**AIMMS Language Reference - Mixed Complementarity Problems**

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com.

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
SOHO Fuxing Plaza No.388
Building D-71, Level 3
Madang Road, Huangpu District
Shanghai 200025
China
Tel.: ++86 21 5309 8733

Email: info@aimms.com
WWW: www.aimms.com

# Chapter 23

# Mixed Complementarity Problems

*This chapter*

This chapter discusses the special identifier types and language constructs that AIMMS offers to allow you to formulate mixed complementarity problems. Although mixed complementarity problems do not involve optimization, they are specified through variables which are linked to constraints in terms of these variables, and thus fall into the common framework of a `Mathematical-Program`. AIMMS also supports nonlinear optimization problems with additional complementarity constraints (also known as MPCC or MPEC problems)

## 23.1 Complementarity problems

*Complementarity problems*

Complementarity relations arise in a variety of engineering and economics applications, most commonly to express an equilibrium of quantities such as forces or prices. One standard application in engineering arises in contact mechanics, where complementarity expresses the fact that friction occurs only when two bodies are in contact. Other applications are found in structural mechanics, structural design, traffic equilibrium and optimal control.

*Economic models*

Interest among economists in solving complementarity problems is due in part to increased use of computational general equilibrium models, where for instance complementarity is used to express Walras' Law, and in part to the equivalence of various games to complementarity problems.

*Nonlinear optimization*

Some generalizations of nonlinear programming, such as multi-level optimization—in which auxiliary objectives are to be minimized—may be reformulated as problems with complementarity conditions. Also, by formulating the Kuhn-Tucker conditions of a nonlinear optimization model one obtains a complementarity problem, which could be solved by a complementarity solver. In the latter case, however, one requires second-order derivative information of all constraints in the original optimization model.

*Complementarity conditions*

Complementarity problems are, in general, systems of nonlinear constraints where variables in the system are linked to constraints in the form of complementarity conditions. There are two forms of complementarity conditions, the classical complementarity condition, and its generalization, the mixed complementarity condition.

*Classical complementarity conditions*

The classical form of a complementarity condition involves a nonnegative variable $x_i$ and an associated function $f_i(x)$. It requires that

$$\begin{aligned} x_i = 0 \quad &\text{and} \quad f_i(x) \geq 0, \text{ or} \\ x_i > 0 \quad &\text{and} \quad f_i(x) = 0. \end{aligned}$$

This condition states that of both inequalities, at least one must reach its bound. Alternatively, one can formulate this complementarity condition as $x_i \geq 0$, $f_i(x) \geq 0$ and $x_i \cdot f_i(x) = 0$.

*Mixed complementarity condition*

The mixed form of a complementarity condition involves a bounded variable $l_i \leq x_i \leq u_i$ with an associated function $f_i(x)$. It requires that

$$\begin{aligned} x_i = l_i \quad &\text{and} \quad f_i(x) \geq 0, \text{ or} \\ x_i = u_i \quad &\text{and} \quad f_i(x) \leq 0, \text{ or} \\ l_i < x_i < u_i \quad &\text{and} \quad f_i(x) = 0. \end{aligned}$$

This condition states that either $x_i$ must reach one of its bounds, or the function $f_i(x)$ must be zero. A mixed complementarity condition can be split into two classical complementarity conditions (albeit by introducing auxiliary variables). The classical complementarity condition, on the other hand, is a special case of the mixed complementarity condition by choosing $l_i = 0$ and $u_i = \infty$.

*Special cases*

By choosing $l_i = -\infty$ and $u_i = \infty$, the mixed complementarity condition reduces to the special case

$$x_i \text{ is “free”} \quad \text{and} \quad f_i(x) = 0.$$

Another special case is obtained when $l_i = u_i$. The mixed complementarity condition then reduces to

$$x_i = l_i \quad \text{and} \quad f_i(x) \text{ is “free”}$$

*Supported complementarity conditions in* AIMMS

All complementarity conditions described above can be represented by associating a variable with a single constraint, which will form the basis for representing complementarity conditions in AIMMS. Consider the inequalities

$$l_{x_i} \leq x_i \leq u_{x_i} \quad \text{and} \quad l_{f_i} \leq f_i(x) \leq u_{f_i}$$

where $f_i(x)$ is a nonlinear expression, and *exactly* two of the constants $l_{x_i}$, $u_{x_i}$, $l_{f_i}$ and $u_{f_i}$ are finite. The six possible cases are enumerated in Table 23.1, and are discussed below.

| **Case** | $l_{x_i}$ | $u_{x_i}$ | $l_{f_i}$ | $u_{f_i}$ |
|---|---|---|---|---|
| 1 | finite | finite | $-\infty$ | $\infty$ |
| 2 | finite | $\infty$ | finite | $\infty$ |
| 3 | finite | $\infty$ | $-\infty$ | finite |
| 4 | $-\infty$ | finite | finite | $\infty$ |
| 5 | $-\infty$ | finite | $-\infty$ | finite |
| 6 | $-\infty$ | $\infty$ | finite | finite |

Table 23.1: Six allowed cases with exactly two finite bounds

*Case 1*

The case $l_{x_i} \le x_i \le u_{x_i}$ and $-\infty \le f_i(x) \le \infty$ corresponds to the mixed complementarity condition already discussed above:

$$\begin{aligned} x_i = l_{x_i} \quad &\text{and} \quad f_i(x) \ge 0, \text{ or} \\ x_i = u_{x_i} \quad &\text{and} \quad f_i(x) \le 0, \text{ or} \\ l_{x_i} < x_i < u_{x_i} \quad &\text{and} \quad f_i(x) = 0. \end{aligned}$$

*Case 2*

The case $l_{x_i} \le x_i \le \infty$ and $l_{f_i} \le f_i(x) \le \infty$ corresponds to the classical complementarity condition

$$\begin{aligned} \hat{x}_i = 0 \quad &\text{and} \quad \hat{f}_i(x) \ge 0, \text{ or} \\ \hat{x}_i > 0 \quad &\text{and} \quad \hat{f}_i(x) = 0. \end{aligned}$$

where $\hat{x}_i = x_i - l_{x_i}$ and $\hat{f}_i(x) = f_i(x) - l_{f_i}$.

*Case 3*

The case $l_{x_i} \le x_i \le \infty$ and $-\infty \le f_i(x) \le u_{f_i}$ corresponds to the classical complementarity condition

$$\begin{aligned} \hat{x}_i = 0 \quad &\text{and} \quad \hat{f}_i(x) \ge 0, \text{ or} \\ \hat{x}_i > 0 \quad &\text{and} \quad \hat{f}_i(x) = 0. \end{aligned}$$

where $\hat{x}_i = x_i - l_{x_i}$ and $\hat{f}_i(x) = u_{f_i} - f_i(x)$.

*Case 4*

The case $-\infty \le x_i \le u_{x_i}$ and $l_{f_i} \le f_i(x) \le \infty$ corresponds to the classical complementarity condition

$$\begin{aligned} \hat{x}_i = 0 \quad &\text{and} \quad \hat{f}_i(x) \ge 0, \text{ or} \\ \hat{x}_i > 0 \quad &\text{and} \quad \hat{f}_i(x) = 0. \end{aligned}$$

where $\hat{x}_i = u_{x_i} - x_i$ and $\hat{f}_i(x) = f_i(x) - l_{f_i}$.

*Case 5*

The case $-\infty \le x_i \le u_{x_i}$ and $-\infty \le f_i(x) \le u_{f_i}$ corresponds to the classical complementarity condition

$$\begin{aligned} \hat{x}_i = 0 \quad &\text{and} \quad \hat{f}_i(x) \ge 0, \text{ or} \\ \hat{x}_i > 0 \quad &\text{and} \quad \hat{f}_i(x) = 0. \end{aligned}$$

where $\hat{x}_i = u_{x_i} - x_i$ and $\hat{f}_i(x) = u_{f_i} - f_i(x)$.

*Case 6: $l_{f_i} = u_{f_i}$*

The case $-\infty \le x_i \le \infty$ and $l_{f_i} \le f_i(x) \le u_{f_i}$ with $l_{f_i} = u_{f_i}$ corresponds to the first special case of the mixed complementarity condition

$$x_i \text{ is “free”} \quad \text{and} \quad \hat{f}_i(x) = 0.$$

where $\hat{f}(x) = f_i(x) - l_{f_i}$.

*Case 6: $l_{f_i} < u_{f_i}$*

After the introduction of variables $x_i^+, x_i^- \ge 0$ and functions

$$\begin{aligned} f_i^x(x) &= x_i - x_i^+ - x_i^- \\ f_i^+(x) &= f_i(x) - l_{f_i} \\ f_i^-(x) &= u_{f_i} - f_i(x) \end{aligned}$$

the case $-\infty \le x_i \le \infty$ and $l_{f_i} \le f_i(x) \le u_{f_i}$ with $l_{f_i} < u_{f_i}$ corresponds to a system of three simultaneous complementarity conditions

$$x_i \text{ is “free”} \quad \text{and} \quad f_i^x(x) = 0$$

$$\begin{aligned} x_i^+ = 0 \quad &\text{and} \quad f_i^+(x) \ge 0, \text{ or} \\ x_i^+ > 0 \quad &\text{and} \quad f_i^+(x) = 0 \end{aligned}$$

$$\begin{aligned} x_i^- = 0 \quad &\text{and} \quad f_i^-(x) \ge 0, \text{ or} \\ x_i^- > 0 \quad &\text{and} \quad f_i^-(x) = 0. \end{aligned}$$

AIMMS *support*

AIMMS supports the variable-constraint couples with two finite bounds, as discussed above, through the special `ComplementaryVariable` data type. The declaration and attributes of this data type are discussed in the next section, while section 23.3 describes the declaration of mixed complementarity models through the common `MathematicalProgram` declaration.

*Well-behaved systems*

Like with nonlinear optimization models, not all mixed complementarity systems that can be formulated are well-behaved. For instance, a variable $x \ge 0$ with an associated constraint $1 - x \ge 0$, only admits the solutions 0 and 1, which would destroy the continuous character of complementarity problems. For systems of complementarity conditions that are not well-behaved, the solution process may produce no, or unexpected results.

## 23.2 ComplementaryVariable declaration and attributes

*Complementarity variables*

To support you in formulating a complementarity model, AIMMS provides a special type of variable, the ComplementaryVariable. The attributes of a complementarity variable allow you to declare an (indexed) class of variables in a complementarity model along with their associated constraints. The attributes of a ComplementaryVariable are listed in Table 23.2.

*Automatic sanity checks*

By construction, this new variable type automatically ensures that every variable in a complementarity model is associated with a single constraint. Also, when AIMMS detects that the total number of (finite) bounds on both the complementarity variable and its associated constraint is not equal to two (as required above), a compilation error will result. Thus, ComplementaryVariable will help to reduce the most common declaration errors for this type of model.

| Attribute | Value-type | See also page |
|---|---|---|
| IndexDomain | *index-domain* | 44, 212, 212 |
| Range | *range* | 212 |
| Unit | *unit-valued expression* | 47, 215 |
| Text | *string* | 19, 48 |
| Comment | *comment string* | 19 |
| Complement | *expression* | 221 |
| NonvarStatus | *reference* | 216 |
| Property | NoSave, Complement | |

Table 23.2: ComplementaryVariable attributes

*The IndexDomain attribute*

Through the IndexDomain attribute of a complementarity variable you can specify domain of tuples for which you want AIMMS to generate a variable and its associated constraint. During generation, AIMMS will only generate a variable for all tuples that satisfy all domain restrictions that you have imposed on the domain.

*The Range attribute*

In the Range attribute you can specify the lower and upper bound of a complementarity variable, in a similar manner for ordinary Variables (see also Section 14.1). During generation, AIMMS will perform a runtime check, for every individual tuple in the index domain, whether the number of finite bounds specified here, plus the number of finite bounds in the constraint specified in the Complement attribute, exactly equals two.

*The Complement attribute*

The Complement attribute allows you to specify the constraint that must be associated with the complementarity variable at hand. With $f(x,\dots)$ a general nonlinear function, the following types of expressions are allowed

- $f(x,\dots) \geq a$ (variable must have a single-sided Range),
- $f(x,\dots) \leq a$ (variable must have a single-sided Range),
- $a \leq f(x,\dots) \leq b$ (variable must be free),
- $f(x,\dots) = a$ (variable must be free), or
- $f(x,\dots)$ (variable must be bounded).

In addition, the Complement attribute can refer to an existing Constraint in your model, which then should hold a definition as one of the cases above. The Complement attribute can also hold a scalar element parameter into the set AllConstraints, which offers the possibility to assign different constraints to the complementarity variable in sequential solves.

*Constraint listing*

In the constraint listing, the constraints associated with a complementarity variable will be listed with a generated name consisting of the name of the ComplementarityVariable with an additional suffix "_complement".

*The NonvarStatus attribute*

With the NonvarStatus attribute you can indicate for which tuples you want AIMMS to consider the complementarity variable as a parameter, i.e. with the lower and upper bound set equal to the level value prior to solving the model (see also Section 14.1.1). From the mixed complementarity condition it follows that the function in the corresponding constraint is then allowed to assume arbitrary values, whence there is no strict need to generate the variable and constraint for the solver.

*Positive and negative values*

The value of the NonvarStatus attribute must be an expression in some or all of the indices in the index list of the variable, allowing you to change the nonvariable status of individual elements or groups of elements at once. When the NonvarStatus assumes a positive value, AIMMS will not generate the variable and its associated constraint. For negative values, the variable and constraint will be generated, but reduces to the second special case of the mixed complementarity condition

$$\hat{x}_i = x_i - x_i^0 = 0 \quad \text{and} \quad f_i(x) \text{ is "free"},$$

i.e. the function in the constraint will be allowed to assume arbitrary values.

*The Unit attribute*

Providing a Unit for a complementarity variable will help you in a number of ways.

- AIMMS will help you to check the consistency of all the constraints and assignments in your model (including the expression in the Complement attribute), and
- AIMMS will use the units to scale the model that is sent to the solver.

Proper scaling of a model will generally result in a more accurate and robust solution process. You can find more information on the definition and use of units to scale mathematical programs in Chapter 32.

*The Property attribute*

Complementarity variables support the properties NoSave and Complement. With the property NoSave you indicate that you do not want to store data associated with this variable in a case. The Complement property indicates that you are interested in the level values of the constraint defined in the Complement attribute. When this property is set, AIMMS will make the level value of this constraint available through the .Complement suffix of the complementarity variable at hand.

*Example*

The declaration of the complementarity variable MembraneHeight expresses a complementarity condition for the height of a membrane in a rectangular $(x, y)$-grid, with a uniform external force acting on each cell in the grid.

```
ComplementaryVariable MembraneHeight {
    IndexDomain  : (x,y);
    Range        : [MembraneLowerBound(x,y), MembraneUpperBound(x,y)];
    Complement   : {
        4*MembraneHeight(x,y)
        - MembraneHeight(x+1,y) - MembraneHeight(x-1,y)
        - MembraneHeight(x,y+1) - MembraneHeight(x,y-1)
        - CellForce
    }
}
```

The complementarity condition expresses that either the membrane reaches one its given bounds (for instance, an obstacle placed in the way of the membrane), or the external force on the cell must be equal to the internal forces acting on the cell caused by differences in height with neighboring cells.

## 23.3 Declaration of mixed complementarity models

*Mixed complementarity models*

To define a pure mixed complementarity model, you must declare a MathematicalProgram (see also Section 15.1) and specify mcp as the Type attribute of the MathematicalProgram. In the Variables attribute you can specify a subset of the set of all ComplementarityVariables to be included in the mixed complementarity model at hand. Based on this specification, AIMMS will automatically generate all constraints associated with these complementarity variables, resulting in a square system.

*Additional variables and constraints*

In addition, AIMMS allows you to add ordinary variables to the Variables attribute, and to specify additional constraints in the Constraints attribute of the MathematicalProgram that must be satisfied as well. If the solver used to solve the mixed complementarity model requires a square system, AIMMS will automatically add auxiliary constraints or variables to the generated system, and provide the linkages with the ordinary variables and constraints you have added to the system.

*No optimization*

For a mixed complementarity problem you should not specify the Objective and Direction attributes, as a complementarity solver will only compute a feasible solution that satisfies all the complementarity conditions specified. If these attributes are not empty, AIMMS will produce a runtime error when you apply the SOLVE statement the corresponding MathematicalProgram (see also Section 15.3).

*Example*

A mixed complementarity model containing the declaration of the complementarity variable MembraneHeight declared in the previous section, is defined by the following declaration.

```
MathematicalProgram Membrane {
    Variables   : AllVariables;
    Type        : mcp;
}
```

As usual, you can solve the Membrane through the statement

```
solve Membrane;
```

which will generate the mixed complementarity model and invoke a suitable solver for mcp problem type.

## 23.4 Declaration of MPCC models

*MPCC models*

Through the KNITRO solver, AIMMS also supports mathematical programs with complementarity constraints (MPCC models). MPCC models are also more commonly denoted by other modeling languages as MPEC models, which form a more general, and much more difficult, class of optimization problems. A MPCC model is an ordinary NLP model with additional complementarity constraints that have to be satisfied.

*Declaring MPCC models*

To define a MPCC model, you must declare a MathematicalProgram (see also Section 15.1) and specify mpcc as the Type attribute of the MathematicalProgram. The variable set of a MathematicalProgram of a MPCC model can contain ordinary variables as well as complementarity variables. Contrary to pure mixed complementary models, a MPCC model has an objective function.

*Solving MPCC models*

To solve MPCC models in AIMMS, you need a license for the KNITRO solver. If you do not have a license for the KNITRO solver, AIMMS will return an error that it has no suitable solver available for the mpcc class, whenever you try to solve a MPCC model. The KNITRO solver can also be used for solving pure mixed complementarity problems, but is, in general, far less efficient in that case than dedicated mcp solvers.