# ModAEM Users Guide
# Version 1.4pre2

Phil DiLavore
Vic Kelson
WHPA Inc.

August 10, 2001

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Supported Analytic Elements

This section will describe the analytic elements supported by ModAEM.

# Chapter 2

# Package Description - Analytic Elements

This chapter will describe analytic element modeling with ModAEM.

# Chapter 3

# Input Formats

## 3.1 Input Files

This section describes the files that are required as input for ModAEM.

### 3.1.1 MODAEM.NAM

This file is used by ModAEM to retrieve the name of the file which contains the input which describes the analytic element model. At this time, the file name ’modaem.nam’ is hard coded into ModAEM. modaem.nam is a flat text file that can be created with any text editor. modaem.nam must have a single line of text which specifies the name of the AEM input file. For example, if the model input data are contained in a file called ’modaem.aem’, then the contents of modaem.nam would be:

```
modaem
```

The extension ’.aem’ is appended to the file name found in modaem.nam.

## 3.2 AEM input file

The AEM input file is used to input the model elements and commands to ModAEM. The AEM input file can have any name (specified in the modaem.nam file) with the extension ’.aem’. The AEM input file is a flat text file that can be created with any text editor. Commands are entered one to a line, with carriage-control/line-feed characters at the end of each line. The valid commands that can be entered are as follows.

### 3.2.1 Input commands

Input commands are entered into the AEM input file using any standard text editor. Some commands can be (and indeed must be) nested within other commands. For instance, commands which create elements must be nested within the ’aem’ command. It is recommended (though not required) that nested commands be indented. Commands which start modules (i.e. ’aem’) are ended with the ’end’ command. Commands are not case-sensitive - upper or lower case (or mixed case) can be used.

### 3.2.2 Commands which are common to all input modules

The following commands are available in all ModAEM input modules.

#### 3.2.2.1 Comments

Comment lines in the AEM input file start with a hash mark (#) in the first column. Comment lines are ignored. For example:

```
# This is a comment line
```

#### 3.2.2.2 END – Exit a module section

The END directive causes ModAEM to leave the module that it is currently in. For example, when in the WL0 module (which is started with the WL0 command, the END command signals the end of input for the element modules:

```
# the aem section is used to define the problem
aem
  # other module sections go here...
  wl0 10
```

```
    ... well data goes in here ...
    # end of well data
  end
  # end of aem data
end
# processing directives go here
```

#### 3.2.2.3 PCD – Set program behavior on runtime error detections

Sets the ’proceed’ flag for the error handler (useful for debugging input files). The standard behavior of ModAEM is to terminate on any error detected during execution. The PCD command takes a single logical argument:

```
# Allow execution to proceed on error
PCD t
# Terminate execution on error
PCD f
```

#### 3.2.2.4 DBG – enable debug code

The DBG command is used to turn code marked as ’debug’ code on or off during execution (useful for program debugging). The ModAEM code contains many assertions which test to ensure that the ModAEM’s internal data structures are

### 3.2.3 Top-level ModAEM commands

In ModAEM, each element module has its own ’Read’ routine, which populates the data structures associated with that module. The top–level ModAEM commands discussed here are used only when ModAEM is running as a stand-alone program.

#### 3.2.3.1 AEM – begin defining a model problem domain

The AEM command is used to enter the AEM_Read routine, in order to populate all elements associated with the model domain (see the AEM module discussion in Section 3.2.6).

Usage (this command takes no arguments):

```
aem
  ... put model definition commands here ...
end
```

#### 3.2.3.2 SOL – solve the model

After a model has been defined (using the AEM module input section), it must be solved prior to performing any analyses. ModAEM uses an iterative solution scheme — at each iteration, the solution is improved based on the previous iteration.

```
sol [number of iterations]
```

Example:

```
aem
  ... put model definition commands here ...
end
# solve using 4 iterations
sol 4
```

#### 3.2.3.3 HEA – report the head at a point in the model

Directs ModAEM to report the hydraulic head at a specified point. Note: a solution must be present prior to using this command. The command:

```
hea [z]
```

reports the head at the complex coordinate $z = x + iy$. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs. Thus, the command

```
HEA (100.0,100.0)
```

reports the head at the coordinate $(100, 100)$.

#### 3.2.3.4 POT – report the discharge potential

Directs ModAEM to report the discharge potential at a specified point. Note: a solution must be present prior to using this command (see the SOL command in Section 3.2.3.2). The command:

```
pot [z]
```

reports the head at the complex coordinate $z = x + iy$. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x,y)$ pairs. Thus, the command

```
pot (100.0,100.0)
```

reports the head at the coordinate $(100,100)$.

#### 3.2.3.5 GRA – report the numerical gradient (potential)

Directs ModAEM to report the numerical gradient (potential) at a specified point. Used in program debugging; the numerical gradient should have the same value as the total discharge (see the DIS command in 3.2.3.6). Note: a solution must be present prior to using this command (see th SOL command in Section 3.2.3.2). The command:

```
gra [z] delta
```

reports the gradient at the complex coordinate $z = x + iy$. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x,y)$ pairs. Thus, the command

```
gra (100.0,100.0) 1.0
```

reports the gradient at the coordinate $(100,100)$.

#### 3.2.3.6 DIS – report the complex discharge

Directs ModAEM to report the complex discharge at a specified point. Note: a solution must be present prior to using this command (see th SOL command in Section 3.2.3.2). The command:

```
dis [z]
```

reports the gradient at the complex coordinate $z = x + iy$. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x,y)$ pairs. Thus, the command

```
dis (100.0,100.0)
```

reports the discharge at the coordinate $(100,100)$.

```
DIS (100.0,100.0)
```

#### 3.2.3.7 FLO – report the total flow

Directs ModAEM to report the total flow between two specified points. SOL must be called before FLO. Note: a solution must be present prior to using this command (see th SOL command in Section 3.2.3.2). The command:

```
flo [z1] [z2]
```

reports the gradient at the complex coordinate $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x,y)$ pairs. Thus, the command

```
flo (50.0,50.0) (100.0,100.0)
```

reports the discharge between the coordinates $(50,50)$ and $(100,100)$.

#### 3.2.3.8 GRI – Enter the grid generation module

The GRI command enters the submodule GRI, which provides for the creation of 2–D SURFER$^{TM}$– or MATLAB$^{TM}$–compatible grid files. See Section 3.2.4 for details. Usage:

```
GRI
```

### 3.2.4 Module GRI – Make a SURFER$^{\text{TM}}$– or MATLAB$^{\text{TM}}$–compatible grid

This command instructs ModAEM to enter the grid module. Within the grid module, grids of various results values from the model are created. The GRI command must have a matching END command. Within the grid module, the window to be gridded must be specified, along with the dimension of the grid. The following commands occur within the grid module.

#### 3.2.4.1 OPT – specifiy SURFER$^{\text{TM}}$– or MATLAB$^{\text{TM}}$–compatible grid

This command instructs the grid module which type of grid to create. Usage:

```
OPT grid-type
```

**grid-type** Choose ’surfer’ for SURFER$^{\text{TM}}$–compatible grids and ’matlab’ for MATLAB$^{\text{TM}}$–compatible grids. SURFER$^{\text{TM}}$–compatible output files will be named with the ’.GRD’ extension while for MATLAB$^{\text{TM}}$–compatible output files will be named with the ’.m’ extension. If the OPT command is omitted, the grid-type will default to ’surfer’.

#### 3.2.4.2 WIN – Define the grid window

Defines the window to be gridded. The command :

```
win [z1] [z2]
```

sets the lower-left and upper-right corners of the window to be gridded to the complex coordinates $z_1 = x_1 + iy_1$ and $z_2 = x_2 + iy_2$.Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs. Thus, the command

```
win (-100.0,-100.0) (100.0,100.0)
```

sets the lower-left and upper-right corners of the window for the GRI module at the coordinates $(-100, -100)$ and $(100, 100)$, respectively.

#### 3.2.4.3 DIM – Specify number of grid points

Sets the number of grid points along the long axis of the window. The command

```
dim d
```

sets the number of grid points along the long axis to the value of d, where d is a positive integer.

#### 3.2.4.4 HEA – Create a grid of heads

#### 3.2.4.5 POT – Create a grid of potentials

#### 3.2.4.6 PSI – Create a grid of stream functions

#### 3.2.4.7 Q_X – Create a grid of discharges

#### 3.2.4.8 Q_Y – Create a grid of discharges

Sample grid module commands to create SURFER$^{TM}$–compatible grids of heads, potentials, stream functions, and discharges:

```
GRI
   WIN (-100.0,-100.0) (100.0,100.0)
   DIM 50
   HEA modaem
   POT modaem
   PSI modaem
   Q_X modaem
   Q_Y modaem
END
```

The parameters passed to the HEA, POT, PSI, Q_X, and Q_Y commands specify the name of the output file for the grids.

### 3.2.5 Module TR0 – Trace

The TR0 command instructs ModAEM to enter the trace module, which is used to trace 2-D streamlines. The TR0 command must have a matching END command. Within the TR0 module, the following commands are valid:

#### 3.2.5.1 WIN – Set the tracing window.

Default tuning parameters are derived from the window size.

#### 3.2.5.2 TUN – Set tuning parameters

Sets tuning parameters for the tracing algorithm. Usage:

```
TUN step prox frac small
```

**step** The base step size

**prox** The proximity (in terms of the current step size) to boundary conditions for reducing the step size

**frac** The factor for step size reductions

**small** Smallest allowable step size

#### 3.2.5.3 TIM – Specify maximum time allowed for particle tracing

#### 3.2.5.4 POI – Release a single particle at the specified location

#### 3.2.5.5 LIN – Release particles along a line

N particles along a line

#### 3.2.5.6 GRI – Release a grid of particles in the sub-window

#### 3.2.5.7 WL0 – Release N particles in reverse from the well bore of a WL0 element.

### 3.2.6 Module AEM – Input a problem definition

The AEM module corresponds to the AEM_DOMAIN object within ModAEM. An AEM_DOMAIN object contains all of the information associated with a 2–D analytic element model. As a result, all of the AEM module commands are associated with the specification of the model elements. The AEM module is entered with the AEM command (see Section 3.2.3.1)

#### 3.2.6.1 AQU – Enter the AQU module

The AQU command is used to enter the AQU_Read routine, in order set properties associated with the model domain. A number of commands can be used within the AQU module to describe features of the model domain (see the AQU module discussion in Section 3.2.7).

Usage:

```
aqu domains base thickness conductivity porosity
  ... put aquifer definition commands here ...
end
```

The parameters on the AQU command describe the aquifer properties. The AQU command must occur within the AEM module. The AQU command must have a corresponding END command.

Parameters for the AQU command:

**domains** The (integer) number of domains in the aquifer

**base** The base elevation (real) of the aquifer

**thickness** The thickness (real) of the aquifer

**conductivity** The hydraulic conductivity (real) of the aquifer

**porosity** Porosity (real) of the aquifer

Example AQU command (with no inhomogeneities - i.e. 1 domain):

```
AQU 1 0.0 10.0 100.0 0.25
END
```

See section 3.2.7 for a description of valid commands in the AQU module.

#### 3.2.6.2 WL0 – Enter the WL0 (discharge-specified well) module.

This command occurs within the AEM module. It is used to enter the WL0_Read routine in order to set the properties of discharge-specified well elements in the model. The WL0 command must have a matching END command. Non-comment lines between the WL0 command and the END command specify the parameters of the wells in the module. Usage:

```
WL0 wells
```

Parameters for the WL0 command:

**wells** The (integer) number of wells

Sample WL0 command:

```
WL0 10
   ... well info goes here ...
END
```

The input format for WL0 well data is described in section 3.2.15

#### 3.2.6.3 WL1 – Enter the WL1 (Head-specified well) module.

This command occurs within the AEM module. It is used to enter the the WL1_Read routine in order to set the properties of head-specified well elements in the model. The WL1 command must have a matching END command. Non-comment lines between the WL1 command and the END command specify the parameters of the wells in the module. Usage:

```
WL1 wells
```

Parameters for the WL1 command:

**wells** The (integer) number of wells

Example WL1 command:

```
WL1 10
   ... Well info goes here ...
END
```

The input for for WL1 well data is described in section 3.2.10

#### 3.2.6.4 LS0 – Enter the LS0 (discharge-specified line-sink) module

This command occurs within the AEM module. It is used to enter the LS0_Read routine in order to set the properties of discharge-specified line-sink elements in the model. The LS0 command must have a matching END command. Usage:

```
LS0 strings
```

Parameters for the LS0 command
Parameters for the LS0 command:

**strings** The number of strings of line-sinks in the LS0 module.

Sample LS0 command

```
LS0 10
   ... String info goes here ...
END
```

The input format for LS0 data is specified in section 3.2.11

#### 3.2.6.5 LS1 – Enter the LS1 (head-specified line-sink) module

This command occurs within the AEM module. It is used to enter the LS1_Read routine in order to set the properties of head-specified line-sink elements in the model. The LS1 command must have a matching END command. Usage:

```
LS1 strings
```

Parameters for the LS1 command:

**strings** The number of strings of line-sinks

Sample LS1 command:

```
LS1 10
   ... string info goes here ...
END
```

The input format for LS1 data is specified in section 3.2.14.

#### 3.2.6.6 LS2 - Enter the resistance line-sink module

This command occurs within the AEM module. It is used to enter the properties of head-specified line-sink elements with resistance. The LS2 command must have a matching END command. Usage:

```
LS2 strings
```

Parameters for the LS2 command:

**strings** The number of strings of line-sinks

Sample LS1 command:

```
LS2 10
   ... string info goes here ...
END
```

The input format for LS2 Strings is specified in section 3.2.13

#### 3.2.6.7 HB0 - Enter the HB0 (No-flow boundary) module

This command is used to enter the HB0_Read routine in order to set the properties of no-flow boundaries in the model. The HB0 command must have a matching END command. Usage:

```
HB0 strings
```

Parameters for the HB0 command:

**strings** The number of strings of vertices defining the no-flow boundaries

Sample HB0 command

```
HB0 10
   ... String info goes here ...
END
```

The input format for data lines in HB0 module is specified in section 3.2.14.

#### 3.2.6.8 PD0 – Enter the PD0 (Discharge-specified pond) module

This command occurs within the AEM module. It is used to enter the PD0_Read routine in order to set the properties of discharge-specified pond elements in the model. The PD0 command must have a matching END command. Non-comment lines between the PD0 command and the END command specify the parameters of the wells in the module. Usage:

```
PD0 ponds
```

Parameters for the PD0 command:

**ponds** The (integer) number of ponds

Sample PD0 command:

```
PD0 10
   ... pond info goes here ...
END
```

The input format for PD0 pond data is described in section3.2.15

#### 3.2.6.9 AS0 – Enter the AS0 (discharge-specified area sink) module

This command occurs within the AEM module. It is used to enter the AS0_Read routine in order to set the properties of discharge-specified area sink elements in the model. The AS0 command must have a matching END command. Non-comment lines between the AS0 command and the END command specify the vertices which define the boundaries of the discharge-specified area sink. Usage:

```
AS0 Vertices Top-Recharge Bottom-Recharge
```

Parameters for the AS0 command:

**Vertices** The (integer) number of vertices which delineate the boundaries of the area sink

**Top-Recharge** The (real) recharge rate through the top of the area sink.

**Bottom-Recharge** The (real) recharge rate through the bottom of the area sink.

Sample AS0 command:

```
AS0 10 1.0 1.0
   (10,10)
   (20,20)
   ...
END
```

The data lines within the AS0 module take the form:

```
[z]
```

where the location of the vertex is given by $z = x + iy$ with given discharge (real), radius (real), and id (integer).Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs. Thus, the command

```
(10,10)
```

defines a vertex of the boundary of the discharge-specified area sink at $(10, 10)$.

### 3.2.7 AQU Module

#### 3.2.7.1 REF – Reference point command

The REF command occurs within the AQU module. The REF command is optional. It gives the head at a reference point and the (optional) uniform flow for the aquifer. Usage:

```
REF [z] head {[Q]}
```

Parameters for the REF command:

**[z]** the location, $z = x + iy$ of the reference point. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs.

**head** The (real) head at the reference point.

**[Q](optional)** The (complex) uniform flow, $Q = Q_x + iQ_y$ at the reference point. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs.

Sample REF command:

```
REF (0,0) 100.0
```

Establishes a reference point at coordinate $(0, 0)$ with head 100, and gives no uniform flow.

*Vic - you'd better explain the uniform flow argument.*

#### 3.2.7.2 BDY - Boundary command

The BDY command is used to delineate the elements at the boundary of a finite aquifer. It must occur within the AQU module. The BDY command is followed by a string of data records and ends when another command is encountered. Each data line consists of two end-points, a reference head (real), a flux (real) and a logical flag . Sample BDY command:

```
AQU ...
   REF ...
   BDY
      (10,10) (20,20) 100.0
      (20,20) (30,30) 99.5
       ...
END
```

*Vic - here's another one that could use a better explanation than I know how to give.*

#### 3.2.7.3 PRM – Perimeter command

The PRM command is used to delineate the perimeter of a finite aquifer. It occurs within the AQU module. An error results if the perimeter is specified and the user requests any flow conditions outside the perimeter. Usage:

```
PRM default-val
```

Parameters for the PRM command:

**default-val** The default value to be used if the user requests a value (head, potential, recharge, velocity, discharge) at a point outside the perimeter. Use an easily recognizable number (e.g. -9999).

Sample PRM command:

```
AEM ...
   REF ...
   PRM -9999
      (1000,1000)
      (1000,-1000)
      (-1000, -1000)
      (-1000, 1000)
END
```

#### 3.2.7.4 IN0 – Enter the IN0 (inhomogeneity) module

Used to enter the inhomogeneity module. The IN0 command occurs within the AQU module and must have a corresponding END command. This command takes no parameters.

### 3.2.8 IN0 Module

Inhomogeneities in the model domain are delineated in the IN0 module.

#### 3.2.8.1 DOM – Define inhomogeneity.

The DOM command is used to define an inhomogeneity in the aquifer. It must occur within the IN0 module. There should be one DOM command for each inhomogeneity specified in the AQU command. The DOM command is allow only within the IN0 module. Usage:

```
DOM Vertices Base Thickness Conductivity Porosity
```

Parameters for the DOM command:

**Domains** The (integer) number of vertices which delineate the boundaries of the inhomogeneity

**Base** The base elevation (real) of the aquifer

**Thickness** The thickness (real) of the aquifer

**Conductivity** The hydraulic conductivity (real) of the aquifer

**Porosity** Porosity (real) of the aquifer

Sample DOM command:

```
AQU 2 0.0 10.0 100.0 0.25
   IN0
      DOM 10 0.0 10.0 50.0 0.25
         (10,10)
         (20,20)
         ...
   END
END
```

### 3.2.9 WL0 Module

The WL0 module is used to add discharge-specified wells to the analytic element model. See section 3.2.6.2for a description of the WL0 command used to enter the WL0 module. Within the WL0 module, only data lines can be entered. Data lines take the form:

```
[z] discharge radius id
```

defines a discharge-specified well at location $z = x + iy$ with given discharge (real), radius (real), and id (integer).Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs. Thus, the command

```
(50,50) 100.0 0.1 1
```

defines well 1 at coordinate $(50, 50)$ with discharge 100.0 and radius 0.1.

### 3.2.10 WL1 Module

The WL1 module is used to add head-specified wells to the analytic element model. See section 3.2.6.3 for a description of the WL1 command used to enter the WL1 module. Within the WL1 module, only data line can be entered. Data lines take the form:

```
    [z1] radius [z2] head id
```

defines a well at location $z_1 = x_1 + iy_1$ with given radius (real) and with head (real) specified at location $z_2 = x_2 + iy_2$and id (integer). Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs. Thus, the line

```
    (50.0,50.0) 0.1 (100.0,100.0) 100.0 1
```

defines well 1 at coordinate $(50, 50)$ with radius 0.1 and with the head specified at 100 at coordinate $(100, 100)$.

### 3.2.11 LS0 Module

The LS0 module is used to add discharge-specified line sink elements within the AEM module. See section 3.2.6.4 for a description of the LS0 command used to enter the LS0 module. Within the LS0 module, the only command allowed is the STR command, which is used to define a string of line sinks.

#### 3.2.11.1 STR Command – string of discharge-specified line sinks.

The STR command within the LS0 module is used to add strings of discharge-specified line sinks to the model domain. Usage:

```
    STR vertices id
```

parameters for the STR command:

**vertices** the number of vertices (integer) for the string

**id** the string id (integer)

Sample STR command:

```
    LS0 10
       STR 2 1
          (10,10) 100
```

```
        (20,20) 95
     STR 10 2
        ...
     STR...
  END
```

The data records for which occur after each STR command specify the vertices for the line sinks within that string. The data record takes the format:

```
[z] discharge
```

defines a one vertex of a string of discharge-specified line sinks at location $z = x + iy$ with given discharge (real). Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs.

### 3.2.12 LS1 Module

The LS1 module is used to add head-specified line sink elements within the AEM module. See section 3.2.6.5for a description of the LS1 command used to enter the LS1 module. Within the LS1 module, the only command allowed is the STR command, which is used to define a string of line sinks.

#### 3.2.12.1 STR Command – string of head-specified line sinks.

The STR command within the LS1 module is used to add strings of discharge-specified line sinks to the model domain. Usage:

```
STR vertices id
```

parameters for the STR command:

**vertices** the number of vertices (integer) for the string

**id** the string id (integer)

Sample STR command:

```
LS1 10
   STR 2 1
      (10,10) 100
      (20,20) 95
   STR 10 2
      ...
   STR ...
END
```

The data records for which occur after each STR command specify the vertices for the line sinks within that string. The data record takes the format:

```
[z] head
```

defines one vertex of a string of head-specified line sink at location $z = x + iy$ with given head (real). Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs.

### 3.2.13 LS2 Module

The LS2 module is used to add head-specified line sink elements with resistance within the AEM module. See section 3.2.6.6for a description of the LS2 command used to enter the LS2 module. Within the LS2 module, the only command allowed is the STR command, which is used to define a string of line sinks.

#### 3.2.13.1 STR Command – string of head-specified line sinks.

The STR command within the LS1 module is used to add strings of discharge-specified line sinks to the model domain. Usage:

```
STR vertices c w d route-id drain-en route-en id
```

Parameters for the STR command:

**vertices** The maximum number of vertices along the modeled stream reach

**c** The 'resistance' of the line-sink, in units of time (e.g. days). This is the reciprocal of the MODFLOW 'leakance', and is computed as $\frac{Thickness}{K}$where $Thickness$ is the thickness of the resistance layer and $K$ is the vertical hydraulic conductivity? of the resistance layer)

**w** The width of the stream

**d** The 'depth', defined as the distance from the water level in the stream (the specified heads at the vertices) to the bottom of the resistance layer. This is used to determine whether the line-sink is 'percolating', in a manner similar to the MODFLOW RIV package.

**route-id** The ID number of the stream reach below this one in a stream network. Set route-id = 1 if there is no downstream route.

**drain-en** Flag for 'drain' elements. .true. for drains, .false. if not. if the drain-en flag is set (*Vic - does this mean true*?), the line-sink cannot lose water (as in MODFLOW DRN package).

**route-en** Flag for enabling routine decision-making. This will allow line-sinks to be 'turned off' if there is no net flow in the stream reach, according to the routing package. This behavior is analogous to the MODFLOW STR package.

**id** The ID number for the stream reach. Note that this is the tool for linking in the routing package.

### 3.2.14 HB0 Module

The HB0 module is used to no-flow boundary elements within the AEM module. See section 3.2.6.7for a description of the HB0 command used to enter the HB0 module. Within the HB0 module, the only command allowed is the STR command, which is used to define a string of vertices defining a no-flow boundary.

#### 3.2.14.1 STR Command – string of no-flow boundary vertices.

The STR command within the HB0 module is used to add strings of vertices which define no-flow boundaries to the model domain. Usage:

```
STR vertices id
```

parameters for the STR command:

**vertices** the number of vertices (integer) for the string

**id** the string id (integer)

Sample STR command:

```
HB0 10
   STR 2 1
      (10,10)
      (20,20)
   STR 10 2
      ...
   STR ...   decison
END
```

The data records for which occur after each STR command specify the vertices for the no-flow boundary defined by that string. The data record takes the format:

```
[z]
```

defines one vertex at location $z = x + iy$. Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs.

### 3.2.15 PD0 Module

The PD0 module is used to add discharge-specified ponds to the analytic element model. See section 3.2.6.8for a description of the PD0 command used to enter the PD0 module. Within the PD0 module, only data lines can be entered. Data lines take the form:

```
[z] discharge radius id
```

defines a discharge-specified pond with center at location $z = x + iy$ with given discharge (real), radius (real), and id (integer). Note that in Fortran free–format reads, the two parts of the complex coordinate are provided as $(x, y)$ pairs. Thus, the command

```
(50,50) 100.0 100.0 1
```

defines pond 1 at coordinate $(50, 50)$ with discharge 100.0 and radius 100.0.

# Chapter 4

# Output Formats

This chapter will describe ModAEM output files.

# Chapter 5

# Validation

This chapter will describe validation of ModAEM results.