Grid

Exchange

Method

GEM: A Specification for a Cutting-Edge Technique for Integrating Finite Difference Modeling and Geographic Information Systems using INI-Style Files

Greg Pouch
Kansas Geological Survey

Abstract

Finite difference models are often the method of choice to model fluid flow. Such models require two-dimensional arrays of input parameters for each model cell as well as input data and options that are specific to a particular program. The arrays of input parameters, such as permeability and withdrawal through wells, have geographic meaning and it is desirable to be able to use GISs to manage the information and view the results.

Traditionally, the input arrays were obtained from hand-edited text files, by using a dedicated pre-/post-processor, or by using a set of transfer programs specific to the combination of GIS and model. The Grid Exchange Method (GEM) improves on this situation by providing a common format that can be used by any GIS and by any finite-difference model. This divides the problem into two much more tractable parts: for the GIS or GIS helper-program, reading and writing the arrays from and to the GEM file; for the model or model helper program, reading and writing the arrays from and to the GEM file.

This paper is the specification of the Grid Exchange Method and includes information on how a GEM file is organized and what information it contains and how. A server program, which can be called by other programs through an OLE interface, that provides basic utility functions to read and write various parts of a GEM file under Microsoft Windows is available and is described in a separate paper. A translation program that lets GEM files be used with the Windows-based Hydrogeologic Exploration and Appraisal Toolkit (WHEAT) Electronic Mapping programs is available and is described in a separate paper.

Table of Contents

Overview

Each finite difference model requires input in its own particular format. Preparing the input datasets and reading the output results can require as much effort as the actual modeling of the system. Such a situation significantly impedes productivity and hinders modeling.

Two solutions have been used to solve this problem: dedicated pre-processor and post-processor programs for specific models or model-families, such as Processing Modflow and Visual Modflow; and GIS-Model translator packages such as ArcMod. The pre-processor/post-processor approach has been more common on desktop systems and the GIS-Model link has been more common on mainframe/workstation systems.

The dedicated pre-processor/post-processor packages generally lack the ability to perform the spatial analysis needed to effectively model and area, such as assigning wells to a model cell. The GIS-Model translator packages are generally complicated enough to make them non-portable between systems and difficult to use without extensive training.

The GEM file standard described herein solves this problem by breaking one large problem into a small number of tractable pieces that can be easily modified to take advantage of new models, new model features, and new GISs. It does this by storing model information in an easily-read Windows INI-style file.

GEM uses Windows INI-style files for two reasons: they are easy for humans to read and edit and they are easy for computer programs to read and edit. Although somewhat slower than a traditional dedicated file format or a database, this compromise file type allows more flexibility, extendibility, and portability. The analyst can hand-edit the file and not have to keep track of line and column number—a major task in a file containing only numbers for property values and program options. Computer programs can rapidly extract only the portions of interest to them and not be confused by comment-lines, line labels, and extra information used by the analyst or other programs.

The GEM file is a derivative of the standard MSWindows initialization (INI) file, a labeled, ascii file. An INI file is organized into sections and entries within sections. Sections are enclosed in brackets. Each entry follows the format EntryName=EntryValue. Semi-colons should be interpreted as comment characters.

The following is an example from my Windows.INI file

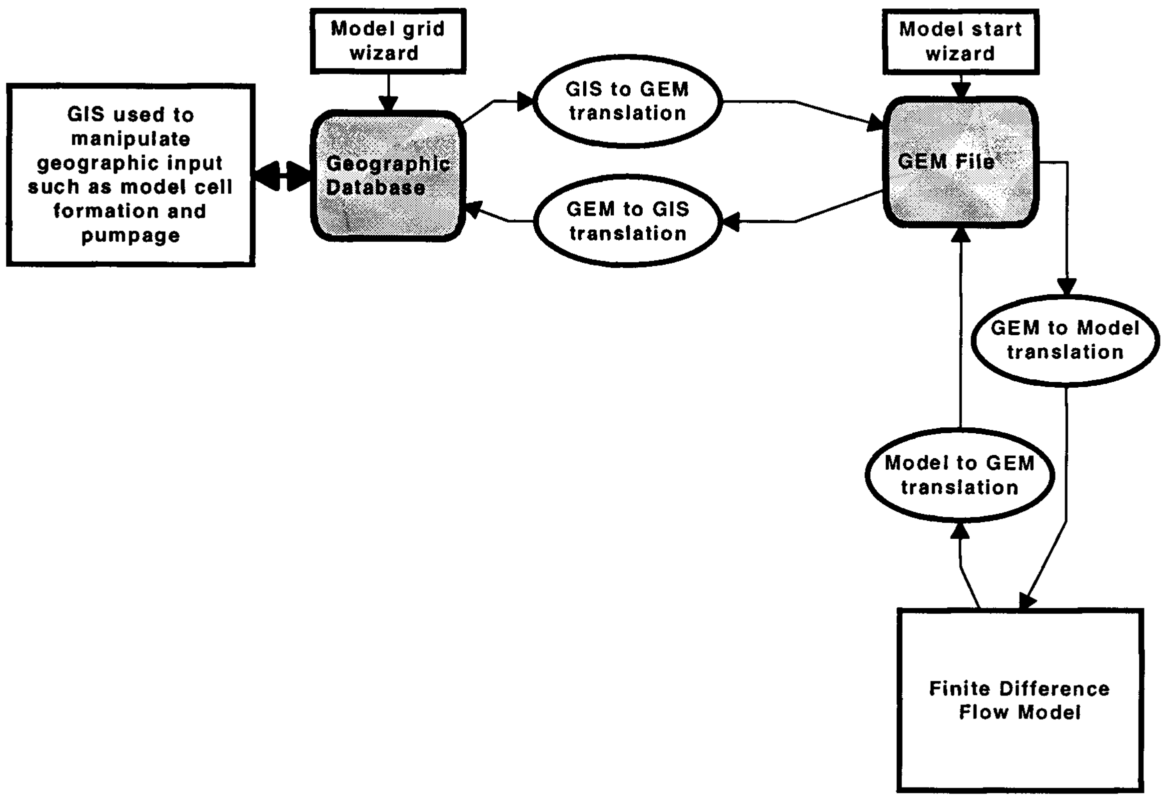
```
[Desktop]
IconTitleFaceName=MS Sans Serif
IconSpacing=75
Pattern=(None)
TileWallPaper=0
GridGranularity=0
WallPaper=(None)
IconTitleStyle=0
IconTitleSize=8

[Extensions]
crd=cardfile.exe ^.crd
trm=terminal.exe ^.trm
txt=notepad.exe ^.txt
ini=notepad.exe ^.ini
pcx=C:\MSOFFICE\CORELDRW\PHOTOPNT\corelpnt.exe ^.pcx
bmp=pbrush.exe ^.bmp
```

This shows two sections [Desktop] and [Extensions]. The first entry in [Desktop]

```
IconTitleFaceName=MS Sans Serif
```

indicates that icons use the MS Sans Serif typeface. The entry name is IconTitleFaceName, and the entry value is MS Sans Serif. Reading an INI file requires knowing the meaning of the sections and the entries. This document provides guidelines for using an INI file to store the data needed for finite difference modeling.



Minimum Requirements of a GEM file

The INI file format can easily accommodate extra information, including user comments and entries not recognized by the programs using it, which are ignored. This section describes the sections that must be present in a GEM file and the organization of a parameter array for use with the GEMEDIT software.

[INTRODUCTION]

Each file must have an [Introduction] section that indicates the number of rows, columns, and layers. Other information about the file/model-run as a whole can be put here, such as author, organization, geographic area name, and purpose. Information specific to a model or program that modifies this file should be kept in its own section or sections. The required entries are

NumberOfColumns

An integer indicating number of columns in the X-direction of the model grid. **Required.**

NumberOfRows

An integer indicating number of rows in Y-direction of the model grid. **Required.**

NumberOfLayers

An integer indicating the number of layers in the Z-direction (vertical). If using a one-layer model, set this entry to 1. **Required.**

Example

```
[Introduction]
NumberOfColumns=10
NumberOfRows=3
CreatedAt=7/12/96 12:44:43 PM
NumberOfLayers=2
```

[STANDARD ARRAY LAYOUT]

Each array needed for running the user's model must be either stored explicitly as an array in standard array layout or derivable from an array in standard array layout. For example, the permeabilities could be explicitly stored as an array, or the user can store geologic units as a text array and provide a look-up tables to generate permeabilities and presumably other properties from the geologic units. Additionally, the model output can be stored in the GEM file as arrays, to allow the GIS to import the output as model-cell attributes for display to the user.

All arrays in a GEM file should be organized as shown in this section. The preferred method is to store each row of data as an entry: this is currently the only format that GEMEDIT can write. In addition to the array values, there are several other required entries that contain information such as Units, VariableName, and StorageStyle.

StorageStyle

A text string indicating whether the values in this array are stored by row, by column, or by cell. **Required**. The value must be one of {RowYYYYYY|ColumnXXXXXX|CellXXXXXX_YYYYYY}. X and Y are digit place-holders for the row (Y) number and the column (X) number. The preferred method is to store each row on a line, which makes it the most readable.

Data Array as Entries

Depending on the value of StorageStyle, the actual array of data will be stored as a set of rows, columns, or each cell as an entry. All data for that row or column is written out as space-delimited values. **Required**.

For row storage, the preferred method, this would look like

```
Row000001= 1  2   3   4   5   6   7   8   9  10
Row000002=11 12  13  14  15  16  17  18  19  20
Row000003=21 22  23  24  25  26  27  28  39  40
```

The same data, in StorageStyle=ColumnXXXXXX format, would look like

```
Col000001=1 11 21
Col000002 =2 12 22
Col000010=10 20 40
Col000009=9 19 39
et cetera
```

The same data, in StorageStyle=CellXXXXXX_YYYYYY format, would look like

```
Col000001_000003=21
Col000001_000002=11
et cetera
```

Currently, GEMEDIT writes data only in the Row format, and this should be the preferred method

DataType

A text string that indicates the kind of data stored in this array. The value must be one of {REAL|INTEGER|TEXT} **Required**. DATE might be added to this list if the need arises.

```
DDataType=REAL
```

ThisIsAnArraySection

Used to help maintain the list of arrays. **Required**.

```
ThisIsAnArraySection=true
```

Units

The Units entry consists of three parts delimited with | symbols: the common name of the unit, such as barrels or acre-feet, a multiplier to convert that into basic SI units, and a text string indicating the basic SI units. **Optional but strongly recommended**.

Units=Feet/Day| 3.52777e-06|m^+1 s^-1 ;3.52777e-06 =.3048 m/(24*3600s)

Units=CubicFeet/Day| 3.2774e-7|m^3 s^-1 ; 3.2774e-7=(.3048 m)^3 /(24*3600s)

Units=Centimeters|0.01|m^+1

For reference purposes, the basic SI units and auxiliary units and the abbreviations that should be used are given in the following table.

Measurement	Unit	Abbreviation
Length	Meters	m
Mass	Kilograms	kg
Time	Seconds	s
Electrical Current	Amperes	A
Thermodynamic Temperature	Kelvins	K
Luminous Intensity	Candelas	cd
Amount of Substance	Moles	mol
Plane Angle	Radians	rad
Solid Angle	Steradians	sr

Just use Dimensionless|1| for simple numbers such as serial number, count of houses, and such.

VariableName

A text string indicating what measurement the array contains, such as Hydraulic Conductivity, Flow Velocity, or Wind Speed (These were chosen to have the same units, but radically different meanings.) **Optional but strongly recommended.**

```
VariableName=Hydraulic Conductivity
```

CommentForUser

This can be used for letting the user leave comments about the array, such as origin or special purpose. **Optional.**

```
CommentForUser=Flow rates based on pumping 40% of capacity
```

MissingValuesShownAs

For cases where there is need to indicate missing values, such as the permeability of a formation which might pinch out, this lets the user indicate what value denotes a missing value. Select a value that can be represented using the data type used for the rest of the array, and that does not occur in the array. You cannot use a blank or series of blanks to indicate missing values. **Optional**

```
MissingValuesShownAs=-9.999e34
MissingValuesShownAs=XXX
MissingValuesShownAs=0
```

RevisionNumber

An integer used to indicate how many times this array has been modified. If you are modifying arrays with source code, you should increment this number each time you modify the array. **Optional.**

LastModfiedTime
A time-date entry to indicate when the last modification occurred. **Optional.**

LastModfiedAuthor
A text to indicate who modified this array. **Optional**

Accuracy
A numeric entry (integer or real) to indicate the analyst's belief in the accuracy of the array. Should be in the same units as the array values. **Optional.**

Examples
```
[ModelCellSerialNumbers]
ThisIsAnArraySection=true
StorageStyle=RowYYYYYY
Units=Dimensionless|1|
Row000001=1  2  3  4  5  6  7  8  9  10
Row000002=11 12  13  14  15  16  17  18  19  20
Row000003=21  22  23  24  25  26  27  28  39  40
DataType=INTEGER
CommentForUser=Do not modify these values.  They are needed by WHEAT.

[Permeability Layer 1]
ThisIsAnArraySection=true
StorageStyle=RowYYYYYY
Row000001=1.1  0  0  4.1  5.1  0  0  0  0  0
Row000002=0  0  0  4.2  0  0  7.2  0  0  0
Row000003=1.3  0  0  0  0  0  0  0  0  10.3
DataType=REAL
RevisionNumber=2
LastModfiedTime=7/19/96 4:14:42 PM
```

[LIST OF ARRAYS]
Each GEM file should have a section called [List of Arrays] that acts as an index to the arrays sections. Each EntryName must be distinct. Programs that modify GEM files should be responsible for maintaining this section.

Additional Sections
The GEM file specification in the preceding section dealt with those sections that are completely general that all GEM files need. The use of sections specific to a particular model will be dealt with in the next section. This section deals with the organization of optional, auxiliary sections of general interest.

PROPERTY MAPPINGS
One of the problems with most models is that they require the user to enter numeric property values for each grid cell, and the user would much rather enter other proxy information such as geologic formation or facies and let the computer convert it to permeabilities, porosities, and so forth. The GEM file format makes it easy to design a property mapping to implement that scheme. (Of course, the model-spawning software

needs to be able to read the property mapping and use it when writing grids to the model input file.)

The key is to define a look-up table as a section. For a deterministic property mapping, where there is a one-to-one correspondence from formation to properties, a property mapping table section could look like

```
[Permeability from Formation]
Units=Darcy|9.870|m^-2
Variable=Permeability
Members=Kgh Kdk Kch
Kgh=1.1e-9
Kdk=2.3e-2
ThisIsAPropertyMapping=true
DefaultValue=-1e34
DataType=REAL
```

For a stochastic model, where a value will be generated from a distribution of possible values at model run-time, the property mapping table would need to include all the distribution parameters necessary, such as

```
[Permeability from Formation]
Units=Darcy|9.870|m^-2
Variable=Permeability
Members=Kgh Kdk Kch
Kgh=1.0e-9| 1e-10
Kdk=2.3e4| 1.1e4
MemberNotFound=1e34
Organization=Mean|StandardDeviation
Distribution=Gaussian
ThisIsAPropertyMapping=true
DefaultValue=-1e34
DataType=REAL
```

Each property mapping table in a GEM file should be organized as shown in this section. Each formation/facies is stored on a separate line. Each property mapping table should describe only one property. In addition to the formation-property correspondence, there are several other required entries that contain information such as a list of the formations/facies and the variable being described.

A look-up table can also provide legend information to the user, such as `Kgh=Greenhorn Shale` or even longer descriptive information.

The required and recommended entries in a property mapping section are

ThisIsAPropertyMapping

Used to help maintain the list of look-up tables. **Required.**

```
ThisIsAPropertyMapping=true
```

DataType

A text string that indicates the kind of data stored in this array. The value must be one of {REAL|INTEGER|TEXT} **Required.**

```
DDataType=Real
```

Members

A space delimited list of the formations/facies/members found in the section. This is used to load the other entries in the section containing property values. **Required.**

```
Members=Kgh Kdk Kch
```

Member Entries

A property value is needed for each formation encountered in the array to which this property mapping applies, specified in the form MemberCode=PropertyValue. **Required**.

```
Kgh=1.1e-9
```

DefaultValue

If the member-property value for some member is not found in the property mapping, use this value. Use a conspicuous value not encountered elsewhere. **Required.**

```
DefaultValue=-1e34
```

Units

The Units entry consists of three parts delimited with | symbols: the common name of the unit, a multiplier to convert the entries into the basic SI units that follow, and a text string indicating the basic SI units (same style used with arrays). **Required**.

```
Units=Darcy|9.870|m^-2
```

VariableName

A text string indicating what measurement the look-up table contains, such Hydraulic Conductivity, Flow Velocity, or Wind Speed **Required**.

```
VariableName=Permeability
```

CommentForUser

This can be used for letting the user leave comments about the look-up table, such as origin or special purpose. **Optional**.

```
CommentForUser=Permeabilities from Watking (1932)
```

RevisionNumber

An integer used to indicate how many times this table has been modified. If you are modifying arrays with source code, you should increment this number each time you modify the array. **Optional**.

[ModelCellSerialNumbers]

The GIS can keep the link between row and column in the model and geographic features in the geographic database in either the geographic database or the GEM file. If it uses the GEM file, it should store the model cell serial numbers as an array in a section called [ModelCellSerialNumbers].

The [ModelCellSerialNumbers] section contains unique serial numbers that lets the GIS link geometric features to the model cells. Each cell must have a unique serial number. Under no circumstances should the user modify entries in the [ModelCellSerialNumbers] section by hand: doing so will completely confuse the GIS, which would then be unable to connect array locations to geometric features.

OTHER SECTIONS

Other sections can be defined by the user/programmer to meet particular needs. The organization should be patterned after those illustrated in the preceding sections and should be made public. Please try to make additional sections as readable as possible, even at the expense of execution speed.

Using GEM Files with a Particular Finite-Difference Model

In order to use a GEM file with a particular finite-difference model, the user needs to supply a program that translates the GEM file into model input and can convert the model output into arrays to place in the GEM file. This may need to get choices about which arrays correspond to which properties and write this information, along with any run-control and other auxiliary information, to an input file or set of files for the model. The additional input data can stored in the GEM file as well, such as a section containing entries for the confined/unconfined status of model layers for MODFLOW.

A separate model-spawning program is needed for each finite-difference model to be used. An additional wizard-like program could perform initial setup of the GEM file and create the sections based on user choices.

A model-spawning program might include such additional features as saving the user's last set of set of choices about which arrays are used for which properties and even allow the user to simply repeat the last run: this last would be particularly useful during the typical modify input – run model – compare results to measurements iterative phase that occupies most of the modeling effort. The model-spawning program should also provide the ability to use a lithology array mapped through a property mapping to produce the needed input such: i.e., the user should be able to specify rock-type for a layer, a mapping from rock-type to permeability, and the model-spawning programming should produce the needed array of permeability.

A model-reading program could be developed that would read the array output from the model and write it to a GEM file, which could then be imported into a GIS, as discussed in the next section.

Using GEM Files with a Particular GIS

In order to use a GEM file with a particular GIS, the user needs to supply a program that transfers data to and from the GIS database. The GIS must allow the user to modify attribute values on a model-cell basis.

In order to use the GIS to develop GEM files, the translator program needs to load the array of data for model cells from the geographic database and write it to the GEM file, and must be able to read an array of values from the GEM file and associate them with geographic features.

If the GIS can use linked tables to store information, it is probably best to organize the data into a series of tables, a main table that contains the model-cells (geometry, serial number, row, and column) and subsidiary tables (containing cell property values and the serial number that can be linked to the main table). This architecture would make it easy to protect the model geography from inadvertent modification and allow multiple copies of the model variables to be present at once. If possible, column names in the GIS database should be the same as the section names in the GEM file.

Sample GEM File

```
[Introduction]
NumberOfColumns=10
NumberOfRows=3
CreatedAt=7/12/96 12:44:43 PM
NumberOfLayers=2

[Permeability Layer 1]
StorageStyle=RowYYYYYY
Row000001=1.1 0 0 4.1 5.1 0 0 8.1 0 0
Row000002=0 0 0 4.2 0 0 7.2 0 0 0
Row000003=1.3 0 0 0 0 6.3 0 0 0 10.3
DDataType=REAL
RevisionNumber=7
LastModfiedTime=7/24/96 4:03:48 PM
ThisIsAnArraySection=TRUE

[Permeability Layer 2]
ThisIsAnArraySection=TRUE
StorageStyle=RowYYYYYY
Row000001=1.1 1.2 0 0 0.009 0 0 0 0 0
Row000002=1.2 1.2E-07 0 4.2 0 0 0 0 9.2 0
Row000003=1.3 0.012 0 0 0 0 0 0 0 10.3
DDataType=REAL
RevisionNumber=4
LastModfiedTime=7/24/96 4:03:55 PM

[ModelCellSerialNumbers]
ThisIsAnArraySection=TRUE
StorageStyle=RowYYYYYY
Units=Dimensionless|1|
Row000001=1 2 3 4 5 6 7 8 9 10
Row000002=11 12 13 14 15 16 17 18 19 20
Row000003=21 22 23 24 25 26 27 28 39 40
DDataType=INTEGER
CommentForUser=Do not modify these values. They are needed by WHEAT.
RevisionNumber=4
LastModfiedTime=7/24/96 4:03:59 PM

[Lithology Layer 1]
ThisIsAnArraySection=TRUE
StorageStyle=RowYYYYYY
Units=Dimensionless|1|
Row000001=Kgh Kgh Kgh To To To Kgh Kgh Kgh To
Row000002=Kgh Kgh Kgh To To Kgh Kgh Kgh Kgh Kgh
Row000003=Kgh Kgh Kgh Kgh To To Kgh Kgh Kgh Kgh
DDataType=TEXT
CommentForUser=Formation codes in layer 1, based on cell center.
RevisionNumber=1
LastModfiedTime=7/24/96 4:08:35 PM

[LithologyToPermeabilityMapping]
ThisIsAPropertyMapping=true
DDataType=REAL
Units=ft/day|3.527777777778e-6|m^+1 s^-1
Members=To Kgh Kgr Kd Kch Kk
To=21
Kgh=0.01
Kgr=0.0001
Kd=5
Kch=2.5
Kk=0.001
DefaultValue=-1e20
VariableName=Hydraulic Conductivity

[List Of Arrays]
Array1=Permeability Layer 1
Array2=Permeability Layer 2
Array3=ModelCellSerialNumbers
Array4=Lithology Layer 1
```

Kansas Geological Survey
Open-file Report

Disclaimer