GWM–VI File Input Instructions

GWM–VI (Banta and Ahlfeld, 2013) reads an input file that references a number of other input files containing information that defines the groundwater-management problem. This file is similar to the GWM file required by GWM-2005 and described in detail in Ahlfeld and others (2011). The GWM–VI input file consists of a series of lines, each with a keyword followed by a file name. The GWM–VI input file has the following items:

0. #Text [optional]
1. **OUT** Fname [optional]
2. **DECVAR** Fname
3. **STAVAR** Fname [optional]
4. **OBJFNC** Fname
5. **VARCON** Fname
6. **SUMCON** Fname [optional]

   HEDCON Fname [optional]

   STRMCON Fname [optional]
7. **SOLN** Fname
8. **CONTROL** Fname
9. **NAM** Fname

Keywords are in **bold** font. Fname is a file name or path of an output or input file of the type indicated by the corresponding keyword. Explanation of each item follows:

0. Item 0 is optional and consists of one or more comment lines designated by a “#” character in column 1. Text is printed to the output file.
1. Item 1 is optional. A filename for output from GWM–VI may be specified as Fname following the keyword OUT in this item. If item 1 is omitted, a default name of GWM.OUT, written in the directory in which GWM–VI is invoked, is used.
2. Each management problem must include a file that provides information about the decision variables; the name of this file is specified by Fname following the keyword DECVAR in this item.
3. Item 3 is optional. Each management problem may include a file that provides information about the state variables; the name of this file is specified by Fname following the keyword STAVAR in this item.
4. Each management problem must include a file that provides information about the objective function; the name of this file is specified by Fname following the keyword OBJFNC in this item.
5. Each management problem must include a file that provides information on the lower and upper bounds specified for the flow-rate and external decision variables; the name of this file is specified by Fname following the keyword VARCON in this item.
6. Item 6 is optional. Each management problem may include up to three files that provide information about the types of constraints of the management problem that are allowed in GWM–VI. These files are specified by the records of item 6, which can be listed in any order. The keyword SUMCON identifies a file used to specify linear summation constraints; HEDCON identifies a file used to specify head constraints; and STRMCON identifies a file used to specify streamflow constraints.
7. Each management problem must include a file that provides information about the solution and output-control parameters necessary for a GWM simulation; the name of this file is specified by Fname following the keyword SOLN in this item.
8. Each management problem must include a file of information to define additional output options, to define the way in which GWM–VI will invoke MODFLOW and MMProc, and, optionally, to enable parallel processing; the name of this file is specified by Fname following the keyword CONTROL. Instructions for preparing the CONTROL input are described in the next section.
9. GWM–VI also needs to read the MODFLOW Name file. The Name file is expected to follow the form described in Harbaugh (2005). The name of the Name file is specified by Fname following the keyword **NAM**.

Items 0 to 7 of the GWM–VI input file are common to both GWM-2005 and GWM–VI. Input instructions for these items (files of type **DECVAR**, **STAVAR, OBJFNC**, **VARCON**, **SUMCON**, **HEDCON**, **STRMCON**, and **SOLN)** are described in Ahlfeld and others (2005, 2011) and Ahlfeld and Barlow (2013); input instructions for these files are not provided in this report.

The GWM Process information provided in items 2 through 7 of the GWM–VI input file is closely linked to the output that must be produced by MODFLOW. For example, if streamflow constraints are specified at particular stress periods in the file identified by the STRMCON keyword, then the output from the flow process must include a binary file that contains streamflows at those specified stress periods. A detailed description of the requirements of binary-file output is provided in the “Output Requirements for MODFLOW” section.

A GWM–VI input file can be used by GWM-2005; comparison of results between GWM-2005 and GWM–VI is thereby facilitated for management problems that are based on a MODFLOW-2005 executable. Items 8 and 9 of the GWM–VI input file are required for GWM–VI but not for GWM-2005. When a GWM–VI input file is run with GWM-2005, these two lines will be ignored.

When using GWM–VI, names of flow-rate decision variables, head-based constraints, streamflow-based constraints, and drain-based constraints need to conform to the JUPITER naming convention (Banta and others, 2006) because JUPITER modules are used to implement communication between GWM–VI and JRunnerM. The following two rules comprise the JUPITER naming convention:

Rule 1. The first character needs to be a letter of the English alphabet.

Rule 2. All characters after the first letter need to be a letter, digit, or member of the set: “_”; “.”; “:”; “&”; “#”; and “@” (underscore, dot, colon, ampersand, number sign, and at symbol).

Lengths of names specified in GWM input are not restricted by the JUPITER naming convention because the maximum length permitted for decision-variable and constraint names in the DECVAR, HEDCON, and STRMCON files is less that the length limit imposed by the JUPITER API.

CONTROL File Input Instructions

A file identified by the keyword CONTROL in the GWM–VI input file provides input to define output options, to define the way in which GWM–VI invokes MODFLOW and MMProc, and, optionally, to enable parallel processing. JUPITER API-style input blocks provide information in the CONTROL file. For detailed information concerning the structure and use of input blocks, the reader is referred to Banta and others (2006). The instructions provided in this section, however, are sufficient for constructing the relatively simple input blocks required by GWM–VI. An example CONTROL file, which is used for the Dewater problem described in appendix 1, is provided in figure 2 for reference.

Input blocks have the basic structure:

```
BEGIN Blocklabel [Blockformat]
Blockbody
END Blocklabel
```

Where:

BEGIN is a keyword that defines the first line of an input block.

Blocklabel is one of several keywords recognized by GWM–VI. GWM–VI recognizes the following blocklabels: **Options, Simulation**, **Model_Command_Lines**, **Parallel_Control**, and **Parallel_Runners**.

Blockformat is an optional keyword that defines the format in which the input block is structured. Valid blockformats include **Keywords** and **Table**. If Blockformat is omitted or misspelled, the blockformat defaults to **Keywords**. Any input block can use either of the two blockformats, but for simplicity in describing the CONTROL file, the **Keywords** blockformat will be used in constructing the Options, Simulation, Model_Command_Lines, and Parallel_Control input blocks, and the **Table** blockformat will be used in constructing the Parallel_Runners input block.

Blockbody consists of one or more lines containing data to be used as input. Content and format of the blockbody depend on the specified blocklabel and blockformat.

END is a keyword that indicates the end of the input block. The blocklabel following the **END** keyword needs to match the blocklabel specified in the corresponding **BEGIN** line.

The format of the blockbody is determined by the choice of blockformat on the BEGIN line. When blockformat is **Keywords**, data are provided in the form of “keyword=value” entries, where “keyword” is one of the keywords recognized in an input block identified by a particular blocklabel, and “value” is an integer, floating point number, text string, or Boolean (true or false) value, as appropriate for the keyword preceding the “=” sign. In the keyword=value entry, spaces may be used on either side of the “=” sign as desired, and single or double quotation marks may be used in pairs. If value is a text string containing embedded blanks, quotation marks must enclose either the value or the entire keyword=value entry. For Boolean values, “True,” “T,” “Yes,” and “Y” are synonymous; similarly, “False,” “F,” “No,” and “N” are synonymous. Keywords and Boolean values are case-insensitive. Case is retained in text-string values, but comparison of strings is case-insensitive.

When blockformat is Table, the data in the blockbody are organized in rows and columns. A line of the form:

NROW=nr **NCOL**=nc **COLUMNLABELS**

must follow the BEGIN line, where
nr is the number of rows of data and
nc is the number of columns.

One line of column labels must appear next, where the column labels include the keywords recognized within the input block. The recognized column labels are the same as the keywords that are recognized when the blockformat is specified as **Keywords**. A total of nc column labels are read. Following the line of column labels, GWM–VI reads nr lines of data, where nc values are read from each line. Text strings containing embedded blanks must be enclosed in single or double quotes. Column labels and Boolean values are case-insensitive; case is retained in text strings, and columns may appear in any order. Blockbody may contain extra columns headed by column labels that are not required or recognized in a particular input block; extra columns are ignored.

Note that keywords and column labels are “recognized” in JUPITER input blocks and that input blocks, by design, are allowed to contain keywords or columns that are not recognized and, therefore, unused. This logic precludes identification of spelling errors by GWM–VI. Many input values have appropriate defaults, as noted in the instructions for each input-block type in following sections. If a keyword or column label is misspelled or not present, no error message is issued, and the default value (if applicable) is used. If GWM–VI generates unexpected results, check the spelling of keywords.

GWM–VI reads up to five input blocks from the CONTROL file; three of these are optional. The blocklabels and order of the input blocks conform to the conventions of the JUPITER API. The input blocks are documented in the following sections.

Options Input Block

The Options input block is optional; blockformat **Keywords** is convenient. GWM–VI recognizes two keywords in this input block: **MessageFile** and **Verbose**.

MessageFile: Value is a text string to be used as a filename for an output file. If the MessageFile option is used, output generated by GWM–VI that is not directly related to the groundwater-management problem is directed to the file identified in value rather than the main GWM–VI output file. The output that is affected is generated by parts of the code that are used to implement communication between GWM–VI and either MMProc or JRunnerM. Output generated during reading of the Discretization file also is affected.

Verbose: Value is an integer that controls verbosity of output not directly related to the groundwater-management problem; the default value is 3. When MessageFile is specified, Verbose affects only the output directed to the MessageFile file. Valid values and meanings are:

0. No extraneous output;
1. Write warnings only;
2. Write warnings and notes;
3. Write warnings and notes, and echo selected input (default);
4. Write warnings and notes, and echo all input; and
5. Write warnings, notes, echoed input, and miscellaneous information.

EpsQnet: EpsQnet applies when MNW2 multi-node wells (Konikow and others, 2009) are used as flow-rate decision variables. A problem can arise when MNW2 calculates a simulated withdrawal rate that differs from the rate specified in the MNW2 input file. EpsQnet is an error tolerance used in comparing Qdes (the desired flow from an MNW2 well) with Qnet (the flow from an MNW2 well as simulated). The default value of EpsQnet is 1.0E-5 (dimension: L3/T in the units used in model input). If the absolute value of the scaling expression ((Qnet-Qdes)/Qdes) is greater than EpsQnet, GWM-VI interprets the difference as being indicative of a dry cell. When this happens, GWM-VI tries a new, smaller withdrawal rate for the decision variable and attempts to make another model run. Problems arise because scaled differences greater than the default value of EpsQnet can be produced even when the heads in the cells specified for an MNW2 well are substantially above the cell bottoms. To avoid unnecessary, additional model runs if this situation is encountered, EpsQnet can be specified as a value larger than the default. An appropriate value to use for EpsQnet is problem-dependent; experimentation may be needed to obtain acceptable results. If a larger value of EpsQnet seems to be needed, a possible approach would be to increase EpsQnet from its default value by an order of magnitude at a time and make a trial run of GWM-VI with each change.

Simulation Input Block

The Simulation input block is required because the command that MMProc will use to invoke MODFLOW is read from the Simulation input block; blockformat **Keywords** is convenient. GWM–VI recognizes one keyword in this input block: **SimCommand**.

SimCommand: Value is a text string to be used as an operating-system command that MMProc will use to start a MODFLOW run. The command needs to be valid if issued in the directory where GWM–VI is invoked or, when parallel processing is enabled, in any of the runner directories. This requirement imposes a degree of uniformity in the directory structure used for runner directories, as well as the directory where GWM–VI is invoked.

Model_Command_Lines Input Block

The Model_Command_Lines input block is required; blockformat **Keywords** is convenient. Two keywords are recognized: **Command** and **CommandID**; only **Command** is required. If a **CommandID** entry is used, it must follow the **Command** entry.

Command: Value is a text string to be used as an operating-system command to invoke MMProc. Value may be either an operating-system command that invokes MMProc directly, or it may be the name of a script file, such as an MS-DOS batch file, that invokes MMProc. Command is invoked by GWM–VI when run in serial-processing mode or when making base groundwater-flow simulations in parallel-processing mode. Command is invoked by JRunnerM to make groundwater-flow simulations required to populate the response matrix when GWM–VI is run in parallel-processing mode.

CommandID: Value is a text string used to identify the command in output generated by GWM–VI. CommandID plays no role in the functionality of GWM–VI. If omitted, the command identifier defaults to a blank string.

Parallel_Control Input Block

The Parallel_Control input block is required in order to invoke parallel processing but otherwise is optional; blockformat **Keywords** is convenient. Six keywords are recognized: **Parallel**, **Wait**, **VerboseRunner**, **AutoStopRunners**, **OperatingSystem**, and **TimeoutFactor**.

Parallel: Value is a Boolean value, which, when True, invokes parallel processing. If omitted, **Parallel** defaults to False. When **Parallel** is specified as True, the Parallel_Runners input block is required.

Wait: Value is a floating-point number. **Wait** is a time delay, in seconds, used as needed in checking for presence of, and reading from, the JUPITER signal files (Banta and others, 2006) used to communicate between GWM–VI and JRunnerM. If omitted, **Wait** defaults to 0.001 seconds (sec). In most cases the default value is appropriate; however, in some cases **Wait** may need to be set to a larger value. The need to increase **Wait** can be identified by the appearance of the following message on the screen running either GWM–VI or JRunnerM: “Warning: WAIT time may be too small.” Little is to be gained by setting **Wait** smaller than the default value.

VerboseRunner: Value is an integer that controls verbosity of output generated by JRunnerM. The valid values and meanings are the same as those defined for **Verbose**. If omitted, **VerboseRunner** defaults to 3.

AutoStopRunners: Value is a Boolean value. If **AutoStopRunners** is set to True, at the conclusion of a GWM–VI run, all runners (instances of JRunnerM) will terminate execution. If a sequence of runs of GWM–VI is anticipated and runner directories are located on remote computers in a network, users may find it preferable to allow runners to reset and continue running when GWM–VI execution finishes. To have runners reset rather than terminate, specify **AutoStopRunners** as False. If omitted, **AutoStopRunners** defaults to True.

OperatingSystem: Value is a text string indicating the operating system under which GWM–VI and JRunnerM are run. Valid values are “WINDOWS,” “DOS,” “UNIX,” and “LINUX.” In the context of GWM–VI, WINDOWS and DOS are synonymous, and UNIX and LINUX are synonymous. If omitted or misspelled, **OperatingSystem** defaults to “WINDOWS.”

TimeoutFactor: Value is a floating-point number. The product of **TimeoutFactor** and **RunTime** of the Parallel_Runners input block is used to determine if a model run is overdue. If omitted, **TimeoutFactor** defaults to 3.0.

Parallel_Runners Input Block

The Parallel_Runners input block is required when **Parallel** is set to True in the Parallel_Control input block; blockformat Table is convenient. Three keywords (column labels) are recognized: **RunnerName**, **RunnerDir**, and **RunTime**; **RunnerName** and **RunnerDir** are required. For each processor or processor core involved in a parallel-processing setup, a row in the Parallel_Runners input block defines a name, a runner directory, and, generally, an expected run time for one model simulation.

RunnerName: Value is a text string that is a name used to identify the runner.

RunnerDir: Value is a text string identifying the runner directory where an instance of JRunnerM will run. The string needs to end in the directory-separator character used by the operating system, either “\” or “/”.

RunTime: Value is a floating-point number defining the expected model run time (seconds). If omitted, **RunTime** defaults to 10 sec. The expected run time for each runner is adjusted as the runner completes model runs. When the parallel-processing capability is activated, GWM–VI keeps track of model run times and, if a model run is overdue, it starts the model run on a different runner. The product of the expected run time and **TimeoutFactor** of the Parallel_Control input block is used to determine if a model run is overdue. The expected run time does not need to be particularly accurate. However, to avoid unnecessary restarts due to overdue model runs, set **RunTime** to a value at least as large as the time expected for a model run to complete on each runner.

References

Ahlfeld, D.P., and Barlow, P.M., 2013, Use of multi-node wells in the Groundwater-Management Process of MODFLOW-2005 (GWM-2005): U.S. Geological Survey Techniques and Methods, book 6, chap. A47, 26 p., http://pubs.usgs.gov/tm/06/a47/.

Ahlfeld, D.P., Barlow, P.M., and Baker, K.M., 2011, Documentation for the State Variables Package for the Groundwater-Management Process of MODFLOW-2005 (GWM-2005): U.S. Geological Survey Techniques and Methods 6-A36, 45 p., http://water.usgs.gov/nrp/gwsoftware/mf2005_gwm/MF2005-GWM.html.

Ahlfeld, D.P., Barlow, P.M., and Mulligan, A.E., 2005, GWM--A ground-water management process for the U.S. Geological Survey modular ground-water model (MODFLOW-2000): U.S. Geological Survey Open-File Report 2005-1072, 124 p. Available online at http://water.usgs.gov/nrp/gwsoftware/mf2k-gwm/MF2K-GWM.html.

Banta, E.R., and Ahlfeld, D.P., 2013, GWM-VI--Groundwater management with parallel processing for multiple MODFLOW versions: U.S. Geological Survey Techniques and Methods, book 6, chap. A48, 33 p., http://dx.doi.org/10.3133/tm6a48.

Banta, E.R., Poeter, E.P., Doherty, J.E., and Hill, M.C., 2006, JUPITER: Joint Universal Parameter IdenTification and Evaluation of Reliability--An application programming interface (API) for model analysis: U.S. Geological Survey Techniques and Methods, book 6, chap. E1, 268 p. Available online at http://pubs.usgs.gov/tm/2006/tm6e1/.

Harbaugh, A.W., 2005, MODFLOW-2005, the U.S. Geological Survey modular ground-water model--The Ground-Water Flow Process: U.S. Geological Survey Techniques and Methods, book 6, chap. A16, variously paginated. Available online at http://pubs.usgs.gov/tm/2005/tm6A16/PDF/TM6A16.pdf.

Konikow, L.F., Hornberger, G.Z., Halford, K.J., and Hanson, R.T., 2009, Revised Multi-Node Well (MNW2) package for MODFLOW ground-water flow model: U.S. Geological Survey Techniques and Methods, book 6, chap. A30, 67 p., http://pubs.usgs.gov/tm/tm6a30/.