $=$, $\geq$, $\leq$, or a range constraint. If a constraint is a range constraint, sensitivity analysis is not performed for that constraint.

- **SASTTL** is used to set the tolerances required to determine what constitutes a zero-valued reduced cost, pivot or variable. These tolerances are based on what is used by MINOS.

A.1.3 COST COEFFICIENT RANGING (SACOSTRA.FOR)

SACOSTRA consists of all routines relating to the sensitivity analysis of the LP objective cost coefficients.

- **SACOSA** is the main routine for analysis of the LP cost coefficients; it is called by **MATMOD**. **SACOSA** calls **SACOST**, which computes and reports on the range of optimality for the cost coefficients.
- **SACOST** generates a report of optimality ranges on cost coefficients. **SACOST** calls a routine (**SACORA**) to compute the ranges, opens the report file, calls another routine (**SACORP**) to write the report, and closes the report file.
- **SACORA** computes the optimality range for each cost coefficient of a linear program. **SACORA** is called by **SACOST**.
- **SABSEF** is called by **SACORA** in order to compute the effect of a change in the cost coefficient of a basic variable on the reduced cost of a non-basic variable.
- **SACORP** generates a report of the optimality ranges of the LP cost coefficients. **SACORP** is called by **SACOST**.

A.1.4 RIGHT HAND SIDE RANGING (SARHSRA.FOR)

SARHSRA consists of all routines relating to the sensitivity analysis of the LP right-hand sides.

- **SARHSA** is the main routine for the analysis of the LP right-hand sides. **SARHSA** calls the right-hand side range generator **SARHS**, which computes the ranges of feasibility for the LP right hand sides.
- **SARHS** generates a report of feasibility ranges on LP right-hand sides. **SARHS** calls **SARHRA** to compute the ranges, opens the report file, calls **SARHRP** to write the report, and closes the report file. This routine is called by **SARHSA**.
- **SARHRA** computes the feasible range for the LP right-hand sides. **SARHRA** is called by **SARHS**.
- **SARHI** computes the effect of a change in the right-hand side of one of the LP constraints on the feasibility of the current basis. **SARHI** is called by **SARHRA**.
- **SARHRP** writes a report of the feasibility ranges of the LP right-hand sides.

A.1.5 COEFFICIENT MATRIX RANGING (SAMATRA.FOR)

SAMATRA consists of all routines relating to the sensitivity analysis of the LP matrix coefficients.

- **SAMATA** calls the constraint matrix range generator **SAMAT**, which computes the range of feasibility and optimality for the LP constraint matrix.
- **SAMAT** generates a report of feasibility/optimality ranges of the LP constraint matrix elements. **SAMAT** calls **SAMARA** to compute the ranges, opens the report file, calls **SAMARP** to write the report, and closes the report file. This procedure is called by **SAMATA**.
- **SAMARA** computes the feasibility/optimality range for the LP constraint matrix elements. **SAMARA** is called by **SAMAT**.
- **SAMARP** writes a report about the feasibility/optimality ranges of the LP constraint matrix elements. **SAMARP** is called by **SAMAT**.
- **SAGTBD** computes upper and lower feasibility/optimality bounds on LP constraint matrix elements in the following manner. Suppose δ is a change made to a_{ij}. Then **SAGTBD** computes upper and lower bounds on allowable values of δ if the variable x_j is non-basic, or of $\delta/(1+\delta B_{li}^{-1})$ if x_j is the lth basic variable and B is the basis matrix. **SAGTBD** is called by **SAMARA**.
- **PHIINV** calculates the inverse of the function $\phi(y) = y/(1+by)$, which is used in the computation of the upper and lower feasibility/optimality bounds on the LP constraint matrix elements. **PHIINV** is called by **SAMARA**.

A.2 MODIFYING THE OUTPUT REPORTS

SAMINOS is designed to report on all costs, all right-hand sides (except for those corresponding to range constraints), and all constraint matrix elements (except for those corresponding to range constraints). If you wish to limit these reports to certain variables or constraints, you will need to modify SAMINOS.

A.2.1 MODIFYING THE COST REPORT

If you wish to limit the cost report to certain variables, you will have to modify subroutine **SACOAL**, which is found in SAUTIL.FOR. You will need to alter two blocks of code in **SACOAL**.

First, find the following block of code in **SACOAL**.

```
      lstart = (n+1)/2 + 1
```

The variable `lstart` refers to how much of the workspace array Z will be used to store the list of variables whose costs are to be analyzed. You will need to change the variable `n` to a constant or variable representing the number of variables whose costs you wish to report.

To address the variable list itself, you will need to change the following block of code.

```
            do 10 i = 1, n
                nz(i) = i
10          continue
```

The array `nz` is an integer copy of the double precision workspace array Z; note that the elements at the start of Z are used to store the list. To assign the ith item in the list of variables, you will need to address `nz(i)`.

A.2.2 MODIFYING THE RIGHT HAND SIDE REPORT

If you wish to limit the right-hand side report to certain constraints, you will have to modify subroutine **SARHAL**, which is found in SAUTIL.FOR. You will need to alter two blocks of code in **SARHAL**.

First, find the following block of code in **SARHAL**.

```
      lstart = (m+1)/2 + 1
```

The variable `lstart` refers to how much of the workspace array Z will be used to store the list of constraints whose right-hand sides are to be analyzed. You will need to change the variable `m` to a constant or variable representing the number of right-hand sides you wish to report.

To address the right-hand sides list itself, you will need to change the following block of code.

```
            do 10 i = 1, m
                nz(i) = i
10          continue
```

The array `nz` is an integer copy of the double precision workspace array Z; note that the elements at the start of Z are used to store the list. To assign the ith item in the list of constraints, you will need to address `nz(i)`.

A.2.3 MODIFYING THE COEFFICIENT MATRIX REPORT

If you wish to limit the constraint matrix report to a subset of all of the non-zero matrix elements, you will have to modify subroutine **SAMAAL**, which is found in SAUTIL.FOR. You will need to alter two blocks of code in **SAMAAL**.

First, find the following block of code in **SAMAAL**.

```
lstart = hstart + (ne+1)/2
```

The variable `lstart` refers to how much of the workspace array Z will be used to store the list of matrix coefficients to be analyzed. You will need to change the variable `ne` to a constant or variable representing the number of matrix elements to report.

Next, you will need to change the following block of code.

```
      do 10 i = 1, n+1
          nz(i) = ka(i)
10    continue
```

The array `nz` is an integer copy of the double precision workspace array Z. Recall that MINOS stores matrix elements by column; first the explicitly user-specified elements of column 1 are stored in order of row, followed by all user-specified elements of column 2, and so on. For example, the matrix

$$A = \begin{pmatrix} 3 & 0 & 8 & 4 \\ 1 & 2 & 0 & 1 \\ 3 & 1 & 3 & 2 \end{pmatrix}$$

would, if we did not wish to store the zero elements, be stored in an array in the following order:

$$3, 1, 3, 2, 1, 8, 3, 4, 1, 2$$

The array element `ka(i)` refers to the first element of this matrix array that belongs to column i; the array element `nz(i)` will refer to the first element of the list of coefficients to be reported that belongs to column i. `nz(n+1)` will refer to one plus the number of coefficients to be reported on. For example, if we wished to report on elements a_{21}, a_{31}, a_{23}, and a_{34} of our sample matrix, then we would set up `nz` as follows:

```
nz(1) = 1
nz(2) = 3
nz(3) = 3
nz(4) = 4
nz(5) = 5
```

Note that since we would not report on any elements in column 2, we set `nz(2)` equal to `nz(3)`. Also note that we can report on a_{23}, even though a_{23} is zero-valued and may not have been specified as part of the MINOS input.

Now that storage for each column has been set, you will need to alter the following block of code in order to access the matrix elements themselves.

```
          do 20 ie = 1, ne
              i     = ie + nhstrt
              nz(i) = ha(ie)
20        continue
```

The array element `nz(i)` refers to the row index of the `ie`th item in the list of matrix coefficients to report. For example, if we wished to report elements a_{21}, a_{31}, a_{23}, and a_{34} of our sample matrix, we would set `nz` up as follows:

```
nz(1+nhstrt) = 2
nz(2+nhstrt) = 3
nz(3+nhstrt) = 2
nz(4+nhstrt) = 3
```

Note: When setting up this portion of the workspace, it is very important that you refer to elements of `nz` in terms of a position relative to `nhstrt` rather than in terms of an absolute index. This requirement results from the use of the double precision workspace array Z as an integer array in subroutine **SAMAAL**.

A.3 ERROR HANDLING

SAMINOS has limited error handling capabilities; it checks to see that:

- the problem considered is indeed a linear program which has been solved to optimality, and
- there is enough memory to perform the sensitivity analysis.

A list of error messages that the user may encounter is displayed in Table A.3.

Message	Error
SAST01 Error: required storage is x, available storage is y	Not enough memory for bounds; increase WORKSPACE (USER)
SACOAL Error: required storage is $> x$, available storage is y	Not enough memory for list of costs to report; increase WORKSPACE (USER)
SACOSA Error: This problem is an NLP	SAMINOS can only process linear programs; reformulate the program as an LP, or use MINOS instead
SACOSA Error: LP not solved to optimality	Either the linear program does not have an optimal solution, or iterations limit was reached before reaching the optimal solution
SACOSA Error: Insufficient memory for cost analysis	Not enough memory for cost bounds; increase WORKSPACE (USER)
SARHAL Error: required storage is $> x$, available storage is y	Not enough memory for list of right-hand sides to report; increase WORKSPACE (USER)
SARHSA Error: This problem is an NLP	SAMINOS can only process linear programs; reformulate the program as an LP, or use MINOS instead
SARHSA Error: LP not solved to optimality	Either the linear program does not have an optimal solution, or iterations limit was reached before reaching the optimal solution
SARHSA Error: Insufficient memory for RHS analysis	Not enough memory for right-hand side bounds; increase WORKSPACE (USER)
SAMAAL Error: required storage is $> x$, available storage is y	Not enough memory for list of matrix coefficients to report; increase WORKSPACE (USER)
SAMATA Error: This problem is an NLP	SAMINOS can only process linear programs; reformulate the program as an LP, or use MINOS instead
SAMATA Error: LP not solved to optimality	Either the linear program does not have an optimal solution, or iterations limit was reached before reaching the optimal solution
SAMATA Error: Insufficient memory for analysis	Not enough memory for storing matrix bounds; increase WORKSPACE

Table A.1: SAMINOS Error Messages

REFERENCES

Chvátal, V. (1983). "Linear Programming", *W.H. Freeman and Company*, New York, New York.

Dantzig, G.B. (1963)."Linear Programming and Extensions", *Princeton University Press*, Princeton, New Jersey.

Dantzig, G.B. and Thapa M.N. (1997). "Linear Programming 1: Introduction", *Springer-Verlag*, Berlin and New York.

Dantzig, G.B. and Thapa M.N. (1998a). "Linear Programming 2: Theory and Implementation", To be published by *Springer-Verlag*, Berlin and New York.

Dantzig, G.B. and Thapa M.N. (1998b). "Linear Programming 3: Extensions", To be published by *Springer-Verlag*, Berlin and New York.

Lahey Computer Systems, Inc. (1995). "F77L-EM/32 Fortran Reference Manual", *Lahey Computer Systems, Inc.*, Incline Village, Nevada.

Murtagh, B.A. and Saunders, M.A. (1993). "MINOS 5.4 User's Guide", *Report SOL 83-20R*, Department of Operations Research, Stanford University.

Murty, K.G. (1983). "Linear Programming", *John-Wiley and Sons*, New York, New York.