DECIS

Gerd Infanger; Vienna University of Technology; Stanford University

Contents

1 DECIS

1.1 Introduction

DECIS is a system for solving large-scale stochastic programs, programs, which include parameters (coefficients and right-hand sides) that are not known with certainty, but are assumed to be known by their probability distribution. It employs Benders decomposition and allows using advanced Monte Carlo sampling techniques. DECIS includes a variety of solution strategies, such as solving the universe problem, the expected value problem, Monte Carlo sampling within the Benders decomposition algorithm, and Monte Carlo pre-sampling. When using Monte Carlo sampling the user has the option of employing crude Monte Carlo without variance reduction techniques, or using as variance reduction techniques importance sampling or control variates, based on either an additive or a multiplicative approximation function. Pre-sampling is limited to using crude Monte Carlo only.

For solving linear and nonlinear programs (master and subproblems arising from the decomposition) DECIS interfaces with MINOS or CPLEX. MINOS, see Murtagh and Saunders (1983) [5], is a state-of-the-art solver for large-scale linear and nonlinear programs, and CPLEX, see CPLEX Optimization, Inc. (1989–1997) [2], is one of the fastest linear programming solvers available.

For details about the DECIS system consult the DECIS User's Guide, see Infanger (1997) [4]. It includes a comprehensive mathematical description of the methods used by DECIS. In this Guide we concentrate on how to use DECIS directly from GAMS, see Brooke, A., Kendrik, D. and Meeraus, A. (1988) [1], and especially on how to model stochastic programs using the GAMS/DECIS interface. First, however, in section 1.2 we give a brief description of what DECIS can do and what solution strategies it uses. This description has been adapted from the DECIS User's Guide. In section 2 we discuss in detail how to set up a stochastic problem using GAMS/DECIS and give a description of the parameter setting and outputs obtained. In Appendix A we show the GAMS/DECIS formulation of two illustrative examples (APL1P and APL1PC) discussed in the DECIS User's Guide. A list of DECIS error messages are represented in Appendix B.

1.2 What DECIS Can Do

DECIS solves two-stage stochastic linear programs with recourse:

$$\begin{array}{rrrrrrrrl} \min\ z & = & cx & + & E\ f^\omega y^\omega & & & & \\ s/t & & Ax & & & & = & b & \\ & & -B^\omega x & + & D^\omega y^\omega & = & d^\omega & & \\ & & x, & & y^\omega & \geq & 0, & \omega \in \Omega. & \end{array}$$

where x denotes the first-stage, y^ω the second-stage decision variables, c represents the first-stage and f^ω the second-stage objective coefficients, A, b represent the coefficients and right hand sides of the first-stage constraints, and B^ω, D^ω, d^ω represent the parameters of the second-stage constraints, where the transition matrix B^ω couples the two stages. In the literature D^ω is often referred to as the technology matrix or recourse matrix. The first stage parameters are known with certainty. The second stage parameters are random parameters that assume outcomes labeled ω with probability $p(\omega)$, where Ω denotes the set of all possible outcome labels.

At the time the first-stage decision x has to be made, the second-stage parameters are only known by their probability distribution of possible outcomes. Later after x is already determined, an actual outcome of the second-stage parameters will become known, and the second-stage decision y^ω is made based on knowledge of the actual outcome ω. The objective is to find a feasible decision x that minimizes the total expected costs, the sum of first-stage costs and expected second-stage costs.

For discrete distributions of the random parameters, the stochastic linear program can be represented by the

corresponding *equivalent deterministic linear program*:

$$\begin{array}{rcrcrcrcrcrcl}
\min\ z & = & cx & + & p^1fy^1 & + & p^2fy^2 & + & \cdots & + & p^Wfy^W & & \\
s/t & & Ax & & & & & & & & & = & b \\
 & & -B^1x & + & Dy^1 & & & & & & & = & d^1 \\
 & & -B^2x & & & + & Dy^2 & & & & & = & d^2 \\
 & & \vdots & & & & & & \ddots & & & & \vdots \\
 & & -B^Wx & & & & & & & + & Dy^W & = & d^W \\
 & & x, & & y^1, & & y^2, & & \ldots, & & y^W & \geq & 0,
\end{array}$$

which contains all possible outcomes $\omega \in \Omega$. Note that for practical problems W is very large, e.g., a typical number could be 10^{20}, and the resulting equivalent deterministic linear problem is too large to be solved directly.

In order to see the two-stage nature of the underlying decision making process the folowing representation is also often used:

$$\begin{array}{rcrcrcl}
\min & cx & + & E\ z^\omega(x) & & & \\
 & Ax & & & = & b & \\
 & x & & & \geq & 0 &
\end{array}$$

where

$$\begin{array}{rcrcl}
z^\omega(x) & = & \min & f^\omega y^\omega & \\
 & & & D^\omega y^\omega & = \quad d^\omega + B^\omega x \\
 & & & y^\omega & \geq \quad 0,\ \omega \in \Omega = \{1, 2, \ldots, W\}.
\end{array}$$

DECIS employs different strategies to solve two-stage stochastic linear programs. It computes an exact optimal solution to the problem or approximates the true optimal solution very closely and gives a confidence interval within which the true optimal objective lies with, say, 95% confidence.

1.3 Representing Uncertainty

It is favorable to represent the uncertain second-stage parameters in a structure. Using $V = (V_1, \ldots, V_h)$ an h-dimensional independent random vector parameter that assumes outcomes $v^\omega = (v_1, \ldots, v_h)^\omega$ with probability $p^\omega = p(v^\omega)$, we represent the uncertain second-stage parameters of the problem as functions of the independent random parameter V:

$$f^\omega = f(v^\omega), \quad B^\omega = B(v^\omega), \quad D^\omega = D(v^\omega), \quad d^\omega = d(v^\omega).$$

Each component V_i has outcomes $v_i^{\omega_i}$, $\omega_i \in \Omega_i$, where ω_i labels a possible outcome of component i, and Ω_i represents the set of all possible outcomes of component i. An outcome of the random vector

$$v^\omega = (v_1^{\omega_1}, \ldots, v_h^{\omega_h})$$

consists of h independent component outcomes. The set

$$\Omega = \Omega_1 \times \Omega_2 \times \ldots \times \Omega_h$$

represents the crossing of sets Ω_i. Assuming each set Ω_i contains W_i possible outcomes, $|\Omega_i| = W_i$, the set Ω contains $W = \prod W_i$ elements, where $|\Omega| = W$ represents the number of all possible outcomes of the random vector V. Based on independence, the joint probability is the product

$$p^\omega = p_1^{\omega_1} p_2^{\omega_2} \cdots p_h^{\omega_h}.$$

Let η denote the vector of all second-stage random parameters, e.g., $\eta = \text{vec}(f, B, D, d)$. The outcomes of η may be represented by the following general linear dependency model:

$$\eta^\omega = \text{vec}(f^\omega, B^\omega, d^\omega, d^\omega) = Hv^\omega, \quad \omega \in \Omega$$

where H is a matrix of suitable dimensions. DECIS can solve problems with such general linear dependency models.

1.4 Solving the Universe Problem

We refer to the universe problem if we consider all possible outcomes $\omega \in \Omega$ and solve the corresponding problem exactly. This is not always possible, because there may be too many possible realizations $\omega \in \Omega$. For solving the problem DECIS employs Benders decomposition, splitting the problem into a master problem, corresponding to the first-stage decision, and into subproblems, one for each $\omega \in \Omega$, corresponding to the second-stage decision. The details of the algorithm and techniques used for solving the universe problem are discussed in The DECIS User's Manual.

Solving the universe problem is referred to as strategy 4. Use this strategy only if the number of universe scenarios is reasonably small. There is a maximum number of universe scenarios DECIS can handle, which depends on your particular resources.

1.5 Solving the Expected Value Problem

The expected value problem results from replacing the stochastic parameters by their expectation. It is a linear program that can also easily be solved by employing a solver directly. Solving the expected value problem may be useful by itself (for example as a benchmark to compare the solution obtained from solving the stochastic problem), and it also may yield a good starting solution for solving the stochastic problem. DECIS solves the expected value problem using Benders decomposition. The details of generating the expected value problem and the algorithm used for solving it are discussed in the DECIS User's Manual. To solve the expected value problem choose strategy 1.

1.6 Using Monte Carlo Sampling

As noted above, for many practical problems it is impossible to obtain the universe solution, because the number of possible realizations $|\Omega|$ is way too large. The power of DECIS lies in its ability to compute excellent approximate solutions by employing Monte Carlo sampling techniques. Instead of computing the expected cost and the coefficients and the right-hand sides of the Benders cuts exactly (as it is done when solving the universe problem), DECIS, when using Monte Carlo sampling, estimates the quantities in each iteration using an independent sample drawn from the distribution of the random parameters. In addition to using crude Monte Carlo, DECIS uses importance sampling or control variates as variance reduction techniques.

The details of the algorithm and the different techniques used are described in the DECIS User's Maual. You can choose crude Monte Carlo, referred to as strategy 6, Monte Carlo importance sampling, referred to as strategy 2, or control variates, referred to as strategy 10. Both Monte Carlo importance sampling and control variates have been shown for many problems to give a better approximation compared to employing crude Monte Carlo sampling.

When using Monte Carlo sampling DECIS computes a close approximation to the true solution of the problem, and estimates a close approximation of the true optimal objective value. It also computes a confidence interval within which the true optimal objective of the problem lies, say with 95% confidence. The confidence interval is based on rigorous statistical theory. An outline of how the confidence interval is computed is given in the DECIS User's Manual. The size of the confidence interval depends on the variance of the second-stage cost of the stochastic problem and on the sample size used for the estimation. You can expect the confidence interval to be very small, especially when you employ importance sampling or control variates as a variance reduction technique.

When employing Monte Carlo sampling techniques you have to choose a sample size (set in the parameter file). Clearly, the larger the sample size the better will be the approximate solution DECIS computes, and the smaller will be the confidence interval for the true optimal objective value. The default value for the sample size is 100. Setting the sample size too small may lead to bias in the estimation of the confidence interval, therefore the sample size should be at least 30.

1.7 Monte Carlo Pre-sampling

We refer to pre-sampling when we first take a random sample from the distribution of the random parameters and then generate the approximate stochastic problem defined by the sample. The obtained approximate problem is then solved exactly using decomposition. This is in contrast to the way we used Monte Carlo sampling in the previous section, where we used Monte Carlo sampling in each iteration of the decomposition.

The details of the techniques used for pre-sampling are discussed in the DECIS User's Manual. DECIS computes the exact solution of the sampled problem using decomposition. This solution is an approximate solution of the original stochastic problem. Besides this approximate solution, DECIS computes an estimate of the expected cost corresponding to this approximate solution and a confidence interval within which the true optimal objective of the original stochastic problem lies with, say, 95% confidence. The confidence interval is based on statistical theory, its size depends on the variance of the second-stage cost of the stochastic problem and on the sample size used for generating the approximate problem. In conjunction with pre-sampling no variance reduction techniques are currently implemented.

Using Monte Carlo pre-sampling you have to choose a sample size. Clearly, the larger the sample size you choose, the better will be the solution DECIS computes, and the smaller will be the confidence interval for the true optimal objective value. The default value for the sample size is 100. Again, setting the sample size as too small may lead to a bias in the estimation of the confidence interval, therefore the sample size should be at least 30.

For using Monte Carlo pre-sampling choose strategy 8.

1.8 Regularized Decomposition

When solving practical problems, the number of Benders iterations can be quite large. In order to control the decomposition, with the hope to reduce the iteration count and the solution time, DECIS makes use of regularization. When employing regularization, an additional quadratic term is added to the objective of the master problem, representing the square of the distance between the best solution found so far (the incumbent solution) and the variable x. Using this term, DECIS controls the distance of solutions in different decomposition iterations.

For enabling regularization you have to set the corresponding parameter. You also have to choose the value of the constant rho in the regularization term. The default is regularization disabled. Details of how DECIS carries out regularization are represented in the DECIS User's Manual.

Regularization is only implemented when using MINOS as the optimizer for solving subproblems. Regularization has proven to be helpful for problems that need a large number of Benders iteration when solved without regularization. Problems that need only a small number of Benders iterations without regularization are not expected to improve much with regularization, and may need even more iterations with regularization than without.

2 GAMS/DECIS

GAMS stands for General Algebraic Modeling Language, and is one of the most widely used modeling languages. Using DECIS directly from GAMS spares you from worrying about all the details of the input formats. It makes the problem formulation much easier but still gives you almost all the flexibility of using DECIS directly.

The link from GAMS to DECIS has been designed in such a way that almost no extensions to the GAMS modeling language were necessary for carrying out the formulation and solution of stochastic programs. In a next release of GAMS, however, additions to the language are planned that will allow you to model stochastic programs in an even more elegant way.

2.1 Setting up a Stochastic Program Using GAMS/DECIS

The interface from GAMS to DECIS supports the formulation and solution of stochastic *linear* programs. DECIS solves them using two-stage decomposition. The GAMS/DECIS interface resembles closely the structure of the

SMPS (stochastic mathematical programming interface) discussed in the DECIS User's Manual. The specification of a stochastic problem using GAMS/DECIS uses the following components:

- the deterministic (core) model,
- the specification of the decision stages,
- the specification of the random parameters, and
- setting DECIS to be the optimizer to be used.

2.2 Starting with the Deterministic Model

The core model is the deterministic linear program where all random parameters are replaced by their mean or by a particular realization. One could also see it as a GAMS model model without any randomness. It could be a deterministic model that you have, which you intend to expand to a stochastic one. Using DECIS with GAMS allows you to easily extend a deterministic linear programming model to a stochastic one. For example, the following GAMS model represents the a deterministic version of the electric power expansion planning illustrative example discussed in Infanger (1994).

```
*   APL1P test model
*   Dr. Gerd Infanger, November 1997
*   Deterministic Program

set g generators / g1, g2/;
set dl demand levels /h, m, l/;

parameter alpha(g) availability / g1  0.68,  g2  0.64 /;
parameter ccmin(g) min capacity / g1  1000,  g2  1000 /;
parameter ccmax(g) max capacity / g1 10000,  g2 10000 /;
parameter c(g) investment       / g1   4.0,  g2   2.5 /;

table f(g,dl) operating cost
             h    m    l
   g1        4.3  2.0  0.5
   g2        8.7  4.0  1.0;

parameter d(dl) demand                   / h  1040, m  1040, l  1040 /;
parameter us(dl) cost of unserved demand / h    10, m    10, l    10 /;

free variable tcost            total cost;
positive variable x(g)         capacity of generators;
positive variable y(g, dl)     operating level;
positive variable s(dl)        unserved demand;

equations
cost        total cost
cmin(g)     minimum capacity
cmax(g)     maximum capacity
omax(g)     maximum operating level
demand(dl)  satisfy demand;

cost .. tcost =e=     sum(g, c(g)*x(g))
                    + sum(g, sum(dl, f(g,dl)*y(g,dl)))
                    + sum(dl,us(dl)*s(dl));

cmin(g) ..    x(g) =g= ccmin(g);
cmax(g) ..    x(g) =l= ccmax(g);
omax(g) ..    sum(dl, y(g,dl)) =l= alpha(g)*x(g);
demand(dl) .. sum(g, y(g,dl)) + s(dl) =g= d(dl);

model apl1p /all/;

option lp=minos5;
solve apl1p using lp minimizing tcost;
```

```
scalar ccost capital cost;
scalar ocost operating cost;
ccost = sum(g, c(g) * x.l(g));
ocost = tcost.l -  ccost;
display x.l, tcost.l, ccost, ocost, y.l, s.l;
```

2.3 Setting the Decision Stages

Next in order to extend a deterministic model to a stochastic one you must specify the decision stages. DECIS solves stochastic programs by two-stage decomposition. Accordingly, you must specify which variables belong to the first stage and which to the second stage, as well as which constraints are first-stage constraints and which are second-stage constraints. First stage constraints involve only first-stage variables, second-stage constraints involve both first- and second-stage variables. You must specify the stage of a variable or a constraint by setting the stage suffix ".STAGE" to either one or two depending on if it is a first or second stage variable or constraint. For example, expanding the illustrative model above by

```
* setting decision stages
x.stage(g)       = 1;
y.stage(g, dl)   = 2;
s.stage(dl)      = 2;
cmin.stage(g)    = 1;
cmax.stage(g)    = 1;
omax.stage(g)    = 2;
demand.stage(dl) = 2;
```

would make x(g) first-stage variables, y(g, dl) and s(dl) second-stage variables, cmin(g) and cmax(g) first-stage constraints, and omax(g) and demand(g) second-stage constraints. The objective is treated separately, you don't need to set the stage suffix for the objective variable and objective equation.

It is noted that the use of the `.stage` variable and equation suffix causes the GAMS scaling facility through the `.scale` suffices to be unavailable. Stochastic models have to be scaled manually.

2.4 Specifying the Stochastic Model

DECIS supports any linear dependency model, i.e., the outcomes of an uncertain parameter in the linear program are a linear function of a number of independent random parameter outcomes. DECIS considers only discrete distributions, you must approximate any continuous distributions by discrete ones. The number of possible realizations of the discrete random parameters determines the accuracy of the approximation. A special case of a linear dependency model arises when you have only independent random parameters in your model. In this case the independent random parameters are mapped one to one into the random parameters of the stochastic program. We will present the independent case first and then expand to the case with linear dependency. According to setting up a linear dependency model we present the formulation in GAMS by first defining independent random parameters and then defining the distributions of the uncertain parameters in your model.

2.4.1 Specifying Independent Random Parameters

There are of course many different ways you can set up independent random parameters in GAMS. In the following we show one possible way that is generic and thus can be adapted for different models. The set-up uses the set stoch for labeling outcome named "out" and probability named "pro" of each independent random parameter. In the following we show how to define an independent random parameter, say, v1. The formulation uses the set omega1 as driving set, where the set contains one element for each possible realization the random parameter can assume. For example, the set omega1 has four elements according to a discrete distribution of four possible outcomes. The distribution of the random parameter is defined as the parameter v1, a two-dimensional array of outcomes "out" and corresponding probability "pro" for each of the possible realizations of the set omega1, "o11", "o12", "o13", and "o14". For example, the random parameter v1 has outcomes of $-1.0, -0.9, -0.5, -0.1$ with probabilities $0.2, 0.3, 0.4, 0.1$, respectively. Instead of using assignment statements for inputting the different realizations and corresponding probabilities you could also use the table statement. You could also the table

statement would work as well. Always make sure that the sum of the probabilities of each independent random parameter adds to one.

```
* defining independent stochastic parameters
set stoch /out, pro /;
set omega1 / o11, o12, o13, o14 /;

table v1(stoch, omega1)
      o11   o12   o13   o14
out  -1.0  -0.9  -0.5  -0.1
pro   0.2   0.3   0.4   0.1
;
```

Random parameter v1 is the first out of five independent random parameters of the illustrative model APL1P, where the first two represent the independent availabilities of the generators g1 and g2 and the latter three represent the independent demands of the demand levels h, m, and l. We also represent the definitions of the remaining four independent random parameters. Note that random parameters v3, v4, and v5 are identically distributed.

```
set omega2 / o21, o22, o23, o24, o25 /;
table v2(stoch, omega2)
      o21   o22   o23   o24   o25
out  -1.0  -0.9  -0.7  -0.1  -0.0
pro   0.1   0.2   0.5   0.1   0.1
;


set omega3 / o31, o32, o33, o34 /;
table v3(stoch, omega1)
      o11   o12   o13   o14
out   900  1000  1100  1200
pro  0.15  0.45  0.25  0.15
;


set omega4 / o41, o42, o43, o44 /;
table v4(stoch,omega1)
      o11   o12   o13   o14
out   900  1000  1100  1200
pro  0.15  0.45  0.25  0.15
;

set omega5 / o51, o52, o53, o54 /;
table v5(stoch,omega1)
      o11   o12   o13   o14
out   900  1000  1100  1200
pro  0.15  0.45  0.25  0.15
;
```

2.4.2 Defining the Distributions of the Uncertain Parameters in the Model

Having defined the independent stochastic parameters (you may copy the setup above and adapt it for your model), we next define the stochastic parameters in the GAMS model. The stochastic parameters of the model are defined by writing a file, the GAMS stochastic file, using the put facility of GAMS. The GAMS stochastic file resembles closely the stochastic file of the SMPS input format. The main difference is that we use the row, column, bounds, and right hand side names of the GAMS model and that we can write it in free format.

Independent Stochastic Parameters

First we describe the case where all stochastic parameters in the model are independent, see below the representation of the stochastic parameters for the illustrative example APL1P, which has five independent stochastic parameters.

First define the GAMS stochastic file "MODEL.STG" (only the exact name in uppercase letters is supported) and set up GAMS to write to it. This is done by the first two statements. You may want to consult the GAMS manual for how to use put for writing files. The next statement "INDEP DISCRETE" indicates that a section of independent stochastic parameters follows. Then we write all possible outcomes and corresponding probabilities for each stochastic parameter best by using a loop statement. Of course one could also write each line separately, but this would not look nicely. Writing a "*" between the definitions of the independent stochastic parameters is merely for optical reasons and can be omitted.

```
* defining distributions (writing file MODEL.STG)
file stg /MODEL.STG/;
put stg;

put "INDEP DISCRETE" /;
loop(omega1,
put "x g1  omax g1   ", v1("out", omega1), " period2 ", v1("pro", omega1) /;
);
put "*" /;
loop(omega2,
put "x g2  omax g2   ", v2("out", omega2), " period2 ", v2("pro", omega2) /;
);
put "*" /;
loop(omega3,
put "RHS  demand h   ", v3("out", omega3), " period2 ", v3("pro", omega3) /;
);
put "*" /;
loop(omega4,
put "RHS  demand m   ", v4("out", omega4), " period2 ", v4("pro", omega4) /;
)
put "*" /;
loop(omega5,
put "RHS  demand l   ", v5("out", omega5), " period2 ", v5("pro", omega5) /;
);
putclose stg;
```

In the example APL1P the first stochastic parameter is the availability of generator g1. In the model the parameter appears as the coefficient of variable x(g1) in equation omax(g1). The definition using the put statement first gives the stochastic parameter as the intersection of variable x(g1) with equation omax(g1), but without having to type the braces, thus *x g1 omax g1*, then the outcome *v1("out", omega1)* and the probability *v1("pro", omega1)* separated by *"period2"*. The different elements of the statement must be separated by blanks. Since the outcomes and probabilities of the first stochastic parameters are driven by the set omega1 we loop over all elements of the set omega1. We continue and define all possible outcomes for each of the five independent stochastic parameters.

In the example of independent stochastic parameters, the specification of the distribution of the stochasic parameters using the put facility creates the following file "MODEL.STG", which then is processed by the GAMS/DECIS interface:

```
INDEP DISCRETE
x g1  omax g1          -1.00 period2        0.20
x g1  omax g1          -0.90 period2        0.30
x g1  omax g1          -0.50 period2        0.40
x g1  omax g1          -0.10 period2        0.10
*
x g2  omax g2          -1.00 period2        0.10
x g2  omax g2          -0.90 period2        0.20
x g2  omax g2          -0.70 period2        0.50
x g2  omax g2          -0.10 period2        0.10
x g2  omax g2           0.00 period2        0.10
*
RHS  demand h         900.00 period2        0.15
RHS  demand h        1000.00 period2        0.45
RHS  demand h        1100.00 period2        0.25
RHS  demand h        1200.00 period2        0.15
*
RHS  demand m         900.00 period2        0.15
RHS  demand m        1000.00 period2        0.45
RHS  demand m        1100.00 period2        0.25
```

```
RHS  demand m        1200.00 period2         0.15
*
RHS  demand l         900.00 period2         0.15
RHS  demand l        1000.00 period2         0.45
RHS  demand l        1100.00 period2         0.25
RHS  demand l        1200.00 period2         0.15
```

For defining stochastic parameters in the right-hand side of the model use the keyword *RHS* as the column name, and the equation name of the equation which right-hand side is uncertain, see for example the specification of the uncertain demands *RHS demand h*, *RHS demand m*, and *RHS demand l*. For defining uncertain bound parameters you would use the keywords *UP*, *LO*, or *FX*, the string *bnd*, and the variable name of the variable, which upper, lower, or fixed bound is uncertain.

Note all the keywords for the definitions are in capital letters, i.e., "INDEP DISCRETE", "RHS", and not represented in the example "UP", "LO", and "FX".

It is noted that in GAMS equations, variables may appear in the right-hand side, e.g. "`EQ..  X+1 =L= 2*Y`". When the coefficient 2 is a random variable, we need to be aware that GAMS will generate the following LP row `X - 2*Y =L= -1`. Suppose the probability distribution of this random variable is given by:

```
set s scenario /pessimistic, average, optimistic/;
parameter outcome(s) / pessimistic 1.5
                       average     2.0
                       optimistic  2.3 /;
parameter prob(s)    / pessimistic 0.2
                       average     0.6
                       optimistic  0.2 /;
```

then the correct way of generating the entries in the stochastic file would be:

```
loop(s,
   put  "Y EQ ",(-outcome(s))," PERIOD2 ",prob(s)/;
);
```

Note the negation of the outcome parameter. Also note that expressions in a PUT statement have to be surrounded by parentheses. GAMS reports in the *row listing* section of the listing file how equations are generated. You are encouraged to inspect the row listing how coefficients appear in a generated LP row.

Dependent Stochastic Parameters

Next we describe the case of general linear dependency of the stochastic parameters in the model, see below the representation of the stochastic parameters for the illustrative example APL1PCA, which has three dependent stochastic demands driven by two independent stochastic random parameters. First we give the definition of the two independent stochastic parameters, which in the example happen to have two outcomes each.

```
* defining independent stochastic parameters
set stoch /out, pro/;

set omega1 / o11, o12 /;
table v1(stoch,omega1)
       o11  o12
out    2.1  1.0
pro    0.5  0.5 ;

set omega2 / o21, o22 /;
table v2(stoch, omega2)
       o21  o22
out    2.0  1.0
pro    0.2  0.8 ;
```

We next define the parameters of the transition matrix from the independent stochastic parameters to the dependent stochastic parameters of the model. We do this by defining two parameter vectors, where the vector $hm1$

gives the coefficients of the independent random parameter $v1$ in each of the three demand levels and the vector $hm2$ gives the coefficients of the independent random parameter $v2$ in each of the three demand levels.

```
parameter hm1(dl) / h  300., m  400., l  200. /;
parameter hm2(dl) / h  100., m  150., l  300. /;
```

Again first define the GAMS stochastic file "MODEL.STG" and set GAMS to write to it. The statement *BLOCKS DISCRETE* indicates that a section of linear dependent stochastic parameters follows.

```
* defining distributions (writing file MODEL.STG)
file stg / MODEL.STG /;
put stg;

put "BLOCKS DISCRETE" /;
scalar h1;
loop(omega1,
put "BL  v1   period2 ", v1("pro", omega1)/;
loop(dl,
h1 = hm1(dl) * v1("out", omega1);
put "RHS demand ", dl.tl:1, " ", h1/;
);
);
loop(omega2,
put " BL v2   period2 ", v2("pro", omega2) /;
loop(dl,
h1 = hm2(dl) * v2("out", omega2);
put "RHS demand ", dl.tl:1, " ", h1/;
);
);
putclose stg;
```

Dependent stochastic parameters are defined as functions of independent random parameters. The keyword *BL* labels a possible realization of an independent random parameter. The name besides the *BL* keyword is used to distinguish between different outcomes of the same independent random parameter or a different one. While you could use any unique names for the independent random parameters, it appears natural to use the names you have already defined above, e.g., $v1$ and $v2$. For each realization of each independent random parameter define the outcome of every dependent random parameter (as a function of the independent one). If a dependent random parameter in the GAMS model depends on two or more different independent random parameter the contributions of each of the independent parameters are added. We are therefore in the position to model any linear dependency model. (Note that the class of models that can be accommodated here is more general than linear. The functions, with which an independent random variable contributes to the dependent random variables can be any ones in one argument. As a general rule, any stochastic model that can be estimated by linear regression is supported by GAMS/DECIS.)

Define each independent random parameter outcome and the probability associated with it. For example, the statement starting with *BL v1 period2* indicates that an outcome of (independent random parameter) *v1* is being defined. The name *period2* indicates that it is a second-stage random parameter, and *v1("pro", omega1)* gives the probability associated with this outcome. Next list all random parameters dependent on the independent random parameter outcome just defined. Define the dependent stochastic parameter coefficients by the GAMS variable name and equation name, or "RHS" and variable name, together with the value of the parameter associated with this realization. In the example, we have three dependent demands. Using the scalar $h1$ for intermediately storing the results of the calculation, looping over the different demand levels dl we calculate *h1 = hm1(dl) * v1("out", omega1)* and define the dependent random parameters as the right-hand sides of equation $demand(dl)$.

When defining an independent random parameter outcome, if the block name is the same as the previous one (e.g., when *BL v1* appears the second time), a different outcome of the same independent random parameter is being defined, while a different block name (e.g., when *BL v2* appears the first time) indicates that the first outcome of a different independent random parameter is being defined. You must ensure that the probabilities of the different outcomes of each of the independent random parameters add up to one. The loop over all elements of $omega1$ defines all realizations of the independent random parameter $v1$ and the loop over all elements of $omega2$ defines all realizations of the independent random parameter $v2$.

Note for the first realization of an independent random parameter, you *must* define all dependent parameters and their realizations. The values entered serve as a base case. For any other realization of an independent random parameter you only need to define the dependent parameters that have different coefficients than have been defined in the base case. For those not defined in a particular realization, their values of the base case are automatically added.

In the example of dependent stochastic parameters above, the specification of the distribution of the stochastic parameters using the put facility creates the following file "MODEL.STG", which then is processed by the GAMS/DECIS interface:

```
BLOCKS DISCRETE
BL  v1   period2         0.50
RHS demand h       630.00
RHS demand m       840.00
RHS demand l       420.00
BL  v1   period2         0.50
RHS demand h       300.00
RHS demand m       400.00
RHS demand l       200.00
 BL v2   period2         0.20
RHS demand h       200.00
RHS demand m       300.00
RHS demand l       600.00
 BL v2   period2         0.80
RHS demand h       100.00
RHS demand m       150.00
RHS demand l       300.00
```

Again all the keywords for the definitions are in capital letters, i.e., "BLOCKS DISCRETE", "BL", "RHS", and not represented in the example "UP", "LO", and "FX".

Note that you can only define random parameter coefficients that are nonzero in your GAMS model. When setting up the deterministic core model put a nonzero entry as a placeholder for any coefficient that you wish to specify as a stochastic parameter. Specifying a random parameter at the location of a zero coefficient in the GAMS model causes DECIS to terminate with an error message.

2.5 Setting DECIS as the Optimizer

After having finished the stochastic definitions you must set DECIS as the optimizer. This is done by issuing the following statements:

```
* setting DECIS as optimizer
* DECISM uses MINOS, DECISC uses CPLEX
option lp=decism;
apl1p.optfile = 1;
```

The statement *option lp = decism* sets DECIS with the MINOS LP engine as the optimizer to be used for solving the stochastic problem. Note that if you do not use DECIS, but instead use any other linear programming optimizer, your GAMS model will still run and optimize the deterministic core model that you have specified. The statement *apl1p.optfile = 1* forces GAMS to process the file DECIS.OPT, in which you may define any DECIS parameters.

2.5.1 Setting Parameter Options in the GAMS Model

The options iteration limit and resource limit can be set directly in your GAMS model file. For example, the following statements

```
option iterim = 1000;
option reslim = 6000;
```

constrain the number of decomposition iterations to be less than or equal to 1000, and the elapsed time for running DECIS to be less than or equal to 6000 seconds or 100 minutes.

2.5.2 Setting Parameters in the DECIS Options File

In the DECIS options file DECIS.OPT you can specify parameters regarding the solution algorithm used and control the output of the DECIS program. There is a record for each parameter you want to specify. Each record consists of the value of the parameter you want to specify and the keyword identifying the parameter, separated by a blank character or a comma. You may specify parameters with the following keywords: "istrat", "nsamples", "nzrows", "iwrite", "ibug", "iscratch", "ireg", "rho", "tolben", and "tolw" *in any order*. Each keyword can be specified in lower case or upper case text in the format (A10). Since DECIS reads the records in free format you don't have to worry about the format, but some computers require that the text is inputted in quotes. Parameters that are not specified in the parameter file automatically assume their default values.

istrat — Defines the solution strategy used. The default value is istrat = 3.

- istrat = 1 Solves the expected value problem. All stochastic parameters are replaced by their expected values and the corresponding deterministic problem is solved using decomposition.
- istrat = 2 Solves the stochastic problem using Monte Carlo importance sampling. You have to additionally specify what approximation function you wish to use, and the sample size used for the estimation, see below.
- istrat = 3 Refers to istrat = 1 plus istrat = 2. First solves the expected value problem using decomposition, then continues and solves the stochastic problem using importance sampling.
- istrat = 4 Solves the stochastic universe problem by enumerating all possible combinations of realizations of the second-stage random parameters. It gives you the exact solution of the stochastic program. This strategy may be impossible, because there may be way too many possible realizations of the random parameters.
- istrat = 5 Refers to istrat = 1 plus istrat = 4. First solves the expected value problem using decomposition, then continues and solves the stochastic universe problem by enumerating all possible combinations of realizations of second-stage random parameters.
- istrat = 6 Solves the stochastic problem using crude Monte Carlo sampling. No variance reduction technique is applied. This strategy is especially useful if you want to test a solution obtained by using the evaluation mode of DECIS. You have to specify the sample size used for the estimation. There is a maximum sample size DECIS can handle. However, this maximum sample size does not apply when using crude Monte Carlo. Therefore, in this mode you can specify very large sample sizes, which is useful when evaluating a particular solution.
- istrat = 7 Refers to istrat = 1 plus istrat = 6. First solves the expected value problem using decomposition, then continues and solves the stochastic problem using crude Monte Carlo sampling.
- istrat = 8 Solves the stochastic problem using Monte Carlo pre-sampling. A Monte Carlo sample out of all possible universe scenarios, sampled from the original probability distribution, is taken, and the corresponding "sample problem" is solved using decomposition.
- istrat = 9 Refers to istrat = 1 plus istrat = 8. First solves the expected value problem using decomposition, then continues and solves the stochastic problem using Monte Carlo pre-sampling.
- istrat = 10 Solves the stochastic problem using control variates. You also have to specify what approximation function and what sample size should be used for the estimation.
- istrat = 11 Refers to istrat = 1 plus istrat = 10. First solves the expected value problem using decomposition, then continues and solves the stochastic problem using control variates.

nsamples — Sample size used for the estimation. It should be set greater or equal to 30 in order to fulfill the assumption of large sample size used for the derivation of the probabilistic bounds. The default value is nsamples = 100.

nzrows — Number of rows reserved for cuts in the master problem. It specifies the maximum number of different cuts DECIS maintains during the course of the decomposition algorithm. DECIS adds one cut during each iteration. If the iteration count exceeds nzrows, then each new cut replaces a previously generated cut, where the cut is replaced that has the maximum slack in the solution of the (pseudo) master. If nzrows is specified as too small then DECIS may not be able to compute a solution and stops with an error message.

If nzrows is specified as too large the solution time will increase. As an approximate rule set nzrows greater than or equal to the number of first-stage variables of the problem. The default value is nzrows = 100.

iwrite — Specifies whether the optimizer invoked for solving subproblems writes output or not. The default value is iwrite = 0.

iwrite = 0 No optimizer output is written.

iwrite = 1 Optimizer output is written to the file "MODEL.MO" in the case MINOS is used for solving subproblems or to the file MODEL.CPX in the case CPLEX is used for solving subproblems. The output level of the output can be specified using the optimizer options. It is intended as a debugging device. If you set iwrite = 1, for every master problem and for every subproblem solved the solution output is written. For large problems and large sample sizes the files"MODEL.MO" or "MODEL.CPX" may become very large, and the performance of DECIS may slow down.

ibug — Specifies the detail of debug output written by DECIS. The output is written to the file "MODEL.SCR", but can also be redirected to the screen by a separate parameter. The higher you set the number of ibug the more output DECIS will write. The parameter is intended to help debugging a problem and should be set to ibug = 0 for normal operation. For large problems and large sample sizes the file "MODEL.SCR" may become very large, and the performance of DECIS may slow down. The default value is ibug = 0.

ibug = 0 This is the setting for which DECIS does not write any debug output.

ibug = 1 In addition to the standard output, DECIS writes the solution of the master problem on each iteration of the Benders decomposition algorithm. Thereby it only writes out variable values which are nonzero. A threshold tolerance parameter for writing solution values can be specified, see below.

ibug = 2 In addition to the output of ibug = 1, DECIS writes the scenario index and the optimal objective value for each subproblem solved. In the case of solving the universe problem, DECIS also writes the probability of the corresponding scenario.

ibug = 3 In addition to the output of ibug = 2, DECIS writes information regarding importance sampling. In the case of using the additive approximation function, it reports the expected value for each i-th component of $\bar{\Gamma}_i$, the individual sample sizes N_i, and results from the estimation process. In the case of using the multiplicative approximation function it writes the expected value of the approximation function $\bar{\Gamma}$ and results from the estimation process.

ibug = 4 In addition to the output of ibug = 3, DECIS writes the optimal dual variables of the cuts on each iteration of the master problem.

ibug = 5 In addition to the output of ibug = 4, DECIS writes the coefficients and the right-hand side of the cuts on each iteration of the decomposition algorithm. In addition it checks if the cut computed is a support to the recourse function (or estimated recourse function) at the solution $\hat{x}^k$ at which it was generated. If it turns out that the cut is not a support, DECIS writes out the value of the (estimated) cut and the value of the (estimated) second stage cost at $\hat{x}^k$.

ibug = 6 In addition to the output of ibug = 5, DECIS writes a dump of the master problem and the subproblem in MPS format after having decomposed the problem specified in the core file. The dump of the master problem is written to the file "MODEL.P01" and the dump of the subproblem is written to the file "MODEL.P02". DECIS also writes a dump of the subproblem after the first iteration to the file "MODEL.S02".

iscratch — Specifies the internal unit number to which the standard and debug output is written. The default value is iscratch = 17, where the standard and debug output is written to the file "MODEL.SCR". Setting iscratch = 6 redirects the output to the screen. Other internal unit numbers could be used, e.g., the internal unit number of the printer, but this is not recommended.

ireg — Specifies whether or not DECIS uses regularized decomposition for solving the problem. This option is considered if MINOS is used as a master and subproblem solver, and is not considered if using CPLEX, since regularized decomposition uses a nonlinear term in the objective. The default value is ireg = 0.

rho — Specifies the value of the ρ parameter of the regularization term in the objective function. You will have to experiment to find out what value of rho works best for the problem you want to solve. There is no rule of thumb as to what value should be chosen. In many cases it has turned out that regularized decomposition reduces the iteration count if standard decomposition needs a large number of iterations. The default value is rho = 1000.

tolben — Specifies the tolerance for stopping the decomposition algorithm. The parameter is especially important for deterministic solution strategies, i.e., 1, 4, 5, 8, and 9. Choosing a very small value of tolben may result in a significantly increased number of iterations when solving the problem. The default value is 10^{-7}.

tolw — Specifies the nonzero tolerance when writing debug solution output. DECIS writes only variables whose values are nonzero, i.e., whose absolute optimal value is greater than or equal to tolw. The default value is 10^{-9}.

Example

In the following example the parameters istrat = 7, nsamples = 200, and nzrows = 200 are specified. All other parameters are set at their default values. DECIS first solves the expected value problem and then the stochastic problem using crude Monte Carlo sampling with a sample size of nsamples = 200. DECIS reserves space for a maximum of nzrows = 50 cuts.

```
7       "ISTRAT"
200     "NSAMPLES"
50      "NZROWS"
```

2.5.3 Setting MINOS Parameters in the MINOS Specification File

When you use MINOS as the optimizer for solving the master and the subproblems, you must specify optimization parameters in the MINOS specification file "MINOS.SPC". Each record of the file corresponds to the specification of one parameter and consists of a keyword and the value of the parameter in free format. Records having a "*" as their first character are considered as comment lines and are not further processed. For a detailed description of these parameters, see the MINOS Users' Guide (Murtagh and Saunders (1983) [5]. The following parameters should be specified with some consideration:

AIJ TOLERANCE — Specifies the nonzero tolerance for constraint matrix elements of the problem. Matrix elements a_{ij} that have a value for which $|a_{ij}|$ is less than "AIJ TOLERANCE" are considered by MINOS as zero and are automatically eliminated from the problem. It is wise to specify "AIJ TOLERANCE 0.0 "

SCALE — Specifies MINOS to scale the problem ("SCALE YES") or not ("SCALE NO"). It is wise to specify "SCALE NO".

ROWS — Specifies the number of rows in order for MINOS to reserve the appropriate space in its data structures when reading the problem. "ROWS" should be specified as the number of constraints in the core problem or greater.

COLUMNS — Specifies the number of columns in order for MINOS to reserve the appropriate space in its data structures when reading the problem. "COLUMNS" should be specified as the number of variables in the core problem or greater.

ELEMENTS — Specifies the number of nonzero matrix coefficients in order for MINOS to reserve the appropriate space in its data structures when reading the problem. "ELEMENTS" should be specified as the number of nonzero matrix coefficients in the core problem or greater.

Example

The following example represents typical specifications for running DECIS with MINOS as the optimizer.

```
BEGIN SPECS
PRINT LEVEL                1
LOG FREQUENCY             10
SUMMARY FREQUENCY         10
MPS FILE                  12
ROWS                   20000
COLUMNS                50000
ELEMENTS              100000
ITERATIONS LIMIT       30000
*
FACTORIZATION FREQUENCY  100
AIJ TOLERANCE            0.0
*
SCALE                     NO
END OF SPECS
```

2.5.4 Setting CPLEX Parameters Using System Environment Variables

When you use CPLEX as the optimizer for solving the master and the subproblems, optimization parameters must be specified through system environment variables. You can specify the parameters "CPLEXLICDIR", "SCALELP", "NOPRESOLVE", "ITERLOG", "OPTIMALITYTOL", "FEASIBILIITYTOL", and "DUALSIMPLEX".

CPLEXLICDIR — Contains the path to the CPLEX license directory. For example, on an Unix system with the CPLEX license directory in /usr/users/cplex/cplexlicdir you issue the command *setenv CPLEXLICDIR /usr/users/cplex/cplexlicdir*.

SCALELP — Specifies CPLEX to scale the master and subproblems before solving them. If the environment variable is not set no scaling is used. Setting the environment variable, e.g., by issuing the command *setenv SCALELP yes*, scaling is switched on.

NOPRESOLVE — Allows to switch off CPLEX's presolver. If the environment variable is not set, presolve will be used. Setting the environment variable, e.g., by setting *setenv NOPRESOLVE yes*, no presolve will be used.

ITERLOG — Specifies the iteration log of the CPLEX iterations to be printed to the file "MODEL.CPX". If you do not set the environment variable no iteration log will be printed. Setting the environment variable, e.g., by setting *setenv ITERLOG yes*, the CPLEX iteration log is printed.

OPTIMALITYTOL — Specifies the optimality tolerance for the CPLEX optimizer. If you do not set the environment variable the CPLEX default values are used. For example, setting *setenv OPTIMALITYTOL 1.0E-7* sets the CPLEX optimality tolerance to 0.0000001.

FEASIBILIITYTOL — Specifies the feasibility tolerance for the CPLEX optimizer. If you do not set the environment variable the CPLEX default values are used. For example, setting *setenv FEASIBILITYTOL 1.0E-7* sets the CPLEX optimality tolerance to 0.0000001.

DUALSIMPLEX — Specifies the dual simplex algorithm of CPLEX to be used. If the environment variable is not set the primal simplex algorithm will be used. This is the default and works beautifully for most problems. If the environment variable is set, e.g., by setting *setenv DUALSIMPLEX yes*, CPLEX uses the dual simplex algorithm for solving both master and subproblems.

2.6 GAMS/DECIS Output

After successfully having solved a problem, DECIS returns the objective, the optimal primal and optimal dual solution, the status of variables (if basic or not), and the status of equations (if binding or not) to GAMS. In the case of first-stage variables and equations you have all information in GAMS available as if you used any other solver, just instead of obtaining the optimal values for deterministic core problem you actually obtained the optimal values for the stochastic problem. However, for second-stage variables and constraints the expected values of the optimal primal and optimal dual solution are reported. This saves space and is useful for the calculation of

risk measures. However, the information as to what the optimal primal and dual solutions were in the different scenarios of the stochastic programs is not reported back to GAMS. In a next release of the GAMS/DECIS interface the GAMS language is planned to be extended to being able to handle the scenario second-stage optimal primal and dual values at least for selected variables and equations.

While running DECIS outputs important information about the progress of the execution to your computer screen. After successfully having solved a problem, DECIS also outputs its optimal solution into the solution output file "MODEL.SOL". The debug output file "MODEL.SCR" contains important information about the optimization run, and the optimizer output files "MODEL.MO" (when using DECIS with MINOS) or "MODEL.CPX" (when using DECIS with CPLEX) contain solution output from the optimizer used. In the DECIS User's Guide you find a detailed discussion of how to interpret the screen output, the solution report and the information in the output files.

2.6.1 The Screen Output

The output to the screen allows you to observe the progress in the execution of a DECIS run. After the program logo and the copyright statement, you see four columns of output beeing written to the screen as long as the program proceeds. The first column (from left to right) represents the iteration count, the second column the lower bound (the optimal objective of the master problem), the third column the best upper bound (exact value or estimate of the total expected cost of the best solution found so far), and the fourth column the current upper bound (exact value or estimate of the total expected cost of current solution). After successful completion, DECIS quits with "Normal Exit", otherwise, if an error has been encountered, the programs stops with the message "Error Exit".

Example

When solving the illustrative example APL1P using strategy 5, we obtain the following report on the screen:

```
 T H E   D  E  C  I  S   S Y S T E M
 Copyright (c) 1989  -- 1999 by Dr. Gerd Infanger
 All rights reserved.

iter                lower           best upper        current upper

   0          -0.9935E+06
   1          -0.4626E+06           0.2590E+05           0.2590E+05
   2           0.2111E+05           0.2590E+05           0.5487E+06
   3           0.2170E+05           0.2590E+05           0.2697E+05
   4           0.2368E+05           0.2384E+05           0.2384E+05
   5           0.2370E+05           0.2384E+05           0.2401E+05
   6           0.2370E+05           0.2370E+05           0.2370E+05

iter                lower           best upper        current upper

   6           0.2370E+05
   7           0.2403E+05           0.2470E+05           0.2470E+05
   8           0.2433E+05           0.2470E+05           0.2694E+05
   9           0.2441E+05           0.2470E+05           0.2602E+05
  10           0.2453E+05           0.2470E+05           0.2499E+05
  11           0.2455E+05           0.2470E+05           0.2483E+05
  12           0.2461E+05           0.2467E+05           0.2467E+05
  13           0.2461E+05           0.2467E+05           0.2469E+05
  14           0.2461E+05           0.2465E+05           0.2465E+05
  15           0.2463E+05           0.2465E+05           0.2467E+05
  16           0.2463E+05           0.2465E+05           0.2465E+05
  17           0.2464E+05           0.2465E+05           0.2465E+05
  18           0.2464E+05           0.2464E+05           0.2464E+05
  19           0.2464E+05           0.2464E+05           0.2464E+05
  20           0.2464E+05           0.2464E+05           0.2464E+05
  21           0.2464E+05           0.2464E+05           0.2464E+05
  22           0.2464E+05           0.2464E+05           0.2464E+05

Normal Exit
```

2.6.2 The Solution Output File

The solution output file contains the solution report from the DECIS run. Its name is "MODEL.SOL". The file contains the best objective function value found, the corresponding values of the first-stage variables, the corresponding optimal second-stage cost, and a lower and an upper bound on the optimal objective of the problem. In addition, the number of universe scenarios and the settings for the stopping tolerance are reported. In the case of using a deterministic strategy for solving the problem, exact values are reported. When using Monte Carlo sampling, estimated values, their variances, and the sample size used for the estimation are reported. Instead of exact upper and lower bounds, probabilistic upper and lower bounds, and a 95% confidence interval, within which the true optimal solution lies with 95% confidence, are reported. A detailed description of the solution output file can be found in the DECIS User's Guide.

2.6.3 The Debug Output File

The debug output file contains the standard output of a run of DECIS containing important information about the problem, its parameters, and its solution. It also contains any error messages that may occur during a run of DECIS. In the case that DECIS does not complete a run successfully, the cause of the trouble can usually be located using the information in the debug output file. If the standard output does not give enough information you can set the debug parameter ibug in the parameter input file to a higher value and obtain additional debug output. A detailed description of the debug output file can be found in the DECIS User's Guide.

2.6.4 The Optimizer Output Files

The optimizer output file "MODEL.MO" contains all the output from MINOS when called as a subroutine by DECIS. You can specify what degree of detail should be outputted by setting the appropriate "PRINT LEVEL" in the MINOS specification file. The optimizer output file "MODEL.CPX" reports messages and the iteration log (if switchwd on using the environment variable) from CPLEX when solving master and sub problems.

A GAMS/DECIS Illustrative Examples

A.1 Example APL1P

```
*   APL1P test model
*   Dr. Gerd Infanger, November 1997

set g generators /g1, g2/;
set dl demand levels /h, m, l/;

parameter alpha(g) availability / g1  0.68,  g2  0.64 /;
parameter ccmin(g) min capacity / g1  1000,  g2  1000 /;
parameter ccmax(g) max capacity / g1 10000,  g2 10000 /;
parameter c(g) investment       / g1   4.0,  g2   2.5 /;

table f(g,dl) operating cost
             h    m    l
   g1        4.3  2.0  0.5
   g2        8.7  4.0  1.0;

parameter d(dl) demand                   / h  1040, m  1040, l  1040 /;
parameter us(dl) cost of unserved demand / h    10, m    10, l    10 /;

free variable tcost            total cost;
positive variable x(g)         capacity of generators;
positive variable y(g, dl)     operating level;
positive variable s(dl)        unserved demand;

equations
cost        total cost
cmin(g)     minimum capacity
cmax(g)     maximum capacity
omax(g)     maximum operating level
demand(dl)  satisfy demand;

cost .. tcost =e=     sum(g, c(g)*x(g))
                    + sum(g, sum(dl, f(g,dl)*y(g,dl)))
                    + sum(dl,us(dl)*s(dl));

cmin(g) ..    x(g) =g= ccmin(g);
cmax(g) ..    x(g) =l= ccmax(g);
omax(g) ..    sum(dl, y(g,dl)) =l= alpha(g)*x(g);
demand(dl) .. sum(g, y(g,dl)) + s(dl) =g= d(dl);

model apl1p /all/;

* setting decision stages
x.stage(g)       = 1;
y.stage(g, dl)   = 2;
s.stage(dl)      = 2;
cmin.stage(g)    = 1;
cmax.stage(g)    = 1;
omax.stage(g)    = 2;
demand.stage(dl) = 2;

* defining independent stochastic parameters
set stoch /out, pro /;

set omega1 / o11, o12, o13, o14 /;
table v1(stoch, omega1)
      o11   o12   o13   o14
out  -1.0  -0.9  -0.5  -0.1
pro   0.2   0.3   0.4   0.1
;

set omega2 / o21, o22, o23, o24, o25 /;
table v2(stoch, omega2)
      o21   o22   o23   o24   o25
out  -1.0  -0.9  -0.7  -0.1  -0.0
```

```
pro   0.1   0.2   0.5   0.1   0.1
;


set omega3 / o31, o32, o33, o34 /;
table v3(stoch, omega1)
      o11   o12   o13   o14
out   900  1000  1100  1200
pro  0.15  0.45  0.25  0.15
;


set omega4 / o41, o42, o43, o44 /;
table v4(stoch,omega1)
      o11   o12   o13   o14
out   900  1000  1100  1200
pro  0.15  0.45  0.25  0.15
;

set omega5 / o51, o52, o53, o54 /;
table v5(stoch,omega1)
      o11   o12   o13   o14
out   900  1000  1100  1200
pro  0.15  0.45  0.25  0.15
;

* defining distributions
file stg /MODEL.STG/;
put stg;
put "INDEP DISCRETE" /;
loop(omega1,
put "x g1  omax g1   ", v1("out", omega1), " period2 ", v1("pro", omega1) /;
);
put "*" /;
loop(omega2,
put "x g2  omax g2   ", v2("out", omega2), " period2 ", v2("pro", omega2) /;
);
put "*" /;
loop(omega3,
put "RHS demand h   ", v3("out", omega3),  " period2 ", v3("pro", omega3) /;
);
put "*" /;
loop(omega4,
put "RHS demand m   ", v4("out", omega4),  " period2 ", v4("pro", omega4) /;
);
put "*" /;
loop(omega5,
put "RHS demand l   ", v5("out", omega5),  " period2 ", v5("pro", omega5) /;
);
putclose stg;


* setting DECIS as optimizer
* DECISM uses MINOS, DECISC uses CPLEX
option lp=decism;
apl1p.optfile = 1;

solve apl1p using lp minimizing tcost;

scalar ccost capital cost;
scalar ocost operating cost;
ccost = sum(g, c(g) * x.l(g));
ocost = tcost.l -  ccost;
display x.l, tcost.l, ccost, ocost, y.l, s.l;
```

A.2 Example APL1PCA

```
*   APL1PCA test model
*   Dr. Gerd Infanger, November 1997

set g generators /g1, g2/;
set dl demand levels /h, m, l/;

parameter alpha(g) availability / g1  0.68,  g2  0.64 /;
parameter ccmin(g) min capacity / g1  1000,  g2  1000 /;
parameter ccmax(g) max capacity / g1 10000,  g2 10000 /;
parameter c(g) investment       / g1   4.0,  g2   2.5 /;

table f(g,dl) operating cost
             h    m    l
   g1        4.3  2.0  0.5
   g2        8.7  4.0  1.0;

parameter d(dl) demand                   / h  1040, m  1040, l  1040 /;
parameter us(dl) cost of unserved demand / h    10, m    10, l    10 /;

free variable tcost           total cost;
positive variable x(g)        capacity of generators;
positive variable y(g, dl)    operating level;
positive variable s(dl)       unserved demand;

equations
cost        total cost
cmin(g)     minimum capacity
cmax(g)     maximum capacity
omax(g)     maximum operating level
demand(dl)  satisfy demand;

cost .. tcost =e=     sum(g, c(g)*x(g))
                    + sum(g, sum(dl, f(g,dl)*y(g,dl)))
                    + sum(dl,us(dl)*s(dl));

cmin(g) ..    x(g) =g= ccmin(g);
cmax(g) ..    x(g) =l= ccmax(g);
omax(g) ..    sum(dl, y(g,dl)) =l= alpha(g)*x(g);
demand(dl) .. sum(g, y(g,dl)) + s(dl) =g= d(dl);

model apl1p /all/;

* setting decision stages
x.stage(g)       = 1;
y.stage(g, dl)   = 2;
s.stage(dl)      = 2;
cmin.stage(g)    = 1;
cmax.stage(g)    = 1;
omax.stage(g)    = 2;
demand.stage(dl) = 2;


* defining independent stochastic parameters
set stoch /out, pro/;

set omega1 / o11, o12 /;
table v1(stoch,omega1)
       o11  o12
out    2.1  1.0
pro    0.5  0.5 ;

set omega2 / o21, o22 /;
table v2(stoch, omega2)
       o21  o22
out    2.0  1.0
pro    0.2  0.8 ;

parameter hm1(dl) / h  300., m  400., l  200. /;
```

```
parameter hm2(dl) / h  100., m  150., l  300. /;

* defining distributions (writing file MODEL.STG)
file stg / MODEL.STG /;
put stg;

put "BLOCKS DISCRETE" /;
scalar h1;
loop(omega1,
put "BL  v1   period2 ", v1("pro", omega1)/;
loop(dl,
h1 = hm1(dl) * v1("out", omega1);
put "RHS demand ", dl.tl:1, " ", h1/;
);
);
loop(omega2,
put " BL v2   period2 ", v2("pro", omega2) /;
loop(dl,
h1 = hm2(dl) * v2("out", omega2);
put "RHS demand ", dl.tl:1, " ", h1/;
);
);
putclose stg;

* setting DECIS as optimizer
* DECISM uses MINOS, DECISC uses CPLEX
option lp=decism;
apl1p.optfile = 1;

solve apl1p using lp minimizing tcost;

scalar ccost capital cost;
scalar ocost operating cost;
ccost = sum(g, c(g) * x.l(g));
ocost = tcost.l -  ccost;
display x.l, tcost.l, ccost, ocost, y.l, s.l;
```

B Error Messages

1. ERROR in MODEL.STO: kwd, word1, word2 was not matched in first realization of block
   The specification of the stochastic parameters is incorrect. The stochastic parameter has not been specified in the specification of the first outcome of the block. When specifying the first outcome of a block always include all stochastic parameters corresponding to the block.

2. Option word1 word2 not supported
   You specified an input distribution in the stochastic file that is not supported. Check the DECIS manual for supported distributions.

3. Error in time file
   The time file is not correct. Check the file MODEL.TIM. Check the DECIS manual for the form of the time file.

4. ERROR in MODEL.STO: stochastic RHS for objective, row name2
   The specification in the stochastic file is incorrect. You attempted to specify a stochastic right-hand side for the objective row (row name2). Check file MODEL.STO.

5. ERROR in MODEL.STO: stochastic RHS in master, row name2
   The specification in the stochastic file is incorrect. You attempted to specify a stochastic right-hand side for the master problem (row name2). Check file MODEL.STO.

6. ERROR in MODEL.STO: col not found, name1
   The specification in the stochastic file is incorrect. The entry in the stochastic file, name1, is not found in the core file. Check file MODEL.STO.

7. ERROR in MODEL.STO: invalid col/row combination, (name1/name2)
   The stochastic file (MODEL.STO) contains an incorrect specification.

8. ERROR in MODEL.STO: no nonzero found (in B or D matrix) for col/row (name1, name2)
   There is no nonzero entry for the combination of name1 (col) and name2(row) in the B-matrix or in the D-matrix. Check the corresponding entry in the stochastic file (MODEL.STO). You may want to include a nonzero coefficient for (col/row) in the core file (MODEL.COR).

9. ERROR in MODEL.STO: col not found, name2
   The column name you specified in the stochastic file (MODEL.STO) does not exist in the core file (MODEL.COR). Check the file MODEL.STO.

10. ERROR in MODEL.STO: stochastic bound in master, col name2
    You specified a stochastic bound on first-stage variable name2. Check file MODEL.STO.

11. ERROR in MODEL.STO: invalid bound type (kwd) for col name2
    The bound type, kwd, you specified is invalid. Check file MODEL.STO.

12. ERROR in MODEL.STO: row not found, name2
    The specification in the stochastic file is incorrect. The row name, name2, does not exist in the core file. Check file MODEL.STO.

13. ERROR: problem infeasible
    The problem solved (master- or subproblem) turned out to be infeasible. If a subproblem is infeasible, you did not specify the problem as having the property of "complete recourse". Complete recourse means that whatever first-stage decision is passed to a subproblem, the subproblem will have a feasible solution. It is the best way to specify a problem, especially if you use a sampling based solution strategy. If DECIS encounters a feasible subproblem, it adds a feasibility cut and continues the execution. If DECIS encounters an infeasible master problem, the problem you specified is infeasible, and DECIS terminates. Check the problem formulation.

14. ERROR: problem unbounded
    The problem solved (master- or subproblem) turned out to be unbounded. Check the problem formulation.

15. ERROR: error code: inform
    The solver returned with an error code from solving the problem (master- or subproblem). Consult the users' manual of the solver (MINOS or CPLEX) for the meaning of the error code, inform. Check the problem formulation.

16. ERROR: while reading SPECS file
    The MINOS specification file (MINOS.SPC) containes an error. Check the specification file. Consult the MINOS user's manual.

17. ERROR: reading mps file, mpsfile
    The core file mpsfile (i.e., MODEL.COR) is incorect. Consult the DECIS manual for instructions regarding the MPS format.

18. ERROR: row 1 of problem ip is not a free row
    The first row of the problem is not a free row (i.e., is not the objective row). In order to make the fist row a free row, set the row type to be 'N'. Consult the DECIS manual for the MPS specification of the problem.

19. ERROR: name not found = nam1, nam2
    There is an error in the core file (MODEL.COR). The problem cannot be decomposed correctly. Check the core file and check the model formulation.

20. ERROR: matrix not in staircase form
    The constraint matrix of the problem as specified in core file (MODEL.COR) is not in staircase form. The first-stage rows and columns and the second-stage rows and columns are mixed within each other. Check the DECIS manual as to how to specify the core file. Check the core file and change the order of rows and columns.

DECIS References

[1] Brooke, A., Kendrik, D. and Meeraus, A. (1988): *GAMS, A Users Guide*, The Scientific Press, South San Francisco, California.

[2] CPLEX Optimization, Inc. (1989–1997): *Using the CPLEX Callable Library*, 930 Tahoe Blvd. Bldg. 802, Suite 279, Incline Village, NV 89451, USA.

[3] Infanger, G. (1994): *Planning Under Uncertainty – Solving Large-Scale Stochastic Linear Programs*, The Scientific Press Series, Boyd and Fraser.

[4] Infanger, G. (1997): *DECIS User's Guide*, Dr. Gerd Infanger, 1590 Escondido Way, Belmont, CA 94002.

[5] Murtagh, B.A. and Saunders, M.A. (1983): MINOS User's Guide, SOL 83-20, Department of Operations Research, Stanford University, Stanford CA 94305.

DECIS License and Warranty

The software, which accompanies this license (the "Software") is the property of Gerd Infanger and is protected by copyright law. While Gerd Infanger continues to own the Software, you will have certain rights to use the Software after your acceptance of this license. Except as may be modified by a license addendum, which accompanies this license, your rights and obligations with respect to the use of this Software are as follows:

- You may
  1. Use one copy of the Software on a single computer,
  2. Make one copy of the Software for archival purposes, or copy the software onto the hard disk of your computer and retain the original for archival purposes,
  3. Use the Software on a network, provided that you have a licensed copy of the Software for each computer that can access the Software over that network,
  4. After a written notice to Gerd Infanger, transfer the Software on a permanent basis to another person or entity, provided that you retain no copies of the Software and the transferee agrees to the terms of this agreement.
- You may not
  1. Copy the documentation, which accompanies the Software,
  2. Sublicense, rent or lease any portion of the Software,
  3. Reverse engineer, de-compile, disassemble, modify, translate, make any attempt to discover the source code of the Software, or create derivative works from the Software.

Limited Warranty:

Gerd Infanger warrants that the media on which the Software is distributed will he free from defects for a period of thirty (30) days from the date of delivery of the Software to you. Your sole remedy in the event of a breach of the warranty will be that Gerd Infanger will, at his option, replace any defective media returned to Gerd Infanger within the warranty period or refund the money you paid for the Software. Gerd Infanger does not warrant that the Software will meet your requirements or that operation of the Software will be uninterrupted or that the Software will be error-free.

THE ABOVE WARRANTY IS EXCLUSIVE AND IN LIEU OF ALL OTHER WARRANTIES, WHETHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.

Disclaimer of Damages:

REGARDLESS OF WHETHER ANY REMEDY SET FORTH HEREIN FAILS OF ITS ESSENTIAL PURPOSE, IN NO EVENT WILL GERD INFANGER BE LIABLE TO YOU FOR ANY SPECIAL, CONSEQUENTIAL, INDIRECT OR SIMILAR DAMAGES, INCLUDING ANY LOST PROFITS OR LOST DATA ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF GERD INFANGER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

IN NO CASE SHALL GERD INFANGER'S LIABILITY EXCEED THE PURCHASE PRICE FOR THE SOFTWARE. The disclaimers and limitations set forth above will apply regardless of whether you accept the Software.

General:

This Agreement will be governed by the laws of the State of California. This Agreement may only be modified by a license addendum, which accompanies this license or by a written document, which has been signed by both you and Gerd Infanger. Should you have any questions concerning this Agreement, or if you desire to contact Gerd Infanger for any reason, please write:

Gerd Infanger, 1590 Escondido Way, Belmont, CA 94002, USA.