FDM Groundwater Simulation System

1. Introduction

These lecture notes describe roughly the development steps for an academic software system to simulate groundwater flow using the Finite Difference Method. The theoretical background of the groundwater flow and finite difference method is described in the related lecture notes. It is assumed that the content is known. The development of such system is a process to create a computer model for groundwater flow. This modelling process includes four main parts in some iterating cycles for software extensions and adaptations:

- analysis
- design
- implementation
- test application

Target of the analysis is the specification of the prerequisites for the model application and the definition of the core units and main processes within the software system. In these lecture notes this will be done for 1D and 2D problems, considering steady and unsteady flow. The design contains the complete definition of the model type in UML. It starts with a coarse specification of the core model type elements and their relationships and is done in an iterative process of refinement up to a complete definition of all necessary details. The implementation transforms the design into software, including suitable model editors. The implementation is done in Java, using some standard packages for numerics, user interfaces and data management. The test application of the implemented software checks the correctness of the model and the utilizability of the system from the engineering point of view.

2. Analysis

The first step of the analysis is to identify the main components of the planned software system. Typical and suitable components are:

- model processor (numerical simulation engine)
- user interface
- data base / data storage
- project manager
- model reporter (document generator)
- model analysis (pre- and post-processing)

All six components will be handled separately within the lecture notes.

2.1 Model Processor (Numerical Engine)

The analysis for the model processor is based on the applied numerical simulation method. In these lecture notes the finite difference method is used. The model processor is structured towards an optimized handling of the numerical code. This includes a compromise between flexibility, adaptation options and re-usabilities of model parts of the one side and computer performance on the other side.

First step is the identification of the main units of the model processor derived from the finite difference method. The methods subdivided the model domain in sections using nodes. This leads to two entities:

- Model
- Node

The model entity covers all properties of the model domain and related groundwater flow. The nodes are used to describe the model geometry and to store the discrete function values for the physical state variables related to groundwater flow.

The topology of the nodes can be handled implicitly in 1D by a sequence/array of the nodes in order of their location/coordinate or in 2D by an uniform, rectangular grid of the nodes. Otherwise a Grid entity is introduced to handle the topological relationships of the nodes and other topological/geometrical objects.

- Grid

The physical description is done by a differential equation within the model domain and two options of equations (given head or given flux) at the boundaries. In 1D there are typically two boundaries: one at the left hand side node, one at the right hand side node. In 2D the boundary conditions for a rectangular model domain can be described by four side oriented boundary conditions (East, North, West and South) or more generalized by a specific boundary condition for each node located at the boundary of the model domain. The differential equation considers source terms within the model domain. Boundary conditions and source terms are linked in the numerical approximation independent from the dimension of the problem to nodes. The boundary conditions and source terms leads to two entities:

- BoundaryCondition
- SourceTerm

The numerical approximation method leads to a linear equation system as implicit scheme or a vector calculation equation as explicit scheme. The equation system is defined by one matrix and two vectors. This leads to three entities:

- LinearEquationSystem
- Matrix
- Vector

This kind of problem analysis is in principle the same for 1D and 2D, steady and unsteady groundwater flow. The specific properties of 1D or 2D as well as of steady and unsteady groundwater flow will be considered in the system design.

3. Design

The system design covers two main activities: the specification of the relationship between the system entities and the specification of the attributes and methods for the system entities. These lecture notes are using UML notification for these tasks. The entities will be modelled by objects, described and classified by classes.

3.1 Relationships

The problem analysis leads to eight entities described by eight classes. Besides the principle of generalization/specialization the main relationships can be specified by three types of relationship:

- association
- aggregation
- composition

The class Model covers all properties to describe the groundwater flow within the model domain. This can be considered by using aggregation or composition to integrate all other entities in the Model class. EquationSystem, Source Term and BoundaryCondition are specific for the FDM model and will be covered by composition within Model. Their life cycles depend on the life cycle of the Model. For each node of the model domain one source term is used to optimize the numerical processor structure. The number of BoundaryConditions is in 1D 2, in 2D the number of nodes at the model domain boundary.

Nodes might by used in other modelling environments as geometrical descriptors or to exchange physical state variables with other simulation models. They are considered by aggregation within the Model. Two (four) nodes are the minimum in 1D (2D) to describe a model, the total number of nodes is N. The topology of the nodes can be described implicitly in the Model class or explicitly in one Grid class. The Grid is considered by composition in the Model. The Grid defines the topological relationships between the Nodes by associations and maybe additional entities depending on the Grid type.

A LinearEquationSystem consists of one Matrix and two Vectors. These three objects are set up to solve the equation system. Matrix and Vector are both part of LinearEquationSystem using the principle of composition.

SourceTerm and BoundaryCondition have to be located within the model domain. This can be done by associations to Nodes: each SourceTerm and each BoundaryConditions have one association to one (or more) Node to describe the location.

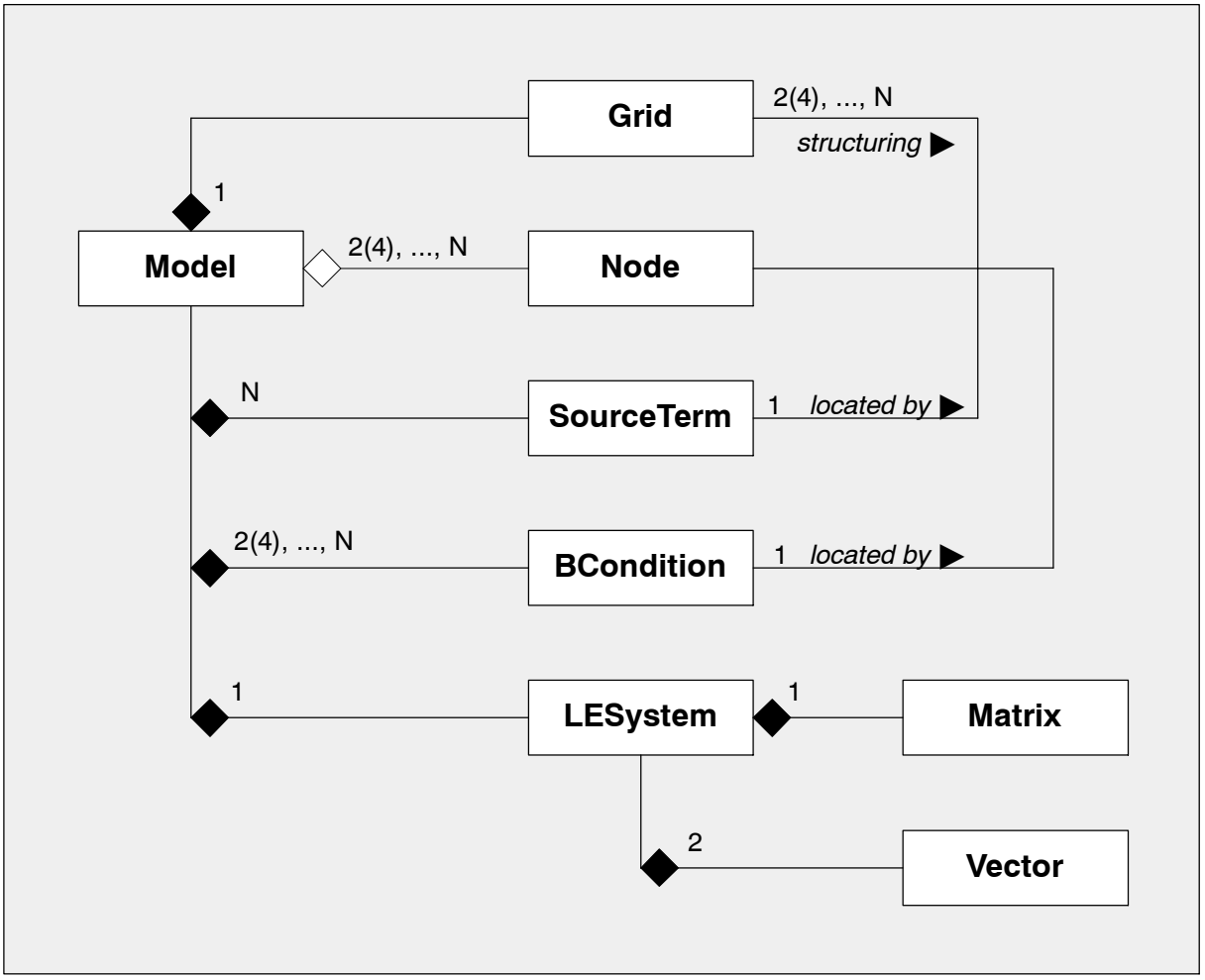

Figure 1: Model Processor FDM: Relationships

3.2 Properties FDM 1D Steady Flow

For each class of the model design the main properties (attributes, methods and operators) are specified. Attributes define the data structure of the classes. Methods and operators specify the functional structure. As methods are used to describe the internal behaviour, operators are used to get access from outside. To prepare the implementation some additional feature might be added to the specification. Typical Get/Set operators for attributes are implicitly specified. All classes includes an identifier as attribute for all related objects

FDM1DModel

float	global hydraulic conductivity	
float	length of the model domain	
integer	number of nodes	
Node[]	nodearray	Aggregation
SourceTerm[]	sourcetermarray	Composition
BoundaryCondition[]	boundaryconditionarray	Composition
LinearEquationSystem	equationssytem	Composition
simulateBehaviour()	numerical simulation	

The class FDM1DModel contains the method simulateBehaviour() to run the model processor, defined by the sequence of numerical statements.

```
initialize EquationSystem by number of nodes as dimension and 0. values
Loop on all SourceTerms
     set up the right hand side vector by related source term values
Loop on all BoundaryConditions
     in case of head boundary condition:
          set up head value in right hand site vector
          set 1. on Node related ES Matrix main diagonal element
     in case of flux boundary condition:
          set up flux value in right hand site vector
          set flux related appr. term in Node related ES Matrix row
Loop on all Nodes except those with boundary condition
     set up EquationSystem Matrix values for DE approximation terms
solve EquationSystem
store results in Node attributes
```

FDM1DNode

The node describes the geometry information and the related physical state variables. This can be done by extension of a generalized node class. This specification covers the necessary attributes explicitly for 1D:

float	coordinate x	
float	current head value as physical state	

FDM1DBoundaryCondition

flag	type of boundary condition (head or flux)	
Node	node	Association
float	boundary condition value	

FDM1DSourceTerm

Node	node	Association
float	source/sink term	

EquationSystem

Matrix	system matrix	Composition
Vector	system left hand side vector	Composition
Vector	system right hand side vector	Composition

Matrix

double[][]	matrix elements

Vector

double[]	vector elements

3.3 Properties FDM2DSteady Flow

The classes for FDM 2D steady flow simulation are designed based on the design for the FDM 1D steady flow simulation. Using the principle of generalization/specialization the classes for FDM 2D are sub-classes of the FDM 1D classes. The classes for the equation system, matrix and vector are the same.

FDM2DModel -> FDM1DModel

```
float                   global hydraulic conductivity direction 1
float                   global hydraulic conductivity direction 2
float                   length of the model domain direction 1
float                   length of the model domain direction 2
integer                 number of nodes direction 1
integer                 number of nodes direction 2
```

The class FDM2DModel overrides the method simulateBehaviour() to run the model processor, defined by the sequence of numerical statements.

```
initialize EquationSystem by node number as dimension and 0. for values
Loop on all SourceTerms
     set up the right hand side vector by related source term values
Loop on all BoundaryConditions
     in case of head boundary condition:
          set up head value in right hand site vector
          set 1. on Node related ES Matrix main diagonal element
     in case of flux boundary condition:
          set up flux value in right hand site vector
          set flux related appr. term in Node related ES Matrix row
Loop on all Nodes except except those with boundary condition
     set up EquationSystem Matrix values for DE approximation terms
solve EquationSystem (ES)
store results in Node attributes
```

FDM2DNode -> FDM1DNode

```
float                   coordinate x1
float                   coordinate x2
```

FDM2DBoundaryCondition -> FDM1DBoundaryCondition

The boundary condition in 2D is linked to a node in 2D (FDM2DNode). As the FDM2DNode class is a FDM1DNode sub class, the FDM2DBoundary condition can use the FDM1DBoundaryCondition attribute FDM1DNode as association to location node.

FDM2DSourceTerm -> FDM 1D BoundaryCondition

As for the boundary condition the FDM2DSourceTerm class can use the attribute FDM1DNode attribute of the super class to specify the location.

3.4 Properties FDM 2D Unsteady

The unsteady FDM 2D groundwater flow simulation extends the steady FDM 2D groundwater flow simulation. Main extension is the time coordinate for the simulation and related time depending physical state variable head in the model domain as well as time series for given values at the boundary conditions and source terms.

FDM2DUnsteady Model -> FDM2DModel

```
long                    start time in milliseconds
long                    time increment in milliseconds
long                    current time in milliseconds
integer                 number of time steps
integer                 current time step
simulateTimeStep()      numerical simulation for one time step
simulateTimeWindow()    numerical simulation for all time steps
Vector                  ht old head value vector
Vector                  ht+1 current new head value vector
```

The class FDM2DUnsteady Model contains the method simulateTimeStep() and simulateTimeWindow() to run the model processor, defined by the sequence of numerical statements.

simulateTimeWindow()

```
call simulateBehaviour() of parent class
     for initial conditions (time step 1)
store results in head vector ht
Loop on all further time steps 2, ... number of time steps
     call simulateTimeStep() for current time step
```

simulateTimeStep()

```
init all SourceTerm values with value for current time step
init all BoundaryCondition values with value for current time step
in case of explicit scheme
     calculate ht+1 vector elements with explicit scheme
in case of implicit scheme
     set up of equation system (similar to simulateBehaviour())
     solve EquationSystem
store results of time step t+1 in head vector ht
store results in Node attributes or external (file, data base)
```

FDM2DUnsteadyBoundaryCondition -> FDM2DBoundaryCondition

TimeSeries	flux/head	time series for bc values

FDM2DUnsteady SourceTerm -> FDM2DSourceTerm

TimeSeries	sink/source	time series for sink/source values

TimeSeries

timearray	time	array of time coordinate values
valuearray	value	array of time related values

3.5 Properties FDM 2D Unsteady Adaptive

The unsteady FDM 2D groundwater flow simulation with adaptive quad-tree based grid extends the unsteady FDM 2D groundwater flow simulation. Main extension is the adaptive grid and related method to the Model entity.

FDM2DUnsteadyAdaptiveModel -> FDM2DUnsteadyModel

```
Grid                    quad-tree based grid quadrilateral structure
```

The model is using a basic regular, structured grid for the determination of the initial conditions and the description of the boundary conditions and source terms. This allows to reuse the attributes and methods of the parent classes FDM2DUnsteadyModel and FDM2DModeldirectly. The adaptive grid entity is the new attribute for the class.

The class FDM2DUnsteadyAdaptiveModel extends the method simulateTimeStep() by the adaptation of the grid structure to the actual physical state and time depending changes.

simulateTimeStep()

```
generate adaptive grid
     reset simulation grid to basic grid
     recursive loop on all grid cells
          check refinement criteria for the grid cell
          in case of refinement: sub-divide grid cell
     set up simulation nodes based on adaptive grid at cell corner
     transfer actual physical state to grid cells (ht values)
call simulateTimeStep() from FDM 2D Unsteady Model
```