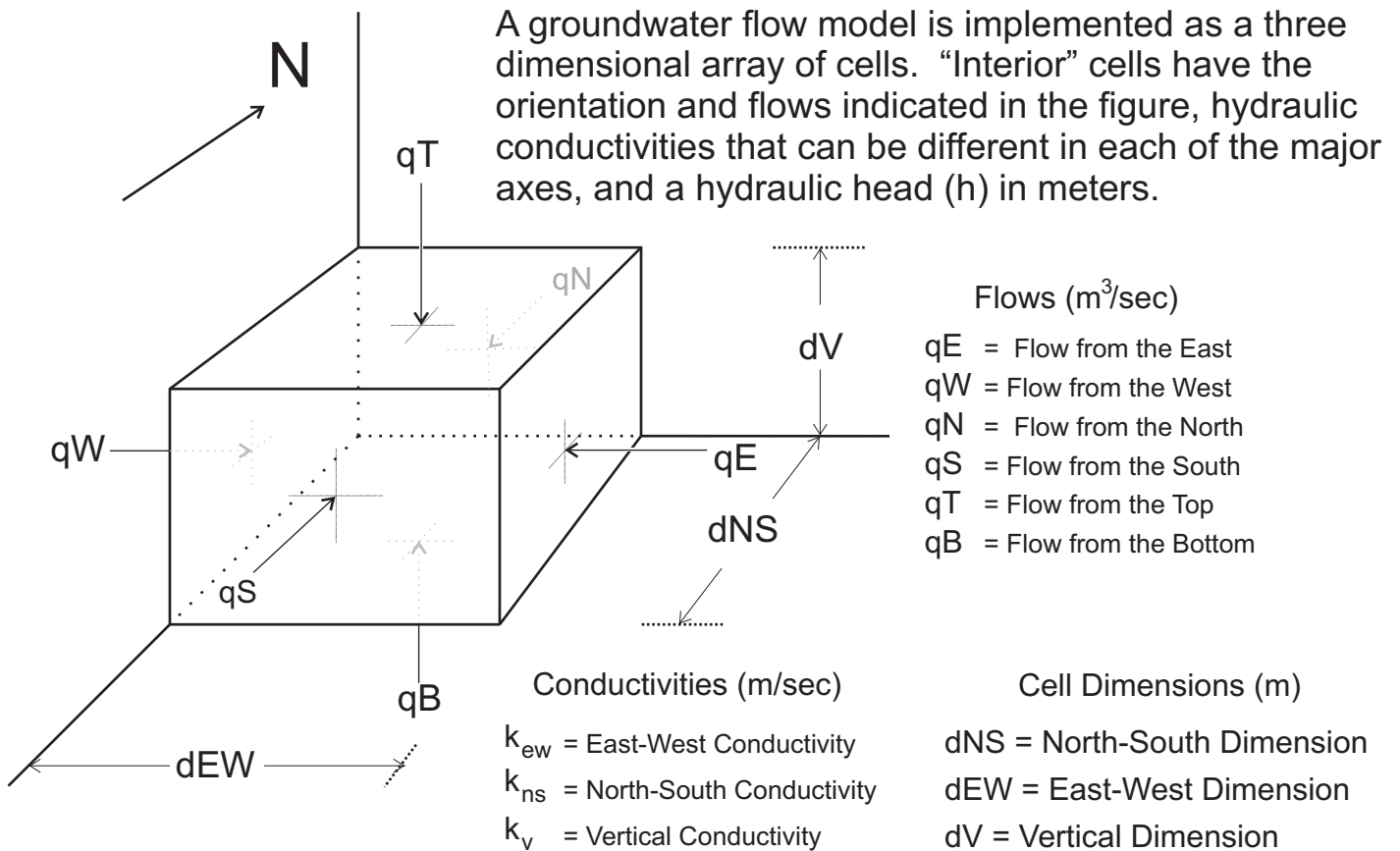


Numerical Flow Modeling using Cellular Automata Techniques

Copyright September 2007
Steve Baker
Umtanum Enterprises



Assuming saturated, non-compressible flow, the mass balance equation is valid:
 $qN + qS + qE + qW + qT + qB = 0$.

For each interior cell with coordinates i, j, k (North-South, East-West, Vertical), Darcy's law applies. If cell i,j,k is surrounded by interior cells then flows can be calculated as follows:

$$\begin{aligned}
 [qN]_{i,j,k} &= 2 * \left([k_{ns}]_{i,j,k}^{-1} + [k_{ns}]_{i-1,j,k}^{-1} \right)^{-1} * \frac{h_{i-1,j,k} - h_{i,j,k}}{dNS} * dEW * dV \\
 [qS]_{i,j,k} &= 2 * \left([k_{ns}]_{i,j,k}^{-1} + [k_{ns}]_{i+1,j,k}^{-1} \right)^{-1} * \frac{h_{i+1,j,k} - h_{i,j,k}}{dNS} * dEW * dV \\
 [qE]_{i,j,k} &= 2 * \left([k_{ew}]_{i,j,k}^{-1} + [k_{ew}]_{i,j+1,k}^{-1} \right)^{-1} * \frac{h_{i,j+1,k} - h_{i,j,k}}{dEW} * dNS * dV \\
 [qW]_{i,j,k} &= 2 * \left([k_{ew}]_{i,j,k}^{-1} + [k_{ew}]_{i,j-1,k}^{-1} \right)^{-1} * \frac{h_{i,j-1,k} - h_{i,j,k}}{dEW} * dNS * dV \\
 [qT]_{i,j,k} &= 2 * \left([k_v]_{i,j,k}^{-1} + [k_v]_{i,j,k-1}^{-1} \right)^{-1} * \frac{h_{i,j,k-1} - h_{i,j,k}}{dV} * dNS * dEW \\
 [qB]_{i,j,k} &= 2 * \left([k_v]_{i,j,k}^{-1} + [k_v]_{i,j,k+1}^{-1} \right)^{-1} * \frac{h_{i,j,k+1} - h_{i,j,k}}{dV} * dNS * dEW
 \end{aligned}$$

The conductivity used in Darcy's law above is weighted to allow the lower conductivity cell to exert more control on groundwater flow than the higher conductivity cell.

Define the following constants to simplify the equations.

$$\begin{aligned} \overline{cN}_{i,j,k} &= 2 * \left(\overline{k_{ns}}_{i,j,k}^{-1} + \overline{k_{ns}}_{i-1,j,k}^{-1} \right)^{-1} * \frac{1}{dNS} * dEW * dV \\ \overline{cS}_{i,j,k} &= 2 * \left(\overline{k_{ns}}_{i,j,k}^{-1} + \overline{k_{ns}}_{i+1,j,k}^{-1} \right)^{-1} * \frac{1}{dNS} * dEW * dV \\ \overline{cE}_{i,j,k} &= 2 * \left(\overline{k_{ew}}_{i,j,k}^{-1} + \overline{k_{ew}}_{i,j+1,k}^{-1} \right)^{-1} * \frac{1}{dEW} * dNS * dV \\ \overline{cW}_{i,j,k} &= 2 * \left(\overline{k_{ew}}_{i,j,k}^{-1} + \overline{k_{ew}}_{i,j-1,k}^{-1} \right)^{-1} * \frac{1}{dEW} * dNS * dV \\ \overline{cT}_{i,j,k} &= 2 * \left(\overline{k_v}_{i,j,k}^{-1} + \overline{k_v}_{i,j,k-1}^{-1} \right)^{-1} * \frac{1}{dV} * dNS * dEW \\ \overline{cB}_{i,j,k} &= 2 * \left(\overline{k_v}_{i,j,k}^{-1} + \overline{k_v}_{i,j,k+1}^{-1} \right)^{-1} * \frac{1}{dV} * dNS * dEW \end{aligned}$$

The darcy flow equations can now be written as follows.

$$\begin{aligned} \overline{qN}_{i,j,k} + \overline{cN}_{i,j,k} * h_{i,j,k} &= h_{i-1,j,k} * \overline{cN}_{i,j,k} \\ \overline{qS}_{i,j,k} + \overline{cS}_{i,j,k} * h_{i,j,k} &= h_{i+1,j,k} * \overline{cS}_{i,j,k} \\ \overline{qE}_{i,j,k} + \overline{cE}_{i,j,k} * h_{i,j,k} &= h_{i,j+1,k} * \overline{cE}_{i,j,k} \\ \overline{qW}_{i,j,k} + \overline{cW}_{i,j,k} * h_{i,j,k} &= h_{i,j-1,k} * \overline{cW}_{i,j,k} \\ \overline{qT}_{i,j,k} + \overline{cT}_{i,j,k} * h_{i,j,k} &= h_{i,j,k-1} * \overline{cT}_{i,j,k} \\ \overline{qB}_{i,j,k} + \overline{cB}_{i,j,k} * h_{i,j,k} &= h_{i,j,k+1} * \overline{cB}_{i,j,k} \end{aligned}$$

The mass balance equation can now be expressed:

$$h_{i-1,j,k} * [cN]_{i,j,k} + h_{i+1,j,k} * [cS]_{i,j,k} + h_{i,j,k+1} * [cE]_{i,j,k} + h_{i,j,k-1} * [cW]_{i,j,k} + h_{i,j,k} * [cT]_{i,j,k} + h_{i,j,k} * [cB]_{i,j,k}$$

$$= \left([cN]_{i,j,k} + [cS]_{i,j,k} + [cE]_{i,j,k} + [cW]_{i,j,k} + [cT]_{i,j,k} + [cB]_{i,j,k} \right) * h_{i,j,k}$$

and,

$$h_{i,j,k} = \frac{h_{i-1,j,k} * [cN]_{i,j,k} + h_{i+1,j,k} * [cS]_{i,j,k} + h_{i,j,k+1} * [cE]_{i,j,k} + h_{i,j,k-1} * [cW]_{i,j,k} + h_{i,j,k} * [cT]_{i,j,k} + h_{i,j,k} * [cB]_{i,j,k}}{\left([cN]_{i,j,k} + [cS]_{i,j,k} + [cE]_{i,j,k} + [cW]_{i,j,k} + [cT]_{i,j,k} + [cB]_{i,j,k} \right)}$$

Equation 1

A model has six types of cells:

- not in model,
- constant head boundary,
- constant flow boundary,
- head stressor (pumping well),
- flow stressor (pumping well), and
- interior.

A model calculation requires:

- assigned heads for constant head boundary cells,
- assigned flows for constant flow boundary cells,
- assigned heads for head stressor cells,
- assigned flows for flow stressor cells,
- assigned conductivities for interior cells, and
- an initial estimate of heads for interior cells

Equation 1 can be used to refine the estimate of head in the cell being considered (cell i,j,k). It works when all six adjacent cells are interior cells, but doesn't work if an adjacent cell is one of the other types (by convention, an adjacent cell can not be of type "not in model"). If an adjacent cell has a head (i.e., is of type "constant head boundary" or "head stressor") then the only adjustment necessary to use Equation 1 is to use the cell's conductivity instead of a weighted average of two conductivities. If an adjacent cell doesn't have a head (i.e., is of type "constant flow boundary" or "flow stressor") then the Head property of that cell is set to a value that gives the specified flow. Equation 1 can then be applied. This approach won't work for a flow stressor cell if it is adjacent to more than one saturated cell; the "pseudo" head will be set more than once, and the last one is the one that will be used.

A more general approach is to assign a face "type" for each of the six faces or each saturated cell: "head" or "flow". If the face is a "head" face then the term in Equation 1 applies: i.e., the face constant times the adjacent cell head. If the face is a "flow" face, then the Darcy flow equations allow us to replace the head term with a flow term, i.e., the face constant times the cell head plus the face flow.

The initial heads assigned to the interior cells is a “guess” at what the actual head is. The initial guess can be refined by sequencing through the three dimensional array of cells, applying Equation 1 for each, and replacing the guess with the calculated value. This process can be repeated until a convergence criterion is met. Convergence can be determined by observing the change in head at each iteration, by flow error, or a combination of both.

Each in-model cell has both an elevation and a groundwater flow potential head. If the calculated head is greater than the elevation, then that cell is saturated. Otherwise, it is dry, or a vadose cell. If the top of a model contains constant flow or flow stressor boundary cells (simulating recharge) and there are vadose cells below them, the vadose cells simply pass the recharge down until it reaches the water table (i.e., a saturated cell).

This software architecture can utilize parallel processing hardware architectures. Each cell can be updated independently of the other cells, so they could all be updated at the same time if the calculation for each could be done by a single processor. It is not feasible to have as many processors as cells, but the cell array can be divided into disjoint groups of cells and each group updated in parallel. Using a multiple processor system should decrease execution time in direct proportion to the number of processors available. In order for this to work, each cell must be able to read the data associated with the six cells that are immediately adjacent to it.

The following listing is a declaration of the types used to implement the cell structures. The listing is displayed in Object Pascal.

```

Type
TCellType = (ctNotInModel, ctInModel, ctBoundaryHead, ctHeadStressor,
ctBoundaryFlow, ctFlowStressor, ctInterior);
TInteriorCellType = (itSaturated, itVadose);
TDirection = (dtNorth, dtSouth, dtEast, dtWest, dtTop, dtBottom);
TFaceType = (ftHead, ftFlow);
TPropertyType = (ptNone, ptCell, ptIndexNorthSouth, ptIndexEastWest,
ptIndexVertical, ptTopElevation, ptBottomElevation, ptCellElevation,
ptVolumetricFlow, ptGroundwaterHead, ptConductivityNorthSouth,
ptConductivityEastWest, ptConductivityVertical, ptFlowNorth, ptFlowSouth,
ptFlowEast, ptFlowWest, ptFlowTop, ptFlowBottom, ptCellFlowError);
TConvergenceType = (ctNone, ctFlowError, ctHeadDigits, ctBoth);

TCell = Class
Public
    NorthSouth: Integer;
    EastWest: Integer;
    Vertical: Integer;
    Elevation: Single;
    CellType: TCellType;
    CellColor: Color;
    Constructor Create; Virtual;
    Function GetCellType: String;
    Function GetPropertyValue(pt: TPropertyType): String; Overload;
    Function GetPropertyValue(pt: TPropertyType; fmt: String): String; Overload;
    Procedure SetPropertyValue(pt: TPropertyType; Value: String); Overload;
    Procedure SetPropertyValue(pt: TPropertyType; Value: Single); Overload;
End;

TInModelCell = Class(TCell)
Private
    FHead: Real; //Heads in m
    procedure SetHead(const Value: Real);
Public
    Constructor Create; Override;
    Property Head: Real Read FHead Write SetHead;
End;

TBoundaryHeadCell = Class(TInModelCell)
Public
    Constructor Create; Override;
End;

THeadStressorCell = Class(TBoundaryHeadCell)
Public
    Constructor Create; Override;
End;

```

```

TCustomFace = Class
Public
  Flow: Single;
End;

TBoundaryFlowCell = Class(TInModelCell)
Public
  qBoundary: Single;
  Constructor Create; Override;
End;

TFlowStressorCell = Class(TInModelCell)
private
  FCustomFace: Array Of TCustomFace;
  function GetCustomFace(Index: TDirection): TCustomFace;
  procedure SetCustomFace(Index: TDirection; const Value:
TCustomFace);
Public
  Constructor Create; Override;
  Property Faces[Index: TDirection]: TCustomFace Read GetCustomFace
    Write SetCustomFace;
End;

THydroGeologicUnit = Class
Public
  Index: Integer;
  UnitName: String;
  kNS: Single;
  kEW: Single;
  kV: Single;
  EffectivePorosity: Single;
  UnitColor: Color;
  Constructor Create;
  Function ToString: String; Override;
End;

TFace = Class(TCustomFace)
Public
  Constant: Single;
  FaceType: TFaceType;
End;

```

```

TInteriorCell = Class(TInModelCell)
Private
    NewHead: Real;
    FFace: Array Of TFace;
    Function GetAdjacentCell(Direction: TDirection): TInModelCell;
    Procedure CalculateConstants;
    Procedure GetFaceTypesAndFlows;
    Procedure CalculateNewHead;
    Procedure CalculateError;
    function GetFace(Index: TDirection): TFace;
    procedure SetFace(Index: TDirection; const Value: TFace);
Public
    InteriorCellType: TInteriorCellType;
    HydroGeologicUnit: THydroGeologicUnit;
    Error: Single;
    Constructor Create; Override;
    Procedure Calculate;
    Property Faces[Index: TDirection]: TFace Read GetFace Write
SetFace;
    End;

```

{Copyright (c) 2007 Umtanum Enterprises

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Third Party components must be licensed separately in accordance with the Third Party terms and conditions.

For questions or comments please contact:

Steve Baker
Steve@Umtanum.com

509-946-5863

2128 Hudson Avenue
Richland WA 99354